UDC 519.1:51-74

K. GRONDZAK, P. KORTIS

University of Zilina, Zilina, Slovakia

MODIFIED MAXIMUM CLIQUE EXACT ALGORITHM

A lot of practical problems can be formulated in terms of graph theory in as wide range of areas as theory of coding, data classification, computer vision and computational biology. Among them, the problem of finding maximum or maximal clique is often used. In this paper a modification of exact maximum clique algorithm is proposed and implemented. Results obtained on set of DIMAC benchmark problems are discussed. The results demonstrate the potential of modified algorithm to reduce computational burden when solving maximum clique problem. Main advantage of this modification is the possibility of usage as either heuristic or exact algorithm.

Keywords: *combinatorial problems, maximum clique algorithm, heuristic algorithm.*

Introduction

Combinatorial problems are still a big challenge for researchers. The NP-completeness of many of them is a source of limitations to the size of the problem, which can be solved. Recently new type of processors, equipped with several cores appears on the market. Clusters of such a computers are combined together to create powerful supercomputers. On such supercomputers, it is possible to expand the size of combinatorial problems. Using such approach, some of the combinatorial problems were solved for reasonable large.

Let us mention two very impressive results. Applegate at al. 0 have announced exact solution of the Symmetric Traveling Salesman Problem (TSP) for more than 10000 cities (respectively, 13,509 and 15,112 cities; instances usa13509, d15112).

To find solution, 48 workstations were involved in the calculation, exploring in parallel a tree of 9539 nodes.

Second story of success is the solution of the Quadratic Assignment Problem (QAP). In 2001 and 2002, successful solution of benchmark instances up to size 32, Nugent 30 (900 variables) and Krarup 32 (1024 variables) was announced by Anstreicher, Brixius, Goux, and Linderoth, respectively 2, 3. The instance Nugent 30 (30 firms to be assigned to 30 sites) needed the exploration of a tree with 11892208412 nodes and a network of 2510 heterogeneous machines with an average number of 700 working machines (Pc, Sun, and SGI Origin2000). These machines were distributed in two national laboratories (Argonne, NCSA), five American universities (Wisconsin, Georgia tech, New Mexico, Colombia, Northwestern), and was connected to the Italian network INFN. The time spent was approximately 1 week.

Other attractive combinatorial problems are the problems of finding maximum clique of graph, or evaluating all maximal cliques of the graph. There are many practical applications associated with these problems in science and engineering.

Problems solved by theory of coding, data classification, computer vision, economics and information retrieval lead to the maximum clique search.

Many problems of the computational biology require enumeration of maximal cliques of certain graph. Let us mention applications in genome mapping, 3-D protein structure alignment, etc.

Maximal clique exact algorithm

Let us consider undirected graph $G = (V, E)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. If we denote $E' \subset E$ a subset of edges set we can define a subgraph as a graph $G' = (E', V')$.

Set V' contains only vertices incident with edges from set E'.

A graph G is complete if and only if $E = V \times V$, e.g. all the vertices are pairwise adjacent. Complete subgraph of graph G is called clique. It is quite obvious, that generally there can be many complete subgraphs of graph G. In fact, any single edge is a trivial clique of graph G. These are usually of no interest. For many purposes, cliques exhibiting some special properties are required.

Maximal clique of graph $G = (V, E)$ is a clique, which cannot be expanded using vertices from set V. In another words, it is a clique, which is not part of larger clique, hence the name maximal. Evaluation of maximal cliques is essential for applications like gene expression analysis or detection of social hierarchies.

Among maximal cliques, there are some constructed by using the largest amount of vertices. These are called maximum cliques. Maximum clique evaluation is applied in theory of coding. To construct a largest binary code consisting of binary words, that can correct a certain number of errors, one has to find maximum clique of specially constructed graph.

To determine maximum clique of some graph G, it is necessary to construct all possible subgraphs of it. For each constructed subgraph we have to check, if it is a clique and record maximal solution. It is NP-complete combinatorial problem.

Simple and elegant exact serial algorithm was published by Carraghan and Pardalos 4. Later, Pardalos et al developed and published parallel version, using Message Passing Interface (MPI) 5. It runs on any of the distributed memory supercomputer architectures. They have tested it on graphs containing up to 500 vertices and 75000 edges. Maximum execution time on four processor cluster was approximately 3000s 5.

The main idea of this algorithm is as follows. Let us order somehow vertices of set V to get an ordered list $L = \{v_1, v_2, \ldots, v_k\}$, where k is number of vertices of graph G. Then we apply following steps:

1. Set actual solution to empty set.
2. Construct all possible subgraphs containing first vertex of list L.
3. Evaluating subgraphs find maximum clique.
4. If maximum clique is larger than actual solution, record new actual solution.
5. Remove first vertex from list L and from set V.
6. If list is not empty, continue with step 2.
7. Actual solution contains maximum clique of graph.

This algorithm is guaranteed to find maximum solution. As was mentioned above, the NP-completeness limits the size of problem, solvable using this algorithm.

Essential for parallelization of this algorithm are the changes of list L.

This algorithm generates a sequence of lists $L_i = \{v_i, v_{i+1}, \ldots v_k\}, i = 1,2,\ldots k$. It is obvious, that the size of list decreases by one vertex each time we construct all subgraphs containing first vertex of the list. Because of it, sets of subgraphs constructed in step 2 of the algorithm are disjunctive.

This property can be used to design simple parallel algorithm. Generating a sequence of lists L_i and distributing it among a set of processors, algorithm is executed in parallel. Comparing sizes of results obtained by different processors, the largest clique is determined. The final computational time of parallel algorithm is shorter when compared to serial algorithm, because of implemented parallelism, but this simple form of parallelism is not sufficient for extremely computational complex problems. The problem is, that the number of operations necessary to find clique strongly depends on the cardinality of lists L_i. The higher is the cardinality $\|L_i\|$ the higher is the computational complexity.

Obtained results

As was mentioned above, the main disadvantage of the exact maximum clique algorithm is the computational complexity. To overcome it, we proposed a reversed order of generating sequence of lists L_i. Algorithm starts with list L_i for $i = k - 1$, e.g. with only two nodes. All possible subgraphs are constructed and maximum clique is determined. Then the list is expanded by one vertex. The procedure is repeated until the list L_1, containing all the vertices is reached.

Advantage of this approach is obvious. We are getting partial solutions sooner than with the original approach. It means it can be used as a simple heuristic method to get some estimation of the maximum clique. If the algorithm is executed for all possible lists L_i, the solution is exact.

To test behavior of proposed algorithm, we have tested it on a set of benchmark problems. This set contained five graphs with number of nodes ranging from 100 to 500 and density approximately 50% 6. The graph density is defined as follows:

$$d = \frac{2\|E\|}{\|V\|^2}.$$

Three different algorithms were used to find maximum clique of those graphs. First, the reference algorithm provided with benchmark data 6 (alg1) was executed. Next the exact algorithm proposed by Carraghan and Pardalos was used (alg2). As a third one, modified exact algorithm (alg3) was executed [6]. For each algorithm, size of maximum clique, number of maximum cliques and execution time were recorded. Results were obtained on a single computer equipped with processor Intel Pentium D 3.00 GHz, 1GB RAM and operating system GNU/Linux.

Because values of maximum clique size and number of solutions agreed for all algorithms, these are presented only once. Fastest is the benchmark algorithm. Following is modified exact algorithm. Slowest is the exact algorithm.

The achieved speedup of alg3 with respect to alg2 is in the fact, that amount of constructed and processed subgraphs significantly differs for these two algorithms (there are no available data for alg1). To demonstrate this behavior, we have counted number of subgraphs

Table 1

Summary of obtained results for three different algorithms

Graph parameters	Graph file	r100.5.b	r200.5.b	r300.5.b	r400.5.b	r500.5.b
	Nodes	100	200	300	400	500
	Edges	2508	10036	22361	40061	62161
	Density	0.5016	0.5018	0.496911	0.500762	0.497288
Maximum clique parameters	Number of cliques	31	16	5	3	41
	Clique size	9	11	12	13	13
alg1	Execution time [s]	0.015	0.08	0.6	3.65	13.96
alg2	Execution time [s]	0.025	0.213	1.64	9.78	37.2
alg3	Execution time [s]	0.023	0.141	1.18	6.15	25.8

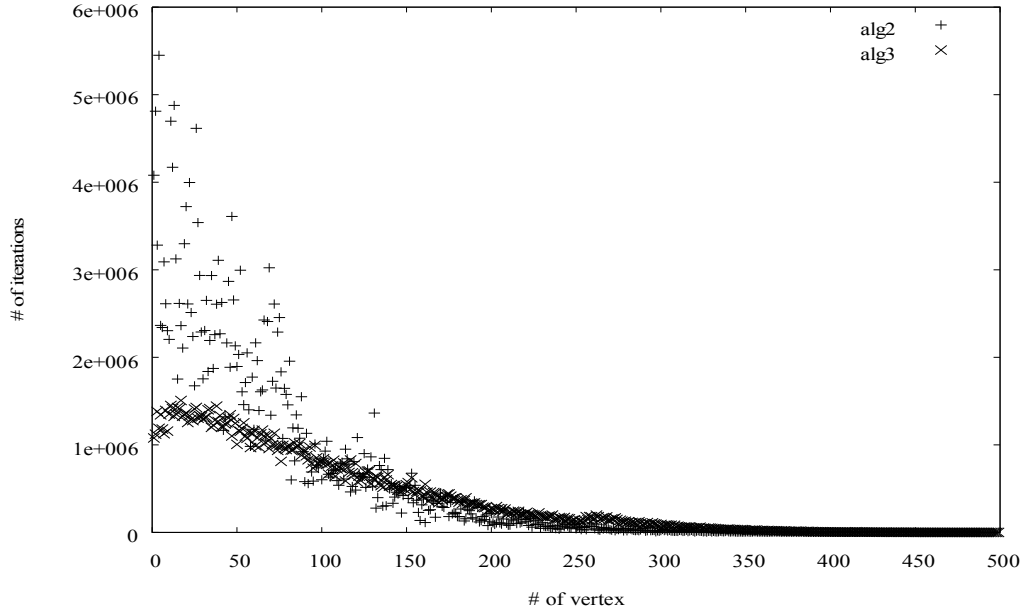

Fig. 1. Number of iterations with respect to the vertex number

constructed and processed during calculation (denoted as iterations) for each list $L_i, i = 1,2,\ldots k$. The lower is the cardinality of list; the lower is the amount of constructed subgraphs (Fig. 1). This behavior is quite natural and expected. But it also demonstrates the significant difference between amount of iterations to process the same list of vertices by alg2 and alg3.

Difference is caused by the fact, that alg3 evaluates list in reversed order.

To achieve better performance, both algorithms apply pruning, e.g. they consider actual maximum clique (AMC) size value and do not consider cliques, which are smaller than this value. Larger number of iterations is needed to process lists with larger cardinality.

Let us consider the evaluation of list L_1. Value of AMC for alg2 is zero and it has to be updated during the evaluation of list L_1. Situation is different for alg3. With high probability the size of AMC is equal to the optimal value (or is very close to it) and it can prune more subgraphs during the iteration process.

Conclusion and Future Work

In this paper, modification of Carraghan and Pardalos exact algorithm was proposed, implemented and tested. We have demonstrated the potential of the modification to reduce computational time on the set of DIMAC benchmark problems. Reasons for achieving speedup were proposed and demonstrated.

This algorithm can be used both as a heuristic algorithm to achieve estimation of the optimal solution, or it can provide optimal results, when run completely.

This algorithm is very suitable for parallelization. To improve the scalability of it, version suitable for GRID architectures is to be developed.

Acknowledgement

This work was partially supported by national VEGA Grant No. 1/0761/08 "Design of Microwave Methods for Materials Nondestructive Testing" and national VEGA Grant No. 1/0796/08 "Large Data Modeling and Processing". Results presented in this paper were obtained when realizing the project "Centre of the translation medicine" in the framework of the Operation program Research and Development sponsored by European Regional Development Fund.

References

1. On the solution of traveling salesman problem / D. Applegate, R.E. Bixby, V. Chvatal, W. Cook // Doc. Math. – 1998. – ICM(III). – P. 645-656.

2. Anstreicher K.M. A new bound for the quadratic assignment problem based on convex quadratic programming / K.M.Anstreicher, N.W. Brixius // Math. Prog. – 2001. – № 89, 3. – P. 341–357

3. Solving large quadratic assignment problems on computational grids / J.P. Goux, K.M. Anstreicher, N.W. Brixius, J. Linderoth// Math. Prog. – 2002. – № 91, 3. – P. 563–588.

4. Carraghan R. An exact algorithm for the maximum clique problem / R. Carraghan, P.M. Pardalos // Operations Research Letters. – 1990. – № 9. – P. 375-382.

5. Pardalos P.M. An exact parallel algorithm for the maximum clique problem/ P.M. Pardalos, J. Rappe, M.G.C. Resende // High performance algorithms and software in nonlinear optimization. – Norwell, Massachusetts, 1997. – P. 279-300.

6. Johnson D.S. Network Flows and Matching: First DIMACS Implementation Challenge/ D.S. Johnson, C.C. McGeoch // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. – Providence, Rhode Island, 1993, Volume 12. – 592 p.

Поступила в редакцию 2.06.2010

Рецензент: д-р техн. наук, проф., профессор, заведующий кафедрой компьютерных систем и сетей В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

МОДИФИКАЦИЯ ТОЧНОГО АЛГОРИТМА ПОИСКА КЛИКИ МАКСИМАЛЬНОГО РАЗМЕРА

К. Гронджак, П. Кортис

Ряд прикладных задач в таких областях знаний как теория кодирования, классификация данных, машинное зрение и биоинформатика формулируется в терминах теории графов. При решении этих задач достаточно часто возникает необходимость поиска максимального числа клик в графе или клики максимального размера. В данной работе предлагается модификация точного алгоритма поиска клики максимального размера. Полученный алгоритм был апробирован на множестве данных DIMAC. Анализ решений для указанного множества показал, что предложенный алгоритм позволяет сократить вычислительную сложность точного алгоритма поиска клики максимального размера в графе. Важным достоинством предложенного алгоритма является то, что он может быть использован как для точного, так и для эвристического решения задачи поиска клики максимального размера в заданном графе.

Ключевые слова: комбинаторика, клика максимального размера, эвристический алгоритм.

МОДИФІКАЦІЯ ТОЧНОГО АЛГОРИТМУ ПОШУКУ КЛІКИ МАКСИМАЛЬНОГО РОЗМІРУ

К. Гронджак, П. Кортис

Ряд практичних задач у таких сферах знань як теорія кодування, класифікація даних, машинний зір і біоінформатика формулюється у термінах теорії графів. При вирішенні цих задач достатньо часто виникає необхідність пошуку максимального числа клік у графі або кліки максимального розміру. У даній статті пропонується модифікація точного алгоритму пошуку кліки максимального розміру. Отриманий алгоритм був апробований на множині даних DIMAC. Аналіз рішень для означеної множини надав можливість визначити, що запропонований алгоритм дозволяє зменшити обчислювальну складність точного алгоритму пошуку кліки максимального розміру у графі. Важливою перевагою запропонованого алгоритму є те, що він може бути використаний як для точного, так і для евристичного рішення задачі пошуку кліки максимального розміру у заданому графі.

Ключові слова: комбінаторика, кліка максимального розміру, евристичний алгоритм.

Grondzak Karol – PhD, assistant professor, University of Zilina, e-mail: Karol.Grondzak@fri.uniza.sk.
Kortis Peter – PhD, assistant professor, University of Zilina, e-mail: Peter.Kortis@fel.uniza.sk.