Frequency Fitness Assignment: Optimization without Bias for Good Solution outperforms Randomized Local Search on the Quadratic Assignment Problem

Jiayang Chen[1][a], Zhize Wu[1][b], Sarah L. Thomson[2][c], and Thomas Weise[1][d]

[1]*Institute of Applied Optimization, School of Artificial Intelligence and Big Data, Hefei University, Jinxiu Dadao 99, Hefei, 230601, Anhui, China*

[2]*School of Computing, Engineering & the Built Environment, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, U.K.*

cjy65820607@163.com, wuzz@hfuu.edu.cn, S.Thomson4@napier.ac.uk, tweise@hfuu.edu.cn

Keywords: Quadratic Assignment Problem, Frequency Fitness Assignment, Randomized Local Search

Abstract: The Quadratic Assignment Problem (QAP) is one of the classical $\mathcal{NP}$-hard tasks from operations research with a history of more than 65 years. It is often approached with heuristic algorithms and over the years, a multitude of such methods has been applied. All of them have in common that they tend to prefer better solutions over worse ones. We approach the QAP with Frequency Fitness Assignment (FFA), an algorithm module that can be plugged into arbitrary iterative heuristics and that removes this bias. One would expect that a heuristic that does not care whether a new solution is better or worse compared to the current one should not perform very well. We plug FFA into a simple randomized local search (RLS) and yield the FRLS, which surprisingly outperforms RLS on the vast majority of the instances of the well-known QAPLIB benchmark set.

1 INTRODUCTION

The Quadratic Assignment Problem (QAP) is a challenging and very important combinatorial optimization problem (Koopmans and Beckmann, 1957; Burkard et al., 1998; Loiola et al., 2007). Here, the goal is to assign a set of n facilities to a set of n locations. Such an assignment can be represented as a permutation s of the first n natural numbers, where $s[i]$ specifies the location where facility i should be placed. For each QAP, a distance matrix A is given, where A_{pq} specifies the distance from location p to location q, as well as a flow matrix B, where B_{ij} is the amount of material flowing from facility i to facility j. The objective function f subject to minimization then rates a permutation s as follows:

$$f(s) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{s[i]s[j]} B_{ij} \tag{1}$$

The QAP has a wide range of applications including, e.g., building layout (Elshafei, 1977; Çubukçuoğlu et al., 2021; Krarup and Pruzan, 1978), keyboard layout (Burkard and Offermann, 1977), circuit design (Eschermann and Wunderlich, 1990), wiring (Steinberg, 1961), and scheduling (Soroush, 2011). While there has been notable success in applying exact methods to the QAP (Drezner et al., 2005), QAPs are $\mathcal{NP}$-hard (Sahni and Gonzalez, 1976; Dréo et al., 2006) and thus are often solved with heuristic algorithms such as simulated annealing (Thonemann and Bölte, 1994; Wilhelm and Ward, 1987), tabu search (Skorin-Kapov, 1990; Taillard, 1991; Misevičius, 2005; Misevičius, 2008), iterated local search (Stützle, 2006), evolutionary methods (Horng et al., 2000; Taillard and Gambardella, 1997), memetic algorithms (Fleurent and Ferland, 1993; Merz and Freisleben, 1999), estimation of distribution algorithms (Zhang et al., 2006), ant colony optimization (Gambardella et al., 1999; Talbi et al., 2001; Taillard and Gambardella, 1997), or even particle swarm optimization (Hafiz and Abdennour, 2016).

All such heuristic approaches that have been applied to the QAP have one design principle in common: Their (iterative) search procedure is biased towards good solutions. Regardless of whether they employ diversity strategies or methods to increase exploration, on average over time, they do prefer (to

[a] https://orcid.org/0009-0004-2320-2980
[b] https://orcid.org/0000-0001-7416-5711
[c] https://orcid.org/0000-0001-6971-7817
[d] https://orcid.org/0000-0002-9687-8509

exploit) better solutions (in terms of their objective value) over worse ones. Indeed, this is maybe the most fundamental concept of metaheuristic optimization.

In (Weise et al., 2014), a mechanism called Frequency Fitness Assignment (FFA) was proposed, which was later shown to render optimization processes invariant under all injective transformations of the objective function value (Weise et al., 2021b) and, as a result, removing the bias towards better solutions (Weise et al., 2023). By replacing the objective value $f(s)$ of a solution s with its encounter frequency $H[f(s)]$ in all selection decisions of a heuristic, FFA offers this new concept of optimization, which breaks with the existing ideas upon which all metaheuristics are built. The only algorithms that have similar properties are random walks, random sampling, and exhaustive enumeration – none of which are ranked as good approaches to the QAP. FFA has been shown to improve the performance of a randomized local search (RLS) on the Max-Sat problem (Weise et al., 2021b; Weise et al., 2023), the Job Shop Scheduling Problem (JSSP) (Weise et al., 2021a; de Bruin et al., 2023), and on Traveling Salesperson Problem (TSP) instances (Liang et al., 2022; Liang et al., 2024).

However, whether it can improve algorithm performance on a wide set of QAP instances has not yet been studied. In this work, we do not aim to outperform any of the related heuristics listed above. It instead is our goal to establish that FFA is indeed a suitable technique for the QAP. Our **first contribution** is to conduct the first large experiment of FFA on the QAP involving all instances from the QAP benchmark set QAPLIB by (Burkard et al., 1997). We publish all of our code, results, as well as the scripts used for generating the tables and figures in an immutable online archive at https://doi.org/10.5281/zenodo.13324662. As a **second contribution**, we show that, if plugged into a simple RLS, FFA yields a significant improvement in the quality of the discovered results. We show that, despite using a computational budget 100 times smaller than in prior works on FFA, this tangible improvement can be observed.

Finally, our **third contribution** is to provide lower bounds m for the numbers M of possible different objective values for all instances of the QAPLIB. While lower bounds lb for the objective function f exist (Peng et al., 2010; de Klerk and Sotirov, 2010; Drezner et al., 2005), we are the first to investigate m on the QAPLIB instances. This lower bound m can give us an impression about other aspects that may be relevant for optimization and may be related to the amount of neutrality present.

The rest of our work is structured as follows. In Section 2, we discuss related works both on FFA and the QAP before defining the algorithms used in our study in Section 3. In Section 4, we present the results of our experiment before concluding the paper in Section 5 with a summary and outlook on future work.

2 RELATED WORK

2.1 Related Works on the QAP

A wide variety of heuristics has been applied to the QAP, which differ in their algorithmic design philosophies, search strategies, operators, and parameters (Dréo et al., 2006). In this work here, we investigate whether the new paradigm FFA is applicable to the QAP. Beating the state of the art is not our goal. Nevertheless, it is important to at least provide a brief overview of some of the diverse historical heuristic solution ideas for the QAP.

(Wilhelm and Ward, 1987) studied the application of simulated annealing to the QAP. They showed that the simulated annealing algorithm produces good results but is sensitive to the setting of parameters and tested the effect of several parameters on the performance of the algorithm and CPU usage time.

(Taillard, 1991) developed a robust tabu search algorithm for the QAP, which today still is considered as competitive. It explores the neighborhood of the current solution by pairwise exchanges. The aspiration criterion allows forbidden moves if they produce a solution better than the best so far one. A subset of the QAPLIB instances with scales from 5 to 100 were used to investigate the algorithm performance.

Soon thereafter, (Fleurent and Ferland, 1993) presented a hybrid genetic algorithm, which combines the population-based evolutionary heuristic with local search. In traditional genetic algorithms, the quality of individuals can only be improved by crossover, mutation, and other operators. However, hybrid genetic algorithms can improve the solution also by local search or even tabu search. In experiments on the sko-class of instances (Skorin-Kapov, 1990) with scales up to 100, the hybrid algorithm outperformed its component algorithms. (Merz and Freisleben, 1999) introduced a memetic algorithm (MA), which, basically, is another hybrid evolutionary algorithm. The experiment was based on another subset of the QAPLIB instances and the MA outperformed several other heuristics on all instances of practical significance (i.e., except for the randomly generated ones).

In the same year, (Gambardella et al., 1999) presented an ant colony system hybridized with a local search. A comprehensive comparison experiment on several QAPLIB instances with scales n between 19 and 90 showed that this algorithm performs especially well on irregular problems (that is, instances whose distance and/or flow matrix contain disparate values) and representative real-world instances.

(Horng et al., 2000) applied an evolutionary strategy (ES) to the QAP. In order to prevent premature convergence to local optima, this method adds the concept of clustering and family competition to the population handling. The resulting higher diversity leads to good performance on instances with $n \in 19 \ldots 90$. In this work we take the alternative approach of FFA, which – different from the clustering-based idea of that work – does not require any population. Also, diversity is often considered from the search space perspective, whereas FFA tries to create diversity in the objective space in the hope that this induces diversity also in the genotypic representation of the solutions.

As maybe the last of these historical research directions to approach the QAP, (Hafiz and Abdennour, 2016) proposed a discretization framework for particle swarm optimization. This continuous optimization technique, too, can produce good results on the QAP.

Some of the above algorithms, like tabu search or the ES, introduce methods to increase the diversity of the solutions under investigation. Thus, they have components that try to prevent the algorithms from converging to local optima. However, all of them prefer better solutions over worse ones. In the following section, we therefore discuss why FFA is a uniquely different approach to diversity and optimization and why investigating its performance on the QAP is necessary.

It should be noted that in (Thomson et al., 2024), we applied fitness landscape analysis to FFA on the taie27 set of 20 QAP instances of the same scale $n = 27$, which are not part of QAPLIB. In that paper, our goal was to explain why and how FFA-based search works. We presented visualizations of metrics for algorithm trajectories which substantiate the good exploration ability of FFA-based algorithms. The question of whether FFA is a suitable technique for more general QAPs, however, was explicitly left unanswered. We answer it now, by using many more and entirely different instances. We also complement the analysis with several new perspectives, such as an analysis of the last improvement step or which kind of instances FRLS can solve to optimality within a reasonable computational budget.

2.2 Related Works against Convergence to Local Optima

The problem of premature convergence to local optima is well-known in many fields of soft computing. It occurs, for example, in k-means clustering (Shalev-Shwartz and Ben-David, 2014; Arthur and Vassilvitskii, 2007) and the training of ANNs (Shalev-Shwartz and Ben-David, 2014; Treadgold and Gedeon, 1998). In optimization, it has been researched for a long time (Weise et al., 2012; Weise et al., 2009).

Tabu Search (TS) (Glover and Taillard, 1993), one of the most prominent methods to prevent premature convergence, improves upon local search by declaring solutions (or solution traits) that have been visited as tabu, which prevents the algorithm from getting stuck. It has found application in the QAP in several different variants (Misevičius, 2008; Merz and Freisleben, 1999; Skorin-Kapov, 1990).

In the field of Evolutionary Algorithms, the old ideas of sharing, niching, and clearing (Mahfoud, 1997; Goldberg and Richardson, 1987; Deb and Goldberg, 1989; Pétrowski, 1996) as well as clustering (Weise et al., 2011) combine density information with the objective values into so-called fitness values to increase the diversity in the populations of candidate solutions. These methods only consider the present populations and do not consider the history of the search, whereas FFA incorporates and aggregates knowledge over the whole course of optimization.

Methods that try to balance between solution quality and (population) diversity are today grouped under the term Quality-Diversity (QD) algorithms (Cully and Demiris, 2018; Gravina et al., 2019). QD algorithms are mainly applied to games, maze solving, and shape or robotics behavior evolution, but rarely in the context of discrete or hard optimization tasks from operations research.[1]

Novelty Search (NS) (Lehman and Stanley, 2008; Lehman and Stanley, 2011a) is an early QD algorithm. NS is driven by a dynamic novelty metric ρ measuring the mean behavior difference to the k-nearest neighbors in the set of past solution "behaviors." NS with Local Competition (NSLC) (Lehman and Stanley, 2011b) combines the search for diverse solutions with a local competition objective rewarding solutions that can outperform those most similar to them.

In the QD method Surprise Search (SS) (Gravina et al., 2016), a solution is rated by the difference between its observed behavior from the expected

[1] At least the comprehensive paper QD paper list by (Mouret and Cully, 2024) does not list a single work referring to the QAP or the TSP in its abstract.

Algorithm 1: RLS($f : \mathbb{S} \mapsto \mathbb{N}$)

sample s_c from $\mathbb{S}$ u.a.r.; $z_c \leftarrow f(s_c)$;
for $10^8 - 1$ times **do** ▷ *our termination criterion*
  $s_n \leftarrow$ swap 2 values in s_c u.a.r.; $z_n \leftarrow f(s_n)$;
  if $z_n \leq z_c$ **then** $s_c \leftarrow s_n$; $z_c \leftarrow z_n$;
return s_c, z_c

Algorithm 2: FRLS($f : \mathbb{S} \mapsto \mathbb{N}$)

$H \leftarrow (0, 0, \cdots, 0)$; ▷ *H-table initially all 0s*
sample s_c from $\mathbb{S}$ u.a.r.; $z_c \leftarrow f(s_c)$;
$s_b \leftarrow s_c$; $z_b \leftarrow z_c$; ▷ *best may otherwise get lost*
for $10^8 - 1$ times **do** ▷ *our termination criterion*
  $s_n \leftarrow$ swap 2 values in s_c u.a.r.; $z_n \leftarrow f(s_n)$;
  if $z_n < z_b$ **then** $s_b \leftarrow s_n$; $z_b \leftarrow z_n$;
  $H[z_c] \leftarrow H[z_c] + 1$; $H[z_n] \leftarrow H[z_n] + 1$;
  if $H[z_n] \leq H[z_c]$ **then** $s_c \leftarrow s_n$; $z_c \leftarrow z_n$;
return s_b, z_b ▷ *return preserved best*

behavior. A history of discovered solution behaviors is maintained and used to predict the behavior of new solutions. (Gravina et al., 2019) combine SS and NSLC.

Finally, the MAP-Elites algorithm by (Mouret and Clune, 2015) combines a performance objective f and a user-defined space of features that describe candidate solutions. MAP-Elites searches for the highest-performing solution in each cell of the discretized feature space.

Sharing techniques require a population and all the other methods discussed above were designed as optimization algorithms themselves. FFA, however, can be plugged into a wide range of optimization algorithms as long as their objective functions are discrete. Instead of using the objective values z computed by the objective function $f(s) = z$ when comparing solutions s, FFA prescribes using their observed encounter frequencies $H[z]$. This makes FFA invariant under all injective transformations of the objective function value, a property further distinguishing it from all related techniques (Weise et al., 2021b; Weise et al., 2023).

3 OUR APPROACH

The pure randomized local search algorithm RLS is illustrated in Algorithm 1. This algorithm starts by sampling a solution s_c from the set $\mathbb{S}$ of all permutations of the first n natural numbers uniformly at random (u.a.r.). It evaluates the objective function f and obtains the quality z_c of s_c. In a loop, it then creates a copy s_n of s_c in which two values are swapped, u.a.r.. The quality $z_n = f(s_n)$ of s_n is computed. If s_n is better than or equally good as s_c, it will replace s_c. The loop is repeated until the termination criterion is met, which, in our case, is the consumption of a total of 10^8 objective function evaluations (FEs, including the evaluation of the random initial solution).

We plug FFA into this algorithm and obtain the FRLS in Algorithm 2. While RLS accepts the new solution s_n if its objective value z_n is not worse than the objective value z_c of the current solution s_c, FRLS accepts s_n if the encounter frequency $H[z_n]$ of z_n in the selection decision is not higher than the encounter frequency $H[z_c]$ of z_c. For this purpose, it begins by filling the frequency table H with zeros at the beginning of the algorithm. In each iteration, $H[z_n]$ and $H[z_c]$ are both incremented by one and then replace z_n and z_c in the selection decision. This means that FRLS is not biased towards better solutions and will replace s_c with a worse s_n if its objective value z_n is encountered less than or equally often as z_c. Therefore, instead of returning s_c and z_c at the end, FRLS must remember the best-encountered solution and objective value in additional variables s_b and z_b, respectively.

4 EXPERIMENTS AND RESULTS

The QAPLIB by (Burkard et al., 1997) is a commonly used and continuously updated database of QAP benchmark instances and their solutions. It contains both real-life instances and randomly generated instances. In our experiments, we use all 134 instances of the latest version of the QAPLIB at the time of this writing, which is maintained by (Hahn and Anjos, 2018) and was last updated in 2018. From this resource, we also take the lower bounds lb of the objective functions f. For each instance, we perform 3 independent runs which, together with the many instances, are already sufficient to observe very clear differences in performance. The instances have the following properties:

- bur$n$* (Burkard and Offermann, 1977), 8 instances, $n = 26$, all optima known
- chr$n$* (Christofides and Benavent, 1989), 14 instances, $n \in \{12, 15, 18, 20, 22, 25\}$, all optima known
- els19 (Elshafei, 1977), 1 instance, $n = 19$, optimum known
- esc$n$* (Eschermann and Wunderlich, 1990), 19 instances, $n \in \{16, 32, 64, 128\}$, all optima known
- had$n$ (Hadley et al., 1992), 5 instances, $n \in \{12, 14, 16, 18, 20\}$, all optima known

- kran^* (Krarup and Pruzan, 1978), 3 instances, $n \in \{30, 32\}$, all optima known
- lipan^* (Li and Pardalos, 1992), 16 instances, $n \in \{20, 30, 40, 50, 60, 70, 80, 90\}$, all optima known
- nugn^* (Nugent et al., 1968), 15 instances, $n \in \{12, 14, 15, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 30\}$, all optima known
- rou$n$ (Roucairol, 1987), 3 instances, $n \in \{12, 15, 20\}$, all optima known
- scr$n$ (Scriabin and Vergin, 1975), 3 instances, $n \in \{12, 15, 20\}$, all optima known
- skon^* (Skorin-Kapov, 1990), 13 instances, $n\{42, 49, 56, 64, 72, 81, 90, 100\}$, *all optima unknown*
- ste36* (Steinberg, 1961), 3 instances, $n = 36$, all optima known
- tain^* (Taillard, 1991; Taillard, 1995), 26 instances, $n \in \{12, 15, 17, 20, 25, 30, 35, 40, 50, 60, 64, 80, 100, 150, 256\}$, *optima of 16 instances unknown*
- tho$n$ (Thonemann and Bölte, 1994), 3 instances, $n \in \{30, 40, 150\}$, *only optimum of tho30 known*
- wil$n$ (Wilhelm and Ward, 1987), 2 instances, $n \in \{50, 100\}$, *no optimum known*

We implement our algorithms using the moptipy (Weise and Wu, 2023) framework and run the experiments on a Windows 10 machine using Python 3.10 and the numba JIT.

Table 1 (continued in Table 2) shows the arithmetic mean of the best objective values achieved by RLS and FRLS over the 3 runs per QAPLIB instance. The last row, **# best**, tells us that FRLS achieved the best average result 113 times, while RLS did this only 35 times. The average result of FRLS hits the lower bound lb, i.e., is optimal 73 times. Its best-of-3-runs results (not tabulated) reach it 78 times. RLS achieves this feat only 14 respectively 20 times. In other words, not only does FRLS outperform RLS on 74% of the QAPLIB instances in terms of its average result, it also solves 58% of them to optimality.

In (Liang et al., 2022), it was found that the performance of FRLS may strongly depend on the number M of different objective values that an optimization problem exhibits. The good performance of FRLS on the esc$n$ problems may be caused by the many zeros in their flow matrix resulting in few different possible object values.

Exactly determining M for the QAPLIB instances would be another $\mathcal{NP}$-hard problem in itself. Therefore, we do not have the exact values of this measure available. However, we can approximate it using the estimate, or better, a lower bound m: Each run of FRLS maintains its own frequency table H and we collect these tables in our log files. We also log all improving moves that any algorithm makes, so we additionally have at least the strictly monotonous sequence of visited f-values for RLS. Finally, the website of the QAPLIB offers the best-known or even optimal solutions for all instances, which are better than our results on 42% of the instances. Therefore, by setting m to be the size of the joint set of all of these values of all runs, we can get a lower bound for M. When m is much smaller than our total computational budget over all runs of FRLS (for which we collect the complete H-tables), i.e., where $m \ll 3 * 10^8$, it should be a reasonable estimate of M. Otherwise, at least it informs us whether M is probably small or large. We therefore also include it in the tables.

Revisiting the results of both algorithms in Table 1 and Table 2 and considering them from the perspective of m confirms the findings by (Liang et al., 2022). If m of an instance is small, FRLS tends to solve the instance to optimality (and hit the lower bound lb), even if the scale n is not small (e.g., at lipa50a). Vice versa, the tables also show that FRLS is outperformed by RLS even on small problems if their m is large, see, e.g., tai15b. The comparatively good performance of FRLS on the tai$n$a instances versus the tai$n$b instances is also interesting because the former are usually considered as harder (Ochoa and Herrmann, 2018).

A remarkable piece of evidence of the exploration power of FRLS, which discovers most of the encountered objective values, are the high m-values for many instances. FRLS contributed 215 196 721 values to the estimation $m = 215\,196\,971$ for tai30b. Since we conducted only 3 runs at 10^8 FEs each, this means that 71% of *all* the solutions that these FRLS runs have sampled had *unique* objective values. If all solutions on a problem instance would have unique objective values, then FRLS would always accept the new solution s_n and hence become a random walk. But this does not seem to be the case: On tai20b, FRLS encountered 173 058 828 different objective values – and outperformed RLS by a margin of over 10%.

The strong ability to explore and keep improving of FRLS is further illustrated in Figure 1. Here, we plot the average *life* index of the objective function evaluation (FE) where the last improving move was made over the problem scale n. In other words: Each run of an algorithm on a given problem instance eventually stops improving its best-so-far solution. It may or may not have discovered the optimal solution by then, but after that, no more improvement is made (within the provided computational budget, at least). The index of the algorithm step when, for the last time in a run, a new (better) best-so-far solution is discov-

Table 1: The average result over 3 runs of the RLS and the FRLS on the 134 QAPLIB instances, in comparison with the lower bound lb of f and the number m of observed and known objective values as a lower bound for the number of possible different objective values. The best result is marked in **boldface**. (continued in Table 2)

instance	lb	m	RLS	FRLS	instance	lb	m	RLS	FRLS
bur26a	5 426 670	1 480 802	5 442 929	**5 434 256**	lipa60a	107 218	4 915	108 368	**107 461**
bur26b	3 817 852	1 021 194	3 838 077	**3 818 291**	lipa60b	2 520 135	456 660	3 016 957	**3 005 080**
bur26c	5 426 795	1 384 071	5 440 307	**5 428 857**	lipa70a	169 755	6 880	171 358	**170 429**
bur26d	3 821 225	945 677	3 833 028	**3 821 540**	lipa70b	4 603 200	651 696	**5 569 556**	5 642 958
bur26e	5 386 879	1 579 830	5 405 301	**5 389 526**	lipa80a	253 195	7 772	255 351	**254 606**
bur26f	3 782 044	1 111 729	3 793 182	**3 782 454**	lipa80b	7 763 962	940 457	**9 423 095**	9 650 856
bur26g	10 117 172	2 672 208	10 145 555	**10 127 889**	lipa90a	360 630	9 976	363 412	**362 571**
bur26h	7 098 658	1 876 059	7 141 228	**7 101 399**	lipa90b	12 490 441	1 277 577	**15 173 637**	15 617 417
chr12a	9 552	33 801	14 899	**9 552**	nug12	578	232	606	**578**
chr12b	9 742	33 627	14 589	**9 742**	nug14	1 014	366	1 037	**1 014**
chr12c	11 156	33 377	14 939	**11 156**	nug15	1 150	432	1 182	**1 150**
chr15a	9 896	52 353	16 015	**9 896**	nug16a	1 610	532	1 673	**1 610**
chr15b	7 990	53 657	11 952	**7 990**	nug16b	1 240	483	1 297	**1 240**
chr15c	9 504	50 900	14 913	**9 504**	nug17	1 732	608	1 813	**1 732**
chr18a	11 098	66 156	18 142	**11 098**	nug18	1 930	650	1 978	**1 930**
chr18b	1 534	3 083	1 648	**1 534**	nug20	2 570	826	2 681	**2 570**
chr20a	2 192	8 429	3 325	**2 192**	nug21	2 438	970	2 510	**2 438**
chr20b	2 298	8 307	3 556	**2 335**	nug22	3 596	1 531	3 759	**3 596**
chr20c	14 142	91 709	31 659	**14 142**	nug24	3 488	1 254	3 608	**3 488**
chr22a	6 156	16 932	6 824	**6 156**	nug25	3 744	1 277	3 950	**3 744**
chr22b	6 194	16 846	6 861	**6 215**	nug27	5 234	1 860	5 470	**5 234**
chr25a	3 796	21 052	6 509	**3 796**	nug28	5 166	1 728	5 417	**5 166**
els19	17 212 548	30 545 903	25 266 593	**18 821 866**	nug30	6 124	2 018	6 439	**6 124**
esc16a	68	34	**68**	**68**	rou12	235 528	58 475	248 938	**235 528**
esc16b	292	22	**292**	**292**	rou15	354 210	97 118	378 899	**354 210**
esc16c	160	73	**160**	**160**	rou20	725 522	175 690	759 802	**725 522**
esc16d	16	36	**16**	**16**	scr12	31 410	28 833	33 079	**31 410**
esc16e	28	29	**28**	**28**	scr15	51 140	53 073	56 646	**51 140**
esc16f	0	1	**0**	**0**	scr20	110 030	120 453	126 571	**110 030**
esc16g	26	35	**26**	**26**	sko42	15 332	4 201	16 351	**15 812**
esc16h	996	272	**996**	**996**	sko49	22 650	5 802	23 909	**23 403**
esc16i	14	37	**14**	**14**	sko56	33 385	8 202	35 337	**34 467**
esc16j	8	20	**8**	**8**	sko64	47 017	10 379	49 509	**48 524**
esc32a	130	253	151	**130**	sko72	64 455	13 439	67 707	**66 378**
esc32b	168	124	183	**168**	sko81	88 359	17 353	92 575	**91 107**
esc32c	642	194	**642**	**642**	sko90	112 423	20 859	117 639	**115 853**
esc32d	200	117	205	**200**	sko100a	143 846	25 618	153 965	**152 557**
esc32e	2	50	**2**	**2**	sko100b	145 522	26 389	156 111	**154 557**
esc32g	6	37	**6**	**6**	sko100c	139 881	25 903	151 014	**148 430**
esc32h	438	175	467	**438**	sko100d	141 289	25 616	151 863	**150 203**
esc64a	116	124	**116**	**116**	sko100e	140 893	26 623	151 569	**149 795**
esc128	64	192	65	**64**	sko100f	140 691	25 266	151 695	**149 570**
had12	1 652	228	1 665	**1 652**	ste36a	9 526	14 213	10 213	**9 526**
had14	2 724	394	2 753	**2 724**	ste36b	15 852	96 128	17 766	**15 852**
had16	3 720	478	3 815	**3 720**	ste36c	8 239 110	7 804 921	**8 970 338**	10 698 219
had18	5 358	622	5 413	**5 358**	tai12a	224 416	64 051	240 311	**224 416**
had20	6 922	856	6 969	**6 922**	tai12b	39 464 925	60 287 923	45 156 248	**39 492 474**
kra30a	88 900	8 200	94 843	**88 900**	tai15a	388 214	94 668	400 575	**388 214**
kra30b	91 420	8 581	95 967	**91 420**	tai15b	51 765 268	35 623 423	**51 943 701**	52 001 756
kra32	88 700	8 828	93 730	**88 700**	tai17a	491 812	122 615	523 634	**491 812**
lipa20a	3 683	435	3 795	**3 683**	tai20a	703 482	178 309	751 881	**704 195**
lipa20b	27 076	10 778	31 241	**27 076**	tai20b	122 455 319	173 058 953	143 287 002	**129 766 839**
lipa30a	13 178	1 088	13 442	**13 178**	tai25a	1 167 256	252 308	1 231 845	**1 174 603**
lipa30b	151 426	57 162	178 015	**151 426**	tai25b	344 355 646	202 832 378	**384 043 042**	395 447 601
lipa40a	31 538	1 976	32 042	**31 538**	tai30a	1 706 855	319 665	1 918 997	**1 853 616**
lipa40b	476 581	184 489	563 999	**476 581**	tai30b	637 117 113	215 196 971	**710 795 743**	721 008 038
lipa50a	62 093	3 296	62 902	**62 093**	tai35a	2 216 627	397 009	2 559 439	**2 509 553**
lipa50b	1 210 244	348 151	1 438 601	**1 308 415**	tai35b	269 532 400	125 920 739	**317 695 376**	334 904 454

Table 2: Table 1 continued.

instance	*lb*	*m*	RLS	FRLS	instance	*lb*	*m*	RLS	FRLS
tai40a	2 843 274	468 546	3 307 957	**3 281 287**	tai100b	1 151 591 000	152 455 325	**1 240 769 163**	1 543 004 655
tai40b	608 808 400	165 765 853	**693 265 760**	806 047 127	tai150b	441 786 736	43 282 106	**509 821 471**	612 165 283
tai50a	4 390 920	612 489	**5 152 389**	5 245 447	tai256c	44 095 032	5 758 252	**44 940 419**	48 194 560
tai50b	431 090 700	129 307 016	**504 050 091**	590 061 093	tho30	149 936	63 292	157 237	**149 936**
tai60a	6 325 978	752 294	**7 553 963**	7 717 479	tho40	226 490	102 223	251 221	**240 708**
tai60b	592 371 800	137 418 134	**643 368 525**	791 205 717	tho150	7 620 628	918 879	**8 319 988**	8 855 816
tai64c	1 855 928	1 691 310	**1 860 059**	1 861 098	wil50	48 121	6 760	49 465	**48 835**
tai80a	11 657 010	1 019 112	**14 030 598**	14 523 961	wil100	268 955	26 012	275 203	**273 622**
tai80b	786 298 800	120 691 529	**873 374 711**	1 075 394 622			**# best**	35	**113**
tai100a	17 853 840	1 268 760	**21 828 809**	22 720 707					

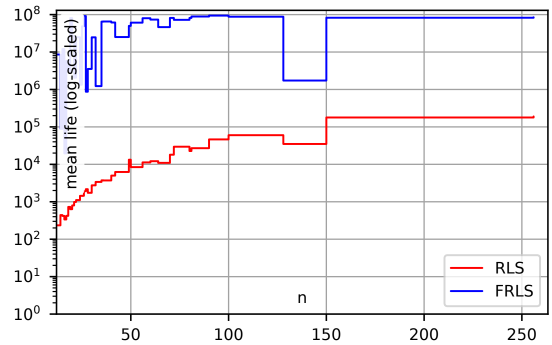

Figure 1: The average *life* index of the objective function evaluation (FE) where the last improving move was made, plotted in log-scale over the problem scale n.

ered, averaged over the runs, is presented as *life*.

We find that the time during which the RLS can keep improving increases slightly with n. However, it remains roughly in the range of at most a few 100 000 FEs. Over almost all problem scales, FRLS can keep improving for, basically, the complete available budget of 10^8 FEs. This strongly indicates that if we had allocated not 10^8 FEs but 10^{10}, as it was done in (Weise et al., 2021b; Weise et al., 2023; Liang et al., 2022; Liang et al., 2024), we very likely would have seen several more instances solved to optimality. The single downward rectangular slot in both curves in the diagram is caused by esc128, at which both algorithms converge earlier (FRLS to the optimum, after which no further improvement is possible). The next larger instances are at $n = 150$ where the trend resumes.

We now plot the progress of the two algorithms in terms of the best-so-far objective value divided by the lower bound lb of the objective function f over time measured in FEs and averaged over all the runs and instances in each of the 15 groups of QAPLIB. Instance esc16f with $lb = 0$ is omitted. From Figure 2, it is visible that FRLS finds better average end result qualities on all groups except ste$n$ and tai$n$. Even on these groups, it would have probably overtaken RLS if we had given more runtime. In most of the diagrams, RLS is initially faster and then stagnates, while FRLS steadily and continuously keeps improving.

5 CONCLUSIONS

In the past, Frequency Fitness Assignment (FFA) has led to surprisingly good results on several $\mathcal{NP}$-hard optimization problems, including Max-Sat (Weise et al., 2021b; Weise et al., 2023), the JSSP (Weise et al., 2021a; de Bruin et al., 2023), and the TSP (Liang et al., 2022; Liang et al., 2024). In this work, we conclusively showed that FFA can achieve this on one more of these classical hard tasks from operations research: the Quadratic Assignment Problem (QAP).

We find that the FFA-based randomized local search FRLS does not just find better solutions than the objective-guided RLS algorithm on the vast majority of the QAPLIB instances, it also keeps improving its current best solution for the complete computational budget of 10^8 FEs that we assigned to the runs. With this budget, it can discover the optimal solutions of over 58% of the QAPLIB instances. Had we assigned a larger budget – (Liang et al., 2022; Liang et al., 2024; Weise et al., 2021b; Weise et al., 2023) use 10^{10} FEs – we would likely have seen even more instances solved.

We furthermore confirm the remarkable ability of FFA to discover very diverse solutions (at least from the perspective of the objective function). It is known that on the QAP, many solutions tend to have the same objective values (Tayarani-N. and Prügel-Bennett, 2015). Yet, on some of the instances, more than half of the objective values discovered by FRLS were unique.

The QAP is strongly related to the TSP (Dréo et al., 2006). (Liang et al., 2022; Liang et al., 2024) found that the FFA performance strongly depends on the number M of possible different objective values. We are the first to report a lower bound and estimate m of M for each of the QAPLIB instances. We confirm that, indeed, if m is high, then the performance of the FRLS declines in comparison to the objective-guided RLS, adding to our understanding of the performance of this algorithm.

(Liang et al., 2022; Liang et al., 2024) showed

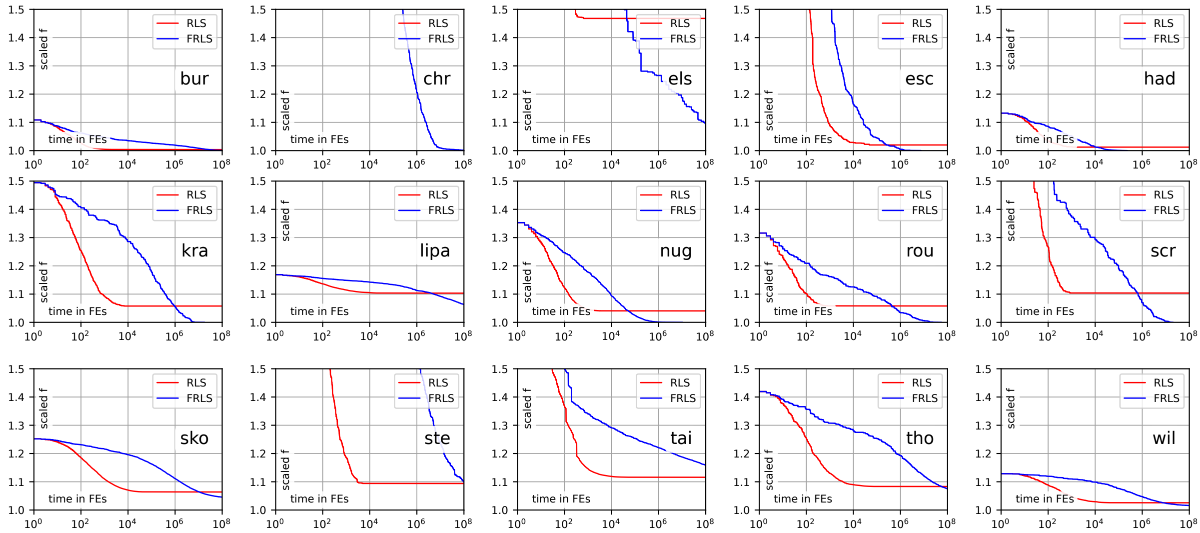

Figure 2: The progress in terms of the best-so-far objective value divided by the lower bound *lb* of *f* averaged over all runs and instances of an instance group and plotted over the time measured in FEs (log-scaled). Instance esc16f is omitted from this statistic (the esc group) due to having a lower bound of 0. On the chr*n* instances, RLS is off the scale.

that the performance of the FRLS can significantly be improved if it is hybridized with RLS sharing the budget in a round-robin fashion and if simulated annealing (SA) is used as a basic algorithm. Investigating plugging FFA in other algorithms on the QAP, such as the SA by (Wilhelm and Ward, 1987), the tabu search by (Taillard, 1991), the hybrid evolutionary algorithms by (Fleurent and Ferland, 1993; Merz and Freisleben, 1999), or the ant colony optimization method by (Gambardella et al., 1999), is therefore an important branch of our future work.

ACKNOWLEDGEMENTS

The authors acknowledge support from the Project of National Natural Science Foundation of China 62406095, the Project of Natural Science Foundation of Anhui Province 2308085MF213, the Key Research Plan of Anhui Province 2022k07020011, the University Scientific Research Innovation Team Project of Anhui Province 2022AH010095, as well as the Hefei Specially Recruited Foreign Expert program and the Hefei Foreign Expert Office program.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2007). *k*-means++: The advantages of careful seeding. In Bansal, N., Pruhs, K., and Stein, C., editors, *18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07), January 7–9, 2007, New Orleans, LA, USA*, pages 1027–1035, Philadelphia, PA, USA. SIAM.

Burkard, R. E., Çela, E., Pardalos, P. M., and Pitsoulis, L. S. (1998). The quadratic assignment problem. In Du, D. and Pardalos, P. M., editors, *Handbook of Combinatorial Optimization*, pages 1713–1809. Springer, New York, NY, USA. doi:10.1007/978-1-4613-0303-9_27.

Burkard, R. E., Karisch, S. E., and Rendl, F. (1997). QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 10:391–403. doi:10.1023/A:1008293323270.

Burkard, R. E. and Offermann, J. (1977). Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research*, 21:B121–B132. doi:10.1007/BF01918175.

Christofides, N. and Benavent, E. (1989). An exact algorithm for the quadratic assignment problem on a tree. *Operations Research*, 37(5):760–768. doi:10.1287/opre.37.5.760.

Çubukçuoğlu, C., Nourian, P., Tasgetiren, M. F., Sariyildiz, I. S., and Azadi, S. (2021). Hospital layout design renovation as a quadratic assignment problem with geodesic distances. *Journal of Building Engineering*, 44:102952. doi:10.1016/j.jobe.2021.102952.

Cully, A. and Demiris, Y. (2018). Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259. doi:https://doi.org/10.1109/TEVC.2017.2704781.

de Bruin, E., Thomson, S. L., and van den Berg, D. (2023). Frequency fitness assignment on JSSP: A critical review. In Correia, J., Smith, S. L., and Qaddoura, R., editors, *26th European Conference on Applications of Evolutionary Computation (EvoApplications'23), Held as Part of EvoStar 2023, April 12-14, 2023, Brno, Czech Republic*, volume 13989 of *Lecture Notes in Computer Science*, pages 351–363, Cham, Switzerland. Springer. doi:10.1007/978-3-031-30229-9_23.

de Klerk, E. and Sotirov, R. (2010). Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. *Mathematical Programming*, 122(2):225–246. doi:10.1007/S10107-008-0246-5.

Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In Schaffer, J. D., editor, *3rd International Conference on Genetic Algorithms, June 1989, George Mason University, Fairfax, VA, USA*, pages 42–50, San Francisco, CA, USA. Morgan Kaufmann.

Dréo, J., Pétrowski, A., Siarry, P., and Taillard, É. D. (2006). *Metaheuristics for Hard Optimization – Methods and Case Studies*. Springer, Berlin/Heidelberg, Germany. doi:10.1007/3-540-30966-7.

Drezner, Z., Hahn, P. M., and Taillard, É. D. (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):65–94. doi:10.1007/S10479-005-3444-Z.

Elshafei, A. N. (1977). Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, 28:167–179.

Eschermann, B. and Wunderlich, H. (1990). Optimized synthesis of self-testable finite state machines. In *20th International Symposium on Fault-Tolerant Computing (FFTCS'20), June 26–28, 1990, Newcastle upon Tyne, UK*, Piscataway, NJ, USA. IEEE. doi:10.1109/FTCS.1990.89393.

Fleurent, C. and Ferland, J. A. (1993). Genetic hybrids for the quadratic assignment problem. In Pardalos, P. M. and Wolkowicz, H., editors, *Quadratic Assignment and Related Problems, Proceedings of a DIMACS Workshop, May 20–21, 1993*, pages 173–187, Providence, RI, USA. American Mathematical Society.

Gambardella, L., Taillard, É. D., and Dorigo, M. (1999). Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176. doi:10.1057/palgrave.jors.2600676.

Glover, F. W. and Taillard, É. D. (1993). A user's guide to tabu search. *Annals of Operational Research*, 41(1):1–28. doi:10.1007/BF02078647.

Goldberg, D. E. and Richardson, J. T. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., editor, *2nd International Conference on Genetic Algorithms and their Applications, July 28–31, 1987, Cambridge, MA, USA*, pages 41–49, East Sussex, England, UK. Psychology Press.

Gravina, D., Liapis, A., and Yannakakis, G. N. (2016). Surprise search: Beyond objectives and novelty. In Friedrich, T., Neumann, F., and Sutton, A. M., editors, *Genetic and Evolutionary Computation Conference (GECCO'16), July 20–24, 2016, Denver, CO, USA*, pages 677–684, New York, NY, USA. ACM. doi:10.1145/2908812.2908817.

Gravina, D., Liapis, A., and Yannakakis, G. N. (2019). Quality diversity through surprise. *IEEE Transactions on Evolutionary Computation*, 23(4):603–616. doi:10.1109/TEVC.2018.2877215.

Hadley, S. W., Rendl, F., and Wolkowicz, H. (1992). A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17(3):727–739.

Hafiz, F. M. F. and Abdennour, A. (2016). Particle swarm algorithm variants for the quadratic assignment problems – A probabilistic learning approach. *Expert Systems with Applications*, 44:413–431. doi:10.1016/J.ESWA.2015.09.032.

Hahn, P. and Anjos, M. (2018). *Website for "QAPLIB – A Quadratic Assignment Problem Library"*. Polytechnique Montréal, Montréal, Canada. https://qaplib.mgi.polymtl.ca.

Horng, J., Chen, C. C., Liu, B., and Kao, C. (2000). Resolution of quadratic assignment problems using an evolutionary algorithm. In Zalzala, A. M. S., editor, *Congress on Evolutionary Computation (CEC'00), July 16–19, 2000, La Jolla, CA, USA*, pages 902–909, Piscataway, NJ, USA. IEEE. doi:10.1109/CEC.2000.870736.

Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: Journal of the Econometric Society*, 25(1):53–76. doi:10.2307/1907742.

Krarup, J. and Pruzan, P. M. (1978). Computer-aided layout design. *Mathematical Programming Study*, 9:75–94.

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In Bullock, S., Noble, J., Watson, R. A., and Bedau, M. A., editors, *Eleventh International Conference on the Synthesis and Simulation of Living Systems (ALIFE'08), August 5–8, 2008, Winchester, UK*, pages 329–336, Cambridge, MA, USA. MIT Press. http://mitpress2.mit.edu/books/chapters/0262287196chap43.pdf.

Lehman, J. and Stanley, K. O. (2011a). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223. doi:10.1162/EVCO_A_00025.

Lehman, J. and Stanley, K. O. (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. In Krasnogor, N. and Lanzi, P. L., editors, *13th Annual Genetic and Evolutionary Computation Conference (GECCO'11), July 12–16, 2011, Dublin, Ireland*, pages 211–218, New York, NY, USA. ACM. doi:10.1145/2001576.2001606.

Li, Y. and Pardalos, P. M. (1992). Generating quadratic assignment test problems with known optimal permutations. *Computational Optimization and Applications*, 1(2):163–184. doi:10.1007/BF00253805.

Liang, T., Wu, Z., Lässig, J., van den Berg, D., Thomson, S. L., and Weise, T. (2024). Addressing the traveling salesperson problem with frequency fitness assignment and hybrid algorithms. *Soft Computing*. doi:10.1007/s00500-024-09718-8.

Liang, T., Wu, Z., Lässig, J., van den Berg, D., and Weise, T. (2022). Solving the traveling salesperson problem using frequency fitness assignment. In

Ishibuchi, H., Kwoh, C., Tan, A., Srinivasan, D., Miao, C., Trivedi, A., and Crockett, K. A., editors, *IEEE Symposium Series on Computational Intelligence (SSCI'22), December 4–7, 2022, Singapore*, pages 360–367, Piscataway, NJ, USA. IEEE. doi:10.1109/SSCI51031.2022.10022296.

Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690. doi:10.1016/j.ejor.2005.09.032.

Mahfoud, S. W. (1997). Niching methods. In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, pages C6.1:1–4. Institute of Physics Publishing, Bristol, UK. ISBN: 0-7503-0392-1.

Merz, P. and Freisleben, B. (1999). A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Congress on Evolutionary Computation (CEC'99), July 6–9, 1999, Washington, DC, USA*, pages 2063–2070, Los Alamitos, CA, USA. IEEE. doi:10.1109/CEC.1999.785529.

Misevičius, A. (2005). A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, 30:95–111. doi:10.1007/s10589-005-4562-x.

Misevičius, A. (2008). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. Working paper, Kaunas University of Technology, Kaunas, Lithuania.

Mouret, J. and Clune, J. (2015). *Illuminating Search Spaces by Mapping Elites*. Cornell University Library, Ithaca, NY, USA. arXiv:1504.04909v1 [cs.AI] 20 Apr 2015, http://arxiv.org/abs/1504.04909.

Mouret, J. and Cully, A. (2024). *Quality-Diversity Optimisation Algorithms: List of Papers*. GitHub, Inc., San Francisco, CA, USA. https://quality-diversity.github.io/papers visited on 2024-05-30.

Nugent, C. E., Vollmann, T. E., and Ruml, J. (1968). An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16(1):150–173. doi:10.1287/opre.16.1.150.

Ochoa, G. and Herrmann, S. (2018). Perturbation strength and the global structure of QAP fitness landscapes. In Auger, A., Fonseca, C. M., Lourenço, N., Machado, P., Paquete, L., and Whitley, L. D., editors, *15th International Conference on Parallel Problem Solving from Nature (PPSN XV), September 8–12, 2018, Coimbra, Portugal, Part II*, volume 11102 of *Lecture Notes in Computer Science*, pages 245–256, New York, NY, USA. Springer. doi:10.1007/978-3-319-99259-4_20.

Peng, J., Mittelmann, H. D., and Li, X. (2010). A new relaxation framework for quadratic assignment problems based on matrix splitting. *Mathematical Programming Computation*, 2(1):59–77. doi:10.1007/S12532-010-0012-6.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In Fukuda, T. and Furuhashi, T., editors, *1996 IEEE International Conference on Evolutionary Computation, May 20–22, 1996 Nayoya University, Japan*, pages 798–803, Piscataway, NJ, USA. IEEE. doi:10.1109/ICEC.1996.542703.

Roucairol, C. (1987). Du séquentiel au parallèle, la recherche arborescente et son application á la programmation quadratique en variables 0 et 1. Thése doct.sc. mathematiques, Université Pierre et Marie Curie, Paris, France.

Sahni, S. and Gonzalez, T. (1976). $\mathcal{P}$-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565. doi:10.1145/321958.321975.

Scriabin, M. and Vergin, R. C. (1975). Comparison of computer algorithms and visual based methods for plant layout. *Management Science*, 22(2):172–187. doi:10.1287/mnsc.22.2.172.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA. ISBN: 1107057132.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2(1):33–45. doi:10.1287/ijoc.2.1.33.

Soroush, H. M. (2011). Scheduling in stochastic bicriteria single machine systems with set-up times. *International Journal of Planning and Scheduling*, 1(1–2):109–145. doi:10.1504/IJPS.2011.044605.

Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *SIAM Review*, 3(1):37–50. doi:10.1137/1003003.

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539. doi:10.1016/j.ejor.2005.01.066.

Taillard, É. D. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4–5):443–455. doi:10.1016/S0167-8191(05)80147-4.

Taillard, É. D. (1995). Comparison of iterative searches for the quadratic assingnment problem. *Location Science*, 3(2):87–105. doi:10.1016/0966-8349(95)00008-6.

Taillard, É. D. and Gambardella, L. (1997). Adaptive memories for the quadratic assignment problem. Technical Report IDSIA-87-97, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland. http://mistic.heig-vd.ch/taillard/articles.dir/TaillardGambardella1997.pdf.

Talbi, E., Roux, O., Fonlupt, C., and Robillard, D. (2001). Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4):441–449. doi:10.1016/S0167-739X(99)00124-7.

Tayarani-N., M. and Prügel-Bennett, A. (2015). Quadratic assignment problem: A landscape analysis. *Evolutionary Intelligence*, 8(4):165–184. 10.1007/s12065-015-0132-z.

Thomson, S. L., Ochoa, G., van den Berg, D., Liang, T., and Weise, T. (2024). Entropy, search trajectories, and explainability for frequency fitness assignment. In *18th International Conference on Parallel Problem Solving from Nature (PPSN XVII), September 14-18, 2024, Hagenberg im Mühlkreis, Austria*, Berlin/Heidelberg, Germany. Springer. Accepted for Publication.

Thonemann, U. W. and Bölte, A. (1994). An improved simulated annealing algorithm for the quadratic assignment problem. Working paper, School of Business, Department of Production and Operations Research, University of Paderborn, Paderborn, Germany.

Treadgold, N. K. and Gedeon, T. D. (1998). Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm. *IEEE Transactions on Neural Networks*, 9(4):662–668. doi:10.1109/72.701179.

Weise, T., Chiong, R., and Tang, K. (2012). Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936. doi:10.1007/s11390-012-1274-4.

Weise, T., Li, X., Chen, Y., and Wu, Z. (2021a). Solving job shop scheduling problems without using a bias for good solutions. In *Genetic and Evolutionary Computation Conference (GECCO'21), July 10-14, 2021, Lille, France, Companion Volume*, pages 1459–1466, New York, NY, USA. ACM. doi:10.1145/3449726.3463124.

Weise, T., Niemczyk, S., Chiong, R., and Wan, M. (2011). A framework for multi-model EDAs with model recombination. In Chio, C. D., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A., Guervós, J. J. M., Neri, F., Preuss, M., Richter, H., Togelius, J., and Yannakakis, G. N., editors, *Applications of Evolutionary Computation (EvoApplications'11), April 27–29, 2011, Torino, Italy, Part I*, volume 6624 of *Lecture Notes in Computer Science*, pages 304–313, Berlin/Heidelberg, Germany. Springer. doi:10.1007/978-3-642-20525-5_31.

Weise, T., Wan, M., Tang, K., Wang, P., Devert, A., and Yao, X. (2014). Frequency fitness assignment. *IEEE Transactions on Evolutionary Computation*, 18(2):226–243. doi:10.1109/TEVC.2013.2251885.

Weise, T. and Wu, Z. (2023). Replicable self-documenting experiments with arbitrary search spaces and algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO'23 Companion), July 15-19, 2023, Lisbon, Portugal*, New York, NY, USA. ACM. doi:10.1145/3583133.3596306.

Weise, T., Wu, Z., Li, X., and Chen, Y. (2021b). Frequency fitness assignment: Making optimization algorithms invariant under bijective transformations of the objective function value. *IEEE Transactions on Evolutionary Computation*, 25(2):307–319. doi:10.1109/TEVC.2020.3032090.

Weise, T., Wu, Z., Li, X., Chen, Y., and Lässig, J. (2023). Frequency fitness assignment: Optimization without bias for good solutions can be efficient. *IEEE Transactions on Evolutionary Computation*, 27(4):980–992. doi:10.1109/TEVC.2022.3191698.

Weise, T., Zapf, M., Chiong, R., and Nebro, A. J. (2009). Why is optimization difficult? In Chiong, R., editor, *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*, pages 1–50. Springer, Berlin/Heidelberg, Germany. doi:10.1007/978-3-642-00267-0_1.

Wilhelm, M. R. and Ward, T. L. (1987). Solving quadratic assignment problems by 'simulated annealing'. *IIE Transactions*, 19(1):107–119. doi:10.1080/07408178708975376.

Zhang, Q., Sun, J., Tsang, E. P. K., and Ford, J. A. (2006). Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem. In Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a New Evolutionary Computation – Advances in the Estimation of Distribution Algorithms*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 281–292. Springer, Berlin/Heidelberg, Germany. doi:10.1007/3-540-32494-1_12.