Local Optima Networks in solving Algorithm Selection Problem for TSP

Wojciech Bożejko[1], Andrzej Gnatowski[1], Teodor Niżyński[1], and Michael Affenzeller[2] and Andreas Beham[2]

[1] Department of Automatics, Mechatronics and Control Systems,
Wrocław University of Technology,
11–17 Janiszewskiego St., 50-372 Wrocław, Poland,
{wojciech.bozejko, andrzej.gnatowski, teodor.nizynski}@pwr.edu.pl
[2] University of Applied Sciences Upper Austria,
Heuristic and Evolutionary Algorithms Laboratory,
Softwarepark 11, 4232 Hagenberg, Austria
{michael.affenzeller,andreas.beham}@fh-hagenberg.at
Johannes Kepler University Linz,
Institute for Formal Models and Verification,
Altenberger Straße 69, Linz, Austria

Abstract. In the era of commonly available problem-solving tools for, it is especially important to choose the best available method. We use Local Optima Network analysis and machine learning to select appropriate algorithm on the instance-to-instance basis. The preliminary results show that such method can be successfully applied for sufficiently distinct instances and algorithms.

Keywords: algorithm selection problem, fitness landscape analysis, local optima networks, travelling salesma problem

1 Introduction

NP-complete problems are among the most researched problems in operational research. Since optimal solutions are usually unobtainable for the real-world instances, heuristic algorithms are commonly used. There is a wide variety of metaheuristic algorithms, successfully employed to solve multiple optimization problems such as: Tabu Search [4, 3, 2], Genetic Algorithm [8] or Simulated Annealing [5]. However, there is no agreement on which algorithm is the best for each task. For example, for the Flexible Job Shop Scheduling Problem, review [6] describes at least 14 categories of optimization algorithms. Within each category, there are multiple specific variations. For commonly used Genetic Algorithm and the Travelling Salesman Problem (TSP), in survey [20], there is a number of techniques presented.

The mentioned challenge is known in literature as the Algorithm Selection Problem (ASP). In order to make an informed choice, one must obtain as much information about the task being solved, as practically possible. To provide the

data describing the instances, Fitness Landscape (FL) analysis (FLA) is usually employed. The gathered data is usually utilized to solve ASP, using machine learning techniques or even simple methods based on correlation. A comprehensive survey over ASP can be found in [10]. In [14], ASP was researched on the example of the Quadratic Assignment Problem. Two different metaheuristic algorithms were considered: Robust Taboo Search and Variable Neighborhood Search. Several well-known classification methods were used to determine the best algorithm for each tested instance, among others: one variable rule learner (OneR), sequential minimal optimization (SMO) of a support vector machine and Gaussian processes (GProc) and linear regression (LR). The features used were chosen from the set of 34 different FL measures. Results were promising with up to 80% of correctly classified instances. Unfortunately, the time required for measuring the features was longer then the optimization algorithm runs, negatively affecting the practicality of the proposed solution. The ASP for QAP was also tackled by Smith-Miles [16], where a meta-learning framework based on simple feed forward neural networks [15] and self-organizing feature maps was used to analyze (and as a result—predict with up to 94% accuracy) the performance of 3 metaheuristic algorithms. In [9], 6 metaheuristics were tested for the protein structure prediction problem. Correlation was used to identify the relationship between several FL measures and performance of the algorithms.

While the mentioned results suggest that FL analysis is a potent tool for solving ASP, due to an enormous amount of data, it is hard to process and share even a sampled FL between researchers. The concept of Local Optima Networks [13] is a promising solution to this problem. LON is a network consisting of locally optimal solutions (nodes) and probabilities of navigating a search process between them (edges). The LON model provides a way to compress the information about the search space and was successfully applied to gathering the data over various optimization problems, such as: TSP [12, 11], QAP [13, 18, 8], or NK [19, 13]. However, unlike FLA, LON analysis has not yet become a recognized method of obtaining the instance features for ASP.

In this paper we investigate a viability of Local Optima Networks analysis in the context of the Algorithm Selection Problem on the example of the Travelling Salesman Problem. The solving algorithm were taken from the Google Optimization Tools (OR-Tools); while test instances were both generated randomly and chosen from TSPLib.

2 Basic concepts

2.1 Travelling Salesman Problem

In this paper, we use the symmetric TSP as a benchmark problem. Since TSP is a widely known in optimization community, it will be only briefly described. For more details, refer to [1]. In TSP, there is a set of cities $\mathcal{M} = \{1, 2, \ldots, n\}$, that must be visited by a *travelling salesman*. A solution is represented by a permutation of cities from the set, constituting a tour. The tour must be closed and pass through each city exactly once. In the researched variant of TSP, the

distance between each pair of cities is equal in both directions. The problem is to find an order of visiting the cities, minimizing the length of the tour.

2.2 Fitness Landscape

A fitness landscape, as described in [17], is a triple $\{S, V, f\}$, where:

- S is a search space, consisting of all solutions. Since usually $|S|$ grows exponentially with the problem size, for larger instances it is impossible to analyze all the solutions. Therefore, in practical applications, various sampling methods are utilized.
- V is a neighborhood function, $V : S \to \mathcal{P}(S)$. For any given solution $s \in S$, function V assigns a set $V(s)$, consisting of the neighbors of s.
- f is a fitness function, $f : S \to \mathbb{R}$. For any given solution $s \in S$, function f assigns a real number evaluating a "quality" of the solution. In this paper, we assume that the values of f are to be minimized.

2.3 Local Optima Network

LON is a network of nodes symbolizing problem solutions, connected by edges with weights reflecting the probabilities of traversing between them, using given search operator.

Nodes. The nodes are local optima, i.e. in their neighborhoods there are no solutions with a lower value of the fitness function. Formally, a solution $s \in S$ is a local optimum if and only if $\forall a \in V(s) \ \big(f(a) \geq f(s)\big)$. We use the classic 2-change neighborhood, also utilized in the tested metaheuristic algorithms. As it is impossible to list all local optima of a reasonably big instance, we used a sampling method, described in sec. 3.1. The set of LON nodes is denoted by N_{LON}.

Edges. There are at least two edge models for LON: basin-transition and escape edges [13]. We selected escape edges, as it is easier to estimate their weights. A directed edge (s, t) between local optima s and t exists only if t can be obtained by applying a kick-operator on s, followed by a hill stepping algorithm (here—first-improvement 2-opt). The weight of an edge is a number, reflecting the probability of transition from s to t; and is estimated during sampling process. The set of LON edges is denoted by E_{LON}.

2.4 Algorithm Selection Problem

The Algorithm Selection Problem is a problem of selecting, from a given set, the best algorithm on an instance-by-instance basis. In this paper, as a method of comparing the algorithms, we measure the best result obtained after a fixed calculation time.

Algorithm 1: Sampling LON nodes

Data : I_{nmax}, the desired number of nodes;
I_{natt}, the number of node generation attempts;
a TSP instance

Result: N_{LON}, the set of LON nodes

```
1 N_LON ← {};
2 for i = 1, 2, ..., I_nmax do
3     for i = 1, 2, ..., I_natt do
4         s ← generateRandomSolution();
5         s ← 2-opt(s);
6         if s is a local optimum then
7             if s ∉ N_LON then
8                 N_LON ← N_LON ∪ {s};
9                 break
```

3 LON extraction and analysis

3.1 Sampling method

Due to a very large search space, the LON nodes and edges are obtained by a sampling method, similar to the one described in [8]. However, the method was slightly modified to fit a smaller computational budget.

The algorithm 1 describes the method of obtaining LON nodes. First, a random solution $s \in S$ is generated. Then, the solution is optimized by a greedy descend algorithm—2-opt (a classic heuristic for TSP [7]). If the solution cannot be further improved by the 2-opt, and is unique; it becomes a node. Otherwise, another random solution is generated. Parameters I_{nmax} and I_{natt} determine the desired number of nodes in LON and the number of attempts to generate each node. We decided to set $I_{nmax} = 1000$ and $I_{natt} = 10000$. The values were tuned so that the sets of the local minima of the smallest random instances are thoroughly sampled. Following the trends in LON research, we also tested larger values of $I_{nmax} = 10000$ and $I_{natt} = 10000$ for the TSPLib as well as larger random instances.

Algorithm 2 summarizes the sampling process of LON edges. For each node $s \in N_{LON}$ in LON, a kick-move is applied to the related solution. The kick is defined as $k = 2$ random 2-change moves performed one by one. The obtained tour s' is optimized by a first improvement descending algorithm (modified 2-opt, the first-improvement strategy was chosen because it performed better in [8]). If the solution can be found in N_{LON}, the edge (s, s') is added to the set of edges. The process is repeated I_{eatt}-times for each node. We set $I_{nmax} = 100000$ for $I_{nmax} = 1000$ and $I_{nmax} = 10000$ for $I_{natt} = 10000$. Again, the numbers were tuned for the smallest instances, where for any solution $s \in S$, the size of the

neighborhood is $|V(s)|^2 =,\ s \in S$

$$|V(s)|^2 = \left(\frac{1}{2}(n-2)(n-1+1)\right)^2 = \frac{1}{4}n^2(n-2)^2.$$

The weight of an edge (s, s') is equal to the number of additions of (s, s') to E_{LON} during sampling process. Therefore, the bigger the weight of an edge s, s' is, the more probable the transition between solution s and s' is.

Algorithm 2: Sampling LON edges

Data : N_{LON}, the set of LON nodes;
I_{eatt}, the number of random kicks applied to each node;
a TSP instance
Result: E_{LON}, the set of LON edges;
the weights of LON edges

```
1 E_LON ← {};
2 set the weight of each possible edge to 0;
3 foreach s ∈ N_LON do
4     for i = 1, 2, ..., I_eatt do
5         s' ← applyRandomKick(s);
6         s' ← firstImprovement_2-opt(s');
7         if s' ∈ N_LON then
8             E_LON ← E_LON ∪ {(s, s')};
9             increase the weight of {(s, s')} by 1;
```

3.2 Measures

We measured various LON parameters, including:

edgeToNode — the edge to node ratio, $edgeToNode = \frac{|E_{LON}|}{|N_{LON}|}$,
escRate — the average number of kick moves required to leave a local optimum,
numSubSinks — the number of subsinks. Solution $s \in N_{LON}$ is a subsink if and only if it has no outgoing edges to the solutions with the lower value of the fitness function, $\forall (s, i) \in E_{LON}\ \big(f(i) \geq f(s)\big)$,
distLO — the average distance form each node to the node s^* with the lowest value of the fitness function. The distance between any two nodes is defined as reciprocal of the corresponding edge weight. The nodes not connected to s^* are omitted.
conRel — the number of nodes connected to s^* to the number of nodes not connected to s^* ratio,
assortativity — the measure of preference for LON nodes to be connected with similar nodes (nodes with similar number of in-going or out-going nodes, values of fitness function),

clustering — global clustering coefficient, calculated with the `grap-tools` package for Python.

4 Experiments

4.1 Empirical setting

Test instances. The most popular test instances for TSP are gahtered in TSPLIB[3]. We chose 19 relatively small ones: `eil51`, `berlin52`, `st70`, `pr76`, `eil76`, `rat99`, `rd100`, `kroA100`, `kroB100`, `kroC100`, `kroD100`, `kroE100`, `eil101`, `lin105`, `pr107`, `pr124`, `ch130`, `pr136`, `pr144`, with the sizes varying from from 51 to 176 cities. To further diversify the dataset, we generated random instances consisting of 30, 50 and 100 uniformly distributed cities (rnd30, rnd50, rnd100); 30 for each instance size.

Algorithms. We chose a commonly available set of metaheuristic algorithms, Google Optimization Tools (OR-Tools)[4]. OR-Tools allowed the use of various algorithms without implementation concerns. Moreover, the toolbox provides automatic mode, which is supposed to choose the appropriate algorithm for a given task. The initial idea was to compare the selection mechanism, to the one proposed in this paper. For each problem instance, we used the following search options, with default settings: Automatic (Auto), Greedy Descent (GD), Guided Local Search (GLS, most efficient for solving vehicle routing problems according to OR-Tools documentation), Simulated Annealing (SA), Tabu search (TS), Objective Tabu Search (OTS).

To evaluate the algorithms, we launched each one for 1 second for each instance (the short calculation time is due to the small size of the problems). We calculated the relative quality of the obtained solutions from equation

$$\Delta(s) = \frac{f(s) - f(s^*)}{f(s^*)} \cdot 100\%,$$

where s^* is the best known solution for the instance and f is the fitness function (see Table 1).

The results eliminated the possibility of comparing our method of solving ASP to the solution from OR-Tools. Automatic mode yield nearly the same results as GD, one of the worst performing algorithms (no differences between Auto and GD in Table 1). Moreover, GLS algorithm provided the best results for the vast majority of instances. Thus, the task of choosing the best algorithm for TSP from those available in OR-Tools is trivial.

This fact, however, does not eliminate the possibility of using LON analysis to select the appropriate algorithm from a smaller algorithm portfolio. For the ASP to be meaningful, the portfolio should contain complementary solving methods.

[3] `https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`
[4] `https://developers.google.com/optimization/`

Table 1. Average relative quality of the solutions obtained by the algorithms.

Instances	Average Δ [%]					
	Auto	GD	GLS	SA	TS	OTS
rnd30	2,525	2,525	0,000	2,513	0,152	0,206
rnd30,50,100	2,304	2,304	0,093	2,408	0,532	0,381
TSPLib	3,527	3,527	0,172	3,458	2,822	1,063

Table 2. Pairwaise comparison of the algorithms performance.

Dataset	Result	Algorithms														
		1v2[1]	1v3	1v4	1v5	1v6	2v3	2v4	2v5	2v6	3v4	3v5	3v6	4v5	4v6	5v6
TSPlib	1st won	0	0	**4**	4	3	0	**4**	4	3	**10**	**9**	**7**	1	1	1
	draw	19	5	**7**	6	5	5	**7**	6	5	**2**	**2**	**2**	15	10	11
	2nd won	0	14	**8**	9	11	14	**8**	9	11	**6**	**7**	**9**	2	8	7
rnd	1st won	0	0	**34**	19	15	0	**34**	19	15	69	48	**36**	1	1	8
	draw	90	25	**28**	27	15	25	**28**	27	15	14	32	**34**	59	30	43
	2nd won	0	65	**28**	44	60	65	**28**	44	60	7	10	**20**	30	59	39

1-Auto, 2-GD, 3-GLS, 4-SA, 5-TS, 6-OTS. Results suggesting that the algorithms are complementary bold. 1v2 column contains the comparison of the best results of algorithms 1 and 2.

Let us consider a pair of algorithms A and B. The number of instances in which algorithm A outperformed B is denoted by $w(A, B)$, while the number of draws, by $d(A, B)$. We assumed that A and B are complementary, when $w(B) \geq w(A) \geq 0.5w(B)$ or $w(A) \geq w(B) \geq 0.5w(A)$ and $d(A, B) \leq w(A) + w(B)$. The results the described pairwise comparison is presented in Table 2.

4.2 Local Optima Networks

Local Optima Networks were generated with the method described in sec. 3.1. For the random instances with $n = 30$ and $n = 50$ cities, we sampled LONs with $I_{nmax} = 1000$ nodes and $I_{eatt} = 10000$ edge-creating attempts. For rnd100 and for TSPLib instances, the sampling parameters were set for $I_{nmax} = 10000$ and $I_{eatt} = 10000$.

One can define many different measures, capturing specific aspects of LON. However, some measures are correlated, causing the need for a feature selection. To eliminate redundant data, we calculated Spearman correlation coefficients for each pair of LON measures (listed in sec. 3.2) and each test instance. The correlation matrix is shown in Figure 1.

The values of assortativity measures correlates with each other in a significant way, therefore only one is used for further analysis. There is also a clear correlation between the size of the instance n and most of the other measures. This is due to the lesser variation in the value of the measures among instances

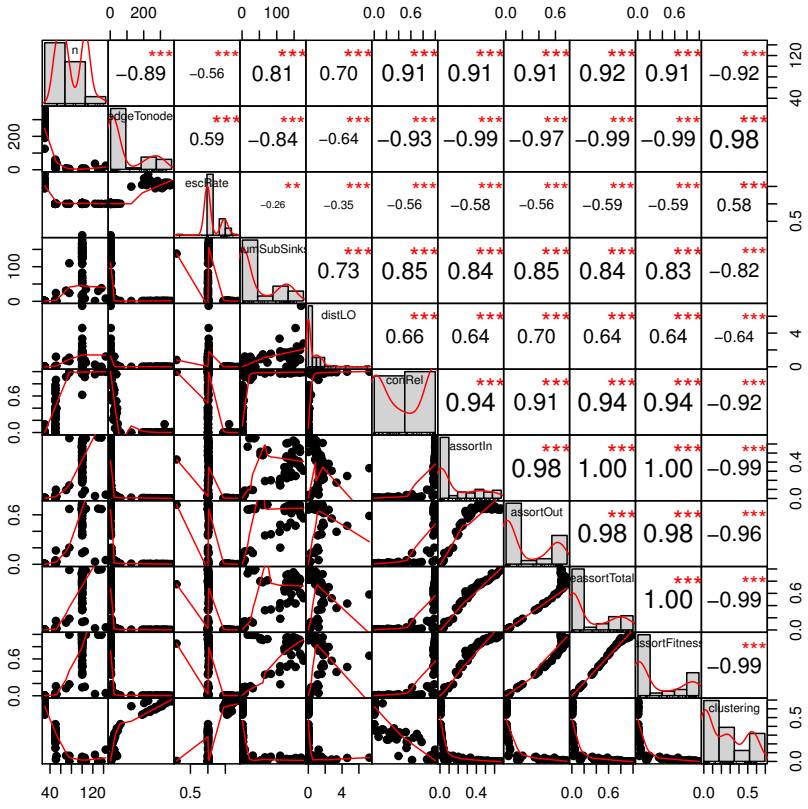

Fig. 1. Correlation matrix for all LON measures. Lower triangle: relationship scatterplots. Diagonal: histograms. Upper triangle: Spearman correlation coefficient. P-values are indicated by asterisks: *** for $p < 0.001$, ** for $p < 0.01$ and * for $p < 0.1$.

of the same size. Such phenomenon is probably caused by the bias induced by the LON sampling method. For example, an average *edgeToNode* ratio for random instances with $n = 30$ and 1000-node LON is 282, while for $n = 50$—30. For $n = 100$ and 1000 nodes, almost no edges (that are not self-loops) are present. Increasing the size of LON to 10000 sampled nodes yields in average *edgeToNode* $= 2.6$; while the intuition suggests, that with the increase of n, *edgeToNode* should also increase. The problem was also signalized e.g. in [12], where the influence of sampling effort on the values of measures is shown. Unfortunately, there is no commonly accepted sampling method so far, that we know of, to avoid such an effect.

4.3 Algorithm Selection Problem for TSP

As the ASP algorithm portfolio, we took pairs of complementary algorithms from Table 2. For each pair of algorithms A-B, the problem was to predict if algorithm A outperforms B, B outperforms A, or if they will provide the same results; using features from sec. 3.2. We chose classifiers from WEKA[5] for this task. The software allows an easy use of many machine learning classifiers, from the simple ones, like NaiveBayes or DecisionStump, to more the sophisticated MultilayerPerceptron or RandomForest. As a comparison baseline, zeroR classifier was used—a simple method which always predicts the majority category class (as a side effect, it also indicates how unbalanced the dataset is). The percentage of correctly predicted instances was measured.

For TSPLib instances, and the algorithm pairs: GLS-SA and GLS-TS, the differences between the prediction performances of zeroR and other classifiers were lower then 5%. However, for the GD (or Auto) and SA algorithms, classifier scored 57.9% correct predictions, while zeroR scored only 42.1%. For the GLS-OTS pair, zeroR was outperformed by 21.1%, 57.9% to 36.8%. Also worth mentioning are the predictions for the SA-OTS and TS-OTS pairs: 73.7% and 68.4% respectively, while zeroR scored 52.6% and 57.9%. Unfortunately, these pairs of algorithms were not complementary. The classifier was only able to determine whether the algorithms would give the same result or whether the better-performing one would win. Randomly generated instances proved to be much harder to be classified. No classifier outperformed zeroR by more then 5% for each algorithms pair. Probably the differences between instances were too subtle to be captured by a simple LON sampling method used in this paper.

5 Conclusions

We presented a method of solving the Algorithm Selection Problem for the Travelling Salesman Problem. As our main contribution, we investigated using Local Optima Network analysis to provide feature extraction of the problem instances. The initial results suggests, that such method can be successfully applied, provided well diversified instances and complementary solving algorithms. Unfortunately, the computation time required for sampling LON is still longer than that of the solving the instance itself. An interesting observation was the poor quality of the automatic algorithm selection mechanism provided by the researched OR-Library, leaving much room for improvement.

In further studies, we plan to investigate the relationship between the LON sampling method and the values of LON measures. Due to the intense computational demand, this part of LON analysis, in our opinion, requires more attention from the research community.

[5] `https://www.cs.waikato.ac.nz/~ml/weka/`

References

[1] Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton Series in Applied Mathematics, Princeton University Press, Princeton, NJ, USA (2006)

[2] Bożejko, W., Gnatowski, A., Idzikowski, R., Wodecki, M.: Cyclic flow shop scheduling problem with two-machine cells. Archives of Control Sciences 27(2), 151–167 (jan 2017)

[3] Bożejko, W., Gnatowski, A., Niżyński, T., Wodecki, M.: Tabu search algorithm with neural tabu mechanism for the cyclic job shop problem. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 409–418. Springer International Publishing, Cham (2016)

[4] Bożejko, W., Gnatowski, A., Pempera, J., Wodecki, M.: Parallel tabu search for the cyclic job shop scheduling problem. Computers & Industrial Engineering 113, 512–524 (nov 2017)

[5] Bożejko, W., Pempera, J., Wodecki, M.: Parallel simulated annealing algorithm for cyclic flexible job shop scheduling problem. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 603–612. Springer International Publishing, Cham (2015)

[6] Chaudhry, I.A., Khan, A.A.: A research survey: Review of flexible job shop scheduling techniques. International Transactions in Operational Research 23(3), 551–591 (2016)

[7] Croes, G.A.: A method for solving traveling-salesman problems. Operations Research 6(6), 791–812 (1958)

[8] Iclanzan, D., Daolio, F., Tomassini, M.: Data-driven local optima network characterization of QAPLIB instances. In: Proceedings of the 2014 conference on Genetic and evolutionary computation—GECCO'14. pp. 453–460. ACM Press, New York, New York, USA (2014)

[9] Jana, N.D., Sil, J., Das, S.: Selection of appropriate metaheuristic algorithms for protein structure prediction in AB off-lattice model: a perspective from fitness landscape analysis. Information Sciences 391392, 28–64 (2017)

[10] Kotthoff, L.: Algorithm selection for combinatorial search problems: A survey. In: Bessiere, C., De Raedt, L., Kotthoff, L., Nijssen, S., O'Sullivan, B., Pedreschi, D. (eds.) Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach, pp. 149–190. Springer International Publishing, Cham (2016)

[11] Ochoa, G., Veerapen, N.: Additional Dimensions to the Study of Funnels in Combinatorial Landscapes. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference—GECCO'16. pp. 373–380. ACM Press, New York, New York, USA (2016)

[12] Ochoa, G., Veerapen, N.: Mapping the global structure of TSP fitness landscapes. Journal of Heuristics pp. 1–30 (may 2017)

[13] Ochoa, G., Verel, S., Daolio, F., Tomassini, M.: Local Optima Networks: A New Model of Combinatorial Fitness Landscapes. In: Richter, H., Engelbrecht, A. (eds.) Recent Advances in the Theory and Application of Fitness Landscapes, chap. 9, pp. 233–262. Springer Berlin Heidelberg (2014)
[14] Pitzer, E., Beham, A., Affenzeller, M.: Automatic Algorithm Selection for the Quadratic Assignment Problem Using Fitness Landscape Analysis. In: Proceedings of the 13th European Conference on Evolutionary Computation in Combinatorial Optimization, pp. 109–120. EvoCOP'13, Springer-Verlag, Berlin, Heidelberg (2013)
[15] Smith-Miles, K.A.: Neural networks for prediction and classification. In: Wang, J. (ed.) Encyclopaedia of Data Warehousing and Mining, pp. 865–869. Information Science Publishing (2006)
[16] Smith-Miles, K.A.: Towards insightful algorithm selection for optimisation using meta-learning concepts. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). pp. 4118–4124. IEEE (jun 2008)
[17] Stadler, P.F.: Fitness landscapes. In: Lässig, M., Valleriani, A. (eds.) Biological Evolution and Statistical Physics, pp. 183–204. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
[18] Thomson, S.L., Ochoa, G., Daolio, F., Veerapen, N.: The effect of landscape funnels in QAPLIB instances. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion on—GECCO'17. pp. 1495–1500. ACM Press, New York, New York, USA (2017)
[19] Tomassini, M., Verel, S., Ochoa, G.: Complex-network analysis of combinatorial spaces: The nk landscape case. Physical Review E 78(6), 066114 (2008)
[20] Vaishnav, P., Choudhary, N., Jain, K.: Traveling Salesman Problem Using Genetic Algorithm: A Survey. International Journal of Scientific Research in Computer Science, Engineering and Information Technology 3(3), 105–108 (2017)