Algorithm Selection on Generalized Quadratic Assignment Problem Landscapes

Andreas Beham
[1] Research Group HEAL
FH Upper Austria
Hagenberg, Austria
[2] Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria
andreas.beham@fh-ooe.at

Stefan Wagner
Research Group HEAL
FH Upper Austria
Hagenberg, Austria
stefan.wagner@fh-ooe.at

Michael Affenzeller
[1] Research Group HEAL
FH Upper Austria
Hagenberg, Austria
[2] Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria
michael.affenzeller@fh-ooe.at

ABSTRACT

Algorithm selection is useful in decision situations where among many alternative algorithm instances one has to be chosen. This is often the case in heuristic optimization and is detailed by the well-known no-free-lunch (NFL) theorem. A consequence of the NFL is that a heuristic algorithm may only gain a performance improvement in a subset of the problems. With the present study we aim to identify correlations between observed differences in performance and problem characteristics obtained from statistical analysis of the problem instance and from fitness landscape analysis (FLA). Finally, we evaluate the performance of a recommendation algorithm that uses this information to make an informed choice for a certain algorithm instance.

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, assignment problems

ACM Reference Format:
Andreas Beham, Stefan Wagner, and Michael Affenzeller. 2018. Algorithm Selection on Generalized Quadratic Assignment Problem Landscapes. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205585

1 INTRODUCTION

Algorithm selection is an important task when solving hard optimization problems. The approaches that perform best for a certain problem may vary from instance to instance. A conclusion from the NFL-theorem [35] is that algorithms may only perform better than other algorithms on a subset of the problems and/or instances. Thus, it is an important task in the application of heuristic algorithms to characterize the subsets and choose an accordingly best performing algorithm instance. Even more as we progress to apply heuristic optimization as automated decision makers (ADM) in environments that are only occasionally surveyed by human experts. For instance in logistics scenarios the storage space of some good is determined by an optimization algorithm upon admission automatically. In production, a machine may pick the next job for processing from its buffers, etc. In the light of the digitalization trend and industry 4.0 such scenarios will become more and more common.

The main contributions of this paper to the state of the art are:

(1) Fitness landscape analysis of the generalized quadratic assignment problem
(2) Comparing a range of approaches, including both open source implementations and commercial solvers
(3) Evaluation of a nearest-neighbor-based algorithm selection approach

1.1 Previous Work

One way to pick a suitable algorithm instance is *tuning*. It is attempted to derive a set of problem instances that are expected to be solved in the ADM environment. These instances are then used to compare the performance of algorithm instances. Identifying a well performing algorithm instance may be done by hand through design of experiments [4] or by automated methods such as metaoptimization [5, 17, 22]. With *algorithm selection* the intent is to find a mapping between algorithm instances and problem instances. This approach is described as the algorithm selection problem (ASP) [20, 30]. In heuristic optimization, the ASP is complicated due to a very high number of different algorithm instances, i.e. complete parametrizations of an algorithm and even different implementations which is the actual entity that can be tested and compared with each other. Nevertheless, portfolios of algorithm instances may be better suited to solve a certain group of problems if good discriminators can be identified on which algorithm instance from the portfolio to apply [21, 36].

A further difficulty is given by a large range of different problem instances. Thus, rather than identifying a mapping between concrete elements of the sets of problem instances, it is proposed to

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205585

describe them through a feature vector [30]. This may be achieved by fitness landscape analysis (FLA) which has received much attention in recent years. A fitness landscape is given by the triple (S, N, f) where S denotes the set of solutions $s \in S$ that form the landscape, $N : S \rightarrow S^n$ denotes the neighborhood function and $f : S \rightarrow \mathbb{R}$ denotes the fitness function.

Fitness landscapes may be studied through mathematical analysis. One successful approach is e.g. by decomposing the fitness function into so called *elementary landscapes* [34]. This enables to compute landscape features such as an autocorrelation coefficient exactly [12]. In addition problems may be described in terms of statistical features of the input data, for instance by calculating coefficients of variation, e.g. of a distance matrix [26]. When a mathematical analysis is not tractable or has not yet been performed features may also be obtained by sampling [25, 27, 29]. The later approach is called *exploratory* analysis as the landscape is explored by sampling points and calculating features from the obtained sample. One example would be to perform a random walk and estimate the aforementioned autocorrelation coefficient. Another approach would be to create a local optima network, i.e. a directed graph that describes local optima as nodes and the probability to transition between these in form of weighted edges [27].

2 GENERALIZED QUADRATIC ASSIGNMENT

The generalized quadratic assignment problem (GQAP) is an NP-hard combinatorial optimization problem. Solving this problems requires to find the minimal costs of the assignment of all tasks N to a set of resources M. Each task $t \in N$ has a certain demand $Q_t \in \mathbb{R}$ while each resource $r \in M$ provides a certain capacity $C_r \in \mathbb{R}$. The objective is formed by installation cost $I_{tr} \in \mathbb{R}$ of task t in resource r on the one hand and by the product of communication effort $W_{tu} \in \mathbb{R}$ between two tasks $t, u \in N$ and a communication barrier $D_{rs} \in \mathbb{R}$ between two resources $r, s \in M$. Given a decision variable $x_{tr} \in \{0, 1\}$ that is 1 iff task t is assigned to resource r the GQAP can be described as the following quadratic integer programming problem [28]:

$$\min \sum_{t \in N} \sum_{r \in M} \left(x_{tr} \cdot I_{tr} + e \cdot \sum_{u \in N} \sum_{s \in M} x_{tr} \cdot x_{us} \cdot W_{tu} \cdot D_{rs} \right) \quad (1)$$

$$\text{s.t.} \sum_{t \in N} x_{tr} \cdot Q_t \leq C_r \quad \forall r \in M \quad (2)$$

$$\sum_{r \in M} x_{tr} = 1 \quad \forall t \in N \quad (3)$$

$$x_{tr} \in \{0, 1\} \quad \forall t \in N \wedge r \in M \quad (4)$$

The objective in equation (1) describes to minimize the sum of installation and communication costs given a cost conversion factor e. Equation (2) constrains the capacity of the resource and equation (3) requires that each task *must be* assigned exactly one resource. While the QAP has received much attention in the scientific literature and being mentioned in thousands of publications, the GQAP is still fairly unexplored. Nevertheless, heuristic approaches to solve this problem have been described [13, 24]. The GQAP although it shares the name is different from the QAP: In heuristic optimization the assignment in GQAP is typically encoded as a vector of integers, i.e. encodes the resource that each task is assigned to instead of a permutation as is the case with the QAP. The size of the solution space is M^N as opposed to $N!$ which grows faster when $M = N$ as is the case of the QAP. In addition, solving the GQAP requires handling infeasible solutions which may arise out of a violation of constraint (2). We used a lexicographic fitness function in this work. The infeasible fitness values were offset by a penalty $P = \max(I) \cdot N + e \cdot \max(W) \cdot \max(D) \cdot N^2$ high enough that the fitness intervals of feasible and infeasible solutions are disjunctive. In the feasible domain we would minimize the objective value as given in equation (1) while in the infeasible domain the amount of "overbooked capacity" is to be minimized.

2.1 Benchmark Instances

For the experiments we used 21 benchmark instances from Cordeau et al. [13] with dimensions from 20 to 50 tasks and 6 to 20 resources. Instances are generated with different levels of utilization ranging from 35% to 95%. A utilization of x% means that the total demand $\sum_{t \in N} Q_t$ is x% of the total capacity $\sum_{r \in M} C_r$.

In addition we generated benchmark instances based on problem instances of the QAPLIB [10]. For every QAPLIB instance of dimension $N \geq 20$ we generated *three groups* with (rounded) $N/6$, $N/3$, and $2N/3$ resources. We ran a hierarchical clustering on the distance matrix and computed the reduced matrix D by averaging the distances between the clusters using the original distance matrix. Installation Costs I were generated by sampling independently from a uniform distribution in the half-open interval $[1, N \cdot \overline{W} \cdot \overline{D})$ where $\overline{W}$ and $\overline{D}$ describe the respective mean. The demands Q_t were sampled independently from a uniform distribution in the half-open interval $[1, 100)$. Then we computed 4 instances per group with utilizations $U = \{35\%, 50\%, 75\%, 95\%\}$. This was achieved by distributing the total capacity $C = \sum_t Q_t / (100 - U_i)$ randomly over the resources adhering to the constraint that $\exists_{t \in N} \forall_{r \in M} C_r \geq Q_t \wedge \exists_{r \in M} \forall_{t \in N} Q_t \leq C_r \wedge \forall_{r \in M} C_r < \sum_{t \in N} Q_t$. Nevertheless, this constraint does not guarantee that every generated instance is also feasible to solve. The generation of I, Q, and C is similar to the generator described by Cordeau et al. [13], but these instances feature a wider range of different matrices for W and D. We used 168 instances from this newly generated set. All generated instances are available online[1].

2.2 Feature Extraction

In this work we aim to evaluate how the performance of the applied algorithm instances correlates to characteristics of the problem instances. For this purpose we calculate problem specific features that can be calculated from the problem data, i.e. Q, C, I, W, D as well as features from exploratory walks. In total the following characteristics are obtained:

- Problem specific: dimension ($|N|$), MN-ratio ($|M|/|N|$), dominance (CV .. coefficient of variation) for W, D, C, Q, sparsity for W, D (W_0, D_0), utilization ($\sum_t Q_t / \sum_r C_r$), and basic feasibility
- Random walk: autocorrelation(1), correlation length, information content (ic), partial information content (pic), density basin information (dbi), information stability (is), diversity,

[1] dev.heuristiclab.com/AdditionalMaterial

regularity, total entropy ($H(X)$), peak information content (ic*), peak density basin information (dbi*).
- Directed walk: sharpness, bumpiness, flatness

Sparsity is calculated as the ratio between the number of zero entries to the size of the matrix while basic feasibility describes whether all tasks can be assigned to all resources given Q and C: $\sum_{t \in N} |\{r \in M | Q_t \leq C_r\}|/(N \cdot M)$.

The random walk-based landscape measures are adequately described in the existing literature [2, 29, 32]. We used a walk length of 5,000 and averaged the results of 30 independent walks. Directed walks are a newer method for characterizing fitness landscapes [7]. Directed walks are based on the path-relinking heuristic, i.e. they repeatedly connect two points in the search space via a path in which the best among all alternative choices is made in each step. All paths are then analyzed and the three aforementioned characteristics are extracted. Sharpness is the average absolute fitness change in the paths, bumpiness is the relative number of "inflection points", i.e. where the fitness delta would change sign, but does not become 0, while flatness describes the "undulation points" where the fitness remains the same. Sharpness is normalized to $[0; 1]$ by the minimum and maximum observed fitness among all paths for each problem instance.

2.3 Fitness Landscape Analysis

Table 1 shows the correlations among the obtained characteristics. We observe a moderate ($\rho \approx -0.57$) correlation between W_0 and $|N|$. This indicates that larger instances did not tend to include sparse W matrices. We observe a correlation ($\rho \approx -0.57$) between M/N and basic feasibility we conclude that larger M create search spaces with more infeasible regions. There was also moderate ($\rho \approx 0.47$) correlation between CV(C) and flatness indicating that instances where the capacities are very unevenly distributed are also more flat. However, as there is correlation ($\rho \approx -0.54$) between CV(C) and basic feasibility and also between flatness and basic feasibility ($\rho \approx -0.56$) this may be explained by larger infeasible regions. It may be assumed that the lexicographic fitness function with objective and infeasibility minimization respectively, is responsible for these observations. Utilization has been observed as having stronger correlations with FLA characteristics, especially on (ic, pic, dbi, $H(X)$) ($\rho \approx (-0.58, 0.68, -0.73, -0.60)$), but there was also correlation between utilization and bumpiness ($\rho \approx 0.44$). Interestingly, correlation between basic feasibility and utilization was low ($\rho \approx -0.31$). Basic feasibility thus cannot adequately describe an increase in infeasible solutions due to higher utilization.

3 ALGORITHM INSTANCES

The algorithm instances that we applied to the set of problem instances form a heterogeneous set including single-solution and population-based methods, open source implementations and commercial solvers, and approximate and exact algorithms. Still, the considered algorithms only cover a portion of possible algorithms, for instance there was no representative of a memetic algorithm. In total for this study we considered:

(1) Iterated Local Search (2 instances)
(2) Evolution Strategy (1 instance)
(3) Iterated Genetic Algorithm (1 instance)
(4) Greedy Randomized Adaptive Search Procedure (1 instance)
(5) Late-acceptance Hill Climber (1 instance)
(6) Age-Layered Population Structure (1 instance)
(7) Linearized Integer Programming (2 instances)
(8) Hybrid Mathematical Programming Solver (2 instances)
(9) Random Search (1 instance)

In the following the algorithm instances shall be detailed, though it would be outside the scope of this paper to describe them in all their details. We include random search in this study, not because we expect it to work well, but to act as a baseline and give confidence in the obtained results.

Iterated Local Search

Iterated local search (ILS) is a very general framework that is based on an efficient neighborhood-based local search [23]. In this metaheuristic four heuristics are used: 1) initial solution generation, 2) local search, 3) solution perturbation, and 4) solution acceptance. In the case of the GQAP we use the *1-shift* neighborhood, i.e. relocating one task from its currently assigned resource to another. The size of the neighborhood is $N \cdot (M - 1)$ if enumerated exhaustively. A locally optimal solution for this neighborhood is said to be *1-opt*. Evaluating the fitness delta of the move can be performed in $O(N)$.

In this study, both instances accept the new local optimum only when it is strictly better than the old. But one instance performs a reassignment of all tasks during perturbation, i.e. "multi-start local search" (MLS) and uses a uniformly random initialized s_0, while the second instance uses a probabilistic reassignment of on average 10% of the tasks and generates s_0 using the greedy randomized construction described by Mateus et al. [24]. The value of 10% was determined using irace 2.4 on several benchmark instances in 5,000 experiments. The instances did not make use of a history. Tags: single-solution, open source, approximate

Late-acceptance Hill Climber

Late-acceptance hill climber (LAHC) [11] is a recent metaheuristic. Its outstanding feature is a probabilistic acceptance criterion that is similar to simulated annealing, but does not rely on a temperature parameter. It is closely related to the ILS framework and also employs randomly selected 1-shift moves as perturbation. However, there is no local search heuristic. Two parameterless variants have been recently introduced [6] of which pLAHC-s (with seeding) constitutes the implemented instance. This algorithm does not feature parameters that need to be tuned. Tags: single-solution, open source, approximate

Evolution Strategy

Evolution strategy (ES) is well known for finding minima or maxima in real-valued search spaces. The outstanding characteristic of ES-based algorithms is a dynamic adaptation of the perturbation, also called mutation, strength. ES allows to use a population of solutions and typically generates multiple descendants in each generation.

In this study we chose a self-adaptive (10,1000)-ES with recombination and use the same perturbation method that is used in ILS as mutation. A single strategy parameter σ_i per solution i determines the probability of how many tasks should be reassigned by converting it into the interval $(0, 1)$ by a sigmoidal function. Thus the ES

Table 1: Correlations among FLA characteristics using Spearman's ρ. The upper diagonal displays significant correlations: "*" indicate a p-value $< 1e^{-6}$, "**" indicates $p < 1e^{-3}$, "*" indicates $p < 0.05$. All p-values are Bonferroni-adjusted.**

	feas.	$CV(C)$	$CV(Q)$	$\|N\|$	$CV(D)$	D_0	$CV(W)$	W_0	M/N	util.	bump.	flat.	sharp.	ac1	corrlen	dbi	div.	ic	is	pic	dbi*	ic*	reg.	$H(X)$
feas.	1	***				***			***	*	***	***	***	***	**		*	*	***				***	*
$CV(C)$	-0.54	1									***	***								*				
$CV(Q)$	-0.05	0.07	1				**																	
$\|N\|$	-0.15	0.18	0.11	1		***		***				***	***				***	***					**	***
$CV(D)$	-0.17	-0.04	-0.19	0.25	1	**			**															
D_0	0.44	-0.01	0.1	-0.52	-0.36	1		***	***		**	***	**					*						*
$CV(W)$	0	0.1	-0.36	-0.01	0.03	-0.21	1																	
W_0	-0.08	0.19	0	-0.57	0.07	0.49	-0.04	1						*										
M/N	-0.57	-0.03	-0.08	-0.01	0.41	-0.64	0	-0.06	1		***	***					**		**				***	
util.	-0.31	-0.06	-0.08	0.03	0.1	-0.02	-0.03	-0.05	0.01	1	***					***		***	***	***	***	***	*	***
bump.	0.47	-0.43	-0.15	-0.16	-0.12	0.37	-0.08	0.04	-0.44	0.44	1	***	***	*		**				**	**	**		
flat.	-0.56	0.47	0.16	0.43	0.23	-0.48	-0.01	-0.15	0.43	-0.06	-0.76	1	***	*									*	
sharp.	0.42	-0.25	-0.01	-0.47	-0.14	0.36	-0.08	0.23	-0.18	-0.17	0.48	-0.58	1	***	***			***	**	**				***
ac1	-0.44	0.19	0.15	0.23	-0.24	-0.24	-0.2	-0.3	0.2	0.19	-0.3	0.3	-0.49	1	***			**	***					**
corrlen	-0.41	0.11	0.15	0.17	-0.24	-0.23	-0.21	-0.26	0.26	0.18	-0.27	0.26	-0.42	0.94	1			*	***					*
dbi	0.13	0.26	0.12	0.11	-0.1	0.01	0.04	0.03	-0.06	-0.73	-0.4	0.07	0.11	-0.07	-0.07	1	**	***	***	***	***	***	**	***
div.	0.29	0	-0.08	0.43	-0.11	-0.06	0.18	-0.26	-0.36	-0.27	-0.05	-0.14	-0.2	0.11	0.04	0.35	1	*	**		***	***	***	
ic	0.31	0.03	0.01	-0.45	-0.09	0.28	-0.03	0.27	-0.02	-0.58	-0.1	-0.11	0.56	-0.37	-0.32	0.54	-0.28	1	***	***	***	***		***
is	0.61	-0.16	-0.05	-0.01	-0.11	0.18	0.16	-0.07	-0.34	-0.63	0	-0.21	0.38	-0.52	-0.48	0.53	0.37	0.47	1	***	***	***	***	***
pic	-0.15	-0.28	-0.06	0.2	0.07	-0.14	0.01	-0.19	-0.02	0.68	0.35	-0.12	-0.35	0.23	0.21	-0.75	0.1	-0.91	-0.47	1	***	***		***
dbi*	0.16	0.25	0.06	0.13	-0.13	0.01	0.09	0.01	-0.12	-0.68	-0.37	0.06	0.13	-0.02	-0.03	0.84	0.5	0.47	0.52	-0.65	1	***	***	***
ic*	-0.16	-0.25	-0.07	-0.16	0.12	0.03	-0.07	0.03	0.11	0.66	0.34	-0.07	-0.16	0.01	0.03	-0.81	-0.49	-0.49	-0.51	0.66	-0.98	1	***	***
reg.	0.45	-0.11	-0.11	0.33	-0.14	0.04	0.13	-0.25	-0.46	-0.31	0.1	-0.32	-0.01	-0.01	-0.07	0.39	0.95	-0.15	0.48	0.03	0.51	-0.51	1	
$H(X)$	0.32	0.04	0.01	-0.43	-0.09	0.28	-0.02	0.26	-0.02	-0.6	-0.11	-0.11	0.54	-0.36	-0.31	0.57	-0.24	1	0.48	-0.92	0.51	-0.52	-0.12	1

instance may dynamically adapt the parameter that we fixed in the 2^{nd} ILS instance to 10%. For the strategy parameter mutation we use an additive mutation scheme as this parameter may become negative. The parameters $\mu = 10, \lambda = 1000$ and the use of recombination and comma selection was determined by irace 2.4 on several benchmark instances in 5,000 experiments. Tags: population, open source, approximate

Iterated Genetic Algorithm

Genetic algorithms (GA) are a family of evolutionary methods that are population-based and which use *crossover* and to a lesser degree *mutation* as variation operators. In this study we chose a variant called *offspring selection genetic algorithm* (OSGA) [1]. It is a hybrid between evolution strategies and genetic algorithms. In each generation OSGA samples more offspring than the population size. However, it does not use a fixed ratio λ/μ as in ES, but dynamically adjust this *success ratio*. Each offspring is compared to its parents and is then either accepted or rejected. OSGA continues to produce offspring until enough have been accepted to fulfil the success criterion or an upper bound on sampled offspring has been reached in which case the algorithm terminates. We added a restart procedure in which we reinitialize the population randomly, but not allowing to keep the same assignment that it had previously.

We use a new crossover, called "discrete location crossover (DLX)" shown in Algorithm 1. As mutation it reuses the perturbation method of ILS reassigning on average 25% of the tasks. It is applied with a probability of 5% to a new offspring. A population size of 500 was chosen which was well suited to solve certain problem instances to optimality. Similar to ES, parents were selected randomly with equal probability. Tags: population, open source, approximate

Algorithm 1 Discrete Location Crossover (DLX)

procedure DLX($C\downarrow, Q\downarrow, P\downarrow, child\uparrow$) ▷ $P \mathrel{\widehat{=}} parents$
  $(slack_r, child_t) \leftarrow (C_r\ \forall_{r\in M}, -1\ \forall_{t\in N})$
  for r in R **do** ▷ Randomize order
    $p \leftarrow$ `random from` $\{p' \in P | \exists_{t\in N}, p'_t = r\}$
    if $p \neq \emptyset$ **then** ▷ Perform crossover for resource r
      $N' \leftarrow \{t \in N | p_t = r \wedge child_t = -1\}$
      $(child_t, slack_r) \leftarrow (r\ \forall_{t\in N'}, slack_r - \sum_{t\in N'} Q_t)$
    end if
  end for
  $N'' \leftarrow \{t \in N | child_t = -1\}$ ▷ Unassigned tasks
  for t in N'' **do** ▷ Randomize order
    $p \leftarrow$ `random from` $\{p' \in P | slack_{p'_t} \geq Q_t\}$ ▷ prio 1
    if $p \neq \emptyset$ **then**
      $(child_t, slack_{p_t}) \leftarrow (p_t, slack_{p_t} - Q_t)$
    else
      $r \leftarrow$ `random from` $\{r' \in M | slack_{r'} \geq Q_t\}$ ▷ prio 2
      if $r \neq \emptyset$ **then**
        $(child_t, slack_r) \leftarrow (r, slack_r - Q_t)$
      else
        $r \leftarrow$ `random from` M ▷ prio 3
        $(child_t, slack_r) \leftarrow (r, slack_r - Q_t)$
      end if
    end if
  end for
end procedure

Age-Layered Population Structure

Age-Layered Population Structure (ALPS) [16] is an evolutionary algorithm that parts the population of solutions into a number of layers. Each layer performs a similar variation loop to that of a genetic algorithm, i.e. selection, crossover, and mutation. The outstanding characteristic of ALPS is the mitigation of premature convergence by frequently introducing random solutions into lower layers of the population. Solutions move from lower to higher layers as they become better.

ALPS uses the same crossover and mutation as in the iterated OSGA instance. It was configured with a maximum of 10 layers and 100 solutions per layer, age gap was set to 20, and a polynomial

ageing scheme was used. The algorithm was set to use 1-elitism, and a generalized linear rank selection scheme was used to select parents [31]. The selection pressure parameter of the generalized linear rank selector was set to 4. Tags: population, open source, approximate

Greedy Randomized Adaptive Search Procedure

The greedy randomized adaptive search procedure (GRASP) in this study is described by Mateus et al. [24] and makes use of a path relinking heuristic. The described heuristics are highly specific to the GQAP, for instance the path relinking performs a repair operation after each step while the local search is of a customized approximate nature not necessarily finishing in a local optimum. Several parameterizations are analyzed and a recommendation for default parameters is made [24]. The employed algorithm instance uses the default recommendation that performed well in the tests ("f-r-g-g") and the recommended default parameters have been used (η = 50%, elite set size = 10, maximum candidate list size = 10, δ = 4, minimum elite set size = 2, one-move probability = 50%, maximum iterations for local search = 100). Tags: population, open source, approximate

Linearized Integer Programming

Two linearized integer programming (IP) models have been implemented that are described in the literature for the quadratic assignment problem (QAP) and which have been adapted to the GQAP. On the one hand there is the linearization described by Frieze and Yadegar [14] and on the other hand the linearization described by Kaufman and Broeckx [19]. The models are implemented in the optimization programming language (OPL) and solved by CPLEX[2] 12.7.0 for which an academic license had been obtained. The CPLEX solver was run with all parameters set to their default value. Tags: single-solution, commercial, exact

Hybrid Mathematical Programming Solver

LocalSolver[3] 7.5 is, at the time this work was written, the most recent version of a commercial mathematical programming solver [9]. Academic licenses can be obtained free of charge and are valid for one month. Models can be described in a similar way to OPL, but LocalSolver uses heuristics to solve them. Due to the heuristic solver it is allowed to use non-linear functions and operators in the model and thus it is not necessary to perform a linearization. Two models have been implemented, the first model uses binary decision variables $x_{ij} \in \{0, 1\}$ to denote the assignment of task i to resource j while the second model uses integer decision variables $x_i \in M$ that directly encode the resource to which task i is assigned to. Default parameters have been used. Tags: single-solution, commercial, approximate

4 EXPERIMENT SETUP AND EXECUTION

Due to the heterogeneous set of algorithm instances, tests were performed on the same machine and run-length was measured in elapsed wall clock time. All open source instances where implemented in C# using .Net Framework 4.5 and were implemented to run in a single-thread, except CPLEX which could use up to 28 cores. Running CPLEX sequentially gave worse results, but we decided to give CPLEX an advantage as it is the only exact solver in the set. The target machine is an Intel®Xeon®E5-2660 running at 2.0 Ghz to which a Windows 10 (version 1607) virtual machine (VM) was deployed. It is the only VM instance that was deployed to this machine while the tests have been conducted. The VM has access to 125Gb RAM, and 28 cores.

We performed 30 to 45 repetitions per algorithm / problem instance combination for the open source implementations and a single repetition for the commercial solvers. All algorithm instances were given a maximum execution time of 1 minute which was enough that we found optimal solutions in several problem instances. The experiment of all open source instances was distributed among 28 cores. The experiments involving the commercial solvers were run sequentially. Only one instance of the commercial solver was running at the same time. In total more than 60,000 runs have been recorded.

We obtained the performance data by tracking a variable that stores the best fitness per run. In the open source implementations this variable was updated after calling the evaluation function, but not during an operator such as the local search. In CPLEX this was done in a MIPInfoCallback, while in LocalSolver the IterationTicked callback was used. The wall clock time for all runs has been measured using the high-precision System.Diagnostics.*Stopwatch* class. The experiment data is available online[4].

4.1 Performance Measurement

The performance of heuristic methods is determined by two dimensions: 1) the solution quality and 2) the execution time. Ideally, we would seek an algorithm that achieves better quality in less time than any other. However, on the one hand the no-free-lunch (NFL) theorem suggests this is not attainable over all problems [35]. And on the other hand it is generally observed that a longer runtime leads to better quality. In experiments often the run-length is bounded by some value X and algorithms are compared based on the distribution of solution qualities obtained after X units. Hoos and Stützle however proposed to treat the runtime or run-length as a random variable that is dependent on the target fitness that should be achieved [15]. They then describe the expected runtime (ERT) as a measure to describe and compare the performance of algorithm instances.

In our experiment the algorithm instances will be clustered given their performance into six classes. We use an optimal 1-dimensional clustering algorithm [33] to cluster these algorithm instances by their $\log_{10}(\mathsf{ERT})$ performance values [3, 15]. 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. Class 6 is reserved for algorithm instances that did not achieve the target quality in any of the observed runs and thus have an ERT of ∞.

5 POST-HOC EXPERIMENT ANALYSIS

Some runs of pLAHC-s ran into limitations due to a very large memory size and terminated before the time limit and without having

[2] https://www.ibm.com/products/ilog-cplex-optimization-studio
[3] http://www.localsolver.com
[4] dev.heuristiclab.com/AdditionalMaterial

Table 2: Percentage of achieved ranks per algorithm

Alg.Inst	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}
ILS	19%	31%	26%	14%	1%	10%
MLS	4%	5%	5%	7%	13%	66%
pLAHC-s	3%	5%	13%	23%	23%	34%
ES	1%	4%	12%	15%	14%	55%
OSGA	28%	26%	18%	10%	5%	14%
ALPS	1%	4%	24%	22%	17%	32%
GRASP+PR	25%	32%	13%	8%	4%	17%
CPLEX-FY	6%	14%	8%	5%	2%	65%
CPLEX-KB	7%	7%	8%	7%	4%	67%
LocalSolver 01	40%	11%	12%	6%	2%	31%
LocalSolver N	10%	14%	10%	5%	4%	58%
RS	0%	0%	0%	0%	0%	100%

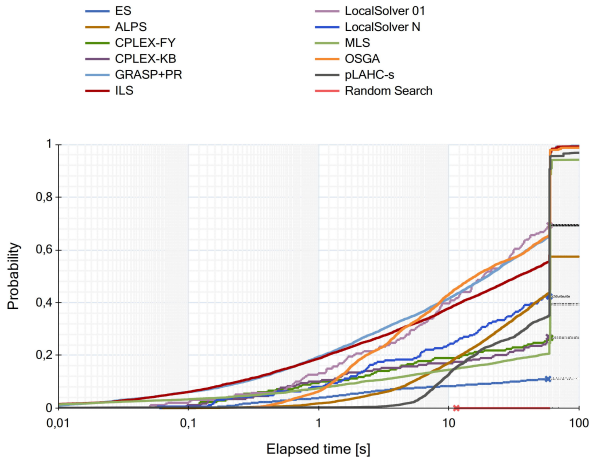
achieved the target quality. We thus simulated a random restart for this runs by a process similar to bootstrapping in statistics. Instead of performing an actual restart, a run is sampled from the pool of observed runs and replayed on top of the shorter run.

In total we evaluated 12 algorithm instances on 189 problem instances. In the post-hoc analysis we use a target quality with a gap of 2% to the optimum respectively the best-found quality. We observed that on average, there were 1.4 algorithm instances ranked first, 1.5 algorithm instances in the 2^{nd} best class, 1.5 instances ranked 3^{rd}, while 1.2 and 0.9 algorithm instances were ranked 4^{th} and 5^{th} respectively. The majority of algorithm instances: 5.5 on average were ranked in the least suitable class, which means they did not find the overall best-found solution in any of the runs. Which of the algorithm instances ranked best depends on the problem instance. In Table 2 we state the number of ranks observed for each algorithm instance. It should be stressed again that the ranks indicate that the respective algorithm instance is expected to achieve the best-found solution *first*. It does not mean that the best found solution was only achieved by the instances ranked 1^{st}. On the contrary all instances up to rank 5 achieved the best-found solution at least once in a run, but with an increasingly higher expected runtime (ERT).

At a 2% gap and out of all problem instances, in 126 (66.7%) cases there was only a single algorithm instance ranked 1^{st} and in 11 (5.8%) cases all, but one algorithm instance ranked 6^{th}.

Table 3 shows correlations among the classes calculated for the algorithm instances and between classes and FLA values. We note that the OSGA instance does not indicate many significant correlations with other instances indicating that it performs different to other algorithms and would be a suitable member of a portfolio. The only positive correlation to ALPS ($\rho \approx 0.51$) may be explained that both use the same crossover. While the two CPLEX models showed similar performance ($\rho \approx 0.71$) the two LocalSolver models do not correlate. The binary LocalSolver model does not correlate with any other algorithm. The strongest average positive correlations among FLA features and algorithm ranks was observed with problem dimension (($\overline{\rho} \approx 0.26$)), especially concerning the exact solvers. We also observed mild average correlation ($\overline{\rho} \approx 0.21$) with the flatness characteristic, suggesting landscapes with added flatness are more difficult to solve. Though, we must add that a negative correlation ($\rho \approx -0.56$) between basic feasibility and flatness has been recorded, thus it may also be an indication of problem instances with a higher amount of infeasible solutions. We may also find a

Figure 1: ECDF graphs of the algorithms' performances over all problem instances for the 2% target.



possible explanation for the stark contrast in performance between the crossover-based algorithm instances and the others. Utilization (util.) separates the performance of algorithm instances, i.e. lower utilization lead to better results for most algorithms, but the correlation with OSGA ($\rho \approx -0.38$) and ALPS ($\rho \approx -0.3$) is negative. We hypothesize that crossover is beneficial to find good solutions for problem instances with high utilization, whereas it prevents good performance in cases with low utilization. Additionally, we observe that pLAHC-s performance negatively correlates with M/N ratio ($\rho \approx -0.18$) whereas all other correlations are (slightly) positive.

In the empirical cumulative distribution function (ECDF) curves in Figure 1 [8] we observe that ILS and GRASP+PR work well in the beginning, but while ILS does not perform that much better with added runtime, OSGA and LocalSolver 01 become stronger indicating that these two need slightly more runtime. With OSGA, GRASP, and the LocalSolver 01 model we can cover about 74% of the problem instances in that one of these is ranked 1^{st}. If ILS is added this can be increased to 83%.

6 ALGORITHM SELECTION

We evaluate a k-nearest neighbor-based approach (k-NN) for algorithm selection. The k-NN will use Euclidean distance between two feature vectors. All features are normalized to a standard normal distribution $N(0, 1)$. k-NN is an instance-based learning algorithm which does not require training. The hyperparameter $k \in \mathcal{N}$ can be tuned by enumeration. But its performance depends on the features of course. We will thus use 4 different feature sets that we compare:

(1) Problem specific features only
(2) Random walk features only
(3) Directed walk features only
(4) Problem specific and FLA features

The algorithm instances will be ranked based on the observed rankings of the k closest problem instances. For $k > 1$ we must somehow average different rankings, however as has been hypothesized previously [7] such an averaging does not favor the best, but

Table 3: Correlation between fitness landscape characteristics and algorithm instances' ranks using Spearman's ρ. The upper diagonal displays significant correlations equally to Table 1

	ILS	MLS	pLAHC-s	ES	OSGA	ALPS	GRASP+PR	CPLEX-FY	CPLEX-KB	LocalSolver 01	LocalSolver N	RS	Average
ILS	1.00		***				**						0.18
MLS	0.15	1.00	**	***	*		***	***	***		***		0.34
pLAHC-s	0.50	0.37	1.00	***	**		***						0.23
ES	0.19	0.49	0.40	1.00		*	***	**	***		***		0.37
OSGA	0.01	-0.26	-0.34	0.01	1.00	***							0.06
ALPS	-0.04	0.12	-0.15	0.25	0.51	1.00		**	*		*		0.24
GRASP+PR	0.33	0.49	0.44	0.62	0.01	0.19	1.00	**	**		***		0.36
CPLEX-FY	-0.04	0.62	0.12	0.35	-0.07	0.37	0.32	1.00	***		**		0.31
CPLEX-KB	-0.01	0.63	0.19	0.48	-0.06	0.30	0.38	0.71	1.00		***		0.35
LocalSolver 01	-0.04	-0.01	0.05	0.20	-0.12	0.07	0.17	-0.09	0.04	1.00			0.11
LocalSolver N	0.08	0.42	0.12	0.44	0.04	0.26	0.40	0.37	0.48	0.07	1.00		0.31
RS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.08
feas.	-0.14	-0.38	-0.14	-0.36	-0.19	-0.34	-0.42	-0.2	-0.37	-0.09	-0.33	0.00	-0.25
$CV(C)$	-0.02	0.06	0.06	0.15	0.2	0.31	0.1	0	0.12	0.11	0.03	0.00	0.09
$CV(Q)$	-0.03	-0.01	-0.08	0.09	0.1	0.17	0.08	0.03	0.02	0.06	0.06	0.00	0.04
$\|N\|$	-0.06	0.48	0.03	0.32	0.03	0.48	0.34	0.74	0.57	-0.11	0.32	0.00	0.26
$CV(D)$	0.1	0.22	0.04	0.17	0.13	0.35	0.17	0.4	0.28	-0.11	0.27	0.00	0.17
D_0	0.01	-0.45	0.07	-0.25	-0.16	-0.43	-0.26	-0.46	-0.57	0.09	-0.4	0.00	-0.23
$CV(W)$	0.07	-0.02	0.1	-0.14	-0.1	-0.09	-0.1	-0.02	0.11	-0.1	-0.01	0.00	-0.03
W_0	0.19	-0.31	0.1	-0.05	0.11	-0.02	-0.04	-0.49	-0.36	0.26	-0.1	0.00	-0.06
M/N	0.02	0.31	-0.18	0.26	0.3	0.31	0.2	0.23	0.36	0.03	0.4	0.00	0.19
util.	0.39	0.47	0.52	0.26	-0.38	-0.3	0.47	0.17	0.22	-0.05	0.09	0.00	0.16
bump.	0.21	0	0.31	-0.12	-0.46	-0.5	0.06	-0.16	-0.19	-0.04	-0.22	0.00	-0.09
flat.	-0.14	0.38	-0.14	0.34	0.26	0.48	0.18	0.4	0.45	-0.03	0.36	0.00	0.21
sharp.	-0.03	-0.41	-0.1	-0.31	-0.13	-0.26	-0.29	-0.39	-0.34	0.09	-0.29	0.00	-0.21
ac1	0.01	0.34	0.02	0.29	0.05	0.09	0.34	0.18	0.22	0	0.19	0.00	0.14
corrlen	0.01	0.32	0	0.28	0.06	0.08	0.32	0.18	0.22	0.02	0.19	0.00	0.14
dbi	-0.33	-0.41	-0.36	-0.08	0.35	0.32	-0.27	-0.11	-0.12	0.05	-0.09	0.00	-0.09
div.	-0.05	-0.15	-0.06	-0.1	0.08	0.23	-0.03	0.17	-0.02	-0.04	-0.07	0.00	0.00
ic	-0.29	-0.46	-0.32	-0.25	0.14	-0.12	-0.51	-0.41	-0.35	0.04	-0.26	0.00	-0.23
is	-0.32	-0.52	-0.34	-0.36	0.08	0.04	-0.51	-0.17	-0.29	0.02	-0.31	0.00	-0.22
pic	0.36	0.38	0.35	0.13	-0.23	-0.11	0.43	0.25	0.2	-0.06	0.16	0.00	0.16
dbi*	-0.33	-0.35	-0.35	-0.11	0.27	0.28	-0.28	-0.06	-0.09	0.07	-0.1	0.00	-0.09
ic*	0.32	0.32	0.36	0.1	-0.26	-0.28	0.28	0.05	0.08	-0.06	0.1	0.00	0.08
reg.	-0.08	-0.27	-0.11	-0.19	0.05	0.13	-0.14	0.06	-0.14	-0.05	-0.17	0.00	-0.08
$H(X)$	-0.3	-0.46	-0.33	-0.25	0.15	-0.1	-0.51	-0.4	-0.35	0.04	-0.26	0.00	-0.23
Average(Abs)	0.17	0.33	0.23	0.26	0.19	0.25	0.30	0.27	0.29	0.10	0.24	0.03	

rather generally well performing methods. In this work we thus set $k = 1$ and as a target set a 2% gap to the best fitness that has been recorded, respectively the optimum if it is known. We do not apply Euclidean distance to the raw features, but normalize them to a standard normal distribution $N(0, 1)$. We evaluate this approach using leaf-one-out crossvalidation (LOOCV). This creates a total of 189 folds with 188 problem instances known and one being the new one for which algorithm instances should be selected.

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) Spearman's ρ

The NDCG [18] measure originates in the domain of information retrieval and describes the gain of a certain document based on its position in the produced ranking. The discount penalizes documents with worse rank so that a good document provides less gain if it is ranked later. This DCG measure is normalized with the ideal DCG (iDCG) which is the discounted gain of a perfect ranking. The NDCG is thus in the range $[0; 1]$ and a value closer to 1 indicates better performance. One can apply this measure only to the first n ranked documents which is then called the NDCG_n measure.

Table 4: Correlation analysis of an ideal selection vs k-NN

Features	NDCG_1	ρ
statistical	0.69	0.49
random walk (fla)	0.56	0.39
directed walk (fla)	0.65	0.43
statistical + fla	0.68	0.48
statistical + directed walk	0.69	0.49

6.2 Results

When using all features we observed that the combined algorithm instance achieved ranks $1 - 6$ in 41%, 22%, 12%, 5%, 3%, and 17% of the cases respectively. Thus, we would not pick an unsuitable configuration so often if LocalSolver 01 was applied alone. In Table 4 we give the NDCG and ρ values. The best results are achieved with statistical features only, although there is only very little difference. It is noteworthy that directed walks, although this are just three features performs much better than characteristics obtained from a random walk which does not really improve selection. We used all 12 algorithm instances for these tests, but it would be meaningful to limit the analysis to only the top three or four as outlined before.

7 CONCLUSION AND OUTLOOK

We have evaluated the application of algorithm selection to the generalized quadratic assignment problem. We performed fitness landscape analysis (FLA) and studied possible correlations to algorithm performances. In looking at correlations between algorithm performances we observed that many algorithm instances actually perform somewhat similar, but that algorithm instances using crossover may work well on problem instances that other instances do not work well for. We hypothesized that crossover may be a necessary heuristic to solve problem instances with a high utilization based on the correlations that we observed. We also evaluated two commercial solvers and found them to be very well suited to solve instances of the GQAP. While CPLEX did have its limitations in such a short time, especially with respect to larger instances, LocalSolver could provide good solutions to a number of instances and in short time.

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.

REFERENCES

[1] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. 2009. *Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications.* CRC Press.

[2] Eric Angel and Vassilis Zissimopoulos. 1998. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science* 191, 1-2 (1998), 229–243.

[3] 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. https://doi.org/10.1109/CEC.2005.1554903

[4] Thomas Bartz-Beielstein. 2006. *Experimental Research in Evolutionary Computation: The New Experimentalism.* Springer.

[5] Thomas Bartz-Beielstein, Christian WG Lasarczyk, and Mike Preuß. 2005. Sequential parameter optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 1. IEEE, 773–780.

[6] Mosab Bazargani and Fernando G. Lobo. 2017. Parameter-less Late Acceptance Hill-climbing. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*. ACM, New York, NY, USA, 219–226. https://doi.org/10.1145/3071178.3071225

[7] Andreas Beham, Michael Affenzeller, and Stefan Wagner. 2017. Instance-based Algorithm Selection on Quadratic Assignment Problem Landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. ACM, New York, NY, USA, 1471–1478. https://doi.org/10.1145/3067695.3082513

[8] Andreas Beham, Stefan Wagner, and Michael Affenzeller. 2016. Optimization Knowledge Center: A Decision Support System for Heuristic Optimization. In *Companion Publication of the 2016 Genetic and Evolutionary Computation Conference, GECCO'16 Companion*. ACM, 1331–1338.

[9] Thierry Benoist, Bertrand Estellon, Frédéric Gardi, Romain Megel, and Karim Nouioua. 2011. Localsolver 1.x: a black-box local-search solver for 0-1 programming. *4OR: A Quarterly Journal of Operations Research* 9, 3 (2011), 299–316.

[10] 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/

[11] Edmund K. Burke and Yuri Bykov. 2017. The late acceptance Hill-Climbing heuristic. *European Journal of Operational Research* 258, 1 (2017), 70 – 78. https://doi.org/10.1016/j.ejor.2016.07.012

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

[13] Jean-François Cordeau, Manlio Gaudioso, Gilbert Laporte, and Luigi Moccia. 2006. A memetic heuristic for the generalized quadratic assignment problem. *INFORMS Journal on Computing* 18, 4 (2006), 433–443.

[14] Alan M Frieze and Joseph Yadegar. 1983. On the Quadratic Assignment Problem. *Discrete applied mathematics* 5, 1 (1983), 89–98.

[15] H. H. Hoos and T. Stützle. 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.

[16] Gregory S. Hornby. 2006. ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*. ACM, New York, NY, USA, 815–822. https://doi.org/10.1145/1143997.1144142

[17] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. 2009. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36, 1 (2009), 267–306.

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

[19] L Kaufman and Fernand Broeckx. 1978. An Algorithm for the Quadratic Assignment Problem using Bender's Decomposition. *European Journal of Operational Research* 2, 3 (1978), 207–211.

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

[21] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. 2003. A portfolio approach to algorithm selection. In *IJCAI*, Vol. 3. 1542–1543.

[22] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. 2011. *The irace package, Iterated Race for Automatic Algorithm Configuration.* Technical Report TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium. http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf

[23] Helena R Lourenço, Olivier C Martin, and Thomas Stutzle. 2003. Iterated local search. *International series in operations research and management science* (2003), 321–354.

[24] Geraldo R Mateus, Mauricio GC Resende, and Ricardo MA Silva. 2011. GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics* 17, 5 (2011), 527–565.

[25] 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. https://doi.org/10.1145/2001576.2001690

[26] 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.

[27] 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.

[28] Artur Alves Pessoa, Peter M. Hahn, Monique Guignard, and Yi-Rong Zhu. 2010. Algorithms for the generalized quadratic assignment problem combining Lagrangean decomposition and the Reformulation-Linearization Technique. *European Journal of Operational Research* 206, 1 (2010), 54–63. https://doi.org/10.1016/j.ejor.2010.02.006

[29] Erik Pitzer and Michael Affenzeller. 2012. A comprehensive survey on fitness landscape analysis. In *Recent Advances in Intelligent Engineering Systems*. Springer, 161–191.

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

[31] David M Tate and Alice E Smith. 1995. A genetic approach to the quadratic assignment problem. *Computers & Operations Research* 22, 1 (1995), 73–83.

[32] Vesselin K Vassilev, Terence C Fogarty, and Julian F Miller. 2000. Information Characteristics and the Structure of Landscapes. *Evolutionary Computation* 8, 1 (2000), 31–60.

[33] 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.

[34] Darrell Whitley, Andrew M Sutton, and Adele E Howe. 2008. Understanding elementary landscapes. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 585–592.

[35] David H. Wolpert and William G. Macready. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82.

[36] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research* 32 (2008), 565–606.