Maximum Weight Clique Search (MWCS) 구현

지능기전공학부 무인이동체공학전공
18012635 양원석

1. 구현 아이디어

clique는 모든 정점이 이어져있는 서브 그래프를 의미한다. 한 그래프 안에서 가장 작은 clique는 간선으로 이어진 두 정점이다. 최소 clique의 두 정점과 모두 연결되어 있는 새로운 정점을 하나씩 추가하면 clique를 확장해 나갈 수 있다. 이후에도 같은 방법으로 clique를 한단계씩 확장해나가며 찾은 모든 clique 중 가장 가중치가 높은 clique를 찾는 방식으로 구현하였다.

2. 알고리즘 분석

2.1 정의

알고리즘에서 사용된 변수들에 대한 정의는 다음과 같다.

N, M : 정점의 개수와 간선의 개수

matrix :정점 연결 정보와 가중치를 담은 N * N의 2차원 정수 배열

maximal : 최고 가중치를 가진 clique의 정보를 담는 노드

cand_list : 현재 clique들의 정보를 담고 있는 리스트 ADT

2.2 알고리즘

1단계 : N, M으로 matrix 초기화

2단계 : 입력된 그래프 정보를 matrix에 저장

3단계 : maximal의 가중치를 0으로 초기화

4단계 : cand_list를 초기화. 초기화된 cand_list에는 두개의 정점으로 구성된 clique가 담겨있음

5단계 : cand_list를 순회하며 maximal 가중치 갱신

6단계 : cand_list에 clique가 있다면 7단계 과정 수행, clique가 더이상 없다면 8단계 수행

7단계 : 기존 cand_list에 있던 clique를 확장하여 cand_list를 갱신. 이후 5단계 과정 수행

8단계 : maximal에 저장된 clique의 정점과 가중치 정보 출력

2.3 주요 함수

```c
// Update clique list
// Each clique extend new vertice
t_list *update_cand_list(t_graph graph, t_list *cand_list)
{
    t_list *new_cand_list = init_list();
    t_node *cand, *new_cand;
    int    new_weight;

    cand = cand_list->head->next;
    while (cand)
    {
        for (int i = 1; i < graph.size + 1; i++)
        {
            // calculate new clique's weight
            new_weight = calculate_new_weight(graph, cand, i);
            if (new_weight == -1)
                continue ;
            // make new clique node and add to new clique list
            new_cand = copy_cand(cand, i, new_weight);
            lst_addback(new_cand_list, new_cand);
        }
        cand = cand->next;
    }
    // deallocate existing clique list
    free_cand_list(cand_list);
    return (new_cand_list);
}
```

기존 clique를 구성하는 모든 정점과 연결되어 있는 새 정점을 찾아 확장한 새 clique들을 반환한다.
clique들을 가지고 있는 리스트 ADT인 cand_list에 중복된 clique가 들어갈 경우 메모리 및 시간 효율이 감소하므로 중복된 clique를 넣지 않도록 했다. 이를 구현하기 위해 clique를 구성하는 정점을 오름차순으로 배열에 저장하여 하나의 clique에는 하나의 배열만 나올 수 있는 유일성을 보장하였다.

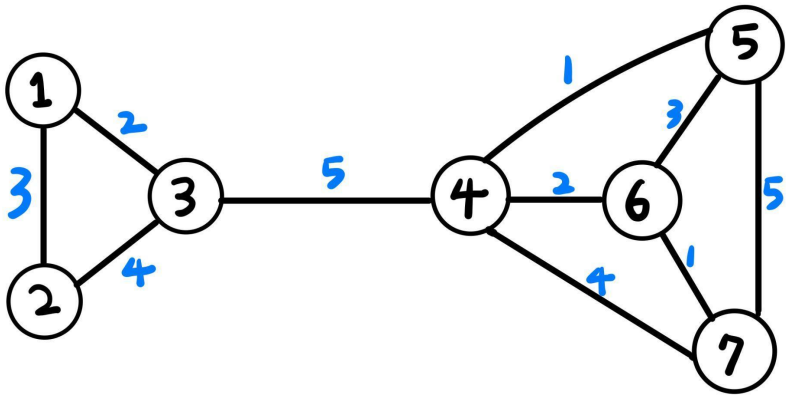
```c
// Used in lst_addback function
// Check that both nodes have the same content
// cand->vertice_list are always sorted
static int is_same_cand(t_node *cand1, t_node *cand2)
{
    for (int i = 0; i < cand1->size; i++)
    {
        if (cand1->vertice_list[i] != cand2->vertice_list[i])
            return (0);
    }
    return (1);
}

// List ADT method
// Add a node to list
// If there is the same content in the list, this work is ignored
void    lst_addback(t_list *list, t_node *node)
{
    t_node *now;

    now = list->head;
    while (now->next)
    {
        // check that nodes such as nodes to add are in the list
        if (is_same_cand(now->next, node))
            return ;
        now = now->next;
    }
    // if there is no same node, add a node to list.
    now->next = node;
    list->size += 1;
}
```

오름차순 정렬을 통해 유일성을 보장하게 되면, cand_list에 추가하는 과정에서 중복된 clique를 발견할 경우 무시할 수 있게 된다.
clique 노드들을 저장하는 자료구조로 연결리스트 ADT가 아닌 hash ADT를 사용하면 시간복잡도 측면에서 개선될 여지가 있다. 다만, 과제 분량상 이는 생략하였다.

3. 실행 예시 및 결과



여러 개의 clique가 나올 수 있는 위 그래프를 사용하여 실행하였다.
위 그래프에서 나올 수 있는 Maximum weight clique(MWC)는 가중치 16의 정점 4, 5, 6, 7로 구성된 clique이다.

```
wonyang 🐂  ~/goinfre
 cd "/Users/wonyang/goinfre/" && gcc assignment.c -o assignment && "/Users/wonyang/goinfre/"assignment
7 10
1 2 3
1 3 2
2 3 4
3 4 5
4 5 1
4 6 2
4 7 4
5 6 3
6 7 1
5 7 5
maximum weight clique : 4 5 6 7
weight : 16
```

위 그래프에서 정점 1과 2를 연결하는 간선의 가중치를 300으로 늘리면 MWC는 가중치 306의 정점 1, 2, 3으로 구성된 clique이다.

```
wonyang 🐂  ~/goinfre
 cd "/goinfre/wonyang/" && gcc assignment.c -o assignment && "/goinfre/wonyang/"assignment
7 10
1 2 300
1 3 2
2 3 4
3 4 5
4 5 1
4 6 2
4 7 4
5 6 3
6 7 1
5 7 5
maximum weight clique : 1 2 3
weight : 306
```

변경된 그래프에서도 MWC를 잘 찾아냄을 확인할 수 있다.

4. 참고 자료

[1] CAI, Shaowei; LIN, Jinkun. Fast Solving Maximum Weight Clique Problem in Massive Graphs. In: *IJCAI*. 2016. p. 568-574.
[2] LEE, G. H.; CHO, J. H. Comparison and Analysis on the Maximal Clique Finding Algorithms. *Electronics and Telecommunications Trends*, 1993, 8.4: 177-185.