

Review of Neural Networks Application in UAV Routing Problems

Maksym Ogurtsov

V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, Akademika Glushkova Avenue, 40, Kyiv, 03187, Ukraine

Abstract

The paper reviews the ways and methods to the neural networks (NN) usage for solving combinatorial optimization (CO) problems, appearing when using unmanned aerial vehicles and based on their specifics. It is determined that the use of neural networks (with supervised learning, reinforcement learning and deep learning) is possible for many types of CO routing problems (salesman's problem, VRP problem in different versions, etc.) and other unmanned aerial vehicles CO problems. NN with reinforcement learning, as well as recurrent NN with controlled learning may be successfully used to directly solve combinatorial optimization problems mentioned above, or to adjust combinatorial optimization algorithms parameters.

Keywords ¹

Neural networks, combinatorial optimization, VRP, unmanned aerial vehicles, deep learning

1. Introduction

Today, decision support tools are used more and more often all over the world, for example, in transportation, military affairs, supply chains and logistics, energy, finance, and operations planning [1, 2]. But the use of neural networks (NN) in these tools is quite limited yet [3, 4].

Research and development of decision support tools, which are also called administrative analytics, began during World War II as an initiative to use mathematics and computer science to help military planners to make decisions [5].

Problem statement: the analysis of the possible ways to use algorithms of machine learning (ML) in NN to solve problems of combinatorial optimization (CO) [2], appearing when using unmanned aerial vehicles (UAVs), consisting of UAVs, their operators, etc., for which the distribution of initial data is unknown in advance [3] to find out possibility of use ML for solving such problems.

Typically, the use of NN is well suited for processing signals of natural origin, for which it is difficult to perform formalization and clear mathematical formulation [3], because the distribution of initial data in this case cannot be determined analytically, for example, image processing, text, voice, or molecule analysis. etc.

Significant progress has recently been made with the use of machine learning - through in-depth learning [3]. Deep learning provides better results than classical algorithms when used in multidimensional spaces with large amounts of input data.


Routing problems that arise when using UAVs has led to the appearance of numerous studies conducted in recent decades. There were many attempts (including successful ones) to solve the problems of CO with the use of ML. They were considered by relevant researchers and publications all over the world [3, 4, 6, 7, 8, 9, 10, 11].

World experience in the UAV development of shows that in the next 10-15 years UAVs will be

II International Scientific Symposium «Intelligent Solutions» IntSol-2021, September 28–30, 2021, Kyiv-Uzhhorod, Ukraine

EMAIL: ogurtsov.maksym@incyb.kiev.ua (A. 1)

ORCID 0000-0002-6167-5111 (A. 1)

 © 2021 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

able to perform most of the tasks that are currently solved by manned aerial vehicles. And in Ukraine, due to the difficult political and economic situation, a deeply rational solution will be to use the methods of CO to increase the efficiency of such tasks for UAVs. Moreover, the rapid expansion of UAV applications in recent years has created a need for new classes of problems, like route building for swarms of UAVs, performing same task as a team etc.

2. Analysis of recent research and publications

As it was mentioned earlier, recently, significant progress has been made with the use of ML – through in-depth training [3]. As it was said earlier, ML usage ways for solving the CO problems was considered by both foreign [3, 4, 6, 7, 8, 9, 10] and domestic researchers [11, 12, 13, 14]. In this work attention will be paid mostly to discrete optimization problems: integer optimization with constraints, i.e., optimization problems with integer / binary variables. Although not all such problems are difficult to solve (for example, the shortest path finding problem), we will focus on NP-complex problems of CO. For these problems, the existence of an algorithm able to find a solution in a time, which is a polynomial of the size of the input data is considered unlikely. However, in practice, CA algorithms can find solutions to problems containing up to millions of variables, parameters, and constraints.

How NP-complex problems can be solved in a reasonable time? Let us consider the classical example of an NP-complex problem, the traveling salesman problem (TSP), that could be presented in a graph where we look for a minimum length solution by visiting each node once and only once [15]. Let us focus on the Euclidean TSP [16]. In this version, every TSP node contains coordinates in Euclidean two-dimensional space (or even in a space with number of dimensions, that could be more than two), and the objective function for the segment connecting the two nodes is the Euclidean distance between them. Although theoretically this example of the problem is as complex as the general case of the TSP, the approximate solution can be effectively found in Euclidean space using the structure of graphs [17]. But we should consider that for UAVs three-dimensional Euclidean space should be used instead of two-dimensional in many cases to get valid solutions.

There is a large amount of literature on heuristic algorithms, i.e., algorithms designed to calculate practically acceptable solutions of CO problems with no guaranteeing optimality. These heuristic methods are very important, using in CO and some of them will be considered in this paper. The most popular are the following [3]. We should notice 2 preferred motivations for using ML: approximation and the creation of new policies.

According to current research, next types of NN are or were applicable for solving CO problems (some of them aren't using for this task at modern time):

- Hopfield Networks
- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Deep learning

There are two popular ways for the ML use to solve CO problems – with and without expert knowledge. In this case, the decision-making function (DMF) is called a policy. DMF provides all present information (state of the environment, if the information able to describe the environment in enough details at this step), issues output (possibly stochastically), which should be performed in the next step of the algorithm. Policy is a function that we want to teach with the help of ML. Consider how the above two motivations determine the parameters of learning.

3. Selection of previously unsolved parts of the overall problem

The purpose of this work is to determine the applicability of NN to solve routing problems that arise when using UAVs, i.e., to highlight promising areas of research of the NN usage to solve the CO

problems. NN can be successfully used for CA tasks that arise when using UAVs, primarily – when planning routes for UAVs and especially – for UAV groups. This is because the task of constructing routes and allocating points to be visited between vehicles is a classic CO task – the task of VRP (Vehicle Routing Problem) [18]. And such a task constantly arises when planning operations using UAVs.

4. Novelty

Although, as mentioned above, the use of NN for solving the CO problems is actively studied all over the world, its use for UAV-specific tasks has not been studied enough to date.

5. General scientific significance

The use of NN to solve the UAV-specific CA problems can improve the quality of the resulting solutions, which will increase the efficiency of the UAVs use in general and potentially – to expand the UAVs scope.

6. NN types, applicable for CO problems

NN can help improve CO algorithms in two ways. First, the researcher can use expert knowledge of the CO algorithm to replace resource-intensive calculations with rapid approximation. NN can be used to calculate such approximations. The challenge is to explore the space for possible solutions and thus be able to choose the appropriate policies that have led to the best of these solutions. Second, from the point of view of using NN to solve CO problems, NN can decompose the problem into smaller, simpler problems.

Consider the main types of NN that can be used to solve UAV-specific CO problems.

6.1. Hopfield Network

The first attempt to formalize the description of nerve cell function for applying in computational algorithms was a model proposed in 1943 by McCulloch and Pitts. The McCulloch-Pitts model became the starting point for building a simple unidirectional neural network called a perceptron. Such a network was proposed and researched by Rosenblatt in the late 1950s and early 1960s. And in 1960, Adaline systems (Adaptive Linear Neuron) were proposed. Later, the Hopfield neural network was suggested.

The Hopfield NN is a fully connected, with a symmetric matrix of connections [19]. If we modify the objective function of this NN so that it is equivalent to, for example, the objective function of the salesman problem for UAV, and use Lagrange multipliers to set penalties for violating the constraints of the problem, then such a network can be used to solve multiple CO problems.

Such a network can be used as auto-associative memory, as a filter, and to solve many CO problems. Unlike many NN, which work to receive a response through a certain number of cycles, Hopfield networks work until equilibrium is reached, when the next state of the network is equal to the previous one.

Each neuron in the system y_i can take one of two states (like the output of a neuron with a threshold activation function):

$$y_i = \begin{cases} 1 \\ -1 \end{cases}. \quad (1)$$

A limitation of Hopfield networks is that they are vulnerable to the problem of hyperparameters [20]. Although such neural networks have several promising properties, their practical application has

remained limited, in most cases only as research work.

6.2. Supervised learning

In supervised learning during the input, we have a set of input data (functions) / target pairs, the task is to find the function of converting input data into output, which for each set of input data provides output that meets the specified constraints and has the value of the objective function closest to the target. This is how we define learning process. This process may be completed by finding a solution for an optimization problem on a set of available functions. The loss function, i.e., the degree of discrepancy between the source and target values, can be selected depending on the task selected (for example, regression, classification, etc.) and optimization methods.

Let X and Y , which have a common probability distribution P , be variables for the input data and the objective function. Let ℓ be a minimization function of losses and let $\{f_\theta \mid \theta \in R^p\}$ is a set of ML models for CO (in this case, parametric). The problem of controlled learning is defined as:

$$\min_{\theta \in R^p} E_{X, Y \sim P} \ell(Y, f_\theta(X)). \quad (2)$$

6.3. Unsupervised learning

In this type of learning [3] the NN does not have the goals of the problem it solves. Instead, it tries to get some characteristics of the general distribution of random variables observed during solving the problem. Because unattended learning has so far received little attention in CO context and its immediate use seems difficult, this method would not be considered in detail.

6.4. Reinforcement learning

Controlled learning is not applicable to most CO problems because it does not have access to optimal distribution of source data. However, you can compare the quality of the solution set using an evaluation data set for validation and determine the reward for the learning algorithm. Therefore, it is possible to adhere to the paradigm of neural reinforcement learning (RL) to be used for finding solutions of the CO problems. But unfortunately, even when we know optimal solutions of CO problem already and may use this information as training data to optimize controlled learning, the quality of the results is quite poor compared to machine RL, which receives a corresponding reward for good solutions [3, 4, 21].

Sometimes it may be complicated to build the reward function. Because RL is built on a dynamic process, it is naturally able to anticipate states / transitions that lead to future rewards. However, the above setting of rewards is difficult and not always effective, as it does not allow learning until the agent (using some kind of policy, or just randomness) looking for the solution. In addition, when an approximate policy is applied (for example, using greedy search), RL does not have 100% probability of finding the best possible solution and may fall to local lows.

6.5. Deep learning

Deep learning is a method of constructing NN, consisting of the large number of layers. In the case of the simplest NN architecture, a NN with a direct connection, also called a multilayer perceptron (MLP), the input data is sequentially transmitted through several layers. For each layer on the input vector, an affine transformation is applied, followed by the application of a nonlinear scalar function. The output, given by the current layer, is known as the function of intermediate activation. This output is passed to the next layer.

All operations, presented on different layers are independent and may be shown as different

matrices of coefficients. They are trainable, i.e., optimized with, for example, a stochastic gradient descent (SGD), an algorithm for obtaining minimum of the selected loss function. Stochasticity is generated by the finite number of data used to calculate losses before using SGD routine.

Deep neural networks (DNNs) may be difficult to optimize, so several methods and approaches have been developed to make a better optimization, often by changing the architectural aspects of the network. Because DNNs have a large dimension amount, i.e., they can correspond to essentially any data set, they are prone to excessive content problem. DNNs are also subject to active regularization. Learning such networks with SGD also regulates them through gradient noise, which makes DNNs generally reliable for excessive content problems. In addition, there are many hyperparameters for DNN and various combinations of them are evaluated (this approach is known as hyperparameter optimization). DNNs move away from more classical ML methods and algorithms, processing all present input data, for example, all pixels of the digital image file, to learn, while traditional ML usually requires the use of a limited number of features specific to a particular task.

DNN researchers have created various methods for adapting to a variety of structured input data in such a way as to process these input data of different dimension or size, for example – variable-length sequences.

Consider the following modern techniques.

6.5.1. Recurrent neural networks

The first presented architectures, applicable for CO problems, are recurrent neural networks (RNN). RNNs can work on sequential data by exchanging parameters at different stages of the sequence. More precisely, the same NN block is consistently applied at each step of the sequence, i.e., with the same values of architecture and parameters at each step of time. The specificity of such a network is the presence of repeating layers. Individual layer uses as input two sets of data: first one is the activation vector, which is the output, generated by the previous layer. Second one is the actual activation vector at the previous stage of the sequence, as it is shown on Figure 1.

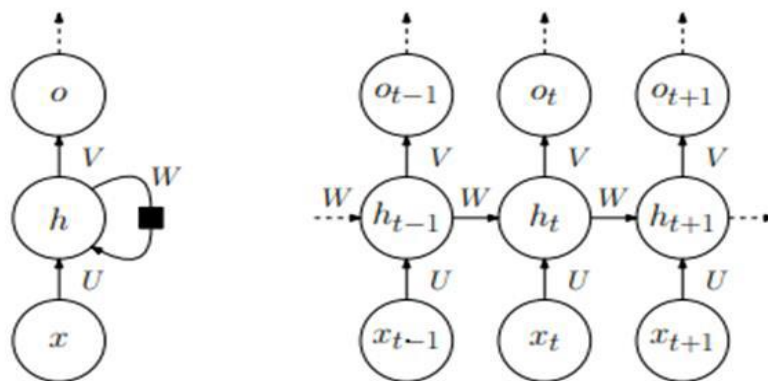


Figure 1: RNN architecture

On the left side of the Figure 1, the dot of the black color presents a one-step delay. More detailed indication of this RNN layer with previous and next ones shown the right. U , V and W are parameters sets, which are presented and reused at each step of the RNN sequence.

6.5.2. Attention mechanism

One more important technique that makes the problem invariant to input data sizes is the attention mechanism [3]. It can be used for data processing, where each point of the output data corresponds to a set of input data. In this context, parameter exchange is used to consider the fact that different sets of input do not have to be the same dimension. The attention mechanism is used to request

information about all the elements in the set and combine this information for further processing in the NN, as shown in Figure 2.

The classical mechanism of attention – the query q is calculated for a set of input values $(v_i)_i$. The f function is used for pairs of queries and sets of input data values. If it includes some parameters, the mechanism of attention is amenable to learning.

The affinity function takes a request at the input (which is any info type, used to find out, what exactly should be put into attention, where to focus) and a representation of the input set element (as we know, both will be used for the next layer) and produces a scalar. This process is happening on the set, including all its elements, and repeated for every other query. The obtained scalarized source data are normalized (for example, using the softmax function) and used to determine the operation result for the elements in the set. This weighted amount, in turn, can be used in the NN.

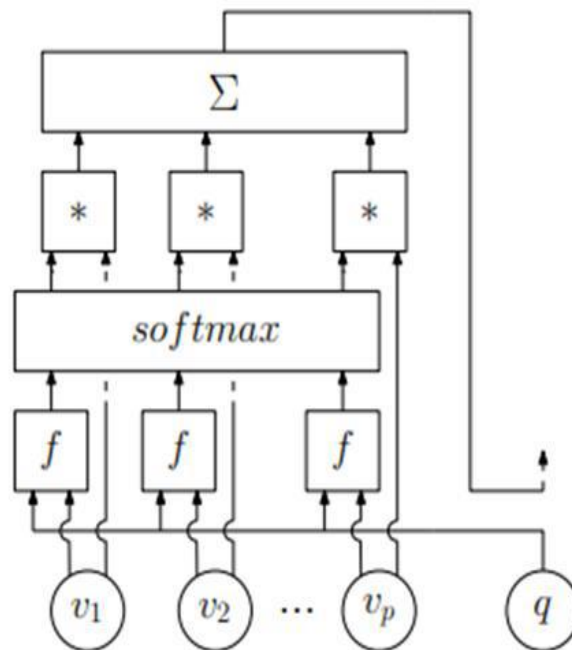


Figure 2: NN attention mechanism

The attention mechanism can be used to construct NNs of graphs, i.e., NNs capable of processing structured input data of graphs [22]. In this architecture, each node pays attention to all closest elements. Then this routine is repeated several times to further collect information about the nodes.

Deep learning and RNN can be used for supervised, unsupervised or reinforced learning [3].

7. Approaches of using NN for solving UAV CO problems

There are several approaches of using NN for solving UAV CO problems. Let us start with the RNN.

7.1. Usage of RNN

Neural CO can be used to solve CO problems using RL. We will examine 2 possible approaches based on gradient policy [23]. The first approach, called pre-training with reinforcement (PTR), uses a training kit to optimize an RNN, which performs parameterization of stochastic policy on test dataset, using the expected reward as a goal. During the tests, the reward is fixed, which provides conclusions about the quality of solutions based on the use of greedy sampling.

Second way of using RL for solving CO is an active search. It doesn't involve prior RNN training.

The search begins with a randomly chosen way to build a solution (for example, greedy search) and then from run to run optimizes the RNN parameters on a test data set, also using the expected reward as a goal, while maintaining the best solution selected during the search. The combination of PTR and active search provided best results after multiple practical comparisons [4].

7.2. Approximation and creation of new policies

There are different ways to use ML to solve CO problems. There are two main motivations for using ML: approximation and the creation of new policies. There are also various ways to combine ML and traditional algorithmic elements.

As mentioned earlier, there are two main ways of ML usage – with and without expertise knowledge. Both motivations are identified within the MDP states/actions described in the reinforced learning section.

In the case of using ML to obtain approximate solutions, policy is often taught using simulation training, through demonstrations, with data from the expert, provided by the ML model. This process is illustrated in Figure 3. When using this approach, the NN is trained not to optimize performance, but to blindly follow the actions of the expert.

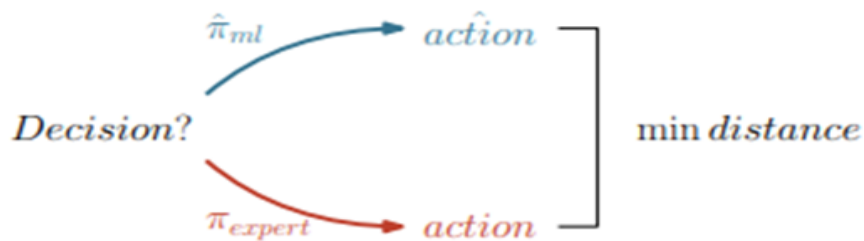


Figure 3: The process of the NN simulation training

When using simulation training, the policy learns to how to repeat success of the expert (trainer), looking for ways to obtain minimal differences in the space of action.

In the case where the goal is to create new policies, i.e., to optimize the solution-finding function from the scratch, the policy can be trained using reinforcement learning, as shown in Figure 4. Even if we present the problem of learning under the main MDP through RL, this does not limit the use of basic RL algorithms (such as, for example, approximate dynamic programming algorithm or usage of the gradient policy) to get the best amount of reward that is possible to get. Alternative optimization methods, such as greedy algorithms, branch-and-bound, genetic algorithms, metaheuristic approaches, direct/local search, can be applied as well to find solutions of RL problems.



Figure 4: Experience-based RL

In the case of RL a reward signal is used, no expert participates in the training process; only the maximization of the expected amount of future rewards is very important.

It is important to understand that when using simulation training, the NN learns with controlled goals set by the expert for each step of the algorithm (and without getting any reward), when in the case of experiential learning, the NN learns by receiving a reward (possibly with a delay) and using reinforced training (and without an expert). When using simulation training, the NN is taught what to do, while in training with reinforced NN, it is recommended to use rewards as main parameter and do the best trying to maximize it. However, it should be taken into account, that connections of these two parameters is a much more complex and broader issue than the information provided here.

Results of approaches to use NN for solving UAV CO problems comparison are presented in Table 1.

7.3. The method of branches and boundaries

NN can be used to improve the quality of solutions by improving the current local solution with cutting planes (boundaries) [24]. Therefore, it is proposed to determine the improvement associated with the consideration of specific submatrices. This will also affect the quality of the sections, which can be separated from the same submatrices. In this context, controlled (simulated) offline learning is used to approach the optimal solution of the CO problem related to the choice of submatrix, and subsequently trained NN can be trained pretty quickly (on small amount of training input data) to find potentially best submatrices – and you wouldn't have the computational burden of solving NP-hard tasks. Of course, the potentially best submatrices should lead us to the potentially best cut-out planes and, according to [24], it is possible to train NNs exclusively to solve the problem, to add only the most promising cut-off planes.

After analyzing significant amount of heuristics, it could be said that a well-functioning approach is the use of strong branching [25]. When using it, before making any decision on branching, the strong branching algorithm performs one step forward, pre-considering the approximate branching on many potential variables, calculates the current solution by the maximum likelihood method to obtain a potential improvement of the current solution, and finally the use of strong branching together with the method of branches and borders provides the best improvement of the output. However, even if you do not study all the variables, but only approximate the current value of the solution, it will be still a strategy, leading to requiring a lot amount of time, needed for computations.

Table 1

The results of approaches to use NN for solving UAV CO problems comparison

ML usage approach	Pros	Cons	Where to use
Simulation training	Can avoid local minimums May be trained quicker than with RL	Demands expert knowledge for whole training process	Well known CO problems with known optimal distribution of source data
Reinforcement learning	Can solve problems with unknown optimal data distribution No need of expert knowledge	Mostly demands more time for training, than simulation training	CO problems with unknown optimal distribution of source data

7.4. Practical application

Considering the task of the TSP, that can be shown on the graph described above, it is easy to make a greedy heuristic that builds a route, consistently selecting nodes among those that have not yet been visited, and therefore, building a permutation. The choice of the nearest node is an intuitive choice of criterion, but in most cases, it is far from optimal. Therefore, a greedy heuristic structure is built, where the node selection policy is studied using NN on a graph, a type of NN capable of processing at the input of a graph of any limited size using a message transmission mechanism [9]. Thus, to select each node, the input of the NN getting a graphical image of the problem, supplemented by functions that marking nodes, that travelling salesman already visited (to not visit them twice). At the output of the NN issues the selected next node. This selected node is then used as the training example for the whole NN (based on the RL), and the partial length of the route is used as a reward.

Of course, there are much better algorithms that allow you to get better solutions. However, it should be noted that given the above information, NN can be used to identify new, potentially more effective policies.

On a two-dimensional Euclidean graph containing up to 100 vertices, the RL for the travelling salesman task significantly exceeds the controlled approach to learning [3] and allows to obtain results close to optimal, if enough time was used for learning. These results provide insights into how NN can be used as a tool to solve CO problems, especially those for which heuristics are difficult to develop.

On KnapSack problem, which is NP-hard, in [4] authors used pretrained NN with RL and active search to successfully solve KNAP50, KNAP100 and KNAP200 problems all instances to optimality.

8. Conclusions

Thus, the paper reviews the approaches to the use of NN in the problems of CO. It is determined that the use of NN (including deep learning NN) is possible in the tasks of CO for UAVs in routing problems, so they can be used to solve problems that arise when planning the operation of UAVs. At the same time, recurrent NNs with nonparametric normalized exponential functions of controlled learning may be undoubtedly used for finding solutions of the CO problems. It has been determined that the most effective to date is the use of RL training to solve such problems. This method is recommended to use.

Prospects for the use of research results: preparing mathematical model of the specific CA problem for UAVs, developing an algorithm for solving this problem using the proposed NN architecture and comparing the results with the results of classical methods for solving such CA problem.

9. References

- [1] A. Naseem, S. T. H. Shah, S. A. Khan, A. W. Malik 2017 Decision support system for optimum decision making process in threat evaluation and weapon assignment: Current status, challenges and future directions *Annual reviews in control* 43, 169-187. <https://doi.org/10.1016/j.arcontrol.2017.03.003>
- [2] L. F. Hulyantskyi, I. I. Riasna 2017 Formalization and classification of combinatorial optimization problems *Optimization Methods and Applications (eds. Butenko S., Pardalos P. M., Shylo V.)*, Cham: Springer International Publishing AG 239–250 https://doi.org/10.1007/978-3-319-68640-0_11
- [3] Y. Bengio, A. Lodi, A. Prouvost 2020 Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon *European Journal of Operational Research* 290(2), pp. 405-421 <https://doi.org/10.1016/j.ejor.2020.07.063>
- [4] I. Bello et al. 2016 Neural combinatorial optimization with reinforcement learning, *arXiv preprint arXiv: 1611.09940*.
- [5] M. Fortun, S. S. Schweber 1993 Scientists and the legacy of World War II: The case of operations research (or) *Social Studies of Science* 23(4), pp. 595–642 <https://doi.org/10.1177/030631293023004001>
- [6] L. Wang, S. Li, F. Tian, F. Xiuju 2004 A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34(5), pp. 2119-2125 <https://doi.org/10.1109/tsmcb.2004.829778>
- [7] E. Khalil et al. 2017 Learning combinatorial optimization algorithms over graphs *Advances in Neural Information Processing Systems* 6348-6358 <https://doi.org/10.5555/3295222.3295382>
- [8] Z. Li, Q. Chen, V. Koltun 2018 Combinatorial optimization with graph convolutional networks and guided tree search *Advances in Neural Information Processing Systems* 539-548.

- [9] K. Lei, P. Guo, Y. Wang, X. Wu, W. Zhao 2021 Solve routing problems with a residual edge-graph attention neural network *arXiv preprint arXiv:2105.02730*.
- [10] K. Braekers, K. Ramaekers, I. V. Nieuwenh 2016 The Vehicle Routing Problem: State of the Art Classification and Review *Computers & Industrial Engineering* 99, 300–313 <https://doi.org/10.1016/j.cie.2015.12.007>
- [11] V. P. Horbulin, L. F. Hulianytskyi, I. V. Sergienko 2020 Optimization of UAV Team Routes in the Presence of Alternative and Dynamic Depots *Cybernetics and Systems Analysis* 56(2), pp. 195–203 <https://doi.org/10.1007/s10559-020-00235-8>
- [12] O. Turchyn 2007 Comparative analysis of metaheuristics solving combinatorial optimization problems 2007 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics. IEEE <https://doi.org/10.1109/cadsm.2007.4297548>
- [13] A. Oliinyk, I. Fedorchenko, A. Stepanenko, M. Rud, D. Goncharenko 2019 Combinatorial optimization problems solving based on evolutionary approach, 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM) pp. 41-45 <https://doi.org/10.1109/cadsm.2019.8779290>
- [14] I. V. Sergienko, L. F. Hulianytskyi, S. I. Sirenko 2009 Classification of applied methods of combinatorial optimization *Cybernetics and Systems Analysis* 45(5), pp. 732-741 <https://doi.org/10.1007/s10559-009-9134-0>
- [15] S. Lin 1965 Computer solutions of the traveling salesman problem *Bell System Technical Journal* 44(10), pp. 2245-2269 <https://doi.org/10.1002/j.1538-7305.1965.tb04146.x>
- [16] A. Hamacher, C. Moll 1996 The Euclidian Traveling Salesman Selection Problem *Operations Research Proceedings* 1995, Springer, Berlin, Heidelberg 54-59 https://doi.org/10.1007/978-3-642-80117-4_10
- [17] R. C. Larson, A. R. Odoni 1981 Urban operations research *No. Monograph*.
- [18] D. Naddef, G. Rinaldi 2002 Branch-and-cut algorithms for the capacitated VRP *The vehicle routing problem. Society for Industrial and Applied Mathematics*, pp. 53-84 <https://doi.org/10.1137/1.9780898718515.ch3>
- [19] J. J. Hopfield, D. W. Tank 1985 "Neural" computation of decisions in optimization problems *Biological Cybernetics* 52(3), pp. 141–152 <https://doi.org/10.1007/BF00339943>
- [20] G. V. Wilson, G. S. Pawley 1988 On the stability of the travelling salesman problem algorithm of hopfield and tank *Biological Cybernetics* 58(1), pp. 63–70 <https://doi.org/10.1007/bf00363956>
- [21] O. Alagoz, H. Hsu, A. J. Schaefer, M. S. Roberts 2010 Markov decision processes: a tool for sequential decision making under uncertainty *Medical Decision Making*, 30(4), pp. 474-483 <https://doi.org/10.1177/0272989x09353194>
- [22] P. Veličković et al. 2018 Graph attention networks *International Conference on Learning Representations* <https://doi.org/10.17863/CAM.48429>
- [23] R. J. Williams 1992 Simple statistical gradient-following algorithms for connectionist reinforcement learning *Machine learning* 8(3-4), pp. 229-256 <https://doi.org/10.1007/BF00992696>
- [24] R. Baltean-Lugojan, R. Misener, P. Bonami, A. Tramontani 2018 Strong sparse cut selection via trained neural nets for quadratic semidefinite outer-approximations *Technical report, Imperial College, London*.
- [25] D. Applegate, R. Bixby, V. Chvátal, and W. Cook 2007 The traveling salesman problem. A computational study *Princeton University Press* <https://doi.org/10.1515/9781400841103>