

On Mathematical Models for the Maximum d -Cut Problem

Lorenzo dos Santos Corrêa

Departamento de Ciência da Computação - Instituto de Ciências Exatas UFMG - Universidade
Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
lorenzoscortea@ufmg.br

Marcio Costa Santos

Departamento de Ciência da Computação - Instituto de Ciências Exatas UFMG - Universidade
Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
marciocs@dcc.ufmg.br

Guilherme C. M. Gomes

Departamento de Ciência da Computação - Instituto de Ciências Exatas UFMG - Universidade
Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
gcm.gomes@dcc.ufmg.br

ABSTRACT

This paper investigates the *Maximum d -Cut Problem*, a generalization of the classic Maximum Cut and Matching Cut problems. Given a weighted graph and a parameter d , the goal is to find a vertex partition that maximizes the total weight of edges crossing the cut, under the constraint that each vertex has at most d neighbors in the opposite partition. We present mathematical programming formulations for the problem, including quadratic and linear integer programming models, and propose a preprocessing technique based on the identification of indivisible subgraphs. This technique reduces problem size by detecting vertex subsets that must belong to the same partition in any feasible solution. We also prove a sufficient condition under which a subgraph admits a d -cut, contributing to the theoretical understanding.

KEYWORDS. d -Cut Problem. Partition Problem. Maximum d -Cut Problem.

Paper topics: Graph partitioning. Maximum d -Cut Problem. Linear mathematical modelling.

1. Introduction

Graph partitioning is a genre of problems in mathematics that requires the separation of the vertices of a graph into mutually exclusive groups. Depending on the underlying application some extra constraints must be applied to the partitions in order to embed meaning in the solution of the partition problem. Some common constraints or restrictions are linked to the specific kind of forbidden subgraphs and conflict between vertices.

For example, *The Matching Cut Problem*, as one of these problems, requires a partition of the vertices of a graph $G = (V, E)$ into two disjoint subsets A and B such that,

$$\forall v \in A, |N(v) \cap B| \leq 1$$

$$\forall v \in B, |N(v) \cap A| \leq 1$$

where $N(v)$ is the neighborhood of v on G (Chvátal, 1984; Graham, 1970).

Defining if a graph admits a matching-cut is an NP-complete problem, with studies having already been done regarding other limitations to a subset of graphs. When considering graphs whose maximum degree is four or above, the problem remains NP-complete, but for graphs in which the maximum degree is three or below, the problem is polynomial-time solvable (Chvátal, 1984).

The *Maximum Cut Problem* is another problem regarding partitioning on graphs. It also asks for the partition of a graph into two sets, but with the intent of maximizing the cut size. The size of a cut is the number of edges with one endpoint on each set. The maximum cut problem has applications in various fields, such as social networks (Harary, 1959) and image segmentation (de Sousa et al., 2013). This problem also has some variants, such as the weighted version, where the objective is to maximize the sum of weights from the cut's edges, and the decision variant, where it is asked if, given a graph G and an integer k , there exists a cut with size at least k in this graph. The decision variant is proven to be NP-Complete (Garey et al., 1976), while the Maximum Cut (and the weighted variant) are NP-Hard (Woeginger, 2003).

A d -cut is a generalization of matching-cut. Instead of limiting the maximum number of neighbors in the other partition to 1, it is limited to a parameter d . The d -cut problem is the corresponding decision problem that aims to determine if a particular graph has an d -cut. Similarly, the *Maximum d -Cut Problem* consists of determining a partition that maximizes the size of the cut, in the same fashion as the maximum cut problem.

Regarding the complexity of the d -Cut Problem and the *Maximum d -Cut Problem* there are instances where one of them or both problems can be solved in polynomial time. For example, if the graph is complete with size over $2d$, it doesn't admit a d -cut, and if the graph has at least one vertex whose degree is equal to or below d , separating this vertex from the rest of the graph is a valid d -cut. Still, for the general case, it's not trivial to determine whether a graph has a cut and what that cut would be (Gomes, 2019).

As a problem of recent study, there's little research on the matter. It is known, for the generalization, that for graphs whose maximum degree $\delta \leq d + 2$, the problem admits a polynomial time solution, and for those whose $\delta \geq 2d + 2$, the problem is NP-complete (Gomes and Sau, 2019). For $d + 2 < \delta < 2d + 2$, the complexity of the problem is undiscovered, so far. When setting $d = 1$ for comparison with the matching cut, a similar problem with more research, we can see this unknown region is nonexistent, as $\delta \in \mathbb{N}$

The current fastest algorithm for the Maximum Cut Problem was proposed by Williams (Williams, 2005), with a running time of approximately $O^*(1.74^n)$. This algorithm, though, has an exponential space complexity, and it's currently unknown whether a polynomial space algorithm with a time complexity of $O^*(1.99^n)$ exists for the general graph (Woeginger, 2008).

Our objective in this work is to tackle a variation of the d -cut, intersecting it with the Maximum Cut problem. We not only want to answer whether a graph has a d -cut, we want to find the cut that maximizes the sum of weights while still respecting the constraint enforced by the parameter d .

The remainder of this paper is organized as follows: Section 2 presents some basic definitions related to the problem. Section 3 presents mixed integer formulations to the problem. And, Section 4 presents an initial result regarding the presented formulations.

2. Problem Definition

Given a graph $G = (V, E)$, edges with weights w in its edges, define an d -cut as a partition $V(G)$, $A \cap B = \emptyset$, $A \cup B = V$ such that $\forall v \in A, d_B(v) \leq d$, and $\forall v \in B, d_A(v) \leq d$, where $d_X(v)$ is the number of neighbors of the vertex v in the partition X . Furthermore, define the weight of a d -cut (A, B) as the sum of the edges that has exactly one endpoint in each set, more formally, let (A, B) be a partition of G , its weights $w(A, B)$ is given by $\sum_{v \in A} \sum_{u \in N(v) \cap B} w_{vu}$.

A subgraph $H = (V', E')$ from a graph $G = (V, E)$ is a graph formed by taking a subset of the edges and vertices of the original graph G . That means that every edge and vertex in H also exists in G . An induced subgraph, on the other hand, is a special type of subgraph. For an induced subgraph $H = (V', E')$ of a graph $G = (V, E)$, $\forall v, u \in V' ((v, u) \in E \Rightarrow (v, u) \in E')$. So for an induced subgraph, if it contains 2 vertices who are neighbors in the original graph, they should also be neighbors in the induced subgraph.

The *Maximum d -Cut Problem* can be defined as, given a weighted graph $G = (V, E)$ with weights w in its edges, determine a d -cut of maximum weight. Note that this problem is quite similar to the *Maximum Cut Problem*.

3. Some Mathematical Models

The Maximum d -Cut Problem can be easily modeled by a quadratic model. Define variables binary variables x_v^A and x_v^B , which represent whether a vertex is in the partition A or B , respectively. To be more precise, $x_v^A = 1$ if and only v belong to the partition A and 0 if it does not. Similarly, $x_v^B = 1$ if and only v belong to the partition B and 0 if it does not.

Using this variables we can model the problem as presented below:

$$\max \sum_{uv \in E} w_{uv} (x_u^A x_v^B + x_u^B x_v^A) \quad (1)$$

$$st \sum_{u \in N(v)} x_u^A \leq n_v - (n_v - d)x_v^B \quad \forall v \in V(G) \quad (2)$$

$$\sum_{u \in N(v)} x_u^B \leq n_v - (n_v - d)x_v^A \quad \forall v \in V(G) \quad (3)$$

$$x_v^A + x_v^B = 1 \quad \forall v \in V(G) \quad (4)$$

$$x_v^A, x_v^B \in \{0, 1\} \quad \forall v \in V(G) \quad (5)$$

The objective function (1) is just what one might expect, since we maximize the value of the edges crossing through the partitions; Constraints (2) and Constraints (3) forces that for each vertex, it only has at most d neighbors in the partition that is not the one its os in; Constraints (4) defines that the solution does induces an partition on the vertex set since each vertex has to be in one of the two partitions and finally, Constraints (5) defines that the variables are binary.

Notice that equation (4) represents that the solution induces an partition and therefore each vertex must be in exactly one set of the partition. Therefore, there is no need to keep two

variables to register the partition of each vertex. Thus, we can replace our two set of binary variables by a single binary variable x_v that is equal to 1 if $v \in A$, and 0 if $v \in B$.

One can easily adapt the model provide using the two variables to a new model for the variable described before by just applying the substitutions $x_v = x_v^A$ and $x_v^B = 1 - x_v^A$. For completeness, we can describe the model obtained by just transformations in what follows:

$$\max \sum_{u,v \in E} w_{uv}(x_u + x_v - 2x_u x_v) \quad (6)$$

$$st \sum_{u \in N(v)} x_u \leq d - (n_v - d)x_v \quad \forall v \in V(G) \quad (7)$$

$$\sum_{u \in N(v)} x_u \geq (n_v - d)x_v \quad \forall v \in V(G) \quad (8)$$

$$x_v \in \{0, 1\} \quad \forall v \in V(G) \quad (9)$$

The objective function (6) is similar to (1); Constraints (7), Constraints (8) and Constraints (9) are the same constraints that appears in (2), (3) and (5), respectively.

Although the model represented by Constraints (7) to (9) is an integer programming, the objective function ((6)) is a quadratic expression, thus turning this model into a quadratic programming problem. Nevertheless, we can linearize the objective function to explore it through the realm of pure binary integer programming. To do so, we must add a binary variable $y_{v,u}$ to represent the multiplication of $x_u x_v$ and add the necessary constraints to ensure that $y_{v,u} = x_u x_v$ to reach a complete binary linear formulation of the problem.

$$\max \sum_{u,v \in E} w_{uv}(x_u + x_v - 2y_{uv}) \quad (10)$$

$$st \sum_{u \in N(v)} x_u \leq d - (n_v - d)x_v \quad \forall v \in V(G) \quad (11)$$

$$\sum_{u \in N(v)} x_u \geq (n_v - d)x_v \quad \forall v \in V(G) \quad (12)$$

$$y_{v,u} \geq x_v + x_u - 1 \quad \forall v, u \in V(G) \quad (13)$$

$$y_{v,u} \leq x_v \quad \forall v, u \in V(G) \quad (14)$$

$$y_{v,u} \leq x_u \quad \forall v, u \in V(G) \quad (15)$$

$$x_v \in \{0, 1\} \quad \forall v \in V(G) \quad (16)$$

$$y_{v,u} \in \{0, 1\} \quad \forall v \in V(G) \quad (17)$$

The objective function(10) is the same expression as (6); Constraints (11), Constraints (12) and Constraints (16) are the same constraints that appears in (2), (3) and (5), respectively. And finally, Constraints (13),Constraints (14), Constraints (15), and Constraints (17) represent the linearization of of the product $x_u x_v$.

As a last observation, we point out that the presented models so far had, as the objective function, a maximization of the sum of weights from the edges that cross the cut. An equivalent definition, though, can be achieved by using the complementary objective - minimize the sum of weights from edges that are internal to both partitions.

$$\min \sum_{u,v \in E} w_{uv}(2y_{v,u} - x_v - x_u) \quad (18)$$

$$st \sum_{u \in N(v)} x_u \leq d - (n_v - d)x_v \quad \forall v \in V(G) \quad (19)$$

$$\sum_{u \in N(v)} x_u \geq (n_v - d)x_v \quad \forall v \in V(G) \quad (20)$$

$$y_{v,u} \geq x_v + x_u - 1 \quad \forall v, u \in V(G) \quad (21)$$

$$y_{v,u} \leq x_v \quad \forall v, u \in V(G) \quad (22)$$

$$y_{v,u} \leq x_u \quad \forall v, u \in V(G) \quad (23)$$

$$x_v \in \{0, 1\} \quad \forall v \in V(G) \quad (24)$$

$$y_{v,u} \in \{0, 1\} \quad \forall v \in V(G) \quad (25)$$

4. On Indivisible Subgraphs

When analyzing the constraints of our models, one aspect becomes apparent: the constraints that account for the x_v of two vertices only exist if these two vertices have an edge between them. Therefore, adding an edge to a graph can only add a new constraint; it can never alter or remove any existing ones.

This observation is important, as it leads to the use of the already defined concept of a subgraph. Having a valid cut is not a hereditary property, as the addition of a new vertex into a graph can make the existing cuts invalid, depending on this new vertex and previous characteristics, as well as create new valid cuts, since if no valid d -cut exists, that means that one can't make two non-empty partitions, so one of the partitions can be the original graph, while the other is the added vertex, and a valid cut is generated.

While having a cut is not a hereditary property, the ways in which you cannot partition the existing vertices are. Let H be a subgraph of G . For any valid d -cut in G , if we look at the distribution of the vertices from H in this partitioning of G , the same partitioning must be a d -cut for the graph H since it respects the constraint enforced by the parameter d in the graph H . So we can look at the reversed statement: *if an attempt to divide the vertices in the subgraph H is not valid, it also will not be valid in the graph G .*

Let H be a subgraph of G that admits no valid d -cut. Adding edges between the vertices of H cannot create a valid d -cut, as one would need to remove the existing constraint, not add new ones. Alternatively, adding a vertex or a set of vertices could create a valid cut, but all the original vertices should be in the same partition in this cut, as already mentioned. Therefore, any valid d -cut in G will require all the vertices from H to be in the same partition.

So the first approach to tackle this problem and simplify the models can be to find indivisible H_i subgraphs of G , $\forall v \in H_i, x_v = c_i$.

4.1. Main Result

Theorem 1. *Let $K \subseteq V(G)$ be a clique of size greater than d and Q be a set of vertices whose neighborhood intersection contains such a clique, $K \subseteq \cap_{q \in Q} N(q)$. The graph $G[K \cup Q]$ induced by the set $K \cup Q$ admits a d -cut if and only if $|K \cup Q| \leq 2d$.*

Proof. First, notice that as $|K| > d$, every cut must divide the vertices of K into the two partitions. That is because every vertex in Q has an edge to every vertex in K , and if $|K| > d$, then all the vertices in Q in the other partition would trespass the restriction imposed by d .

Assume that $|K \cup Q| = 2d + 1$, the smallest union possible that upholds the theorem constraints. For that, we can define a variable c , such that $|Q| = c$ and so $|K| = 2d + 1 - c$, as by definition of the sets K and Q , $K \cap Q = \emptyset$. We divide the vertex in the partitions A and B , following the constraint that no partition can hold more than d vertices from K or Q separately.

So we have x , the number of vertices from K in the partition A , and y , the number of vertices from Q in the partition A . So we have the following system of equations:

$$\begin{aligned} x &\leq d \\ 2d + 1 - c - x &\leq d \\ 0 &\leq y \leq c \\ c &\leq d \end{aligned}$$

Each vertex in K has an edge to every other vertex in the union of the sets. That means we can define how many neighbors each vertex in K has in the other partition, which is the size of the other partition. So for those in A , they have $2d + 1 - c - x + c - y$, and for those in B , they have $x + y$ neighbors. So we can add these to the system.

$$\begin{aligned} x + y &\leq d \\ 2d + 1 - x - y &\leq d \end{aligned}$$

Which can be presented as

$$d + 1 \leq x + y \leq d$$

An impossible system. Since no partition would be able to respect the restriction imposed by d , the set $K \cup Q$ cannot admit an d -cut.

This set, once found, can be extended. Since it does not admit a d -cut, all of its vertices must be in the partition in a d -cut for the graph G . Any vertex v in the graph with neighborhood $N(v)$ whose $|N(v) \cap (K \cup Q)| > d$ must also be in the same partition to respect the constraint imposed by the parameter d . It must be in the same partition, and it can also be included in the indivisible set, allowing an iterative extension process. $\square$

We remark that Theorem 1 provides a manner to reduce the size of the problem by selecting subsets of vertices that can must be in the same set regarding a d -cut. More precisely, let K and Q be two sets of vertices fulfilling the requirements of Theorem 1. Notice that, as such vertices must be in the same partition (if any exists) one can force all the variables associated to this vertices to be equal among themselves.

5. Conclusion

The proposed approach to reduce the size of the problem through finding indivisible subsets of vertices can be very effective if many instances are found, or if these instances does not admits a d -cut, some subsets can be extended to many vertices. Nevertheless, it is important to notice that for this approach to be fully applicable it is necessary an algorithm to find a subgraph $K \cup Q$ for each Theorem 1 holds (of course, for the chosen d value). Finding these structures is unwarranted, though, so for graphs without the indivisible structure, solving the model only through linear programming should be very costly.

The linear model presented here and the simplification provided by Theorem 1 stand to be tested against random graphs to check for their effectiveness in solving the problem for the

Maximum Cut that respects the d -cut constraint. Also, due to being a recent topic of study, the d -cut problem and this maximum cut variation lack a benchmark of hard instances for the testing of possible solutions. It remains as an open challenge the creation of these benchmark instances and also the testing of the approach presented in this article against these benchmarks.

References

- Chvátal, V. (1984). Recognizing decomposable graphs. *Journal of Graph Theory*, 8(1):51–53. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.3190080106>.
- de Sousa, S., Haxhimusa, Y., and Kropatsch, W. G. (2013). Estimation of distribution algorithm for the max-cut problem. In Kropatsch, W. G., Artner, N. M., Haxhimusa, Y., and Jiang, X., editors, *Graph-Based Representations in Pattern Recognition*, p. 244–253, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-642-38221-5.
- Garey, M., Johnson, D., and Stockmeyer, L. (1976). Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267. ISSN 0304-3975. URL <https://www.sciencedirect.com/science/article/pii/0304397576900591>.
- Gomes, G. C. M. (2019). *Algoritmos, parâmetros e complexidade para problemas de partições em grafos*. Doutorado em computação, ICX - Departamento de ciência da computação, Universidade Federal de Minas Gerais, Belo Horizonte.
- Gomes, G. C. M. and Sau, I. (2019). Finding cuts of bounded degree: complexity, FPT and exact algorithms, and kernelization. *CoRR*, abs/1905.03134. URL <http://arxiv.org/abs/1905.03134>.
- Graham, R. L. (1970). On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170–186.
- Harary, F. (1959). On the measurement of structural balance. *Behavioral Science*, 4(4):316–323. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/bs.3830040405>.
- Williams, R. (2005). A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365. ISSN 0304-3975. URL <https://www.sciencedirect.com/science/article/pii/S0304397505005438>. Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004).
- Woeginger, G. J. (2003). *Exact Algorithms for NP-Hard Problems: A Survey*, p. 185–207. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-36478-8. URL https://doi.org/10.1007/3-540-36478-1_17.
- Woeginger, G. J. (2008). Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405. ISSN 0166-218X. URL <https://www.sciencedirect.com/science/article/pii/S0166218X0700128X>. Combinatorial Optimization 2004.