Algorithm Engineering
WS 21/22

Jakob Sanowski

Comparison of an approximation, heuristic and integer-linear program for the max-cut problem on unweighted, undirected graphs

Abstract

The abstract gives a short summary of the project. Begin by stating the motivation of the research at hand, describe the problem and shortly describe what methods you used to solve this problem. Finally, name the most important findings and provide a brief conclusion of your work.

1 Introduction

2 Preliminaries

The maximum-cut of a graph is the cut which size is at least the size of any other possible cut. A cut is a partition of all vertices into two complementary sets S and T, which size is defined by the number of edges between S and T.

For finding an exact solution we can formulate an integer-linear program ILP. Because the max-cut problem is NP-Hard there do not exist any polynominal-time algorithms for it. There are .. numerous approximation algorithms and heuristics for getting an approximate solution. In the following I will briefly explain the ILP-formulation, 0.5-approximation algorithm and heuristic I used.

2.1 Integer-Linear Program

An Integer-Linear program is a maximazation (or minimazation) of a objective function subject to a number of constraints with some or all of the variables being integers. The objective function and the constraints have to be linear. The maximum-cut problem can be formulated as follows:

Given a graph $G = (V, E)$, let $\{x_v\}_{v \in V}, \{z_e\}_{e \in E} \in \{0, 1\}$ where x_v encodes the partition v belongs to and x_e encodes whether e is part of the cut or not. Then the ILP can be formulated as follows:

$$\begin{aligned} \text{maximise} \sum_{uv \in E} z_{uv} & \qquad (1) \\ \text{subject to: } z_{uv} \leq x_u + x_v, & \qquad (2) \\ z_{uv} \leq 2 - (x_u + x_v) & \qquad (3) \end{aligned}$$

This discribes our problem because z_{uv} can only be 1 if uv is part of the cut. When u and v are in the same partition, $x_u + x_v = 0$ (or 2 respectively). Because of (2) and (3) z_{uv} then only can be 0. Only when u and v are in different partitions $x_u + x_v = 1$ and $z_u v$ can become 1 and thus be part of the cut.

2.2 0.5-Approximation Algorithm

The 0.5-approximation algorithm is a greedy which iteratively inserts each vertex in the partition which improves the cut the most. For each vertex v we count the the number of neighbors in S and T and then insert v in the partition wich contains less neighbors of v in order to increase the cut size as much as possible. Algorithm **??** shows the pseudo-code of this algorithm.

This algorithm finds a 0.5-approximation because if we let C_{v_i} be the cut with all vertex added before v_i and $E_{v_i} = \{(v_i, u) | u \in \{v_1, \ldots, v_{i-1}\}\}$ the edges between v_i and all vertices added before v_i then at least $\frac{1}{2}|E_{v_i}|$ get added to C_{v_i}.

Summing over all $v \in V$ we get:

$$|C| \geq \frac{1}{2} \sum_{v} |E_v|$$

$$|C| \geq \frac{1}{2} |E|$$

```
Input: Graph $G = (V, E)$
Output: Cut $C$
begin
    foreach $v \in V$ do
        if $|neigh(v) \in S| \leq |neigh(v) \notin S|$
         then
            $S = S \cup \{v\}$
        end
    end
    return $(S, S \setminus V)$
end
```

Algorithm 1: 0.5-Approximation Algorithm for Maximum-Cut

3 Algorithm & Implementation

This section provides information about the actually used algorithms and their respective implementations. It should roughly cover the following three topics:

- **Advanced Algorithm:**
  Give and explain the advanced algorithms that you used, and compare them to the basic algorithms.
- **Implementation:**
  Explain how you implemented these algorithms and state what external libraries you used.
- **Algorithm Engineering Concepts:**
  State the algorithm engineering concepts that you used and explain why they were helpful (if applicable).

4 Experimental Evaluation

In this section, the experimental setup is described and the results are presented.

4.1 Data and Hardware

Here, the input data and the applied parameters are described. Further, information about the underlying hardware such as main memory and CPU is provided.

4.2 Results

In this part, the results are presented. This includes comparison of running time and memory usage.
To visualize the results we recommend one of the following graphing tools:

- **mathplotlib:**
  matplotlib is a library for python which allows the direct visualization of the produced results.
- **R:**
  R is a programming language for statistical computations, which provides different graphing possibilities. One of the better known is ggplot2.
- **gnuplot:**
  gnuplot is a command-line plotting program. It can also be used to graph data directly from different programming languages, such as Java and C++.

5 Discussion and Conclusion

In this section, the results are discussed and interpreted. Finally, the work is summarized shortly.

6 References

The references list the external resources used in the work at hand. LaTeX offers special ways to list those resources. In this template the references are stored in the 'refs.bib' file and can be referenced with the '\cite{REF}' command, where REF is a label defined in the .bib file. This example shows how such a reference looks like: **?**.