# Combinatorial Optimization

Michael Clausen

CPSC 585-01

Spring 2023

## Combinatorial Optimization

One of the most important problems in computer science is combinatorial optimization, a discipline concerned with finding "an optimal object from a finite set of objects" (Schrijver, 2003). A major part of this field relates to finding important optimization information about graphs. Some of the most prominent problems include finding the minimum spanning tree and the shortest path, which have polynomial time solutions, along with finding the minimum vertex cover (MVC), the maximum cut, and the travelling salesperson problem (TSP). These last three are NP-hard, which means that the fastest known algorithms for them take exponential time (Drori, 2022). Heuristics and approximation algorithms can be tailored for individual problems with improved time complexity. However, these are laborious to generate and are not scalable for large graphs. Much work in recent years has been dedicated to using neural networks to solve such optimization problems more quickly in a manner that is scalable. Graph neural networks (GNNs) appear to be an obvious candidate, since they have been shown to have "algorithmic alignment" with such problems.

One of the seminal papers applying machine learning to this field is "Learning Combinatorial Optimization Algorithms Over Graphs" by Dai et al., published in 2017. It outlines a machine learning model, called Structure2Vec Deep Q-learning (S2V-DQN), that can generalize and solve a range of combinatorial problems, such as TSP, MVC, and MAXCUT (Dai et al., 2017). Their method combines greedy meta-algorithm design, deep embeddings for the graphs, and a reinforcement learning method called Q learning. This results in a flexible model that outperforms competing RNN models.

The first component S2V-DQN, the greedy meta-algorithm, takes a sample graph G from a distribution of graphs *D* (Dai et al., 2017). It then maintains a partial solution data structure *S* that is continually updated. There is also a helper function *h(S)* which helps maintain the data

structure *S* so that it satisfies the constraints of the problem. Finally, there is an objective function *c(h(S), v)*, which calculates the quality of *S*, and the evaluation function *Q(h(S), v)*, which chooses whether to add the node *v* to the solution. The solution is continually updated until a termination criterion *t* is met.

The deep learning portion of S2V-DQN involves learning an embedding for the graph *G* called *structure2vec* (Dai et al., 2017). In structure2vec, a feature vector is maintained for each node *v*. Features "are aggregated recursively according to *G*'s graph topology" (Dai et al., 2017). This updating of the features is carried out once for each node every iteration. The model then parameterizes the evaluation function $\hat{Q}$ for each node *v* by summing the embeddings of its neighbors and then applying the rectified linear unit function (relu). Since there are no ground truth labels for the graphs, the embeddings are "trained end-to-end using reinforcement learning."

For the reinforcement learning, S2V-DQN uses n-step Q reinforcement learning to train the model (Dai et al., 2017). This consists of five components. The first is the *state*, which is the summation of the actions on the nodes *v* in a graph *G*. The second is the *transition*, which tags a new node by setting the feature $x_v$ to 1. Then there is an *action*, which is "a node of *G* that is not part of the current state *S*" represented by its embedding. Next comes the *reward, r(S, v)*, which is a function on state *S* and action *v* based on the cost function associated with them. Finally there is the *policy*, which uses the greedy meta-algorithm to choose the next node to add to the partial solution, resulting in the reward *r*. This is performed over *n* steps instead of one to "collect a more accurate estimate of future rewards."

To test S2V-DQN, the authors used generated data with 15 to 500 nodes along with real world data sets for three combinatorial problems: MVC, MAXCUT, and TSP (Dai et al., 2017).

They compared their results with the best approximation algorithms available for each problem along with an RNN called PN-AC. They measured performance using an approximation ratio which compares the objective value of the solution with the best known solution for a given graph. S2V-DQN outperformed the competition for both the generated and real world data sets for MVC and MAXCUT. For TSP, it performed as well as the competition for the generated data but did better on the real world data.

Now we turn to more recent work in the field. Two papers were selected; “Select and Optimize: Learning to solve large-scale TSP instances” by Cheng et al, and “Memory-efficient Transformer-based network model for Traveling Salesman Problem” by Yang et al. Both were published in 2023 and thus represent cutting edge research. Both attempt to solve issues applying transformer architectures to one specific optimization problem, the travelling salesperson problem.

The first new paper, “Select and Optimize: Learning to solve large-scale TSP instances,” outlines a method to solve TSP problems with graphs ranging from 200 to 20,000 nodes (Cheng et al., 2023). This is normally not possible with current neural networks. The authors achieve this by training a selector-and-optimizer network, using a transformer architecture, to iteratively select optimal sub-problems in the graph. It then uses a destroy-and-repair method in order to avoid local minima.

The selector and optimizer network takes advantage of a neural network’s ability to solve small instances of TSP quickly (Cheng et al., 2023). The network makes sub-paths of length *r*, with *r* being a relatively small value. The network can quickly generate costs for all sub-paths of length *r* in the graph and choose the optimal one. The final solution consists of *N* such sub-paths. This algorithm is *O(N)* or even close to linear time if parallel computing is utilized. The head and

tail of each sub-problem remains fixed, which are observed by the decoder block of the transformer. The transformer is then trained using a reinforcement learning method called Policy Gradient.

In order to prevent the selector-and-optimizer network from becoming stuck in local minima due to a lack of a global perspective, the authors enhanced the model with a destroy-and-repair method (Cheng et al., 2023). For each iteration, the solution is destroyed at a certain frequency by first eliminating the longest connections. Then, further connections are destroyed "to make the fragments more closely resemble each other." Finally, the solution is repaired using the *Lin-Kernighan* algorithm. This method also maintains the variant that the new solution must have a lower cost than the one destroyed.

In terms of results, the selector-and-optimizer network outperforms other learning based methods on both generated and real world data, especially on large graphs (Cheng et al., 2023). Theirs is also the first learning based model able to solve TSP problems with up to 20,000 nodes. They also performed a sub-problem selector comparison and an ablation study on the destroy-and-repair method with promising results.

We now turn to the second paper, "Memory-efficient Transformer-based network model for Traveling Salesman Problem." This work also modifies the transformer architecture in order to solve TSP instances more quickly and efficiently (Yang et al., 2023). Normally, applying transformers to such problems requires prohibitive training times and more memory than modern systems can support. Their model, called Tspformer, reduces memory usage and time complexity in several key ways. First, for the query values Q, they take a sample from the data instead of passing all of it. This reduces the time complexity for the dot product in the attention layer from $L^2$ to $L\log(L)$. The Key values K are also sampled, using the equation $\alpha\log(\alpha)$, with $\alpha$ used as a

hyper parameter. They also remove the multi-head attention module in the decoder, along with the positional encoding for the embedding. The authors call this module an “efficient sampled scaled dot-product attention mechanism.” The encoder remains untouched. This results in Tspformer being able to be trained on graphs with up to 1000 nodes, whereas competitors max out at 200. Though the accuracy of the model is not quite as good as their competitors, Tspformer can be trained much faster and can solve TSP instances with as many as 100,000 nodes due to the decreased memory usage.

The work in the two new papers by Cheng et al and Yang et al both relate directly to the work described in the textbook in that they specifically apply neural networks to the travelling salesperson problem. They also build upon this work by using transformers instead of GNNs. Transformers have been shown to be extremely powerful and flexible in numerous domains including combinatorial optimization (Cheng et al., 2023). However, both papers had to deal with problem of the prohibitive amount of resources that transformers require (Cheng et al., 2023; Yang et al., 2023). This means that they cannot be used effectively for large TSP instances. Cheng et al ameliorate this by using a selector-and-optimizer network to solve smaller sub-problems, while Yang et al modify the transformer architecture by removing unnecessary modules and data. This allows both models to handle large scale TSP instances efficiently as well as improve training time and memory usage.

# References

Cheng, H., Zheng, H., Cong, Y., Jiang, W., & Pu, S. (2023, April). Select and Optimize: Learning to aolve large-scale TSP instances. In International Conference on Artificial Intelligence and Statistics (pp. 1219-1231). PMLR.

Dai, H., Khalil, E., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. Advances in neural information processing systems, 30.

Drori, I. (2023). The Science of Deep Learning. Cambridge University Press.

Schrijver, A. (2003). Combinatorial optimization: polyhedra and efficiency (Vol. 24, p. 2). Berlin: Springer.

Yang, H., Zhao, M., Yuan, L., Yu, Y., Li, Z., & Gu, M. (2023). Memory-efficient Transformer-based network model for Traveling Salesman Problem. Neural Networks, 161, 589-597.