Journal of The Korea Society of Computer and Information
Vol. 20, No. 1, January 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.1.227

# 최대 클릭 문제에 관한 최대차수 정점 기반 알고리즘

이 상 운*

# Maximum Degree Vertex-Based Algorithm for Maximum Clique Problem

Sang-Un Lee *

## 요 약

본 논문은 NP-완전으로 알려진 최대 클릭의 정확한 해를 선형시간으로 찾는 알고리즘을 제안하였다. 먼저, 주어진 그래프 $G=(V,E)$에서 최대 차수 $\Delta(G)$ 정점 $v_i$를 클릭의 대표 정점으로 결정한다. $v_i$ 인접 정점 $N_G(v_i)$에서 $\Delta(G)$정점 $v_j$를 선택하여 $N_G(v_i)\cap N_G(v_j)$를 후보 클릭 $w$와 $v_k$로 결정한다. $d_G(v_k)$ 내림차순으로 $w=w\cap N_G(v_k)$를 얻는다. 마지막으로, $G\setminus w$그래프에서 동일한 절차를 수행하여 얻은 클릭이 기존에 얻은 클릭과 동일하거나 크면 이 클릭을 선정하는 검증과정을 거쳤다. 이와 같은 방법으로 독립된 다수의 클릭도 얻을 수 있는 장점이 있다. 제안된 알고리즘을 다양한 정규와 비정규 그래프에 적용한 결과 모든 그래프에 대해 선형시간 $O(n)$으로 정확한 해를 구하였다.

▸ Keywords : 최대 클릭, 차수, 최대차수, 후보 클릭

## Abstract

In this paper, I propose a linear time algorithm devised to produce exact solution to NP-complete maximum clique problem. The proposed algorithm firstly, from a given graph $G=(V,E)$, sets vertex $v_i$ of the maximum degree $\Delta(G)$ as clique's major vertex. It then selects vertex $v_j$ of $\Delta(G)$ among vertices $N_G(v_i)$ that are adjacent to $v_i$, only to determine $N_G(v_i)\cap N_G(v_j)$as candidate cliques $w$ and $v_k$. Next it obtains $w=w\cap N_G(v_k)$by sorting $d_G(v_k)$ in the descending order. Lastly, the algorithm executes the same procedure on $G\setminus w$ graph to compare newly attained cliques to previously attained cliques so as to choose the lower. With this simple method, multiple independent cliques would also be attainable. When applied to various regular and irregular graphs, the algorithm proposed in this paper has obtained exact

•제1저자 : 이상운
•투고일 : 2014. 12. 15. 심사일 : 2015. 1. 5, 게재확정일: 2015. 1. 12.
* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Eng., Gangneung-Wonju National University)

solutions to all the given graphs linear time $O(n)$.

▸ Keywords : Maximum clique, Degree, Maximum Degree, Candidate clique

## I. 서 론

클릭 (clique)이란 그래프 $G=(V,E), n=|V|, m=|E|$가 포함하고 있는 완전 그래프 (complete graph, $K_k$)로 구성된 부분집합이다[1],[2].

사회 연결망 (social network)에서 정점을 사람으로, 간선을 서로가 아는 지인 (mutual acquaintance)으로 표현하면 모든 사람들이 상호간에 면식 (친분)이 있는 최대 부분집합인 $k$개의 $K_k$-완전 그래프를 찾는 경우가 있다. 이러한 경우에 최대 클릭 (maximum clique, MC) $k$를 찾는다. 최대 클릭 문제 (MC problem, MCP)는 전화 통신망의 패턴, 오류수정코드 설계, 대형 다중프로세서의 고장진단, 컴퓨터 비전과 패턴인식 분야에 적용되고 있다.

최대 클릭 ($w(G)$)을 찾는 문제는 NP-완전 (NP-complete)으로 다항시간 (polynomial time)으로 정확한 해 (exact solution)를 찾는 알고리즘이 제안되지 않고 있다[1]-[3].

$n$개로 구성된 주어진 문제의 최적 해 (optimal solution)를 하노이 타워 (hanoi tower)나 외판원 문제 (traveling salesman problem, TSP)와 같이 식 (1)의 지수시간 (exponential time)으로 찾을 수 있는 NP-완전 (non-deterministic polynomial- complete) 문제를 식 (2)의 다항시간 (P)으로 찾을 수 있는 $P$-문제의 알고리즘이 존재하는지 여부를 증명하는 난제를 $P \neq NP$라 하며, 미국 클레이 수학재단 (clay mathematics institute)에서 21세기에 풀어야할 $1,000,000 상금을 내건 7가지 문제 (millenium problem) 중 첫 번째 문제이다.

$2^n, 3^n, \cdots$ 또는 $(n-1)!$ : 지수시간 ( 1 )

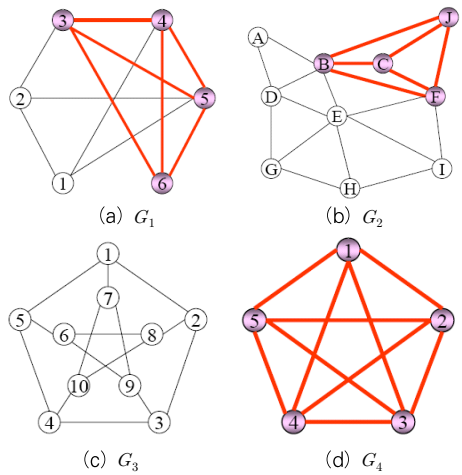
$n^2, n^3, \cdots$ : 다항시간 (2)

본 논문은 NP-완전으로 알려진 그래프 $G=(V,E)$에서 $w(G)=k$ 완전그래프인 클릭의 정확한 해를 선형시간으로 찾는 알고리즘을 제안하여 클릭 문제는 NP-완전이 아닌 P-문제임을 보인다. 2장에서는 $w(G)$의 개념과 관련연구를 고찰해 본다. 3장에서는 $w(G)=k$를 정확히 찾는 선형시간 알고리즘을 제안한다. 4장에서는 다양한 그래프들을 대상으로 제안된 알고리즘을 적용하여 성능을 검증한다.

## II. 관련 연구와 연구 배경

최대 클릭 $w(G)$은 주어진 그래프 $G=(V,E)$에서 최대가 되는 $K_k, (k \geq 3)$를 찾는 문제이다. 그림 1의 다양한 그래프들[4]-[8]을 대상으로 클릭을 고찰해 보자.

$G_1$과 $G_2$는 비정규 그래프 (non-regular graph)이며, $G_3$은 차수가 3인 입방체 정규 그래프 (cubic regular graph)로 Petersen 그래프로 잘 알려져 있다.




(a) $G_1$ (b) $G_2$

(c) $G_3$ (d) $G_4$

그림 1. 최대 클릭
Fig. 1. Maximum clique

$G_4$는 차수가 4인 정규 그래프로 $K_5$-완전 그래프 (complete graph)이다. $G_1$과 $G_2$의 클릭은 $w(G)=4$로, $w(G_1)=\{3,4,5,6\}, w(G_2)=\{B,C,F,J\}$이다. 반면에 $w(G_3)=\{1,2\}$로 $K_k, (k \geq 3)$ 조건을 만족하지 않아 클릭이 존재하지 않는다고 한다. $G_4$는 자체가 $K_5$-완전 그래프로 $w(G_4)=\{1,2,3,4,5,6\}$이다.

클릭 문제는 다양한 분야에 응용되고 있음에도 불구하고 최대 클릭 $w(G)=k$를 다항시간으로 찾는 알고리즘이 존재하지 않아 Karp의 21개 NP-완전 문제들 중 하나로 분류되어 있다[1]-[3].

NP-완전 문제는 최적 해를 다항시간으로 찾을 수 있는 알고리즘이 알려져 있지 않아 부득이 유전자 알고리즘 (genetic algorithm, GA), 진화 알고리즘 (evolutionary algorithm, EA) 등과 같은 메타휴리스틱 (metaheuristic) 방법들을 적용하여 근사 해 (approximate solution)를 다항시간으로 구한다.

MCP는 NP-완전임에도 불구하고, 메타휴리스틱 방법들을 적용하지는 않고 있으며, 지역탐색법 (local search, LS), 그래프 분할법 (graph partition, GP)과 분기한정법 (branch-and- bound, BB)의 다항시간 알고리즘들이 연구되고 있다.

지역탐색법으로는, Östergård[9]는 사전에 주어진 $k$에 대해 $K_k$-완전 그래프를 탐색하는 방법을, Pullane과 Hoos[10]는 동적 지역 탐색법 (dynamic LS, DLS)을, Pullan[11]은 단계적 지역 탐색법 (phased LS, PLS)을, Pullan et al.[12]은 협력 지역 탐색법 (coorporating LS, CLS)을, Xiangmei et al.[13]은 지역 탐색법 (LS)을, Balaji[14]는 새로운 효과적인 지역 탐색법을 제안하였다.

그래프 분할법에 대해서, Akbari[15]는 주어진 그래프에 대해 가지치기 (prune) 방법을 적용하고 그래프를 $K_3, K_4, \cdots$ 등으로 분할하는 그래프 분할법을 제안하였다.

분기한정법에 대해서, Konc와 Janežič[16]은 기존의 BB를 MCP에 적용하는데 문제점이 있어 개선된 BB (improved BB, IBB)를 제안하였다.

이와 같이 다양한 방법들이 제안되었음에도 불구하고, MCP의 최적 해를 보다 정확하게 찾는 방법 (규칙)은 알려져 있지 않고 있다. 따라서, 3장에서는 MCP의 최적 해를 정확하게 찾아가는 선형시간 규칙을 제안하여 MCP가 P-문제임을 보인다.

## III. 최대차수 정점 기반 알고리즘

본 장에서는 최대 클릭 $w(G)=k$를 선형시간으로 찾는 알고리즘을 제안한다. 제안되는 알고리즘은 그래프의 차수 (degree, $d_G(v)$) 개념을 적용한다. 그래프에서 가장 많은 부속 간선 (차수)을 가진 정점의 차수를 최대 차수 (maximum degree)라 하며 $\Delta(G)$로 표기한다. 반면에, 최소 차수 (minimum degree)를 $\delta(G)$로 표기한다[17].

제안 알고리즘은 식 (3)의 범위를 적용한다.

$$3 \le k \le \Delta(G)+1 \tag{3}$$

이를 위해, 주어진 그래프 $G(V,E)$에서 $d_G(v_i) \le 2$인 정점을 삭제하여 $G_1(V,E)$으로 축소시킨다. 다음으로, 최대차수 $\Delta(G)$ 정점을 클릭에 포함된 대표 정점 $v_i$ 로 결정하고, $v_i$ 정점의 이웃 정점 $N_G(v_i)$ 들 중에서 최대차수 $\Delta(G)$인 $v_j$ 정점을 선택한다. $w=N_G(v_i)\cap N_G(v_j)$정점들로만 구성된 $G_2(V,E)$으로 축소시킨다. $G_2(V,E)$에서 $V\setminus v_i, v_j$정점을 $v_k$로 설정한다. $v_k$의 차수 $d_G(v_k)$ 내림차순으로 $w=w\cap N_G(v_k)$를 수행하여 $w(G)=k$를 얻는다. 마지막으로 $w(G)=k$가 최대 클릭인지를 검증하기 위해 $G_3(V,E)=G_1(V,E)\setminus G_2(V,E)$그래프로 축소시키고, $d_G(v_i)>k, v_i \in G_3(V,E)$가 존재하면 다시 클릭을 계산하여 기존에 얻은 클릭과 동일하거나 크면 새로운 클릭으로 저장한다. 결국, 제안된 알고리즘은 주어진 그래프에서 동일한 클릭도 모두 검출할 수 있으며, 최대 클릭을 찾는 장점을 갖고 있다. 제안된 알고리즘을 최대차수 정점 기반 알고리즘 (maximum degree vertex- based algorithm, MDVA)라 하며, 상세한 과정은 그림 2에 제시되어 있다.

---

$G(V,E)$
**{PreProcessing}**: 그래프 단순화 /* 수행시간: $O(n)$ */
$d_G(v_i) \le 2$ 정점 삭제.
if $d_G(v_i)=1$로 변경된 정점 존재 then 해당 정점 삭제.
/* 최소한 $K_3$-클릭을 얻기 위해 $d_G(v_i) \ge 2$이 $d_G(v_i)=2$로 변경된 정점은 삭제 않음. */

$G_1(V,E)$
**{MainProcessing}**: 최대 클릭수 검출 /* 수행시간: $O(n)$ */
if $|\Delta(G)|>2$ then 임의 정점 $v_i$와 $N_G(v_i)$ 선택
$N_G(v_i)$에서 $\Delta(G)$ 정점 $v_j$와 $N_G(v_j)$ 선택 /* 동일 차수 다수 정점 존재시 임의 정점 선택. */
$v_k \leftarrow N_G(v_i)\cap N_G(v_j)$
else if $|\Delta(G)|=2$ then 모든 $\Delta(G)$ 정점$v_i$와 $N_G(v_i)$ 선택
$N_G[v_i]$에서 $\Delta(G)$ 정점$v_j$와 $N_G(v_j)$ 선택
$v_k \leftarrow \max\{N_G(v_i)\cap N_G(v_j)\}$
else if $|\Delta(G)|=1$ then
해당 $\Delta(G)$ 정점 $v_i$와 $N_G(v_i)$ 선택
$N_G(v_i)$에서 $\Delta(G)$ 정점 $v_j$와 $N_G(v_j)$ 선택
$v_k \leftarrow N_G(v_i)\cap N_G(v_j)$
$w=v_k+v_i+v_j$

$G_2(V,E)$

---

while $v_k = \{\phi\}$ do
  $v_k$의 $d_G(v_k)$ 내림차순으로 정점 선택
  $w \leftarrow w \cap N_G(v_k)$, $v_k \leftarrow w \cap N_G(v_k)$
end while
if $|w| \le 2$ then 클릭 수 미존재, 알고리즘 종료
else if $|w| > 2$ then perform **PostProcessing**.

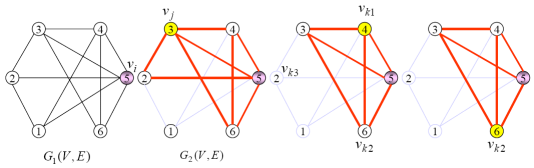
$G_2(V,E) = G_1 \setminus w$
**{PostProcessing}:** 최대 클릭 수 검증 /* 수행시간: $O(n)$ */
while $d_G(v_i) \le w$ do
  if $d_G(v_i) \ge w$ then perform **MainProcessing**
  else if $d_G(v_i) < w$ then 알고리즘 종료.
  if $|w(G_i)| \ge |w(G_{i-1})|$ then $w(G_i)$ 저장
    /* 동일 클릭 수 뿐 아니라 최대 클릭수를 얻음 */
  $G_i(V,E) = G_{i-1} \setminus w$
end while

그림 2. 최대차수 정점 기반 알고리즘
Fig. 2. Maximum degree vertex-based algorithm

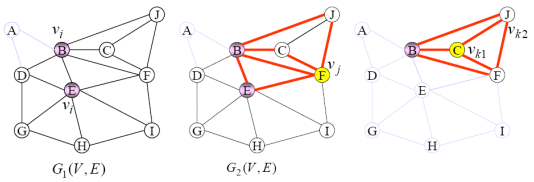
그림 1의 그래프에 대해 제안된 알고리즘을 적용한 결과는 그림 3에 제시하였다. $G_1$과 $G_2$그래프에 대해 제안된 알고리즘을 적용한 과정은 각각 표 1과 표 2에 제시하였다.

제안된 알고리즘은 정규 그래프와 비정규 그래프 모두에서 $w(G)=k$를 선형시간으로 간단히 찾을 수 있었다.

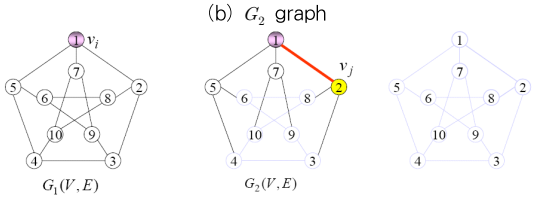
## IV. 실험 및 결과 분석



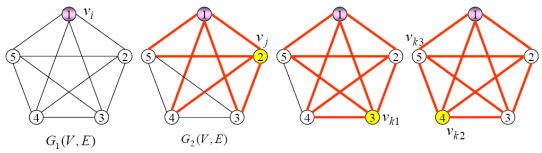

(a) $G_1$ graph




(b) $G_2$ graph




(c) $G_3$ graph




(d) $G_4$ graph
그림 3. MDVA 적용 결과
Fig. 3. Result of MDVA

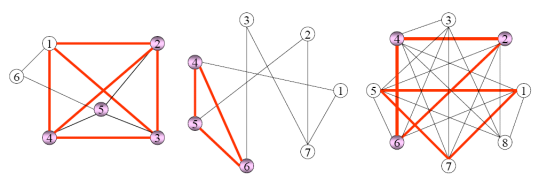
표 1. $G_1$ 그래프의 MDVA 적용 과정
Table 1. MDVA steps for $G_1$ graph

| PreProcessing: $G \rightarrow G_1$, {1,2,3,4,5,6} → {1,2,3,4,5,6} | | | | |
|---|---|---|---|---|
| MainProcessing: $G_1 \rightarrow G_2$ | | | | |
| $v_i$ | $N_G(v_i)$ | $v_j$ | $N_G(v_j)$ | $N_G(v_i) \cap N_G(v_j)$ |
| {5} | {1,2,3,4,5,6} | {3}<br>{4} | {2,3,4,5,6} | {2,3,4,5,6} |
| $v_k$ | $d_G(v_k)$ | $w \cap N_G(v_k)$ | | |
| {4} | 3 | {2,3,4,5,6}∩{3,4,5,6}={3,4,5,6} | | |
| {6} | 3 | {3,4,5,6}∩{3,4,5,6}={3,4,5,6} | | |
| {2} | 2 | {3,4,5,6}∩{3,4,5,6}={3,4,5,6} | | |
| | | $w(G)=4=\{3,4,5,6\}$ | | |
| PostProcessing: $G_3 = G_1 \setminus \{3,4,5,6\} = \{1,2\}$ | | | | |
| $d_G(1)=1$<br>$d_G(2)=1$ | $d_G(1)=1<4$<br>$d_G(2)=1<4$ | 알고리즘 종료 | | |

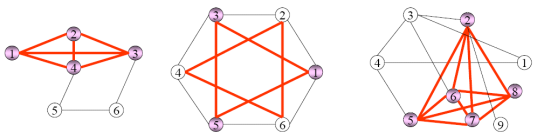
표 2. $G_2$ 그래프의 MDVA 적용 과정
Table 2. MDVA steps for $G_2$ graph

| PreProcessing: $G \rightarrow G_1$<br>{A,B,C,D,E,F,G,H,I,J} → {B,C,D,E,F,G,H,I,J} | | | | |
|---|---|---|---|---|
| MainProcessing: $G_1 \rightarrow G_2$ | | | | |
| $v_i$ | $N_G(v_i)$ | $v_j$ | $N_G(v_j)$ | $N_G(v_i) \cap N_G(v_j)$ |
| {B}<br>{J} | {B,C,D,E,F,J}<br>{B,D,E,F,G,H,I} | {F} | {B,C,E,F,I,J} | {B,C,E,F,J}<br>{B,E,F,J} |
| $v_k$ | $d_G(v_k)$ | $w \cap N_G(v_k)$ | | |
| {C}<br>{J} | 3<br>3 | {B,C,E,F,J}∩{B,C,F,J}={B,C,F,J}<br>{B,C,F,J}∩{B,C,F,J}={B,C,F,J}<br>$w(G)=4=\{B,C,F,J\}$ | | |
| PostProcessing: $G_3 = G_1 \setminus \{B,C,F,J\} = \{A,D,E,G,H,I\}$ | | | | |
| $d_G(A)=1$<br>$d_G(D)=2$<br>$d_G(E)=4$<br>$d_G(G)=3$<br>$d_G(H)=3$<br>$d_G(I)=2$ | $d_G(E)=4 \ge 4$ | {D,G,E}=3<br>{E,G,H}=3<br>{E,H,I}=3<br>{B,C,F,J}와 동일하거나 큰 클릭이 없음. | | |

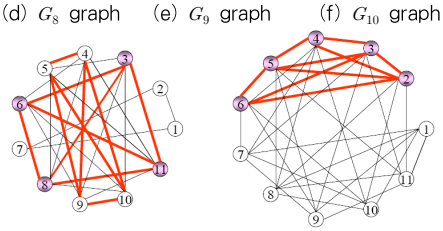
본 장에서는 그림 4의 다양한 그래프[5],[8],[18]-[22]에 대해 제안된 알고리즘을 적용하여 본다.



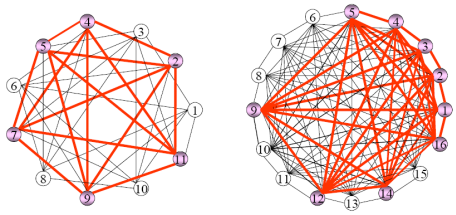
(a) $G_5$ graph (b) $G_6$ graph (c) $G_7$ graph



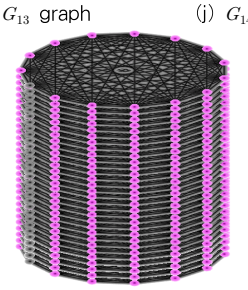
(d) $G_8$ graph (e) $G_9$ graph (f) $G_{10}$ graph



(g) $G_{11}$ graph (h) $G_{12}$ graph



(i) $G_{13}$ graph (j) $G_{14}$ graph

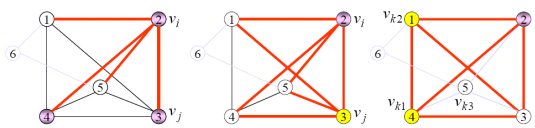


(k) $G_{15}$ graph

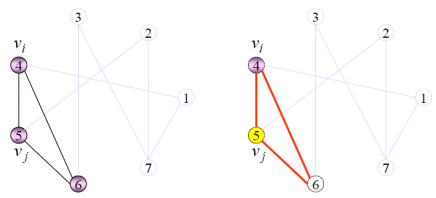
그림 4. 실험 그래프

Fig. 4. Experimental graphs

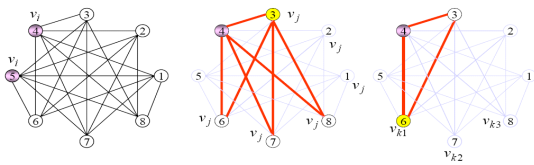
제안된 MDVA를 그림 4의 실험 데이터에 적용한 결과는 그림 5에 제시되어 있다.



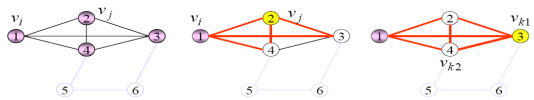
(a) $G_5$ graph



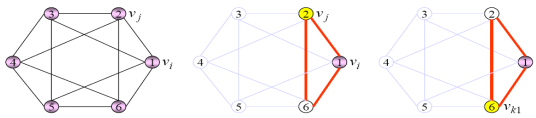
(b) $G_6$ graph



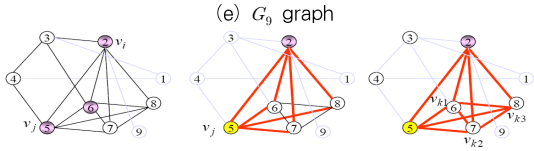
(c) $G_7$ graph



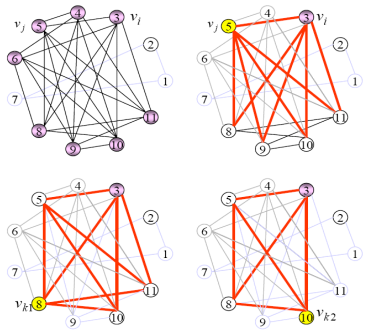
(d) $G_8$ graph



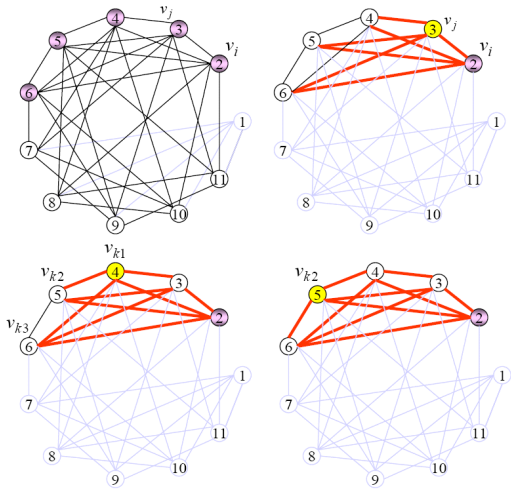
(e) $G_9$ graph



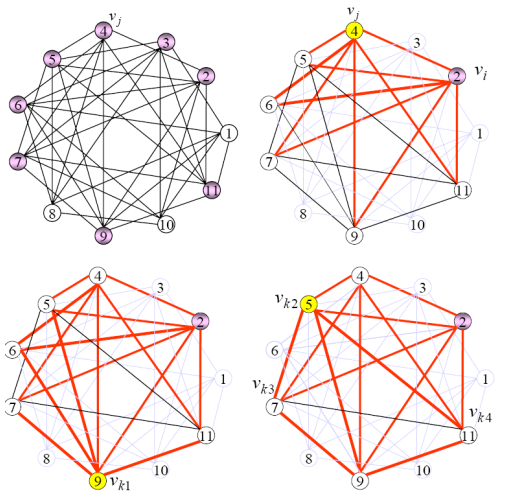
(f) $G_{10}$ graph



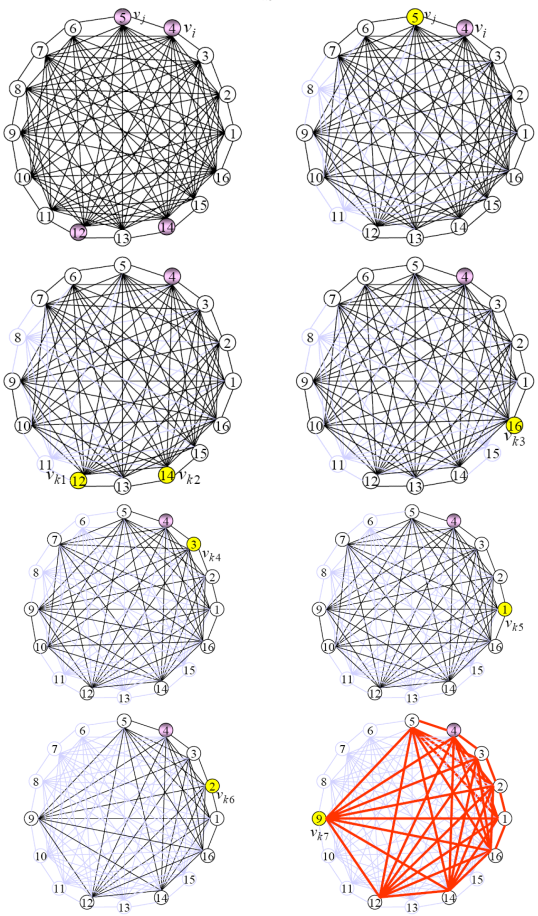
(g) $G_{11}$ graph



(h) $G_{12}$ graph



(i) $G_{13}$ graph



(j) $G_{14}$ graph

그림 5. MDVA 실험 결과
Fig. 5. Experimental result of MDVA

$G_{15}$ 그래프는 Dharwadker[21]와 Xu[22]에서 인용된 그래프로 450개 정점과 17,827개 간선으로 구성되어 있으며, 클릭 수는 30개로 각 클릭의 크기 $w(G)=15$이다. $G_{15}$ 그래프에 제안된 알고리즘을 적용한 결과는 그림 6과 같은 검증 과정을 거쳐 [1,15],[16,30],[31,45],…,[436, 450]의 30개 클릭을 모두 찾아낼 수 있었다.

[MainProcessing]
Rank: $\{v_i, v_j\}$ 포함, $d_G(89)=123, d_G(83)=122$
clique = [76,90]

$v_i = 89, N_G(89) =$
{16,17,18,20,21,22,23,25,27,28,29,30,33,34,37,43,44,62,67,69,74,76,77,78,79,80,81,82,**83**,84,85,86,87,88,89,90,92,94,100,123,125,128,129,130,135,153,155,156,157,160,165,168,173,175,176,202,205,209,210,231,234,237,238,241,245,247,248,249,251,252,257,264,269,272,275,276,278,279,280,283,284,288,294,295,301,302,303,306,307,309,310,313,314,318,320,322,325,329,336,337,338,342,345,361,363,364,365,366,368,371,374,375,391,393,396,397,399,401,404,409,411,415,416}

$v_j = 83, N_G(83) =$
{18,19,20,21,24,25,28,29,30,32,34,36,37,38,39,41,42,43,44,45,63,64,65,67,69,72,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,97,98,99,103,122,123,124,125,127,130,132,133,134,151,155,159,161,173,177,208,210,228,233,234,242,244,246,247,249,250,251,260,264,274,283,284,288,291,296,298,299,300,302,306,307,313,314,315,318,322,325,329,331,332,334,335,336,342,362,364,365,366,367,371,372,392,402,405,406,409,410,411,412,413,414,418}

$w = N_G(v_i) \cap N_G(v_j) =$
{**18,20,21,25,28,29,30,34,36,37,43,44,45,67,69,74**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,**92,93,94,123,125,130,155,173,210,234,247,249,251,264,283,284,288,302,306,307,313,314,318,322,325,329,336,342,365,366,371,409,411**}

$v_k$순위: $d_G(85)=37, d_G(80)=37, d_G(78)=37, d_G(84)=36,$
$d_G(86)=35, d_G(77)=34, d_G(81)=32, d_G(82)=32,$
$d_G(88)=32, d_G(79)=32$

$v_{k1} = 85, N_G(85) =$
{18,20,21,29,30,34,36,45,69,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,130,173,234,249,302,306,307,322,325,365,409,411}

$w = w \cap v_{k1} =$
{**18,20,21,29,30,34,36,45,69,74**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,**130,173,234,249,302,306,307,322,325,365,409,411**}

$v_{k2} = 80, N_G(80) =$
{20,28,29,36,37,43,44,45,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,123,130,249,251,306,307,325,329,342,365,366,409,411}

$w = w \cap v_{k2} =$
{**20,29,36,45,74**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90, **130,249,306,307,325,365,409,411**}

$v_{k3} = 78, N_G(78) =$
{18,20,21,25,28,30,36,37,43,44,45,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,130,155,210,247,251,264,302,307,313,325,366}

$w = w \cap v_{k3} =$
{**20,36,45**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90, **130,307,325**}

$v_{k4} = 84, N_G(84) =$
{21,25,36,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,92,94,123,125,234,247,249,264,302,307,322,336,342,365,366,409,411}

$w = w \cap v_{k4} =$ {**36**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,**307**}

$v_{k5}=86, N_G(86)=$
{20,21,28,29,36,37,43,44,45,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,92,93,123,125,130,234,247,302,313,336,365}
$w=w\cap v_{k5}=$ {**36**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90}
$v_{k6}=77, N_G(77)=$
{18,28,29,30,34,36,37,43,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,92,173,210,251,306,314,336,365,371,409}
$w=w\cap v_{k6}=$ {**36**,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90}
$v_{k7}=81, N_G(81)=$
{18,21,29,30,34,37,43,45,67,74,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,123,234,251,306,313,318,411}
$w=w\cap v_{k7}=$ {76,77,78,79,80,81,82,83,84,85,86,87,88,89,90}

{PostProcessing}
Rank: $\{v_i,v_j\}$ 포함,
$d_G(171)=107, d_G(99)=103, d_G(335)=101,$
$d_G(66)=101, d_G(175)=100$
clique = {166,180}

$v_i=171, N_G(171)=$
{31,33,35,37,40,43,44,47,48,52,58,92,93,95,97,101,113,114,116,118,120,121,124,132,140,146,148,151,153,155,159,160,161,162,163,166,167,168,169,170,171,172,173,174,**175**,176,177,178,179,180,181,184,185,186,188,189,190,191,193,194,198,200,201,203,207,209,210,226,227,233,235,237,240,245,246,249,251,252,253,258,259,260,263,264,265,266,267,268,270,275,278,281,283,285,351,352,360,376,382,388,390,408,409,411,414,415,439}
$v_j=175, N_G(175)=$
{33,34,36,38,39,40,41,42,45,50,56,92,110,111,114,120,124,126,128,135,136,146,147,149,150,154,162,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,186,189,192,193,194,195,197,199,200,201,205,227,229,239,240,244,247,248,249,250,251,252,254,260,261,263,265,266,267,268,270,271,272,273,275,276,277,278,279,280,281,283,284,285,349,355,377,381,382,387,408,410,411,416,417,441,442,447}
$w=N_G(v_i)\cap N_G(v_j)=$
{**33,40,92,114,120,124,146,162**,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,**186,189,193,194,200,201,227,240,249,251,252,260,263,265,266,267,268,270,275,278,281,283,285,382,408,411**}

$v_k$순 위 :
$d_G(173)=31, d_G(172)=30, d_G(178)=30, d_G(174)=30,$
$d_G(177)=29, d_G(169)=29, d_G(166)=29, d_G(179)=28,$
$d_G(176)=26, d_G(260)=25$

$v_{k1}=173, N_G(173)=$
{40,120,146,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,189,193,240,252,260,265,267,270,275,278,281,285,408}
$w=w\cap v_{k1}=$
{**40,120,146**,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,**189,193,240,252,260,265,267,270,275,278,281,285,408**}
$v_{k2}=172, N_G(172)=$
{40,92,114,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,186,189,240,251,252,263,265,266,267,278,281,285}
$w=w\cap v_{k2}=$
{**40**,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,**189,240,252,265,267,278,281,285**}
$v_{k3}=178, N_G(178)=$
{146,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,186,189,193,194,227,249,252,263,265,266,267,275,283,408}
$w=w\cap v_{k3}=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179,180, **189,252,265,267**}
$v_{k4}=174, N_G(174)=$
{33,40,92,120,146,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,193,249,260,265,267,268,275,283,382,408}
$w=w\cap v_{k4}=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179, 180,**265,267**}
$v_{k5}=177, N_G(177)=$
{33,120,124,162,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,189,193,200,240,251,260,263,267,268,285}
$w=w\cap v_{k5}=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179, 180,**267**}
$v_{k6}=169, N_G(169)=$
{92,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,186,193,194,249,252,263,265,267,268,270,275,278,283}
$w=w\cap v_{k6}=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179, 180,**267**}
$v_{k7}=166, N_G(166)=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,186,193,194,201,240,249,251,252,260,265,266,278,285,411}
$w=w\cap v_{k7}=$
{166,167,168,169,170,171,172,173,174,175,176,177,178,179, 180}
...

그림 6. $G_{15}$ 그래프에 대한 MDVA 실험 결과
Fig. 6. Experimental result of MDVA for $G_{15}$ graph

본 논문에서 거론된 15개 그래프에 대해 MDVA로 구한 $w(G)=k$와 기존에 알려진 해를 표 3에 비교하여 제시하였다. 제안된 MDVA는 15개 실험 데이터 모두에서 지금까지 다양한 메타휴리스틱 방법으로 찾아 알려진 최적 해를 단지 선형시간으로 찾을 수 있음을 보였다.

표 3. MDVA의 성능
Table 3. Performance of MDVA

| 그래프 | $w(G)=k$ | |
|---|---|---|
| | 알려진 값 | MDVA |
| $G_1$ | 4 | 4 |
| $G_2$ | 4 | 4 |
| $G_3$ | 0 | 0 |
| $G_4$ | 5 | 5 |
| $G_5$ | 4 | 4 |
| $G_6$ | 3 | 3 |
| $G_7$ | 3 | 3 |
| $G_8$ | 4 | 4 |
| $G_9$ | 3 | 3 |
| $G_{10}$ | 5 | 5 |
| $G_{11}$ | 4 | 4 |
| $G_{12}$ | 5 | 5 |
| $G_{13}$ | 6 | 6 |
| $G_{14}$ | 9 | 9 |
| $G_{15}$ | 15 | 15 |

## V. 결론

본 논문은 정규와 비정규 그래프 $G=(V,E)$를 입력으로 하여 최대 클릭 $w(G)=k$를 선형시간으로 정확히 찾는 알고리즘을 제안하였다.

제안된 알고리즘은 $\Delta(G)$ 정점 $v_i$의 인접정점 $N_G(v_i)$와 $N_G(v_i)$에서의 $\Delta(G)$정점 $v_j$의 인접정점 $N_G(v_j)$에 대해 $w=N_G(v_i)\cap N_G(v_j)$를 결정하고, $v_k=N_G(v_i)\cap N_G(v_j)$정점들의 차수 내림차순으로 $w=w\cap N_G(v_k)$를 연속적으로 선택하는 방법으로 $K_k$-클릭을 얻었다.

다양한 정규와 비정규 그래프에 대해 제안된 알고리즘을 적용한 결과 모든 그래프에 대해 $w(G)=k$를 선형시간으로 정확히 찾을 수 있었다.

결론적으로 MCP를 응용하는 전화 통신망의 패턴, 오류수정코드 설계, 대형 다중프로세서의 고장진단, 컴퓨터 비전과 패턴인식 분야에 제안된 알고리즘을 적용하면 선형시간으로 최적 해를 찾을 수 있어 많은 도움이 될 것이다.

## 참고문헌


〔1〕 Wikipedia, "Clique (Graph Theory), http://en.wikipedia.org/wiki/Clique_(graph_theory), Wikimedia Foundation Inc., 2014.

〔2〕 Wikipedia, "Karp's 21 NP-complete Problems, http://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems, Wikimedia Foundation Inc., 2014.

〔3〕 C. Umans, "CS21: Decidability and Tractability," Computer Science, California Institute of Technology, http://www.cs.caltech.edu/~umans/cs21/lec21.pdf, 2009.

〔4〕 Q. Ouyang, P. D. Kaplan, S. Liu, and L. Shumao, "DNA Solution of the Maximal Clique Problem," Science Vol. 278, pp. 446-449, 1997.

〔5〕 H. J. Kim. "Finding Clique Using Backtracking Algorithm," University of Washington, 2006.

〔6〕 E. W. Weisstein, "Complete Graph," Mathworld, http://mathworld.wolfram.com/CompleteGraph.html, 2014.

〔7〕 E. W. Weisstein, "Peterson Graph," Mathworld, http://mathworld.wolfram.com/PetersonGraph.html, 2014.

〔8〕 A. Dharwadker, "The Clique Algorithm," http://www. geocities.com/dharwadker/clique, 2006.

〔9〕 P. R. J. Östergård, "A Fast Algorithm for the Maximum Clique Problem," Discrete Applied Mathematics, Vol. 120, No. 1-3, pp. 197-207, Aug. 2002.

〔10〕 W. Pullan and H. H. Hoos, "Dynamic Local Search for the Maximum Clique Problem," Journal of Artificial Intelligence Research, Vol. 25, No. 1, pp. 159-185, Jan. 2006.

〔11〕 W. Pullan, "Phased Local Search for the Maximum Clique Problem," Journal of Combinatorial Optimization, Vol. 12, No. 3, pp. 303-323, Nov. 2006.

〔12〕 W. Pullan, F. Mascia, and M. Brunato, "Cooperating Local Search for the Maximum Clique Problem," Journal of Heuristics, Vol. 17, No. 2, pp. 181-199, Apr. 2011.

〔13〕 L. Xiangmei, D. Fei, and Z. Shao, "A Local Search Algorithm for the Maximum Clique Problem," Journal of Convergence Information Technology, Vol. 7, No. 14, pp. 65-70, Aug. 2012.

〔14〕 S. Balaji, "A New Effective Local Search Heuristic for the Maximum Clique Problem," World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical and Quantum Engineering, Vol. 7, No. 5, pp. 523-529, May 2013.

〔15〕 Z. O. Akbari, "A Polynomial-Time Algorithm for the Maximum Clique Problem," IEEE/ACIS 12th International Conference on Computer and Information Science, pp. 503-507, Jun. 2013.

〔16〕 J. Konc and D. Janežič, "An Improved Branch and Bound Algorithm for the Maximum Clique Problem," MATCH Communications in Mathematical and in Computer Chemistry, Vol. 58, No. 3, pp. 569-590, Dec. 2007.

〔17〕 Wikipedia, "Degree (Graph Theory),"

http://en.wikipedia.org/wiki/Degree_(graph-theory), Wikimedia Foundation Inc., 2014.
[18] P. M. Pardalos, "ESI 6448 Discrete Optimization Theory: Cliques, Quasi-cliques and Clique Partitions in Graphs," Department of Industrial and Systems Engineering, University of Florida, 2008.
[19] Y. D. Lyuu, "Clique," Dept. of Computer Science & Information Engineering, National Taiwan University, 2004.
[20] Mukul S. Bansal and V. Ch. Venkaiah, "A Note on Finding a Maximum Clique in a Graph Using BDDs", Australasian Journal of Combinatorics, Vol. 32, No. 1, pp. 253-258, Jan. 2005.
[21] A. Dharwadker, "The Vertex Coloring Algorithm," http://www.geocities.com/dharwadker/vertex_coloring/, 2006.
[22] K. Xu, "Benchmarks with Hidden Optimum Solutions for Set Covering, Set Packing and Winner Determination," Computer Science, National Laboratory of Software Development Environment, Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, 2009.

## 저 자 소 개

이 상 운(Sang-Un, Lee)
1983년 ~ 1987년 :
한국항공대학교 항공전자공학과 (학사)
1995년 ~ 1997년 :
경상대학교 컴퓨터과학과 (석사)
1998년 ~ 2001년 :
경상대학교 컴퓨터과학과 (박사)
2003.3 ~ 현 재 :
강릉원주대학교 멀티미디어공학과 부교수
관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 신뢰성, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr