University of Utrecht

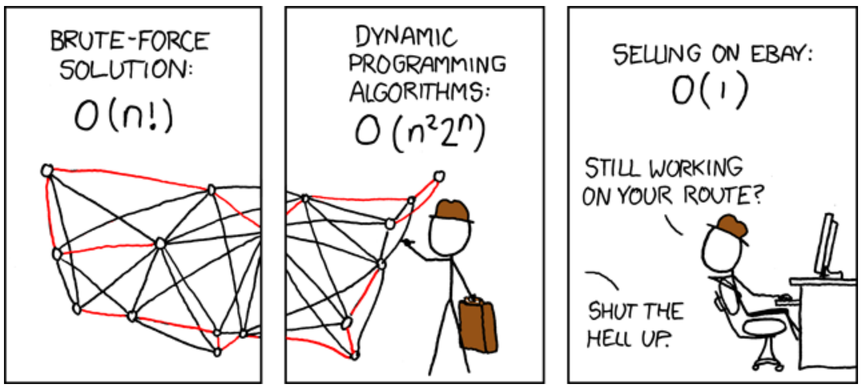
MSc Technical Artificial Intelligence

Thesis Project - ICA-3318583

A Bandit-Inspired Memetic Algorithm for Quadratic Assignment Problems

Author:
Francesco PUGLIERIN

Supervisors:
Prof. Dr. John-Jules MEYER
Dr. Gerard VREESWIJK
Dr. Marco WIERING



August 2012

francesco@puglier.in

Abstract

In this thesis a new metaheuristic for combinatorial optimization is proposed, with focus on the Quadratic Assignment Problem as the hard-problem of choice - a choice that is reflected in the name of the method, BIMA-QAP. The algorithm employs a memetic structure and stores information on the single components along the search. This information is used to guide the search, through an operator inspired by the solution approaches to the Multi-Armed Bandit model.

Once the algorithm has been laid out and its set of parameters defined, its implementation has been extensively tested under a Naive-Bayesian assumption of independence among the parameters. The results show that BIMA-QAP consistently performs better than Multi-start Local Search, and the new operator *perturb()* alters the solutions better than a randomized approach.

to my mother, my father, my sister and my cat

Contents

1 Introduction **1**

1.1 The Scope of this Work 2

1.2 Structure of the Thesis 2

2 Metaheuristics **5**

2.1 A Problem of Optimization 5

2.1.1 Getting More Formal 6

2.1.2 Combinatorial Optimization 7

2.2 Hard Problems 7

2.3 Optimization Methods 8

2.3.1 Exact Methods 8

2.3.2 Approximate Methods 9

2.4 Metaheuristics 9

2.4.1 Fundamental Classification 10

2.4.2 Trajectory Methods 11

2.4.3 Population-based Methods 12

2.4.4 Hybrid Approaches 13

2.4.5 Intensification and Diversification 14

2.5 The Quadratic Assignment Problem 15

2.5.1 Mathematical Formulation 16

2.5.2 Motivation and Applications 17

2.5.3 Solution and Neighbourhood Representation 17

2.5.4 Approaching the QAP 20

3 Contribution **21**

3.1 Hybrid Structure 22

3.2 Selective Components Memory 23

3.3 A Matter of Balance 23

3.4 Tweaking the Formula 25

3.4.1 First Term: Scaling the Fitness 26

3.4.2 Second Term: Managing the Pulls 27

3.4.3 A Dynamic Set of Arms 28

3.5 The Solutions Pool 29

3.6 The *perturb()* Operator Defined 30

3.7 The Algorithm as a Whole 31

3.7.1 Structure and Methods 31

3.7.2 Parameters 34

4 Results **37**
4.1 Implementation 37
4.2 Experimental Approach 37
4.3 Experimental Results 40
4.3.1 Results 41
4.3.2 Comparing the Experiments 45
4.4 Summary and Remarks 54

5 Conclusions and Future Work **57**
5.1 Looking Back 57
5.2 On the Results 57
5.3 Future Work 58

A Detailed Experimental Data **67**

1 Introduction

There is an empirical rule in informatics that goes under the name of Moore's Law. First stated in 1965 [1], up to now it correctly predicted the number of transistors in integrated circuits to double every two years, with a proportional impact on their performance.

Almost fifty years later, the CPU found in a modern cellphone can crunch instructions faster than the first supercomputers in use back then[1]. Mainframes are many orders of magnitude faster, and nowadays we depend on fast computers for tasks that range from complex meteorological simulations to rendering entire blockbusters frame by frame - tasks that would have been unfeasible for the machines of our recent past.

Yet there are limits that the current computational technology is not expecting to catch up anytime soon. Limits that are hard-coded in mathematics itself, under the form of *Hard Problems*. Hard problems cannot be solved in polynomial time, and their complexity greatly increases with size; for some particularly complex problems, the problematic input size can be smaller than many would expect. In the case of the Quadratic Assignment Problem (QAP), an archetypal Hard Problem that deals with optimally matching a set of n facilities to a set of n locations, currently instances of size greater than 30 are considered intractable. This limit is going to increase over time, but there will always be instances too big to be solved by exhaustive search.

The most common approach is then to be happy with an *approximate* solution, and to research techniques that are able to provide solutions of acceptable quality in an affordable time frame. Meta-heuristics are the most common of such techniques, and have the desirable property of being problem-independent. This because they approach the task by analyzing the structure of the solution more than the nature of the problem itself.

Many of such algorithms have been introduced in the past decades. Genetic Algorithms [2] adopt a different approach from Tabu Search [3], which in turn doesn't behave like Simulated Annealing [4]. Yet they have in common many inspiring principles, and design elements are often shared across methods. This thesis aims to combine several ideas that were object of extensive past research, to define a new method for finding approximate solutions to QAP instances.

[1]As of 2012, *smartphones* are nearing the 50 MFLOPS mark, far more than the CDC7600 - the world's fastest supercomputer in 1969/1970.

1.1 The Scope of this Work

As will be briefly discussed in Chapter 2, there are theoretical results that prove that no metaheuristic can outperform every other in all the problems. This pushed researchers to pursue different approaches and to combine existing ones, and it has been a fundamental stimulus to this project as well.

The scope of this work is to design, implement and evaluate a new method for solving instances of the Quadratic Assignment Problem. While the implementation is somewhat of a technicality, in the field of metaheuristics the experimental analysis of an algorithm is as important as the theoretical concepts behind it. It is impossible to assess the long-term effectiveness of a good method without running it first, so the matter of testing has been taken very seriously.

The algorithm has a memetic structure and uses several matrices to collect data about the *components* of the solutions alongside the search. This data is used to modify and combine the solutions, drawing inspiration from the Multi-Armed Bandit framework [5]. The resulting method is called BIMA-QAP, which stands for Bandit-Inspired Memetic Algorithm for Quadratic Assignment Problems.

The process of defining an effective metaheuristic for dealing with hard problems necessarily involves some trial and error. Defining a set of parameters to be tweaked is not only a way to suit the algorithm to the different instances, but it is also important to estimate the influence of the various design elements on the final results.

The final target is to get an algorithm that shows a converging behavior, is effective in exploring the search space and outperforms Multi-start Local Search (MLS) - which is the simplest of metaheuristics and is discussed, among other methods, in Chapter 2.

The next section will describe the structure of this thesis more in details.

1.2 Structure of the Thesis

The extent of Chapter 2 is broader than its name would suggest. It starts by the sheer concept of (combinatorial) optimization problems to then briefly characterize their Hard subset. Talking about approximate optimization methods follows logically, and the discussion on Metaheuristics themselves starts in section 2.4; the section contains some fundamental categorization, a brief historical background and a first mention of the exploration versus exploitation dilemma. The focus then switches back to the problems when the QAP is introduced and formalized in the last section of the chapter; here the important concept of 2-opt neighborhood is discussed, together with an efficient way of encoding solutions.

Once some theoretical background has been laid out, the topic of Chapter 3 is the BIMA-QAP method itself. Designing a metaheuristic is a complex process, and involves a synergy of different elements. After defining the basic structure of the algorithm, the chapter opens a parenthesis on the Multi-Armed Bandit framework and on UCB [6] as a technique to tackle it. It then gets back to the method itself by describing the process of intuitively adapting the formula from an optimization problem to another - a process that culminates in the definition of the operator *perturb()* in section 3.6. Before that, Section 3.5 explains how BIMA-QAP manages its pool of solutions, and suggests the inclusion of additional matrices to better model the search neighborhoods. An overview of the algorithm, its methods and its parameters concludes the Contribution chapter.

The crucial importance of experimentally assessing the performance of a method has been already stressed out; the lengthy Chapter 4 deals precisely with that. After the quick overview on the implementation offered by section 4.1, the experimental approach is discussed in the following section. Finding the optimal settings for a method is an optimization problem in itself, so the effects of the parameters are analyzed individually. A set of standard settings has been chosen for the experiment $e00$; then, for each of the parameters, a number of experiments has been defined. Two additional tests are in place, to compare the performance of BIMA-QAP to MLS and to itself with a random components selection in place of the custom operator. Some results are presented in section 4.3, followed by a comparative analysis of the experiments on each instance. Section 4.4 ends the chapter by summarizing the findings.

Chapter 5 concludes the thesis with some brief final assessment of the research done; Appendix A collects in a tabular form many experimental results that didn't make it to Chapter 4.

2 Metaheuristics

This chapter is meant to provide some theoretical background over the thesis subject. It starts by discussing the nature of optimization problems and their characteristics, to then shift to possible solution methods. It later gets more focused, with a section about metaheuristics and one that deals with the optimization problem of choice, the Quadratic Assignment Problem.

The sources that inspired the bulk of the chapter are the book *"Metaheuristics"* by El-Ghazali Talbi [7], the book *"Essential of Metaheuristics"* by Sean Luke [8], the paper where Blum and Roli introduce the I&D frame ([9], see section 2.4.5) and the work of Stützle on the Quadratic Assignment Problem (e.g. [10]). Several other sources are cited along the chapter.

2.1 A Problem of Optimization

Operations Research is an interdisciplinary mathematical science that aims at providing optimal or near-optimal solutions to complex decision-making problems. While its origin dates back to at least the 19th century, when a systematic study of the UK postal service and railway network was carried by Charles Babbage, Operations Research as a discipline can be considered as a lesser known offspring of World War II. The unprecedented scale of the conflict and the availability of new and more complex technologies put the governments for the first time in the position of forming entire research teams dedicated to sustain the war effort by... *optimizing it.*

The work of those scientists heavily relied on modelling and statistical analysis, and tackled practical problems such as logistics and training schedules. They discussed about the safest convoy size to cross the Atlantic Ocean with minimal losses from enemy submarines, they found the most effective depth at which to detonate depth charges, they optimized the patterns of bombing raids in order to minimize losses. After the war this new science shared the faith of many wartime technological byproducts, and research partially shifted its focus to possible civil applications.

Fast-forward 70 years: nowadays Operations Research is commonly used to deal with - among others - complex problems of supply-chain management, freight transportation and automation. Optimization algorithms heavily influence our daily lives, the items we rely on and even our society as a whole.

2.1.1 Getting More Formal

It now seems appropriate to introduce some definitions, starting from formalizing the problem itself. Note that in the remainder of this work by "optimization" we in fact mean *single-objective optimization*, not considering those problems that sport more than one function to minimize at the same time. The definitions in this subsection are taken from [9] and [7].

Definition 1. *An **optimization problem** can be seen as a couple (S, f), with S being the set of feasible solutions (also known as* search space*) and $f : S \to \mathbb{R}$ the objective function*[2] *to optimize*[3]*. This function assigns to each $s \in S$ a real number indicating its quality, allowing to define a total order relation over the solutions in the search space.*

It is then possible to describe the desired kind of solution.

Definition 2. *A solution s^* is a **global optimum** if its objective function value is minimum among the solutions in the search space, that is, $\forall s \in S$, $f(s^*) \leq f(s)$. The set $S^* \subseteq S$ contains all the globally optimal solutions.*

While a good optimization method should ideally reach one of the solutions in S^*, there is a more general category of solutions that is of fundamental importance. But it is first needed to define the concept of *neighborhood.*

Definition 3. *A **neighborhood structure** is a function $N : S \to 2^S$ that assigns to every $s \in S$ a set of neighbours $N(s) \subseteq S$. $N(s)$ is called the neighborhood of s.*

It is now possible to introduce the concept of *locally minimal* solutions.

Definition 4. *A solution s^* is a **local minimum** with respect to a neighbour structure N if $\forall s \in N(s^*)$, $f(s^*) \leq f(s)$. s^* is called a strictly local minimal solution if $\forall s \in N(s^*)$, $f(s^*) < f(s)$.*

While sharing these common concepts, among others, optimization problems have different characteristics among them. A variety of models has consequently been proposed to deal with them.

[2]also known as cost, utility or fitness function.

[3]maximizing a function f is the same as minimizing a function $-f$; in this project optimization and minimization will often be used interchangeably, without a loss of generality.

2.1.2 Combinatorial Optimization

The focus of this work is on a specific subset of optimization problems, the so called *combinatorial optimization* ones. They are defined as follows.

Definition 5. *A* ***combinatorial optimization problem*** $P = (S, f)$ *can be defined by the following elements:*

- *a set of variables* $X = \{x_1, \dots, x_n\}$*;*
- *variable domains* $D_1, \dots, D_n$*;*
- *constraints among variables;*
- *the objective function* f *to be minimized, where* $f : D_1 \times \dots \times D_n \to \mathbb{R}$*;*

The set of all possible feasible assignments is then

$$S = \{s = \{(x_1, v_1), \dots, (x_n, v_n)\} \mid v_i \in D_i, s \textit{ satisfies all the constraints } \}$$

The scope is then restricted to problems with discrete decision variables and a finite search space, while objective function and constraints are quite more flexible to define. These characteristics are actually suitable for modelling a variety of complex real-world problems, as hinted in the next sections.

2.2 Hard Problems

One further way of categorizing optimization problems is by how difficult they are to solve. This depends on several factors and abstracts from the instance size, which is in fact taken as parameter in the complexity analysis.
It may seem that the restrictions introduced while defining the class of problems above could have simplified things quite a bit, but the reality is that combinatorial optimization problems can be extremely hard to tackle, and even relatively small instance sizes can lead to huge search spaces.

Assuming the reader to be somewhat familiar with the basic concepts of NP-completeness (whose extensive analysis is in turn out of the scope of this thesis), it is interesting to note that any optimization problem can be reduced to a homologous *decision problem*. An optimization problem is defined **NP-hard** if its associated decision problem is NP-complete. NP-hard problems are intractable, which means that there are no algorithms able to solve them in polynomial time (unless P = NP). This has brought scientists to research a wide range of techniques to deal with them, as will be later discussed.

The first problem to be proved as NP-complete, by Stephen Cook in 1971 [11], was the Boolean satisfiability problem. The following year Richard Karp published a list of 21 known hard problems that could be reduced to the SAT problem and hence proved NP-complete [12].

2.3 Optimization Methods

Depending on several factors, such as *a*) complexity of the problem; *b*) complexity and size of the problem instance; *c*) time availability; and *d*) actual need for best-in-class solutions, two kinds of approaches are adopted in optimization: *exact* and *approximate* methods.

2.3.1 Exact Methods

Exact methods are guaranteed to provide an optimal solution to the problem in analysis, if given enough time for it - which, incidentally, is not always a feasible option.

The most trivial algorithm to minimize a problem with a finite set of solutions is to evaluate them one by one and then return the best found, an approach that goes under the name of *exhaustive* or *brute-force* search. Unfortunately it is not difficult to think about real-life combinatorial problems with a solution space so big that if the fastest existing computer spent the estimated age of the universe crunching solutions, it would still be a long way from evaluating them all. To avoid being confined just to small instances of simple problems, improved exact methods need to reduce the amount of evaluations performed while guaranteeing not to miss a global optimum on the way. Two main approaches have been employed.

In **dynamic programming** the problem is recursively divided into simpler subproblems. This is consistent with Bellman's principle [13], that states that any subpolicy to an optimal policy should be optimal as well. Once the method detects that a certain set of assignments cannot lead to optimality, these are pruned from any further analysis.

Branch and bound algorithms and **A*** algorithms work instead by dividing the solutions into classes that are then assigned bounds on the quality of the objective function. Those areas of search space whose best possible quality is inferior to the best reached solution can be safely pruned.

Despite the continuous refinements in the field of exact algorithms, there will always be optimization problems that due to stringent time constraints or pure intractability won't be suitable to this kind of approach. In fact, in most practical applications all is needed is a *good enough* solution.

2.3.2 Approximate Methods

Approximate optimization methods exist to complement exact algorithms. An optimal solution is always preferable to a best-effort one, but as mentioned earlier such solution is not always at reach.

A first distinction that needs to be made on this new class of methods is about *guaranteed performances*. The family of **Approximation Algorithms** differs from other approximate solution methods because its members guarantee that the generated solution will be within a maximum bound[4] from the global optimum and reached in polynomial time. These sound like nice properties, but unfortunately there are two big issues: the *specificity* to a target optimization problem and the fact that the bound is often not strong enough to be of much practical use.

The other category of algorithms is the one of **heuristics**. In computer science, heuristics are sort of "good practice" rules, often derived from common sense or experience, whose reason of existence is basically to improve over random choice in the majority of cases. Heuristics often end up sacrificing accuracy in exchange of computing time and conceptual simplicity. Yet they are tweaked on the solution of a problem or even an instance of it, leaving intact the need of an optimization method for intractable problems that abstracts a bit more from the contingent situation. **Metaheuristics** attempt to fill this gap, and they are discussed in the next section.

2.4 Metaheuristics

Legend has it that Archimedes of Syracuse - one of the greatest minds of its time to say the least - overjoyed at his discovery of a method to determine the density of a supposedly golden crown, shouted out a loud *Eureka!*[5].

The most famous exclamation in Science, here reported in its English transliteration, is a form of the ancient Greek verb *heuriskō*, which means "to find, to discover". The word *heuristic* shares the same root. In 1986 Fred

[4]often referred to as ϵ, from which comes the definition of ϵ-approximate algorithms.

[5]Literally: "I found it!"

Glover first used the word *metaheuristic* (where *meta* is a Greek suffix for "beyond") to describe Tabu Search [14].

The term "metaheuristic" these days defines those techniques in the field of *Stochastic Optimization* that are not problem specific[6] and introduce randomness in the search to improve exploration. They often employ subordinates search heuristics (such as local search, later described), enhancing them by allowing sub-optimal choices that permit escape from local minima. Metaheuristics don't provide any theoretical bound on the quality of the solution found and don't guarantee results within a specified time-frame, but work on a *best effort* basis.

Glover's definition was pretty successful, but the scientific research in the field is older than the name itself.

2.4.1 Fundamental Classification

Before getting into a historical overview of metaheuristics, it seems appropriate to introduce two fundamental distinctions between methods.[7]

A first discrimination to be made is between **population-based** and **single-solution** techniques. As the name suggests, the difference is between the amount of solutions the two approaches handle at a time during the search. Single-solution methods are also known as *trajectory methods* because their search process describes a path in the search space. Population methods, on the contrary, evolve a set of points in the search space. Their behaviour is summed up in the figure below, taken from [7].

Another fundamental distinction is between techniques with **memory usage** versus **memory-less** ones. While the latter base the choice on where to direct the search next just on the solutions stored at the time, methods that employ memory try to extract dynamically (or *adaptively*) additional information about the search space to support further decisions.

Now that some order has been made, the following sections will deal with the metaheuristics themselves. For tidier exposition, they have been grouped according to the first categorization presented above.

[6]their implementation, of course, can be

[7]note that many further distinctions can actually be made

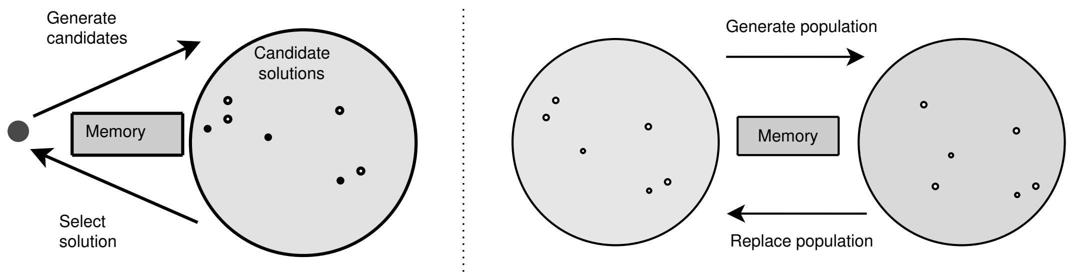


Figure 1: The scheme on the left sketches the behavior of a trajectory method: at each step, given a current solution, a set of candidates is generated and possibly one chosen to replace the existing solution. The scheme on the right displays instead the behavior of population methods; here there are multiple current solutions, and multiple generated solutions can be put in their place at the end of each step. In both approaches the use of memory is optional.

2.4.2 Trajectory Methods

Multi-start Local Search (MLS), in its basic form, is probably the simplest metaheuristic. Local Search is a subordinate heuristic that iteratively scans the neighborhood of the current solution looking for an improvement. In a *first-improvement* local search the hunt stops as soon as a better solution is encountered; a *best-improvement* method, conversely, explores the whole neighbourhood and sticks with the best found solution. Once a step is completed, the search looks for an improvement in the neighborhood of the new solution and so on. This technique has the limit of stopping once a local minimum is reached, which translates into reaching a solution whose neighbourhood doesn't contain any improvement. One simple enhancement consists in restarting the search from a random solution once this minimum has been reached, and repeat until a termination condition is reached; the resulting method, MLS, can be already classified as a metaheuristic due to its ability of exploring different areas of the search space.

A more sophisticated approach is to apply a *perturbation* to the local minimum and restart the search from there. This is the strategy used by **Iterated Local Search**. More precisely, ILS also chooses if to perturb the last found local minimum or the previously found one according to a specified *acceptance criterion*. Determining the entity of the perturbation is the trickiest part of such an algorithm: if the perturbation is too small, the algorithm would not be able to escape the local minima it got trapped in; if it is too big, the algorithm would turn in the less effective multi-start local search described above. To deal with this issue, techniques such as *Adaptive-ILS* [15] have been proposed.

Simulated Annealing has an illustrious past, being derived from the

Metropolis Algorithm [16]. But despite being considered as the oldest metaheuristic method, its first applications to the solution of combinatorial optimization problems date back to the eighties [4]. As the name suggests, the metaheuristic takes inspiration from annealing, a process of cooling molten metal. It shouldn't be surprising then that *temperature* is the parameter of choice. In practice, what happens is that the method tries to escape from a local minimum by sometimes accepting solutions worse than their predecessors. This suboptimal choice happens with a probability that is proportional to the temperature and inversely proportional to how much worse the tweaked solution is in comparison to the original one.

The approach adopted by **Tabu Search** [3] is to use a short-term memory to escape local minima and avoid cycles. Basically, the algorithm keeps track of the recently visited solutions and temporarily blacklists them to the search procedure. This forces the search to explore other areas of the solution space, effectively making the search neighbourhood dynamic. In practice, several additional tweaks are applied to make the method work; the blacklist is often implemented as a list of features to be avoided, and good solutions satisfying specified *aspiration criteria* are considered as acceptable even if they would normally be discarded.

The Greedy Randomized Adaptive Search Procedure (**GRASP**) [17] is an example of synergy between heuristic techniques and problem-independent ones. Assuming each solution is formed by a set of components, GRASP adopts a *constructive* strategy to build the next solution by iteratively choosing the components according to a heuristic criterion that scores them proportionally to the benefit they are supposed to bring to the partial solution. The second phase of the algorithm is a local search process, which refines the composed solution and enhances its fitness.

2.4.3 Population-based Methods

Genetic Algorithms are the archetypal population-based method and were introduced by John Holland in 1975 [2]. This kind of algorithms keeps a pool of good solutions (*individuals*) and try to improve their quality by mimicking the Darwinian evolution theory. The *parent* solutions are matched and mated, breeding new *children* that, if good enough (according to a *selection rule*) will then be part of the population instead of their ancestors. In traditional GAs, consistently with the biological paradigm, solutions are combined using a *crossover* operator, and additional variability is brought in the population by a *mutation* operator. Countless variations of this basic recipe exist.

Another category of population-based algorithms[8] are the **Estimation of Distribution Algorithms** (EDA) [18, 19]. These methods, whose theoretical foundation is in probability theory, build a probabilistic model around good solutions and use that as a guide in exploring the search space. The model is constantly updated to include information about newly generated solutions.

Ant Colony Systems (ACS), first introduced by Marco Dorigo in [20], attempt to solve combinatorial optimization problems by imitating the behaviour of worker ants hunting for food. The small insects initially explore the surroundings of the colony in a random fashion; when a food source is found, they go back leaving behind a trail of pheromones. These pheromones act as a guidance for the next ants, and evaporate over time. The most basic forms of ACS work in a pretty similar way: virtual ants are travelling around the search space, composing solutions by combining heuristic evaluations with values left behind in virtual pheromone trails, to which an evaporation factor is applied at each iteration. The trails that survive are ideally the ones that lead to the best solutions.

2.4.4 Hybrid Approaches

While metaheuristics usually abstract from the problem in analysis, the **No Free Lunch Theorem** for optimization [21] has shown that in fact there isn't an optimization method that performs consistently better on all problems and instances.[9] A further implication of the theorem is that the performance of a metaheuristic on a specific problem is - as a general rule - proportional to the amount of domain-specific knowledge it incorporates.

This supports the use of custom-tailored heuristics inside metaheuristics, as some methods used to do well before the theorem was published. It also supports the idea of *hybridization* between optimization techniques: in hybrid approaches different methods are combined in the attempt of using each other's strengths to support each other's weaknesses.

Several ways of combining algorithms have been proposed, such as *cooperative search* - where more methods are run in parallel with some extent of

[8]while some single-solution EDA methods have been attempted, these algorithms are population-based in the vast majority of the cases

[9]actually in [22] the same authors show that there are in fact "free lunches" in a co-evolutionary setting; several limitations to the theorem have arisen, too, but the underlying idea is still standing

information exchange - or integrating metaheuristics with exact methods like Branch and Bound. What is more interesting to the scope of this thesis is the hybridization between different metaheuristics, in particular between trajectory search methods and population ones.

This is happening, for example, in **Memetic Algorithms** [23]. Loosely taking inspiration from the concept of *meme* first introduced by Richard Dawkins [24], the idea behind these methods is to combine a population-wide evolution with an individual refinement of the single solutions, that improves their quality before they share with their peers. In practice, this is obtained by applying local search to the individuals in the pool of a genetic algorithm.

Many other approaches have been attempted, yet the aim here is not to provide with an extensive list of them. The next section will instead introduce a framework that attempts to add a more systematic perspective on these hybrid techniques.

2.4.5 Intensification and Diversification

A crucial problem in the design of metaheuristics sits in achieving the balance between two contrasting needs: the need to visit random areas of the search space in order to sample them, and the need to use the information already obtained from the search space to bias the search. These two opposed tendencies go under the names, respectively, of **diversification** and **intensification**. Two other terms are often used to refer to the same concept, albeit with a slightly different meaning according to [9]: *exploration* and *exploitation*. While acknowledging this subtle distinction, as a matter of fact in this thesis they will be used interchangeably.

As Glover and Laguna remark in [3], during an intensification phase the search focuses on exploring the neighborhood of promising solutions, while during a diversification phase the search is encouraged to explore unvisited regions of the search space, generating solutions that are ideally different in significant ways from the ones encountered before.

In fact, talking about *phases* can be misleading: the two behaviors often coexist in the same design elements of a method. As a very simple example, consider a first-improvement local search that analyzes the possible modifications to the original solution in random order; the search in itself is purely guided by the resulting fitness, but a dimension of exploration is introduced by randomizing which candidates are considered first.

A way of visualizing the role of metaheuristic components on intensification and diversification has been proposed by Blum and Roli in [9] with the proposed **I&D frame**, reported in Figure 2.

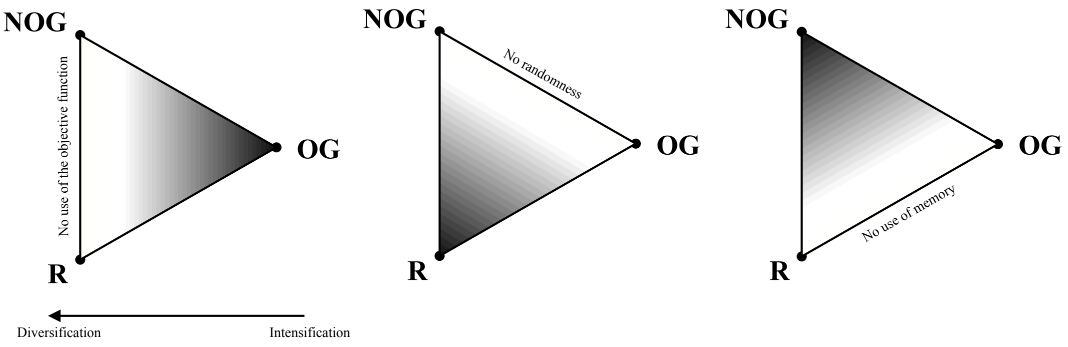


Figure 2: The I&D frame introduced by Blum and Roli.

In this formalization, the three corners of the triangle represent, at the extremes, the three roles a metaheuristic component can assume. The **OG** corner corresponds to those elements guided only by the objective function - think about checking the neighborhood of the current solution in a best-improvement local search. Here exploitation is at work in its purest form. The **NOG** corner refers to those components guided by other functions than the objective one, like for example some of the corollary information gathered by an EDA algorithm during the optimization process. The remaining corner, labeled **R**, comprises those components that are totally random; randomness is an important factor in many metaheuristics, albeit has do be dosed sparingly to avoid losing focus - an example of it is a random restart in a MLS.

Interleaving these three tendencies over time in a synergistic way is the key ingredient to any successful metaheuristic.

After this brief overview on hard combinatorial optimization problems and the techniques used to deal with them, it is time to focus the discussion and present the problem of choice for this thesis: the Quadratic Assignment Problem.

2.5 The Quadratic Assignment Problem

The **Quadratic Assignment Problem** (QAP) is a combinatorial optimization problem introduced by Koopmans and Beckmann in 1957 as a formal model for allocating indivisible economical activities [25]. Informally, there is a given number of facilities to assign to the same number of locations in an optimal way; a mutual distance is given between locations, as is the flow, a number which quantifies the mutual interaction between facilities. The optimality is reached by placing the facilities in the locations so to minimize the summation of the products of distance and flow between all the facilities.

The next section will provide a more formal definition of the QAP.

2.5.1 Mathematical Formulation

Several statements of the problem have been proposed in literature; the following one, proposed by Çela in [26], is maybe the most commonly used:

$$cost(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)}$$

where

- n is the size of the problem instance;
- π represents a possible permutation over (1,2,...,n) and $\pi(i)$ corresponds to the index of the location to which facility i is assigned; as later detailed, π is an ideal way of representing a solution to a QAP problem;
- f is the *flow matrix*, and f_{ij} is the directed flow between facility i and facility j;
- d is the *distance matrix*, and d_{ij} is the directed distance between location i and location j.

The aim is to minimize the cost function defined by the formula above; QAP has been proved NP-hard in [27].

QAP instances can be categorized according to various criteria.

Symmetric instances are those where, given any couple of facilities (or locations) i and j, the flow (distance) between i and j is the same as the one from j to i.

Flow dominance and *Distance dominance* are two further attributes that can be ascribed to QAP instances: an instance with a high flow dominance will have a consistent part of the overall flow exchanged among relatively few facilities; a homologous definition applies to distance dominance.

A fundamental resource for Quadratic Assignment Problems is QAPLIB [28, 29], an online repository of instances, solutions and resolution approaches.

2.5.2 Motivation and Applications

As of the time of writing, instances of the Quadratic Assignment Problem of size greater than 30 or so are considered to be non-solvable in an exact way. The concise formulation of the problem coupled with its intrinsic complexity made it a popular test bench for metaheuristics. But the problem is far from being a purely theoretical one; as seen, it arose from practical needs, and there are many scenarios in which the facility/location model can be applied.

One possible situation that can be effectively modelled by a QAP is the placement of electronic components on a board. The problem was first stated, in the form of a specific instance, by Paul Steinberg in 1961 [30]. In his formulation, locations were the slots where to install the components, and the components themselves were the facilities. The two matrices would then contain the distances among the various slots and the amount of wires that connect each couple of components (as flow). The instance proposed by Steinberg had cardinality thirty-six[10] and took until 2001 [31] to be solved exactly by a specifically tailored branch and bound algorithm - despite having been solved pseudo-optimally since the early nineties.

Another interesting application, in this case an asymmetric one, is related to the design of keyboard layouts. Keys can be seen as locations, and the corresponding letters as facilities. The flow between two letters is the empirical frequency with which they are used one after the other, given a language. The distance between keys i and j is given by the time it takes to press j after typing i. Note that the real-life problem has been modelled more in detail in [32], but the QAP modelling is still interesting because the original paper [33] provided several QAP instances that became rather popular in literature.

2.5.3 Solution and Neighbourhood Representation

Despite the fact that metaheuristics can be fairly problem-independent, whenever applying such methods to a new problem, this needs to be modelled in a convenient way. This translates into providing a representation for solutions

[10]there were actually 34 components to place in 36 slots, so two dummy zero-wired components had to be added in the modelling phase

themselves, and a definition of the neighbourhood structure defined earlier in the chapter.

In general, a good solution representation for an optimization problem should share the following properties [7]:

Completeness the representation should be able to represent all the solutions associated to a problem;

Connexity a search path must exist between any couple of solutions in the search space;

Efficiency the representation should be as easy as possible to manipulate by the search operators.

When defining the QAP formally, the permutation π of numbers in the interval (1,2,...,n) has been deemed as an ideal solution representation to an instance of size n. As mentioned, given a permutation π, $\pi(i)$ corresponds to the index of the location to which facility i is assigned.

But does such encoding fulfil the three criteria listed above?
It is relatively straightforward to see that, given n locations and n facilities, it is possible to represent any n disjoint matchings among them using a permutation of the numbers in the interval (1,2,...,n). The set of all possible permutations is then a complete representation of the solution space.

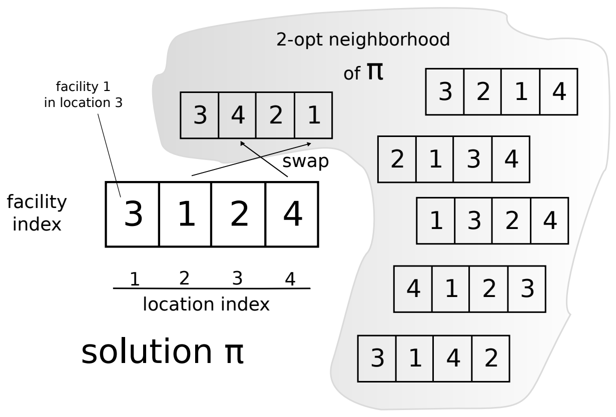


Figure 3: Here π is a solution to a hypothetic QAP instance of size 4, encoded in the discussed permutation representation; the grey area encloses the entire 2-opt neighborhood of the solution.

To show that connexity holds it is needed to proof that a search operator can be able to travel from an arbitrary point in search space to any other. It is problematic to make such a statement while abstracting from the search operator itself; the same issue applies to efficiency. To overcome this limit, the concept of *2-opt* search neighbourhood is introduced. Given a solution S for the QAP, its 2-opt neighbourhood is the set of solutions obtained by simply swapping the locations of two facilities; an example of 2-opt neighborhood can be seen in Figure 3.

It is now possible to define a simple search operator, *2-opt local search*, which is a basic local search that explores the neighbourhood defined above. With regards to 2-opt local search, connexity holds because it is indeed theoretically possible to reach a solution from any other by performing a proper sequence of swaps. Efficiency does too, since any possible swap transforms a given solution into another valid one, and takes constant time to be performed.

2.5.4 Approaching the QAP

The interesting theoretical properties and the extensive range of applications contributed to make the QAP a popular test-bench for many optimization methods.

The best known-solutions listed in QAPLIB have been reached using a variety of metaheuristics, both single-solution and population-based. For the first group algorithms like GRASP [17], Simulated Annealing [34] and several incarnations of Tabu Search [35–37] are used. Among the successful population based approaches are Ant Systems [38] and various forms of Hybrid Genetic Algorithms [39, 40].

The main inspiration for this work came from those methods that try to explicitly deal with scoring or marking some subsets of the solutions; this is a technique common to Ant Colony Systems, Estimation of Distribution Algorithms and Tabu Search, among others. The next Chapter will start by describing the process of designing, choosing and assembling the metaheuristic components of BIMA-QAP, and concludes with an overview of the resulting algorithm.

3 Contribution

The aim of this project is to design, code and evaluate a new metaheuristic for the Quadratic Assignment Problem, the **Bandit-Inspired Memetic Algorithm** (BIMA-QAP). The idea is to combine several concepts found in literature and create a method that is able to efficiently travel across the search space hunting for good solutions; some inspiring works are cited along the chapter. Before getting into details, though, it seems appropriate to introduce a few fundamental concepts.

The method discussed in this chapter shares a fundamental assumption with most (if not all) metaheuristics. The **Proximate Optimality Principle** (POP for short), introduced by Glover and Laguna in [3], stipulates in short that good solutions have a similar structure. This suggests that sharing good features between solutions could generate new, better solutions. Coherently with this assumption, BIMA-QAP will be tested on QAP instances for which the POP has been experimentally verified to hold in [41].

The previous paragraph mentioned that good solutions share good features: the smallest possible of such features is called a solution *component* - not to be confused with the components of a metaheuristic discussed in the last chapter. In the context of this thesis, a solution component is one of the n^2 possible atomic choices that assigns facility i to location j, so that $\pi(i) = j$. The definition comes from the fact that each solution π is composed by n of these elements.

Note that the set of components forming a solution must respect the problem constraints; it is not possible, for example, to assign a single facility to multiple locations. Given that in any valid QAP solution no facilities or locations are left unassigned, every attempt of modifying a solution results in the change of at least two components; this is behind the concept of swap used in 2-opt local search. In the remainder, *enforcing* a component on a solution means applying the associated assignment in place of an existing one, and consequently modifying one further component in the solution in order to keep it coherent.

In the rest of the chapter the design choices for BIMA-QAP are presented and motivated. Section 3.1 starts by describing the structure of the algorithm; section 3.2 underlines a component-focused approach common to many metaheuristics. Section 3.3 takes a step back introducing the Multi-Armed Bandit problems as an example of the exploration/exploitation dilemma, and UCB

[6], a technique used to tackle it. In section 3.4 the formula is altered in several, significant ways; this formula will concretize in an operator as seen in section 3.6. Before getting there, the pool management policies are discussed in section 3.5. The last section offers eventually an overview of the algorithm structure and parameters.

3.1 Hybrid Structure

The first big design choice for a metaheuristic is the one between a trajectory based and a population based approach. As seen in chapter 2, on the other hand, hybrid methods such as memetic algorithms have shown to successfully combine some of the better traits of both worlds. A hybrid method needs two kinds of operators:

1. A local search operator, used to perform neighborhood search and refine solutions;
2. Some form of perturbation/crossover operator, necessary to avoid getting stuck in local minima and ideally share the best traits across neighborhoods.

Note that a crossover operator implies the use of a pool of solutions rather than focusing on a single promising candidate at once.

The structure of BIMA-QAP has much in common with what was just described; the algorithm has a pool of solutions, which are in fact local optima with respect to a local search neighborhood. And, while not using "conventional" genetic operators, solutions in BIMA-QAP share that condensed reworked information that is behind the idea of a *meme* in memetic algorithms.

The local search operator of choice is the 2-opt local search already mentioned in chapter 2. The recombination operator will be the main topic of this chapter, with its components gradually introduced and itself fully defined in section 3.6.

Before getting into further details, it is time to describe two definitions connected to the algorithm structure itself. A *macro-iteration* is a step in the loop which involves randomly selecting a solution in the pool, perturbing it and further enhancing it by local search. The newly obtained solution is either inserted in the pool in place of the initial one or discarded, as described in section 3.5.

Once a macro-iteration is over, a new one is started as long as the stopping condition has not been fulfilled. Stopping conditions are discussed in subsection 3.7.1, and usually involve a reached iterations threshold. In contrast with macro-iterations, the *iterations* counter is updated every time a solution is evaluated, which mostly happens during local search.

3.2 Selective Components Memory

In the previous chapter a distinction has also been made between techniques that memorize information on the search space and techniques that don't. While a pool of solutions is in itself a way of encoding good features found during the optimization, the focus here is on *explicitly* storing data about such features.

An approach common to Tabu Search [3], ACS algorithms [42] and univariate EDA algorithms [19, 41] is to associate to single components flags or values that are then used to guide the search. Ant Colonies and several EDA algorithms, in particular, approach the QAP by keeping a $n \times n$ matrix encoding the desirability of each component $\chi_{i,j}$ in time-step t.

$$X^t = \begin{pmatrix} \chi^t_{1,1} & \chi^t_{1,2} & \cdots & \chi^t_{1,n} \\ \chi^t_{2,1} & \chi^t_{2,2} & \cdots & \chi^t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \chi^t_{n,1} & \chi^t_{n,2} & \cdots & \chi^t_{n,n} \end{pmatrix}$$

This kind of explicit memory is appealing because it can add an additional dimension to the search by efficiently re-using selected data already gathered during the algorithm execution. In fact, BIMA-QAP uses two kinds of such matrices to store information that is used to modify existing solutions, as described in section 3.4.

Note that using only the quality of single components to guide the search can result in overlooking the *linkage* relationships between components. This issue is further commented in subsection 3.4.3.

3.3 A Matter of Balance

Once decided that the components-based approach looks promising, there is the need of defining which information to store about components and how to use it in this context. The technique of choice will have an important role in dealing with the delicate task of balancing exploration and exploitation, the key issue in the design of any metaheuristic, as earlier discussed.

Achieving a good compromise between the two extremes is not a problem found only in metaheuristics; it is in fact common in learning [43], planning [44] and dealing with imperfect knowledge in general. One of the archetypal examples of such dilemma is the **Multi-Armed Bandit** (MAB) model, known since the beginning of the century and first formalized in its current form by Robbins in 1951 [5].

The MAB models the uncertainty faced by a gambler who has to choose how to spend his coins among K one-armed bandits, each behaving according to an independent and initially unknown probability distribution, in order to maximize his final profit. An alternative way of describing the gambler's goal is to talk about regret minimization, where the regret is the difference between the profit obtained by an ideal sequence of plays (always pulling the best arm, if the reward distributions are stationary) and the performed plays.

The problem has been daunting scientists since it was introduced during the Second World War. It was considered so difficult by Allied research personnel that it was suggested to drop leaflets about it over Germany, so the enemy scientists could also waste some time on it. Over the next decades, however, several solution approaches started to surface. The focus here is on optimistic index policies, and on the **Upper Convergence Bound** (UCB for short, [6]) in particular.

In 1979 Gittins [45] has shown that it is possible to maximize the expected discounted reward in the MAB with a policy that first assigns to each arm a score based on the current state, then selects the highest scoring one. Such an approach goes under the name of *index policy.* The idea behind UCB is similar; it involves computing for each arm an upper bound for the returned reward, basing the bound estimation on both the rewards history and the accuracy of the reward estimation (the number of pulls performed on the arm). Whenever the gambler has to choose which arm to pull, he finds out the value of UCB for each arm according to formula 1 and selects the highest scoring one.

$$\bar{x}_j^t + \sqrt{\frac{2 \ln \sum_k p_k^t}{p_j^t}} \tag{1}$$

The first term in the formula, x_j^t, encodes the expected average reward for arm j according to the knowledge available in time-step t. Always choosing the arm with the highest expected reward would result in a purely exploitative algorithm, so the formula includes a second term to deal with exploration. The variable p_j^t represents the number of times arm j has been pulled at time-step t, making the value of the second term in formula 1 inversely proportional to the arm popularity.

Despite not having the best proven bound on regret, a technique like UCB manages to be simple and efficient at the same time. Note that in some applications the numerical constant in the second term is turned into a variable, as formula 2 shows. This allows to tweak the contribution of the exploration term to the score, and the modified formula takes the name of **cUCB**.

$$\bar{x}_j^t + \sqrt{\frac{c \ln \sum_k p_k^t}{p_j^t}} \tag{2}$$

By now, the parallel this section is trying to suggest should be clear to the reader: getting back to the QAP, what if the components to enforce on a solution were to be treated as arms of a bandit machine?

It is *crucial* to note that many of the properties characterizing the MAB are lost in the combinatorial optimization scenario. More than one component needs to be enforced at once, otherwise the chances of getting out of the basin of attraction of the previous minimum would be pretty slim. Furthermore, the quality measure of a component would be dependent on the rest of the components in the solution as well. This implies that components are not independent as arms are assumed to be, and that also non “pulled” components are influencing the feedback. One more consequence is that the feedback associated to a component can change according to the neighborhood(s) being analyzed.

Nonetheless, the loss of theoretical bounds and properties doesn’t take away from the fact that it would be interesting to see how an algorithm like cUCB performs in balancing exploration and exploitation in this new scenario.

3.4 Tweaking the Formula

The index formula in equation 2 seems like a nice start, but requires some extensive modifications to be of any use in BIMA-QAP. The basic idea behind the formula, as noted, is to combine a score encoding the asserted quality of a component with a score inversely proportional to how often it was used - so that also what seem to be lower-quality components may have a chance. This structure is summarized in Fomula 3, below.

$$\textit{component value} \leftarrow \textit{exploitation term} + \textit{exploration term} \tag{3}$$

The two coming subsections try to contextualize the two terms of the sum to the approximate solution of a QAP instance.

3.4.1 First Term: Scaling the Fitness

As seen in the previous section, the first term in equation 2 encodes the expected value of an arm. Stressing the point that the aim is to define a good policy for balancing exploration and exploitation more than to strictly adhere to the MAB model, the most natural candidate for representing the value of a component seems to be the fitness of the solutions in which it appeared. This approach is coherent with the idea that good solutions share similar traits implied by the POP.

The idea is then to keep track of a fitness value for each component, and to store them in the fitness matrix $\tilde{F}^t$. The value $\tilde{f}_{i,j}$ in the matrix represents the aggregate fitness associated to component $\chi^t_{i,j}$.

$$\tilde{F}^t = \begin{pmatrix} \tilde{f}^t_{1,1} & \tilde{f}^t_{1,2} & \cdots & \tilde{f}^t_{1,n} \\ \tilde{f}^t_{2,1} & \tilde{f}^t_{2,2} & \cdots & \tilde{f}^t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{f}^t_{n,1} & \tilde{f}^t_{n,2} & \cdots & \tilde{f}^t_{n,n} \end{pmatrix} \tag{4}$$

At every update, $\tilde{f}_{i,j}$ is computed by linearly combining the average fitness of the solutions containing that component ($\bar{f}_{ij}$, with weight w_1) and the fitness of the best solution with $\chi^t_{i,j}$ in it ($\check{f}_{ij}$, with weight $1 - w_1$), as in Equation 5. These values are updated every time a solution containing the respective components is evaluated.

$$\tilde{f}^t_{ij} = w_1 \bar{f}_{ij} + (1 - w_1)\check{f}_{ij} \tag{5}$$

As noted, the great majority of the evaluations take place during local search. Updating the matrix values during local search is considered sort of compulsory to get a decent estimation of the fitness. Staying in the MAB metaphor, this choice influences the pulls as well: this will be discussed in the next subsection.

A final note is to be made on the exquisitely practical differences between the concept of reward in UCB and the idea of fitness in a minimization problem. In Equation 2 the first term is assumed to be in $[0, 1]$ and to higher values of it is associated a better performance.

The straightforward solution is to first scale the fitness in the interval [0,1] proportionally to the value of the other components involved in the comparison, then consider the complement to 1 of the resulting number to model the concept that in this case to higher fitness corresponds lower solution quality.

The resulting first term, f^t_{ij} is presented in Equation 6,

$$\textit{exploitation term} \leftarrow 1 - \frac{\tilde{f}^t_{ij} - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl})}{\max\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl}) - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl})} \tag{6}$$

where A^t is the restricted set of components described in subsection 3.4.3.

3.4.2 Second Term: Managing the Pulls

The second term doesn't share the same scaling issues: the main question here is when to update the counts associated to each component. Intuitively, this should go together with the update of the fitness estimations - which, once again, is mostly done during local search. Considering that the formula being put together is not going to be involved with local search, this in the MAB paradigm means that in the vast majority of cases the arms are not pulled because of their higher index score. In fact, most of the pulls are performed implicitly while traversing 2-opt neighborhoods.

The pull counts $p^t_{i,j}$ are stored in matrix P^t and updated whenever the associated component is involved in a solution being evaluated. This seems the most coherent choice, since the pulls are meant to represent the accuracy of the estimation on components fitness.

$$P^t = \begin{pmatrix} p^t_{1,1} & p^t_{1,2} & \cdots & p^t_{1,n} \\ p^t_{2,1} & p^t_{2,2} & \cdots & p^t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p^t_{n,1} & p^t_{n,2} & \cdots & p^t_{n,n} \end{pmatrix} \tag{7}$$

An interesting consequence of this is that when the formula is eventually used to select a set of components to enforce on a specified solution, the components not involved in local search will be more likely to be selected due to their relatively higher exploration term. These components are also more likely to be outside the basin of attraction of the 2-optimized solution, which is a very desirable property for an operator that has to complement local search. The resulting exploration term can be seen below.

$$\textit{exploration term} \leftarrow \sqrt{\frac{c \ln \sum_{(k,l)\in A^t} p^t_{kl}}{p^t_{ij}}} \tag{8}$$

In Equation 8 the summation is carried on over the members of set A^t, which is the topic of the next subsection.

3.4.3 A Dynamic Set of Arms

Now that the two terms have been defined, the immediate idea would then be to use the formula to select a set of components from the n^2 available to enforce on a solution in the pool. This seems like a promising approach, but it relies on a Naive Bayesian assumption of independence among the components: it assumes the value of a component to be independent from the remaining $n-1$ which are going to form the solution.

This is clearly not the case, since the fitness of a solution is depending on the values of all the components it contains. The POP suggests that good solutions share features, and in fact the extent of these features is not limited to the single component. To put it in different words, if a *linkage* between the components is assumed then BIMA-QAP would need a way of storing and transmitting good patterns of components.

Storing is not doable by simple book-keeping, since maintaining statistics about all the possible subgroups of components is clearly unfeasible. Pool-based algorithms try to get around this problem by simply storing a set of good solutions, which are in turn expected to contain good patterns. This is also the approach adopted by BIMA-QAP.

Transmitting good patterns between solutions, on the other hand, needs some further thinking. The problem is that the chance of sharing a good set of components from a solution to another in the pool by just selecting them randomly over the n^2 available is pretty slim.

The intuitive workaround is then to restrict the set of components on which the index values are computed. If this restricted set contains only the components found in another solution in the pool, the algorithm will behave similarly to a guided crossover. This should greatly increase the chance of sharing good patterns between solutions.

Another restricted set of components worth some attention is the one containing all the components found in the solutions in the pool. This sort of reminds of the pool-centered matrix updates in PBIL [19], but doesn't take into account the components multiplicity (it is still a *set*, after all).

Finally, the whole set of components is still considered with a certain probability; selecting a component over all the available ones, while not strictly random, is more similar to the concept of mutation common in GA algorithms.

The restricted set of components used by the algorithm at macro-iteration t is called A^t. The three options proposed earlier are summarized in the table below.

1	$A^t \leftarrow$ the components in a donor solution[11]	p_{donor}
2	$A^t \leftarrow$ the components in all the solutions in the pool	p_{pool}
3	$A^t \leftarrow$ the set of all components	p_{all}

At each timestep the restricted set is selected proportionally to the specified probabilities p_{donor}, p_{pool} and p_{all}, intuitively summing to 1.

3.5 The Solutions Pool

Before finalizing the definition of the *perturb()* operator in the coming section, it is appropriate to introduce the solution pool and the concept of *slot*.

In BIMA-QAP, the solution pool is defined as

$$Pool(t) = \{\pi_1^t, \pi_2^t, \ldots, \pi_\varphi^t\}$$

where t is the macro-iterations counter. The pool contains φ solutions, which are initially randomly generated and immediately optimized by 2-opt local search. Each of the solutions in the pool occupies a slot s of the φ available; slots can hence be enumerated in the interval $[1, \varphi]$.

A fundamental property related to the pool of solutions is the *Survival Rule*, which states the conditions that bring a new solution to be inserted in the pool in place of a pre-existing one. In BIMA-QAP the Survival Rule is rather simple: when a new local optimum is reached, it is inserted in the pool if it is better than the one the *perturb()* operator was applied on - the *father*,

[11] randomly chosen from the pool members, as seen later

in GA wording.

This means that each slot in the pool over time hosts only descendants of the solution that was there initially. Each slot, in other words, traces a search path in the search space. This leads to a further take on the concept of components memory: what if a *fitness* and a *pulls* matrix were to be associated to each slot, so that whenever modifying a solution by *perturb()* the algorithm could take into account also some neighborhood specific information?

The matrices $\tilde{F}^t(s)$ and $P^t(s)$ related to a given slot s are its *local matrices*. Their values are updated every time the solution in the associated slot is perturbed to generate a new local minimum; given that the solution to modify is selected randomly among the ones in the pool, each slot has a probability of $\frac{1}{\varphi}$ of being selected. The local matrices associated to the slot of choice are updated during the macro-iteration in the same way as the global matrices; the other local matrices are left untouched. Of course, being updated much less often, their estimation will be less accurate; as mentioned in section 3.7.1 this makes the initialization of these matrices a tad more tricky than their global counterparts.

3.6 The *perturb()* Operator Defined

Now that all the main ingredients have been discussed, the *perturb()* operator can be discussed more in details.

Devised as a complement to local search, **perturb**$(A^t, s, \delta, c, w_1, w_2, w_3)$ is used once in each macro-iteration. A slot s is randomly chosen from the φ available, then δ components are selected to be enforced on the associated solution; these components are the ones for which the formula discussed in the previous sections takes the highest values.

The final structure for the calculation can be seen in Formula 9; the first and the second terms earlier discussed have an index g for global and l for local.

$$\begin{aligned} &w_2(exploitation_g) + (1 - w_2)(exploitation_l) \\ &+w_3(exploration_g) + (1 - w_3)(exploration_l) \end{aligned} \tag{9}$$

where

$$\begin{aligned}
exploitation_g &\leftarrow \left(1 - \frac{\tilde{f}^t_{ij} - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl})}{\max\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl}) - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl})}\right)\\
exploitation_l &\leftarrow \left(1 - \frac{\tilde{f}^t_{ij}(s) - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl}(s))}{\max\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl}(s)) - \min\limits_{(k,l)\in A^t}(\tilde{f}^t_{kl}(s))}\right)\\
exploration_g &\leftarrow \sqrt{\frac{c \ln \sum\limits_{(k,l)\in A^t} p^t_{kl}}{p^t_{ij}}}\\
exploration_l &\leftarrow \sqrt{\frac{c \ln \sum\limits_{(k,l)\in A^t} p^t_{kl}(s)}{p^t_{ij}(s)}}
\end{aligned}$$

The resulting formula is quite more complicated than the original; the effectiveness of the operator *perturb()* and its reactivity to various settings of the parameters are extensively evaluated in the next chapter.

Now that all the fundamental design elements of BIMA-QAP have been defined, it is time to give an overview of the method as a whole.

3.7 The Algorithm as a Whole

It is now time to formalize the structure of the algorithm and to discuss the nature of its parameters. Subsections 3.7.1 and 3.7.2 deal with such issues.

3.7.1 Structure and Methods

The BIMA-QAP structure is sketched in Algorithm 1. The pool is initialized with φ random solutions, which are then optimized by *2-opt* local search.

The method **2opt_local_search**(π_i^{rnd}) returns a local minimum obtained by performing a sequence of swaps, either the best- or the first- improving, on the original solution. The method stops when no improving swaps are found; this project will focus on first-improvement local search, because after a brief experimental assessment it proved to be the better performer. Note that, as discussed, the fitness and pulls matrices are updated every time a solution fitness is computed - a task that also increments the iterations counter. The local search takes up most of the algorithm iterations; hence, most of the

Algorithm 1 BIMA-QAP

```
$Pool(rnd) \leftarrow \{\pi_1^{rnd}, \pi_2^{rnd}, \ldots, \pi_\varphi^{rnd}\}$        ▷ random pool
for all $\pi_i^{rnd}$ in $Pool(rnd)$ do
    $\pi_i^0 \leftarrow$ 2opt_local_search($\pi_i^{rnd}$)        ▷ local search
end for
$Pool(0) \leftarrow \{\pi_1^0, \pi_2^0, \ldots, \pi_\varphi^0\}$        ▷ 2-optimized pool
$t \leftarrow 0$

repeat
    s $\leftarrow$ generate_random_number($1, \varphi$)
    $\sigma \leftarrow$ generate_random_number($0, 1$)

    switch $\sigma$ do
        case $0 \leq \sigma < p_{donor}$
            j $\leftarrow$ generate_random_number($1, \varphi$)        ▷ $j \neq i$
            $A^t \leftarrow$ components($\pi_j^t$)
        case $p_{donor} \leq \sigma \leq (p_{donor} + p_{pool})$
            $A^t \leftarrow$ components($Pool(t)$)
        case $(p_{donor} + p_{pool}) < \sigma \leq 1$
            $A^t \leftarrow$ components($all$)

    $SA^t \leftarrow$ perturb($A^t, s, \delta, c, w_1, w_2, w_3$)        ▷ selecting the components
    $\pi_i^{tmp} \leftarrow$ apply_components($SA^t, \pi_i^t$)
    $\pi_i^{new} \leftarrow$ 2opt_local_search($\pi_i^{tmp}$)

    if fitness($\pi_i^{new}$) $\leq$ fitness($\pi_i^t$) then        ▷ survival rule
        $Pool(t+1) \leftarrow Pool(t) \setminus \{\pi_i^t\} \cup \{\pi_i^{new}\}$
    else
        $Pool(t+1) \leftarrow Pool(t)$
    end if

    $t \leftarrow t + 1$

until stop_condition
```

updates to the matrices are done by the evaluations performed while this method is running.

At the end of the initial optimization, the great majority of the general matrices components will have been updated at least once. In case this didn't happen, the pulls $p^0_{i,j}$ are initialized to 1 and the corresponding $\tilde{f}^0_{i,j}$ to 0, in a form of optimistic initialization. Initializing all the values in the local matrices the same way would take a lot more time; the approach adopted is to link the values in the local matrices to the values in the main ones instead: once a component is used locally, its estimation will decouple from the main one for the rest of the algorithm execution.

Once the data structures have been initialized, the proper algorithm loop can start. Two random numbers, i and s, are generated by calling the method **generate_random_number(min,max)**, where the two parameters represent the lowest and highest number that can be extracted by the generator. The first generated number is an integer in [1,φ], that determines which solution to modify; the second is a decimal in the range (0,1) and is used to select the set of components A^t to draw from, as discussed in subsection 3.4.3.

The function **components**($\pi^t_j|pool|all$) is used to obtain A from the specified source. Setting a solution as parameter (case 1) will return the n components in π^t_j. If the solution pool is set as parameter, as in case 2, A^t will include all the components present in the solutions in the pool. In case 3 the dummy parameter all is passed to the function to signal that A^t should comprise all n^2 components.

When the set has been defined, the operator **perturb**($A^t, s, \delta, c, w_1, w_2, w_3$) generates the vector of components SA to be enforced on π^t_i, as described in the previous section. The method **apply_components**(SA^t, π^t_i) takes care of altering the initial solution by carrying on the specified sequence of swaps - enforcing the components in increasing order of score, to minimize the risk of displacing the best components by the end of the task.

The resulting solution π^{tmp}_i is then optimized by local search, until the minimum, π^{new}_i, is found. This is in turn compared with π^t_i, and inserted in the pool instead of the original solution if having a better fitness. Fitness is computed (once) and retrieved by the function **fitness**(π^t_i), which is extensively used during the local search phase.

Once the macro-iteration is over, BIMA-QAP continues iterating over the loop until the specified stopping condition is reached. This condition can be related to a solution quality threshold or to an iterations limit (or both); for example, the algorithm could just stop once the first known minimum is inserted in the pool. Since the analysis carried out in the next chapter is also about the *amount* of different minima found, the stopping condition of choice is an arbitrary and rather high iterations limit.

3.7.2 Parameters

All the parameters of the algorithm have been introduced in the previous sections; it is now time to briefly summarize their expected influence on BIMA-QAP's behavior.

Setting a proper solutions pool size, $\boldsymbol{\varphi}$, is presumably going to have a decisive effect on the effectiveness of the algorithm. A too small value would risk an early convergence, while an oversized pool would dilute too much the calls to *perturb()*. As previously noted, pool sizes for algorithms adopting a memetic approach are usually significantly smaller than the canonical settings for GA.

The amount of components enforced $\boldsymbol{\delta}$ is also expected to impact the performance of BIMA-QAP. If nothing else, a low setting could result in not being able to use the *perturb()* operator to get out of a basin of attraction; a too high setting would be highly disruptive with regards to the solutions currently in the pool.

Tweaking the probabilities of restricting the components allowed in A^t at each given macro-iteration should result in a noticeable change in the meta-heuristic behavior. Altering the values $\mathbf{p_{donor}}$, $\mathbf{p_{pool}}$ and $\mathbf{p_{all}}$ (which sum to 1, effectively resulting in two degrees of freedom) changes BIMA-QAP's operating mode from an algorithm performing multiple single mutations to one transferring entire sets of components among solutions.

The weight $\mathbf{w_1}$, as seen, is the one governing the computation of $\tilde{f}_{i,j}$ by balancing it between the average and the minimum fitness of the solutions the component is involved into. Estimating the behavior of this setting is more complicated than many others; the weight is actually there to allow an experimental evaluation of which balance between the two values works best.

The two other weights, $\mathbf{w_2}$ and $\mathbf{w_3}$, govern the use of global matrices over local ones, as seen in Section 3.5. In principle, the use of local matrices should enhance the awareness of a solution to its own neighborhood. In practice, without a mechanism in place to keep the neighborhoods of the various slots in the pool relatively disjoint, the influence of this setting on overall performances is probably not going to be huge.

Finally, **c** is the parameter taking care of balancing the extent to which the exploration term influences the computation of index policies - making it the setting with the most direct impact on the exploration/exploitation dilemma.

The actual effect of the discussed parameters will be object of rigorous experimental analysis in the next chapter.

4 Results

Now that the algorithm structure has been laid down and the parameters defined, an experimental study is necessary to analyze its performances. Section 4.1 provides few details about the implementation of BIMA-QAP, section 4.2 discusses the methods adopted in analyzing the metaheuristic performances. In section 4.3 a synthesis of the experimental results is proposed, with a brief overview over the behavior of the algorithm on each instance. Section 4.4 concludes the chapter with some general remarks over the behavior of the parameters.

4.1 Implementation

Several frameworks are available for quickly implementing various types of metaheuristics [46–48]. Despite this, for increased flexibility the code for BIMA-QAP has been mostly written from scratch. The language of choice is C++ for sheer performance reasons, the software was compiled with gcc 4.4.5 and run under linux Debian OS with amd64 architecture.

As predictable, the algorithm is processor- rather than memory-intensive; the experiments were executed on an Intel i7 3930K 6-cores cpu. The multi-core capability has been extensively taken advantage of in the experimental scenario, by allowing to execute multiple runs of an experiment at once. OpenMP [49] was the library of choice for implementing this naive parallelism.

As in any stochastic algorithm, selecting a proper pseudo-random number generator (PRNG) is crucial in BIMA-QAP. The PRNG of choice is the Mersenne Twister [50], due to its performance and very long period; in an attempt to guarantee a random enough initialization, the seed is obtained by seeding in turn the function *rand()* with the current system time.

As a final note, most of the post-processing and all of the graphical output found in this chapter is performed by R scripts, with the aid of the *multcomp* package for the statistical comparisons.

4.2 Experimental Approach

Testing the algorithm serves two purposes: understanding if it indeed works as expected, and finding the best set of parameters to tune it properly. While the first part is straightforward, the second is not: since there is some sort of interdependency between the parameters, finding a good set is in fact a meta-optimization problem in itself - a multi-parameter optimization problem, to be precise.

To avoid over-complications, the approach adopted is to analyze the impact of each parameter individually. The first step is to define a set of *standard* settings for the various parameters, chosen mostly according to intuition, experience and the review of previous approaches - for example, the setting of δ to 1/6 was inspired by the results in [51]. These standard settings are summarized in Table 1.

Parameter	*Value(s)*
φ	70
δ	1/6 n
$p_{sources}$	0.34, 0.33, 0.33
w_2	0.5
w_3	0.5
c	2
w_1	0.5

Table 1: Standard Experiment *e00* settings. Note that $p_{sources}$ has three values: this because, despite being discussed about as a single parameter, it in fact represents three probabilities (p_{donor}, p_{pool} and p_{all}) with the constraint of summing to one. This practically results in *two* parameters.

The parameters are then altered individually over a predefined set of values, mostly uniform across the instances. Each modification results into an experiment, for a total of 30 proper experiments. The results of these tests should provide a useful insight on the behavior of the single settings, without getting into analyzing their mutual relationships.

The altered settings for each experiment are displayed in Table 2. The last two experiments, $e31$ and $e32$, are respectively about a MLS with 2-opt first-improvement local search and about a version of BIMA-QAP where the components to enforce are chosen randomly from the A^t of choice instead of being selected according to formula 9. This allows to isolate the effect of the operator *perturb()* from the one of the memetic structure combined with the restricted subsets of components, and is often referred as RND in the remainder. MLS, being the simplest metaheuristic conceivable, is used as a baseline for performances - even if the operator is not effective, RND population based hybrid approach is expected to perform better.

Experiment	*Altered Parameter*	*Parameter Value*
e01	φ	10
e02	φ	20
e03	φ	40
e04	φ	100
e05	φ	150
e06	δ	1/12
e07	δ	1/8
e08	δ	1/4
e09	$p_{sources}$	1, 0, 0
e10	$p_{sources}$	0, 1, 0
e11	$p_{sources}$	0, 0, 1
e12	$p_{sources}$	0.5, 0.5, 0
e13	$p_{sources}$	0, 0.5, 0.5
e14	$p_{sources}$	0.5, 0, 0.5
e15	w_2	0
e16	w_2	0.25
e17	w_2	0.75
e18	w_2	1
e19	w_3	0
e20	w_3	0.25
e21	w_3	0.75
e22	w_3	1
e23	c	0
e24	c	1
e25	c	10
e26	c	100
e27	w_1	0
e28	w_1	0.25
e29	w_1	0.75
e30	w_1	1
e31	MLS	NA
e32	RND	NA

Table 2: The list of experiments with the varied parameter and the according value. Experiment *e31* measures the performances of a Multi-start Local Search on the given instance, while *e32* chooses the components randomly from the selected A^t instead of ranking them according to Formula 9.

By analyzing each parameter individually it is possible to operate a proper statistical analysis on the effect of each parameter values on the performance of BIMA-QAP. In order to achieve statistical relevance, each experiment is repeated for a total of 24 times. Then an ANOVA procedure is performed, confidence intervals are computed and the results direcly fed to a Tukey test, as suggested in [52]. More practical details are discussed in the next section, before presenting the results.

4.3 Experimental Results

In Chapter 2 QAPLIB was mentioned as a repository for QAP instances and results. To test BIMA-QAP, 8 instances of various size have been selected from QAPLIB: *bur26a*, *nug30*, *ste36a*, *tai60a*, *tai60b*, *tai80a*, *tai80b*, *sko100a*. The choice was not random: these are the instances for which the POP is verified in [41].

Being the instances of different sizes and complexity, the iterations threshold had to be intuitively modified accordingly - up to a limit, to keep the runtime manageable. The table below provides for each instance the fitness of the known minimum and the amount of iterations specified for the stopping condition.

Instance	*Known Min.*	*Iterations*	*Runs*
bur26a	5,426,670	10,000,000	24
nug30	6,124	40,000,000	24
ste36a	9,526	100,000,000	24
tai60a	7,205,962	300,000,000	24
tai60b	608,215,054	200,000,000	24
tai80a	13,499,184	300,000,000	24
tai80b	818,415,043	300,000,000	24
sko100a	152,002	300,000,000	24

Table 3: The instances with the fitness of the best solution known, the amount of iterations for each run and the number of runs for each experiment.

As will be reported, BIMA-QAP could not always stretch down to the known minimum in the specified amount of iterations. In the smaller instances, on the other hand, the algorithm converges quickly to a first minimum, proving then very effective in finding multiple solutions having fitness equal to the known minimum.

Two different metrics are then adopted to rank the results of the experiments. The comparisons between experiments in bur26a, nug30 and ste36a is made analyzing the average amount of different minima found by BIMA-QAP. The behavior of the experiments on the bigger instances is compared focusing on how close - on average - they get to the known minimum, within the allowed execution boundaries. Note that the first metric is more centered on the exploration capabilities of the algorithm, while the second could value a more exploitative approach. For example, an experiment that reaches the minimum a bit more slowly - but once there is better at keeping looking for others - would be ranked better than a faster but less explorative one. Ultimately, a fair balance of the two opposed tendencies is required to reach even a single minimum.

Before getting to the comparison, though, it seemed appropriate to provide an additional insight over the standard run, and of course the results of the experiments themselves. Note that this chapter hosts a very condensed view of both the results and the comparisons, for further reference please check Appendix A.

4.3.1 Results

The next two pages contain two plots of $e00$ for each instance, in Figures 4 and 5. The plots on the left display the average fitness of the φ solutions in pool over time; the plots on the right show the fitness of the local minima generated across the macro-iterations - the candidates for pool admission. While the average fitness in the pool of solutions can just improve with time due to the survival rule, the generated local minima don't have such a constraint and can get worse over time - although this is an undesirable behavior if persistent and the algorithm didn't converge to a minimum first.

The x axis of each plot starts when all the runs in $e00$ are done with the initial pool optimization; different starting pools take different time to optimize, so actually the macro-iterations start on average slightly before the plot does. Each of the thousand dense points in the plots represents the average of ten samplings per run in the given window, averaged over the 24 runs that constitute the experiment - no additional interpolation has been applied in either case, apart from a simple fitted curve in the plots on the left.
Just after the plots, a tabular overview of the average amount of minima found for each experiment (for bur26a, nug30, ste36a) and of the average percentage over the known minimum of the best solution found (for all the instances) is offered by Tables 4 and 5.

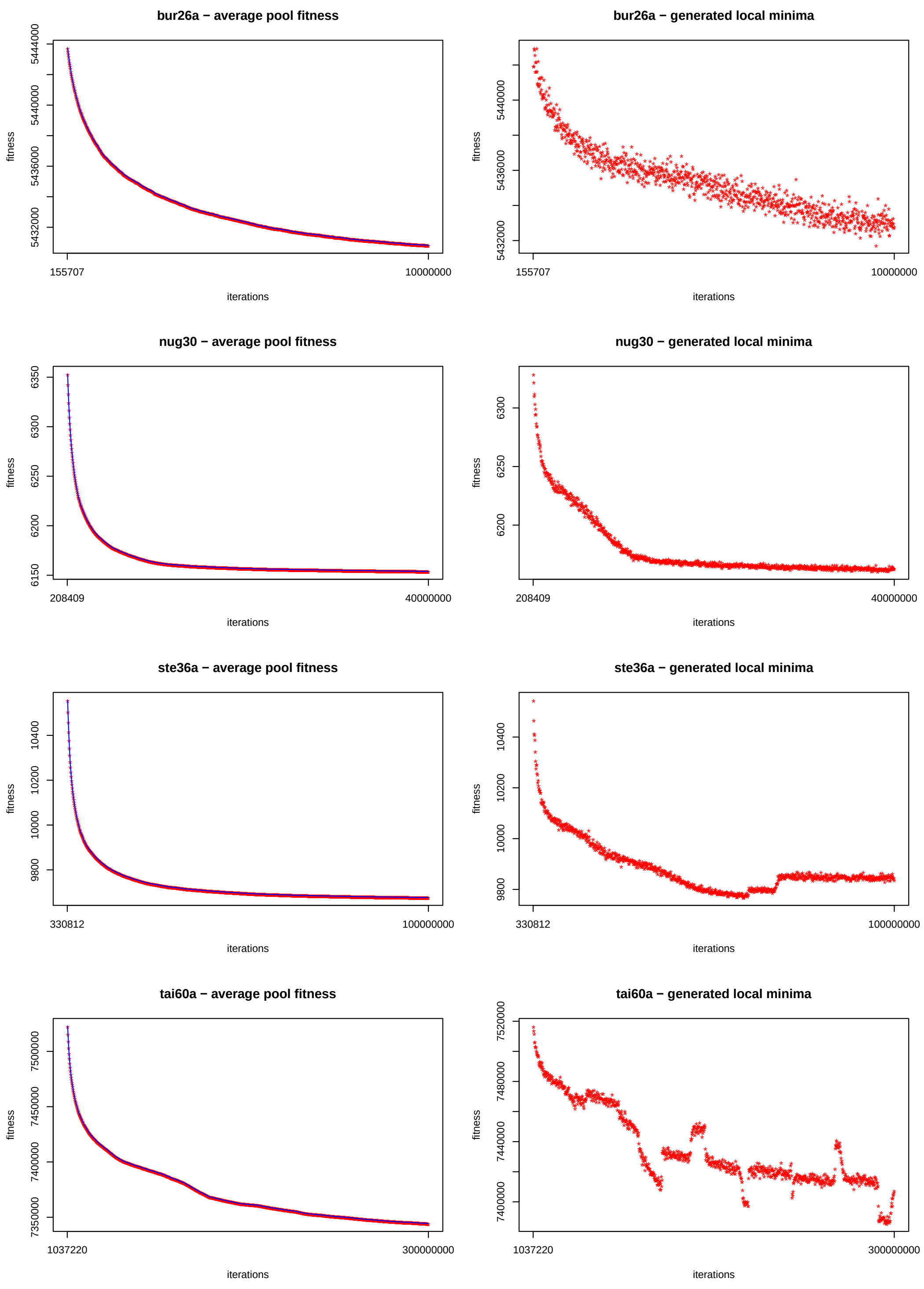


Figure 4: Average fitness in pool *(left)* and average fitness of the generated local minima *(right)* for bur26a, nug30, ste36a and tai60a.

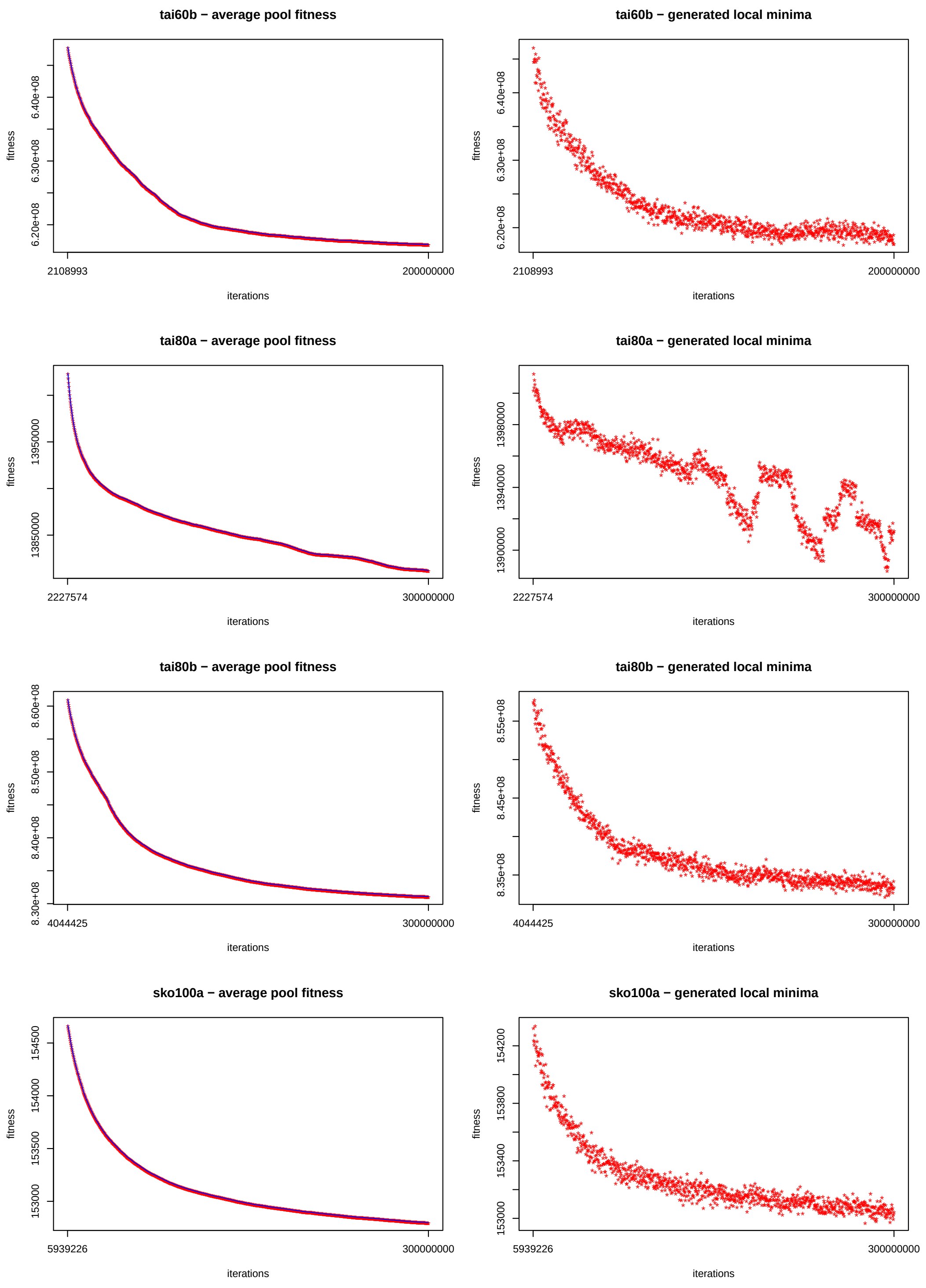


Figure 5: Average fitness in pool *(left)* and average fitness of the generated local minima *(right)* for tai60b, tai80a, tai80b and sko100a.

exp.	bur26a	nug30	ste36a	tai60a	tai60b	tai80a	tai80b	sko100a
e00	100.000	100.000	100.000	101.070	100.000	101.490	100.056	100.183
e01	100.000	100.035	100.061	101.157	100.016	101.370	100.232	100.113
e02	100.000	100.005	100.036	101.090	100.000	101.410	100.098	100.124
e03	100.000	100.000	100.004	101.108	100.000	101.429	100.032	100.129
e04	100.000	100.003	100.000	101.163	100.000	101.595	100.038	100.192
e05	100.000	100.000	100.000	101.225	100.000	101.697	100.060	100.205
e06	100.000	100.008	100.000	101.171	100.000	101.366	100.088	100.164
e07	100.000	100.000	100.000	101.108	100.001	101.393	100.055	100.162
e08	100.000	100.000	100.000	101.309	100.000	101.823	100.027	100.210
e09	100.000	100.000	100.000	101.062	100.000	101.796	100.015	100.152
e10	100.000	100.008	100.006	101.182	100.000	101.488	100.043	100.186
e11	100.000	100.003	100.000	101.257	100.004	101.460	100.143	100.195
e12	100.000	100.000	100.000	101.094	100.000	101.555	100.026	100.175
e13	100.000	100.011	100.000	101.202	100.000	101.470	100.068	100.183
e14	100.000	100.000	100.000	101.093	100.000	101.605	100.030	100.172
e15	100.000	100.000	100.000	101.155	100.000	101.527	100.054	100.180
e16	100.000	100.003	100.000	101.160	100.000	101.522	100.046	100.188
e17	100.000	100.003	100.000	101.222	100.000	101.572	100.019	100.162
e18	100.000	100.000	100.010	101.441	100.000	101.868	100.006	100.182
e19	100.000	100.000	100.000	101.138	100.000	101.533	100.049	100.179
e20	100.000	100.000	100.000	101.162	100.000	101.559	100.030	100.190
e21	100.000	100.003	100.000	101.164	100.000	101.486	100.032	100.165
e22	100.000	100.000	100.004	101.162	100.000	101.496	100.076	100.162
e23	100.000	100.003	100.004	101.237	100.000	101.485	100.063	100.155
e24	100.000	100.000	100.000	101.154	100.000	101.479	100.045	100.180
e25	100.000	100.000	100.000	101.152	100.000	101.603	100.036	100.188
e26	100.000	100.000	100.000	101.181	100.000	101.950	100.029	100.189
e27	100.000	100.000	100.000	101.139	100.000	101.483	100.072	100.148
e28	100.000	100.000	100.000	101.162	100.000	101.525	100.033	100.160
e29	100.000	100.000	100.000	101.135	100.000	101.544	100.036	100.198
e30	100.000	100.000	100.000	101.153	100.000	101.836	100.037	100.196
e31	100.000	100.068	100.574	102.318	100.110	102.468	100.375	100.534
e32	100.000	100.000	100.000	101.733	100.000	102.231	100.057	100.248

Table 4: This table displays the average value of the best found solution at the iterations threshold for each experiment. It looks like it is going to be difficult to infer something from tai60b average best found solution, since the value converged to 100.000 by the end of the experiments. The workaround will be to consider a different timestep in the comparison, as later remarked.

exp.	bur26a	nug30	ste36a
e00	92.08	2.96	6.29
e01	78.59	0.75	2.33
e02	88.67	1.75	2.92
e03	92.58	2.46	3.83
e04	88.79	3.08	6.62
e05	83.00	3.62	6.04
e06	80.21	1.92	4.75
e07	90.08	2.92	5.92
e08	93.92	3.37	5.50
e09	94.08	3.29	5.67
e10	90.33	1.58	3.29
e11	87.25	1.87	3.71
e12	92.75	3.29	5.67
e13	91.08	1.87	4.25
e14	92.37	3.00	5.92
e15	91.37	3.00	5.62
e16	91.46	2.96	5.58
e17	91.79	2.92	5.33
e18	85.79	3.04	3.00
e19	93.21	3.17	5.83
e20	91.04	3.04	5.83
e21	90.67	2.29	5.46
e22	86.29	2.25	6.00
e23	84.87	2.58	4.83
e24	92.67	3.08	5.37
e25	89.21	3.46	5.83
e26	89.21	3.75	6.54
e27	94.67	2.83	5.62
e28	92.54	3.17	6.67
e29	91.33	3.12	5.87
e30	89.92	3.25	5.71
e31	14.2	0.33	0.08
e32	86.2	3.87	3.79

Table 5: This table displays the average amount of different minima found at the iterations threshold for each experiment on bur26a, nug30 and ste36a.

Figures 4 and 5 clearly show that BIMA-QAP has an oscillating behavior on tai60a and tai80a. While not being able to provide an exhaustive explaination for it, the phenomenon is not uncommon nor inherently wrong. As will be noted in the next few pages, the two instances share the same origin.

4.3.2 Comparing the Experiments

The remainder of this subsection will present a brief overview of the behavior of the parameters on each considered QAP instance, which is accompanied by a graphical representation of the Tukey test on the experiments about a single parameter. The four columns are meant to display the changing behavior of the parameters over time: they are based on values sampled at 1/10, 3/10, 6/10 and 10/10 of the iterations threshold.

A comparison shows significant differences if the bar associated to it doesn't overlap with the dotted vertical line in the middle. While analyzing the amount of minima found, the first experiment in the comparison is better than the second if the bar is to the right of the vertical line. When the analysis switches to the best found solution, a smaller value is preferred and the situation is inverted.

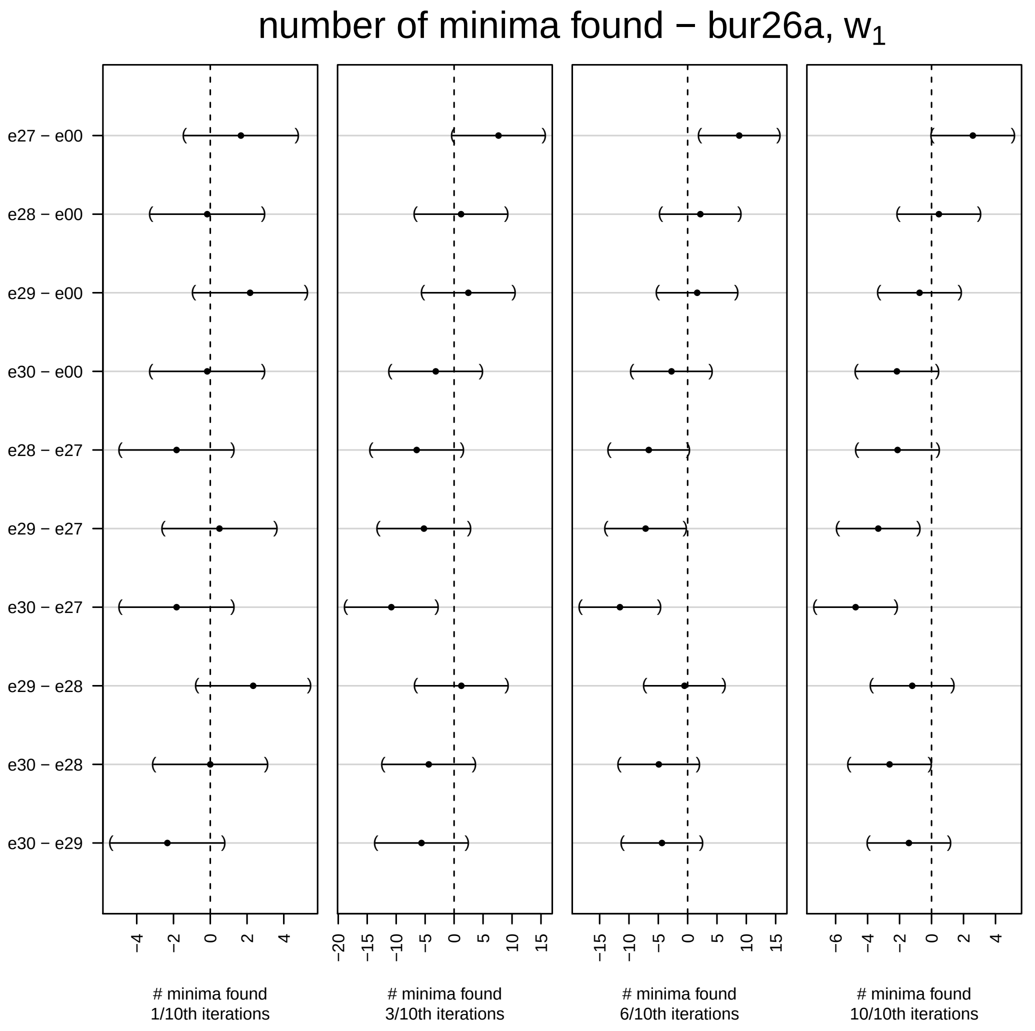


Figure 6: Bur26a [33] is an asymmetric instance having as flow matrix the frequency of typing two letters one after the other, and as distance matrix the average time required for such a task. Bur26a is solved very easily, but once the first minimum has been found the algorithm keeps crunching. The standard φ works just ok, bigger and smaller values perform worse; small δ is also noticeably reducing the amount of minima found. Setting p_{all}, w_2 or w_3 to 1 also leads to a similar subpar result. The standard value of c is optimal, while *as pictured, the smaller w_1 the more minima the algorithm finds.* In fact, using just the minimum fitness and discarding the average one seems to work best. MLS is performing much worse, reaching on average less than a sixth of the minima found by BIMA-QAP. Randomly choosing the components instead of using the components matrices leads to discovering less solutions than the standard settings of BIMA-QAP.

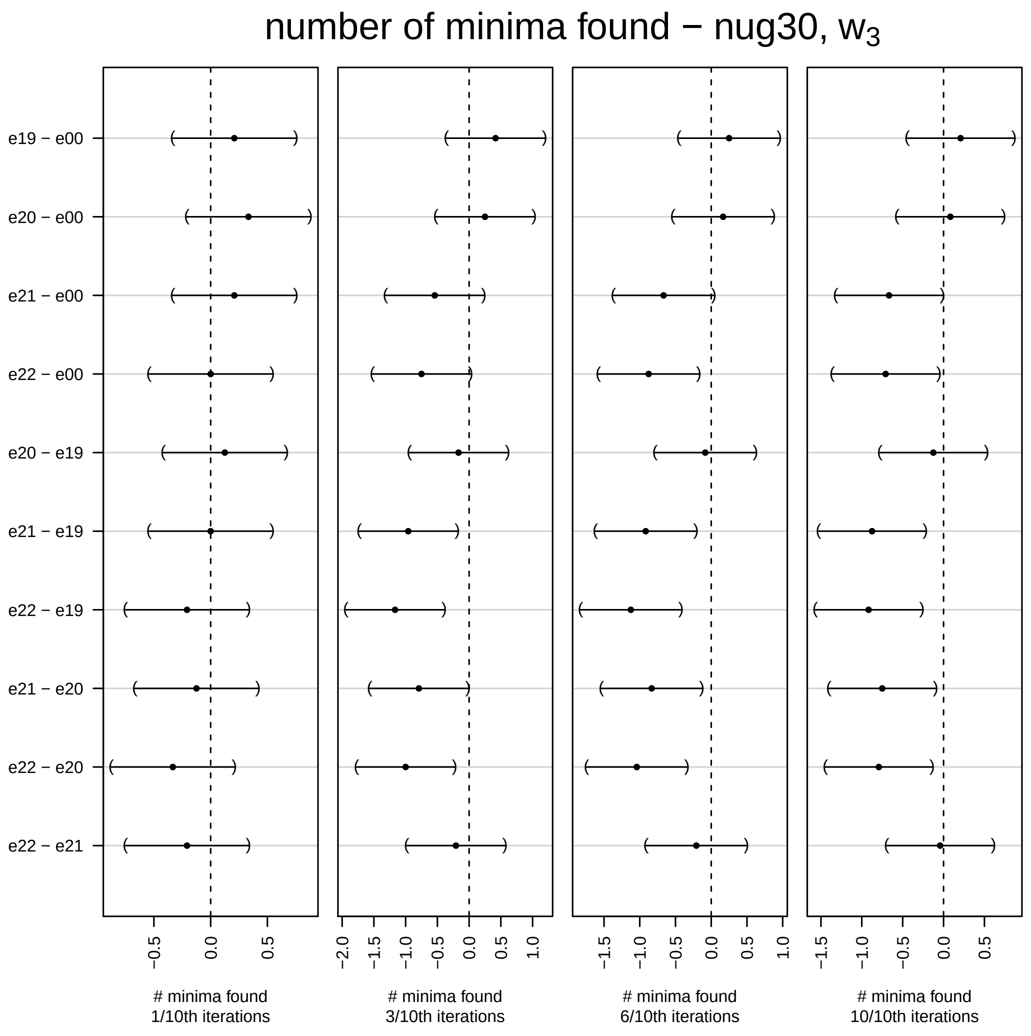


Figure 7: Nug30 [53] is a popular instance whose distance matrix contains Manhattan distances of regular grids. It is once again very straightforward for BIMA-QAP to reach a minimum, so the amount of minima is compared instead. Here to higher settings of φ and δ correspond straightforward and marked improvements in the results, while the opposite happens for values lower than the standard. It is interesting to note how every experiment where p_{donor} equals 0 performs much worse, while once again $e00$ holds the ground quite well. *As the figure above suggests, increasing the value of w_3 leads to worse results*, while nothing significant can be said about w_2 or w_1. To higher c correspond better performance, with the highest setting being almost on par with the random components selection. Nug30 is the only instance in which RND seems to perform slightly better (not always significantly so) than BIMA-QAP. The general trend here seems to suggest that a very high level of exploration is needed to reach all the minima. MLS results are half as good as those obtained by the worst BIMA-QAP settings.

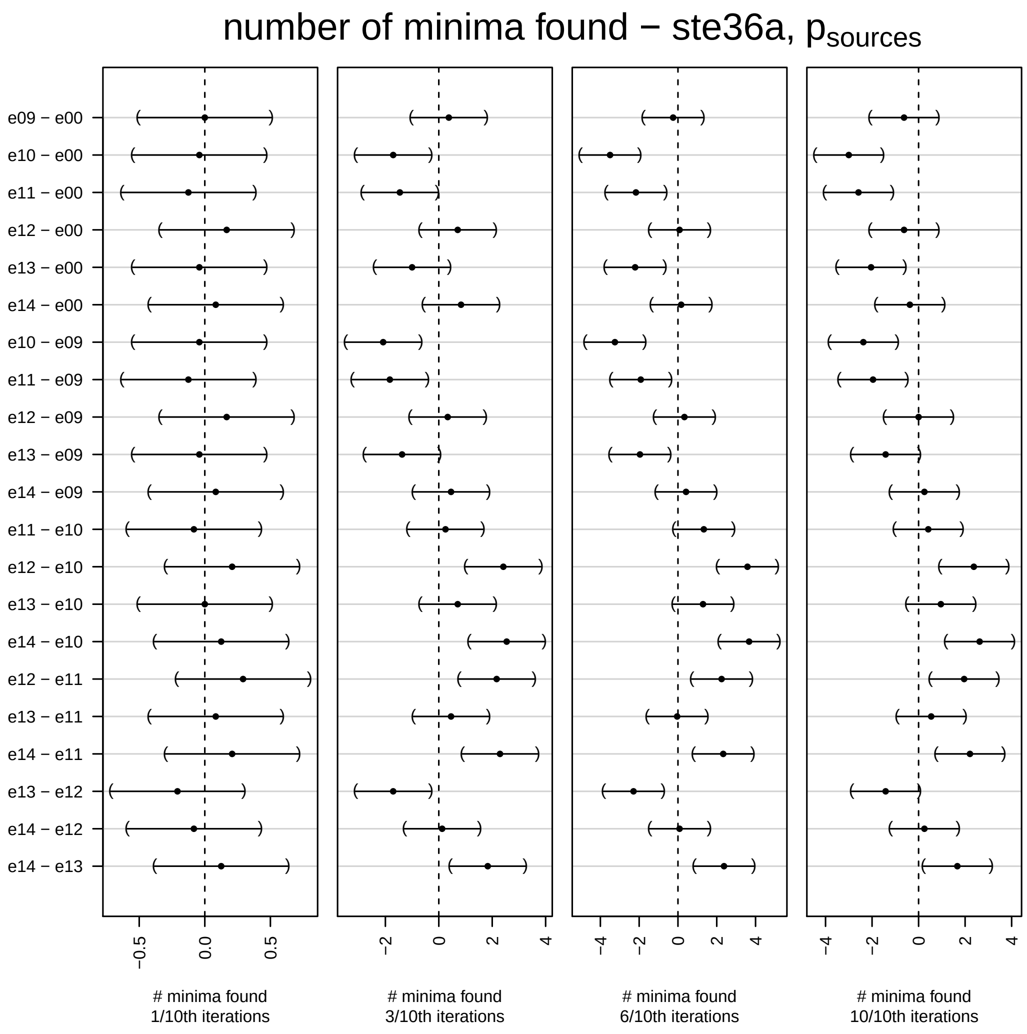


Figure 8: Ste36a [30] models an instance of the Steinberg Wiring Problem (mentioned in subsection 2.5.2) having Manhattan distances. This is the third and last of the instances where finding multiple minima is a common occurrence and the focus of the analysis. The standard settings perform very well; experiments with lower value of φ perform much worse. The smallest value of δ also lowers the performance a bit. *As in the previous case and shown in the figure, here also experiments where p_{donor} is set to 0 yield a clearly lower amount of minima.* While the various settings of w_3 don't cause any significant difference in performance, for some reason the $e18$ where w_2 is at 1 finds about half the minima compared to the other experiments on the parameter - this is a sharp difference with no straightforward explanation. Bigger c values yield once again higher results; while $e00$ and $e28$ perform better on w_1, no significance is achieved. On *ste36a* the difference between RND and $e00$ is quite marked in favor of the latter, but what really stands out is the horrible performance of MLS - which almost never manages to reach a minimum.

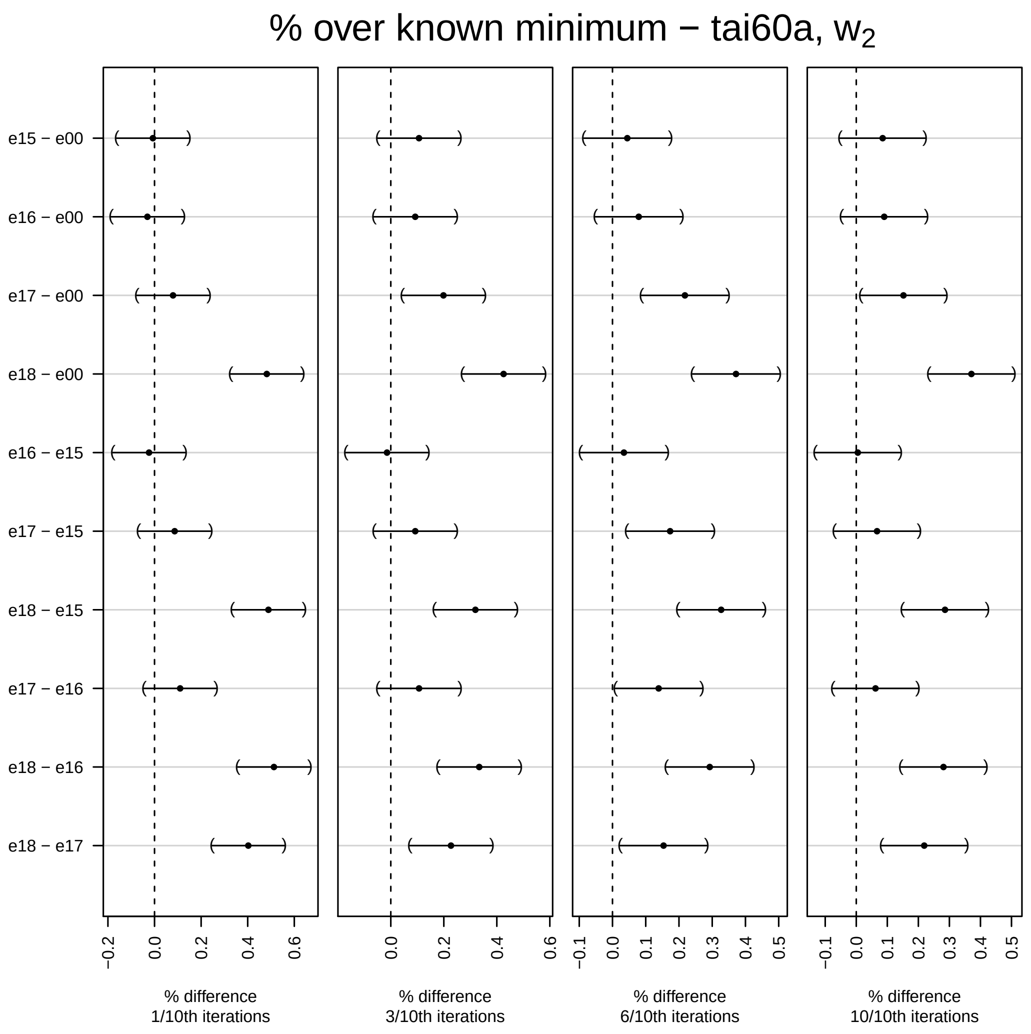


Figure 9: Quoting QAPLIB [28], *''the instances in Taixxa are uniformly [randomly] generated ... problems in Taixxb are asymmetric and randomly generated"*. Solving these bigger instances (with the exception of tai60b) is much more challenging to BIMA-QAP than the smaller ones, and the analysis switches to the average quality of the best solutions found across the runs. Quite astonishingly, in Tai60a [35] $e00$ performs better than any other experiment. Setting φ to 40 as in $e02$ works just as fine, but any other setting worsens the solution quality. Increasing the value of δ to 1/6 results in the worst performance obtained on this instance by BIMA-QAP. Despite the different evaluation metrics, here the results of experiments with $p_{donor} = 0$ are markedly not as good. *Also this time setting $w_2 = 1$ results in a very bad performance, as depicted above.* No significant differences can be found among the experiments on w_3, and all experiments on c and w_1 perform a bit worse than the standard settings. MLS clearly lags behind, and RND average best solution is worse than any experiment with BIMA-QAP.

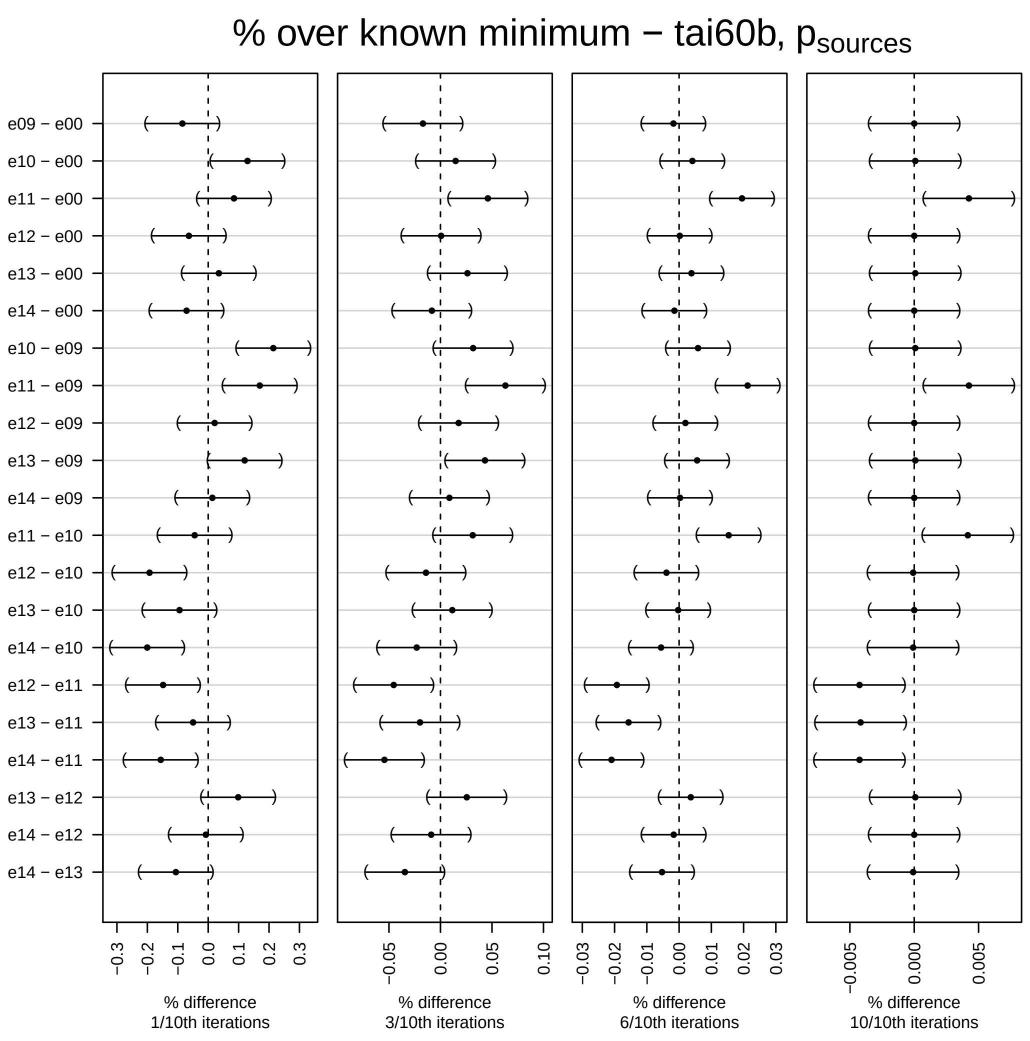


Figure 10: As noted, the number of iterations allotted to tai60b [36] proved to be pessimistic; in fact, the algorithm starts finding the first minima already at $\frac{1}{10}$ of the total run-length. By the time the 200M iterations mark is reached, most of the runs ended up in the minimum. This doesn't allow for any statistically relevant end-experiment comparisons, so the earlier time-steps are considered instead. Early on, a small φ helps the algorithm to lower the average best solution fitness. Altering δ doesn't modify the behavior of the algorithm. *On the other hand, the difference on $p_{sources}$ is quite sharp, and mimics what was seen on ste36a despite the different metrics: p_{donor} is crucial to the performance of the algorithm, and shouldn't be set to 0. Setting p_{all} to 1 is not a very good idea either.* Nothing significant comes out of w_3, w_1 and c, while the experiments on w_2 provide some minor variability. On the long run, the performance of RND converges to match BIMA-QAP; before then, the latter has an edge. MLS doesn't get to 100.000 within the iterations threshold.

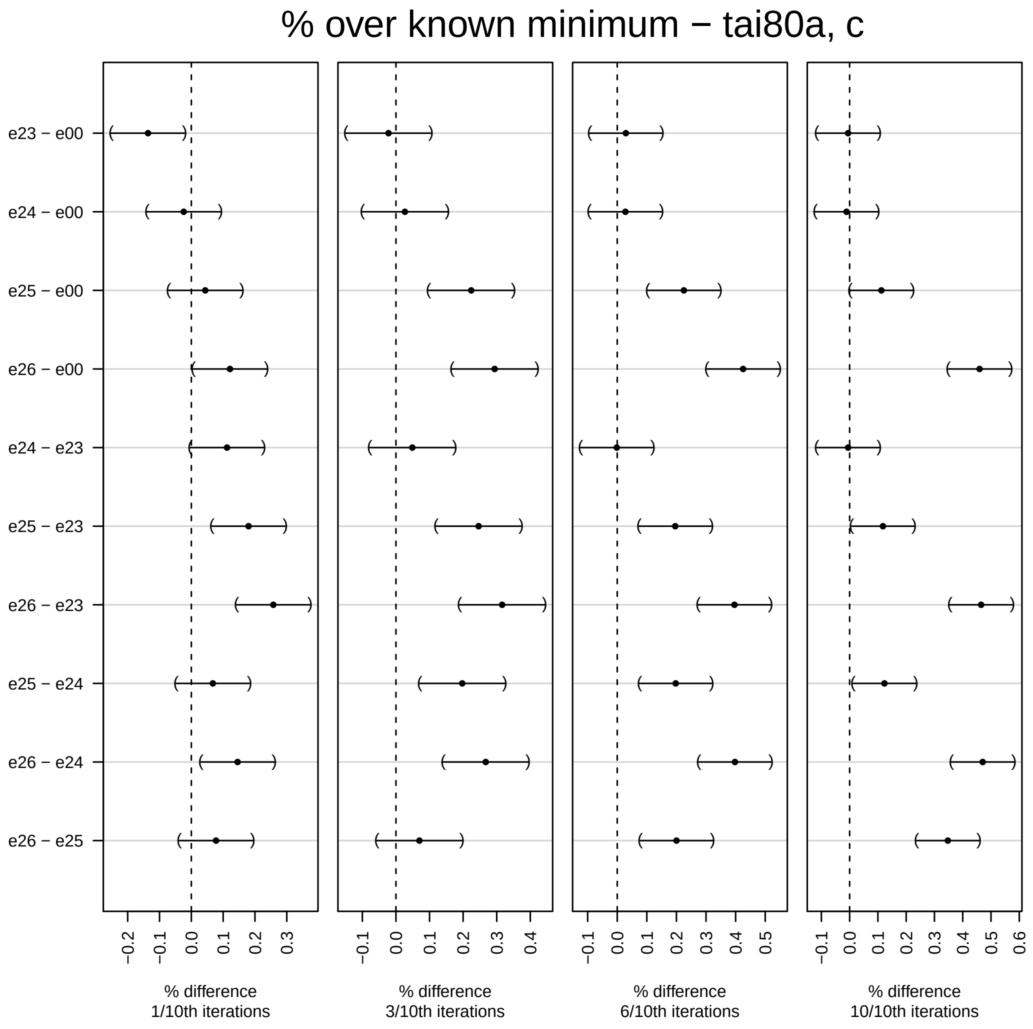


Figure 11: Tai80a [35] seems the most difficult instance among the considered ones: the average best solution here is well above the 101% of the known minimum. The standard settings prove here to be optimal for some parameters, improvable for others. In particular, smaller settings for φ and δ seem to improve the algorithm behavior; on the other hand, setting a pool size of 100 (or greater) or δ to 1/4 results into a noticeable reduction of the solution quality. Interestingly, on tai80a the parameter $p_{sources}$ works opposite than what was seen up to now: all the experiments where p_{donor} is *different* from 0 reach worse minima, apart from $e00$. Once again no significant differences come from altering the value of w_3, and the infamous setting of w_2 to 1 performs much worse. *A high value of c brings the search to lose focus, and for c = 100 BIMA-QAP reaches the worst result on this instance.* Finally, setting w_1 to 1 works very bad, suggesting that it is important to consider the minimum fitness also. MLS and RND perform rather similarly here, and noticeably worse than any setting of BIMA-QAP.

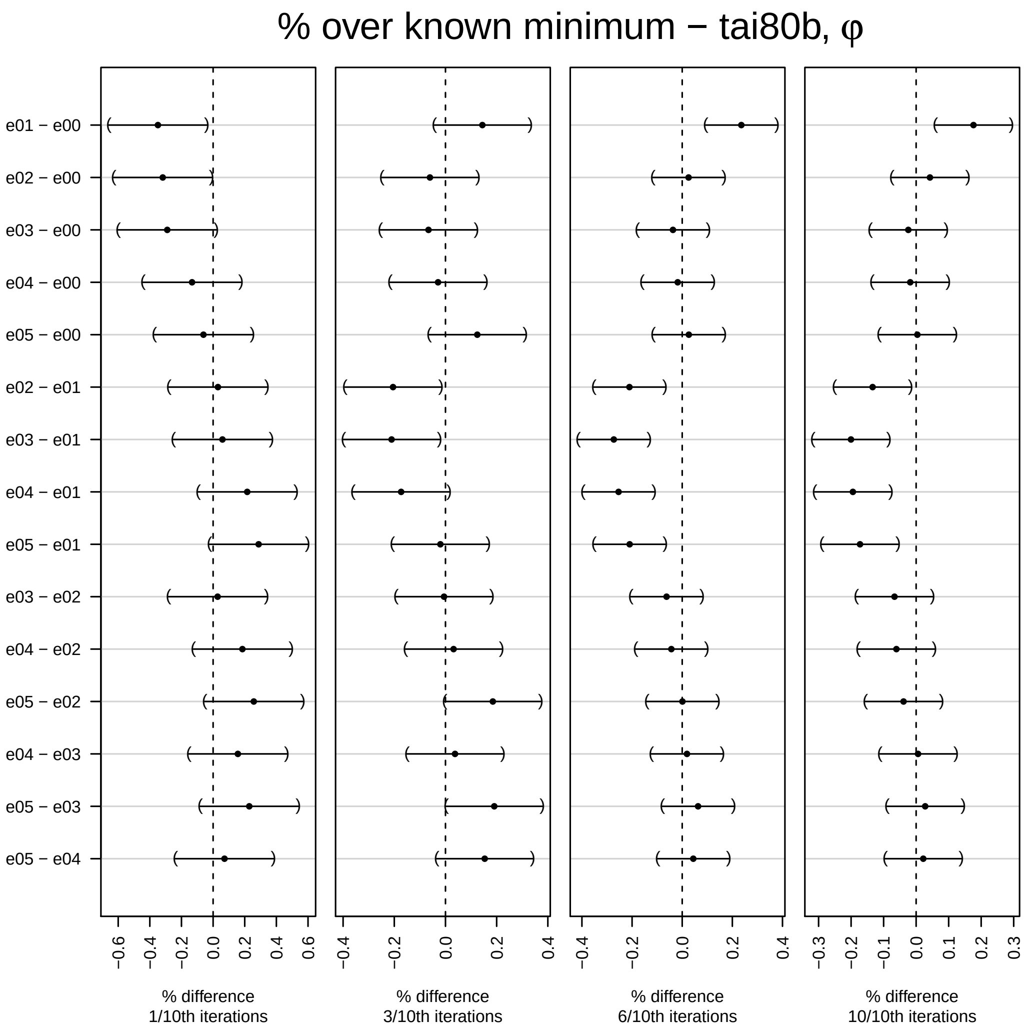


Figure 12: Tai80b [36], on the other hand, proves to be much easier to tackle; several runs in various experiments are able to reach the known minimum. *As in figure, reducing φ to 10 has dramatically negative effects on the performance of BIMA-QAP, making* $e01$ *the worst of the proper experiments.* No significant differences can be inferred from the results on δ. The various settings for $p_{sources}$ perform more or less the same, with the exception of setting $p_{all} = 1$ which results in the second worst result. While once again w_3 doesn't really influence too much the behavior of the algorithm, w_2, the best mean value for this instance is obtained in $e18$. This is quite interesting, since up to now setting $w_2 = 1$ had largely negative results on the performance of the algorithm. Despite the fact that a higher w_1 seems to perform a little bit better, no significant results are obtained for the experiments on c and w_1. As usual MLS lags behind, but in this case RND average best found solution is on par with $e00$ - note that the performance of BIMA-QAP can be improved by tweaking settings like the mentioned w_2.

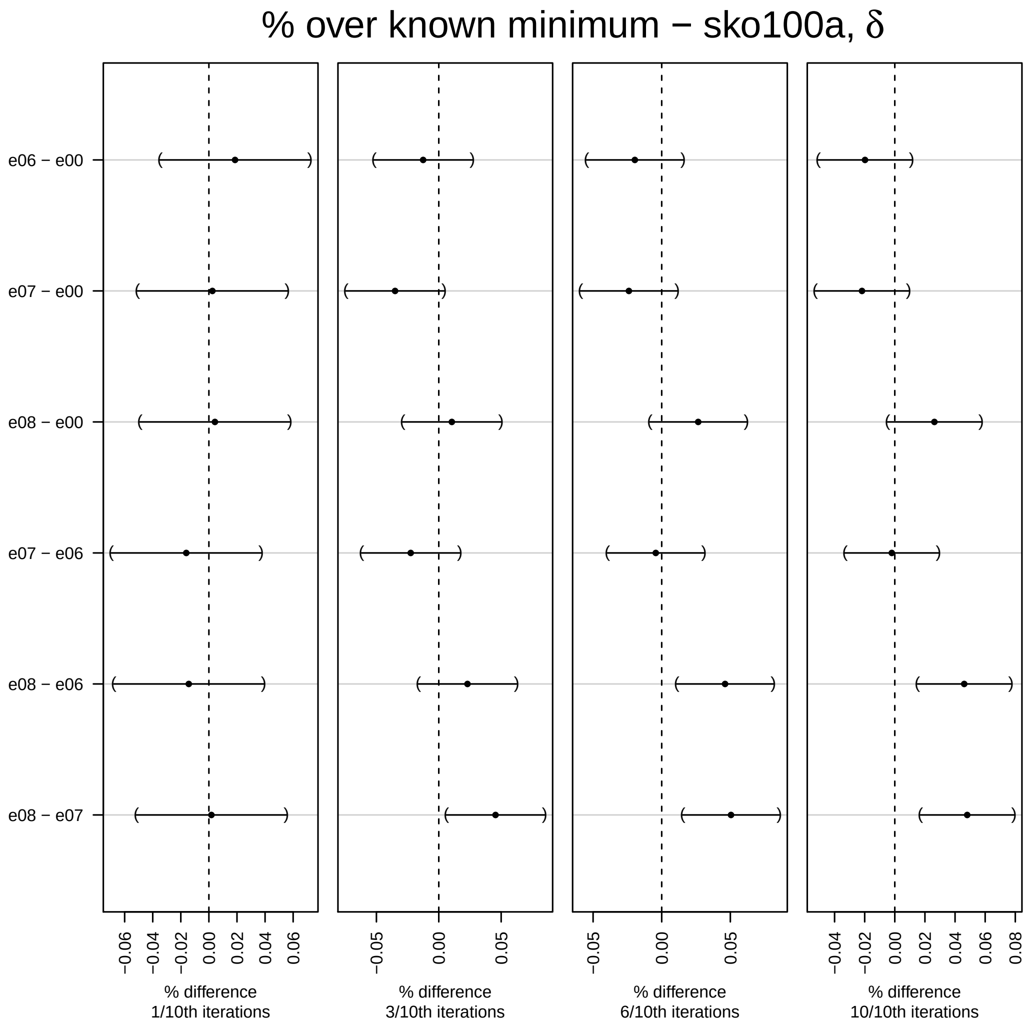


Figure 13: Sko100a [54] is the biggest instance considered in this analysis; in this case, the distances are rectangular and the entries in the flow matrix pseudo-random numbers. Given the instance size and the fact that the iteration threshold is the same as some smaller instances, it is no surprise if BIMA-QAP struggles to reach the minimum here as well. The paradigm for good performance here looks to be a limited exploration. Smaller values of φ lead to the best results, *and - as depicted above - reducing δ also improves the average best solution found by a small margin.* In this instance is rather difficult to infer general rules about $p_{sources}$, but setting p_{donor} to 1 leads to better results. Remarkably, if c is at 0 (the exploration term is ruled out from the equation) BIMA-QAP seems to perform better here; a smaller pool size still leads to a lower fitness, but it would be interesting to combine the two settings. A lower w_1 also somewhat improves the performance, which suggests that the minimum fitness is the one to be taken into account while computing $\tilde{f}$. Sko100a in not challenging just for BIMA-QAP: MLS and the random components selection perform worse than any tested setting of BIMA-QAP.

4.4 Summary and Remarks

Overall, the behavior of the algorithm is pretty satisfactory. In smaller instances BIMA-QAP easily manages to reach a global minimum, and keeps exploring looking for more. In bigger instances, while often not managing to reach the global minimum, it shows a converging behavior. BIMA-QAP beats MLS in every single analyzed instance by a margin, and - since the random components selection is generally a worse performer - the *perturb()* operator is shown to significantly contribute to the quality of the results in most of the cases.

BIMA-QAP has also shown to be reactive to different settings of its parameters, whose optimal values change according to the instance. Overall, it is pretty remarkable how $e00$ always manages to be among the best performers - its intuitively chosen set of values proved to be a good bet.

It was quite reasonable to expect the optimal pool size φ to change with the instance size, but this is not the case. Bigger values look especially suited to find more minima, but in general the optimal φ is different from instance to instance. The difference in performance dictated by a wrong pool size is often quite marked, so it is a rather important parameter to take into account. The standard value of 70 works pretty well, in general.

The algorithm reacts rather sharply to different valued perturbation size δ also. This was expected, as mentioned at the end of the previous chapter. Higher settings tend to find more minima but often result in a worse average solution in the bigger instances, with the exception of tai80b.

To different values of $p_{sources}$ correspond different behaviors of BIMA-QAP. It is often the case that some guided crossover among solutions (which means setting p_{donor} different from 0) improves the algorithm performance; tai80a is the exception to the rule, since the crossover seems to be counterproductive. An even balance between the three restricted sets of components works ok in most of the cases.

The weights w_2 and w_3 are the most difficult parameters to argue about. Experimental results show that they *do* have some influence on the algorithm, but it is not very straightforward to justify the extent of some results. In particular, the terrible performance of $e18$ in ste36a, tai60a and tai80a is quite striking - and it gets more puzzling when the same experiment gets ahead of its peers in $tai80b$. The impact of w_3 is less pronounced, but very evident in nug30, where reducing the weight brings a steep improvement in the average number of minima found.

The parameter c was conceived with the explicit task of regulating the balance between exploration and exploitation; in nug30 and ste36a it is possible to see how to a bigger c correspond more minima found, on average. The optimal c varies greatly among the instances: apart from the aforementioned case, in some (tai80a, sko100a) the smaller is c the better are BIMA-QAP performances, to the point of ruling out the exploration term in formula 9. In other cases a middle value performs best.

It was difficult to estimate the influence of w_1 on the algorithm beforehand, and the results of the experiments seem to suggest that using both $\bar{f}$ and $\check{f}$ was a good idea. In general, the suggested trend is that using mostly the minimum fitness results in a better performance, but including a percentage of the average fitness in $\tilde{f}$ makes the method more versatile.

Having so many parameters is not necessarily a good thing, since it means that the algorithm requires a process of meta-optimization to perform at its best. On the other hand, the great majority of experiments with BIMA-QAP perform better than their RND counterpart, and they all beat MLS. This is a very positive result, because it suggests that using the algorithm is worth considering even if its parameters have not been fine-tuned.

5 Conclusions and Future Work

This brief section sums up the work introduced in the past chapters, provides a simple self-assessment and suggests several aspects that could be the topic of further research.

5.1 Looking Back

The aim of this thesis was to design, implement and evaluate a new metaheuristic for the Quadratic Assignment Problem. This was partially motivated by the No Free Lunch Theorem for Combinatorial Optimization, and started with an extensive literature review - part of which made it to Chapter 2. Then, piece by piece, the elements of the metaheuristic have been assembled to obtain BIMA-QAP.

BIMA-QAP is memetic in structure, but instead of applying genetic operators on the solutions, generation after generation, it alters one solution at a time with the custom operator *perturb()*. This method enforces the δ highest scoring solution components on the original solution; the score is computed with a formula inspired by the Multi-Armed Bandit model and UCB in particular. The formula combines an exploitation term, to provide a rough estimation of the quality of a component, with an exploration term which is inversely proportional to its popularity. Two global matrices and 2φ (with φ representing the size of the solutions pool) local matrices are used to store the values for *fitness* and *pulls* needed in the computation. As a further feature, the set of components to consider can be restricted to the ones in a donor solution or to the ones involved in the solutions pool.

While drawing much inspiration from literature, the resulting method is a novelty. To study the effect of its components and try to get the best results from the test instances, several parameters have been isolated while defining the algorithm structure. These parameters have then been put to test and the results are split between Chapter 4 and Appendix A.

5.2 On the Results

The target for BIMA-QAP had been set in building a method able to converge and to outperform Multi-start Local Search. Experimental testing has shown both conditions to hold, by a margin.

Not only the method is seemingly able to converge towards a global minimum, but it actually keeps looking for more once the first one has been found. It must be noted that - despite showing a converging behavior - the algorithm is not able to reach a solution with fitness equal to the known best minimum in few of the bigger instances. This is where MLS comes handy as a baseline for performance: in every single experiment, on each instance, BIMA-QAP is performing significantly better than Multi-Start Local Search.

This doesn't mean by any chance that altering the parameters in the algorithm doesn't yield different results; in fact, the behavior of BIMA-QAP is relatively sensitive to a change in configuration. Different settings work best with different instances of the QAP, but it is possible to find some patterns in the behavior of the parameters, as pointed out in section 4.4. It was particularly interesting to see how, in the smaller instances, to higher exploration coefficient c corresponded a higher number of minima found. To better analyze the influence of the operator *perturb()* on the final results, it has been disabled in favor of a random components selection in the method RND. The experiments show the performance of RND to be much better than the one of MLS, but in the great majority of the cases significantly worse than the one of BIMA-QAP. This suggests that the biggest impact on the quality of solutions found by BIMA-QAP is due to its hybrid structure, but also shows that the operator *perturb()* is effectively improving the metaheuristic.

A comparison with existing algorithms proved to be difficult due to the different ways of analyzing performance adopted in literature; it would definitely be interesting to see how BIMA-QAP fares against some of the methods that inspired it.

5.3 Future Work

The process of creating a new hybrid metaheuristic combining various ideas found in literature can be a bit disorientating at the beginning, due to the amount of different approaches and techniques attempted in the past. Defining the design elements to combine in a single algorithm is a task that involves a lot of choices, and leaves many promising options unexplored. Similar observations can be made about the experimental phase, since the whole analysis of the interaction between the parameters has been left out - being, as noted, a multi-parameter optimization problem in itself. It doesn't come then as a surprise that the possibilities for improving this work are quite varied.

In most of the analyzed instances, either the flow or the distance matrices are symmetric. As noted in [55] the process of evaluating the fitness during local search - when a single swap is operated at a time - can be faced incrementally if both of the matrices of the instance in analysis are symmetrical. This translates into computing the fitness of a new solution by considering the fitness of the previous one and the swap that was applied on it. The paper referenced above also describes a simple way of turning instances with a single symmetric matrix into instances where both the matrices are symmetric, and states that the speed of local search has increased by a factor of 4 - definitely worth a try.

Talking about local search, it would be interesting to see how the algorithm would react if a more sophisticated method were to be used in place of 2-opt local search, for example some form of Tabu Search. The general approach during this work has often been to go for the simplest option when many were available; this translated also in the choice of the traditional MAB model and UCB as inspiration for the operator *perturb()*. Section 3.3 stressed out how several of the constraints in the model are violated; over the years several derivatives of the original MAB have been proposed, each having different properties and solution techniques - maybe a better suiting model could be adopted, if existing.

One of the issues left unsolved at design time is the effective separation among the neighborhoods encoded by the local matrices. Simply put, there is no mechanism in place to enforce a minimum distance between the solutions in the various slots, and this can limit the effectiveness of the local matrices - interpreting the results on the global exploitation weight w_2 and the global exploration weight w_3 has been difficult also due to this fact. The matter is quite delicate, because introducing additional checks could seriously hamper the speed of the method.

It has been noted that also the experimental phase could use some additional research; apart from the analysis of the mutual interaction among parameters, a technique that has not been attempted is to change the settings during BIMA-QAP execution. This can be done in three different ways. The first one is to randomly allocate the values from a set of possible ones, similarly to what was done for the restricted set of components A^t. A good candidate for this approach would be δ, since the number of swaps needed to flee an attraction basin is far from being constant. One more technique involves changing the parameters at predetermined time-steps, or following a trend chosen before starting the algorithm; for example, c could be set to monotonically increase over the iterations, in an attempt to increase the algorithm exploration over time. Finally, some parameters could dynamically adapt to better suit the current search state; a similar approach has been adopted in Adaptive-ILS [15] for the perturbation size.

Overall, the performance of BIMA-QAP can be said to be very satisfactory; in particular it proved to be really effective in exploring the search space, managing to find almost all the minima for the smaller problems.

The method is at its first iteration, and some further research would probably enhance its performance. Even in its current form, it easily reaches the targets set at the beginning of the work.

While being based on previous research, the framework adopted in BIMA-QAP is a novelty, and the use of an index formula to score the solution components is a very versatile approach that could be easily extended in the near future.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Cambridge, MA, USA: MIT Press, 1992.

[3] F. Glover and M. Laguna, *Tabu Search.* Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–80, 1983.

[5] H. Robbins, "Some Aspects of the Sequential Design of Experiments," in *Bulletin of the American Mathematical Society*, vol. 58, 1951, pp. 527–535.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, May 2002.

[7] E.-G. Talbi, *Metaheuristics : from design to implementation.* John Wiley & Sons, 2009.

[8] S. Luke, *Essentials of Metaheuristics.* Lulu, 2009.

[9] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[10] T. Stutzle, "Iterated local search for the quadratic assignment problem," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1519–1539, Nov. 2006.

[11] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, ser. STOC '71, New York, NY, USA: ACM, 1971, pp. 151–158.

[12] R. M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds., Plenum Press, 1972, pp. 85–103.

[13] R. Bellman, "An introduction to the theory of dynamic programming," 1953.

[14] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, May 1986.

[15] D. Thierens, “Adaptive operator selection for iterated local search,” in *Proceedings of the Second International Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, ser. SLS ’09, Brussels, Belgium: Springer-Verlag, 2009, pp. 140–144.

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[17] T. A. Feo and M. G. C. Resende, “A probabilistic heuristic for a computationally difficult set covering problem,” *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[18] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, “BOA: the bayesian optimization algorithm,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, Orlando, Florida, USA: Morgan Kaufmann, 1999, pp. 525–532.

[19] S. Baluja, “Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning,” Pittsburgh, PA, USA, Tech. Rep., 1994.

[20] M. Dorigo, “Optimization, Learning and Natural Algorithms,” PhD thesis, Politecnico di Milano, Italy, 1992.

[21] D. Wolpert and W. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[22] D. H. Wolpert and W. G. Macready, “Coevolutionary free lunches,” *Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, Dec. 2005.

[23] P. Moscato, “On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms,” *Caltech concurrent computation program, C3P Report*, vol. 826, 1989.

[24] R. Dawkins, *The Selfish Gene.* Oxford University Press, Oxford, UK, 1976.

[25] M. Beckman and T. Koopmans, “Assignment problems and the location of economic activities,” *Econometrica*, vol. 25, pp. 53–76, 1957.

[26] E. Cela, *The Quadratic Assignment Problem: Theory and Algorithms.* Kluwer Academic Publishers, 1998.

[27] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976.

[28] R. E. Burkard, S. E. Karisch, F. Rendl, and P. Hahn, QAPLIB - *a quadratic assignment problem library.* [Online]. Available: `http://www.seas.upenn.edu/qaplib/`.

[29] R. E. Burkard, S. E. Karisch, and F. Rendl, "QAPLIB - a quadratic assignment problem library," Department of Mathematics, Graz University of Technology, Graz, Austria, Tech. Rep., 1996.

[30] L. Steinberg, "The Backboard Wiring Problem: A Placement Algorithm," *SIAM Review*, vol. 3, no. 1, pp. 37–50, 1961.

[31] N. W. Brixius and K. M. Anstreicher, "The Steinberg Wiring Problem," *Management*, pp. 1–17, 2001.

[32] M. O. Wagner, B. Yannou, S. Kehl, D. Feillet, and J. Eggers, "Ergonomic modelling and optimization of the keyboard arrangement with an ant colony algorithm," *Journal of Engineering Design*, vol. 14, no. 2, pp. 187–208, 2003.

[33] R. Burkard and J. Offermann, *Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme*, ser. Angewandte Mathematik: Report. 1976.

[34] A. Misevičius, "A modified simulated annealing algorithm for the quadratic assignment problem," *Informatica*, vol. 14, no. 4, pp. 497–514, December 2003.

[35] É. D. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, vol. 17, no. 4-5, pp. 443–455, 1991.

[36] E. D. Taillard, "Comparison of iterative searches for the quadratic assignment problem," *Location Science*, vol. 3, no. 2, pp. 87–105, 1995.

[37] A. Misevičius, "A tabu search algorithm for the quadratic assignment problem," *Computational Optimization and Applications*, vol. 30, pp. 95–111, 2005.

[38] T. Stützle, "Max-min ant system for quadratic assignment problems," 1997.

[39] C. Fleurent, Jacques, and J. A. Ferland, "Genetic hybrids for the quadratic assignment problem," in *DIMACS Series in Mathematics and Theoretical Computer Science*, American Mathematical Society, 1993, pp. 173–187.

[40] É. D. Taillard and L. Gambardella, “Adaptive memories for the quadratic assignment problems,” Tech. Rep., 1997.

[41] Q. Zhang, J. Sun, E. Tsang, and J. Ford, “Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem,” in *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithm*, Springer-Verlag, 2006, pp. 281–292.

[42] L. Gambardella, É. Taillard, and M.Dorigo, “Ant colonies for the quadratic assignment problem,” *Journal of the Operational Research Society*, vol. 50, pp. 167–176, 1999.

[43] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[44] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. A. Castellanos, and A. Doucet, “A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot,” *Autonomous Robots*, pp. 93–103, 2009.

[45] J. Gittins, “Bandit processes and dynamic allocation indices,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 148–177, 1979.

[46] R. Dorne and C. Voudouris, “Hsf: a generic framework to easily design meta-heuristic methods,” in *4th Metaheuristics International Conference (MIC’2001), Porto, Portugal*, John Wiley & Sons, 2001, pp. 423–428.

[47] L. Di Gaspero and A. Schaerf, “EasyLocal++: an object-oriented framework for flexible design of local search algorithms,” *Software — Practice & Experience*, vol. 33, no. 8, pp. 733–765, 2003.

[48] S. Cahon, N. Melab, and E.-G. Talbi, “ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics,” *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.

[49] OpenMP Architecture Review Board, *OpenMP application program interface version 3.0*, May 2008. [Online]. Available: `http://www.openmp.org/mp-documents/spec30.pdf`.

[50] M. Saito and M. Matsumoto, “Simd-oriented fast mersenne twister: a 128-bit pseudorandom number generator,” in *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Springer Berlin Heidelberg, 2008, pp. 607–622.

[51] M. M. Drugan and D. Thierens, “Path-guided mutation for stochastic pareto local search algorithms,” in *PPSN (1)*, 2010, pp. 485–495.

[52] F. Bretz, T. Hothorn, and P. Westfall, *Multiple Comparisons Using R*. Boca Raton, Florida, USA: Chapman & Hall/CRC Press, 2010.

[53] C. E. Nugent, T. E. Vollmann, and J. Ruml, “An experimental comparison of techniques for the assignment of facilities to locations,” *Operations Research*, vol. 16, no. 1, pp. 150–173, January/February 1968.

[54] J. Chakrapani and J. Skorin-Kapov, “A connectionist approach to the quadratic assignment problem,” *Computers & Operations Research*, vol. 19, no. 3-4, pp. 287–295, 1992.

[55] P. Merz and B. Freisleben, “Fitness landscape analysis and memetic algorithms for the quadratic assignment problem,” *Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, Nov. 2000.

A Detailed Experimental Data

The aim of this appendix is to provide an overview of the huge amount of experimental data collected while testing the algorithm. The results are grouped in several tables, ordered by instance. The data has been collected at intervals of 10% of the total iterations, and in the following the intermediate timesteps at 1/10, 3/10, 6/10 and 10/10 of the total are proposed.

The section contains two types of tables, the first containing the sheer experimental results of BIMA-QAP for each instance, experiment and timestep. The second provides a list of the p-values of the pairwise comparison among the experiments run to test each parameter. As described in chapter 4, this is obtained by applying first an ANOVA and then a Tukey test on the results.

For bur26a, nug30 and ste36a two kinds of results are listed: the ones related to the average amount of different minima found, and the ones based on the best solution found on average.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e02	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e03	100.000	0.002	100.000	0.000	100.000	0.000	100.000	0.000
e04	100.001	0.002	100.000	0.000	100.000	0.000	100.000	0.000
e05	100.004	0.016	100.000	0.000	100.000	0.000	100.000	0.000
e06	100.006	0.018	100.000	0.000	100.000	0.000	100.000	0.000
e07	100.007	0.019	100.000	0.000	100.000	0.000	100.000	0.000
e08	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e09	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e10	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e11	100.003	0.015	100.000	0.000	100.000	0.000	100.000	0.000
e12	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e13	100.003	0.008	100.000	0.000	100.000	0.000	100.000	0.000
e14	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e15	100.005	0.015	100.000	0.000	100.000	0.000	100.000	0.000
e16	100.002	0.006	100.000	0.000	100.000	0.000	100.000	0.000
e17	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e18	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e19	100.002	0.006	100.000	0.000	100.000	0.000	100.000	0.000
e20	100.004	0.014	100.000	0.000	100.000	0.000	100.000	0.000
e21	100.005	0.019	100.000	0.000	100.000	0.000	100.000	0.000
e22	100.006	0.019	100.000	0.000	100.000	0.000	100.000	0.000
e23	100.002	0.008	100.000	0.000	100.000	0.000	100.000	0.000
e24	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e25	100.002	0.007	100.000	0.000	100.000	0.000	100.000	0.000
e26	100.004	0.015	100.000	0.000	100.000	0.000	100.000	0.000
e27	100.000	0.000	100.000	0.000	100.000	0.000	100.000	0.000
e28	100.005	0.013	100.000	0.000	100.000	0.000	100.000	0.000
e29	100.001	0.004	100.000	0.000	100.000	0.000	100.000	0.000
e30	100.001	0.005	100.000	0.000	100.000	0.000	100.000	0.000
e31	100.004	0.010	100.000	0.002	100.000	0.000	100.000	0.000
e32	100.001	0.005	100.000	0.000	100.000	0.000	100.000	0.000

Table A.1: Mean and standard deviation of the best solution found by each experiment in their runs on bur26a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.962	0.514	0.514	0.514
e02 - e00	0.962	0.514	0.514	0.514
e03 - e00	0.986	0.514	0.514	0.514
e04 - e00	0.997	0.514	0.514	0.514
e05 - e00	0.769	0.514	0.514	0.514
e02 - e01	1.000	1.000	1.000	1.000
e03 - e01	1.000	1.000	1.000	1.000
e04 - e01	1.000	1.000	1.000	1.000
e05 - e01	0.264	1.000	1.000	1.000
e03 - e02	1.000	1.000	1.000	1.000
e04 - e02	1.000	1.000	1.000	1.000
e05 - e02	0.264	1.000	1.000	1.000
e04 - e03	1.000	1.000	1.000	1.000
e05 - e03	0.352	1.000	1.000	1.000
e05 - e04	0.452	1.000	1.000	1.000
e06 - e00	0.634	0.494	0.494	0.494
e07 - e00	0.526	0.494	0.494	0.494
e08 - e00	1.000	0.494	0.494	0.494
e07 - e06	0.999	1.000	1.000	1.000
e08 - e06	0.553	1.000	1.000	1.000
e08 - e07	0.448	1.000	1.000	1.000
e09 - e00	0.984	0.503	0.503	0.503
e10 - e00	0.984	0.503	0.503	0.503
e11 - e00	0.995	0.503	0.503	0.503
e12 - e00	1.000	0.503	0.503	0.503
e13 - e00	0.995	0.503	0.503	0.503
e14 - e00	1.000	0.503	0.503	0.503
e10 - e09	1.000	1.000	1.000	1.000
e11 - e09	0.767	1.000	1.000	1.000
e12 - e09	1.000	1.000	1.000	1.000
e13 - e09	0.767	1.000	1.000	1.000
e14 - e09	0.998	1.000	1.000	1.000
e11 - e10	0.767	1.000	1.000	1.000
e12 - e10	1.000	1.000	1.000	1.000
e13 - e10	0.767	1.000	1.000	1.000
e14 - e10	0.998	1.000	1.000	1.000
e12 - e11	0.940	1.000	1.000	1.000
e13 - e11	1.000	1.000	1.000	1.000
e14 - e11	0.974	1.000	1.000	1.000
e13 - e12	0.940	1.000	1.000	1.000
e14 - e12	1.000	1.000	1.000	1.000
e14 - e13	0.974	1.000	1.000	1.000

comparison	1/10	3/10	6/10	10/10
e15 - e00	0.598	0.513	0.513	0.513
e16 - e00	1.000	0.513	0.513	0.513
e17 - e00	0.951	0.513	0.513	0.513
e18 - e00	0.997	0.513	0.513	0.513
e16 - e15	0.598	1.000	1.000	1.000
e17 - e15	0.197	1.000	1.000	1.000
e18 - e15	0.372	1.000	1.000	1.000
e17 - e16	0.951	1.000	1.000	1.000
e18 - e16	0.997	1.000	1.000	1.000
e18 - e17	0.997	1.000	1.000	1.000
e19 - e00	1.000	0.513	0.513	0.513
e20 - e00	0.988	0.513	0.513	0.513
e21 - e00	0.900	0.513	0.513	0.513
e22 - e00	0.804	0.513	0.513	0.513
e20 - e19	0.988	1.000	1.000	1.000
e21 - e19	0.900	1.000	1.000	1.000
e22 - e19	0.804	1.000	1.000	1.000
e21 - e20	0.996	1.000	1.000	1.000
e22 - e20	0.975	1.000	1.000	1.000
e22 - e21	1.000	1.000	1.000	1.000
e23 - e00	0.998	0.513	0.513	0.513
e24 - e00	0.998	0.513	0.513	0.513
e25 - e00	0.999	0.513	0.513	0.513
e26 - e00	0.930	0.513	0.513	0.513
e24 - e23	0.963	1.000	1.000	1.000
e25 - e23	1.000	1.000	1.000	1.000
e26 - e23	0.991	1.000	1.000	1.000
e25 - e24	0.975	1.000	1.000	1.000
e26 - e24	0.787	1.000	1.000	1.000
e26 - e25	0.984	1.000	1.000	1.000
e27 - e00	0.920	0.513	0.513	0.513
e28 - e00	0.409	0.513	0.513	0.513
e29 - e00	1.000	0.513	0.513	0.513
e30 - e00	1.000	0.513	0.513	0.513
e28 - e27	0.081	1.000	1.000	1.000
e29 - e27	0.978	1.000	1.000	1.000
e30 - e27	0.945	1.000	1.000	1.000
e29 - e28	0.274	1.000	1.000	1.000
e30 - e28	0.361	1.000	1.000	1.000
e30 - e29	1.000	1.000	1.000	1.000
e31 - e00	0.468	0.443	0.443	0.443
e32 - e00	0.997	1.000	0.443	0.443
e32 - e31	0.421	0.443	1.000	1.000

Table A.2: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on bur26a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	23.708	12.042	54.042	15.791	70.375	15.205	78.583	14.355
e02	16.042	8.995	61.792	12.793	82.333	10.124	88.667	7.251
e03	10.625	5.492	48.083	17.695	82.708	12.876	92.583	5.725
e04	3.917	3.216	29.542	9.551	67.167	11.200	88.792	5.532
e05	4.000	2.654	22.583	8.876	56.625	10.858	83.000	5.846
e06	6.667	4.967	27.125	9.940	56.208	11.673	80.208	7.729
e07	6.750	6.641	34.625	14.018	69.875	12.224	90.083	5.963
e08	5.375	4.052	37.292	10.593	76.708	8.579	93.917	2.992
e09	6.458	2.859	44.583	10.202	84.667	6.062	94.083	1.954
e10	5.292	3.394	25.208	11.632	68.708	10.045	90.333	4.622
e11	4.500	2.766	32.292	9.072	66.042	9.836	87.250	5.689
e12	6.083	4.596	35.500	11.279	78.292	10.845	92.750	3.745
e13	6.333	4.678	32.333	9.788	66.458	8.526	91.083	3.438
e14	5.208	4.384	38.708	10.511	76.542	10.446	92.375	3.334
e15	4.833	4.061	33.667	11.605	72.625	11.390	91.375	4.412
e16	6.000	4.423	35.375	11.443	72.125	9.750	91.458	3.064
e17	7.167	4.565	42.250	9.289	78.417	9.422	91.792	4.644
e18	4.833	3.497	39.083	19.328	71.833	22.983	85.792	16.529
e19	6.708	4.123	41.083	11.504	76.208	8.449	93.208	2.085
e20	5.333	4.410	35.000	11.041	72.625	8.647	91.042	3.544
e21	5.292	4.258	30.208	10.455	69.875	10.768	90.667	5.427
e22	5.500	4.773	30.500	11.306	68.125	12.099	86.292	7.298
e23	4.833	3.384	34.000	10.966	67.500	12.525	84.875	7.380
e24	7.042	5.361	37.000	14.074	75.833	8.575	92.667	2.792
e25	5.500	4.181	32.917	10.558	68.417	10.450	89.208	6.580
e26	5.667	3.852	34.208	6.600	71.417	5.445	89.208	4.201
e27	6.125	4.153	41.167	7.257	80.208	6.136	94.667	1.435
e28	4.292	3.569	34.708	10.925	73.583	10.533	92.542	3.297
e29	6.625	4.528	35.958	11.922	73.042	8.605	91.333	2.599
e30	4.292	3.368	30.333	8.879	68.667	9.494	89.917	5.090
e31	1.500	1.285	4.375	2.183	8.583	3.063	14.208	3.901
e32	3.875	2.755	24.833	7.867	60.958	10.166	86.208	5.626

Table A.3: Mean and standard deviation of the amount of minima found by each experiment in their runs on bur26a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.001	0.001	1.000	0.001
e02 - e00	0.001	0.001	0.016	0.649
e03 - e00	0.029	0.002	0.011	1.000
e04 - e00	1.000	0.894	0.794	0.685
e05 - e00	1.000	0.044	0.001	0.002
e02 - e01	0.003	0.300	0.006	0.001
e03 - e01	0.001	0.597	0.004	0.001
e04 - e01	0.001	0.001	0.927	0.001
e05 - e01	0.000	0.001	0.001	0.365
e03 - e02	0.078	0.005	1.000	0.503
e04 - e02	0.001	0.001	0.001	1.000
e05 - e02	0.001	0.000	0.001	0.123
e04 - e03	0.013	0.001	0.001	0.540
e05 - e03	0.015	0.001	0.001	0.001
e05 - e04	1.000	0.422	0.022	0.108
e06 - e00	0.411	0.208	0.001	0.001
e07 - e00	0.378	0.986	0.952	0.550
e08 - e00	0.918	0.647	0.270	0.620
e07 - e06	1.000	0.102	0.001	0.001
e08 - e06	0.801	0.012	0.001	0.001
e08 - e07	0.769	0.844	0.094	0.061
e09 - e00	0.537	0.006	0.001	0.527
e10 - e00	0.989	0.089	0.947	0.679
e11 - e00	1.000	1.000	0.390	0.001
e12 - e00	0.758	0.995	0.128	0.997
e13 - e00	0.614	1.000	0.492	0.970
e14 - e00	0.994	0.589	0.450	1.000
e10 - e09	0.939	0.001	0.001	0.013
e11 - e09	0.563	0.002	0.001	0.001
e12 - e09	1.000	0.043	0.195	0.885
e13 - e09	1.000	0.002	0.001	0.092
e14 - e09	0.916	0.441	0.037	0.703
e11 - e10	0.992	0.219	0.950	0.076
e12 - e10	0.992	0.013	0.007	0.292
e13 - e10	0.965	0.213	0.979	0.994
e14 - e10	1.000	0.001	0.051	0.501
e12 - e11	0.780	0.936	0.001	0.001
e13 - e11	0.639	1.000	1.000	0.011
e14 - e11	0.996	0.332	0.002	0.001
e13 - e12	1.000	0.939	0.001	0.727
e14 - e12	0.986	0.936	0.995	1.000
e14 - e13	0.948	0.340	0.004	0.899
e15 - e00	0.998	1.000	0.998	0.999
e16 - e00	0.676	0.987	1.000	0.999
e17 - e00	0.143	0.133	0.368	1.000
e18 - e00	0.998	0.559	1.000	0.062
e16 - e15	0.854	0.991	1.000	1.000
e17 - e15	0.269	0.146	0.561	1.000
e18 - e15	1.000	0.588	1.000	0.127
e17 - e16	0.854	0.346	0.478	1.000
e18 - e16	0.854	0.854	1.000	0.118
e18 - e17	0.269	0.913	0.431	0.085
e19 - e00	0.354	0.117	0.413	0.914
e20 - e00	0.953	0.990	0.993	0.934
e21 - e00	0.960	0.831	0.981	0.821
e22 - e00	0.913	0.873	0.754	0.001
e20 - e19	0.792	0.300	0.691	0.476
e21 - e19	0.774	0.007	0.152	0.312
e22 - e19	0.860	0.009	0.032	0.001
e21 - e20	1.000	0.545	0.856	0.999
e22 - e20	1.000	0.605	0.478	0.005
e22 - e21	1.000	1.000	0.969	0.011
e23 - e00	0.998	1.000	0.572	0.001
e24 - e00	0.196	0.787	0.451	0.995
e25 - e00	0.905	1.000	0.785	0.290
e26 - e00	0.847	1.000	1.000	0.290
e24 - e23	0.344	0.867	0.017	0.001
e25 - e23	0.981	0.997	0.997	0.030
e26 - e23	0.956	1.000	0.572	0.030
e25 - e24	0.692	0.676	0.044	0.133
e26 - e24	0.775	0.894	0.451	0.133
e26 - e25	1.000	0.994	0.785	1.000
e27 - e00	0.563	0.063	0.005	0.046
e28 - e00	1.000	0.994	0.902	0.988
e29 - e00	0.296	0.911	0.964	0.926
e30 - e00	1.000	0.802	0.793	0.136
e28 - e27	0.467	0.165	0.059	0.150
e29 - e27	0.992	0.365	0.033	0.004
e30 - e27	0.467	0.003	0.001	0.001
e29 - e28	0.226	0.993	1.000	0.684
e30 - e28	1.000	0.544	0.267	0.041
e30 - e29	0.226	0.287	0.385	0.540
e31 - e00	0.001	0.000	0.000	0.000
e32 - e00	0.729	0.001	0.001	0.001
e32 - e31	0.008	0.001	0.000	0.000

Table A.4: The p-values resulting from the pairwise comparison of the amount of minima found by the experiments on bur26a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	100.083	0.134	100.054	0.092	100.041	0.081	100.035	0.082
e02	100.024	0.032	100.016	0.029	100.008	0.022	100.005	0.018
e03	100.014	0.027	100.005	0.018	100.000	0.000	100.000	0.000
e04	100.047	0.083	100.003	0.013	100.003	0.013	100.003	0.013
e05	100.043	0.054	100.000	0.000	100.000	0.000	100.000	0.000
e06	100.050	0.053	100.016	0.029	100.008	0.022	100.008	0.022
e07	100.028	0.042	100.003	0.013	100.003	0.013	100.000	0.000
e08	100.042	0.048	100.000	0.000	100.000	0.000	100.000	0.000
e09	100.027	0.033	100.005	0.018	100.000	0.000	100.000	0.000
e10	100.035	0.044	100.016	0.029	100.011	0.025	100.008	0.022
e11	100.038	0.044	100.008	0.022	100.005	0.018	100.003	0.013
e12	100.030	0.050	100.003	0.013	100.000	0.000	100.000	0.000
e13	100.020	0.033	100.011	0.025	100.011	0.025	100.011	0.025
e14	100.022	0.031	100.005	0.018	100.000	0.000	100.000	0.000
e15	100.018	0.040	100.000	0.000	100.000	0.000	100.000	0.000
e16	100.031	0.035	100.003	0.013	100.003	0.013	100.003	0.013
e17	100.026	0.042	100.005	0.018	100.003	0.013	100.003	0.013
e18	100.024	0.032	100.000	0.000	100.000	0.000	100.000	0.000
e19	100.019	0.045	100.000	0.000	100.000	0.000	100.000	0.000
e20	100.019	0.045	100.000	0.000	100.000	0.000	100.000	0.000
e21	100.027	0.047	100.011	0.025	100.003	0.013	100.003	0.013
e22	100.042	0.050	100.016	0.029	100.003	0.013	100.000	0.000
e23	100.050	0.061	100.008	0.022	100.003	0.013	100.003	0.013
e24	100.014	0.027	100.000	0.000	100.000	0.000	100.000	0.000
e25	100.027	0.037	100.000	0.000	100.000	0.000	100.000	0.000
e26	100.042	0.058	100.003	0.013	100.000	0.000	100.000	0.000
e27	100.014	0.027	100.000	0.000	100.000	0.000	100.000	0.000
e28	100.014	0.027	100.000	0.000	100.000	0.000	100.000	0.000
e29	100.028	0.035	100.000	0.000	100.000	0.000	100.000	0.000
e30	100.037	0.054	100.000	0.000	100.000	0.000	100.000	0.000
e31	100.419	0.236	100.175	0.157	100.095	0.079	100.068	0.061
e32	100.058	0.069	100.005	0.018	100.000	0.000	100.000	0.000

Table A.5: Mean and standard deviation of the best solution found by each experiment in their runs on nug30. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.230	0.001	0.002	0.008
e02 - e00	0.992	0.729	0.966	0.995
e03 - e00	0.876	0.998	1.000	1.000
e04 - e00	0.995	1.000	1.000	1.000
e05 - e00	1.000	1.000	1.000	1.000
e02 - e01	0.060	0.017	0.019	0.038
e03 - e01	0.014	0.001	0.002	0.008
e04 - e01	0.531	0.001	0.004	0.018
e05 - e01	0.405	0.001	0.002	0.008
e03 - e02	0.996	0.938	0.966	0.995
e04 - e02	0.874	0.853	0.995	1.000
e05 - e02	0.941	0.729	0.966	0.995
e04 - e03	0.574	1.000	1.000	1.000
e05 - e03	0.701	0.998	1.000	1.000
e05 - e04	1.000	1.000	1.000	1.000
e06 - e00	0.725	0.004	0.133	0.058
e07 - e00	0.925	0.935	0.885	1.000
e08 - e00	0.976	1.000	1.000	1.000
e07 - e06	0.347	0.021	0.465	0.058
e08 - e06	0.925	0.004	0.133	0.057
e08 - e07	0.725	0.935	0.885	1.000
e09 - e00	0.981	0.965	1.000	1.000
e10 - e00	1.000	0.076	0.162	0.365
e11 - e00	1.000	0.794	0.871	0.993
e12 - e00	0.997	1.000	1.000	1.000
e13 - e00	0.779	0.493	0.162	0.085
e14 - e00	0.841	0.965	1.000	1.000
e10 - e09	0.991	0.493	0.162	0.365
e11 - e09	0.961	1.000	0.871	0.993
e12 - e09	1.000	1.000	1.000	1.000
e13 - e09	0.997	0.965	0.162	0.086
e14 - e09	1.000	1.000	1.000	1.000
e11 - e10	1.000	0.794	0.871	0.806
e12 - e10	1.000	0.224	0.162	0.365
e13 - e10	0.841	0.965	1.000	0.993
e14 - e10	0.892	0.493	0.162	0.365
e12 - e11	0.992	0.965	0.871	0.993
e13 - e11	0.707	1.000	0.871	0.365
e14 - e11	0.777	1.000	0.871	0.993
e13 - e12	0.981	0.794	0.162	0.085
e14 - e12	0.991	1.000	1.000	1.000
e14 - e13	1.000	0.965	0.162	0.085
e15 - e00	0.391	1.000	1.000	1.000
e16 - e00	0.987	0.887	0.797	0.797
e17 - e00	0.848	0.349	0.797	0.797
e18 - e00	0.782	1.000	1.000	1.000
e16 - e15	0.710	0.887	0.797	0.797
e17 - e15	0.941	0.349	0.797	0.797
e18 - e15	0.970	1.000	1.000	1.000
e17 - e16	0.987	0.887	1.000	1.000
e18 - e16	0.969	0.887	0.797	0.797
e18 - e17	1.000	0.349	0.797	0.797
e19 - e00	0.654	1.000	1.000	1.000
e20 - e00	0.654	1.000	1.000	1.000
e21 - e00	0.949	0.183	0.797	0.513
e22 - e00	0.994	0.011	0.797	1.000
e20 - e19	1.000	1.000	1.000	1.000
e21 - e19	0.971	0.183	0.797	0.513
e22 - e19	0.388	0.011	0.797	1.000
e21 - e20	0.971	0.183	0.797	0.513
e22 - e20	0.388	0.011	0.797	1.000
e22 - e21	0.777	0.804	1.000	0.513
e23 - e00	0.839	0.109	0.513	0.513
e24 - e00	0.408	1.000	1.000	1.000
e25 - e00	0.952	1.000	1.000	1.000
e26 - e00	0.994	0.925	1.000	1.000
e24 - e23	0.049	0.109	0.513	0.513
e25 - e23	0.406	0.109	0.513	0.513
e26 - e23	0.973	0.479	0.513	0.513
e25 - e24	0.840	1.000	1.000	1.000
e26 - e24	0.199	0.925	1.000	1.000
e26 - e25	0.787	0.925	1.000	1.000
e27 - e00	0.201	0.513	0.513	0.513
e28 - e00	0.201	0.513	0.513	0.513
e29 - e00	0.941	0.513	0.513	0.513
e30 - e00	1.000	0.513	0.513	0.513
e28 - e27	1.000	1.000	1.000	1.000
e29 - e27	0.629	1.000	1.000	1.000
e30 - e27	0.200	1.000	1.000	1.000
e29 - e28	0.629	1.000	1.000	1.000
e30 - e28	0.200	1.000	1.000	1.000
e30 - e29	0.940	1.000	1.000	1.000
e31 - e00	0.001	0.001	0.001	0.001
e32 - e00	0.859	0.977	1.000	1.000
e32 - e31	0.001	0.001	0.001	0.001

Table A.6: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on nug30, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	0.292	0.464	0.417	0.504	0.667	0.637	0.750	0.608
e02	0.917	0.830	1.250	0.989	1.458	0.977	1.750	1.073
e03	1.250	0.897	1.875	0.850	2.125	0.741	2.458	0.833
e04	0.667	0.637	2.500	0.933	2.917	1.060	3.083	1.100
e05	0.583	0.654	2.625	0.824	3.167	0.702	3.625	0.576
e06	0.625	0.875	1.500	1.180	1.792	1.215	1.917	1.248
e07	1.083	1.060	2.583	1.060	2.708	1.042	2.917	0.929
e08	0.792	0.932	2.625	0.970	2.958	0.955	3.375	0.770
e09	0.958	0.955	2.417	1.100	2.792	0.721	3.292	0.690
e10	0.792	0.932	1.250	0.989	1.458	0.977	1.583	0.974
e11	0.750	0.897	1.667	1.049	1.708	0.999	1.875	0.947
e12	0.917	0.881	2.333	1.090	3.000	0.933	3.292	0.806
e13	0.792	0.588	1.667	1.049	1.708	1.083	1.875	1.116
e14	1.083	0.881	2.333	1.090	2.792	0.932	3.000	0.885
e15	0.917	0.584	2.125	0.947	2.583	1.018	3.000	0.722
e16	0.792	0.884	2.167	1.007	2.667	1.129	2.958	1.083
e17	1.042	0.908	2.042	0.999	2.542	0.932	2.917	0.929
e18	0.708	0.624	1.750	0.737	2.625	0.711	3.042	0.624
e19	0.833	0.482	2.542	0.833	2.958	0.751	3.167	0.702
e20	0.958	0.624	2.375	0.875	2.875	0.741	3.042	0.690
e21	0.833	0.761	1.583	1.100	2.042	0.955	2.292	0.955
e22	0.625	0.711	1.375	1.096	1.833	1.049	2.250	0.897
e23	0.542	0.658	1.750	1.113	2.125	1.191	2.583	1.213
e24	1.542	1.103	2.583	0.974	2.875	0.741	3.083	0.830
e25	1.000	0.885	2.708	0.908	3.250	0.794	3.458	0.833
e26	0.708	0.751	2.375	1.096	3.500	0.590	3.750	0.532
e27	1.167	0.816	2.208	0.932	2.667	0.816	2.833	0.702
e28	1.250	0.847	2.625	0.875	2.833	0.868	3.167	0.761
e29	0.708	0.690	2.375	0.924	3.000	0.780	3.125	0.797
e30	0.958	0.999	2.708	0.908	3.167	0.761	3.250	0.737
e31	0.083	0.282	0.125	0.338	0.208	0.415	0.333	0.637
e32	0.542	0.833	2.042	1.083	3.333	0.868	3.875	0.338

Table A.7: Mean and standard deviation of the amount of minima found by each experiment in their runs on nug30. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.602	0.001	0.001	0.001
e02 - e00	0.729	0.007	0.001	0.001
e03 - e00	0.038	0.911	0.165	0.336
e04 - e00	1.000	0.645	0.957	0.996
e05 - e00	1.000	0.324	0.417	0.083
e02 - e01	0.038	0.011	0.018	0.002
e03 - e01	0.001	0.001	0.001	0.001
e04 - e01	0.471	0.001	0.001	0.001
e05 - e01	0.729	0.001	0.000	0.000
e03 - e02	0.602	0.117	0.075	0.054
e04 - e02	0.838	0.001	0.001	0.001
e05 - e02	0.602	0.001	0.001	0.001
e04 - e03	0.065	0.117	0.018	0.124
e05 - e03	0.022	0.031	0.001	0.001
e05 - e04	0.999	0.996	0.908	0.250
e06 - e00	1.000	0.162	0.014	0.002
e07 - e00	0.312	0.420	1.000	0.999
e08 - e00	0.922	0.342	0.834	0.437
e07 - e06	0.312	0.003	0.014	0.003
e08 - e06	0.922	0.002	0.001	0.001
e08 - e07	0.688	1.000	0.834	0.352
e09 - e00	0.824	0.960	1.000	0.858
e10 - e00	0.994	0.062	0.001	0.001
e11 - e00	0.999	0.729	0.006	0.001
e12 - e00	0.899	0.993	0.934	0.858
e13 - e00	0.994	0.729	0.006	0.001
e14 - e00	0.507	0.993	1.000	1.000
e10 - e09	0.994	0.003	0.001	0.001
e11 - e09	0.980	0.168	0.002	0.001
e12 - e09	1.000	1.000	0.988	1.000
e13 - e09	0.994	0.168	0.002	0.001
e14 - e09	0.999	1.000	1.000	0.920
e11 - e10	1.000	0.808	0.968	0.920
e12 - e10	0.999	0.008	0.001	0.001
e13 - e10	1.000	0.809	0.968	0.920
e14 - e10	0.899	0.008	0.001	0.001
e12 - e11	0.994	0.291	0.001	0.001
e13 - e11	1.000	1.000	1.000	1.000
e14 - e11	0.824	0.291	0.002	0.001
e13 - e12	0.999	0.291	0.001	0.001
e14 - e12	0.994	1.000	0.988	0.920
e14 - e13	0.899	0.291	0.002	0.001

comparison	1/10	3/10	6/10	10/10
e15 - e00	0.680	1.000	0.991	1.000
e16 - e00	0.943	1.000	1.000	1.000
e17 - e00	0.331	0.998	0.973	1.000
e18 - e00	0.996	0.625	0.999	0.998
e16 - e15	0.980	1.000	0.999	1.000
e17 - e15	0.980	0.998	1.000	0.998
e18 - e15	0.880	0.625	1.000	1.000
e17 - e16	0.790	0.990	0.991	1.000
e18 - e16	0.996	0.524	1.000	0.998
e18 - e17	0.559	0.810	0.999	0.987
e19 - e00	0.825	0.570	0.862	0.903
e20 - e00	0.437	0.899	0.966	0.997
e21 - e00	0.825	0.303	0.072	0.044
e22 - e00	1.000	0.063	0.007	0.027
e20 - e19	0.969	0.976	0.998	0.985
e21 - e19	1.000	0.008	0.005	0.003
e22 - e19	0.825	0.001	0.001	0.002
e21 - e20	0.969	0.043	0.012	0.016
e22 - e20	0.437	0.005	0.001	0.010
e22 - e21	0.825	0.946	0.924	1.000
e23 - e00	0.998	0.695	0.136	0.570
e24 - e00	0.003	0.511	0.962	0.988
e25 - e00	0.543	0.265	0.193	0.278
e26 - e00	0.998	0.910	0.016	0.018
e24 - e23	0.001	0.038	0.025	0.278
e25 - e23	0.337	0.011	0.001	0.007
e26 - e23	0.961	0.203	0.001	0.001
e25 - e24	0.182	0.993	0.556	0.570
e26 - e24	0.008	0.952	0.093	0.068
e26 - e25	0.756	0.779	0.851	0.774
e27 - e00	0.167	0.998	1.000	0.980
e28 - e00	0.076	0.320	0.985	0.878
e29 - e00	0.997	0.875	0.731	0.943
e30 - e00	0.636	0.178	0.302	0.676
e28 - e27	0.997	0.507	0.955	0.555
e29 - e27	0.318	0.969	0.622	0.676
e30 - e27	0.908	0.320	0.220	0.327
e29 - e28	0.167	0.875	0.955	1.000
e30 - e28	0.743	0.998	0.622	0.996
e30 - e29	0.836	0.709	0.955	0.980
e31 - e00	0.019	0.001	0.000	0.000
e32 - e00	0.905	0.937	0.014	0.001
e32 - e31	0.056	0.001	0.000	0.000

Table A.8: The p-values resulting from the pairwise comparison of the amount of minima found by the experiments on nug30, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	100.100	0.121	100.067	0.111	100.067	0.111	100.061	0.104
e02	100.131	0.164	100.036	0.086	100.036	0.086	100.036	0.086
e03	100.101	0.119	100.037	0.065	100.013	0.035	100.004	0.021
e04	100.278	0.213	100.031	0.058	100.000	0.000	100.000	0.000
e05	100.315	0.244	100.023	0.065	100.000	0.000	100.000	0.000
e06	100.261	0.256	100.015	0.055	100.000	0.000	100.000	0.000
e07	100.228	0.246	100.004	0.021	100.000	0.000	100.000	0.000
e08	100.337	0.225	100.059	0.095	100.000	0.000	100.000	0.000
e09	100.156	0.125	100.017	0.040	100.006	0.023	100.000	0.000
e10	100.243	0.210	100.074	0.103	100.031	0.072	100.006	0.023
e11	100.235	0.151	100.032	0.071	100.004	0.021	100.000	0.000
e12	100.138	0.140	100.009	0.030	100.000	0.000	100.000	0.000
e13	100.255	0.217	100.025	0.057	100.004	0.021	100.000	0.000
e14	100.227	0.234	100.021	0.071	100.000	0.000	100.000	0.000
e15	100.249	0.272	100.013	0.035	100.000	0.000	100.000	0.000
e16	100.268	0.193	100.006	0.023	100.000	0.000	100.000	0.000
e17	100.184	0.141	100.031	0.069	100.000	0.000	100.000	0.000
e18	100.214	0.163	100.089	0.110	100.042	0.086	100.010	0.047
e19	100.276	0.247	100.015	0.055	100.000	0.000	100.000	0.000
e20	100.152	0.210	100.030	0.069	100.004	0.021	100.000	0.000
e21	100.184	0.155	100.029	0.059	100.002	0.009	100.000	0.000
e22	100.132	0.151	100.023	0.057	100.004	0.021	100.004	0.021
e23	100.178	0.197	100.024	0.060	100.004	0.021	100.004	0.021
e24	100.187	0.154	100.036	0.081	100.000	0.000	100.000	0.000
e25	100.248	0.245	100.038	0.080	100.010	0.047	100.000	0.000
e26	100.300	0.235	100.062	0.098	100.000	0.000	100.000	0.000
e27	100.137	0.152	100.004	0.021	100.000	0.000	100.000	0.000
e28	100.263	0.154	100.013	0.035	100.000	0.000	100.000	0.000
e29	100.286	0.216	100.012	0.031	100.000	0.000	100.000	0.000
e30	100.297	0.247	100.051	0.097	100.000	0.000	100.000	0.000
e31	101.418	0.454	100.934	0.266	100.799	0.300	100.574	0.312
e32	100.369	0.192	100.086	0.098	100.012	0.031	100.000	0.000

Table A.9: Mean and standard deviation of the best solution found by each experiment in their runs on ste36a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.340	0.071	0.002	0.003
e02 - e00	0.722	0.798	0.293	0.234
e03 - e00	0.350	0.775	0.973	1.000
e04 - e00	0.722	0.908	1.000	1.000
e05 - e00	0.287	0.987	1.000	1.000
e02 - e01	0.991	0.676	0.441	0.618
e03 - e01	1.000	0.702	0.023	0.008
e04 - e01	0.011	0.515	0.002	0.003
e05 - e01	0.001	0.294	0.002	0.003
e03 - e02	0.992	1.000	0.767	0.375
e04 - e02	0.061	1.000	0.293	0.234
e05 - e02	0.008	0.990	0.293	0.234
e04 - e03	0.011	1.000	0.973	1.000
e05 - e03	0.001	0.987	0.973	1.000
e05 - e04	0.982	1.000	1.000	1.000
e06 - e00	0.834	0.984	0.494	0.494
e07 - e00	0.985	0.994	0.494	0.494
e08 - e00	0.200	0.019	0.494	0.494
e07 - e06	0.962	0.923	1.000	1.000
e08 - e06	0.663	0.050	1.000	1.000
e08 - e07	0.366	0.009	1.000	1.000
e09 - e00	0.970	1.000	0.993	1.000
e10 - e00	0.992	0.007	0.012	0.178
e11 - e00	0.998	0.846	0.999	1.000
e12 - e00	0.876	1.000	1.000	1.000
e13 - e00	0.967	0.976	0.999	1.000
e14 - e00	1.000	0.994	1.000	1.000
e10 - e09	0.658	0.031	0.089	0.178
e11 - e09	0.750	0.982	1.000	1.000
e12 - e09	1.000	1.000	0.993	1.000
e13 - e09	0.516	1.000	1.000	1.000
e14 - e09	0.839	1.000	0.993	1.000
e11 - e10	1.000	0.236	0.053	0.178
e12 - e10	0.441	0.007	0.012	0.178
e13 - e10	1.000	0.089	0.053	0.178
e14 - e10	1.000	0.054	0.013	0.178
e12 - e11	0.538	0.846	0.999	1.000
e13 - e11	1.000	1.000	1.000	1.000
e14 - e11	1.000	0.996	0.999	1.000
e13 - e12	0.313	0.976	0.999	1.000
e14 - e12	0.648	0.994	1.000	1.000
e14 - e13	0.999	1.000	0.999	1.000
e15 - e00	0.934	1.000	1.000	1.000
e16 - e00	0.801	1.000	1.000	1.000
e17 - e00	0.996	0.714	1.000	1.000
e18 - e00	1.000	0.001	0.003	0.513
e16 - e15	0.998	0.996	1.000	1.000
e17 - e15	0.776	0.846	1.000	1.000
e18 - e15	0.973	0.001	0.003	0.513
e17 - e16	0.577	0.623	1.000	1.000
e18 - e16	0.881	0.001	0.003	0.513
e18 - e17	0.983	0.015	0.003	0.513
e19 - e00	0.701	0.996	1.000	1.000
e20 - e00	0.881	0.683	0.819	1.000
e21 - e00	0.996	0.716	0.993	1.000
e22 - e00	0.691	0.905	0.819	0.513
e20 - e19	0.178	0.885	0.819	1.000
e21 - e19	0.462	0.905	0.993	1.000
e22 - e19	0.080	0.988	0.819	0.513
e21 - e20	0.980	1.000	0.968	1.000
e22 - e20	0.997	0.993	1.000	0.513
e22 - e21	0.887	0.996	0.968	0.513
e23 - e00	0.993	0.957	0.966	0.513
e24 - e00	0.999	0.707	1.000	1.000
e25 - e00	0.952	0.656	0.604	1.000
e26 - e00	0.500	0.096	1.000	1.000
e24 - e23	1.000	0.979	0.966	0.513
e25 - e23	0.774	0.965	0.935	0.513
e26 - e23	0.253	0.373	0.966	0.513
e25 - e24	0.849	1.000	0.604	1.000
e26 - e24	0.326	0.732	1.000	1.000
e26 - e25	0.904	0.779	0.604	1.000
e27 - e00	0.753	0.999	0.513	0.513
e28 - e00	0.835	0.999	0.513	0.513
e29 - e00	0.599	1.000	0.513	0.513
e30 - e00	0.470	0.040	0.513	0.513
e28 - e27	0.173	0.976	1.000	1.000
e29 - e27	0.069	0.984	1.000	1.000
e30 - e27	0.041	0.018	1.000	1.000
e29 - e28	0.995	1.000	1.000	1.000
e30 - e28	0.974	0.086	1.000	1.000
e30 - e29	1.000	0.074	1.000	1.000
e31 - e00	0.000	0.000	0.000	0.000
e32 - e00	0.155	0.244	0.968	1.000
e32 - e31	0.000	0.000	0.000	0.000

Table A.10: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on ste36a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	1.333	1.404	2.208	1.769	2.250	1.800	2.333	1.926
e02	0.917	1.283	2.833	1.926	2.917	2.041	2.917	2.041
e03	0.792	0.977	2.500	2.000	3.417	2.083	3.833	1.949
e04	0.125	0.448	2.167	1.993	5.375	1.765	6.625	1.715
e05	0.167	0.482	1.625	1.209	4.542	1.956	6.042	1.334
e06	0.333	0.637	2.833	1.711	4.542	1.693	4.750	1.847
e07	0.625	1.013	3.708	2.196	5.417	2.165	5.917	2.165
e08	0.083	0.282	1.583	1.472	4.625	1.637	5.500	1.642
e09	0.292	0.550	3.292	1.989	5.042	2.312	5.667	1.834
e10	0.250	0.532	1.208	1.351	1.792	1.474	3.292	1.829
e11	0.167	0.381	1.458	1.062	3.125	1.424	3.708	1.301
e12	0.458	0.779	3.625	2.018	5.375	2.163	5.667	2.180
e13	0.250	0.532	1.917	1.349	3.083	1.381	4.250	1.452
e14	0.375	0.647	3.750	1.871	5.458	1.978	5.917	1.717
e15	0.292	0.550	2.417	1.316	4.792	2.021	5.625	1.789
e16	0.083	0.282	2.333	1.435	5.250	2.048	5.583	1.954
e17	0.250	0.532	2.792	2.303	4.875	1.918	5.333	1.834
e18	0.250	0.532	1.000	1.063	1.708	1.268	3.000	1.560
e19	0.292	0.550	3.167	2.014	5.250	2.027	5.833	1.949
e20	0.375	0.576	2.500	1.911	5.292	1.922	5.833	1.579
e21	0.500	1.142	2.458	1.933	4.500	2.043	5.458	1.933
e22	0.500	0.722	3.083	2.020	5.125	2.193	6.000	2.106
e23	0.417	0.776	2.417	1.840	4.250	2.152	4.833	2.200
e24	0.333	0.761	2.458	1.933	4.542	1.668	5.375	1.689
e25	0.375	0.770	2.750	2.069	4.875	2.252	5.833	1.857
e26	0.167	0.381	1.167	1.274	4.333	1.949	6.542	1.318
e27	0.458	0.721	3.708	1.853	5.417	1.816	5.625	1.884
e28	0.167	0.816	3.875	2.383	6.333	1.523	6.667	1.274
e29	0.208	0.509	2.917	1.717	5.167	1.736	5.875	1.569
e30	0.167	0.381	1.667	1.404	4.958	1.546	5.708	1.398
e31	0.000	0.000	0.000	0.000	0.000	0.000	0.083	0.282
e32	0.000	0.000	0.458	0.721	1.833	1.239	3.792	1.318

Table A.11: Mean and standard deviation of the amount of minima found by each experiment in their runs on ste36a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.003	0.734	0.001	0.001
e02 - e00	0.208	1.000	0.001	0.001
e03 - e00	0.453	0.964	0.009	0.001
e04 - e00	0.991	0.683	1.000	0.986
e05 - e00	0.998	0.122	0.734	0.997
e02 - e01	0.651	0.824	0.820	0.856
e03 - e01	0.360	0.993	0.263	0.040
e04 - e01	0.001	1.000	0.001	0.001
e05 - e01	0.001	0.862	0.001	0.001
e03 - e02	0.998	0.987	0.940	0.457
e04 - e02	0.050	0.781	0.001	0.001
e05 - e02	0.075	0.174	0.037	0.001
e04 - e03	0.152	0.987	0.006	0.001
e05 - e03	0.208	0.524	0.302	0.001
e05 - e04	1.000	0.896	0.637	0.856
e06 - e00	0.997	0.999	0.462	0.018
e07 - e00	0.343	0.406	0.995	0.885
e08 - e00	0.723	0.049	0.563	0.418
e07 - e06	0.463	0.317	0.324	0.113
e08 - e06	0.594	0.073	0.999	0.466
e08 - e07	0.039	0.001	0.413	0.850
e09 - e00	1.000	0.986	1.000	0.861
e10 - e00	1.000	0.007	0.001	0.001
e11 - e00	0.991	0.038	0.001	0.001
e12 - e00	0.958	0.745	1.000	0.861
e13 - e00	1.000	0.348	0.001	0.001
e14 - e00	0.999	0.574	1.000	0.988
e10 - e09	1.000	0.001	0.001	0.001
e11 - e09	0.991	0.003	0.006	0.002
e12 - e09	0.958	0.993	0.996	1.000
e13 - e09	1.000	0.062	0.004	0.064
e14 - e09	0.999	0.960	0.984	0.999
e11 - e10	0.999	0.999	0.138	0.979
e12 - e10	0.884	0.001	0.001	0.001
e13 - e10	1.000	0.745	0.165	0.445
e14 - e10	0.991	0.001	0.001	0.001
e12 - e11	0.607	0.001	0.001	0.002
e13 - e11	0.999	0.960	1.000	0.925
e14 - e11	0.884	0.001	0.001	0.001
e13 - e12	0.884	0.008	0.001	0.064
e14 - e12	0.999	1.000	1.000	0.999
e14 - e13	0.991	0.003	0.001	0.015
e15 - e00	1.000	0.812	0.868	0.663
e16 - e00	0.632	0.710	1.000	0.609
e17 - e00	0.999	0.999	0.928	0.304
e18 - e00	0.999	0.001	0.001	0.001
e16 - e15	0.632	1.000	0.900	1.000
e17 - e15	0.999	0.926	1.000	0.977
e18 - e15	0.999	0.022	0.001	0.001
e17 - e16	0.798	0.856	0.950	0.987
e18 - e16	0.798	0.036	0.001	0.001
e18 - e17	1.000	0.002	0.001	0.001
e19 - e00	1.000	0.991	1.000	0.905
e20 - e00	0.996	0.941	1.000	0.905
e21 - e00	0.875	0.919	0.627	0.503
e22 - e00	0.875	0.999	0.999	0.981
e20 - e19	0.996	0.741	1.000	1.000
e21 - e19	0.875	0.695	0.674	0.952
e22 - e19	0.875	1.000	1.000	0.998
e21 - e20	0.979	1.000	0.627	0.952
e22 - e20	0.979	0.824	0.999	0.998
e22 - e21	1.000	0.784	0.802	0.838
e23 - e00	0.969	0.862	0.339	0.032
e24 - e00	1.000	0.896	0.663	0.351
e25 - e00	0.994	0.998	0.945	0.887
e26 - e00	0.969	0.007	0.425	0.987
e24 - e23	0.994	1.000	0.985	0.811
e25 - e23	1.000	0.966	0.795	0.265
e26 - e23	0.708	0.107	1.000	0.008
e25 - e24	1.000	0.979	0.975	0.887
e26 - e24	0.915	0.088	0.996	0.136
e26 - e25	0.826	0.020	0.867	0.611
e27 - e00	0.890	0.558	0.999	0.549
e28 - e00	0.959	0.362	0.184	0.912
e29 - e00	0.991	1.000	0.999	0.876
e30 - e00	0.959	0.127	0.955	0.671
e28 - e27	0.497	0.998	0.300	0.128
e29 - e27	0.645	0.558	0.985	0.979
e30 - e27	0.497	0.002	0.867	1.000
e29 - e28	1.000	0.362	0.104	0.372
e30 - e28	1.000	0.001	0.034	0.190
e30 - e29	1.000	0.127	0.992	0.996
e31 - e00	0.018	0.001	0.000	0.000
e32 - e00	0.018	0.001	0.001	0.001
e32 - e31	1.000	0.252	0.001	0.000

Table A.12: The p-values resulting from the pairwise comparison of the amount of minima found by the experiments on ste36a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	101.605	0.206	101.356	0.175	101.242	0.181	101.157	0.174
e02	101.627	0.146	101.293	0.208	101.169	0.197	101.090	0.163
e03	101.706	0.151	101.378	0.179	101.198	0.182	101.108	0.147
e04	101.855	0.245	101.493	0.186	101.283	0.135	101.163	0.149
e05	102.028	0.167	101.629	0.192	101.371	0.164	101.225	0.171
e06	101.614	0.189	101.354	0.133	101.247	0.147	101.171	0.149
e07	101.647	0.235	101.278	0.199	101.183	0.168	101.108	0.175
e08	102.165	0.189	101.744	0.178	101.521	0.132	101.309	0.164
e09	101.998	0.180	101.522	0.168	101.240	0.135	101.062	0.146
e10	101.850	0.212	101.468	0.168	101.264	0.184	101.182	0.164
e11	101.824	0.166	101.508	0.159	101.392	0.126	101.257	0.184
e12	101.862	0.173	101.496	0.162	101.223	0.186	101.094	0.152
e13	101.817	0.247	101.454	0.185	101.251	0.132	101.202	0.142
e14	101.844	0.190	101.460	0.231	101.200	0.138	101.093	0.160
e15	101.824	0.177	101.505	0.182	101.220	0.118	101.155	0.140
e16	101.801	0.168	101.491	0.157	101.254	0.140	101.160	0.128
e17	101.911	0.218	101.597	0.161	101.393	0.101	101.222	0.145
e18	102.313	0.214	101.824	0.255	101.547	0.238	101.441	0.244
e19	101.941	0.196	101.502	0.240	101.254	0.149	101.138	0.146
e20	101.932	0.224	101.571	0.131	101.287	0.115	101.162	0.145
e21	101.806	0.150	101.472	0.169	101.242	0.196	101.164	0.184
e22	101.826	0.179	101.466	0.200	101.242	0.142	101.162	0.113
e23	101.846	0.115	101.574	0.113	101.339	0.137	101.237	0.135
e24	101.843	0.129	101.483	0.153	101.287	0.095	101.154	0.140
e25	101.958	0.187	101.555	0.220	101.283	0.144	101.152	0.115
e26	102.070	0.225	101.733	0.282	101.522	0.237	101.181	0.178
e27	101.697	0.168	101.367	0.174	101.210	0.144	101.139	0.115
e28	101.745	0.205	101.421	0.185	101.210	0.170	101.162	0.152
e29	101.887	0.196	101.493	0.251	101.233	0.150	101.135	0.121
e30	102.050	0.206	101.732	0.196	101.357	0.179	101.153	0.129
e31	102.669	0.167	102.517	0.195	102.380	0.191	102.318	0.169
e32	102.420	0.157	102.105	0.146	101.914	0.171	101.733	0.146

Table A.13: Mean and standard deviation of the best solution found by each experiment in their runs on tai60a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.001	0.973	0.774	0.437
e02 - e00	0.004	0.396	1.000	0.999
e03 - e00	0.195	1.000	0.998	0.967
e04 - e00	0.998	0.524	0.271	0.366
e05 - e00	0.006	0.001	0.003	0.017
e02 - e01	0.999	0.861	0.696	0.710
e03 - e01	0.426	0.999	0.956	0.902
e04 - e01	0.001	0.136	0.963	1.000
e05 - e01	0.001	0.001	0.114	0.713
e03 - e02	0.691	0.635	0.992	0.999
e04 - e02	0.001	0.006	0.211	0.636
e05 - e02	0.001	0.001	0.002	0.056
e04 - e03	0.072	0.300	0.545	0.854
e05 - e03	0.001	0.001	0.011	0.141
e05 - e04	0.021	0.137	0.511	0.781
e06 - e00	0.002	0.826	0.398	0.162
e07 - e00	0.011	0.100	0.998	0.856
e08 - e00	0.001	0.001	0.001	0.001
e07 - e06	0.945	0.463	0.507	0.565
e08 - e06	0.001	0.001	0.001	0.025
e08 - e07	0.001	0.001	0.001	0.001
e09 - e00	0.054	0.239	0.778	1.000
e10 - e00	1.000	0.843	0.441	0.202
e11 - e00	1.000	0.382	0.001	0.002
e12 - e00	0.999	0.526	0.940	0.999
e13 - e00	1.000	0.944	0.627	0.072
e14 - e00	1.000	0.907	0.999	0.999
e10 - e09	0.127	0.953	0.999	0.138
e11 - e09	0.038	1.000	0.018	0.001
e12 - e09	0.195	0.999	1.000	0.993
e13 - e09	0.026	0.859	1.000	0.045
e14 - e09	0.095	0.908	0.973	0.994
e11 - e10	1.000	0.990	0.076	0.668
e12 - e10	1.000	0.999	0.971	0.494
e13 - e10	0.997	1.000	1.000	1.000
e14 - e10	1.000	1.000	0.789	0.479
e12 - e11	0.995	1.000	0.005	0.011
e13 - e11	1.000	0.949	0.035	0.901
e14 - e11	1.000	0.973	0.001	0.010
e13 - e12	0.986	0.986	0.996	0.238
e14 - e12	1.000	0.994	0.999	1.000
e14 - e13	1.000	1.000	0.915	0.227

comparison	1/10	3/10	6/10	10/10
e15 - e00	1.000	0.323	0.879	0.427
e16 - e00	0.983	0.473	0.453	0.368
e17 - e00	0.617	0.005	0.001	0.022
e18 - e00	0.001	0.001	0.001	0.001
e16 - e15	0.994	1.000	0.949	1.000
e17 - e15	0.536	0.471	0.004	0.659
e18 - e15	0.001	0.001	0.001	0.001
e17 - e16	0.289	0.321	0.031	0.721
e18 - e16	0.001	0.001	0.001	0.001
e18 - e17	0.001	0.001	0.013	0.001
e19 - e00	0.268	0.340	0.428	0.549
e20 - e00	0.350	0.019	0.116	0.242
e21 - e00	0.991	0.676	0.589	0.224
e22 - e00	1.000	0.742	0.598	0.244
e20 - e19	1.000	0.715	0.955	0.983
e21 - e19	0.104	0.983	1.000	0.978
e22 - e19	0.224	0.966	0.999	0.984
e21 - e20	0.148	0.375	0.871	1.000
e22 - e20	0.298	0.315	0.865	1.000
e22 - e21	0.997	1.000	1.000	1.000
e23 - e00	0.999	0.026	0.008	0.002
e24 - e00	1.000	0.598	0.143	0.308
e25 - e00	0.094	0.063	0.169	0.330
e26 - e00	0.001	0.001	0.001	0.086
e24 - e23	1.000	0.522	0.814	0.323
e25 - e23	0.181	0.998	0.771	0.301
e26 - e23	0.001	0.057	0.002	0.709
e25 - e24	0.160	0.732	1.000	1.000
e26 - e24	0.001	0.001	0.001	0.971
e26 - e25	0.169	0.023	0.001	0.963
e27 - e00	0.122	0.984	0.951	0.438
e28 - e00	0.538	0.996	0.952	0.164
e29 - e00	0.853	0.492	0.750	0.489
e30 - e00	0.002	0.001	0.003	0.245
e28 - e27	0.911	0.890	1.000	0.980
e29 - e27	0.008	0.208	0.990	1.000
e30 - e27	0.001	0.001	0.022	0.997
e29 - e28	0.088	0.735	0.989	0.967
e30 - e28	0.001	0.001	0.022	1.000
e30 - e29	0.034	0.001	0.077	0.993
e31 - e00	0.000	0.000	0.000	0.000
e32 - e00	0.000	0.000	0.000	0.000
e32 - e31	0.001	0.001	0.001	0.000

Table A.14: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on tai60a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	100.056	0.110	100.022	0.077	100.016	0.077	100.016	0.077
e02	100.074	0.108	100.007	0.015	100.001	0.004	100.000	0.000
e03	100.077	0.055	100.010	0.015	100.000	0.001	100.000	0.000
e04	100.168	0.113	100.030	0.030	100.002	0.007	100.000	0.000
e05	100.213	0.136	100.053	0.031	100.009	0.013	100.000	0.001
e06	100.174	0.146	100.033	0.041	100.006	0.012	100.000	0.000
e07	100.182	0.146	100.027	0.026	100.003	0.010	100.001	0.003
e08	100.114	0.070	100.016	0.020	100.003	0.007	100.000	0.000
e09	100.094	0.047	100.007	0.013	100.000	0.000	100.000	0.000
e10	100.308	0.198	100.039	0.026	100.006	0.010	100.000	0.000
e11	100.263	0.179	100.070	0.080	100.021	0.025	100.004	0.011
e12	100.115	0.081	100.025	0.021	100.002	0.006	100.000	0.000
e13	100.214	0.177	100.050	0.071	100.006	0.011	100.000	0.000
e14	100.107	0.080	100.016	0.017	100.000	0.001	100.000	0.000
e15	100.158	0.127	100.025	0.024	100.001	0.003	100.000	0.000
e16	100.200	0.153	100.033	0.027	100.008	0.016	100.000	0.000
e17	100.102	0.051	100.009	0.014	100.000	0.002	100.000	0.000
e18	100.068	0.089	100.004	0.011	100.000	0.000	100.000	0.000
e19	100.096	0.075	100.024	0.021	100.003	0.010	100.000	0.000
e20	100.147	0.116	100.015	0.017	100.001	0.003	100.000	0.000
e21	100.141	0.125	100.018	0.018	100.001	0.002	100.000	0.000
e22	100.119	0.137	100.021	0.024	100.000	0.001	100.000	0.000
e23	100.134	0.131	100.014	0.021	100.001	0.005	100.000	0.000
e24	100.176	0.154	100.018	0.020	100.001	0.002	100.000	0.000
e25	100.152	0.092	100.021	0.020	100.002	0.006	100.000	0.000
e26	100.116	0.064	100.016	0.021	100.004	0.011	100.000	0.000
e27	100.179	0.149	100.017	0.017	100.001	0.005	100.000	0.000
e28	100.156	0.155	100.017	0.026	100.001	0.003	100.000	0.000
e29	100.122	0.063	100.019	0.016	100.002	0.006	100.000	0.000
e30	100.111	0.068	100.020	0.022	100.001	0.003	100.000	0.000
e31	100.236	0.171	100.159	0.069	100.125	0.057	100.110	0.052
e32	100.095	0.052	100.033	0.026	100.005	0.010	100.000	0.000

Table A.15: Mean and standard deviation of the best solution found by each experiment in their runs on tai60b. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.003	1.000	0.656	0.514
e02 - e00	0.019	0.631	1.000	1.000
e03 - e00	0.026	0.773	1.000	1.000
e04 - e00	1.000	0.996	1.000	1.000
e05 - e00	0.895	0.087	0.976	1.000
e02 - e01	0.994	0.724	0.597	0.514
e03 - e01	0.986	0.849	0.547	0.517
e04 - e01	0.009	0.985	0.698	0.514
e05 - e01	0.001	0.060	0.972	0.532
e03 - e02	1.000	1.000	1.000	1.000
e04 - e02	0.047	0.307	1.000	1.000
e05 - e02	0.001	0.001	0.961	1.000
e04 - e03	0.062	0.442	1.000	1.000
e05 - e03	0.001	0.002	0.944	1.000
e05 - e04	0.735	0.266	0.985	1.000
e06 - e00	1.000	0.653	0.347	0.997
e07 - e00	1.000	0.989	0.983	0.505
e08 - e00	0.294	0.761	0.963	1.000
e07 - e06	0.995	0.836	0.565	0.640
e08 - e06	0.364	0.148	0.638	0.997
e08 - e07	0.245	0.563	1.000	0.504
e09 - e00	0.348	0.831	0.999	1.000
e10 - e00	0.025	0.907	0.866	1.000
e11 - e00	0.347	0.007	0.001	0.007
e12 - e00	0.692	1.000	1.000	1.000
e13 - e00	0.976	0.374	0.902	1.000
e14 - e00	0.567	0.995	1.000	1.000
e10 - e09	0.001	0.164	0.547	1.000
e11 - e09	0.001	0.001	0.001	0.007
e12 - e09	0.999	0.803	0.997	1.000
e13 - e09	0.049	0.015	0.607	1.000
e14 - e09	1.000	0.994	1.000	1.000
e11 - e10	0.924	0.175	0.001	0.008
e12 - e10	0.001	0.925	0.892	1.000
e13 - e10	0.226	0.971	1.000	1.000
e14 - e10	0.001	0.536	0.607	1.000
e12 - e11	0.006	0.008	0.001	0.007
e13 - e11	0.877	0.703	0.001	0.008
e14 - e11	0.003	0.001	0.001	0.007
e13 - e12	0.180	0.407	0.924	1.000
e14 - e12	1.000	0.992	0.999	1.000
e14 - e13	0.118	0.098	0.666	1.000
e15 - e00	0.974	1.000	0.998	0.797
e16 - e00	0.968	0.505	0.069	0.797
e17 - e00	0.154	0.090	0.977	1.000
e18 - e00	0.011	0.011	0.937	1.000
e16 - e15	0.720	0.659	0.031	1.000
e17 - e15	0.449	0.050	0.999	0.797
e18 - e15	0.060	0.005	0.991	0.797
e17 - e16	0.033	0.001	0.014	0.797
e18 - e16	0.002	0.001	0.008	0.797
e18 - e17	0.847	0.934	1.000	1.000
e19 - e00	0.120	1.000	0.783	0.513
e20 - e00	0.890	0.557	0.977	1.000
e21 - e00	0.803	0.872	0.944	1.000
e22 - e00	0.417	0.987	0.866	1.000
e20 - e19	0.570	0.521	0.420	0.513
e21 - e19	0.690	0.846	0.325	0.513
e22 - e19	0.962	0.981	0.219	0.513
e21 - e20	1.000	0.982	1.000	1.000
e22 - e20	0.926	0.853	0.996	1.000
e22 - e21	0.971	0.991	1.000	1.000
e23 - e00	0.689	0.462	0.995	0.513
e24 - e00	1.000	0.890	0.994	0.513
e25 - e00	0.939	0.992	0.997	0.513
e26 - e00	0.365	0.666	0.671	0.513
e24 - e23	0.729	0.946	1.000	1.000
e25 - e23	0.984	0.742	0.946	1.000
e26 - e23	0.986	0.998	0.422	1.000
e25 - e24	0.956	0.990	0.940	1.000
e26 - e24	0.403	0.994	0.409	1.000
e26 - e25	0.829	0.898	0.862	1.000
e27 - e00	1.000	0.767	0.988	0.513
e28 - e00	0.969	0.775	0.949	0.513
e29 - e00	0.478	0.910	0.995	0.513
e30 - e00	0.303	0.962	0.920	0.513
e28 - e27	0.965	1.000	1.000	1.000
e29 - e27	0.463	0.998	0.897	1.000
e30 - e27	0.291	0.988	0.998	1.000
e29 - e28	0.855	0.999	0.789	1.000
e30 - e28	0.689	0.989	1.000	1.000
e30 - e29	0.999	1.000	0.732	1.000
e31 - e00	0.267	0.000	0.000	0.000
e32 - e00	0.067	0.746	0.935	1.000
e32 - e31	0.001	0.001	0.000	0.000

Table A.16: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on tai60b, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	101.968	0.138	101.624	0.173	101.458	0.140	101.370	0.125
e02	102.027	0.176	101.671	0.131	101.515	0.113	101.410	0.114
e03	102.184	0.174	101.868	0.130	101.606	0.102	101.429	0.118
e04	102.326	0.126	102.050	0.109	101.784	0.145	101.595	0.140
e05	102.326	0.139	102.041	0.172	101.860	0.152	101.697	0.167
e06	101.991	0.164	101.626	0.132	101.422	0.134	101.366	0.106
e07	102.104	0.166	101.733	0.164	101.485	0.129	101.393	0.112
e08	102.425	0.130	102.271	0.123	102.056	0.143	101.823	0.109
e09	102.441	0.107	102.223	0.135	101.984	0.140	101.796	0.120
e10	102.244	0.091	101.957	0.149	101.679	0.122	101.488	0.146
e11	102.229	0.155	101.949	0.162	101.664	0.136	101.460	0.094
e12	102.320	0.105	102.008	0.151	101.726	0.137	101.555	0.143
e13	102.227	0.132	101.940	0.152	101.637	0.127	101.470	0.108
e14	102.313	0.134	101.970	0.171	101.765	0.128	101.605	0.118
e15	102.209	0.149	101.949	0.110	101.700	0.135	101.527	0.092
e16	102.204	0.196	101.914	0.154	101.681	0.102	101.522	0.117
e17	102.339	0.185	102.015	0.163	101.766	0.158	101.572	0.159
e18	102.475	0.180	102.190	0.193	102.021	0.158	101.868	0.177
e19	102.323	0.164	102.071	0.175	101.784	0.228	101.533	0.180
e20	102.291	0.129	102.068	0.151	101.761	0.170	101.559	0.127
e21	102.205	0.127	101.920	0.183	101.625	0.153	101.486	0.126
e22	102.198	0.115	101.914	0.131	101.630	0.124	101.496	0.111
e23	102.153	0.135	101.866	0.120	101.645	0.103	101.485	0.100
e24	102.265	0.174	101.915	0.170	101.644	0.145	101.479	0.140
e25	102.333	0.130	102.112	0.153	101.841	0.209	101.603	0.172
e26	102.410	0.108	102.182	0.135	102.041	0.110	101.950	0.123
e27	102.186	0.096	101.789	0.187	101.602	0.146	101.483	0.133
e28	102.141	0.199	101.881	0.147	101.660	0.151	101.525	0.142
e29	102.326	0.205	101.960	0.129	101.766	0.127	101.544	0.161
e30	102.362	0.164	102.144	0.102	101.924	0.186	101.836	0.187
e31	102.701	0.158	102.562	0.131	102.506	0.115	102.468	0.114
e32	102.549	0.127	102.436	0.110	102.321	0.132	102.231	0.101

Table A.17: Mean and standard deviation of the best solution found by each experiment in their runs on tai80a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.001	0.001	0.002	0.032
e02 - e00	0.001	0.001	0.126	0.325
e03 - e00	0.177	0.998	1.000	0.627
e04 - e00	0.961	0.006	0.001	0.093
e05 - e00	0.963	0.011	0.001	0.001
e02 - e01	0.771	0.907	0.717	0.913
e03 - e01	0.001	0.001	0.005	0.667
e04 - e01	0.001	0.001	0.001	0.001
e05 - e01	0.001	0.001	0.000	0.001
e03 - e02	0.008	0.001	0.220	0.997
e04 - e02	0.001	0.001	0.001	0.001
e05 - e02	0.001	0.001	0.001	0.001
e04 - e03	0.021	0.002	0.001	0.001
e05 - e03	0.022	0.003	0.001	0.001
e05 - e04	1.000	1.000	0.404	0.108
e06 - e00	0.001	0.001	0.001	0.003
e07 - e00	0.001	0.006	0.013	0.031
e08 - e00	0.018	0.001	0.000	0.001
e07 - e06	0.066	0.096	0.448	0.859
e08 - e06	0.001	0.000	0.000	0.000
e08 - e07	0.001	0.000	0.000	0.000
e09 - e00	0.002	0.001	0.001	0.001
e10 - e00	0.896	0.753	0.693	1.000
e11 - e00	0.670	0.844	0.891	0.981
e12 - e00	0.983	0.143	0.095	0.577
e13 - e00	0.639	0.923	0.999	0.998
e14 - e00	0.996	0.569	0.005	0.034
e10 - e09	0.001	0.001	0.001	0.001
e11 - e09	0.001	0.001	0.001	0.001
e12 - e09	0.024	0.001	0.001	0.001
e13 - e09	0.001	0.001	0.001	0.001
e14 - e09	0.014	0.001	0.001	0.001
e11 - e10	1.000	1.000	1.000	0.988
e12 - e10	0.408	0.933	0.908	0.528
e13 - e10	1.000	1.000	0.939	0.999
e14 - e10	0.527	1.000	0.335	0.027
e12 - e11	0.188	0.873	0.721	0.134
e13 - e11	1.000	1.000	0.994	1.000
e14 - e11	0.270	1.000	0.159	0.002
e13 - e12	0.170	0.773	0.292	0.241
e14 - e12	1.000	0.985	0.958	0.819
e14 - e13	0.247	0.995	0.028	0.006

comparison	1/10	3/10	6/10	10/10
e15 - e00	0.510	0.708	0.290	0.900
e16 - e00	0.452	0.982	0.544	0.935
e17 - e00	0.860	0.071	0.006	0.277
e18 - e00	0.004	0.001	0.001	0.001
e16 - e15	1.000	0.951	0.993	1.000
e17 - e15	0.082	0.652	0.524	0.807
e18 - e15	0.001	0.001	0.001	0.001
e17 - e16	0.066	0.233	0.276	0.748
e18 - e16	0.001	0.001	0.001	0.001
e18 - e17	0.062	0.005	0.001	0.001
e19 - e00	0.919	0.003	0.010	0.835
e20 - e00	1.000	0.004	0.035	0.441
e21 - e00	0.241	0.965	1.000	1.000
e22 - e00	0.173	0.985	0.999	1.000
e20 - e19	0.934	1.000	0.991	0.966
e21 - e19	0.035	0.023	0.016	0.773
e22 - e19	0.022	0.016	0.022	0.898
e21 - e20	0.220	0.026	0.056	0.371
e22 - e20	0.156	0.018	0.074	0.534
e22 - e21	1.000	1.000	1.000	0.999
e23 - e00	0.013	0.989	0.965	1.000
e24 - e00	0.979	0.978	0.971	0.999
e25 - e00	0.832	0.001	0.001	0.046
e26 - e00	0.035	0.001	0.001	0.000
e24 - e23	0.062	0.823	1.000	1.000
e25 - e23	0.001	0.001	0.001	0.032
e26 - e23	0.001	0.001	0.001	0.000
e25 - e24	0.485	0.001	0.001	0.022
e26 - e24	0.007	0.001	0.001	0.000
e26 - e25	0.343	0.547	0.001	0.001
e27 - e00	0.232	0.194	0.998	1.000
e28 - e00	0.026	1.000	0.872	0.938
e29 - e00	0.944	0.502	0.012	0.751
e30 - e00	0.571	0.001	0.001	0.001
e28 - e27	0.888	0.256	0.709	0.884
e29 - e27	0.042	0.003	0.005	0.655
e30 - e27	0.005	0.001	0.001	0.001
e29 - e28	0.003	0.412	0.141	0.994
e30 - e28	0.001	0.001	0.001	0.001
e30 - e29	0.947	0.001	0.007	0.001
e31 - e00	0.001	0.000	0.000	0.000
e32 - e00	0.001	0.000	0.000	0.000
e32 - e31	0.003	0.017	0.001	0.001

Table A.18: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on tai80a, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e01	100.501	0.373	100.360	0.364	100.325	0.346	100.232	0.279
e02	100.531	0.327	100.155	0.229	100.114	0.191	100.098	0.174
e03	100.560	0.430	100.149	0.145	100.052	0.063	100.032	0.058
e04	100.717	0.357	100.187	0.171	100.071	0.039	100.038	0.020
e05	100.789	0.343	100.340	0.210	100.115	0.062	100.060	0.026
e06	101.025	0.359	100.389	0.297	100.166	0.178	100.088	0.108
e07	100.942	0.506	100.254	0.218	100.113	0.142	100.055	0.071
e08	100.546	0.366	100.111	0.060	100.054	0.024	100.027	0.018
e09	100.417	0.361	100.131	0.152	100.066	0.133	100.015	0.015
e10	100.919	0.489	100.269	0.225	100.116	0.163	100.043	0.040
e11	100.918	0.410	100.465	0.308	100.283	0.290	100.143	0.215
e12	100.580	0.360	100.132	0.108	100.049	0.026	100.026	0.018
e13	101.027	0.358	100.278	0.232	100.128	0.137	100.068	0.072
e14	100.615	0.377	100.142	0.143	100.055	0.031	100.030	0.021
e15	100.860	0.482	100.242	0.236	100.094	0.058	100.054	0.028
e16	100.907	0.465	100.269	0.202	100.074	0.068	100.046	0.059
e17	100.472	0.320	100.111	0.092	100.041	0.024	100.019	0.015
e18	100.260	0.246	100.050	0.045	100.010	0.011	100.006	0.010
e19	100.788	0.455	100.222	0.191	100.091	0.072	100.049	0.062
e20	100.764	0.352	100.213	0.219	100.068	0.035	100.030	0.021
e21	100.815	0.421	100.242	0.274	100.071	0.051	100.032	0.022
e22	100.839	0.411	100.371	0.317	100.167	0.211	100.076	0.134
e23	100.833	0.415	100.223	0.198	100.100	0.109	100.063	0.101
e24	100.750	0.421	100.226	0.223	100.092	0.128	100.045	0.060
e25	100.634	0.380	100.235	0.215	100.074	0.064	100.036	0.023
e26	100.508	0.382	100.165	0.168	100.057	0.023	100.029	0.017
e27	100.927	0.507	100.283	0.267	100.169	0.182	100.072	0.090
e28	100.765	0.367	100.189	0.168	100.078	0.083	100.033	0.023
e29	100.723	0.459	100.175	0.140	100.066	0.034	100.036	0.020
e30	100.752	0.439	100.162	0.129	100.069	0.078	100.037	0.023
e31	100.885	0.400	100.627	0.279	100.453	0.191	100.375	0.142
e32	100.334	0.182	100.116	0.040	100.078	0.027	100.057	0.014

Table A.19: Mean and standard deviation of the best solution found by each experiment in their runs on tai80b. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.018	0.236	0.001	0.001
e02 - e00	0.041	0.938	0.996	0.901
e03 - e00	0.082	0.911	0.975	0.992
e04 - e00	0.817	0.998	1.000	0.998
e05 - e00	0.993	0.398	0.995	1.000
e02 - e01	1.000	0.024	0.001	0.016
e03 - e01	0.994	0.019	0.001	0.001
e04 - e01	0.342	0.089	0.001	0.001
e05 - e01	0.086	1.000	0.001	0.001
e03 - e02	1.000	1.000	0.802	0.574
e04 - e02	0.517	0.997	0.951	0.669
e05 - e02	0.164	0.057	1.000	0.932
e04 - e03	0.690	0.993	0.999	1.000
e05 - e03	0.278	0.045	0.792	0.984
e05 - e04	0.985	0.178	0.946	0.995
e06 - e00	0.456	0.021	0.138	0.466
e07 - e00	0.864	0.912	0.910	1.000
e08 - e00	0.057	0.284	0.763	0.542
e07 - e06	0.898	0.107	0.436	0.453
e08 - e06	0.001	0.001	0.012	0.032
e08 - e07	0.007	0.075	0.360	0.555
e09 - e00	0.004	0.762	0.999	0.727
e10 - e00	0.997	0.967	0.996	1.000
e11 - e00	0.997	0.001	0.001	0.022
e12 - e00	0.221	0.777	0.967	0.916
e13 - e00	0.711	0.931	0.972	1.000
e14 - e00	0.385	0.867	0.986	0.957
e10 - e09	0.001	0.204	0.904	0.939
e11 - e09	0.001	0.001	0.001	0.001
e12 - e09	0.784	1.000	1.000	1.000
e13 - e09	0.001	0.146	0.780	0.423
e14 - e09	0.590	1.000	1.000	0.999
e11 - e10	1.000	0.016	0.004	0.005
e12 - e10	0.051	0.216	0.704	0.995
e13 - e10	0.965	1.000	1.000	0.967
e14 - e10	0.115	0.303	0.791	0.999
e12 - e11	0.053	0.001	0.001	0.001
e13 - e11	0.963	0.025	0.008	0.079
e14 - e11	0.117	0.001	0.001	0.001
e13 - e12	0.003	0.155	0.527	0.682
e14 - e12	1.000	1.000	1.000	1.000
e14 - e13	0.008	0.226	0.624	0.775
e15 - e00	1.000	0.980	0.999	1.000
e16 - e00	0.987	0.784	0.892	0.949
e17 - e00	0.010	0.177	0.038	0.043
e18 - e00	0.001	0.006	0.001	0.003
e16 - e15	0.994	0.979	0.756	0.972
e17 - e15	0.008	0.047	0.017	0.058
e18 - e15	0.001	0.001	0.001	0.004
e17 - e16	0.002	0.009	0.288	0.229
e18 - e16	0.001	0.001	0.003	0.023
e18 - e17	0.334	0.693	0.370	0.868
e19 - e00	0.985	1.000	1.000	0.998
e20 - e00	0.950	1.000	0.964	0.745
e21 - e00	0.999	0.996	0.980	0.815
e22 - e00	1.000	0.168	0.115	0.882
e20 - e19	1.000	1.000	0.953	0.906
e21 - e19	1.000	0.999	0.972	0.945
e22 - e19	0.993	0.204	0.129	0.708
e21 - e20	0.993	0.994	1.000	1.000
e22 - e20	0.969	0.153	0.021	0.208
e22 - e21	1.000	0.338	0.027	0.263
e23 - e00	1.000	1.000	0.994	0.996
e24 - e00	0.909	1.000	1.000	0.974
e25 - e00	0.338	0.998	0.980	0.816
e26 - e00	0.030	0.898	0.738	0.586
e24 - e23	0.952	1.000	0.999	0.860
e25 - e23	0.421	1.000	0.857	0.589
e26 - e23	0.044	0.843	0.470	0.352
e25 - e24	0.852	1.000	0.956	0.990
e26 - e24	0.225	0.819	0.656	0.912
e26 - e25	0.807	0.731	0.966	0.996
e27 - e00	0.973	0.695	0.071	0.844
e28 - e00	0.962	0.986	0.996	0.594
e29 - e00	0.852	0.935	0.945	0.736
e30 - e00	0.938	0.841	0.968	0.748
e28 - e27	0.698	0.374	0.027	0.103
e29 - e27	0.487	0.236	0.009	0.169
e30 - e27	0.635	0.144	0.012	0.177
e29 - e28	0.998	0.999	0.996	1.000
e30 - e28	1.000	0.986	0.999	1.000
e30 - e29	1.000	1.000	1.000	1.000
e31 - e00	0.933	0.001	0.000	0.000
e32 - e00	0.001	0.161	0.950	1.000
e32 - e31	0.001	0.001	0.000	0.000

Table A.20: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on tai80b, at the 4 timesteps.

experiment	μ 1/10	σ 1/10	μ 3/10	σ 3/10	μ 6/10	σ 6/10	μ 10/10	σ 10/10
e00	100.445	0.069	100.285	0.057	100.221	0.043	100.183	0.034
e01	100.289	0.059	100.171	0.041	100.130	0.040	100.113	0.036
e02	100.355	0.082	100.229	0.053	100.152	0.036	100.124	0.028
e03	100.371	0.053	100.245	0.050	100.181	0.034	100.129	0.030
e04	100.460	0.090	100.300	0.055	100.232	0.034	100.192	0.032
e05	100.528	0.076	100.347	0.056	100.261	0.036	100.205	0.045
e06	100.464	0.068	100.272	0.058	100.201	0.063	100.164	0.052
e07	100.448	0.078	100.250	0.047	100.197	0.039	100.162	0.039
e08	100.449	0.066	100.296	0.044	100.247	0.036	100.210	0.037
e09	100.421	0.055	100.291	0.045	100.216	0.035	100.152	0.041
e10	100.454	0.068	100.297	0.048	100.211	0.044	100.186	0.035
e11	100.468	0.076	100.311	0.046	100.238	0.040	100.195	0.033
e12	100.411	0.086	100.265	0.050	100.220	0.037	100.175	0.025
e13	100.465	0.070	100.309	0.058	100.219	0.042	100.183	0.040
e14	100.417	0.069	100.278	0.060	100.201	0.037	100.172	0.032
e15	100.450	0.072	100.270	0.059	100.213	0.053	100.180	0.043
e16	100.459	0.070	100.303	0.046	100.239	0.042	100.188	0.035
e17	100.405	0.074	100.265	0.051	100.202	0.039	100.162	0.040
e18	100.456	0.053	100.313	0.048	100.239	0.053	100.182	0.041
e19	100.446	0.083	100.286	0.045	100.228	0.028	100.179	0.027
e20	100.427	0.079	100.277	0.051	100.220	0.032	100.190	0.034
e21	100.452	0.080	100.279	0.048	100.213	0.039	100.165	0.038
e22	100.431	0.068	100.270	0.040	100.191	0.035	100.162	0.025
e23	100.435	0.077	100.280	0.046	100.206	0.036	100.155	0.035
e24	100.441	0.075	100.262	0.056	100.208	0.041	100.180	0.034
e25	100.440	0.066	100.267	0.041	100.217	0.030	100.188	0.038
e26	100.454	0.068	100.311	0.052	100.233	0.034	100.189	0.042
e27	100.408	0.057	100.259	0.056	100.190	0.037	100.148	0.031
e28	100.416	0.089	100.268	0.042	100.200	0.042	100.160	0.042
e29	100.468	0.046	100.300	0.042	100.239	0.037	100.198	0.028
e30	100.436	0.087	100.307	0.042	100.235	0.043	100.196	0.033
e31	100.666	0.071	100.588	0.070	100.573	0.066	100.534	0.072
e32	100.456	0.067	100.356	0.050	100.285	0.045	100.248	0.037

Table A.21: Mean and standard deviation of the best solution found by each experiment in their runs on sko100a. Each pair of columns represents a timestep.

comparison	1/10	3/10	6/10	10/10
e01 - e00	0.001	0.001	0.001	0.001
e02 - e00	0.001	0.005	0.001	0.001
e03 - e00	0.009	0.097	0.005	0.001
e04 - e00	0.979	0.920	0.884	0.959
e05 - e00	0.002	0.001	0.004	0.273
e02 - e01	0.027	0.003	0.306	0.857
e03 - e01	0.003	0.001	0.001	0.571
e04 - e01	0.001	0.001	0.001	0.001
e05 - e01	0.000	0.000	0.000	0.001
e03 - e02	0.970	0.890	0.082	0.997
e04 - e02	0.001	0.001	0.001	0.001
e05 - e02	0.001	0.001	0.000	0.001
e04 - e03	0.001	0.006	0.001	0.001
e05 - e03	0.001	0.001	0.001	0.001
e05 - e04	0.021	0.027	0.099	0.786
e06 - e00	0.799	0.839	0.469	0.347
e07 - e00	1.000	0.100	0.291	0.264
e08 - e00	0.997	0.900	0.205	0.127
e07 - e06	0.859	0.446	0.989	0.999
e08 - e06	0.896	0.426	0.005	0.002
e08 - e07	1.000	0.018	0.002	0.001
e09 - e00	0.906	1.000	1.000	0.028
e10 - e00	1.000	0.986	0.983	1.000
e11 - e00	0.922	0.597	0.731	0.905
e12 - e00	0.631	0.831	1.000	0.982
e13 - e00	0.959	0.696	1.000	1.000
e14 - e00	0.831	1.000	0.629	0.924
e10 - e09	0.676	1.000	1.000	0.014
e11 - e09	0.258	0.819	0.496	0.001
e12 - e09	0.999	0.614	1.000	0.225
e13 - e09	0.335	0.888	1.000	0.031
e14 - e09	1.000	0.984	0.840	0.372
e11 - e10	0.994	0.967	0.229	0.968
e12 - e10	0.344	0.336	0.988	0.935
e13 - e10	0.999	0.987	0.994	1.000
e14 - e10	0.559	0.875	0.979	0.826
e12 - e11	0.082	0.040	0.700	0.418
e13 - e11	1.000	1.000	0.641	0.893
e14 - e11	0.181	0.313	0.027	0.260
e13 - e12	0.118	0.060	1.000	0.985
e14 - e12	1.000	0.974	0.663	1.000
e14 - e13	0.243	0.401	0.720	0.934
e15 - e00	0.999	0.865	0.979	0.997
e16 - e00	0.954	0.750	0.652	0.996
e17 - e00	0.257	0.692	0.644	0.294
e18 - e00	0.982	0.341	0.636	1.000
e16 - e15	0.992	0.194	0.302	0.953
e17 - e15	0.155	0.998	0.930	0.492
e18 - e15	0.999	0.041	0.290	1.000
e17 - e16	0.054	0.097	0.054	0.144
e18 - e16	1.000	0.964	1.000	0.987
e18 - e17	0.080	0.017	0.051	0.368
e19 - e00	1.000	1.000	0.942	0.991
e20 - e00	0.926	0.981	1.000	0.943
e21 - e00	0.999	0.992	0.936	0.260
e22 - e00	0.970	0.823	0.042	0.150
e20 - e19	0.917	0.966	0.935	0.742
e21 - e19	0.999	0.983	0.539	0.523
e22 - e19	0.964	0.776	0.005	0.353
e21 - e20	0.802	1.000	0.943	0.049
e22 - e20	1.000	0.987	0.045	0.023
e22 - e21	0.885	0.973	0.245	0.999
e23 - e00	0.989	0.997	0.672	0.059
e24 - e00	1.000	0.517	0.771	0.997
e25 - e00	1.000	0.722	0.999	0.991
e26 - e00	0.992	0.391	0.795	0.980
e24 - e23	0.998	0.742	1.000	0.133
e25 - e23	0.999	0.900	0.838	0.017
e26 - e23	0.884	0.212	0.109	0.012
e25 - e24	1.000	0.998	0.908	0.927
e26 - e24	0.973	0.010	0.156	0.892
e26 - e25	0.963	0.025	0.618	1.000
e27 - e00	0.378	0.342	0.079	0.004
e28 - e00	0.621	0.736	0.399	0.122
e29 - e00	0.804	0.807	0.493	0.586
e30 - e00	0.992	0.516	0.736	0.681
e28 - e27	0.996	0.969	0.922	0.712
e29 - e27	0.035	0.030	0.001	0.001
e30 - e27	0.657	0.008	0.002	0.001
e29 - e28	0.094	0.144	0.009	0.002
e30 - e28	0.869	0.046	0.028	0.003
e30 - e29	0.533	0.990	0.996	1.000
e31 - e00	0.000	0.000	0.000	0.000
e32 - e00	0.849	0.001	0.001	0.001
e32 - e31	0.000	0.000	0.000	0.000

Table A.22: The p-values resulting from the pairwise comparison of the average best solution obtained by the experiments on sko100a, at the 4 timesteps.