Minimum Cut problem

Luca Zaninotto – 2057074

30 September 2022

1 The problem

The problem we want to solve is to find a cut of a graph such that its weight is minimal. In other words, given a graph

$$G = (V, E)$$

where $V = \{1, 2, 3, \dots, n\}$ a set of n vertices and $E = \{(1, 3), (4, 6), \dots\}$ a set of m edges and a function

$$w : E \to \mathbb{R}$$

a *cost* function, that returns the weight of each node, we want to find

$$p_1, p_2 \subseteq V \mid p_1 \cap p_2 = \emptyset\,,\ p_1 \cup p_2 = V$$

such that

$$\text{cost}(p_1, p_2) = \sum_{u \in p_1, v \in p_2} w((u, v))$$

is minimized.

2 Karger and Stein

Karger and stein works by joining subsets of nodes, until only two remain. Those two are a cut of the graph. More in detail, given a Graph $G = (V, E)$ where $V = \{1, 2, 3, 4, \dots, n\}$ is the set of vertices, the algorithm starts with a partition of V where each node appears by itself

$$\{\{1\}, \{2\}, \{3\}, \dots \{n\}\}$$

Then, let $\mathcal{P}_a(V)$ be the set of partitions of V, a `contraction` procedure joins two subsets and produces a new partition of V.

$$\text{contraction} : \mathcal{P}_a(V) \longrightarrow \mathcal{P}_a(V)$$

This keeps happening, until the partition of V is made of only two sets p_1, p_2. At this stage the sum of the weights that between the two nodes (the cut of the graph) is returned. This works because of the way in which the `contraction` procedure selects the nodes to join in the partitions. The two nodes are selected based on a random selection based on the weight of the nodes. Nodes with an higher weight have an higher probability of being selected.

Since a graph has $2^{n-1} - 1$ possible cuts among which at most $\binom{n}{2}$ are minimum cuts, the probability of picking one of these is at most

$$p_n \geq \binom{n}{2}$$

Therefore by repeating the algorithm

$$T = \binom{n}{2} \log(n)$$

times, the probability of not finding the minimum cut becomes

$$\left[1 - \frac{1}{\binom{n}{2}}\right]^T \leq \frac{1}{e^{\log(n)}} = \frac{1}{n}$$

And so the total running time for T repetitions for a graph with n vertices and m edges is $O(mT) = O(mn^2 \log(n))$

2.1 Results

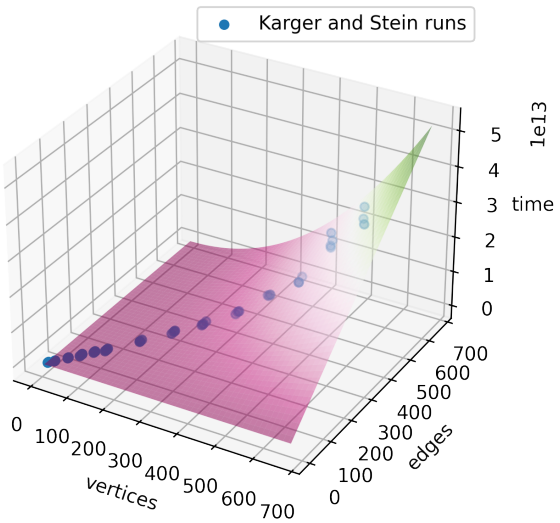

Figure 1: Karger and Stein run times against input size (number of vertices)

The function plotted is $O(mn^2 \log(n))$ that is the overall complexity of the algorithm, the coefficient for the surface (under which all results should fall) is calculated as

$$\max\left(\frac{t}{mn^2 \log(n)}\right)$$

since the time of execution of the algorithm is maximized by this function and the different coefficient depend also on external factors (CPU throttle, memory allocations, etc.) and we're interested the behavior of the algorithm against the input nodes.

3 Stoer and Wagner

Stoer and Wagner works in another way. Instead of relying on random cuts relies on the fact that given a couple of nodes, They either are separated by the minimum cut or are on the same side of the cut. So given a graph $G = (V, E)$, $s, t \in V$, an s, t minimum cut is a cut (S, T) of G s.t.

$$s \in S \vee t \in T$$

$$w(S, T) \text{ is minimum among all } s, t \text{ cuts}$$

So, let (S, T) be a global min-cut for G. For every pair $s, t \in V$ either $s \in S$ and $t \in T$ or s and t are on the same side of the cut.

Stoer and wagner leverages this fact and searches for $s, t \in V$ s.t. (S, T) is a global min cut for G. If that's also a global min-cut its weight is returned, otherwise the global min-cut for $G \backslash \{s, t\}$ (The same graph as G, where the nodes s and t are merged together) is also a global min-cut for G.

3.1 Results

The complexity of this algorithm is

$$O(mn \log(n))$$

where $n = |V|$ and $m = |E|$. so the coefficient for the plot is calculated again as

$$\max\left(\frac{t}{mn \log(n)}\right)$$

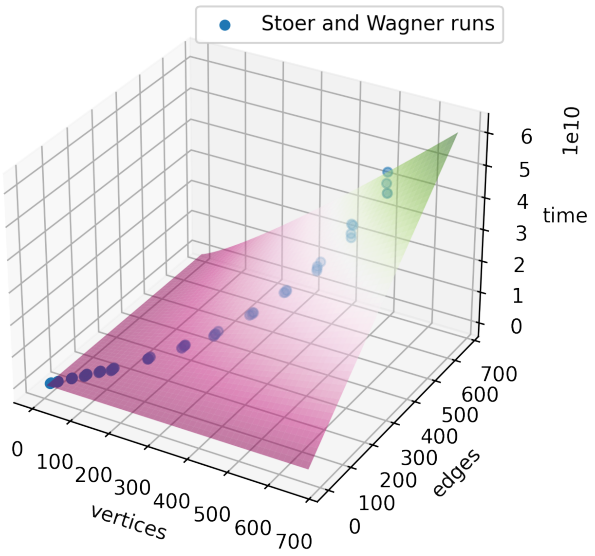

Figure 2: Stoer and Wagner run times against input size (nodes and edges)

4 Hybrid approach

An Hybrid approach consist into merging the two approaches (Karger and Stein and Stoer and Wagner): the algorithm contracts the graph until there are $t = \frac{n}{\sqrt{2}} + 1$ nodes, from there, Stoer and Wagner is run on the contracted graph.

To study the complexity we can take look at the hybrid procedure:

```
def hybrid(graph):
    n = graph.n_vertices
    t = int(np.ceil((n * np.log(n)/ (n-1))))
    amin = np.Inf
    d_time = 0
    for i in range(t):
        cut, d_time = hybrid_iteration(graph)
        if cut < amin:
            amin = cut
            d_time = perf_counter_ns()
    return amin, d_time
```

where the hybrid iteration is described as

```
def hybrid_iteration(graph):
    t = np.ceil(graph.n_vertices / np.sqrt(2) + 1)
```

```
    g = contract(graph, t)
    return stoer_wagner(g)
```

one iteration takes $O(mn\log(n))$, since $t = \frac{n}{\sqrt{2}} + 1$, the contract procedure consists in a single while loop that merges one node per time, running t times, and then apply the Stoer and wagner algorithm, that has complexity $O(mn\log(n))$. Therefore the overall complexity is $O(mn\log(n))$. Since the algorithm is just a matter of iterating the iteration $\frac{n}{n-1}\log(n)$ times (to respect the bound of not cross the minimum cut in the contraction with probability at least $\frac{1}{n}$), the overall complexity is also

$$O(mn\log(n))$$

.

4.1 Results

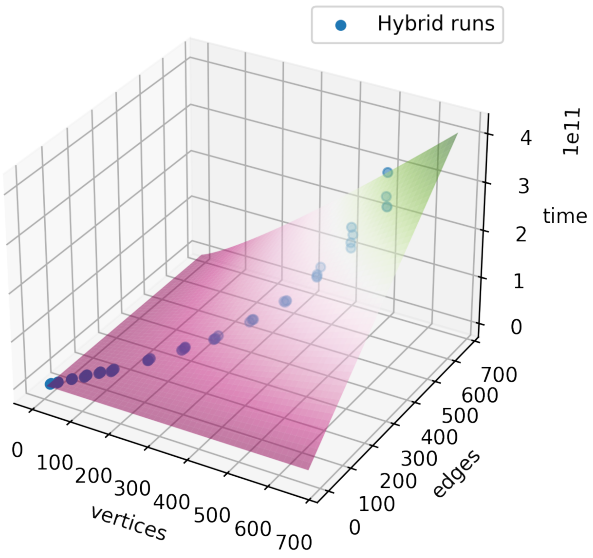

Figure 3: Hybrid run times against input size (nodes and edges)

By plotting the run times against the input size (nodes and edges) we can see that they are all below the surface induced by the complexity function of the algorithm.

5 Results

5.1 On efficiency

By purely relying on the complexity analysis of the algorithms we should be able to see how the Karger and stein algorithm should perform worse than Stoer and Wagner and the Hybrid approach, which have instead comparable complexities. By plotting the run time of each algorithm:

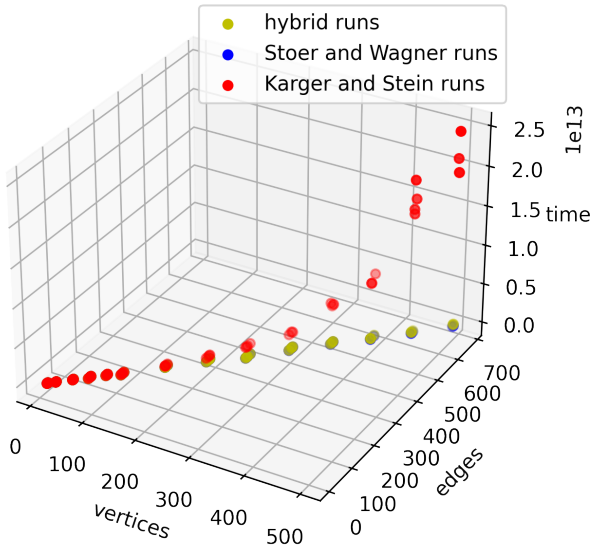

Figure 4: Runtime comparison between the three algorithms

By looking closely to the graph we can see how the results meet our expectations: Karger and Stein perform generally worse than Stoer and Wagner, that performs in a similar way to our approach.

5.2 On discovery time

Discovery time tells us another story. Even though Karger and Stein performs badly compare to the other two algorithms, the discovery time for the graphs in the dataset is not that worse compared to the discovery time of the hybrid algorithm. Stoer and Wagner outperforms the two even from this point of view, showing how is a generally faster algorithm.

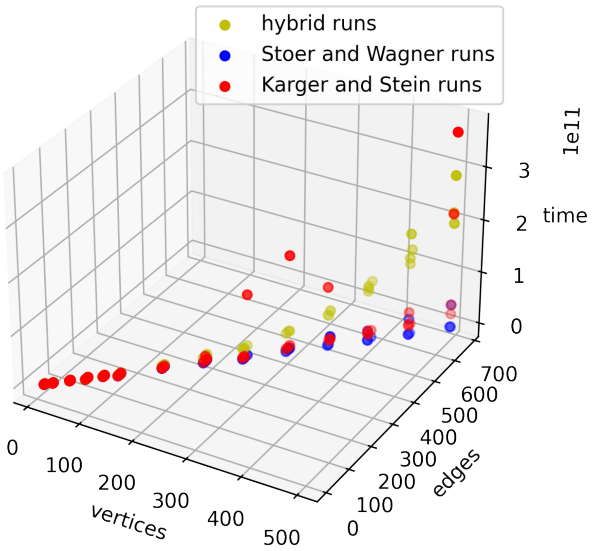

Figure 5: Discovery time comparison between the three algorithms