An Efficient CGM-Based Parallel Algorithm Solving the Matrix Chain Ordering Problem

Jean Frédéric Myoupo, UFR Sciences, University of Picardie Jules Verne, Amiens, France
Vianney Kengne Tchendji, Department of Computer Science, University of Yaounde 1, Yaounde, Cameroon

ABSTRACT

This study focuses on the parallel resolution of the matrix chain ordering problem and the optimal convex polygon triangulation problem on the Coarse grain multicomputer model (CGM for short). There has been intensive work on the parallelization of these dynamic programming problems in PRAM, including the use of systolic arrays, but a BSP/CGM solution is necessary for ease of implementation and portability. Our CGM algorithm is based on Yao’s sequential solution running in $O(n^2)$ time and $O(n^2)$ space. This CGM algorithm uses $p$ processors, each with $O(n/p)$ local memory. It requires at most $O(S/p \times n^2)$ running time with $S$ communication rounds and with $S/p < 1$. Our algorithm performs better than the algorithm proposed in 2012 by Dilson and Marco when $S$ is less than $n/p$. We offer several ways of partitioning the problem to solve and study the impact of each partitioning algorithm performance. A CGM solution exists based on Yao’s algorithm, but the subdivision of tasks is defined according to the BSP cost model. In this paper, we propose a solution based only on the CGM model specifications. Note that $S$ is the number of super-steps of the CGM algorithm.

Keywords: Bulk Synchronous Parallel, Coarse Grain Multicomputer, Dynamic Programming, Parallel Processing, PRAM

1. INTRODUCTION

Dynamic Programming (DP) is a paradigm used to solve optimization problems that is applied to a large number of areas including optimal control, industrial engineering, economics and artificial intelligence (Dehne, Ferreira, Caceres, Song, & Roncato, 2002; Gupta & Tang, 1995). Many practical problems involving a sequence of interrelated decisions can be efficiently solved by DP. The essence of many DP algorithms lies in computing solutions of the smallest sub-problems and sorting the results for later use in computing larger sub-problems. Thus the solution to the original problem is constructed in a bottom-up fashion, from the smallest sub-problems to the largest.

ADP formulation of a problem is expressed as a recursive functional equation whose left-hand side is an expression involving the

DOI: 10.4018/ijghpc.2014040105
maximization (or minimization) of some cost functions (Equation (1)). Guo and Benjamin (1985) have developed a classification of DP schemes according to the form of the functional equations and the nature of the recursion. A DP formulation is monadic if the inherent cost function involves only one recursive term, otherwise it is polyadic. It is serial if the sub-problems can be grouped in levels and the solution to any sub-problem in a certain level can be found using sub-problems that belong only to the levels immediately preceding, otherwise it is non-serial. We are interested in a CGM (coarse grain multicomputer)-based parallel solution for a typical polyadic non-serial dynamic programming problem, such as the optimal string parenthesizing (OSP) problem, the optimal binary search tree (OBST) problem, the optimal convex polygon triangulation (OCPT) problem and all problems that can be modeled by a recurrence equation similar to (1).

2. RELATED WORK

The classical sequential algorithm, or Godbole’s algorithm (Godbole, 1973), for these problems is based on a dynamic programming technique. It requires $O(n^3)$ calculation operations and $O(n^2)$ memory space. By using the monotonicity property of OBST, Knuth (1973) derived an $O(n^2)$ algorithm in the same space. Yao (1982) obtained the same result with the help of the quadrangle inequalities. Eppstein, Galil and Giancarlo (1988) developed an $O(n \log n)$ algorithm using the restrictive assumption of convexity. Whereas the parallelization of the classical version has been extensively studied by the community of parallel processing researchers for the different parallel computing models (Bradford, 1994; Fotso, Kengne, & Myoupo, 2010; Guibas, Kung, & Thompson, 1979; Gupta & Tang, 1995; Karypis & Kumar, 1993; Kengne & Myoupo, 2012; Rytter, 1988), few works have been produced on the parallelization of the Knuth approach (Kechid & Myoupo, 2008a) or the Yao approach (Kechid & Myoupo, 2008b).

In this study, we parallelize the Yao algorithm using the bridging coarse grain, or bulk synchronous parallel/coarse grain multicomputer (BSP/CGM), model (Dehne, Ferreira, Caceres, Song, & Roncato, 2002; Valiant, 1989; Valiant, 1990). CGM seems the best-suited for the design of algorithms that are not overly dependent on an individual architecture. A BSP/CGM machine is a set of $p$ processors with $O(n/p)$ local memory each, connected by an arbitrary interconnection network. ABSP/CGM algorithm consists of alternating local computations with global communication rounds. The model is coarse grained in the sense that the size $O(n/p)$ of each local memory is defined to be considerably larger than $O(1)$, e.g., $n/p \geq p$ or $n/p \geq p^2$. Note that, for determining time complexities, we consider both local computation time and inter-processor communication time in the standard way. To produce an efficient BSP/CGM algorithm, designers commonly maximize speedup and minimize the number of communication rounds (ideally independent from the problem size and optimally constant).

The CGM-based parallel algorithm for MCOP (Kechid and Myoupo, 2008b) requires $O\left(p\right)$ communication rounds and, at most, $O\left(n^3 / p\right)$ time steps on $p$ processors. The one for OSP (Kengne & Myoupo, 2012) runs in $O\left(n^3 / p\right)$ with $f\left(p\right)$ communication rounds, where $f\left(p\right) = \left(2p^2\right)^{\frac{1}{2}}$. The former CGM algorithm for OBST (Kechid & Myoupo, 2008a), based on the Knuth algorithm, requires $O\left(n^2 / p\right)$ time steps and $O\left(p\right)$ communication rounds. A serious drawback of these algorithms is that the loads on the processors are unbalanced and a processor can process and detain up to $O\left(p\right)$ blocks in the worst case. The algorithm used in Dilson and Marco (2012) requires $O\left(n^3 / p^3\right)$ time steps per processors and $O\left(1\right)$ communication rounds. Its main advantage is that the number of communication rounds is ideal. But the algorithm is divided
A Security Prioritized Computational Grid Scheduling Model: An Analysis
[www.igi-global.com/article/security-prioritized-computational-grid-scheduling/3971?camid=4v1a](www.igi-global.com/article/security-prioritized-computational-grid-scheduling/3971?camid=4v1a)

Rough Entropy Clustering Algorithm in Image Segmentation
[www.igi-global.com/chapter/rough-entropy-clustering-algorithm-image/44708?camid=4v1a](www.igi-global.com/chapter/rough-entropy-clustering-algorithm-image/44708?camid=4v1a)