Instance-Based Algorithm Selection on Quadratic Assignment Problem Landscapes

Andreas Beham
[1] University of Applied Sciences Upper Austria
Softwarepark 11
Hagenberg, Austria 4232
[2] Institute for Formal Models and Verification
Johannes Kepler University
Altenberger Straße 69
Linz, Austria 4040
andreas.beham@fh-ooe.at

Michael Affenzeller
[1] University of Applied Sciences Upper Austria
Softwarepark 11
Hagenberg, Austria 4232
[2] Institute for Formal Models and Verification
Johannes Kepler University
Altenberger Straße 69
Linz, Austria 4040
michael.affenzeller@fh-ooe.at

Stefan Wagner
University of Applied Sciences Upper Austria
Softwarepark 11
Hagenberg, Austria 4232
stefan.wagner@fh-ooe.at

ABSTRACT

Among the many applications of fitness landscape analysis a prominent example is algorithm selection. The no-free-lunch (NFL) theorem has put a limit on the general applicability of heuristic search methods. Improved methods can only be found by specialization to certain problem characteristics which limits their application to other problems. This creates a very interesting and dynamic field for algorithm development. However, this also leads to the definition of a large range of different algorithms that are hard to compare exhaustively. An additional challenge is posed by the fact that algorithms have parameters and thus to each algorithm there may be a large number of instances. In this work the application of algorithm selection to problem instances of the quadratic assignment problem (QAP) is discussed.

CCS CONCEPTS

•**Information systems** → *Expert systems; Learning to rank;* •**Theory of computation** → *Facility location and clustering;* •**Computing methodologies** → *Randomized search;*

KEYWORDS

fitness landscapes, algorithm selection, quadratic assignment problem, feature extraction

ACM Reference format:
Andreas Beham, Michael Affenzeller, and Stefan Wagner. 2017. Instance-Based Algorithm Selection on Quadratic Assignment Problem Landscapes. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082513

GECCO '17 Companion, Berlin, Germany

978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3082513

1 INTRODUCTION

The algorithm selection problem (ASP) [13, 20] is concerned with the choice of an efficient algorithm for a set of problems. In heuristic optimization, this problem is complicated due to the vast number of different algorithms. In a strict interpretation every complete parametrization would need to be considered as a separate algorithm. In this work we speak of a complete parametrization as an algorithm instance. In the performance comparison we must further detail these instances by their implementations. Only the implementation can be tested and compared with each other.

A further difficulty is the high amount of different problem instances. Similar to the algorithms, each new parametrization of a problem introduces a new instance which in turn needs to be considered as a problem of its own. Rice describes the possibility of using features to approximate the problem space [20] and perform the ASP on the features instead of on the original space of problems. Extracting useful features is however a difficult task and potentially has to be done for each problem anew.

1.1 Fitness Landscape Analysis

In this regard, fitness landscape analysis (FLA) is a promising method as a more generalizable approach to feature extraction. The features obtained by FLA may be used for the characterization of landscapes. A fitness landscape is given by the triple

$$(S, N, f)$$

where S denotes the set of points $s \in S$ in the landscape, $N : S \to S^n$ denotes the neighborhood function and $f : S \to \mathbb{R}$ denotes the fitness evaluation of each point. There are two potential approaches to characterizing such landscapes:

(1) Exact landscape analysis
(2) Exploratory landscape analysis

In the area of exact analysis a certain neighborhood and fitness function is fixed and studied mathematically. For instance, Chicano et al. show that the autocorrelation coefficient of the swap neighborhood may be computed in polynomial time for instances of the quadratic assignment problem (QAP) [4].

In the area of exploratory analysis samples of the landscape are analyzed. For instance, Pitzer et al. describe three sampling

strategies and corresponding measures for characterization [19]. Merz and Freisleben describe FLA analysis for the QAP, but use problem specific measures only [16]. Mersmann et al. describe algorithm selection for real-valued problems [15].

The goal of such a characterization is to obtain a feature vector that represents a certain problem instance. Ideally, these feature vectors describe properties that are relevant to optimization methods such that their performance is a dependent variable on the degree that a certain feature is observed. The rationale behind this approach is that the various metaheuristics exploit different shapes of fitness landscapes. For instance, iterated local search is efficient when local optima are not much worse than the global optimum.

Further work in fitness landscape analysis concerns the structure and connection between local optima such as the study of so called local optima networks [18]. Closely related to algorithm selection is also the topic of performance prediction [14].

2 FEATURE EXTRACTION

In previous publications [19] a number of sampling strategies have been introduced in the form of walks. These use the neighborhood relationship N of the defined fitness landscape to move from one solution to another which in turn results in a trail through the fitness landscape. To each solution there is an associated fitness or quality value which thus gives the quality trail. They mainly differ in the selection strategy that is used to pick the succeeding solution from the set of neighboring solutions.

- *Random walks* pick a random solution from the neighborhood
- *n-Adaptive walks* pick the best out of a random sample of n solutions from the neighborhood
- *n-Up/down walks* similar to n-adaptive walks, but when no better (or worse) solution is found in the randomly chosen sample the search direction is inverted
- *n-Neutral walks* choose that solution out of a randomly chosen neighborhood sample of size n that is most similar in quality

These sampling strategies are then run for a number of iterations and the resulting quality trail is then used to calculate a number of different features:

- Autocorrelation(1)
- Correlation length
- Information content
- Partial information content
- Density basin information
- Information stability
- Diversity
- Regularity
- Total entropy
- Peak information content
- Peak density basin information
- Down walk length (Up/down walks only)
- Up walk length (Up/down walks only)
- Up walk length variance (Up/down walks only)
- Down walk length variance (Up/down walks only)
- Upper variance (Up/down walks only)
- Lower variance (Up/down walks only)

Algorithm 1 Directed walk

```
1:  procedure DirectedWalk(↓ fit(), ↓ dist(), ↓ N(), ↓ s_0, ↓ s_t)
2:      s ← s_0
3:      while dist(s, s_t) > 0 do
4:          trail.Add(s, fit(s))
5:                  ▷ Calculate the restricted neighborhood N^r
6:          N^r ← {s' ∈ N(s) | dist(s', s_t) < dist(s, s_t)}
7:              ▷ Choose the neighboring solution with best fitness
8:          s ← arg min_{s*∈N^r} fit(s*)
9:      end while
10:     trail.Add(s_t, fit(s_t))
11:     return trail
12: end procedure
```

The downside of this approach is that quite a large number of function evaluations have to be spent for analyzing the fitness landscape rather than for optimization. In addition, as the sample strategies are mutually exclusive due to the different strategies for selection all these walks have to be performed in parallel and then obtain quite a large vector of characteristic information.

Additionally, there are problem specific characteristics that can be computed without sampling given the problem specific data, i.e. the weight and distance matrix of the given QAP instance. Characteristics such as flow dominance have been analyzed in past publications [21], but more may be defined [16]. Here we consider three different characteristics:

- Dominance
- Sparsity
- Asymmetry

These can be applied to both the weight and the distance matrix and results in a total of 6 characteristics. Dominance is basically the coefficient of variation of the matrix' values. Sparsity is the relative number of cells $a_{ij} == 0$ to the total number of cells $n \cdot n$ and asymmetry is the relative number of pairs $a_{ij} \neq a_{ji}$ to the total number of pairs $n \cdot (n-1)/2$.

2.1 Exploratory Analysis Based on Path Relinking

In path relinking (PR) a trajectory is created that links two solutions in the search space [8], for instance by making a greedy choice in each step. It was evaluated to complement or even replace above mentioned walks with a sampling strategy based on PR [2]. It was observed that given two randomly chosen solutions these trajectories typically follow a "U-shaped" pattern (assuming minimization of the fitness value) and that differences in these curves may be attributed to differences of the problem instances. Thus we can introduce a new exploratory landscape walk as *directed walk* with the algorithmic description given in Algorithm 1.

Given solution s and a target solution s_t path relinking considers a restricted neighborhood $N^r(s)$ that consists only of solutions that are more similar to s_t than s is to s_t. Naturally, this neighborhood becomes smaller and smaller the closer s is to s_t already. The trails observed as the result of directed walks combine landscape properties that can be observed in up/down, adaptive and neutral walks. For instance, in the first phase where the restricted neighborhood

$N^r(s)$ is still comparatively large an improving solution may be present with higher probability and a behavior similar to a downwalk can be observed. In the middle phase of the directed walk $N^r(s)$ consists of fewer solutions, but of comparable quality. If the landscape features plateaus and regions with high neutrality, often it may be observed that the fitness of the solutions remains the same in this phase. In the late phase as s continues to become more and more similar to s_t an upwalk behavior can be observed as the fitness also nears that of s_t (if both the initial s as well as s_t are chosen randomly).

Further advantages of exploratory walks such as the one described here are the possibility for an integration within metaheuristic algorithms [2]. An informed metaheuristic may be created that performs PR as part of its search and analyzes the trajectories to detect a similarity with previously seen problem instances where such an information is already available. Three features are proposed to describe these trails:

- Sharpness
- Bumpiness
- Flatness

Sharpness is the average absolute "gradient" in the quality trail, i.e. the ratio of the absolute quality difference and the distance between two succeeding solutions in the trail.

Bumpiness is the relative number of inflection points to the points visited, i.e. where the "gradient" changes sign, but is not equal to 0.

Flatness is the relative number of undulation points to the points visited, i.e. the gradient itself as well as its "derivative" are both 0. These three simple features describe different aspects of the observed curves and thus of the underlying problem instances.

3 ALGORITHM INSTANCES

The algorithm instances that we applied to the set of problem instances are

(1) Robust Taboo Search (3 instances)
(2) Standard Tabu Search (1 instance)
(3) Variable Neighborhood Search (1 instance)
(4) Iterated Genetic Algorithm (1 instance)
(5) Memetic Algorithms (2 instances)
(6) Multi-start Local Search (1 instance)
(7) Random Search (1 instance)

Some of these instances are dependent on the problem dimension. For instance, the aspiration condition in robust taboo search was seen as a factor of dimension. Often human experts, use such simple information to parameterize an algorithm and thus such a parameterization strategy can also be seen as an algorithm instance of its own.

Other instances were applied in an iterated way. In general, genetic algorithms are often observed to converge after a certain number of generations and even if it is continued to run the probability of achieving a better quality is small as only mutation is able to identify new improvements. In several experiments on the quadratic assignment problem converge can be observed with a few hundred generations. An iterated genetic algorithm is simply restarted several times anew and the best of the restarts is considered as output of the algorithm. This simple strategy also enables to compare algorithm instances over the course of an equal number of solution evaluations.

In the following the algorithm instances shall be explained a little more, though it would be outside the scope of this paper to describe them in all their details.

3.1 Robust Taboo Search

Robust taboo search (RTS) was introduced by Glover to avoid effects where tabu search would be trapped in a cycle visiting the same solutions over and over [22]. In RTS *the length of the tabu tenure is a random variable* and each time a move is made a new tenure is drawn from the probability distribution. In addition an alternative aspiration condition was introduced that would greatly diversify the search. After a certain number of iterations moves that have not been performed in a long time would be considered instead of only the best moves. It has been observed that the distribution parameter of the tabu tenure has less of an effect on the performance of RTS and given some prior experimentation the distribution parameter of the tabu tenure was set to 200. However, the amount of iterations for the alternative aspiration condition to come into effect has a greater influence on the performance. Thus, three instances of this algorithm were tested:

(1) *RTS-NOASP* where the alternative aspiration condition was not used
(2) *RTS-20D* where the iterations parameter affecting the alternative aspiration condition was set to 20 times the dimension
(3) *RTS-100D* where the iterations parameter affecting the alternative aspiration condition was set to 100 times the dimension

3.2 Standard Tabu Search

In contrast to RTS standard tabu search uses a fixed tabu tenure which was set to 3 times the dimension. A simple aspiration criterion was used that allows reverting a move when the quality would be better than the quality at which the move entered the tabu list.

3.3 Variable Neighborhood Search

Variable neighborhood search (VNS) combines a local search heuristic with a perturbation strategy to escape local optima [17]. It uses various neighborhoods in the perturbation phase and tries to start from a neighborhood that attempts a small change to a neighborhood that performs bigger changes to the solution. VNS is elitistic in that it bases its search always on the best solution found so far. In this study an exhaustive local search based on the *swap2* neighborhood was used. The perturbation applies k-swap neighborhoods with an increasing value of k which is limited by the problem dimension halved.

3.4 Iterated Genetic Algorithm

The genetic algorithm (GA) was configured to use 1-elitism, a population size of 500, 15% mutation rate using the swap2 mutation, the partially matched crossover (PMX) [7] and tournament selection with a group size of 3. It runs for 200 generations after which it is restarted until it reaches the total number of evaluated solutions required during the test setup.

3.5 Memetic Algorithm

The memetic algorithm (MA) introduced in [16] is here described as "Genetic Local Search (GLS)" describes a closer interplay between local search and the crossover heuristic. In this algorithm the local search manipulates only those parts of the permutation that were not identical in the parents. It is configured to use a population size of 100 and performs 50 crossovers and 20 mutations per generation. The mutations are performed according to a 2-opt change. The local search uses the swap2 neighborhood in the reduced sub-space given by the two parents.

3.6 Iterated Memetic Algorithm

This is another variant of a memetic algorithm that is closer to the iterated genetic algorithm described above, but uses local search during mutation. It does not use elitism, is configured with a population size of 50 and uses linear rank selection. The local search as in the other algorithm instances uses the swap2 neighborhood. It is restarted when it is converged.

3.7 Multi-start Local Search

Multi-start local search is quickly described as using exhaustive local search in the swap2 neighborhood from randomly drawn starting solutions.

3.8 Random Search

Random search basically evaluates a random sample of the solution space which is drawn with repetition, i.e. no memory is used.

4 BENCHMARK DATA GENERATION

We picked a total of 47 problem instances from the QAPLIB [3], microarray instances [5], Drezner's [6], and Taillard's symmetrical and structured QAP instances[1] in dimensions from 19 to 343. The instances were chosen to include preferably two or more of the same authors or generators in order to have some prior knowledge in form of a similar instance present in the database. However some instances, in particular *els19*, *esc32a*, *had20*, *kra32*, and *nug30* were added without a similar counterpart. The full set of problem instances is shown among others in Table 3.

4.1 Performance Modeling

There is hardly a single measure that can express the performance of metaheuristic algorithms. There are at least two relevant dimensions when looking at algorithm performance.

(1) Achievable Quality
(2) Required Runtime

And they are both interdependent: The quality is a function of the runtime. Typically, algorithm instances are compared by their achieved quality: An algorithm instance is better if it finds better solutions on average - for fairness purposes the allowed computational budget is then fixed and equal for all. Nevertheless, such a ranking could be different if a shorter or longer budget is used and is difficult to generalize. On the other hand we can fix a certain quality that has to be achieved and measure the required runtime to achieve that quality. The runtime is then seen as a random variable for which an expectation exists. This expected runtime (ERT) is then used to compare algorithms [10]. Nevertheless, for a different quality targets again the ranking may be different. Thus in the analysis and comparison multiple such targets are used and the performance of an algorithm instance is described as a vector of ERT. Overall, several arguments are in favor of a comparison based on runtime. As Hansen et al [9] note comparing algorithm instances A and B based on the runtime allows making quantitative statements such as "A is two times faster than B". They conclude this is more difficult to perform when comparing quality as it cannot be argued in general that a fitness value twice as large (or small) is also twice as hard to achieve. In Figure 1 the performance of the algorithm instances described in Section 3 are shown for the 1% target of each of the problem instances.

[1] http://mistic.heig-vd.ch/taillard/problemes.dir/qap.dir/qap.html

5 RECOMMENDATION ALGORITHMS

For recommending algorithm instances we evaluate a k-nearest neighbor-based approach (k-NN). It simply uses Euclidean distance between the problem instances based on the features that have been extracted. All features are normalized to a standard normal distribution $N(0, 1)$. Normalization is performed only for the problem instances that are not used for testing and these normalization parameters are applied to the new problem instance.

The k-NN is an instance-based algorithm that does not require training. It simply uses the available data points and a distance in feature space. Its performance depends on the one hand on the hyperparameter $k \in \mathbb{N}$ and on the other hand by the selected features that comprise the distance. The algorithm instances will be ranked based on the observed rankings of the k closest problem instances. A new ranking will be created by the average of the observed ERT values of each algorithm instance. The smaller the expected runtime the higher the algorithm instance will be ranked. Because the target is fixed, we will not consider looking at the qualities that may be achieved if the algorithm instance may be run longer. A recommendation targeted for a fixed budget would also be possible in which case the ranking would have to be computed given the achieved quality at the fixed budget. While this is also an interesting use case, in this work we will assume that a certain target quality should be achieved and the best algorithm instance is the one that is the fastest to achieve it.

6 EVALUATION

We aim to evaluate the algorithm selection performance on the group of problem instances given in Table 3 using cross validation. In the first tests we use the full set of problem instances for training and then testing each problem instance as if it was to be new to the set. Of course, this means that we have seen the exact same instance in training already. But due to the stochastic nature of the exploratory landscape analysis we may not arrive at exactly the same feature vector. We aim to study the dependency on the time to spend exploring the landscape and the accuracy of the features that we observed.

In a second test the training is conducted using leave-one-out crossvalidation such that the problem instance to test is not part of the training. In this test we analyzed 200 paths out of the directed walk described in Section 2.1. This is closer to the case when the

system would be employed to work in a real-world environment making predictions for unknown problem instances.

The algorithm instances will be clustered into six classes. We are using an optimal 1-dimensional clustering algorithm [23] to cluster these algorithm instances by their $\log_{10}(\mathsf{ERT})$ performance values [1, 10]. Then we sort the clusters by their centroids and assign class 1 to 5 to all instances of that cluster in this order. Thus the classes are ranked so that class 1 contains the best performing algorithm instances. The ERT is computed with respect to reaching a solution quality that is within x% to the best-known quality. The log transform is beneficial to separate the algorithm instances by their performances in orders of magnitude rather than by their absolute differences. Class 6 is specially reserved for algorithm instances that did not achieve the x% target quality in any of the observed runs and thus have an ERT of ∞. This class thus contains all algorithm instances that are unsuitable to achieve the required target and should not be chosen.

Given this set-up we can evaluate recommender algorithms. We will choose the best ranked algorithm instance for each problem instance and lookup in which performance class this algorithm instance was observed. The more often we recommend an algorithm instance of class 1 the better our recommendation algorithm, the more often we recommend an algorithm instance of class 6 the worse. It should also be noted that not all problem instances are of equal "difficulty". While for some problem instances many algorithm instances are in class 1, for other problem instances only a single algorithm instance stood out as the most suited and many if not all others could even be ranked class 6.

6.1 Recommendation Performance Measures

We will use the following measures to show the performance of the proposed recommender:

(1) Normalized Discounted Cumulative Gain (NDCG)
(2) Kendall's τ

These measures are common performance indicators in the domain of information retrieval (IR) and learning to rank (LTR) systems. The recommendation of algorithm instances share some properties with IR in that we also compare two rankings with each other. The first ranking is the one that was observed using actual benchmark data, while the second ranking was obtained through the recommendation algorithm.

The NDCG [11] measure describes the gain of a certain document (in information retrieval) based on its position. A discount is applied in that lower ranked documents are not able to provide the same gain as a higher ranked document. This DCG measure is then normalized by the ideal DCG which would rank each document exactly as it should be ranked. The NDCG thus results in a number between 0 and 1 where a higher value indicates a better ranking system than a lower value. In IR the NDCG is often applied to only the first n ranked documents which is called then NDCG_n measure. In our case documents are algorithm instances and their rank is based on their observed performance.

Kendall's τ [12] is a measure of rank correlation. For all pairs (x_i, y_i) and (x_j, y_j) in two rankings X and Y it is described as the ratio of "agreeing" pairs minus the number of "disagreeing" pairs to the total number of pairs. Two pairs are agreeing in rank if both x_i and y_i are both lower or both higher ranked than the respective x_j and y_j. They are disagreeing if one of them is ranked lower, but the other is ranked higher. The value for τ is in the range $[-1; 1]$ where -1 would indicate that two rankings are exactly opposite to each other and 1 indicates that two rankings are equal to each other. A 0 indicates that there is no correlation among the rankings. Thus, the closer τ approaches 1 the better the correlation between the ranks.

Kendall's τ is more strict with respect to getting the ranking exactly right. In our case, if we're only looking at the top 3 recommended algorithm instances we don't care so much whether the worse algorithm instances are also ranked last. For a system that would apply all algorithm instances starting from the best-ranked to the worst-ranked it may be more important to focus on Kendall's τ than on NDCG which values more that the top ranked algorithm instances are among the best ranked.

7 RESULTS

7.1 Accuracy of Instance Detection

As we stated earlier exploratory landscape analysis (ELA) is a stochastic method that will yield different results depending on the sample that is drawn from the overall landscape. In order to assess the robustness of this approach we will evaluate the accuracy of problem instance detection depending on the amount of effort that is spent during ELA. We aim to verify the assumption that more effort also leads to more precise characteristics. As a baseline we have performed 200 directed walks on each of the landscapes and obtained the three features: Sharpness, bumpiness, and flatness.

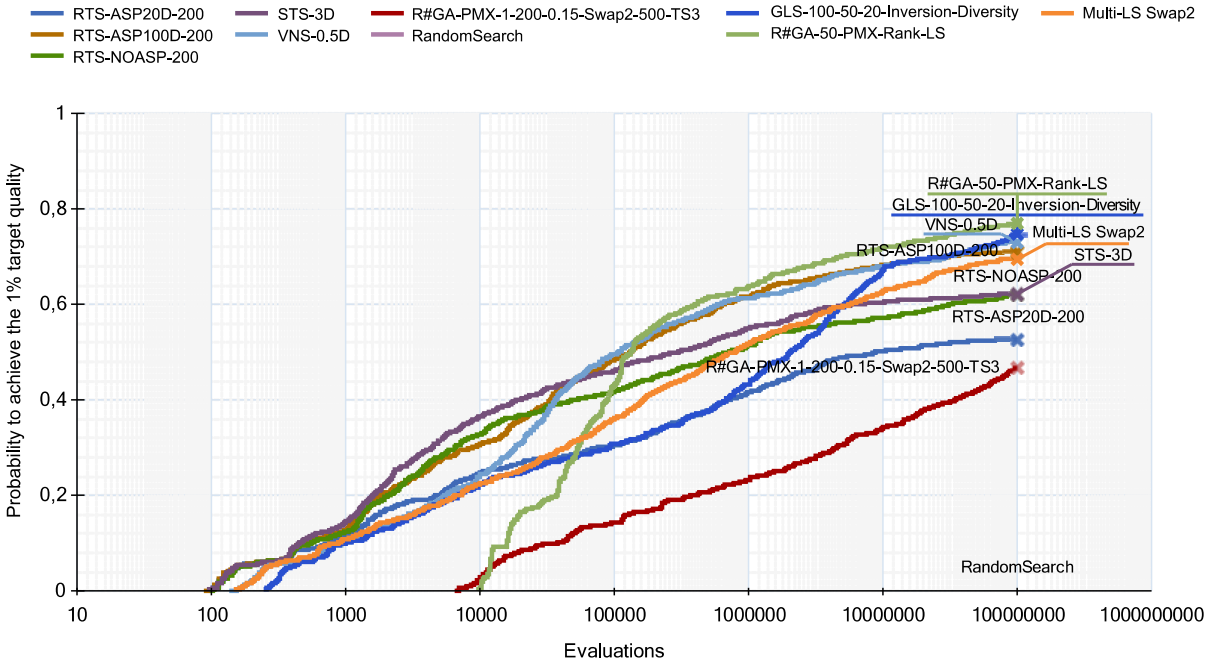
In Table 1 we evaluate the ranking based on this baseline. We take each problem instance, apply directed walk with an increasing number of paths and calculate the characteristics. Given these new characteristics we rank the instances based on the Euclidean distance to the instance characteristics of the baseline data. If the same instance was then ranked as first we record a rank of 0. In Table 1 we state the average rank out of five repetitions averaged over all problem instances. An average rank of 0 would be perfect stating that the instance was always correctly identified.

As can be seen the problem instances can be described rather well by the exploratory analysis. It is possible to identify the same instance again with a high probability using 100 paths already. Of course, even 100 paths may be a lot to compute. However, by using exploratoy analysis techniques that are based on heuristics the potential of integrating them into metaheuristic algorithms allows obtaining these characteristics as a result of the search [2]. It will be the topic of future work to consider the cost of the landscape analysis and contrast this with the savings observed as part of the recommendation.

Table 1: The average rank over 5 repetitions in identifying the correct problem instance through exploratory landscape analysis using Sharpness, Bumpiness, and Flatness as features with a varying number of paths.

	Paths					
	5	**10**	**20**	**50**	**100**	**200**
Average Rank	8.2	7.3	4.6	2.7	1.9	1.4

Figure 1: Runtime analysis of different algorithm instances for the 1% target on the benchmark problem set. ©Andreas Beham



7.2 Real-world like Recommendation

The results given in Table 2 show that an NDCG_n value close to 1 can be achieved. The table compares the NDCG performance for target values given as rows and for several different n given as columns. Also Kendall's τ values show that there is correlation among the observed rank to the recommended rank. We observe that problem specific characteristics are among the best suited for algorithm selection in that we obtain slightly better τ values. Nevertheless, an improvement in NDCG_n could not always be observed. This would lead to the conclusion, that the overall rank is improved, but not that the top n choices are so much better. Additionally, problem specific properties may not transfer to other problems. The other characteristics mentioned are more general and can be computed for any problem. Given the results, the described FLA characteristics sharpness, bumpiness, and flatness also outperform rankings obtained using characteristics of a random walk with 10,000 iterations. It can also be observed that a good recommendation for the 0 or 1% target is harder to achieve. In Table 3 we show in more detail the performance classes for the 5% target of the top 3 ranked algorithm instances for every problem instance. The results suggest that the proposed system is quite well suited to generate a good recommendation in a real-world system. In such a scenario, 46 problem instances would already be present in the knowledge base having recorded the performance of all algorithm instances. The ranking for the "new" problem instance is then computed according to the recommendation algorithm based on the observed performances on the other similar problem instances. This is of course only a snapshot of a real-world situation. In an even more realistic test the problem instances would have to be added one by one and the recommendation would have to be performed for a range of sequences. Again, this topic would be interesting for future studies, but is beyond the scope of this work.

Looking at Table 3 in more detail we can observe that the recommendation includes at least one of class 1 for almost every problem instance in the top 3. For three problem instances (had20, tai20b, and tai100b) we could not include an algorithm instance of class 1 in the top ranked algorithm instances of our recommendation. A total of 17 recommended algorithm instances were deemed unsuitable as they did not once achieve the target quality in 100,000,000 evaluations as observed in the benchmark data set. However, it has to be taken into account that for some problem instances there was not more than 1 suitable algorithm instance available. Given that we always considered the top 3 algorithm instances, 12 out of 17 recommended instances are inevitably class 6 as less than 3 algorithm instances belonging to classes 1-5 were available. Potentially, recommendation should stop recommending algorithm instances when the performance on similar problem instances has been unsatisfying and thus even less than n instances may be recommended. In our case we always output a full ranking of all algorithm instances and then take the top n.

8 CONCLUSIONS AND OUTLOOK

In this work we have shown a study in the field of algorithm instance selection on several problem instances of the quadratic assignment problem. We have introduced several new fitness landscape analysis characteristics and motivated their use. We have

Table 2: The resulting performance measures evaluated for a varying number of ranked algorithm instances. The number n denotes that performance measure if only the top n ranked algorithm instances are taken into account.

Target	NDCG_n						Kendall's τ
	1	2	3	4	5	6	
Sharpness, Bumpiness, Flatness (200 paths)							
0%	0.78	0.84	0.87	0.89	0.90	0.90	0.65
1%	0.80	0.80	0.85	0.85	0.86	0.87	0.61
5%	0.86	0.85	0.88	0.90	0.90	0.90	0.65
10%	0.83	0.85	0.89	0.91	0.91	0.91	0.61
20%	0.80	0.81	0.86	0.87	0.88	0.89	0.58
QAP specific characteristics							
0%	0.77	0.79	0.83	0.86	0.86	0.87	0.67
1%	0.77	0.80	0.83	0.85	0.87	0.88	0.63
5%	0.95	0.94	0.93	0.94	0.94	0.94	0.72
10%	0.93	0.92	0.93	0.93	0.93	0.94	0.70
20%	0.89	0.90	0.92	0.92	0.92	0.92	0.68
Random walk characteristics (10,000 iterations)							
0%	0.72	0.77	0.79	0.81	0.82	0.83	0.53
1%	0.69	0.72	0.74	0.76	0.78	0.79	0.52
5%	0.86	0.84	0.84	0.86	0.86	0.87	0.57
10%	0.86	0.85	0.86	0.87	0.89	0.89	0.59
20%	0.83	0.82	0.84	0.85	0.87	0.87	0.56

shown promising results using a simple k-nearest neighbor recommendation algorithm and evaluated its performance using leave one out crossvalidation measured by NDCG and Kendall's τ.

While a significant amount of time and energy has been invested in creating this study, there are still a large number of open questions. On the one hand we observed that the recommender only works well when we set $k = 1$ and thus implicitely depend on the presence of a very similar problem instance in the knowledge base. In case such a problem instance is not present the recommender's performance deteriorates significantly. Calculating a ranking using multiple neighbors is also non-trivial. One could calculate an average rank or recreate a rank based on the sum of the underlying performance measures such as ERT. We deliberately omitted results regarding the recommender's performance in predicting the actual ERT performance. On the one hand because it would go beyond the scope of the study and on the other hand because the results were not very satisfying. More elaborate recommendation algorithms need to be applied in the future in order to improve the results. So far we can conclude that a nearest neighbor approach with $k = 1$ seems to work quite well in choosing suitable algorithm instances given that our knowledge base is large enough and contains a similar problem instance. Also in the future the effort that has to be spent in analyzing the landscape has to be taken into account. In order to be useful and taking into account the cost of ELA the recommender has to solve the problem instances with less effort than when using a default algorithm instance for all the problem instances.

We hope to be able to provide the data in an open format in the near future so that further studies can be performed.

Table 3: Result of the recommendation per problem instance. In each cell the first number indicates the amount of algorithm instances recommended and the second number indicates the total number of algorithm instances per performance class. Problem instances are marked in bold if the recommendation has been very good.

Problem Instance	Performance Classes 5%					
	1	2	3	4	5	6
bur26a	**3/4**	-/2	-/2	-/1	-/1	
bur26d	**3/4**	-/3	-/1	-/1	-/1	
chr20a	2/3	1/2	-/2	-/1	-/1	-/1
chr20b	1/1	1/2	1/2	-/3	-/1	-/1
chr20c	1/3	1/2	-/2	-/1	1/1	-/1
dre24	**3/3**	-/1	-/1	-/2	-/1	-/2
dre30	**2/2**	**1/1**	-/3	-/1	-/1	-/2
dre56	1/1	-/1	-/1			2/7
dre72	1/1					2/9
dre110	1/1					2/9
els19	1/3	1/2	-/1	-/2	1/1	-/1
esc32a	1/1	1/4	-/1	-/1	1/2	-/1
had20	-/4	2/3	-/1	1/1	-/1	
kra32	2/2	-/3	1/2	-/1	-/1	-/1
lipa20a	**3/4**	-/2	-/1	-/2	-/1	
lipa20b	**1/1**	**2/3**	-/3	-/1	-/1	-/1
lipa50a	2/6	-/1	1/2	-/1		
lipa50b	2/3	-/2	-/1	-/1	-/1	1/2
lipa90a	**3/6**	-/1	-/2	-/1		
lipa90b	**1/1**	**2/3**	-/1	-/1	-/2	-/2
nug30	1/2	1/2	1/3	-/1	-/1	-/1
RAND-S-6x6-...bl	2/3	-/2	1/2	-/1	-/1	-/1
RAND-S-8x8-...ci	**3/4**	-/2	-/1	-/1	-/1	-/1
RAND-S-10x10-...bl	1/4	-/1	1/1	-/1	1/1	-/2
RAND-S-12x12-...ci	**3/4**	-/2	-/1	-/1	-/1	-/1
sko56	**3/4**	-/2	-/1	-/1	-/1	-/1
sko90	**3/4**	-/2	-/1	-/1	-/1	-/1
tai20a	**3/4**	-/2	-/1	-/1	-/1	-/1
tai20b	-/1	1/2	1/3	1/3	-/1	
tai50a	**3/4**	-/1	-/2	-/1	-/1	-/1
tai50b	1/3	1/2	1/2	-/1	-/1	-/1
tai100a	**3/4**	-/2	-/1	-/1	-/1	-/1
tai100b	-/2	1/3	1/1	-/1	1/2	-/1
tai27e01	**2/2**	**1/2**	-/2	-/1	-/2	-/1
tai27e10	**2/2**	**1/1**	-/2	-/1	-/3	-/1
tai45e01	1/1	1/2	1/1	-/1	-/1	-/4
tai45e10	1/1	1/2	1/1	-/1	-/1	-/4
tai75e01	1/1	1/1	-/1	1/1		-/6
tai75e10	1/1	1/1	-/1	1/1		-/6
tai125e01	**1/1**	**1/1**				**1/8**
tai125e10	1/1	-/1				2/8
tai175e01	**1/1**	**1/1**				**1/8**
tai175e10	**1/1**					**2/9**
tai343e01	**1/1**					**2/9**
tai343e10	**1/1**					**2/9**
wil50	**3/4**	-/2	-/1	-/1	-/1	-/1
wil100	**3/4**	-/2	-/1	-/1	-/1	-/1
Recommended	80	24	11	4	5	17
Total	119	79	59	45	40	128

ACKNOWLEDGMENTS

The work described in this paper was done within the COMET Project #843532 Heuristic Optimization in Production and Logistics (HOPL) funded by the Austrian Research Promotion Agency (FFG) and the Government of Upper Austria and the COMET Project #843551 Advanced Engineering Design Automation (AEDA) funded by the Austrian Research Promotion Agency (FFG) and the Government of Vorarlberg.

REFERENCES

[1] A. Auger and N. Hansen. 2005. Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC)*, Vol. 2. 1777–1784. DOI: http://dx.doi.org/10.1109/CEC.2005.1554903

[2] Andreas Beham, Erik Pitzer, and Michael Affenzeller. 2017. Integrating Exploratory Landscape Analysis into Metaheuristic Algorithms. In *Proceedings of the 16th International Conference on Computer Aided Systems Theory (eurocast 2017)*. 134–135.

[3] Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl. 1997. QAPLIB – A Quadratic Assignment Problem Library. *Journal of Global Optimization* 10, 4 (June 1997), 391–403. http://www.opt.math.tu-graz.ac.at/qaplib/

[4] Francisco Chicano, Gabriel Luque, and Enrique Alba. 2012. Autocorrelation measures for the quadratic assignment problem. *Applied Mathematics Letters* 25, 4 (2012), 698–705.

[5] Sérgio A. de Carvalho Jr. and Sven Rahmann. 2006. Microarray Layout as Quadratic Assignment Problem. In *Proceedings of the German Conference on Bioinformatics (GCB), volume P-83 of Lecture Notes in Informatics*.

[6] Zvi Drezner, Peter M. Hahn, and Éeric D. Taillard. 2005. Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-Heuristic Methods. *Annals of Operations Research* 139, 1 (2005), 65–94. DOI: http://dx.doi.org/10.1007/s10479-005-3444-z

[7] David B Fogel. 1988. An evolutionary approach to the traveling salesman problem. *Biological Cybernetics* 60, 2 (1988), 139–144.

[8] Fred Glover, Manuel Laguna, and Rafael Martí. 2000. Fundamentals of scatter search and path relinking. (2000).

[9] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, Dejan Tušar, and Tea Tušar. 2016. COCO: Performance assessment. *arXiv preprint arXiv:1605.03560* (2016).

[10] H. H. Hoos and T. Sttzle. 1998. Evaluating Las Vegas Algorithms - Pitfalls and Remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*. Morgan Kaufmann, San Francisco, CA, 238–245.

[11] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[12] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[13] Lars Kotthoff. 2014. Algorithm selection for combinatorial search problems: A survey. *AI Magazine* 35, 3 (2014), 48–60.

[14] Katherine M Malan and Andries P Engelbrecht. 2014. Fitness landscape analysis for metaheuristic performance prediction. In *Recent advances in the theory and application of fitness landscapes*. Springer, 103–132.

[15] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*. ACM, New York, NY, USA, 829–836. DOI: http://dx.doi.org/10.1145/2001576.2001690

[16] Peter Merz and Bernd Freisleben. 2000. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *Evolutionary Computation, IEEE Transactions on* 4, 4 (2000), 337–352.

[17] Nenad Mladenović and Pierre Hansen. 1997. Variable neighborhood search. *Computers & Operations Research* 24, 11 (1997), 1097–1100.

[18] G. Ochoa, S. Verel, F. Daolio, and M. Tomassini. 2014. Local Optima Networks: A New Model of Combinatorial Fitness Landscapes. In *Recent Advances in the Theory and Application of Fitness Landscapes*. Springer-Verlag Berlin Heidelberg.

[19] Erik Pitzer and Michael Affenzeller. 2012. *Recent Advances in Intelligent Engineering Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter A Comprehensive Survey on Fitness Landscape Analysis, 161–191. DOI: http://dx.doi.org/10.1007/978-3-642-23229-9_8

[20] John R. Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65–118. DOI: http://dx.doi.org/10.1016/S0065-2458(08)60520-3

[21] Thomas Stützle. 2006. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research* 174, 3 (2006), 1519–1539.

[22] Eric D. Taillard. 1991. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Comput.* 17 (1991), 443–455.

[23] H. Wang and M. Song. 2011. Ckmeans.1d.dp: optimal k-means clustering in one dimension by dynamic programming. *The R Journal* 3, 2 (2011), 29–33.