Solving Combinatorial Problems with Transformers

Sahu, Anugya
Department of Data Science
Friedrich Alexander University
Nuremberg, Germany
anugya.sahu@fau.de

Dirauf, Richard
Department of Artificial Intelligence in Biomedical Engineering
Friedrich Alexander University
Erlangen, Germany
richard.dirauf@fau.de

Abstract—**Combinatorial Optimization Problems (COPs) are identified by a set of variables and constraints with a vast search space. They pose challenges to find a solution which can be in any form e.g., ordering, grouping or assignments. While finding any solution may seem straightforward, attaining an optimal one is a formidable task. The report first introduces us to many types of combinatorial problems and their unique constraints. Subsequently, it focuses on finding existing datasets of specific problems such as Vehicle Routing Problem (VRP) & Travelling Salesman Problem (TSP) and their properties. Additionally, it explores ways to generate artificial datasets and find real world COPs. Motivated by the need for efficient solutions to practical problems, the report further investigates heuristic methods like two-opt, city-swap, genetic and simulated annealing to solve symmetric TSPs and their comparison with the optimal solution. Moreover, it inspects and familiarizes us to the Transformers architecture with Pointer Networks, applies it to solve TSPs dataset and provides results, computational efficiency, and accuracy of the algorithm to compare them to heuristic method's results. Two-opt method gives us good results whereas Pointer Networks learns the underlying data but did not generalize well on validation split. Despite that, the results of the transformers architecture are comparable to the best heuristic method on the training split which directs us to future work of improving the network's generalization on validation set to achieve good accuracy and reduce costs.**

Keywords—Combinatorial problems, set-to-sequence models, Heuristic methods, Self-attention, Permutation invariance.

I. Introduction

Combinatorial Optimization Problems (COPs) arise in various fields such as mathematics, operations research, logistics and economics. These problems involve making choices from a discrete set of options to optimize a certain objective, subject to a set of constraints. Many possible permutations of a potential solution make it hard to develop an accurate and deterministic algorithm to find an optimal solution. Some properties of COPs include non-linearity, NP-hardness, enormous search space, infeasible solutions and existence in many variations. Already existing techniques include dynamic programming, meta-heuristic methods, and linear programming methods. These methods are not efficient and require a significant amount of time and memory resources to solve real world COPs. If a problem requires of permutation invariance property, then the complexities increase with the input being in arbitrary initial order. The methods to tackle the problems which have sets as input, but output is a sequence are known as set-to-sequence methods which gives approximate solutions to computationally intractable problems such as NP-hard Travelling Salesman Problems (TSPs) and Vehicle Routing Problems (VRPs) [1]. Some COPs like the Knapsack problem also have multiple optimal solutions based on the constraints with the same total cost or allocation that can make finding a solution easier. The first intuitive way of solving them are Brute-force methods which are not computational efficient but find the optimal solution. Dynamic programming methods are analytical and perform accurately on smaller datasets, but time complexity increases exponentially with larger datasets. Meta-heuristic methods could give a solution that is close to optimal solution which can be used to compare the results of a complex model in case of non-availability of a solution in a dataset.

Vinyals et al., 2015 introduced Pointer Networks which are designed for sequence-to-sequence tasks learns to generate output sequences by pointing to elements from the input sequence [3]. They do not use attention to merge the hidden units of an encoder to a context vector at each decoder step but use it as a pointer to select an element of the input sequence as an output. First, I searched appropriate real-world datasets for the model to train and test on. Consequently, we compared the results of pointer networks to meta-heuristic methods. Further, in order to tackle large symmetric TSPs we instigate self-attention and positional encoding architecture into encoder and decoder of Pointer Networks with Transformers to achieve adequate accuracy on 50 and 100 cities training split dataset.

II. Combinatorial Problems

A. *Assignment Problem*

The problem instance has a number of persons and a number of tasks. It is required to perform as many tasks as possible by assigning at most one agent to each task and at most one task to each agent, in such a way that the total cost of the assignment is minimized. Heuristic methods depend on the number of iterations and cost function involved, Branch and bound method is exponential in worst case. Hungarian method has O(n^3) time complexity but in practice it can be much more efficient. Output size is smaller than input and unordered.

B. *Knapsack Problem*

Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. A special case is the 0-1 Knapsack-problem in which there is no or only one optimal solution. Brute-force, Branch and bound method are exponential in worst case scenario. Greedy algorithm has O(logn) time complexity but does not always give optimal solution. Dynamic programming takes O(nw) time, where n are the number of items and w are the weights. Output will not change w.r.t. any permutations of input and will be unordered.

C. *Travelling Salesman problem*

The problem's goal is to find the shortest possible route so that an agent visits each city exactly once and returns to the starting city. It can have multiple solutions since order of cities can be different. Brute-force takes O(n!) time, Dynamic programming method runs in $O(n^2 2^n)$ time and is not effective for large scale problems. Heuristic methods take $O(n^2)$ time but do not converge to an optimal solution. Output can be different based on the starting point, since its sequential. Input consists of multiple cities that can be permutated to give different outputs, can be termed as permutation equivariant.

D. *Vehicle Routing Problem*

A generalization of the TSP which minimizes the total cost or distance travelled by the vehicles while ensuring that each customer is visited exactly once, and the capacity of the vehicles is not exceeded. Analytic methods which give exact solutions take $O(2^n)$ time and few heuristic methods take $O(n^2)$ time but do not always converge to an optimal solution. Multiple constraints such as capacity of vehicle should not exceed a certain amount, time window and distance constraints make it hard to solve. Order of locations in the output is important as it is sequential whereas input consists of distinct sets.

E. *Integer Programming*

A type of optimization problem where the variables in the objective function and constraints are required to take on integer values, as opposed to continuous values in linear programming. Simplex method, Big-M method have exponential worst case time complexity. Graphical method, or Dynamic programing methods can solve them, but their complexities increase with the number of variables and constraints. The latter has a time complexity of O(nB) where n is the number of variables and B is the maximum allocation. The problem is non-convex which means the presence of several local minimums.

F. *Minimum Spanning Tree*

Given an undirected, weighted graph, the MST problem aims to find the tree that connects all nodes of the graph while minimizing the sum of the weights of the edges in the tree. Only one solution exists i.e., the global optimal solution. Analytical methods like Kruskal's algorithm has time complexity $O(e \log e)$ whereas other algorithms like Prim's and Boruvka's algorithm have $O(e \log v)$ where e is the number of edges and v is the number of vertices. Heuristic methods may take less time but only give approximate solutions. Since input is an undirected graph, it can be considered as a set and output does not change by its permutations.

III. Datasets

For COPs it takes merely a few lines of code to generate synthetic data which has a lot of randomizations and can be used to train any model, but a real-world dataset brings much needed difficulties and scale to the problem. Existing libraries for COPs consists of synthetic datasets which are random and generated using algorithms but also real-world datasets.

Operations Research library (ORLIB) consists of datasets for many of the above discussed COPs and much more. It is publicly available and can be accessed by anyone. The datasets can be used for testing the algorithms or models solving the problems such as Knapsack, TSP, VRP, and Assignment problem. QALIB contains a collection of quadratic assignment problems. MIPLIB and COR@L benchmark library comprises datasets for Integer Programming.

TSPLIB contains datasets that are specifically for TSPs and VRPs with geographical, symmetrical, asymmetrical, and random examples for training. Specifically, it has datasets that are real-world or collected through a process in form of coordinates. Few datasets can also have different number of cities in each instance that requires padding however, we work on two large consistent datasets TSP50 and TSP100 with equal number of cities in which optimal order is given and instances split up into training and test datasets. To calculate the optimal costs Euclidean distance is used. The shape of the train split of datasets are (30000, 50, 2) and (10000, 100, 2) respectively although the validation split of both datasets consists of 100 instances. In order to train the model we shuffle the instances. VRP datasets are much more complex with various tables comprising for constraints on depots, vehicles, customers, costs, and time. They are not easy to use requiring a lot of pre-processing devoid of optimal solutions.

An example of a real-world COP dataset is PROCAT which consists of catalogue data consisting of over 1.5 million set items across a 4-year period from 10000 catalogues, in both raw text form and with pre-processed features containing information about relative visual placement [2]. Text features of offers grouped into sections, the inputs are sequential with variable lengths and substructures. The dataset is divided into the real-world dataset and one synthetically generated set of simplified catalogue structures. The latter allows for the prediction of multiple valid catalogue structures from the same underlying input set but in case of the main dataset only one target permutation is available. Furthermore, it consists of three feature tables which are offers, sections and catalogue.

IV. Heuristic Methods

Heuristic methods find approximate solutions and follow a greedy approach which give importance to speed and efficiency rather than finding an optimal solution. They are intuitive and easy to understand without exhaustive search of all possible combinations of solutions in the domain space. Meta-heuristic methods are problem independent so that the algorithm can be applied to a wide range of problems [6].

1. *Two opt:* A local search algorithm, iteratively swaps pairs of edges in a tour to reduce the total distance, compares every possible valid combination of swapping algorithm, it basically examines pairs of edges. It starts with an initial tour given by any constructive method like nearest neighbour then it chooses any two edges, calculates and compares the total distance. If there is an improvement it swaps otherwise does not, it iteratively performs many swappings until no further upgrade is there.
2. *City-swap:* It is similar to two-opt algorithm, but no constructive algorithm is used, cities are randomly sorted, and a starting solution is created, it tries to swap all cities with nested loops.
3. *Genetic:* It is an iterative optimization process that

uses selection, crossover, and mutation to improve the solution of the problem. Each individual represents a potential tour, initialization is random, then evaluation and selection is done. It combines several potential tours and initiates random changes, mutation can swap two cities, insert or delete a city or any other modification. Termination can be when no further improvement is left or a fixed number of iterations.

4. *Simulated annealing:* The algorithm iteratively explores the optimal solutions search space by taking both betterand worse solutions, not a greedy approach, uses a temperature-controlled probability, if it decreases, algorithm converges towards better solution and vice-versa. Various parameters include setting initial temperature, cooling schedule, stopping conditions. Modifications are accepted with a certain probability that allows for exploration of different solutions, gradually the probability of accepting worse solutions decreases as the temperature is reduced according to cooling schedule.

Comparing the running times and optimal solution average gap of instances of test split of TSP50 dataset. *Optimal solution – 568331.54*

Fig. 1

	A.	*B.*	*C.*	*D.*
Abs.	12537.14	403437.36	55292.95	235214.57
Rel.	2.2%	70.98%	9.72%	41.38%
Time	0.0754	35.35	164.50	657.88

Comparing the running times and optimal solution average gap of instances of test split of TSP100 dataset. *Optimal solution - 764848.60*

Fig.2

	A.	*B.*	*C.*	*D.*
Abs.	45034.19	686543.89	236060.60	666537.43
Rel.	5.88%	86.49%	29.73%	83.97%
Time	19.55	305.15	529.55	657.88

Fig.1, 2. - The tables show us the absolute difference, relative difference between the output & the optimal distance and time complexities in seconds of different heuristic methods on an instance of TSP50 and TSP100 test dataset respectively. Most heuristic methods are random hence data differs with separate runs.

The two-opt algorithm performs best among all the heuristic methods with relative difference error being approximately 6%. A relatively better method is genetic algorithm with a 30% error but city-swap and simulated annealing methods perform the worst with less than 15% accuracy.

V. Poiner Networks

Pointer Networks are an encoder-decoder architecture that learns a target reordering of input elements, can work with desired permutations of varying input sizes [3]. Applications include solving Convex hull problem where we are given n points and it is required to find a minimum subset of those points that geographically include all the points, planar TSP, text summarization, and content selection. Target is a sequence of points to inputs that are defined at inference time not at training time.

RNN based sequence-to-sequence models can not address different output sequence length which depends on the size of the input sequence. In the original paper, RNN is used to map the input sequence to an embeddding and the same RNN to map the embedding to an output sequence [4]. The output dimensionality is same during training and inference. A RNN encoder generates a code from the input sequence which goes to the generating network that produces a vector that influences a content-based attention mechanism over inputs. The output of the attention mechanism is a softmax distribution with dictionary size equal to the length of the input [3]. Improvements can be done by augmenting the decoder by propagating extra contexual information which gives importance to the elements based on their relevance to the current step using a content-based attention mechanism [5].

COPs require varying input sizes and influence the size of the output sequence because of which these methods cannot be directly applied to solve COPs. All models that are applied to COPs use a single layer of LSTMs which are used to model the conditional probabilities and the state is used after the output gate has been component-wise multiplied by the cell activations. For loss computation, stochastic gradient descent, random uniform weight initialization and L2 gradient clipping are utilized. Input sequence index with the highest weight is considered as output for the decoder sequence index and softmax applied in the output directs to the input element having the maximum value [3]. Lastly, advancements are done by replacing the LSTM-based encoder and decoder with transformers.

VI. Transformer Architecture

"Attention is all you need" is an approach to solve sequence-to-sequence tasks which solely rely on self-attention mechanisms without using recurrent neural networks [7]. Self-attention allows the model to capture long term dependencies across different positions in the sequence through generating a weighted sum of the input elements based on their relevance to each other. Parallelization can be used in sequential data using Transformers.

Input embeddings map the input sequence into a space where similar elements are placed closer to each other. Positional encoder is a vector that gives context based on position of an element in that sequence. Together it generates a vector with contexual information which is passed into an encoder block. For every element of a sequence an attention vector can be generated by multi-head attention step to capture relationships. The generated vectors are then passed into feed-forward neural networks to generate the sequence that will be passed into the next encoder or decoder block. The decoder block first generates output sequence at a time and also takes information from the encoder block. As the output embeddings are generated and passed through masked multi-head attention, batch-normalization is done to stabilize training. Attention vectors from input sequence and output sequence are both passed into the decoder block. Lastly it passes through a linear layer which is essentially a feed forward network used to expand the dimensions and softmax layer transforms it into a probability distribution which is the final output.

VII. Results

TSP50 and TSP100 datasets are passed into the pointer network with transformers with a batch size of 16 and 20 epochs.

Fig.3

Epochs ↓	Train Acc.	Val. Acc.	Train Loss	Val. Loss
5	53.05%	3.83%	1.29	26.17
10	64.24%	4.02%	1.03	27.93
15	70.02%	4.24%	0.88	28.43
20	70.97%	4.49%	0.85	28.03

Fig.4

Epochs ↓	Train Acc.	Val. Acc.	Train Loss	Val. Loss
5	30.32%	1.56%	1.78	25.69
10	37.03%	1.83%	1.63	27.63
15	45.46%	1.61%	1.49	28.62
20	49.82%	1.67%	1.39	28.75

Fig.3, 4 - The table show us the accuracy and loss of training split and test split datasets of TSP50 and TSP100 in an interval of 5 epochs.

Both the tables show a large difference in the accuracies between training and validation set. While in the case of TSP50 dataset, the accuracy reaches almost 71% in 20 epochs but only approximately 50% in case of TSP100. Furthermore, validation loss is almost same in both cases, accuracy being less than 5%.

Fig.5

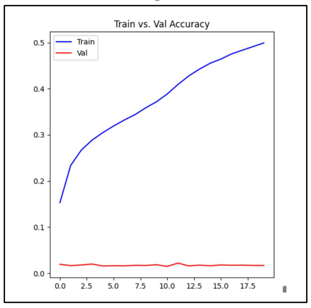

Fig.6

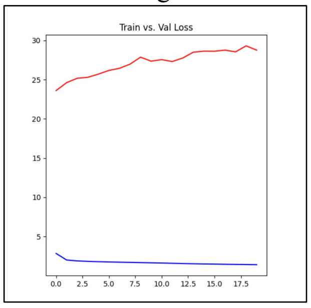

Fig.5, 6. - The figures show us the accuracy and loss curve of training and validating the TSP100 dataset.

Fig.7

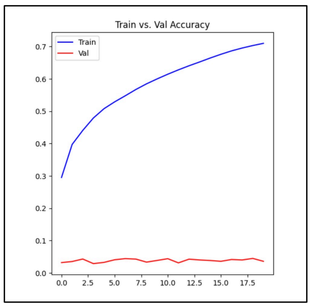

Fig.8

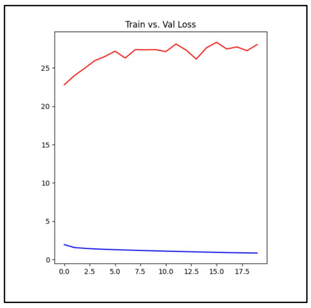

Fig.7, 8. - The figures show us the accuracy and loss curve of training and validating the TSP50 dataset.

The curve shows a steady increase in accuracies and decrease in loss in both cases of training dataset. In case of test dataset, accuracies stays almost the same and loss increases in an oscillating manner.

VIII. Conclusions

The results from the pointer networks with transformers architecture perform better and faster than most heuristic methods. The learn the model well and give good accuracies in few epochs in the training set. I calculated the optimal distance from the given optimally ordered cities and the predicted optimal distance through predicted order in the validation set. Although, the network could not generalize well and decreasing loss in case of test dataset can be a possible future work.

Acknowledgment

I am grateful to my supervisor Richard Darauf, for his guidance, mentorship, and prompt support throughout this project. The valuable insights and feedback were important in shaping the direction of this task.

References

[1] Mateusz Jurewicz and Leon Derczynski, “Set-to-Sequence methods in Machine learning: A Review”, arXiv: 2103.09656, Journal of Artificial Intelligence Research 71 (2021) 885 – 924

[2] Mateusz Jurewicz and Leon Derczynski, “PROCAT: Product Catalogue Dataset for Implicit Clustering, Permutation Learning and Structure Prediction”, doi.org/10.6084/m9.figshare.14709507

[3] Oriol Vinyals, Meire Fortunato, Navdeep Jaitly, “Pointer Networks”, NIPS – 2015, pp. 2692-2700

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks”, In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *In ICLR 2015, arXiv preprint arXiv:1409.0473*, 2014

[6] George Lindfield, John Penny, “Numerical Methods (Fourth Edition)”, Academic Press, 2019, Pages 433-483, ISBN 9780128122563

[7] Ashish Vaswani et al. “Attention Is All You Need”, Google Research, arXiv:1706.03762