Robust greedy algorithms for the Max-Cut relaxation

Bernardo Gonzalez Torres, Angel Eduardo Rodriguez Fernandez, Ricardo Menchaca Mendez

Abstract—A classic result by Goemans and Williamson for the maximum cut (Max-Cut) problem is their randomized approximation algorithm [1] which guarantees to deliver a solution of expected value at least 0.87856 times the optimal value. This algorithm use a simple and elegant technique that randomly rounds the solution to a semidefinite programming relaxation. Solving this relaxation typically requires to use interior point methods or other sophisticated algorithms. We present a simple but robust greedy algorithm to solve this relaxation, with time complexity $O(mn)$ and space complexity $O(n^2)$ (which is the minimum required to store the solution, assuming the solution is full rank), where n is the number of nodes and m is the number of edges of the input. Additionally, we extend our result to the Max-k-Cut problem for $k > 2$, with minimal modification to the main algorithm. We provide analysis and extensive experimentation that supports our results.

Index Terms—Max-Cut, Max-k-Cut, Robust, Greedy.

1 Introduction

The introduction goes here. ??.
??.

The main contributions of this paper are:

1) **Novel algorithms**. We present robust greedy iterative algorithms to solve the Max-k-Cut relaxations for $k \geq 2$, with time complexities $O(mn)$ for $k = 2$ and $O(n^3)$ for $k > 2$, and space complexity $O(n^2)$ (Algorithms 1 and 2).
2) **Analysis**. We provide an analysis of the algorithms proposed.
3) **Empirical Results**. We provide an exhaustive set of empirical results in support of our results. An implementation of our algorithm can be found in [LINK]. In addition, we also prepared a google colab notebook where our results can be replicated [LINK].

2 Related Work

The related work goes here.

3 Preliminaries

We made the following assumptions about the input graph G:

- No self-loops in G ($W_{i,i} = 0 \; \forall i$)
- No parallel edges in G
- $W_{i,j} \in \mathbb{R} \; \forall i, j$

We use the following notation through the remaining of the text: $\mathbb{I}$ denotes the identity matrix of size $(n \times n)$, $\|V_i\|$ denotes the Euclidean norm of vector V_i, and V_i^t indicates the value of vector V_i at time t.

4 Algorithm ($k = 2$)

Our main result is algorithm 1. The first key observation behind this algorithm is to observe that the partial derivative of the function to optimize $f(V)$ with respect to vector V_i is:

$$\frac{\partial f(V)}{\partial V_i} = -\frac{k-1}{k} \sum_j W_{ij} V_j$$

The second key observation behind our algorithm is to realize that updating the vector V_i as our algorithm does is (greedy) optimal for vector V_i (given the other vectors are fixed) and always improves (or at least doesn't degrade) the value of $f(V)$. We prove this in section 5.

Algorithm 1: Main algorithm ($k = 2$)

```
Input: Edge weights $W$ and initial output $V = \mathbb{I}$
Output: $V$
while Convergence criterion is not satisfied do
    for i from 1 to n do
        Update V_i:
            $V_i^{t+1} \leftarrow -\frac{\sum_j W_{i,j} V_j^t}{\|\sum_j W_{i,j} V_j^t\|}$
    Evaluate convergence.
return $V$
```

5 Analysis ($k = 2$)

As can be seen from algorithm 1, the inner *for* loop of our algorithm performs $O(mn)$ operations. To see this, observe that every edge weight $W_{i,j}$ is only used two times inside the *for* loop, one while updating vector V_i and the second one while updating vector V_j. Given that we have m edges and the size of every vector V_i is equal to n, this results in an $O(mn)$ time complexity. Is important to mention that a naive implementation of algorithm 1 may have an $O(n^3)$ complexity if the sum is taken over all possible values of j (from 1 to n) and not only over the neighbors of node i.

Similarly, it can be observed from algorithm 1 that not extra memory is needed besides the input W and output V, therefore having an spatial complexity of $O(n^2)$.

An important point to mention is that the *for* loop does not to be ordered: we can substitute it by any permutation of the set of indices.

In the rest of the text, the notation $f(V|V_i^{t+1})$ denotes the value of the function $f()$ evaluated at V when the i-th row of V is equal to V_i^{t+1}.

Theorem 5.1. *For any symmetric matrix $W \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times n}$ s.t. $\|V_i\| = 1\ \forall i$, any update performed by algorithm 1 implies a difference $f(V|V_i^{t+1}) - f(V|V_i^t)$ equal to:*

$$\frac{k-1}{k}s(1-\cos\theta) \geq 0 \tag{1}$$

where $s = \|\sum_j W_{i,j}V_j^t\|$ and θ is the angle between V_i^t and $\sum_j W_{i,j}V_j^t \propto V_i^{t+1}$.

Proof.

$$\begin{aligned}
&f(V|V_i^{t+1}) - f(V|V_i^t)\\
&= \frac{k-1}{k}\sum_{i<j} W_{i,j}(1 - V_i^{t+1}\cdot V_j^{t+1})\\
&- \frac{k-1}{k}\sum_{i<j} W_{i,j}(1 - V_i^{t}\cdot V_j^{t})\\
&= \frac{k-1}{k}\left(-\sum_{i<j} W_{i,j}V_i^{t+1}\cdot V_j^{t+1} + \sum_{i<j} W_{i,j}V_i^{t}\cdot V_j^{t}\right)\\
&= \frac{k-1}{k}\left(-\sum_{j} W_{i,j}V_i^{t+1}\cdot V_j^{t} + \sum_{j} W_{i,j}V_i^{t}\cdot V_j^{t}\right)\\
&= \frac{k-1}{k}\left(-V_i^{t+1}\cdot(\sum_{j} W_{i,j}V_j^{t}) + V_i^{t}\cdot(\sum_{j} W_{i,j}V_j^{t})\right)\\
&= \frac{k-1}{k}\left(-V_i^{t+1}\cdot(-sV_i^{t+1}) + V_i^{t}\cdot(-sV_i^{t+1})\right)\\
&= \frac{k-1}{k}s\left(\|V_i^{t+1}\|^2 - V_i^{t}\cdot V_i^{t+1}\right)\\
&= \frac{k-1}{k}s(1-\cos\theta) \geq 0
\end{aligned}$$

where the third equality is because only the i-th row of V was modified (greediness of the algorithm) and the fifth equality is implied by the update made by algorithm 1 to V_i. □

5.1 Lagrangian ($k = 2$)

For $k = 2$, the Lagrangian of the optimization problem is:

$$\frac{k-1}{k}\sum_{i<j} W_{ij}(1 - V_i\cdot V_j) + \sum_i \lambda_i(1 - \|V_i\|^2)$$

Taking the partial derivative of the Lagrangian with respect to V_i and setting equal to zero gives:

$$V_i^{t+1} = -\frac{k-1}{2k\lambda_i}\sum_j W_{i,j}V_j$$

from here, we can see that updating V_i as our algorithm does is optimal for vector V_i (given the other vectors are fixed), and the corresponding value of the Lagrange multiplier λ_i is:

$$\lambda_i = \frac{k-1}{2k}\|\sum_j W_{i,j}V_j^t\|$$

6 Experiments ($k = 2$)

Brief description of the B graphs.

Brief description of the G graphs.

Mention of commercial solvers (mosek, gurobi, etc.).

Table with results.

Plots of convergence.

Brief explanation of results.

2 orders of magnitude faster.

7 Max-k-Cut

Describe the Max-k-Cut problem, the relaxation we are working with, the corresponding guarantees, etc.

7.1 Algorithm ($k > 2$)

Our proposed algorithm for the Max-k-Cut relaxation is algorithm 2. As can be observed, the algorithm is practically the same algorithm as algorithm 1, but with a main modification: we need to check that an updated vector V_i^{t+1} does not violate any of the new restrictions.

Algorithm 2: Second algorithm ($k > 2$)

```
Input: Edge weights $W$, number of cuts $k$, and initial
       output $V = \mathbb{I}$
Output: $V$
while Convergence criterion is not satisfied do
    for i from 1 to n do
        Update $\mathbf{V_i}$:
            $V_i^{t+1} \leftarrow -\frac{\sum_j W_{i,j}V_j^t}{\|\sum_j W_{i,j}V_j^t\|}$
        Evaluate restrictions. Execute subroutine 1 for
        candidate vector $V_i^{t+1}$.
    Evaluate convergence.
return $V$
```

Algorithm 3: Subroutine 1

```
Input: Edge weights $W$, current output $V$ and
       current candidate vector $V_i^{t+1}$
Output: $c$
Set $c \leftarrow V_i^{t+1}$
Compute $c \cdot V_j\ \forall j$
Set $r \leftarrow 1$
while Any $c \cdot V_j < -\frac{1}{k-1}$ do
    Set $r \leftarrow \frac{r}{2}$
    Set $c \leftarrow rV_i^{t+1} + (1-r)V_i^t$
    Set $p \leftarrow \|c\|$
    Set $c \leftarrow \frac{1}{p}c$
return $c$
```

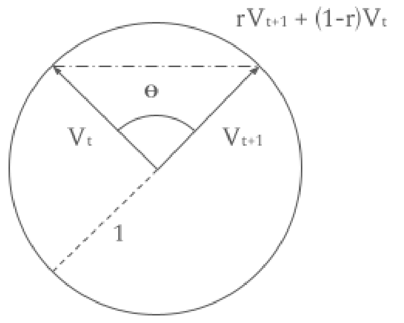

Fig. 1. Example: diagram of vectors V_i^t, V_i^{t+1}, interpolation $rV_i^{t+1} + (1-r)V_i^t$ and angle θ

Subroutine 1 is a very simple search procedure: if the candidate vector c violates any restriction, we move it closer to the previous vector V_i^t through the interpolation $rV_i^{t+1} + (1-r)V_i^t$ (see fig. 1), which starts with $r = 1$, then $r = \frac{1}{2}$, then $r = \frac{1}{4}$, and so on, hoping that the new candidate does not violate any restriction. As we can imagine, the new candidate does not always (actually never, except when $r = 0$ or $r = 1$) has unit norm, so we compute its norm $p = \|c\|$ and normalize every new candidate. In our actual implementation, in order to prevent long searches, we set a small number l such that, if the *while* loop in subroutine 1 is repeated more than l times, we simply set $c = V_i^t$.

7.2 Analysis ($k > 2$)

As can be seen from algorithm 2, now the inner *for* loop of the algorithm performs $O(n^3)$ operations (because the subroutine 1 performs $O(n^2)$ operations). This is the cost to pay for constantly checking if the restrictions has not been violated. Also, as can be observed from algorithm 2, the spatial complexity remains as $O(n^2)$. Keeping this spatial complexity "low" is not a minor issue. At the best of the authors knowledge, trying to solve the Max-k-Cut relaxation with commercial solvers for graphs of moderate size or larger (like the ones we are experiment with) is problematic due to memory issues.

The proof of the following theorem is provided in section A of the Appendix.

Theorem 7.1. *For any symmetric matrix $W \in \mathbb{R}^{n\times n}$, $k \geq 2$, $r \in [0,1]$ and $V \in \mathbb{R}^{n\times n}$ s.t. $\|V_i\| = 1\ \forall i$, any update performed by algorithm 2 implies a difference $f(V|c) - f(V|V_i^t)$ equal to:*

$$\frac{k-1}{k} s \left[\frac{1-p}{p}\cos\theta + \frac{r}{p}(1-\cos\theta)\right] \geq 0 \qquad (2)$$

where $s = \|\sum_j W_{i,j}V_j^t\|$, θ is the angle between V_i^t and $\sum_j W_{i,j}V_j^t$, r is the final interpolation value of subroutine 1 and $p(r,\theta) = \|c\| = \sqrt{1 + 4r(r-1)\sin^2\frac{\theta}{2}}$

Observe that, for $k = 2$, r is always equal to one, and $p(r,\theta) = 1\ \forall\theta \in [0,\pi]$ (this can also be seen from figure 1), so the result from theorem 7.1 is the same as the result from theorem 5.1 for $k = 2$.

7.3 Lagrangian ($k > 2$)

For $k \geq 2$, the Lagrangian of the optimization problem is:

$$\frac{k-1}{k}\sum_{i<j} W_{ij}(1 - V_i\cdot V_j) + \sum_i \lambda_i(1 - \|V_i\|^2) + \sum_{i,j}\beta_{i,j}\left(\frac{1}{k-1} + V_i\cdot V_j\right)$$

Taking the partial derivative of the Lagrangian with respect to V_i and setting equal to zero gives:

$$V_i^{t+1} = \frac{1}{2\lambda_i}\left(\sum_j \left(\beta_{i,j} - \frac{k-1}{k}W_{i,j}\right)V_j\right)$$

Unlike the case when $k = 2$, now is hard to tell if updating V_i as our algorithm does is optimal due to the presence of the Lagrange multipliers $\beta_{i,j}$ (although we do know that every update always improves or at least does not degrade the current solution, as can be seen from theorem 7.1).

7.4 Experiments ($k > 2$)

Mention of commercial solvers (mosek, gurobi, etc.) memory issue.

Table with results.

Brief explanation of results.

Table with results.

Plots of convergence.

Brief explanation of results.

mention that robust properties are (at least theoretically) also extended to this algorithm.

8 Robustness of the algorithm

Two interesting properties can be observed from the proofs of the algorithms:

1) As long as $\|V_i\| = 1\ \forall i$ and the V_i's are not unidirectional (pointing in the same direction), any initialization for V will converge to the optimum.
2) For any positive r, as long as we do not violate any restriction, the algorithm will converge to the optimum.

In other words, the algorithm is robust to any valid choices of r and initial V. In order to observe this, we performed the following experiments for $k = 2$:

- Random initial matrix V where each $V_{i,j}$ is sampled from a Gaussian distribution with mean zero and variance one, and each V_i is normalized. Additionally, we choose a random value of r from a uniform distribution at each step of the algorithm. *There is a possible explanation for the good results in this experiment: given the high dimensionality of the problem and the way we are sampling the entries $V_{i,j}$, it is a known fact that $V_i \cdot V_j = 0$ for any $i \neq j$ with high probability, therefore the initialization is the same as the identity matrix but rotated with high probability.*
- "Adversarial" initial matrix V, random value of r from a uniform distribution at each step of the algorithm: for

a small graph where we already know the solution, we initialized V in a way that the corresponding vectors V_i of nodes that should be in the same cut are in opposite sides of the hyper-sphere (when possible), and the corresponding vectors V_i of nodes that should be in different cuts are the same vector (when possible).

The results of these experiments can be found in section B of the Appendix.

9 Conclusions

We present simple greedy iterative algorithms to solve the Max-k-Cut relaxations for $k \geq 2$, with time complexities $O(mn)$ for $k = 2$, $O(n^3)$ for $k > 2$, and space complexity $O(n^2)$. Both algorithms are robust to initial values of V and interpolation values of r. We provide an analysis of our algorithms and extensive experimentation that supports our results.

9.1 Future research

There are multiple directions to extend or improve this work:

- ?
- ...

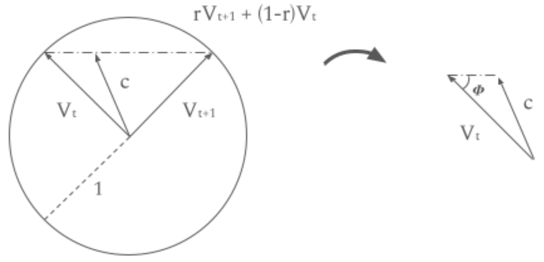

Fig. 2. Example: triangle formed by vectors V_i^t, $c = rV_i^{t+1} + (1-r)V_i^t$ for a specific value of r, and angle $\phi = \frac{\pi-\theta}{2}$

Appendix A Proofs

The following lemmas will be useful to prove theorem 7.1.

Lemma A.1. *For any unitary vectors $V_i^{t+1} \in \mathbb{R}^n$, $V_i^t \in \mathbb{R}^n$, angle $\theta \in [0,\pi]$, and a fixed $r \in \mathbb{R}$, the norm $p = \|c\|$ of the interpolation $c = rV_i^{t+1} + (1-r)V_i^t$ is equal to:*

$$p(r,\theta) = \|c\| = \sqrt{1 + 4r(r-1)\sin^2\frac{\theta}{2}}$$

Proof. From figure 1, we can observe that the vectors V_i^{t+1} and V_i^t, and the path of the interpolation $rV_i^{t+1} + (1-r)V_i^t$ for $r \in [0,1]$, form a triangle. The two sides corresponding to the vectors V_i^{t+1} and V_i^t of this triangle has length equal to one, and by simple geometry it can be shown that the length of the other side is equal to $2\sin\frac{\theta}{2}$. With this in mind and applying the law of cosines to the triangle formed by vector V_i^t, vector $c = rV_i^{t+1} + (1-r)V_i^t$ for a specific value of r and angle $\phi = \frac{\pi-\theta}{2}$ (example shown in figure 2), we can compute p as:

$$\begin{aligned} p &= \sqrt{\|V_i^t\|^2 + \left(2r\sin\frac{\theta}{2}\right)^2 - 4r\|V_i^t\|\sin\frac{\theta}{2}\cos\phi} \\ &= \sqrt{1 + 4r^2\sin^2\frac{\theta}{2} - 4r\sin\frac{\theta}{2}\cos\frac{\pi-\theta}{2}} \\ &= \sqrt{1 + 4r^2\sin^2\frac{\theta}{2} - 4r\sin\frac{\theta}{2}\sin\frac{\theta}{2}} \\ &= \sqrt{1 + 4r^2\sin^2\frac{\theta}{2} - 4r\sin^2\frac{\theta}{2}} \\ &= \sqrt{1 + 4r(r-1)\sin^2\frac{\theta}{2}} \end{aligned}$$

□

Before continuing with the next lemma, it will be useful to compute the partial derivative of $p(r,\theta)$ with respect to r:

$$\begin{aligned} \frac{\partial p(r,\theta)}{\partial r} &= \frac{1}{2p(r,\theta)}(8r-4)\sin^2\frac{\theta}{2} \\ &= \frac{2}{p(r,\theta)}(2r-1)\sin^2\frac{\theta}{2} \\ &= \frac{1}{p(r,\theta)}(2r-1)(1-\cos\theta) \end{aligned}$$

We can observe from this equality that for a fixed value of θ, the minimum value of p is reached when $r = \frac{1}{2}$, as can be observed in figures 1 and 2.

Lemma A.2. *For any $r \in [0,1]$, $\theta \in [0,\pi]$ and $p(r,\theta) = \sqrt{1 + 4r(r-1)\sin^2\frac{\theta}{2}}$, the function:*

$$g(r,\theta) = \frac{1-p}{p}\cos\theta + \frac{r}{p}(1-\cos\theta)$$

is always greater or equal to zero.

Proof. We will follow the next logic to prove the lemma: firstly, we prove that for $r = 0$ the function $g(r,\theta) = 0$. This is easy to prove: from figures 1 and 2, and lemma A.1, we can observe that when $r = 0$, $p(r,\theta) = 1\ \forall\theta \in [0,\pi]$. Due this, the value of $g(r,\theta) = 0$ for $r = 0$, $\forall\theta \in [0,\pi]$.

Secondly, we prove that for $r \in [0,1]$, the partial derivative of $g(r,\theta)$ with respect to r is always non-negative. For clarity and to save space, we will omit the dependence of $p(r,\theta)$ and only write p:

$$\begin{aligned} \frac{\partial g(r,\theta)}{\partial r} &= \frac{1-\cos\theta}{p^2}(p - (2r-1)\cos\theta) \\ &\quad - \frac{1-\cos\theta}{p^2}(2r-1)g(r,\theta) \\ &= \frac{1-\cos\theta}{p^2}(p - (2r-1)(g(r,\theta) + \cos\theta)) \\ &= \frac{1-\cos\theta}{p^2}\left(p - (2r-1)\left(\frac{1}{p}(\cos\theta + r(1-\cos\theta))\right)\right) \\ &= \frac{1-\cos\theta}{p^3}\left(p^2 - (2r-1)(\cos\theta + r(1-\cos\theta))\right) \\ &= \frac{1-\cos\theta}{p^3}\left(1 + 4r(r-1)\sin^2\frac{\theta}{2}\right) \\ &\quad - \frac{1-\cos\theta}{p^3}(2r-1)(\cos\theta + r(1-\cos\theta)) \\ &= \frac{1-\cos\theta}{p^3}(1 + 2r(r-1)(1-\cos\theta)) \\ &\quad - \frac{1-\cos\theta}{p^3}(2r-1)(\cos\theta + r(1-\cos\theta)) \\ &= \frac{1-\cos\theta}{p^3}(1-r)(1+\cos\theta) \\ &= \frac{\sin^2\theta}{p^3}(1-r) \end{aligned}$$

where we multiplied by $1 = (\frac{p}{p})$ in the fourth equality. We can observe that for any $\theta \in [0,\pi]$, the partial derivative $\frac{\partial g(r,\theta)}{\partial r}$ is positive for any $r < 1$, equal to zero for $r = 1$, and negative for any $r > 1$. □

From the last proof we can observe the optimality of $r = 1$ in the interpolation $c = rV_i^{t+1} + (1-r)V_i^t$, i.e, when $c = V_i^{t+1} = -\frac{\sum_j W_{i,j}V_j^t}{\|\sum_j W_{i,j}V_j^t\|}$. We already observed this from the Lagrangian in subsection 7.3 when $k = 2$.

A.1 Proof of theorem 7.1

Observe that, after subroutine 1, the updated vector has the form $c = \frac{rV_i^{t+1} + (1-r)V_i^t}{p}$ for a value of $r \in [0,1]$. The proof starts exactly like the proof of theorem 5.1 until the third equality, so we recycle these steps:

Proof.

$$
\begin{aligned}
&f(V|c) - f(V|V_i^t) \\
&= \frac{k-1}{k}\left(-\sum_j W_{i,j} c \cdot V_j^t + \sum_j W_{i,j} V_i^t \cdot V_j^t\right) \\
&= \frac{k-1}{k}\left(-c \cdot (\sum_j W_{i,j} V_j^t) + V_i^t \cdot (\sum_j W_{i,j} V_j^t)\right) \\
&= \frac{k-1}{k}\left(-c \cdot (-sV_i^{t+1}) + V_i^t \cdot (-sV_i^{t+1})\right) \\
&= \frac{k-1}{k} s\left(c \cdot V_i^{t+1} - V_i^t \cdot V_i^{t+1}\right) \\
&= \frac{k-1}{k} s\left(\frac{rV_i^{t+1} + (1-r)V_i^t}{p} \cdot V_i^{t+1} - \cos\theta\right) \\
&= \frac{k-1}{k} s\left(\frac{r}{p} V_i^{t+1} \cdot V_i^{t+1} + \frac{1-r}{p} V_i^t \cdot V_i^{t+1} - \cos\theta\right) \\
&= \frac{k-1}{k} s\left(\frac{r}{p} \|V_i^{t+1}\|^2 + \frac{1-r}{p}\cos\theta - \cos\theta\right) \\
&= \frac{k-1}{k} s\left(\frac{1-p}{p}\cos\theta + \frac{r}{p}(1-\cos\theta)\right) \\
&= \frac{k-1}{k} s g(r,\theta)
\end{aligned}
$$

and due lemma A.2, the theorem is proven. □

Appendix B
Robustness experiments

Robustness experiments (random interpolation value r, random initial V, adversarial initial V) results goes here.

Acknowledgments

The authors would like to thank...

References

[1] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

PLACE PHOTO HERE

Bernardo Gonzalez Torres Biography text here.

PLACE PHOTO HERE

Angel Eduardo Rodriguez Fernandez Biography text here.

PLACE PHOTO HERE

Ricardo Menchaca Mendez Biography text here.