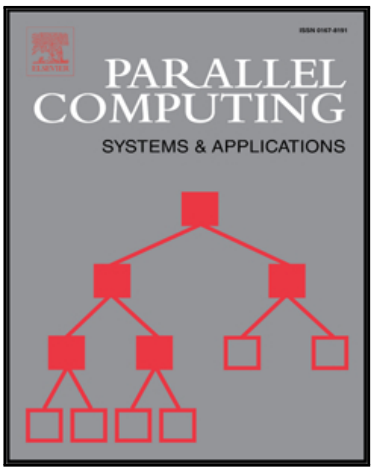

A Matrix-Algebraic Formulation of Distributed-Memory Maximal Cardinality Matching Algorithms in Bipartite Graphs

Ariful Azad, Aydın Buluç

PII: S0167-8191(16)30037-0
DOI: 10.1016/j.parco.2016.05.007
Reference: PARCO 2327

To appear in: *Parallel Computing*

Received date: 29 November 2015
Revised date: 14 April 2016
Accepted date: 15 May 2016

Please cite this article as: Ariful Azad, Aydın Buluç, A Matrix-Algebraic Formulation of Distributed-Memory Maximal Cardinality Matching Algorithms in Bipartite Graphs, *Parallel Computing* (2016), doi: 10.1016/j.parco.2016.05.007

A Matrix-Algebraic Formulation of Distributed-Memory Maximal Cardinality Matching Algorithms in Bipartite Graphs

Ariful Azad, Aydın Buluç

Computational Research Division, Lawrence Berkeley National Laboratory

Abstract

We describe parallel algorithms for computing maximal cardinality matching in a bipartite graph on distributed-memory systems. Unlike traditional algorithms that match one vertex at a time, our algorithms process many unmatched vertices simultaneously using a matrix-algebraic formulation of maximal matching. This generic matrix-algebraic framework is used to develop three efficient maximal matching algorithms with minimal changes. The newly developed algorithms have two benefits over existing graph-based algorithms. First, unlike existing parallel algorithms, cardinality of matching obtained by the new algorithms stays constant with increasing processor counts, which is important for predictable and reproducible performance. Second, relying on bulk-synchronous matrix operations, these algorithms expose a higher degree of parallelism on distributed-memory platforms than existing graph-based algorithms.

We report high-performance implementations of three maximal matching algorithms using hybrid OpenMP-MPI and evaluate the performance of these algorithm using more than 35 real and randomly generated graphs. On real instances, our algorithms achieve up to 200× speedup on 2048 cores of a Cray XC30 supercomputer. Even higher speedups are obtained on larger synthetically generated graphs where our algorithms show good scaling on up to 16,384 cores.

1. Introduction

A matching in a graph is a set of edges without common vertices, and the number of edges in a matching is its cardinality. Computing a maximum cardinality matching (MCM) is an important combinatorial problem in scientific computing with applications to permute a matrix to its block triangular form (BTF) via the Dulmage-Mendelsohn decomposition of bipartite graphs [1, 2], and to compute minimum-weight matchings used by sparse direct solvers [3]. A matching M is *maximal* if any edge not in M is added to M, it is no longer a matching. An algorithm that computes a maximal matching is an approximation algorithm, and the ratio of maximal to maximum cardinality is the approximation ratio of the maximal matching. The primary use case of a maximal cardinality matching is in the initialization of MCM algorithms because the former can be computed much faster than the latter [4, 5, 6, 7]. This paper solely focuses on maximal cardinality matchings in a bipartite graph, $G=(R, C, E)$, where the vertex set V is partitioned into two disjoint sets R and C, such that every edge connects a vertex in R to a vertex in C. Consequently, we will occasionally drop the adjectives "bipartite" and "maximal cardinality" when describing our methods.

Computing a matching in parallel is an interesting research topic on its own. However, our primary interest is in solving sparse systems of linear equations where matching is used as a preprocessing step [2, 3]. The increasing size of the sparse systems encouraged development of many distributed-memory solvers as large-scale problems do not fit into a single node. The lack of distributed-memory matching algorithms and implementations left the preprocessing step as a bottleneck. The current state of the practice [3] involves gathering the data into a single page memory node to run the serial (or multithreaded) matching

Email addresses: azad@lbl.gov (Ariful Azad), abuluc@lbl.gov (Aydın Buluç)