Proceedings of the World Congress on Engineering 2008 Vol II
WCE 2008, July 2 - 4, 2008, London, U.K.

An Exact Algorithm for the Maximum Weight K_3-free Subgraph Problem

Masayasu Fujiwara * Kazuaki Yamaguchi * Sumio Masuda *

Abstract— **The maximum independent set problem is one of the most famous and well-studied NP-complete problems, and has some important applications. Some exact algorithms based on the branch-and-bound technique have been proposed for the problem. This paper deals with one of its variants, the maximum weight K_3-free subgraph problem. This paper shows an interesting property of a K_3-free graph, an exact algorithm for the problem and its efficiency with some computer experiemnts.**

Keywords: ***K_3-free graph, triangle-free graph, branch-and-bound algorithm, clique, maximum weight independent set***

1 Introduction

A vertex induced subgraph $G' = (V', E')$ of a graph G is called a clique if any pair of vertices in V' are adjacent. A clique with r vertices is denoted by K_r. A graph without K_r is called a K_r-free graph (a K_3-free graph is often called a triangle-free graph). Given a graph $G = (V, E)$ with weight $w(v)$ for each vertex $v \in V$ and an integer $r \geq 2$, the maximum weight K_r-free subgraph problem (MWK_r-free problem for short) is to find a K_r-free subgraph in G whose sum of vertex weights is the maximum.

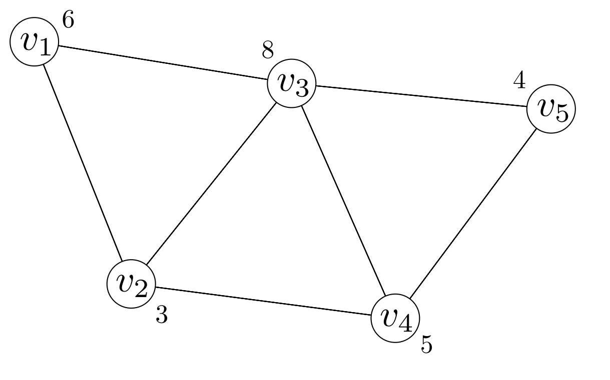

Figure 1: An input for the MWK_3-free problem

Figure 1 shows an example of the input for the MWK_3-free problem, where each number attached to a vertex specifies the weight of the vertex. For simplicity, each solution is denoted with a vertex set. The solution of the problem is a maximal set that does not contain K_3. Thus the candidates of the optimum solution are $\{v_1, v_2, v_4, v_5\}$, $\{v_1, v_3, v_4\}$, $\{v_1, v_3, v_5\}$ and $\{v_2, v_3, v_5\}$. The weights for these sets are 18, 19, 18 and 15, respectively. Therefore, the optimum solution is the subgraph induced by $\{v_1, v_3, v_4\}$.

The MWK_r-free problem for $r = 2$ is merely the maximum weight independent set problem, and is also equivalent to the maximum weight clique problem for the complementary graph of G. Some exact algorithms for these problems and their unweighted versions (namely, the maximum independent set problem and the maximum clique problem) were proposed, and some important applications were shown in Refs.[1]-[8]. But no exact algorithms for MWK_r-free for $r \geq 3$ are known. This is the first paper to propose an exact algorithm for the MWK_3-free problem.

Section 2 shows some definitions and describes two theorems on which our algorithm is based. Section 3 presents our algorithm. Section 4 shows the efficiency of the algorithm with some computer experiments. Finally, Section 5 summaries the results and shows a conjecture which might lead to stronger results.

2 Preliminaries

In this paper, a sequence is denoted with brackets. For two sequences $S = [s_1, s_2, \cdots, s_m]$ and $T = [t_1, t_2, \cdots, t_n]$, a sequence $[s_1, s_2, \cdots, s_m, t_1, t_2, \cdots, t_n]$, which is made by concatenating S and T, is denoted by $S + T$. For $1 < i < m$, $[s_1, s_2, \cdots, s_{i-1}]$ and $[s_i, s_{i+1}, s_{i+2}, \cdots, s_m]$ are denoted by $\sigma^-(S, s_i)$ and $\sigma^+(S, s_i)$, respectively.

Our algorithm is based on the following two theorems.

Theorem 1 *For a K_3-free graph $G = (V, E)$ and a sequence S in which each element of V appears exactly once, there exist two subsequences S_1 and S_2 of S which satisfy property A shown below.*

Property A:

- *Each element in V appears exactly once in either S_1 or S_2.*

*Graduate School of Engineering, Kobe University, 1-1 Rokkodai, Nada, Kobe 657–8501 JAPAN. Email: {ky, masuda}@kobe-u.ac.jp, Received on March 6, 2008.

- *Any consecutive two elements in S_1 or S_2 are not adjacent in G.*

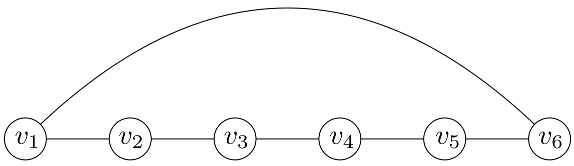

Figure 2: A K_3-free graph G

Before proving the theorem, we show an example. For a K_3-free graph G in Figure 2 and a sequence $[v_1, v_2, v_3, v_4, v_5, v_6]$, two sequences $[v_1, v_3, v_5]$ and $[v_2, v_4, v_6]$ satisfy property A. For G and another sequence $[v_1, v_5, v_2, v_4, v_3, v_6]$, two sequences $[v_1, v_5, v_2, v_4]$ and $[v_3, v_6]$ satisfy property A. The theorem guarantees that two sequences satisfying property A always exsist for any G and S.

Proof of Theorem 1

This theorem is proved by contradiction. Let a K_3-free graph G and a sequence $S = [v_1, v_2, \cdots, v_n]$ be the counter example with the minimum number of vertices.

Let G' be a graph obtained by removing v_n from G. From the definitions of G and S, G' and $\sigma^-(S, v_n)$ has two sequences S_1 and S_2 satisfying property A. Without loss of generality, we assume that v_{n-1} is the last element in S_1.

Let v_{n-k} $(k > 1)$ be the last element in S_2. Clearly, v_{n-k+1}, v_{n-k+2}, $\cdots$, v_{n-1} are contained in S_1 in this order. We choose k to be the maximum number to satisfy property A for G' and $\sigma^-(S, v_n)$.

If v_{n-k} is not adjacent to v_n, two sequences S_1 and $S_2 + [v_n]$ satisfy property A for G and S, which contradicts to the assumption that G and S constitute a counter example. Therefore, v_{n-k} is adjacent to v_n. For a similar reason, v_{n-1} is adjacent to v_n. v_{n-k} and v_{n-1} are not adjacent because G is K_3-free. See Figure 3(a).

Suppose that v_{n-i} is not adjacent to v_{n-k} for some $i \geq 1$. If v_{n-i-1} is not adjacent to v_n, the sequences $\sigma^-(S_1, v_{n-i}) + [v_n]$ and $S_2 + \sigma^+(S_1, v_{n-i})$ satisfy the property A for G and S, which contradicts to the assumption that G and S constitute a counter example. Hence v_{n-i-1} is adjacent to v_n. Because v_{n-k} is adjacent to v_n and G is K_3-free, v_{n-k} is not adjacent to v_{n-i-1}. By mathematical induction, v_{n-i} is adjacent to v_n and not adjacent to v_{n-k} for $1 \leq i \leq k-1$. See Figure 3(b).

If v_{n-k+1} is the first element in S_1, two sequences $S_2 + S_1$ and $[v_n]$ satisfy property A for G and S, which contradicts to the assumption that G and S constitute a counter example. Thus, v_{n-k+1} is not the first element in S_1. But

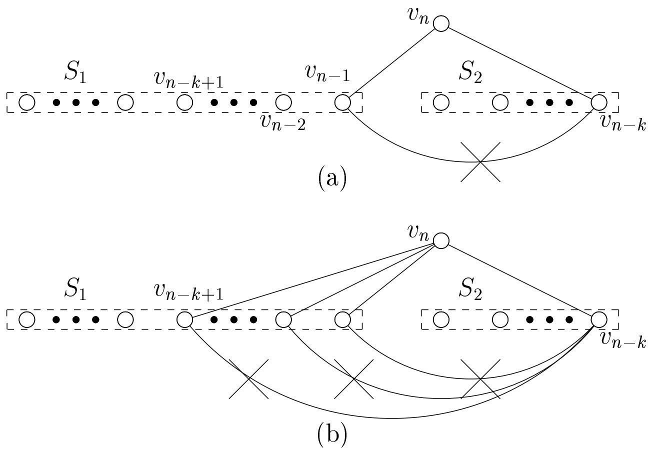

Figure 3: Illustrations for the proof of Theorem 1

this implies that $S_2 + \sigma^+(S_1, v_{n-k+1})$ and $\sigma^-(S_1, v_{n-k+1})$ satisfy property A for G' and $\sigma^-(S, v_n)$, which contradicts the definition that k is the minimum. $\square$

In a directed acyclic graph with vertex weights, we define the length of a path Π to be the sum of the weights of the vertices on Π. Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, v_2, \cdots, v_n\}$. For a sequence $S = [v_1, v_2, \cdots, v_n]$ and an integer $h \leq n$, let $D(G, S, v_h)$ be a directed graph with vertex set $\{v_1, v_2, \cdots, v_h\}$, in which there is an arc from v_i to v_j if and only if $i < j$ and v_i is not adjacent to v_j. Especially, $D(G, S, v_n)$ is denoted by $D(G, S)$. As an example, we show $D(G, S)$ in Figure 4 for the graph G in Figure 1 and $S = [v_1, v_2, v_3, v_4, v_5]$. Let $\ell(G, S, v)$ be the length of the longest path in $D(G, S, v)$ whose endpoint is v. Let $\ell(G, S)$ be the length of the longest path in $D(G, S)$, namely $\ell(G, S) = \max_{v \in V} \ell(G, S, v)$.

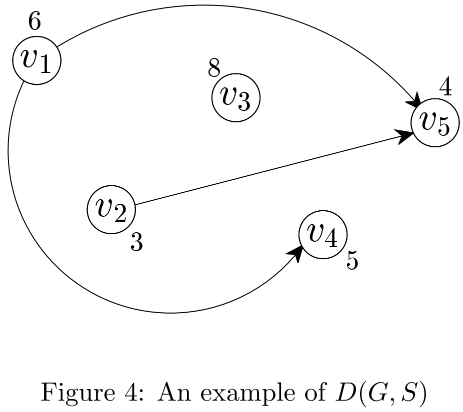

Figure 4: An example of $D(G, S)$

Theorem 2 *Let $G = (V, E)$ be an undirected graph with vertex weights and let $S = [v_1, v_2, \cdots, v_n]$, where $V = \{v_1, v_2, \cdots, v_n\}$. Let $w_3(G)$ be the weight of the maximum weight K_3-free subgraph in G. For any G and S, $w_3(G) \leq 2\ell(G, S)$.*

ISBN:978-988-17012-3-7

Proof of Theorem 2

Let $G' = (V', E')$ be the maximum weight K_3-free subgraph in G. Let S' be the subsequence of S which is obtained by removing the vertices in $V - V'$ from S. By Theorem 1, there are two sequences S_1 and S_2 which satisfy property A for G' and S'. Because any consecutive pair of vertices in S_1 are not adjacent in G, a directed path containing all elements in S_1 exists in $D(G, S)$. Because $\ell(G, S)$ is the length of the longest path in $D(G, S)$, the sum of the vertex weights in S_1 is not greater than $\ell(G, S)$. Same argument holds for S_2. Because the sum of the vertex weights in S_1 and S_2 equals to the sum of the vertex weights in V', $w_3(G) \leq 2\ell(G, S)$ holds. □

3 Our algorithm

Our algorithm for the MWK_3-free problem is an ordinary branch-and-bound algorithm, therefore we just describe the branching rule and the bounding rule. Let $P(X, Y)$ be the subproblem defined by X and Y, where X is the set of the vertices already chosen, and Y is the set of the candidates to be chosen. According to this notation, the original problem is denoted by $P(\emptyset, V)$.

3.1 Branching Rule

At first the algorithm constructs a sequence S in which each element of Y appears exactly once by the procedure shown in Section 3.2. For simplicity, we suppose that the procedure has produced $S = [v_1, v_2, ..., v_{|Y|}]$. Then our algorithm makes subproblems $P(X_i, Y_i) = P(X \cup \{v_i\}, \{v_1, v_2, \cdots, v_{i-1}\})$ for $i = |Y|, |Y| - 1, \cdots, 2, 1$ and solves them in this order.

Each subproblem $P(X_i, Y_i)$ might have useless vertices in Y_i. If a vertex v_h in Y_i is adjacent to v_i and is adjacent to some vertex in X which is also adjacent to v_i, these three vertices construct K_3. Therefore the vertex v_h cannot be added to X_i. Thus, before solving each subproblem $P(X_i, Y_i)$, our algorithm removes useless vertices like v_h in Y_i. This procedure is efficiently executed with using bit vectors.

3.2 Constructing Vertex Sequence

Theorem 2 guarantees that the value of the optimum solution of $P(X_i, Y_i)$ is not greater than $2\ell(G, S, v_i)$ for each i. If we can obtain S such that $\ell(G, S, v_i)$ is small for each i, pruning often occures and the computation time gets shorter. Therefore the desirable sequence is S which makes $\ell(G, S, v_i)$ small for each i. The procedure shown in Figure 5 is a simple heuristic for obtaining such a sequence S.

The procedure calculates S and $a(v_i) = \ell(G, S, v_i)$ for each i at the same time. This procedure is just a greedy algorithm to make each $a(\cdot)$ smaller, so does not guarantee that $a(\cdot)$ becomes minimum, but guarantees that the longest path in $D(G, S, v_i)$ always includes v_i. From this fact we obtain a little better upper bound $\ell(G, S, v_{i-1}) + \ell(G, S, v_i)$ for $P(X_i, Y_i)$, instead of $2\ell(G, S, v_i)$. The computational complexity of this procedure is $O(|V|^2)$.

1. Let S be an empty sequence.
2. Let $a(v) \leftarrow w(v)$ for each $v \in V$.
3. Repeat the following statements until $|S| = |V|$
   (a) Find a vertex v such that $a(v)$ is the minimum among all vertices in $V - S$.
   (b) Let $S \leftarrow S + [v]$.
   (c) For each t that is adjacent to v in $V - S$, let $a(t) \leftarrow a(v) + w(t)$.

Figure 5: Procedure to get S

3.3 Bouding Rule

Our bounding rule is very simple. The algorithm prunes a subproblem whose upper bound is not greater than the value of the temporal best solution. The upper bound for each subproblem is obtained during the construction of the sequence.

4 Computer Experiments

We executed our algorithm for random graphs with edge-density between 0.1 and 0.9. We set the number of vertices to 50, 100, or 200, and assigned integer vertex weights randomly between 1 and 10. For each condition, we executed the algorithm 10 times and measured the minimum and maximum values of CPU time. For this experiment, we used an AMD Athlon(tm) 64 X2 Dual Core Processor 5000+, Linux (kernel 2.6) and C++ programming language. The results are shown in Table 1. The exact solution can be obtained within practical time for each condition, but the variance of computation time is very big. We have not clarified the reason for this yet, but guess it is due to the looseness of upper bounds. Further studies are necessary to prove it.

5 Conclusions and Future Works

In this paper we showed an interesting fact for K_3-free graphs and the practical exact algorithm for the maximum weight K_3-free subgraph problem. This is the first exact algorithm for the problem. It might be possible to improve this algorithm by tighter upper bound calculation.

Although only the algorithm for the MWK_3-free problem was shown in this paper, a similar algorithm can be

Table 1: Benchmark Results

vertices	density	CPU time [sec]	
		min	max
50	0.9	0.00	0.00
50	0.8	0.00	0.00
50	0.7	0.00	0.00
50	0.6	0.00	0.01
50	0.5	0.01	0.04
50	0.4	0.01	0.08
50	0.3	0.10	0.35
50	0.2	0.14	1.87
50	0.1	0.15	8.52
100	0.9	0.00	0.01
100	0.8	0.02	0.04
100	0.7	0.10	0.21
100	0.6	0.61	1.18
100	0.5	3.46	9.15
100	0.4	30.17	146.61
100	0.3	264.56	1689.34
200	0.9	0.16	0.21
200	0.8	1.07	1.93
200	0.7	13.86	22.16
200	0.6	213.90	372.29

constructed for the MWK_r-free problem for any $r > 3$ if the following conjecture holds :

Conjecture 1 *For a K_r-free graph $G = (V, E)$ and a sequence S in which each element in V appears exactly once, there exist $(r-1)$ sequences $S_1, S_2, \cdots, S_{r-1}$ which satisfy the following property.*

Property A' :

- *Each element in V appears exactly once in one of $S_1, S_2, \cdots, S_{r-1}$.*
- *Any consecutive two elements in any one of $S_1, S_2, \cdots, S_{r-1}$ are not adjacent in G.*

References

[1] Babel, L., "A fast algorithm for the maximum weight clique problem," Computing, vol.52, pp.31–38, 1994.

[2] Fahle, T., "Simple and fast: Improving a branch-and-bound algorithm for maximum clique," Proc. 10th Annual European Symp. on Algorithms, Lecture Notes in Computer Science, vol.2461, pp.485-498, Springer, Berlin, 2002.

[3] Östergård, P.R.J., "A fast algorithm for the maximum clique problem," Discrete Applied Mathematics, vol.120, pp.197–207, 2002.

[4] Östergård, P.R.J., "A new algorithm for the maximum-weight clique problem," Nordic Journal of Computing, vol. 8, pp.424–436, 2001.

[5] Sewell, E.C., "A branch and bound algorithm for the stability number of a sparse graph," INFORMS Journal on Computing, vol.10, pp.438–447, 1998.

[6] Tomita, E., and Seki, T., "An efficient branch-and-bound algorithm for finding a maximum clique," Proc. 4th Int'l Conf. on Discrete Mathematics and Theoretical Computer Science, Lecture Notes in Computer Science, vol.2731, pp.278–289, Springer, Berlin, 2003.

[7] Wood, D.R., "An algorithm for finding a maximum clique in a graph," Operations Research Letters, vol.21, pp.211–217, 1997.

[8] Yamaguchi, K., Sakakibara, Y., and Masuda, S., "A generic method to extend an algorithm for the maximum clique problem to an algorithm for the maximum weighted clique problem", Proc. 2004 Int'l Technical Conf. on Circuits/Systems, Computers and Communications (CD-ROM), Sendai, 2004.