# A New Exact Algorithm for the Maximum Weight Clique Problem

Kazuaki Yamaguchi[1] and Sumio Masuda[2]
Faculty of Engineering, Graduate School of Kobe University
1-1 Rokkodai, Nada, Kobe 657–8501 JAPAN.
E-mail : [1]ky@kobe-u.ac.jp, [2]masuda@kobe-u.ac.jp

**Abstract**: Given an undirected graph with weight for each vertex, the maximum weight clique problem is to find the clique of the maximum weight. Östergård proposed a fast exact algorithm for solving this problem. We show his algorithm is not efficient for very dense graphs. We propose an exact algorithm for the problem, which is faster than Östergård's algorithm in case the graph is dense. We show the efficiency of our algorithm with some experimental results.

## 1. Introduction

For an undirected graph $G = (V, E)$ and a set $V' \subseteq V$, $G(V')$ denotes the subgraph of $G$ induced by $V'$. If any two vertices in $G(V')$ are adjacent to each other, $V'$ is called a clique in $G$. For a vertex $v \in V$, let $w(v)$ be the weight of $v$. For $V' \subseteq V$, let $w(V') = \sum_{v \in V'} w(v)$. Given a graph $G$ and vertex-weight $w(\cdot)$, the maximum weight clique problem is to find a clique $V'$ in $G$ of the maximum weight.

The maximum weight clique problem is a well-known NP-hard problem[4], and known to be hard to approximate[1]. On the other hand, some exact algorithms for not large graphs are proposed [2], [3], [5], [8].

We propose another exact algorithm and show the efficiency of our algorithm by comparing with the fastest algorithm known so far[5].

## 2. Preliminaries

Let $N(v)$ be the set of vertices adjacent to $v$ in $G$. For a DAG $\vec{D} = (V_D, E_D)$ and a directed path $P = (V_P, E_P)$ in $\vec{D}$, let $w(P) = w(V_P)$ (the length of $P$). For a DAG $\vec{D}$ and a vertex $v \in V_D$, let $\ell(\vec{D}, v)$ be the length of the longest path whose endpoint is $v$. Let $\ell(\vec{D}) = \max_{v \in V_D} \ell(\vec{D}, v)$. For a sequence $\Pi = [\pi_1, \pi_2, \cdots, \pi_{|V|}]$, let $set(\Pi, i) = \{\pi_1, \pi_2, \cdots, \pi_i\}$ $(i \leq t)$. Let $set(\Pi) = set(\Pi, t)$. For a graph $G = (V, E)$ and $\Pi$ where $set(\Pi) = V$, let $PA(\Pi, \pi_i) = set(\Pi, i) \cap N(\pi_i)$. Let $\vec{G}_\Pi$ be a DAG obtained by replacing each edge $(\pi_i, \pi_j) \in E$ $(i < j)$ in $G$ by a directed edge from $\pi_i$ to $\pi_j$. For example, $PA(\Pi, v_4) = \{v_1, v_2\}, PA(\Pi, v_6) = \{v_4\}$ for a graph $G$ in Figure 1(a) and $\Pi = [v_1, v_2, \cdots, v_7]$. The DAG $\vec{G}_\Pi$ is in Figure 1(b). Let $w^*(G)$ be the weight of the maximum weight clique in $G$.

## 3. Our Algorithm

Now we briefly introduce our algorithm. Our algorithm is based on branch and bound method.

### 3.1 Branch procedure

At first we show a theorem for the branch procedure.

*Theorem 1:* For an undirected graph $G = (V, E)$, vertex weight $w(\cdot)$ and $\Pi = [\pi_1, \pi_2, \cdots, \pi_{|V|}]$ where $set(\Pi) = V$, the following equation holds.

$$w^*(G) = \max_{1 \leq i \leq |V|} (w^*(G(PA(\Pi, \pi_i))) + w(\pi_i)).$$

(Omit the Proof)

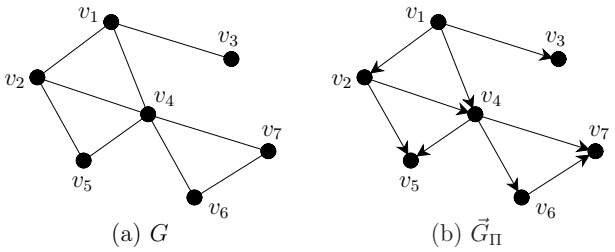


Figure 1. An example of $G$ and $\vec{G}_\Pi$.

For a problem for $G = (V, E)$ and a sequence $\Pi = [\pi_1, \pi_2, \cdots, \pi_t]$ where $set(\Pi) = V$, let $P_i(G, \Pi)$ for $1 \leq i \leq |V|$ be the subproblem whose any solution contains $\pi_i$ and does not contain $\pi_{i+1}, \pi_{i+2}, \cdots, \pi_{|V|}$. It is easy to show that the weight of the optimum solution for the subproblem $P_i(G, \Pi)$ is $w^*(G(PA(\Pi, \pi_i))) + w(\pi_i)$. Namely, the theorem just shows this fact by the equation. Our algorithm solves the subproblem $P_{|V|}(G, \Pi)$ at first, then solves subproblems in the order $P_{|V|-1}(G, \Pi)$, $P_{|V|-2}(G, \Pi)$, $\cdots$, $P_2(G, \Pi)$, $P_1(G, \Pi)$.

The theorem also shows that the optimum solution can be obtained from $|V|$ subproblems made by any sequence $\Pi$. The computation time of branch and bound program strongly depends on the solving order of subproblems. Thus it is very important to choose an appropriate vertex sequence $\Pi$. Our algorithm calculates the sequence and upper bounds at the same time. Therefore we will show the description of the algorithm later.

### 3.2 Bound procedure

The bound procedure prunes subproblems whose upper bounds are not greater than the value of the current best solution. Our upper bound calculation is based on the following two theorems.

*Theorem 2:* For an undirected graph $G = (V, E)$, vertex weight $w(\cdot)$ and a sequence $\Pi$ such that $set(\Pi) = V$, $w^*(G) \leq \ell(\vec{G}_\Pi)$.

(Proof) Let $C$ be one of the maximum weight cliques in $G$. For any sequence $\Pi$, $\vec{G}_\Pi$ has a path which contains all vertices in $C$. Thus $w^*(G) = w(C) \leq \ell(\vec{G}_\Pi)$. □

*Theorem 3:* For an undirected graph $G = (V, E)$, vertex weight $w(\cdot)$, a sequence $\Pi$ such that $set(\Pi) = V$ and any vertex $v \in V$, $w^*(G(PA(\Pi, v))) + w(v) \leq \ell(\vec{G}_\Pi, v)$.

(Proof) Let $G' = G(PA(\Pi, v))$. Let $\Pi'$ be a subsequence of $\Pi$ which consists of vertices in $PA(\Pi, v)$. Let $P'$ be the longest

Inputs: a graph $G = (V, E)$ and vertex weights $w(\cdot)$
Outputs: a sequence $\Pi$ and upper bounds $a(\cdot)$
(1) Let $\Pi$ be the empty sequence.
(2) For each $v \in V$, let $a(v) \leftarrow w(v)$.
(3) Let $S \leftarrow V$.
(4) Choose a vertex $v^*$ from S that minimizes $a(\cdot)$.
(5) Let $S \leftarrow S - \{v^*\}$.
(6) For each $u \in N(v^*) \cap S$, let $a(u) \leftarrow a(v^*) + w(u)$.
(7) Insert $v^*$ into $\Pi$ such that $\Pi$ becomes increasing order according to $a(\cdot)$.
(8) Halt if $set(\Pi) = V$.
(9) Go to (4).

Figure 2. Algorithm 1 (computing $\Pi$ and upper bounds).

Inputs: a graph $G = (V, E)$, vertex weights $w(\cdot)$ and an integer $\theta$
Outputs: the maximum weight clique $C$ if $w(C) > \theta$.
If such a clique does not exist, just return $\emptyset$.
(1) Let $C \leftarrow \emptyset$.
(2) Get a sequence $\Pi$ and $a(\cdot)$ by Algorithm 1.
(3) Let $i \leftarrow |V|$.
(4) Exit if $a(\pi_i) \leq \theta$.
(5) Get the maximum weight clique $C'$ of $P_i(G, \Pi)$.
(6) Go to (8) if $w(C') \leq \theta$.
(7) Let $C \leftarrow C'$ and $\theta \leftarrow w(C')$.
(8) Let $i \leftarrow i - 1$.
(9) Go to (4) if $i > 0$.

Figure 3. Algorithm 2 (obtaining the maximum weight clique).

path in $\vec{G}'_{\Pi'}$. From Theorem 2, $w^*(G') \leq \ell(\vec{G}'_{\Pi'}) = w(P')$. Let $P$ be a path obtained from $P'$ by adding $v$ at the end. Because $\ell(\vec{G}_\Pi, v)$ is the length of the longest path whose endpoint is $v$, $w(P) \leq \ell(\vec{G}_\Pi, v)$. From this inequality, $w^*(G(PA(\Pi, v))) + w(v) \leq w(P) \leq \ell(\vec{G}_\Pi, v)$ is immediately obtained. □

From Theorem 3, $\ell(\vec{G}_\Pi, \pi_i)$ is an upper bound for $P_i(G, \Pi)$. In Figure 2, we show Algorithm 1 which constructs a sequence $\Pi = [\pi_1, \pi_2, \cdots, \pi_{|V|}]$ and calculates upper bounds $a(\pi_1) = \ell(\vec{G}_\Pi, \pi_1)$, $a(\pi_2) = \ell(\vec{G}_\Pi, \pi_2)$, $\cdots$, $a(\pi_{|V|}) = \ell(\vec{G}_\Pi, \pi_{|V|})$. The computation time of Algorithm 1 is $O(|V|^2)$.

At each iteration, Algorithm 1 chooses $v^*$ to make $a(v^*)$ as small as possible, so that the pruning is expected to happen often by such choices.

### 3.3 The whole algorithm

The outline of our algorithm is shown in the following. At first, for a graph $G = (V, E)$ and vertex weights $w(\cdot)$, Algorithm 1 calculates a vertex sequence $\Pi$, where $set(\Pi) = V$. At the same time, upper bounds $a(\cdot)$ of subproblems are obtained. Then, by the recursive call, the optimum solution $C$ ($\subset V$) for the subproblem $P_{|V|}(G, \Pi)$ is obtained. The solution is a candidate for the global optimum solution. If $a(\pi_{|V|-1}) \leq w(S)$, any better solutions do not exist in the subproblem $P_{|V|-1}(G, \Pi)$. In such a case, the subproblem $P_{|V|-1}(G, \Pi)$ can be pruned. Otherwise, the optimum solution of $P_{|V|-1}(G, \Pi)$ is calculated. If a better solution $C'$ is obtained, $C$ is replaced by $C'$. Then similar procedures are executed for $P_{|V|-2}(G, \Pi), P_{|V|-3}(G, \Pi), \cdots, P_2(G, \Pi), P_1(G, \Pi)$. Figure 3 shows the whole algorithm described above. At the beginning, the algorithm is executed with $\theta = -1$.

At the step (5), Algorithm 2 calculates the optimum solution for the subproblem $P_i(G, \Pi)$. That can be done by recursive call for the subgraph induced by $set(\Pi, i-1) \cap N(\pi_i)$ with the threshold value $\theta - w(\pi_i)$.

### 3.4 Comparison with coloring

One of the fastest algorithms for the maximum cardinality clique problem is shown in [7]. The upper bound calculation of the algorithm is based on vertex-coloring. Thus, someone might think that the extended upper bound calculation based on vertex-coloring peforms well for weighted graphs if it is possible. But actually, the upper bound calculation based on vertex-coloring is not better than the upper bound calculation based on the longest path calculation. We will prove it in the following.

For a graph $G = (V, E)$ and a collection $C = \{V_1, V_2, \cdots, V_{|C|}\}$ which satisfies $V = \bigcup_{V_i \in C} V_i$ (a partition of $V$), $C$ is called a vertex-coloring if each element $V_i$ in $C$ is an independent set of $G$.

Let $z(G, C) = \sum_{V_i \in C} \max_{v \in V_i} w(v)$. Obviously $z(G, C)$ is a upper bound of $w^*(G)$, because the maximum weight clique can contain at most one vertex from each $V_i$. In the following lemma, $z(G, C)$ is not better than the upper bound $\ell(\vec{G}_\Pi)$, based on longest path calculation.

*Lemma 1:* For a coloring $C = \{V_1, V_2, \cdots, V_k\}$ of a graph $G$ and a vertex sequence $\Pi$ in which elements of each $V_i$ appear consecutive, $\ell(\vec{G}_\Pi) \leq z(G, C)$
(Proof) A path in $\vec{G}_\Pi$ includes at most one vertex in each $V_i$. Even if a path includes the vertex with the biggest weight in each $V_i$, still the length of the path is not greater than $z(G, C)$. Thus $\ell(\vec{G}_\Pi) \leq z(G, C)$. □

The lemma shows that the upper bound $z(G, C)$ is not better than the upper bound $\vec{G}_\Pi$. Actually, there are some examples of $G, w(\cdot), C$ and $\Pi$ which satisfy $\ell(\vec{G}_\Pi) < z(G, C)$. Let $G$ be the graph in Figure 4. Let

$$C = \{\{v_1\}, \{v_2, v_4\}, \{v_3, v_5\}\}$$
$$\Pi = [v_1, v_2, v_4, v_3, v_5]$$

Then, the values of $\ell(\vec{G}_\Pi) and z(G, C)$ are 8 and 10, respectively. In fact, $z(G, C) \geq 10$ for any vertex-coloring $C$ for the graph $G$.

If Algorithm 1 can be improved to produce the best sequence with regard to the upper bounds, the pruning will happen often and the number of subproblems produced in the branch procedure will be minimized, but actually obtaining the best vertex sequence is NP-hard. We show the theorem and its brief proof in the following.

*Theorem 4:* Let $G = (V, E)$ be a graph with $w(v) = 1$ for any $v \in V$. For any integer $h$, a vertex-coloring $C$ with cardinality $h$ exists in $G$ if and only if there exists a vertex sequence $\Pi$ which satisfies $\ell(\vec{G}_\Pi) \leq h$.

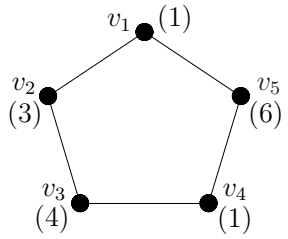


Figure 4. A case in which $\ell(\vec{G}_\Pi) < z(G, C)$.

(Proof) Let $C = \{V_1, V_2, \cdots, V_h\}$ be a vertex-coloring of $G$. Let $\Pi$ be a sequence of $V$ in whose elements of each $V_i$ appears consecutively. It is obvious that at most one vertex in each $V_i$ appears in any path $\vec{G}_\Pi$. Thus $\ell(\vec{G}_\Pi) \leq h$ holds.

Let $\Pi$ be a vertex sequence which satisfies $\ell(\vec{G}_\Pi) \leq h$. Let $V_i = \{x \mid \ell(\vec{G}_\Pi, x) = i\}$ for $i = 1, 2, \cdots, h$. Because any pair of vertices $u$ and $v$ satisfying $\ell(\vec{G}_\Pi, u) = \ell(\vec{G}_\Pi, v)$ are not adjacent, $\{V_1, V_2, \cdots, V_h\}$ is a vertex-coloring of $G$. □

Theorem 4 shows that it is unrealistic to obtain the vertex sequence which minimizes the upper bound.

## 4. Experiments

We implemented our algorithm in C++ to compare with the program "wclique3.c" shown at [6], based on Östergård's algorithm[5]. In the experiments, we use random 10 graphs for each condition. The result with some different graphs where vertex weight is ranged between 1 and 10 is shown in Table 1, and the result with some different graphs where vertex weight is ranged between 1001 and 1010 is shown in Table 2. The CPU and the operating system of the computer used in the experiments are Athlon X2 4800+ and Debian GNU/Linux version 4.0, respectively. The compiler is gcc 4.1.2 with an optimization option –O2.

### 4.1 Results of weight 1 to 10

Our algorithm is faster when the edge density is not smaller than 0.8. In other cases, Östergård's algorithm is faster than our algorithm. The computation time of Östergård's algorithm becomes drastically longer as the edge density of $G$ becomes higher. The computation time of our algorithm does not become as long as Östergård's when the graph is dense. Because of that, our algorithm is relatively faster than Östergård's when the graph is dense.

### 4.2 Results of weight 1001 to 1010

The computation time of two algorithms become almost equal when the edge density is 0.5 in this case. Because the computation time of Östergård's algorithm becomes much longer when the graph is dense, that Östergård's algorithm is not available in such cases.

## 5. Conclusion

In this paper, we proposed a new algorithm for the maximum weight clique problem, and compared the performance of our algorithm and Östergård's algorithm.

Table 1. Computational results (weight 1 to 10)

| $\|V\|$ | density | Östergård's [s] | ours [s] |
|---|---|---|---|
| 1000 | 0.1 | 0.02 | 0.05 |
| 1000 | 0.2 | 0.05 | 0.29 |
| 1000 | 0.3 | 0.26 | 1.97 |
| 1000 | 0.4 | 2.38 | 17.13 |
| 1000 | 0.5 | 39.10 | 244.1 |
| 500 | 0.6 | 8.38 | 27.03 |
| 500 | 0.7 | 359.1 | 754.9 |
| 200 | 0.8 | 7.86 | 7.15 |
| 100 | 0.85 | 0.05 | 0.05 |
| 150 | 0.85 | 3.04 | 2.36 |
| 100 | 0.9 | 0.26 | 0.15 |
| 150 | 0.9 | 633.1 | 15.63 |
| 200 | 0.9 | > 1000 | 466.10 |
| 100 | 0.95 | 1.67 | 0.36 |
| 150 | 0.95 | > 1000 | 81.89 |
| 100 | 0.98 | 5.66 | 0.14 |
| 150 | 0.98 | > 1000 | 124.76 |

Table 2. Computational results (weight 1001 to 1010)

| $\|V\|$ | density | Östergård's [s] | ours [s] |
|---|---|---|---|
| 500 | 0.5 | 8.62 | 8.58 |
| 500 | 0.6 | 264.88 | 168.93 |
| 100 | 0.7 | 0.02 | 0.02 |
| 150 | 0.7 | 0.50 | 0.29 |
| 100 | 0.8 | 0.21 | 0.08 |
| 150 | 0.8 | 17.52 | 3.38 |
| 100 | 0.85 | 2.06 | 0.29 |
| 150 | 0.85 | 287.23 | 20.40 |
| 100 | 0.9 | 17.01 | 0.73 |
| 150 | 0.9 | > 1000 | 107.13 |
| 100 | 0.95 | 247.27 | 1.13 |
| 150 | 0.95 | > 1000 | 430.31 |
| 100 | 0.98 | > 1000 | 0.60 |
| 150 | 0.98 | > 1000 | 69.54 |

Although Östergård's algorithm is faster than our algorithm when the graph is sparse, Östergård's algorithm cannot get the optimum solutions in case the graph is very dense. This fact implies that, even if the graph $G$ is sparse, Östergård's algorithm might take very long time if a very dense subgraph is contained in $G$.

To the contrary, our algorithm is much faster than Östergård's algorithm when the graph is dense. The computation time is reasonable even if the graph is dense and the deviation of the weight is small. That means that our algorithm is available for most cases, although it is slower if the graph is sparse.

It might be a good idea to choose an algorithm according to the density of the graph and the weights. We will investigate such hybrid algorithms.

**References**


[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties, Springer, Berlin, 1999.

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

[3] E. Balas and J. Xue, "Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring," Algorithmica, vol.15, pp.397–412, 1996.

[4] M.R. Garey and D.S. Johonson, Computers and Intractability – A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

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

[6] P.R.J. Östergård, http://www.tcs.hut.fi/~pat/wclique.html

[7] E. Tomita and T. Seki, "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, Dijon (France), vol.2731, pp.278–289, Springer, Berlin, 2003.

[8] K. Yamaguchi, Y. Sakakibara and S. Masuda, "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.