An International Journal of Optimization and Control: Theories & Applications
ISSN:2146-0957 eISSN:2146-5703
Vol.10, No.1, pp.104-112 (2020)
http://doi.org/10.11121/ijocta.01.2020.00826

RESEARCH ARTICLE

Maximum cut problem: new models

Hakan Kutucu[a*] and Firdovsi Sharifov[b]

[a] *Department of Computer Engineering, Karabuk University, Turkey*
[b] *V.M. Glushkov Institute of Cybernetics, Ukraine*
hakankutucu@karabuk.edu.tr, f-sharifov@yandex.ru

ARTICLE INFO

Article History:
Received 27 May 2019
Accepted 16 October 2019
Available 31 January 2020

Keywords:
Convex function
Bases of polymatroid
Submodular function
Network flows

AMS Classification 2010:
90C27; 05C21; 52A41; 52B40

ABSTRACT

The maximum cut problem is known to be NP-hard, and consists in determining a partition of the vertices of a given graph such that the sum of the weights of the edges having one end node in each set is maximum. In this paper, we formulate the maximum cut problem as a maximization of a simple non-smooth convex function over the convex hull of bases of the polymatroid associated with a submodular function defined on the subsets of vertices of a given graph. In this way, we show that a greedy-like algorithm with $O(mn^2)$ time complexity finds a base of a polymatroid that is a solution to the maximum cut problem with different approximation ratio. Moreover, with respect to a base of a polymatroid, we formulate the maximum cut problem as a maximum flow problem between a source and a sink. We then investigate the necessary and sufficient conditions on the optimality of the base in terms of network flow.

1. Introduction

The well-known maximum cut problem consists in determining a partition of the vertices of a given graph such that the sum of the weights of the edges having one end node in each set is maximum. The maximum cut problem is very easy to state but hard to solve. This problem is one of the first problems whose NP-hardness was established in [1] by Karp. Note that the problem remains NP-hard even for unit edge weights [2,3].

The solution of the maximum cut problem has been approached by mathematical programming. In terms of design variables for every vertex, an integer quadratic programming formulation is given in [4]. Further integer linear programming formulations of the maximum cut problem using the boolean design variables are given in [5]. The algorithm in [6] for finding solutions of the maximum cut problem is an efficient method from a practical point of view. Goemans and Williamson use a semidefinite relaxation technique. Their experiments show that exact solutions are obtained in a reasonable time for any maximum cut instance of size up to 100 vertices. Using a semidefinite relaxation, the authors achieve an approximation ratio of 0.87856 for this difficult combinatorial optimization problem. Semidefinite programming is a convex optimization approach with a linear objective function of the design variables for a symmetric matrix, subject to linear constraints, and also convex constraints requiring the matrices to be positive semidefinite. Despite the fact that the algorithm in [6] has one of the best worst-case performance, Bertoni, Campadelli and Grossi [7] show that the algorithm improved by Goemans and Williamson has a complex design and its computation time may be prohibitive on large problem instances having more than 500 vertices. By solving experimental test problems on large random graphs, Bertoni et al. also show that their algorithm is better than the semidefinite programming algorithm of [6] and they define cuts with the same values in less time on standard benchmarks. Ben-Ameur et al. discuss the complexity of the maximum cut problem and some

*Corresponding Author

cases where the problem can be solved in polynomial time [8]. They also introduce some approximation methods for the maximum cut problem, both with and without guarantees.

In all references mentioned, the topological properties of a given graph did not play an essential role in proofs or in solving the maximum cut problem. Differently from the investigations mentioned above, to solve the maximum cut problem, some polynomial time algorithms have been developed based on topological properties of given graphs such as planar graphs [9,10], weakly bipartite graphs with non-negative edge weights [11], graphs without K_5 minors [12]. The problem is solved using a linear time algorithm for series-parallel graph [13].

For definitions used in the paper, we refer readers to [14,15]. Following the success of the theory of polymatroids in solving difficult combinatorial problems, we apply a polymatroid approach to the maximum cut problem.

Section 2 contains necessary notations and definitions in the theory of polymatroids used throughout the paper. In Section 3, we present the maximum cut problem as a maximization of a simple non-smooth function over a special polytope $P(f)$ called a polymatroid [14, 16] associated with the submodular function $f(S)$ defined on subset S of V of a given graph $G = (V, E)$. This model includes variables for each node in V. The convexity of the objective function implies that an optimal solution to the maximum cut problem is among extreme points (bases) of the polytope (polymatroid) $P(f)$ [17].

It is well known that the greedy algorithm defines bases of $P(f)$ according to different linear ordering of vertices, in polynomial time (see [14, 16]). One might say that for each maximum cut problem, *an optimal* linear ordering of vertices has to be chosen such that an optimal base of $P(f)$ (an optimal solution) can be defined by the greedy algorithm in polynomial time. Hardness of the maximum cut problem implies that an optimal linear ordering cannot be defined in polynomial time. In [18], Sharifov proposes a $O(mn^2)$ time algorithm which defines different linear ordering and related bases of $P(f)$ based on the topological properties of a given graph. We show that a solution to the maximum cut problem with the approximation ratio $0,75\lambda$ can be defined in $O(mn^2)$ time by this algorithm, where $\lambda \leq 1.3$ is some positive number and $m = |E|, n = |V|$. In Section 4, we present a new model of the maximum cut problem in terms of flows with respect to a base of $P(f)$. This model is used in the proof of new necessary and sufficient conditions for optimality of a base of $P(f)$.

2. Basic notions and preliminary results

Consider an undirected graph $G = (V, E)$ with non-negative weights $c_e \geq 0$ on the edges $e \in E$. We assume that G is a graph without loops and parallel edges. An edge with endpoints v and u is denoted by (v, u) and uv denotes the arc whose tail is v, and head is u. We use $\overline{S} = V \setminus S$ for $S \subseteq V$ and $S + v$ for $S \cup \{v\}$ when $v \notin S$, and $S - v$ for $S \setminus \{v\}$ when $v \in S$.

Let $\gamma(S)$ and $\kappa(S)$ denote the subsets of edges having at least one of endpoints in $S \subseteq V$ and both endpoints in $S \subseteq V$, respectively. Consider functions

$$f(S) = \sum(c_e : e \in \gamma(S))$$
$$g(S) = \sum(c_e : e \in \kappa(S)).$$

Obviously $f(S)$ and $g(S)$ are monotone functions by definitions, moreover, it is well known that f is submodular, i.e.,

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

and g is supermodular, i.e.,

$$g(S) + g(T) \leq g(S \cup T) + g(S \cap T)$$

for any $S, T \subseteq V$ [16].

The cut given by a subset $S \subset V$ is denoted by $\delta(S)$. We will use $c(E)$ for $\sum_{e \in E} c_e$, and $c(\delta(S))$ for $\sum(c_{ij}; (i, j) \in E, i \in S, j \in \overline{S})$. The vector $d = (d_v = c(\delta(v)); v \in V)$ is called the *weighted degree vector* of the graph G. From the definition of the sets $\gamma(S)$ and $\kappa(S)$ it follows that

$$f(S) + g(S) = d(S) = \sum_{v \in S} d_v,$$

and

$$f(S) - g(S) = c(\delta(S))$$

for the cut $\delta(S)$ given by any $S \subset V$. Clearly, $f(S) - g(S)$ is a submodular function.

Let $\mathbb{R}^V$ denote the set $\{(u(v) \in \mathbb{R} : v \in V)\}$. For $u = (u(v) : v \in V) \in \mathbb{R}^V$ and a subset $S \subseteq V$, we denote $u(S) = \sum_{v \in S} u_v$. The following two sets of vectors in $\mathbb{R}^V$ associated with the functions f

and g are called polymatroid and superpolymatroid [14], respectively:

$$P(f) = \{x \in \mathbb{R}^V;\ x(S) \leq f(S),\ S \subseteq V\},$$
$$Q(g) = \{y \in \mathbb{R}^V;\ y(S) \geq g(S),\ S \subseteq V\}.$$

The following polytope associated with the function $f - g$ is called extended polymatroid [14]:

$$EP(f-g) = \{w \in \mathbb{R}^V;\ w(S) \leq f(S)-g(S),\ S \subseteq V\}.$$

Vectors $x \in P(f)$ and $y \in Q(g)$ are called bases of the polymatroid and the superpolymatroid if $x(V) = f(V)$ and $y(V) = g(V)$, respectively. Note that, for any bases $x \in P(f)$ and $y \in Q(g)$,

$$x(V) - y(V) = f(V) - g(V) = 0,$$

since

$$\gamma(V) = E = \kappa(V)$$

by definition of the sets $\gamma(S)$ and $\kappa(S)$. So, a vector $w \in EP(f-g)$ is a base of $EP(f-g)$ if $w(V) = 0$.

Let $x^L \in P(f)$ and $y^L \in Q(g)$ be bases computed by the greedy algorithm in [18] with respect to any linear ordering L of the vertices. The first observation is that the difference $w^L = x^L - y^L$ of the bases x^L and y^L is a base of $EP(f-g)$ which can also be found by the greedy algorithm with respect to the linear ordering L of the vertices. In what follows, we will write x, y and w instead of x^L, y^L and w^L, respectively.

We write $v \prec_L u$ if v precedes u in the linear ordering L of the vertices. According to the linear ordering L of vertices, one can orient the edges of the graph $G = (V, E)$ in such a way that the resulting digraph $G = (V, A)$ is an acyclic oriented graph. This requires each edge (v, u) to be replaced by an arc vu if $v \prec_L u$ or an arc uv if $u \prec_L v$. The opposite is also true; each acyclic orientation of the edges of the graph $G = (V, E)$ defines a linear ordering L of its vertices. In an acyclic oriented graph $G = (V, A)$ with weights c_{vw} on arcs, let $\delta_+(v)$ be the set of arcs entering to node v, and let $\delta_-(v)$ be the set of arcs leaving from node v.

Our key observation is that the bases $x \in P(f)$ and $y \in Q(g)$ satisfy the equalities

$$\sum_{u \in \delta_+(v)} c_{vu} = c(\delta_+(v_i)) = x_v,\ v \in V, \qquad (1)$$

$$\sum_{u \in \delta_-(v)} c_{uv} = c(\delta_-(v_i)) = y_v,\ v \in V. \qquad (2)$$

In other words, x_v is the sum of weights on the leaving arcs from the node v, and y_v is the sum of weights on the entering arc to the node v.

All the above equalities are satisfied by any bases of $x \in P(f)$ and $y \in Q(g)$ which are computed with respect to any linear ordering of the vertices in any graph. Their proof immediately follows from the greedy algorithm formula for computing bases of $P(f)$ and $Q(g)$ with respect to a given linear ordering of the vertices. So, we can state the following claims.

Claim 1. *Let $x \in P(f)$ and $y \in Q(g)$ be any bases computed by the greedy algorithm developed in [18] with respect to any linear ordering L of the vertices, then*

$$x + y = d$$

and the difference $x - y = w$ is a base of $EP(f-g)$, for which the following the zero sum equality

$$\sum(w_v; w_v > 0) = -\sum(w_v; w_v <= 0)$$

holds.

Proof. Since $d_v = c(\delta_+(v_i)) + c(\delta_-(v_i))$ in the graph, obtained by orientation of the edges $G = (V, E)$ according to the linear ordering L, then $x_v + y_v = d_v$ for any node $v \in V$. From $x(V) = y(V)$ it follows that $w(V) = x(V) - y(V) = 0$. Since $x \in P(f)$ and $y \in Q(f)$, $x(S) - y(S) \in EP(f-g)$. Besides, $w(V) = 0$ can be written as the zero sum equality by performing algebraic operations. Therefore, $x - y = w$ is a base of $EP(f-g)$. □

Claim 2. *For a given linear ordering $L = \{v_1, ...v_n\}$ of the vertices in V, the bases $x(L) \in P(f)$, $y(L) \in Q(g)$ and $w(L) \in EP(f-g)$ can be found in $O(m)$ time.*

Proof. For $L_i = \{v_1, ...v_i\}$ and $i = 1, ..., n$, by the greedy formula

$$x_{v_i} = f(L_i) - f(L_{i-1}) = c(\delta_+(v_i)),$$

it follows that x_{v_i} is the sum of weights on the edges $(v_i, v_j) \in E$ for which $v_i \prec_L v_j$ in L. So, each edge of E appears only once in computing the base x. From $y = d - x \in Q(g)$ and $w = x - y \in EP(f-g)$, we obtain the bases y

and w in $O(m)$ time with respect to the linear order L. □

We make the following useful observation whose proof immediately follows from equalities (1) and (2), for two graphs obtained after the orientation of the edges in G with respect to linear orderings $L = \{v_1, v_2, ..., v_n\}$ and $I = \{v_n, ..., v_2, v_1\}$.

Claim 3. *If the greedy algorithm defines the bases $x^1 \in P(f)$ and $x^2 \in P(f)$ with respect to the L and I, respectively, then $x^2 = y^1 = d - x^1 \in Q(g)$ and $x^1 = y^2 = d - x^2 \in Q(g)$.*

Proof. Consider two acyclic oriented graphs $G(L)$ and $G(I)$ obtained after replacing edges in E by arc according to L and I, as above. If $v \prec_L u$, then an edge (v, u) corresponds to arc vu in $G(L)$ and arc uv in $G(I)$. Let x^1_v and $y^1_v = d_v - x^1_v$ be bases defined by the greedy algorithm [18] with respect to L, that is, equalities (1) and (2) hold for x^1_v and y^1_v with respect to $G(L)$. Let x^2_v and $y^2_v = d_v - x^1_v$ be bases defined by the greedy algorithm with respect to I. Since an edge (v, u) corresponds to arc vu in $G(L)$ and arc uv in $G(I)$, then x^1 defined by (1) for $G(L)$ is y^2 for $G(I)$ and y^1 defined by (2) for $G(L)$ is x^2 for $G(I)$. By Claim 1, $x^1, x^2 \in Q(g)$. □

The greedy algorithm [18] defines the following bases with respect to the linear order $L = (W, U)$ for a bipartite graph $H = (W, U, A)$:

$$x_v = d^H_v, \text{ for } v \in W, \; x_u = 0, \text{ for } u \in U,$$
$$y_v = 0, \text{ for } v \in W, \; y_u = d^H_u, \text{ for } u \in U.$$

Therefore,

$$\sum_{v \in W \cup U} |x_v - y_v| \tag{3}$$

is equal to the double weight of the maximum cut separating the sets W and U, in the bipartite graph $H = (W, U, A)$. To the best of our knowledge, there is no an algorithm to define a linear ordering L of vertices for non-bipartite graph in order to determine a maximum cut. Our goal in the next sections is to develop some new ideas for finding maximum cut in a non-bipartite graph $G = (V, E)$.

3. Models with convex objective function

The maximum cut problem of a graph $G = (V, E)$ is to find the set of vertices S that maximizes the weight of the edges in the cut $(S, \overline{S})$, i.e., the weight of the edges with one end node in S and the other in $\overline{S}$. In this section, we propose an alternative formulation for the maximum cut problem. First, the relationship between cuts and bases of the polymatroids $P(f)$ is established. In a linear ordering L of vertices, if $v \prec_L u$ for any vertices v and u such that $v \in S \subset V$ and $u \in \overline{S}$, we write it as $L = (S, \overline{S})$ for short.

Theorem 1. *The double weight of a cut $\delta(S)$ separating sets S and $\overline{S}$ is equal to*

$$\sum_{v \in V} |x^L_v - y^L_v|, \tag{4}$$

where bases $x^L \in P(f)$ and $y^L \in Q(g)$ are computed by the greedy algorithm with respect to the linear ordering $L = (S, \overline{S})$ of vertices in V.

Proof. Consider the linear ordering $L = (S, \overline{S})$ of the vertices in V, that is, $v \prec_L u$ for any node $v \in S$ and $u \in \overline{S}$, and let $x^L \in P(f)$ and $y^L = d - x^L$. According to the linear ordering L, we can direct each edge (v, u) of the graph as arc vu, if $v \prec_L u$ or as uv if $u \prec_L v$. Then all edges in E will be directed as arcs v_1v_2, if $v_1 \prec_L v_2$ for vertices $v_1, v_2 \in S$, as arcs u_1u_2 if $u_1 \prec_L u_2$ for vertices $u_1, u_2 \in \overline{S}$ and as arcs vu, where $v \in S$ and $u \in \overline{S}$. Clearly, after deleting the arcs v_1v_2 with end nodes $v_1, v_2 \in S$ and the arcs u_1u_2 with end nodes $u_1, u_2 \in \overline{S}$, in G, the resulting subgraph is a bipartite subgraph $H = (S, \overline{S}, A)$ ($S \subset V$ and $A \subseteq E$). With respect to H, one can define the functions f_0 and g_0, also the matroids $P(f_0)$, $Q(g_0)$ and $E(f_0 - g_0)$. Consider the bases $h \in P(f_0)$ and $t \in Q(g_0)$ defined by the greedy algorithm with respect to L, i.e.,

$$h_v = d^H_v, \text{ for } v \in S, \; h_u = 0, \text{ for } u \in \overline{S},$$
$$t_v = 0, \text{ for } v \in S, \; t_u = d^H_u, \text{ for } u \in \overline{S}.$$

Hence, $t_v = d^H_v - h_v$ for any node $v \in V$, and

$$2 \sum_{(u,v) \in A} c_{uv} = \sum_{v \in S} d^H_v + \sum_{v \in \overline{S}} d^H_v =$$
$$= h(S) - t(S) + |h(\overline{S}) - t(\overline{S})|.$$

Since h is defined with respect to L and $t_v = d^H_v - h_v$, then

$$h(T) - t(T) = x^L(T) - y^L(T), \text{ for } T = S, \overline{S}.$$

Therefore,

$$2 \sum_{(u,v) \in A} c_{uv} = x^L(S) - y^L(S) + |x^L(\overline{S}) - y^L(\overline{S})|$$

$$= \sum_{v \in V} |x_v^L - y_v^L|.$$

Thus, H is a maximum bipartite subgraph (the sum of weights on its edges is maximum) of G, if the last sum is maximum for the bases x^L and y^L. □

Since f is a monotone submodular function, the convex hull of bases $P(f)$ is the following polytope [14]

$$B(f) = \{x; x \geq 0, x(S) \leq f(S), S \subset V, x(V) = f(V)\}$$

We recall that $x + y = d$ (Claim 1) for the bases x and y generated by the greedy algorithm with respect to any linear ordering L of vertices. Thus, $y_v = d_v - x_v$ for all $v \in V$, and hence

$$|x_v - y_v| = |2x_v - d_v|.$$

We now present our original formulation of the problem. By Theorem 1, the maximum cut problem can be formulated as the following special convex program;

$$MaxCut_* = \max\{Cut(x) = \sum_{v \in V} |2x_v - d_v|\} \quad (5)$$

subject to

$$x \in B(f). \quad (6)$$

In what follows, we propose further formulations for the max-cut problem. To this end, we first state the following lemma. Let

$$f_+(x) = \sum_{v \in V_+(x)} (2x_v - d_v),$$
$$f_-(x) = \sum_{v \in V_-(x)} (d_v - 2x_v),$$

where

$$V_+(x) = \{v \in V; 2x_v - d_v > 0\},$$
$$V_-(x) = \{v \in V; 2x_v - d_v \leq 0\},$$

for any base $x \in B(f)$.

Lemma 1. *For any base $x \in B(f)$*

$$Cut(x) = 2f_+(x) = 2f_-(x).$$

Proof. From the equality

$$f_+(x) - f_-(x) = \sum_{v \in V} (2x_v - d_v) = 0$$

it follows that

$$Cut(x) = f_+(x) + f_-(x) + 0 = 2f_+(x),$$
$$Cut(x) = f_+(x) + f_-(x) - 0 = 2f_-(x).$$

□

By Lemma 1,

$$MaxCut_* = f_+(x^*) + f_-(x^*) = 2f_+(x^*) = 2f_-(x^*),$$

where x^* is an optimal solution to the problem (5)-(6). Since $z = 2x - d = x - y$ for any $z \in EP(f - g)$ and $x \in B(f)$ by Claims 1-3, the vector $z^+ = \{z_v^+; v \in V\}$, where $z_v^+ = \max\{z_v, 0\}$ can be defined with respect to each base $x \in B(f)$. By the equality $Cut(x) = 2f_+(x)$, the following problem

$$\max\{z^+(V); z \in EP(f - g)\}$$

is equivalent to the maximum cut problem (5)-(6). In addition to the above models, by the equality $Cut(x) = f_+(x) + f_-(x)$, the problem

$$\max\{f_+(x) : x \in B(f)\} = c(E)$$
$$- \min\{x(V_-) + y(V_+), x \in B(f), y = d - x\} \quad (7)$$

is also equivalent to the maximum cut problem (5)-(6). The problem in the right hand side of equality (7) can be considered as *dual* of the problem (5)-(6). We note that $z = 2x - d = x - y$ for $x \in B(f)$ and $y = d - x$. We can also define the vector $z^- = \{z_v^-; v \in V\}$, where $z_v^- = \min\{z_v, 0\}$ for any $v \in V$. It is easy to show that the following problem

$$\min\{z^-(V); z \in EP(f - g)\},$$

is equivalent to the above dual problem (7). Thus, the latter problem can be considered as another dual problem of the problem (5)-(6).

Moreover, Lemma 1 says that to solve the maximum cut problem on a given undirected graph, one can find a base $z \in EP(f - g)\}$ for which either $z^+(V)$ is maximum or $z^-(V)$ is minimum. It is well known that the latter problem is used essentially to design polynomial algorithms for minimizing a submodular function. For more details, the reader can refer to [16].

The above models are also useful for solving the maximum cut problem. For example, since the algorithm in [18] defines $O(n)$ bases of $B(f)$ in $O(mn^2)$ time, we can apply it to handle different bases and related strings $Cut(x), V_+(x), V_-(x)$. Let $LIST(Cuts)$ contains strings $Cut(x), V_+(x), V_-(x)$ for each these bases.

Theorem 2. *In $LIST(Cuts)$, if there are strings for some pair of different bases $x^*, x \in B(f)$ such that*

$$Cut(x^*) \geq x(V_+) + y(V_-),$$

for $y = d - x$, $V_+ = V_+(x)$ and $V_- = V_-(x)$, then

$$\frac{MaxCut_*}{2} \geq \frac{3}{4}c(E).$$

Proof. If $LIST(Cuts)$ contains the string for base x, then Claim 3 implies that $x(V_+) = f(V_+)$ and $y(V_-) = f(V_-)$. Let $LIST(Cuts)$ contains the string for the base x^*, too. Then

$$Cut(x^*) \geq f(V_+) + f(V_-)$$
$$= f(V_+) + g(V_-) + \frac{Cut(x)}{2} = c(E) + \frac{Cut(x)}{2}.$$

To define $Cut(x)$, the algorithm in [18] chooses a node $w \neq s$ for which $2x_w - d_w \geq 2x_v - d_v > 0$ for $v \notin V_+$, and sets $V_+ := V_+ + w$, where $V_+ := s$ at the beginning of the algorithm, and $V_- = V \setminus V_+$. This implies that $Cut(x)/2 \geq c(E)/2$. Thus,

$$Cut(x^*) \geq c(E) + \frac{c(E)}{2} = \frac{3}{2}c(E),$$

which completes the proof of the theorem. □

In $LIST(Cuts)$, let

$Cut(x^*) = \max\{Cut(x); Cut(x) \in LIST(Cuts)\}$,

and let $x(V_+) + y(V_-)$ be minimum for a base x ($y = d-x$). In other words, Theorem 2 states that $Cut(x^*)$ is a solution to the maximum cut problem with the approximation ratio at least 0.75. In this case, clearly the graph G has a cut with value at least $3/4c(E)$. If the graph G does not have a cut with value $3/4c(E)$, then Theorem 2 is not true. In this case, we define λ from the equality

$$Cut(x^*) = \lambda(x(V_+) + y(V_-)).$$

Clearly, λ is a positive number and $\lambda \leq 1.3$. By the proof of Theorem 2, it can be shown that

$$\frac{MaxCut_*}{2} \geq \lambda\frac{3}{4}c(E).$$

So, the graph G has a cut with value at least $3/4\lambda c(E)$. In this case, the algorithm in [18] defines a solution to the maximum cut with the approximation ratio at least 0.75λ for some positive number $\lambda < 1$.

As a conclusion of this section, we note that simplicity of the algorithm in [18] allows to solve real practical large problems effectively by Theorem 2. In future, we plan to do some investigations in this direction.

4. Maximum flow model

Now, we formulate the maximum cut problem by another model. Let $x \in B(f)$ be a base generated by the greedy algorithm with respect to a linear ordering L of vertices in V and let $z = 2x - d = x - y \in EP(f - g)$. Since $z(V) = 0$, we can define subsets

$$V_+ = \{v; z_v = 2x_v - d_v > 0, v \in V\}$$

and

$$V_- = \{w; z_w = 2x_w - d_w < 0, w \in V\}.$$

We consider an acyclic oriented graph $G = (V, A)$ obtained after replacing all edges by arcs according to the linear ordering of L. The capacity on each arc vu equals to the given weight of the edge $(v, u) \in E$. We add two new vertices, a source s and a sink r, to the graph $G = (V, E)$. For each vertex $v \in V_+$ and $w \in V_-$, we add arcs sv and wr with capacity z_v and $|z_w|$ to the graph $G = (V, E)$, respectively. In the resulting network $G_z = (V_z, E_z)$, let $\delta_+(S)$ denote the set of entering arcs to the vertices $S \subset V_z$ and $\delta_-(S)$ denote the set of leaving arcs from the vertices of the subset S. Recall that the capacity of the cut separating a subset of S is defined as the sum of the flows on the leaving arcs entering to vertices $v \in S$ minus the sums of the flows on the entering arcs to vertex $v \in S$. A cut with a minimum capacity is called a minimum cut.

Theorem 3. *In the network $G_z = (V_z, E_z)$, any maximum $s - r$ flow (source s, and sink r) saturates all arcs, i.e., on all arcs vu with end node $v, u \in V$, the value of the maximum flow equals to c_{vu}, and on all arcs sv and rw, the value of the maximum flow equals to z_v and $|z_w|$, respectively.*

Proof. Let x be a base generated by the linear ordering of L and $y = d - x$. From the definitions of the capacity of arcs sv and rw, it follows that $c(\delta_+(v)) = c(\delta_-(v))$ in the network G_z. This means that the sums of capacities of arcs in the sets $\delta_+(v)$ and $\delta_-(v)$ are the same for each vertex $v \in V$. Therefore, to maintain a balance between leaving and entering flows for each vertex $v \in V$, the value of the maximum flow on arcs of the acyclic oriented graph $G = (V, A)$ must be equal to its capacity. In addition, since

$$z(\delta_+(s)) = x(V_+) - y(V_+) = y(V_-) - x(V_-) \\ = |z(\delta_-(r))|$$

it follows that z_v and $|z_w|$ are the maximum flow values on the arcs sv and rw, respectively. □

By Theorem 3, since the value of the flow on all arcs is equal to its capacity, that is, any cut separating the source s and the sink r are the minimum cut in the network $G_z = (V_z, E_z)$. Thus, we obtain that $f_+(x)$ is the value of the maximum flow from the source s to the sink r in the constructed network $G_z = (V_z, E_z)$ according to the base $z = 2x - d \in EP(f - g)$.

So, Theorem 3 implies that to solve the maximum cut problem on a given undirected graph $G = (V, E)$, it needs to find a base $z \in EP(f - g)$ such that the capacity of any minimum cut separating the source and sink is maximum in the constructed network $G_z = (V_z, E_z)$. Such a model of the maximum cut problem can have applications for transportation of natural products from time to time in different directions through pipelines of the transport network.

Definition 1. *Let $G_z = (V_z, E_z)$ be a network constructed for the base $z = 2x - d \in EP(f - g)$. The flow on each arc vw is called transit, if vertices v, w are either in $V_+(x)$ or in $V_-(x)$.*

Theorem 4. *A base $z = 2x - d \in EP(f - g)$ is an optimal solution to the problem (5)–(6) if and only if a maximum flow from source to sink has a minimum sum of transit flows in the network $G_z = (V_z, E_z)$ constructed for the base z.*

Proof. Let x be the base generated by the linear ordering of L and the network G_z constructed for $z = 2x - d$ contains the minimum sum of transit flows on the arcs vw. By definition of a cut in the graph G with respect to the bases x and $y = d - x$, if $v, w \in V_+(x)$, then $y(V_+(x))$ if $v, w \in V_+(x)$, then $x(V_-(x))$ are the total number of transit flows on the arcs vw. Therefore, from the dual equality (7), we obtain that $\delta(V_+(x))$ is a maximal cut in the graph G.

If we consider that $y(V_+(x)) + x(V_-(x))$ is the sum of the transit flows in the network G_z constructed for arbitrary bases $z = 2x - d$ and $x \in B(f)$, then the opposite also follows from the dual equality (7). □

In other words, Theorem 4 states that if the minimum number of variables satisfies inequalities $0 < x_v < d_v$ in solving the problem (5)-(6), then a definite cut for x_v is maximal in the graph G. It is relatively difficult to design an effective algorithm based on this theorem. At a first glance, one might think that the network $G_z = (V_z, E_z)$ should not contain much more transit flows if $x(V_+(x)) = y(V_-(x))$. However, the situation is very complicated, since it is easy to design some small maximum cut problems for which this is not true. Indeed, this theorem states some connection between the maximum independent set and the maximum cut problems, that require new investigations on network flow problems.

5. Concluding remarks

The value of applications of the theory of polymatroids ensures that the optimal solution of many combinatorial optimization problems can be found in polynomial time bounded algorithms. For example, the vector $z \in P(f)$ maximizes cz in polynomial time for the monotonic submodular function f. A deep understanding of this theory makes it possible to use known methods developed for solving the network flows, as a solver of subtasks enumerating in solving optimization problems with a nonlinear objective function over polymatroids structures (see [14]). Considering topological properties of graphs under consideration in solving combinatorial problems over polymatroids leads to a polynomial algorithm as a solver for the maximum cut problem. In [10], topological properties of planar graphs namely the geometric duality is used to develop a polynomial time bounded for finding a maximal cut of these graphs. Since we do not know about unambiguous connections between NP and P, it is difficult to come up with a polynomial time bounded algorithm for solving (5)-(6), only using the above described and other specifics of the problem. But, with respect to the specifics of the objective function (5) and constraints (6), we hope that the next two weaker questions can be solved by a polynomial time algorithm.

(1) Is it possible to design greedy type algorithm by using the subgradient of the objective function at a current point (base) x to compute the next point x^k such that (5) will be strongly increased?
(2) How topological properties of a given graph and the techniques described in the

paper could be combined for finding an exact upper bound of the objective function (5)?

Based on positive answers to these two questions a polynomial time algorithm can be developed for finding an optimal solution to (5)-(6) on graphs with unit edge weights and as a result, we could get $NP = P$.

Acknowledgments

The authors are grateful to the anonymous referees for their constructive comments and valuable suggestions which have helped us very much to improve the quality of the paper.

References

[1] Karp, R.M. (1972). Reducibility among combinatorial problems. In: R.E. Miller and J.W. Thatcher, eds. Complexity of Computer Computations. Plenum Press, 85-103.

[2] Garey, M.R., Johnson, D.S., & Stockmeyer, L. (1976). Some simplified NP-complete graph problems. Theoretical Computer Science, 1(3), 237-267.

[3] Garey, M.R., & Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company.

[4] Rendl, F., Rinaldi, G., & Wiegele, A. (2010). Solving MAX-CUT to optimality by intersecting semidefinite and polyhedral relaxations. Mathematical Programming, 121(2), 307-335.

[5] Boros, E., & Hammer, P.L. (2002). Pseudo-Boolean optimization. Discrete Applied Mathematics, 123(1), 155-225.

[6] Goemans, M., & Williamson, D.P. (1995). Improved approximation algorithms for MAX-CUT and satisfiability problems using semidefinite programming. Journal of the ACM, 42(6), 1115-1145.

[7] Bertoni, A., Campadelli, P. & Grossi, G. (2001). An approximation algorithm for the maximum cut problem and its experimental analysis. Discrete Applied Mathematics, 110(1), 3-12.

[8] Ben-Ameur, W., Mahjoub, A.R., & Neto, J. (2014). The maximum cut problem. In: V.T. Paschos, ed. Paradigms of Combinatorial Optimization: Problems and New Approaches, 2nd Edition, J. Wiley and Sons, 131-172.

[9] Orlova, G.I., & Dorfman, Y.G. (1972). Finding the maximum cut in a graph. Engineering Cybernetics, 10(3), 502-506.

[10] Hadlock, F.O. (1975). Finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing, 4(3), 221-225.

[11] Grötschel, M., & Pulleyblank, W.R. (1981). Weakly bipartite graphs and the max-cut problem. Operations Research Letters, 1(1), 23-27.

[12] Barahona, F. (1983). The max-cut problem in graphs is not contractible to K5. Operations Research Letters, 2, 107-111.

[13] Chaourar, B. (2017). A Linear Time Algorithm for a Variant of the MAX CUT Problem in Series Parallel Graphs. Advances in Operations Research, 2017, 1-4.

[14] Fujishige, S. (2005). Submodular Function and Optimization. Annals of Discrete Mathematics, 2nd ed. Elsevier Science, Amsterdam.

[15] Nemhauser, G. & Wolsey, L.A. (1998). Combinatorial Optimization. Wiley-Interscience, New York.

[16] Iwata, S. (2008). Submodular function minimization. Mathematical Programming, 112(1), 45-64.

[17] Bazaraa, M.S., Sherali, H.D., & Shetty, C.M. (2006). Nonlinear programming: Theory and Algorithms. 3rd ed. John Wiley and Sons, New York.

[18] Sharifov, F.A. (2018). Finding the maximum cut by the greedy algorithm. Cybernetics and Systems Analysis, 54(5), 737-743.

Hakan Kutucu *received one of his master degree from International Computer Institute in 2004 other from Department of Mathematics at Ege University in 2008. He completed doctoral programme of Department of Mathematics at Ege University in 2011. At the present time, he has focused on network design problems, combinatorial optimization and mathematical modeling. He has been working in the Department of Computer Engineering at Karabuk University in Turkey since 2013.*
http://orcid.org/0000-0001-7144-7246

Firdovsi Sharifov *is senior researcher at the Department of combinatorial optimization methods and intellect information technologies at the Glushkov institute of Cybernetics of the National Academy of Sciences of Ukraine in Kyiv. He received his Ph.D. and Doctor of Science degrees in Combinatorial optimization in this institute. He was a team leader for grants projects with a joint professor at the National Aviation University. His research interests include combinatorial optimization, computer science and operation research.*
http://orcid.org/0000-0001-8768-3649

An International Journal of Optimization and Control: Theories & Applications (http://ijocta.balikesir.edu.tr)