DMBA - 2023
Final Project

Due date: 11-10-2023

1 Preliminaries

Machine Learning (ML) and Optimization, the two cores of the course, are intertwined in many profound ways. As explained in the course learnings, advances in ML are tightly linked to advances in optimization algorithms. On the other hand new advances in Optimization are being obtained by using ML techniques to speed up existing optimization algorithms.

There is considerable interest in improving heuristic algorithms (HA) for solving Combinatorial Optimization Problems (COPs). HAs rely on handcrafted heuristics that conduct the process of finding the solutions. Although these heuristics work well in many COPs, their success depends on exploiting the nature of problems. Thus they need to be adapted by their human creators to the different problem classes. In this project, we aim to apply ML to learn and improve handcrafted heuristics, enhancing the solution quality.

In this project a more recent approach to solving COPs is explored, namely combining existing methods with ML (or deep learning) to improve the results. Recently, there has been significant progress in ML methods to solve COPs [4]. A popular ML-based method is Deep Reinforcement Learning (DRL): the integration of Reinforcement Learning (RL) and Deep Neural Networks (DNN) [1, 3, 16, 15, 19, 20]. Your task is to use Machine learning (ML) to obtain better solutions than the solution given by a greedy algorithm. Different approaches could be developed to define the final method, and students will choose which venue to explore.

The Max-cut

Your task is to solve the *Max-Cut* problem via ML-based heuristics. Assume you are given a weighted graph $G = (V, W)$ with vertex set $V = \{v_1, v_2, \dots, v_n\}$ and edge-weight $W_{ij} \geq 0$ between vertices v_i and v_j. We assume $W_{ij} = W_{ji}$; that is, our graph is undirected. Max-Cut is the problem of finding the largest cut in the graph, that is, the problem of finding a subset $S \subset V$ such that $\sum_{i\in S, j\notin S} W_{ij}$ the total weight of the edges between S and $V \setminus S$ is maximized.

Max-Cut is one of the first problems proved to be NP-hard [17], though it is solvable in polynomial time for some special classes of graphs [14]. Besides its theoretical importance, the Max-Cut problem has many applications in various fields, such as statistical physics [2], circuit layout design [2], and data clustering [21]. For a comprehensive survey of the Max-Cut, see Poljak and Tuza [23].

Due to the NP-hardness of Max-Cut, a lot of attention has been given to finding approximation and heuristic algorithms. For instance, Delorme and Poljak [9] and Poljak and Rendl [22] designed eigenvalue relaxation algorithms for the problem. Goemans and Williamson [12] developed an algorithm based on semidefinite programming and randomized rounding with a performance guarantee of 0.878 (i.e. a random procedure that will find a cut with expected weight at least 87.8% of the optimal value).

Recall the formulation of Max-Cut presented in the lecture:

$$\begin{array}{rlllcrll} MC := \max & \frac{1}{4}\sum_{i,j=1}^{n} W_{ij}(1 - x_i x_j) & & & = & \max & \operatorname{trace}(W(J-X)) & \\ \text{s.t.} & x_i \in -1, 1, & i \in V & & & \text{s.t.} & X_{ii} = 1 & i \in V \\ & & & & & & X \succeq 0 & \\ & & & & & & \operatorname{rank}(X) = 1 & \end{array}$$

where the equality of the two formulations is given by the relation $X = xx^T$. To obtain the SDP relaxation of Max-Cut, the (non-convex) constraint $\operatorname{rank}(X) = 1$ is dropped. The Goemans-Williamson algorithm (GW) is based on this SDP-relaxation. It consists of three steps.

step 1: Solve the SDP relaxation of Max-Cut. Let X be the optimal solution.

step 2: Decompose $X = U^T U$ where $U = [u_1, \dots, u_n]$ is a rank$(X) \times n$ matrix. This is always possible as $X \succeq 0$. Notice that from this decomposition, we have that $X_{ij} = u_i^T u_j$. In particular, $\|u_i\|^2 = u_i^T u_i = X_{ii} = 1$. I.e. all the u_i's are points in the rank(X)-dimensional sphere.

step 3: Notice that when rank$(X) = 1$, then each $u_i \in \{-1, 1\}$. In this case, U gives the optimal solution to the Max-Cut. In general, after solving the SDP relaxation, we have a representation of the graph G where node v_i corresponds to point u_i in the rank(X)-dimensional sphere, such that $\frac{1}{4}\sum_{i,j=1}^{n} W_{ij}(1 - u_i^T u_j)$ is maximized. Thus the vertices are spread on the sphere in such a way that cutting the sphere in two will provide a good cut for our original graph. The third step is based on this idea: cut the sphere in two by a random hyperplane through the origin and define the vertex partition accordingly. That is, take a random vector r of norm 1 and define $x_i = -1$ if $r^T u_i < 0$ and $x_i = +1$ if $r^T u_i \geq 0$. The Goemans-Williamson theorem states that the expected weight of the partition defined by x is at least 87.8% of the weight of the optimal cut.

In contrast to the approximation algorithms with a performance guarantee, several efficient heuristics have also been developed (see e.g. [11], [24]).

A greedy heuristic (GH) to solve the Max-Cut is to start with some random partition of the vertices and then decide greedily to move a vertex from one side to the other if this will increases the total weight of the cut. The vertex to be moved can be chosen to maximize the change in the cut weight, could be selected at random, or could be selected in some predetermined order.

2 The work

The project has two parts. First, standard approaches must be implemented and tested to measure their capabilities. The methods considered are:

1. **[2 pts] Binary Programming (BP).** Use binary programming to formulate the Max-Cut. Choose your favourite mixed integer linear programming solver to obtain the optimal solution.

2. [**3 pts**] **Greedy Heuristic (GH).** Implement (either Python or Matlab) the greedy heuristic.

3. [**3 pts**] **Goemans-Williamson (GW).** Use semidefinite programming (SDP) plus random rounding to solve MC.

In the second part ([**12 pts**]), students will choose one of (GW) or (GH) to be enhanced via ML.

A. Using an ML-guided heuristic (MLGH). GH is shortsighted, as it does not consider the combinatorics of changing sides for more than one vertex. I.e. it only considers the benefit/cost of one vertex at a time. Here, a supervised learning approach seems appropriate. For instance, an NN could be trained to include more information than just the cut weight change to select the vertex to switch side.

B. Using ML-guided rounding (MLGW). Steps 2 and 3 of GW are a randomized rounding scheme. This can be seen as a method to produce a ± 1 vector (partition of the vertices) from the given SDP matrix. The only information used in this step is the SDP-matrix. Here, ML techniques can produce better partitions from the SDP solution by replacing the randomized approach in step 3 with an ML approach for partitioning the points on the sphere. Notice that if necessary, step 2 could be adapted.

Due to time constraints (this is a short project), this second part is exploratory. Therefore, the emphasis is placed on the design, implementation, evaluation and presentation of the results. Almost no weight is given to the quality of the results. I.e. you might obtain an excellent grade even if your "great idea" fails. Still, we expect your algorithm to be faster than (BP) and at least as good as (GW/GH) in objective value.

Data

You should generate your own data. There is also plenty of data available on the web (see e.g. the problems under the category W**,** HR **and** M from the webpage `http://bqp.cs.uni-bonn.de/library/html/instances.html`). Make sure to pick hard instances (i.e. where the methods of the first part do not do such a good job) to have room for improvement in the second part. If (GW) or (GH) find the optimal solution (or a solution very close to it) all the time, then this project will make no sense!

3 The Rules

You must adhere to the following rules:

1. You must do and submit your work in groups of (exactly) two. Exceptions need to be permitted and strongly argued when asking for permission.
2. Your work should be handed in (via canvas) no later than the due date. You should submit two files: a single PDF file for the report and a single zip file containing all your code.
3. This project is marked over 20 points.

3.1 Deliverables

Report

You should deliver one final report **2-3 pages** long (not counting the front page, bibliography and appendix). You could include plots and tables in the appendix; other material included there will be ignored.The report must be typed and submitted in one pdf file on canvas. Late reports will not be accepted.

The report should clearly state the problem, mathematical models, assumptions, data generation process and methodology(es) used. For each of the methodologies, you should present the details about implementation. You should evaluate the methods using clear performance measures. Moreover, indicate any difficulties you have had and how you resolved those difficulties.

Code

You should deliver your code in one zip file. It should be clearly explained how to run your code in the report. In your code, there should be a function `maxCutSolver` (that should be callable), which should receive as input a max-cut instance (n,W) and return the partition (U, V) of the vertex set. For example,

- in MATLAB you will need to have the following heading for your function:

```
[U,V] = function maxCutSolver(numVertices,edgeWeights).
```

- in Python you could have something of the form:

```
def  maxCutSolver(numVertices,edgeWeights):
     < code here>
     return part1,part2
```

3.2 Evaluation Criteria

The quality of information, depth of analysis, organization and presentation are vital components of your report. The reports will be assessed based on the following criteria:

- Insightfulness (significance, value of information presented).
- Maturity of analysis.
- Logic and structure of arguments.
- Conciseness and clarity (your ability to explain technical material well).
- Quality of writing and presentation.

General instructions about the quality of presentation:

- Keep the length of your reports reasonable as indicated.
- Use at least 11-point font and margins of 2.5 centimeters on all sides.
- Be clear and concise in the way you write.
- Use appropriate and clearly labeled tables, charts, graphs, etc. (These often convey information more vividly than lengthy text.) However, you must ensure that you make a reference to them; that you explain what they show; and that they are fully integrated into the report.
- Ensure that all sources such as quotes and statistics are properly referenced.

Remark 1: The instructions contained here are more a general guide than a check list. Covering all these items alone do not ensure the maximum mark.

Remark 2: The bibliography below is given to provide arguments and further reading for the statements given in the description of the project. Further reading is targeted towards students inspired by this project. You are not expected to have read and / or understood all the material in the bibliography.

References

[1] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath: A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866 (2017)

[2] F. Barahona, M. Grotschel, M. Junger, G. Reinelt: An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res., 36 (1988), pp. 493-513.

[3] I. Bello, H. Pham, Q.V. Le, M. Norouzi, S. Bengio: Neural combinatorial optimization with reinforcement learning. In: ICLR (Workshop) (2017),

[4] Y. Bengio, A. Lodi, A. Prouvost: Machine learning for combinatorial optimization: a methodological tour d'horizon. arXiv preprint arXiv:1811.06128 (2018)

[5] S. Cook: The P versus NP problem. The millennium prize problems. (2006) pp. 87–104

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C.Stein: Introduction to algorithms. MIT press (2009)

[7] H. Dai, B. Dai, L. Song: Discriminative embeddings of latent variable models for structured data. In: International conference on machine learning. (2016) pp. 2702–2711

[8] S. Dasgupta, C.H. Papadimitriou, U.V. Vazirani: Algorithms. McGraw-Hill Higher Education (2008)

[9] C. Delorme, S. Poljak: Laplacian eigenvalues and the maximum cut problem. Math. Program., 62 (1993), pp. 557-574.

[10] D.Z. Du, P.M. Pardalos: Handbook of combinatorial optimization:supplement. vol. 1. Springer Science & Business Media (2013)

[11] A. Duarte, F. Fernández, A. Sánchez, A. Sanz: A hierarchical social metaheuristic for the max-cut problem. Lecture Notes in Comput. Sci., 3004 (2004), pp. 84–94.

[12] M.X. Goemans, D.P. Williamson: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM, 42 (1995), pp. 1115–1145.

[13] Gurobi Optimization: Gurobi optimizer reference manual (2020). http://www.gurobi.com

[14] F. Hadlock: Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput., 4 (1975), pp. 221–225.

[15] T. Huang, Y. Ma, Y. Zhou, H. Huang, D. Chen, Z. Gong, Y. Liu: A review of combinatorial optimization with graph neural networks. 2019 5th International Conference on Big Data and Information Analytics (BigDIA). IEEE (2019), pp. 72–77.

[16] C.K. Joshi, T. Laurent, X. Bresson: An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227 (2019)

[17] R.M. Karp: Reducibility among combinatorial problems. R. Miller, J. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York (1972), pp. 85-103.

[18] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, L. Song: Learning combinatorial optimization algorithms over graphs. Advances in Neural Information Processing Systems. (2017), pp. 6348–6358

[19] W. Kool, H. van Hoof, M. Welling: Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475 (2018)

[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), (1998), 2278–2324

[21] J. Poland, T. Zeugmann: Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. Lecture Notes in Comput. Sci., 4265 (2006), pp. 197-208.

[22] S. Poljak, F. Rendl: Solving the Max-cut problem using eigenvalues. Discrete Appl. Math., 62 (1995), pp. 249-278.

[23] S. Poljak, Zs. Tuza: Maximum cuts and largest bipartite subgraphs. W. Cook, L. Lováz, P. Seymour (Eds.), Combinatorial Optimization, American Mathematical Society, Providence (1995).

[24] G. Xia, Z. Tang, Y. Li, R. Wang: Hopfield neural network with hysteresis for maximum cut problem. Neural Inf. Process. Lett. Rev., 4 (2) (2004), pp. 19-26.