Chapter 5
Paradigms for Effective Parallelization of Inherently Sequential Graph Algorithms on Multi-Core Architectures

Assefaw Gebremedhin
Washington State University, USA

Mostofa Patwary
NVIDIA, USA

Fredrik Manne
University of Bergen, Norway

ABSTRACT

The chapter describes two algorithmic paradigms, dubbed speculation and iteration and approximate update, for parallelizing greedy graph algorithms and vertex ordering algorithms, respectively, on multicore architectures. The common challenge in these two classes of algorithms is that the computations involved are inherently sequential. The efficacy of the paradigms in overcoming this challenge is demonstrated via extensive experimental study on two representative algorithms from each class and two Intel multi-core systems. The algorithms studied are (1) greedy algorithms for distance-k coloring (for k = 1 and k = 2) and (2) algorithms for two degree-based vertex orderings. The experimental results show that the paradigms enable the design of scalable methods that to a large extent preserve the quality of solution obtained by the underlying serial algorithms.

DOI: 10.4018/978-1-7998-7156-9.ch005

INTRODUCTION

Greedy graph algorithms—where an optimization problem defined on a graph is solved by processing vertices (or edges) sequentially one at a time, at each step making the "best local" decision—occur frequently in computations. For some graph problems, Minimum Spanning Tree, for instance, a greedy algorithm is indeed the way to get an optimal solution. For NP-hard graph problems that occur as a part in a larger computation, greedy algorithms are often the methods-of-choice as they provide good approximate solutions at low, often linear, runtime. Further, greedy algorithms naturally fit in the framework of *streaming algorithms* (Alon et al., 1999), where input is fed one item at a time.

In some greedy algorithms iterating over vertices, the *order* in which vertices are processed determines the quality of the solution obtained by the greedy algorithm. One may then need to find, for example, a *degree-based ordering*, where the vertices of a graph are ranked such that the vertex at each position is of maximum or minimum degree in a suitably defined induced subgraph. Degree-based ordering techniques may also be needed in their own right as a stand-alone procedure for an independent objective.

These two inter-related classes of algorithms, greedy algorithms and ordering procedures, have one common feature: the computations involved are *inherently sequential*. Existing parallel algorithm design techniques, such as divide-and-conquer, partitioning, pipelining, pointer-jumping, etc, that are commonly discussed in parallel computing books (Jájá, 1992; Grama et al., 2003; Kurzak et al., 2010) fall short as useful guidelines for effectively parallelizing such algorithms. The parallel algorithm developer's "design toolbox" thus needs to be augmented with new techniques, especially in the present era where parallel computing has established itself in the mainstream.

Contributions of This Chapter

This chapter contributes to this goal by focusing primarily on multi-core and multi-threaded architectures. Specifically, the chapter examines two design paradigms that turn out to be effective for parallelizing inherently sequential algorithms. The first paradigm, dubbed SPECULATION and ITERATION, aims at parallelizing greedy algorithms. The second, named Approximate Update, targets parallelization of ordering algorithms.

The key idea in SPECULATION and ITERATION is to:

maximize concurrency by tentatively tolerating potential inconsistencies and then detecting and resolving eventual inconsistencies later, iteratively.

For this approach to be successful (in leading to scalable methods), inconsistencies need to be relatively rare occurrences. We demonstrate that this is in fact the case for practical problems by applying the paradigm to parallelize greedy algorithms for *distance-k coloring* (for $k = 1$ and $k = 2$). We find, for instance, that the inconsistencies discovered in the very first iteration in the resultant parallel coloring algorithms run on moderate-scale computing environments typically involve less than one percent of the total number of vertices for large, sparse graphs. More generally, the number of inconsistencies will depend on the ratio between the number of vertices and threads and the density of the input graph.

The key idea in the APPROXIMATE UPDATE paradigm is to: