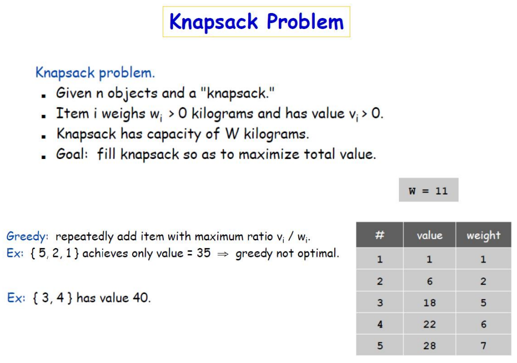
DMBA - Fall 24

Final Project

1 Preliminars

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

In particular, there is a huge 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 and 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, improving the quality of the solutions.



Knapsack Problem

Knapsack problem.

- Given n objects and a "knapsack."
- Item i weighs $w_i > 0$ kilograms and has value $v_i > 0$.
- Knapsack has capacity of W kilograms.
- Goal: fill knapsack so as to maximize total value.

W = 11

Greedy: repeatedly add item with maximum ratio v_i / w_i.
Ex: { 5, 2, 1 } achieves only value = 35 $\Rightarrow$ greedy not optimal.

Ex: { 3, 4 } has value 40.

#	value	weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

We consider the (0-1) knapsack problem (KP), one of the well-known benchmark problems in COPs. As explained in the lecture, in KP we are given a set of items, each with a value and a weight and a knapsack with a given capacity. The objective is to select a subset of the items with maximum total value to fill a knapsack such that the cumulative items weight does not exceed its capacity. KP is one of classical NP-hard problems [4, 5, 11] and it is ubiquitous in every-day human tasks at many different levels of cognition [14, 15]. KP has many practical applications including managing machine loads, cutting stock and budgeting [16].

Empirical evidence shows that solving instances near certain *phase transition* are challenging for humans [17]. The classical algorithms proposed to solve KP range from dynamic programming (e.g. [7]) to meta-heuristics (e.g. [8]). In this project a more recent approach to solve COPs is explored, namely combining existing methods with ML (or deep learning) to improve the results.

Recently, there is a great progress in ML methods to solve COPs [3]. A popular ML based method is Deep Reinforcement Learning (DRL): the integration of Reinforcement Learning (RL) and Deep Neural Networks (DNN) [1, 2, 10, 9, 12, 13]. In this project artificial neural networks (NN) will be used 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.

2 The work

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

1. **[2 pts] Binary Programming.** Use binary programming to solve the KP. Choose your favorite mixed integer linear programming solver to obtain the optimal solution.

2. [**4 pts**] **Dynamic Programming (DP).** Use dynamic programming to solve KP, as explained in the lecture. Here you need to implement yourself the algorithm presented during the lecture using either Matlab or Phyton.

3. [**3 pts**] **Greedy Heuristic (GH).** Implement the greedy heuristic explained in the lecture: repeatedly add the item with highest value to weight ratio that fits the unused capacity.

In the second part ([**10 pts**]), students will choose one of (DP) or (GH) to be enhanced by using ML. Students should choose one of the following two options:

4. **Using a NN to approximately solve the Dynamic Programming (NDP).** The main issue with the DP is the size of the table. The table is $n \times W$ in size, which is a particular issue when W is exponentially big. The idea here will be to have a NN to approximately compute the table entries (i.e. to use a NN to represent the table, which is trained using the known values of the table). A reinforced learning (Q-learning explained in the lecture) approach is recommended.

5. **Using RL to improve the heuristic (RL-GH).** GH is shortsighted, as it does not considers the combinatorics of including more than one item. I.e. it only considers the benefit/cost of one item at a time. Using RL an agent can be trained to include more information when considering which item to select to add next (instead of the simple rule of including the item with highest value to weight ratio). Each states of you MDP could be (a reduced representation of) an instance of the knapsack problem. An action will be picking up one of the items to include it in the knapsack. Notice that GH is a policy for this MDP.

Due to time constrains (this is a short project), this second part is exploratory in nature. Therefore the emphasis is placed in 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 a really good grade even if your "great idea" fails. Still, we expect your algorithm to be faster than (DP) and at least as good as (GH) in objective value.

Data

You should generate your own data. Make sure to pick n and W and generate w_i, v_i such that your instances are hard. Namely, if (DP) and/or (GH) find the optimal solution (or a solution very close to it) all the time, then this project will make no sense. Also, notice that (DP) assumes integer data.

6. [**1 pts**] Explain how is the data generated. Evaluate the quality of your data in terms of hardness of the instances generated.

3 The Rules

You must adhere to the following rules:

1. You are required to 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. There will be a penalty of 1pt for each six hours of of delay.

3. This project is marked over 20 points.

3.1 Deliverables

Report

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

The report should include a clear statement of the problem, mathematical models, assumptions, data generation process and the methodology(es) used. For each of the methodologies you should present the details about implementation, and you should evaluate them using clear performance measures. Moreover, clearly 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 explain how to run your code in the report. In your code there should be a function `knapsackSolver` (that should be callable) which should receive as input a KP-instance (n,W,v,w) and returns the set I of indexes of the objects that should go into the knapsack. For example

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

  `[indexSol] = function knapsackSolver(numObject,Capacity,objectValues,objectWeights).`

- in Python you should have something of the form:

```
 def  knapsackSolver(numObject,Capacity,objectValues,objectWeights):
      < code here>
      return indexSol
```

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

References

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

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

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

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

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

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

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

[8] Feng, Y., Yang, J., Wu, C., Lu, M., Zhao, X.J.: Solving 01 knapsack problems by chaotic monarch butterfly optimization algorithm with gaussian mutation. Memetic Computing 10(2), 135–150 (2018)

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

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

[11] Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, pp. 85–103. Springer (1972)

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

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

[14] Murawski, C. and Bossaerts, P.. How humans solve complex problems: The case of the knapsack problem. Scientific Reports, 6, 10 2016.

[15] Torralva, T., Gleichgerrcht E., Lischinsky A., Roca M., and Manes F. ecological and highly demanding executive tasks detect real-life deficits in high-functioning adult adhd patients. Journal of Attention Disorders, 17(1):11–19, 2013.

[16] Wilbaut, C., Hanafi, S., Salhi, S.: A survey of effective heuristics and their application to a variety of knapsack problems. IMA Journal of Management Mathematics 19(3), 227–244 (2008)

[17] Yadav, N., Murawski, C., Sardina, S., Bossaerts, P.: Phase transition in the knapsack problem. arXiv preprint arXiv:1806.10244 (2018)