Computers & Operations Research 66 (2016) 81–94

Contents lists available at ScienceDirect

Computers & Operations Research

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

A new exact maximum clique algorithm for large and massive sparse graphs

Pablo San Segundo [a,*], Alvaro Lopez [a], Panos M. Pardalos [b,c]

[a] Center for Automation and Robotics (UPM-CSIC), Jose Gutiérrez Abascal 2, Madrid 28006, Spain
[b] Center for Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA
[c] Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, 603093, Russian Federation

ARTICLE INFO

Available online 10 August 2015

Keywords:
Branch and bound
Bitstring
Massive
Sparse
Maximum clique

ABSTRACT

This paper describes a new very efficient branch-and-bound exact maximum clique algorithm BBMCSP, designed for large and massive sparse graphs which appear frequently in real life problems from different fields.

State-of-the-art exact maximum clique algorithms encode the adjacency matrix in full but when dealing with sparse graphs some form of compression is required. The new algorithm is based on a leading bit-parallel non-sparse solver but employs a novel sparse encoding for the adjacency matrix. Moreover, it also improves on recent optimizations proposed in literature for the sparse case such as core-based bounds.

Reported results show that it is several orders of magnitude better than state-of-the-art. Moreover, a number of real networks with many millions of nodes are solved in a few seconds.

1. Introduction

A simple undirected graph $G=(V,E)$ consists of a finite set of vertices $V=\{v_1,v_2,\ldots,v_n\}$ and edges E made up of pairs of distinct vertices ($E \subseteq V \times V$). Two vertices are said to be adjacent (or neighbors) if they are connected by an edge. For any vertex $v \in V$, $N(v)$(or $N_G(v)$ when the graph is explicitly specified) is the neighbor set of v in G. Any subset of vertices $U \subseteq V$ *induces* a new subgraph $G[U]$ with vertex set $U$and edge-set $G[E] \subseteq E$ such that all its edges have both endpoints in U.

In a complete subgraph, or clique, all vertices are pairwise adjacent. For a given graph, finding a clique of a fixed size k is a well known and deeply studied NP-complete problem referred to as *k-clique* [1]. The corresponding optimization problem is the *maximum clique problem* (MCP), which has the goal of finding the largest possible clique. The size of the maximum clique $\omega(G)$ is known as the *clique number* of the graph.

The *vertex coloring problem* (VCP) is another well known NP-complete problem very much related to the maximum clique problem (MCP); the VCP is well known to be an upper bound of the MCP. The goal of VCP is to find a *proper* color number assignment $c(v): V \to \mathbb{N}$ with the minimum number of colors. A *proper* assignment is such that all pairwise adjacent vertices have different color numbers, i.e. $v_2 \in N(v_1) \Rightarrow c(v_2) \neq c(v_1)$. The notation $C(G)=\{C_1,C_2,\ldots,C_k\}$ in the paper refers to a proper coloring of *size* $|C(G)|=k$ (also known as a *k-coloring*), and color class C_i is the set of vertices with color number i. The size of a solution to the vertex coloring problem is known as the *chromatic number* of the graph $\chi(G)$.

Besides its theoretical value as an NP-hard problem, the maximum clique problem is known to have direct applications in a wide spectrum of fields such as data association problems appearing in bioinformatics and computational biology [2,3], computer vision [4] and robotics [5]. Typically the association problem is reduced to a maximum clique search in a *correspondence graph* which subsumes the matching criteria between two entities.

With the upsurge of web technologies there has also been renewed interest in cliques to capture structure over massive networks. For example, in social networks a clique can identify a group of cooperating agents (e.g. a terrorist cell); in the World Wide Web, cliques or quasi-cliques can help to detect frequently visited pages concerning a certain topic, clique kernels help to identify communities and so on.

In many of the above mentioned applications, networks tend to be *large* and *sparse* because relational models in life frequently involve many elements and show some form locality in their

* Corresponding author.
E-mail address: pablo.sansegundo@upm.es (P. San Segundo).

http://dx.doi.org/10.1016/j.cor.2015.07.013
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

structure. The term *large* is employed rather loosely in literature; for example, Stanford's *large* network data set SNAP[1] contains real graphs ranging from 4000 to 96 million vertices. Graphs with millions of vertices are also referred to as *massive* or *monster* in the same informal way. Of interest to this work is the problem of finding a maximum clique in large and massive sparse graphs.

Exact algorithms for maximum cliques are profuse. State-of-the-art are currently branch-and-bound solvers which use approximate coloring to compute bounds for each subproblem, a very active line of the research in the last decade. The theoretical foundation for this is that the inequality $\omega(G) \le \chi(G) \le \Delta G + 1$ – ΔG denotes maximum graph degree – holds for any graph and follows trivially from [6]. Notable examples of this family of solvers in the past are [7,8] and in recent years [9-16]. [17] is a comparative study which claims that bit-parallel BBMC [13,14] performs best for a number of small and middle size graphs from public benchmarks; MCS [12] was the other most successful algorithm reported in that work.

The BBMC kernel uses bitstrings to encode the problem domain in such a way that critical operations, such as subproblem generation and coloring, are reduced to efficient bit masks. BBMC's careful design reduces bit-twiddling (e.g. vertex enumeration in a bit-encoded vertex set) as much as possible, something which is critical for overall efficiency.

Only very recently a number of improvements have appeared connected with more sophisticated approximate coloring techniques: BBMCL [15] improves BBMC using a selective coloring scheme and IncMaxCLQ [16] encodes each colored subproblem to MaxSAT and uses logical inferences to find subsets of vertices assigned k colors which cannot possibly make part of a *k-clique* (referred to as *inconsistent subsets*). In a similar vein is a variant of BBMC denoted as *infra-chromatic* which searches for a simpler form of inconsistent subsets without resorting to logical reasoning [18]. We note, however, that these more sophisticated techniques *are not useful for massive sparse graphs because it is known that the extra effort spent in pruning is not worthwhile in structures with low density* [14].

Many approximate methods for finding maximum cliques in massive graphs have been proposed in literature, as in [19-21]. Exact methods, however, are much less common as the problem quickly becomes intractable as the size of graphs increases. Here, the main line of research has focused on exploiting multi-core parallelism using a state-of-the-art single-core algorithm, such as BBMC in [22], for individual subproblems. Each subproblem is the workload of a different task.

Additional to these attempts is research on specific optimization techniques for massive sparse graphs. Particularly relevant is the fact that the majority of existing exact algorithms encode the adjacency matrix in full, which is the wrong approach for these types of networks. Two publicly available algorithms stand out to the best of our knowledge: PMC [23] and FMC [24]. Both use an adjacency list representation of the network and *unroll the first level of the search tree* to enforce early pruning. However, while FMC uses degree-based bounds PMC employs the much stronger notion of *core* as well as a more sophisticated recursive search procedure similar to the one used by BBMC. Moreover, the PMC web-site reports an excellent performance over a large data set of real graphs[2], so it is considered as reference in this paper.

Motivated by the above considerations this paper describes a new *very efficient* bit-parallel algorithm which uses a novel compressed representation of the bit-encoded adjacency matrix used by BBMC. Moreover, it also improves on the ideas described by PMC and adapts them to the new representation. Reported results in the paper show improvements of up to several orders of magnitude over a set of 276 publicly available real graphs.

The remaining part of the paper is structured as follows: Section 2 covers notation and a detailed outline of the reference bit-parallel exact maximum clique algorithm starting point of this research. Section 3 describes the new algorithm. Section 4 reports empirical validation and finally Section 5 presents some conclusions and summarizes contribution.

2. Previous work on exact bit-parallel maximum clique algorithms

This section gives a detailed outline of a state-of-the-art bit-parallel exact maximum clique solver. The description is somewhat extensive but presents the necessary background to understand the contribution of this paper.

2.1. Bit-parallelism

In this paper we take the view of a bitstring B_n as an encoding of a subset S of the set $[n]=\{0, 1, 2, 3, \ldots, n-1\}$ such that its elements map to 1-bits in B_n, i.e. for every $i \in S$, $B_n[i]=1$. For example, the bitstring encoding of subset $S=\{1,3,5,7\}$ of $\{0,1,2,\ldots,7\}$is 01010101. The *natural* data structure to encode B_n is not an array of bits, but an array of *blocks of bits* having the size of the register word w (typically 32 or 64 in today's commercial computers; unless otherwise stated $w=64$ is assumed).

By bit-parallelism we refer to the ability of a CPU to process simultaneously bitwise operations over data in two registers. If the data has bitstring semantics this can be viewed as a parallel computation with a speedup of O(w) with respect to a *classical* encoding.

Ground operations for bitstrings are LSB(B_n), the index of the least significant bit in B_n, MSB(B_n), the index of the most significant bit in B_n and POPCNT(B_n), the total number of elements (1-bits) in B_n. In the case of a set $S \subseteq [64]$, efficient implementations for these operators are available in many processors and accessible to programmers via assembly code or specific higher level libraries. The extension to B_n, $n>64$, requires ad-hoc software solutions with heavy optimization. One such solution, very much related to this paper, is the C++ BITSCAN library used by BBMC [25].

2.2. The bit-parallel framework

An efficient bit-parallel encoding for the exact maximum clique problem was first described in BBMC. The main data structures are the following:

- Input graph $G=(V,E)$, $|V|=n$: an array of n bitstrings B_n, each corresponding to a row of the adjacency matrix A. If B_n^i is the encoding of A_i then each 1-bit mapped to $[n]$ in B_n^i corresponds to a vertex adjacent to i.
- A set of vertices $U \subseteq V$: a bitstring B_n in which each 1-bit is the index of the corresponding vertex in $[n]$. Vertices not in U will always have their corresponding bit set to 0 in B_n.
- An induced graph $G[U]$: a bitstring B_n representing the set of vertices in U, together with A. Vertices not in U will always have their corresponding bit to 0 in B_n.

One important bottleneck operation for bit-parallel maximum clique is vertex-set enumeration. Unfortunately, maximum clique requires enumeration both during branching (the vertices of each subproblem) and coloring (the candidate vertices to enlarge the current color class). Implementation details related

[1] http://snap.stanford.edu/data/.
[2] https://www.cs.purdue.edu/homes/dgleich/codes/maxcliques/mctables.html