ADAPTIVE OPERATOR SEARCH FOR THE CAPACITATED ARC ROUTING PROBLEM

by

PIETRO A. CONSOLI

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
January 2018

Abstract

Evolutionary Computation approaches for Combinatorial Optimization have been successfully proposed for a plethora of different NP-Hard Problems. This research area has achieved acknowledgeable results and obtained remarkable progresses, and it has ultimately established itself as one of the most studied in AI. Yet, predicting the approximation ability of Evolutionary Algorithms (EAs) on novel problem instances remains a difficult easy task. As a consequence, their application in a real-world optimization context is reduced, as EAs are often considered not reliable and mature enough to be adopted in an industrial scenario.

This thesis proposes new approaches to endow such meta-heuristics with a mechanism that would allow them to extract information during the search and to adaptively use such information in order to modify their behaviour and ultimately improve their performances. We consider the case study of the Capacitated Arc Routing Problem (CARP), to demonstrate the effectiveness of adaptive search techniques in a complex problem deeply connected with real-world scenarios. In particular, the main contributions of this thesis are:

1. An investigation of the adoption of a Parameter Tuning mechanism to adaptively choose the crossover operator that is used during the search;
2. The study of a novel Adaptive Operator Selection technique based on the use of Fitness Landscape Analysis techniques and on Online Learning;

3. A novel approach based on Knowledge Incorporation focusing on the reuse of information learned from the execution of a meta-heuristic on past instances, that is later used to improve the performances on the newly encountered.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

I'm Wiston Wolf. I solve problems.

Wiston Wolf (Harvey Keitel)

Pulp Fiction

Ever since its first steps, halfway through the last century, the research field of Evolutionary Computation has achieved great accomplishments and consolidated as one of the most studied fields of the past decades in Artificial Intelligence. Countless papers have been published to testify the successful application of Evolutionary Algorithms in the most disparate optimization problems. The integration of such meta-heuristics with domain expert knowledge has given birth in several cases to extremely competitive and acknowledged technologies that have also gained commercial success in their application to real world problems. Yet, what EAs still are not and will not be capable to offer (unless P=NP is proven) is the capability of finding the optimal solution for every possible problem instance of every possible NP-Hard problem in what can be measured as a computationally reasonable time. Thus, despite their great potential and their almost effortless capacity of adaptation to countless domains regardless of the degree of complexity, the adoption of EAs in industrial and commercial activities is, if not limited, most certainly downsized. A higher level of interaction between industry and academia would nevertheless be advantageous and auspicable, as the interaction between research and industry would benefit both actors: companies being be more competitive in the market through the adoption of cutting-edge technologies, and academia being fed with new challenges in the form of novel domains to explore and study. In this sense, as Papadimitriou and Steigliz already claimed in 1998 [107], "Developing the mathematical methodology for explaining and predicting the performance of these heuristics is one of the most important challenges facing the fields of optimisation and algorithms today ...".

Consequently, in the past years researchers in Evolutionary Computation have been driving towards this direction. In particular, great progress has been made towards the definition and the development of theories and techniques that can be used to perform an analysis of the computational complexity of EAs [105], which can subsequently used to determine a provable approximation ratio of meta-heuristics.

At the same time, another direction that researchers have been following is the development of techniques for Parameter Tuning of meta-heuristics. Coming to terms with the reality of the well known No Free Lunch Theorem [141], researchers have, in the past decades, designed and devised techniques to optimize the performance of EAs through the use of a plethora of techniques capable of selecting the best parameters settings of the algorithms, choosing one evolutionary operator over the other, either a priori or in an online fashion.

This thesis focuses on the adaptation of existing Parameter Tuning techniques, as well as the definition of novel ones, to the well known optimization problem called Capacitated Arc Routing Problem (CARP). CARP is a NP-Hard routing problem with strong connections with real world problems, such as winter gritting, waste collection or postal service. As a consequence, this problem is of great interest for the scientific community, and a large number of heuristics, exact methods and meta-heuristics have been proposed for both this problem and for its many variants.

This chapter is further divided in the following sections. Section 1.1 formalizes the concept of Approximation Algorithm, describes the Parameter Tuning research field and provides a formal definition of the Capacitated Arc Routing Problem. 1.2 illustrates the research questions that this thesis aims to answer to. Section 1.3 summarizes the main contributions of the work described in this thesis. Finally, Section 1.4 illustrates the organization of the thesis.

1.1 Problem Formulation

1.1.1 Approximation Algorithms

An Approximation Algorithm can be defined as follows [107]. Given a minimization optimization problem P with a cost function c, and an algorithm A that is able to produce a feasible solution $f_A(I)$ for the instance I of P whose optimal solution is $\overline{f}(I)$, then the algorithm A is an $\varepsilon-$approximate algorithm of P if for any instance I of P :

$$\varepsilon \geq \frac{|c(f_A(I)) - c(\overline{f}(I))|}{c(\overline{f}(I))} \tag{1}$$

for some value $\varepsilon \geq 0$ and ε is the *approximation ratio* of A.

Despite their stochastic nature, Evolutionary Algorithms can indeed be considered as Approximation Algorithms. Proving the approximation ratio of an EA is however an extremely complex task. As explained in [52], when evaluating the approximation of EAs, due to their stochastic nature, it is more convenient to evaluate the estimated value $E(f_A(I))$ instead of the results obtained by a single execution of the EA. In this thesis the approximation ability of an Evolutionary Algorithm is intended as a more generic capacity to obtain robust results with comparable approximation ratio, across a heterogeneous set of problem instances of different sizes and characteristics, that is achieved through an increased capacity of adaptation and learning.

1.1.2 Parameter Setting Taxonomy

The performances of Evolutionary Algorithms are greatly influenced by the parameter settings that are adopted and by the choices that are performed during their design. Factors like solution representation, the choice of variation operators or the fitness function have several parameters that can affect greatly the results of EAs. In order to identify the best parameter setting, studies like Sensitivity Analysis are often conducted to understand how the variation of any of these factors affects the performances of the algorithms, along with time-expensive offline parameter optimizations. Parameter Optimization is, in fact, a higher level search conducted in the search space of parameters settings. Several techniques and tools like Statistical Racings [10], ParamILS[60] and Irace[76] have been developed to automatically perform this task.

Parameter Setting is the research area that encompasses a broad selection of techniques dealing with this topic. In order to provide a clearer and more articulate picture, it is useful to follow the taxonomy illustrated in the 1999 survey by Eiben et al.[30], whose terminology is largely adopted by the Parameter Setting community. The authors dis-

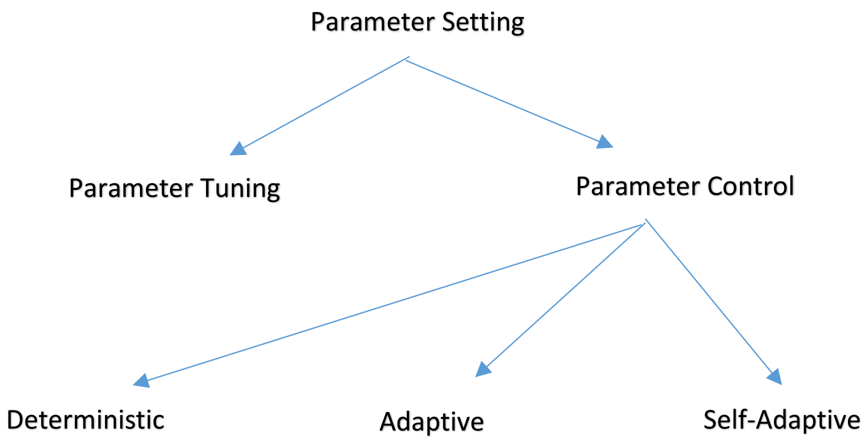


Figure 1.1: Taxonomy of Parameter Setting as described in [30]

tinguish two ways to perform the Parameter Setting, either before the run, or *offline*, or during the run, or *online*. In the former case the term used is that of *Parameter Tuning*, while the latter is called *Parameter Control*. Parameter Control techniques can be further divided into three categories. They can be classified as *Deterministic*, *Adaptive* and *Self-Adaptive*. The first is the case when a specific rule is applied, perhaps in a periodic fashion, to modify the parameter setting of the algorithm. The second case occurs when there is some sort of measurement of the performances of the algorithm, and of its parameters, whose evaluation is later used to influence the choice. Finally, the *Self-Adaptive* case is that of those algorithms that can evolve their parameters without any user preference, by encoding the parameter setting in their representation and allowing the evolutionary mechanisms to identify the most useful configuration. The Parameter Setting taxonomy is illustrated in figure 1.1. An alternative and accurate analysis of the parameter setting scenario is available in [34].

1.1.3 Capacitated Arc Routing Problem

The Capacitated Arc Routing Problem (CARP), the case study of this thesis, is a NP-Hard combinatorial problem belonging to the wide family of Vehicle Routing Problems (VRP). While in the latter case the problem is that of identifying the optimal routing plan of one or more vehicles that need to go through specific vertices of a map, CARP requires the vehicle to service arcs rather than vertices. To provide a concrete example, while in the traditional VRP case a vehicle needs to go through specific road intersections, in the case of CARP the vehicle is required to provide a service along specific roads. Moreover, vehicles have a limited capacity and therefore they can only be used to service a certain amount of goods/load before returning to the depot. Winter gritting, postal delivery, garbage collection, can all be considered typical applications of CARP to real world problem.

$G = (V, A)$ is a connected directed graph where V is the set of vertices, A is the set of arcs, and the subset $A_R \subset A$ is the subset of *required arcs.* Elements of A are also called *tasks*, while A_R will be the *required tasks.* Each tasks t has a *demand* $d(t)$ which indicates the load necessary to serve the task, a *service cost* $sc(t)$ of crossing the task, a *dead-heading cost* $dc(t)$ of crossing the task without serving it, an *ID* and a reference to their *head(t)* and their *tail(t)* task. The *tasks* need to be served by a fleet of m vehicles with a capacity C and whose route starts and ends in a vertex called *depot*. Each task must be served within a single tour and each vehicle is bound to its capacity. A solution for a CARP instance can therefore be represented by a set of routes, which are sequences of tasks that need to be visited in the given order. To distinguish between routes a *dummy task* is commonly used with *ID* 0, which represents the vehicle being in the depot with null *service cost* and *demand.*

$$S = \{\{t_0, t_k, ..., t_l, t_0\}, \cdots, \{t_0, t_p, ..., t_q, t_0\}\}$$

The objective is therefore to minimize the total service cost (TC) of the routing subject to the previously mentioned constraints. The TC is calculated in the following way. When the vehicle is serving a *required task* t_i, the TC will include its serving cost $sc(t_i)$ plus the total cost of the shortest path (sp) necessary to connect the *tail* of the task to the *head* of the next task t_{i+1}, obtained through the use of Dijkstra algorithm[26].

Formally, the Capacitated Arc Routing Problem can be defined as:

$$\min \mathrm{TC}(S) = \sum_{i=1}^{\mathrm{length}(S)-1} (sc(t_i) + sp(t_i, t_{i+1})), \text{subject to the constraints} \tag{2}$$

$$load(R_j) \leq C \tag{3}$$

$$app(t_i) = 1 \forall t_i \in T \tag{4}$$

$$m <= n_{veh} \tag{5}$$

$$load(R_j) = \sum_{i=1}^{\text{length}(R_j)} d(t_i) \tag{6}$$

where R_j is the $j-th$ route of the solution, $app(t_i)$ gives the number of appearances of tasks t_i in the sequence of the tasks in S and n_{veh} is the number of available vehicles.

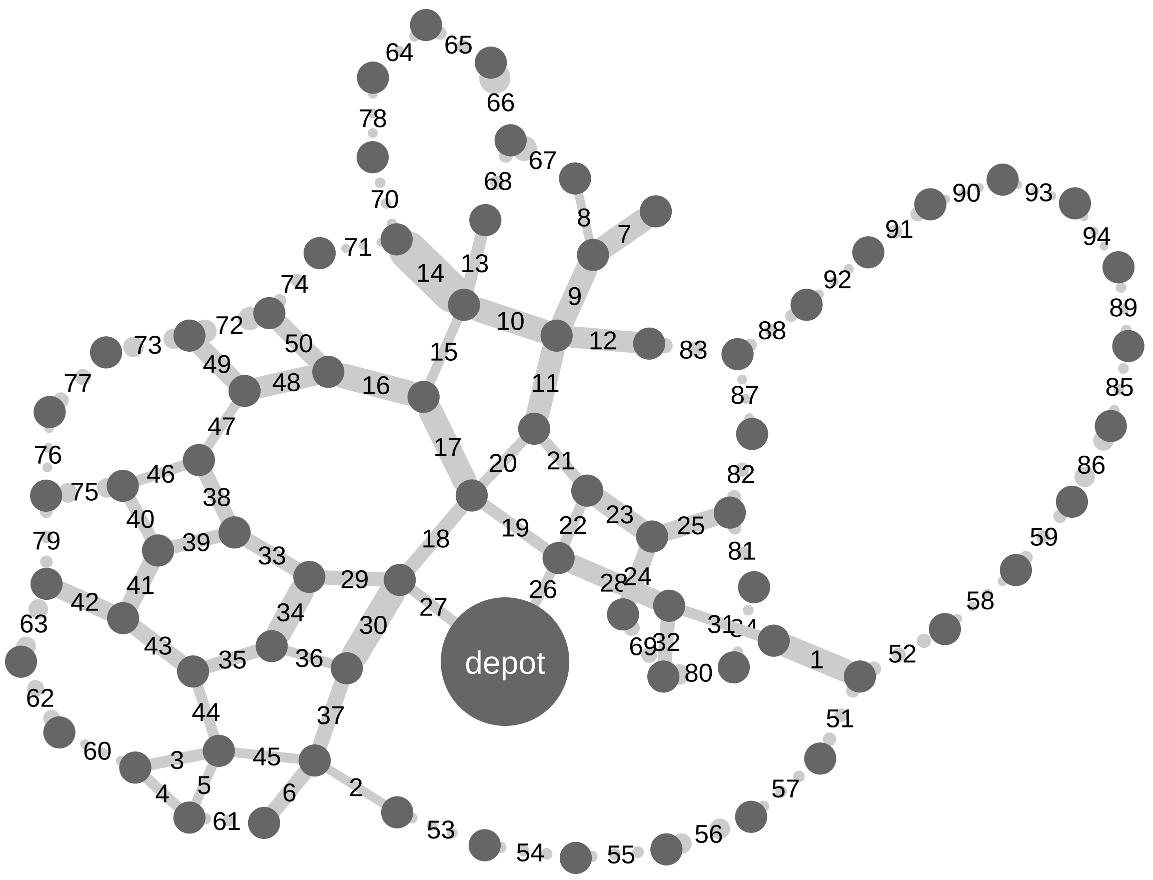


Figure 1.2: F07 CARP instance. The cost of serving each edge is proportional to its thickness. Non required tasks can be identified as dotted edges

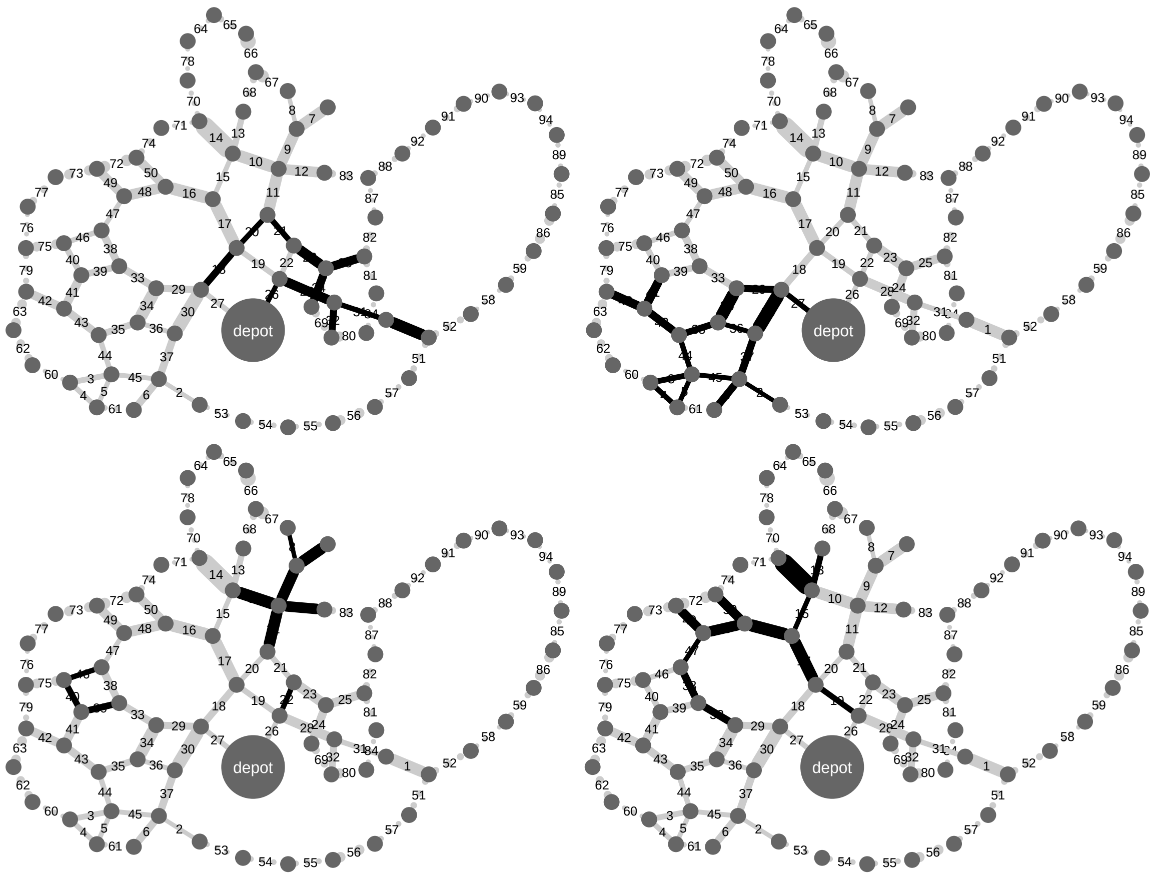


Figure 1.3: The four routes that compose a solution for the CARP instance in 1.2. The tasks served during each route have been highlighted in black

1.2 Research Questions

1.2.1 Adaptive Operator Selection for the Capacitated Arc Routing Problem

Adaptive Operator Selection (AOS) strategies embed into EAs a mechanism that is capable of selecting one operator out of a suite, based on the recorded evaluation of the past performances of such operators. Several studies have been conducted to prove the effectiveness of such approaches, mostly in the case of mutation operators, while a limited number of works focused on studying the case of crossover operators [67]. In most of the cases, moreover, such works show the effectiveness of AOS techniques performing experimental studies on considerably simple EAs and through the use of toy problems[36]. In the context of the Capacitated Arc Routing Problem literature, no crossover operator has emerged to be the most efficient, and those commonly used are adaptations from related routing problems (eg. TSP, CVRP). The Memetic Algorithm with Extended Neighbourhood Search (MAENS) algorithm [128] is a popular algorithm for CARP, and represents therefore a testing ground where new techniques can be implemented in order evaluate the successfulness of AOS techniques in a real-world context, to analyse their effect when they are integrated in a particularly elaborated meta-heuristic such as MAENS[128] and to provide further evidence relatively to the effectiveness of AOS techniques in the crossover scenario. The first research question is therefore:

> **RQ1**: Is it possible to adopt and/or to modify an AOS technique in the CARP domain with the purpose of performing the adaptive selection of the crossover operator within the MAENS algorithm?

A second aspect is relative to the type of information that it is possible to extract and use to influence the behaviour of the algorithm during its research. In particular, population diversity is often considered an important piece of information to maintain the balance between the exploration and the exploitation of population based meta-heuristic. The CARP literature lacked, at the time this work was carried out, of a rigorous and formal definition of a diversity measure, while less accurate approaches were considered [32]. The second research question is:

> **RQ2**: Is it possible to formalize a diversity measure between CARP solutions? How can this information be used to influence the search in MAENS?

1.2.2 Operator Selection Through Fitness Landscape Analysis and Online Learning

Most popular self-adaptive mechanisms for the identification of the best suitable variation operator in Evolutionary Algorithms rely almost exclusively on the measurement of the fitness of the offspring, which may not be sufficient to assess the optimality of an operator (e.g., in a landscape with an high degree of neutrality) [129], [22]. However, there is a great amount of information that can be extracted from CARP solutions, other than fitness, and that can be used to understand what operator can be the best in every moment of the search. A third research question is therefore:

> **RQ3**: What kind of additional information it is possible to provide to the Adaptive Operator Selection technique for a more "aware" calculation of the reward and does this information effectively help to improve the prediction ability of the algorithm?

The presence of a suite of measurements raises the question of adapting the existing techniques, based on the evaluation of only but one credit measure, to the case of selecting from a multitude of options. A further research question is therefore:

> **RQ4**: What technique would be useful to handle such data and to select the most suitable operator in such a dynamic environment?

1.2.3 Knowledge Incorporation Through Task Affinity Prediction

The use of Knowledge Incorporation techniques has effectively shown a capacity of enhancing the performances of the search algorithms they are applied to [65] and has led to the definition of novel meta-heuristic approaches [119]. Case Based Reasoning is one popular paradigm that has been been adopted by most of the works considering the reuse of domain knowledge across multiple instances carried out in the past decades [77] [79], which focuses mostly on the reuse of past solutions when extremely similar problem instances are encountered. The use of this strategy can lead to overfitting and decrease generality, as the technique is more likely to be effective only when the problem instances share specific characteristics. It is however reasonable to assume that good but partial information can be extracted from different cases and that by doing that it might be possible to increase the generality of the algorithm. In this sense, CARP would represent a good test case to develop Knowledge Incorporation techniques and to verify their validity. In particular, two ulterior research questions arising in this scenario are:

> **RQ5**: Existing approaches in the literature adopt a Case Based Reasoning strategy to determine the most useful piece of information for the current instance among the cases stored in memory. However, it would be interesting to test whether the use of alternative strategies such as Ensemble Learning methods, that consider learning across multiple instances, could be successful to increase the robustness of the predictions.

Existing approaches in literature are based on the use of a domain dependent distance measures to determine the most useful information for the current instance. This limits the reproducibility of the techniques among different problems. Therefore, a further research question is:

> **RQ6**: Is it possible to develop a system that does not make use of any elaborate and domain dependent measure, with the exception of simple features extracted from the data? What sort of features can be extracted from CARP instances?

1.3 Main Contributions

The main contributions of this thesis are detailed in chapters 3, 4, and 5. In particular, chapter 3:

1. Introduces four novel crossover operators for the Capacitated Arc Routing Problem, and extends the application of a state-of-the art Adaptive Operator Selection technique in the CARP research domain, which is used to dynamically select the crossover

operator during the execution of the algorithm. This is integrated in the existing CARP meta-heuristic MAENS;

2. Formalizes the definition of a diversity metric between CARP solutions that is used to compute an average pairwise distance within the population. The Stochastic Ranking Operator that performs the individual selection at the end of every iteration is then modified in order to perform its choice taking into account also such information;

3. Formalizes the adoption of a novel Reward Measure, called *Proportional Reward*, that is effectively used during the Adaptive Operator Selection task;

Chapter 4 extends the work carried out in the previous chapter. Moreover, it:

1. Illustrates a study over the applicability of Fitness Landscape Analysis techniques in the context of the Capacitated Arc Routing Problem, in order to identify those techniques that can be used to characterize its landscape, and that can be embedded in the MAENS meta-heuristic without incurring into extra computational cost;

2. Presents a novel Adaptive Operator Selection technique, based on the use of the aforementioned Fitness Landscape Techniques and the Dynamic Weighted Majority online learning algorithm;

3. Introduces a novel Reward Measure focusing on the diversity on the offspring generated by the crossover operators, named *Diversity Based Reward.*

Finally, the contributions of chapter 5 are:

1. A first study to determine a set of features that can be extracted from CARP instances, and that can be used to characterize them at different levels of granularity;

2. The definition and formalization of *Task Affinity* and of the *Affinity Matrix*, which are used to measure the likelihood for pairs of tasks to be part of the same route in sets of optimal solutions of a CARP instance;

3. A lifelong learning framework capable of learning traits from the previously encountered CARP instances in order to predict the *Affinity Matrix* of future CARP instances;

4. A Local Search operator that is used to improve the average *affinity* of individuals within the population.

1.3.1 Publications

The following papers where published during the course of the doctoral studies. They contain part of the material that has been included in this Thesis, along with further work that has been carried out on the adoption of AOS techniques in a different Combinatorial Optimization problem, named Software Project Scheduling Problem. Such papers are:

1. Consoli, P., & Yao, X. (2014, April). Diversity-driven selection of multiple crossover operators for the capacitated arc routing problem. In European Conference on Evolutionary Computation in Combinatorial Optimization (pp. 97-108). Springer Berlin Heidelberg;

2. Consoli, P. A., Minku, L. L., & Yao, X. (2014, December). Dynamic selection of evolutionary algorithm operators based on online learning and fitness landscape metrics. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 359-370). Springer International Publishing;

3. Consoli, P. A., Mei, Y., Minku, L. L., & Yao, X. (2016). Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis. Soft Computing, 20(10), 3889-3914.

4. Wu, X., Consoli, P., Minku, L., Ochoa, G., & Yao, X. (2016, September). An Evolutionary Hyper-heuristic for the Software Project Scheduling Problem. In International Conference on Parallel Problem Solving from Nature (pp. 37-47). Springer International Publishing.

1.4 Thesis Outline

The thesis is structured according to the following outline. Chapter 2 provides a literature review of the past and recent pieces of work in the research domains of Capacitated Arc Routing Problem, Adaptive Operator Selection and Knowledge Incorporation in Combinatorial Optimization. Chapter 3 focuses on the adoption of an Adaptive Operator Selection technique to perform the online selection from a set of crossover operators, implemented in the popular MAENS meta-heuristic for the CARP. This chapter includes the answers to research questions 1 and 2.

The work included in chapter 4, extends the one described in the previous chapter, by introducing the use of Fitness Landscape Analysis techniques and an Online Learning algorithm to influence and perform the crossover operator Selection, answering the research questions 3 and 4. Chapter 5 proposes a Knowledge Incorporation framework for CARP, that is used to learn from previously encountered instances in order to predict useful information that can be exploited to improve the performances when tackling future CARP instances. The chapter provides an answer to the research questions 5 and 6. Finally, chapter 6 summarizes the main findings of this thesis and the several directions that could be followed to extend this work.

CHAPTER 2

Literature Review

The validity of usefulness, adequacy of popular standards can be tested only by research that violates them.

Paul Karl Feyerabend
Science in a Free Society

2.1 Introduction

This chapter offers an overview of the literature relative to the Capacitated Arc Routing Problem (CARP), targeting in particular the analysis of what are the approaches that have been suggested in the past years and how such approaches can be integrated and adapted with some ideas from alternative research areas. In particular, the chapter will focus on several research direction. Firstly, it will provide an analysis of the most common and competitive techniques of Adaptive Operator Selection, and will discuss their possible use in the context of CARP. Secondly, it will discuss the possibility of integrating the use of Online Learning techniques and how these strategies could be used to detect changes in the behaviour of the operators in an online fashion. In addition, it will also provide an analysis of what alternative approaches exist in literature to identify and include different sources of information, alternative to the fitness, in the context of AOS. Finally, the existing approaches of Knowledge Incorporation for CARP are discussed in order to identify methods that can be adopted or extended for the CARP case.

The aim of this chapter is therefore that of highlighting gaps, drawbacks or unexplored areas of the existing CARP literature with the purpose of embedding more information in existing optimization algorithms to improve their approximation ability.

There is sufficient evidence in the literature that Parameter Tuning techniques contribute to the improvement of the performances of meta-heuristics [67] [15] and that this adaptive capability is indeed one of the most beneficial traits of population based approaches [114]. Although great contributions have been produced in the past, several direction are still being unexplored or scarcely followed. In [67], the authors identify some suggestions for further development in the Parameter Control area. Among these, they suggest to "carry out investigations about the combination of different control mechanisms and try to understand their joined effects". Moreover, they advocate that "relevant EA behaviour descriptors (sometimes called observables or monitors) can offer direct advantages for developing better parameter control mechanisms. In particular, it can help designing mechanisms for adaptive control, because these are based on using information (feedback) from the evolutionary search process". They also point out that "This research line could benefit from data mining techniques that disclose the correlations between various EA behaviour descriptors over time and the links between such descriptors and (the online dynamics of) solution quality".

In [15], the authors point out how "Hyper-heuristic research has the potential of bringing together promising ideas in the fields of meta-heuristics and machine learning, with knowledge (in the form of problem-specific heuristics) accumulated over the years in the field of operational research." Finally, Zhang et al. in [151] point out how the crossdisciplinary area between Evolutionary Computation and Machine Learning has focused mainly on single objective optimization problems, that the work in multi objective, dynamic and constrained problems is still insufficient and encourages the application of such techniques to numerous real world problems.

A further research area that is explored in this chapter is that of Knowledge Incorporation. In the preface of his book "Knowledge Incorporation in Evolutionary Computation" [65], Jin points out how although this research area has received increasing interest in the past years, most of the works in this area are "scattered in a wide range of research areas and a systematic handling of this important topic in evolutionary computation still lacks". To summarize, this chapter addresses three main research directions:

1. As previously mentioned, the adoption of Parameter Tuning techniques on complex problems and real world applications would be beneficial for the research area, as new algorithms and strategies may be derived through the process. The Capacitated Arc Routing Problem, a complex problem with strong real world connections, is therefore a good candidate for this research direction;

2. A further research idea follows the idea of studying and testing alternative approaches to Parameter Tuning, as well as the exploration and the analysis of novel descriptors to be integrated in the evolutionary process. The adoption of Online Learning algorithms to perform Adaptive Operator Selection, and the study of Fitness Landscape Analysis techniques provide a possible answer to this question;

3. The last research direction focuses on the promising and relatively new area of Knowledge Incorporation. The existing approaches are mainly based on the reuse of domain-related information and have consequently a limited generality. There is also a lack of approaches based on the use of Machine Learning techniques to learn

and predict the information that can be reused.

The rest of the chapter is structured as follows. Section 2.2 reviews the previous approaches for solving the Capacitated Arc Routing Problem, including exact methods, heuristics and meta-heuristics. The works discussed in this section are summarized to highlight their strengths and possible gaps in the literature. Section 2.3 introduces the research area of Adaptive Operator Selection and proceeds to review and discuss several of the most common approaches, with their advantages and disadvantages. It also produces evidence of the necessity of introducing new source of information and how a multitude of techniques can be dealt with by the use of Online Learning algorithms. Furthermore, 2.4 includes a review over the past and current Knowledge Incorporation techniques, with a focus on the ones that have been applied in the context of the Capacitated Arc Routing Problem. Finally, section 2.5 summarizes the research directions that have been identified in this chapter and how the work described in this thesis attempts to answer them.

2.2 Capacitated Arc Routing

The *Capacitated Arc Routing Problem* (CARP) is a NP-Hard Combinatorial Problem which has been the object of study and of interest of researchers for more than 30 years, because of its close relationship with real world problems. The problem was firstly formalized in a 1981 work by Golden and Wong [45]. A full definition of the problem has been included in section 1.1.3. CARP belongs to the wide and always expanding family of Vehicle Routing Problems, and shares several relationships with other well known problems. In particular, CARP is a constrained version of the Rural Postman Problem [106], which in turn, is a version of the Chinese Postman Problem limited to a subset of the vertices of the graph. CARP also shares a deep connection with the Capacitated Vehicle Routing Problem (CVRP), where tasks are represented by the vertices and not by the arcs. In fact, some of the existing approaches for CARP rely on the transformation of this problem into the CVRP[3], and the literature on CVRP [42] is itself still very relevant for CARP, as well as the work carried out on other routing problems, such as TSP. It is worth noting that the terms Arc Routing and Rural Postman are often used in the literature with the same meaning. Readers interested in more detailed surveys can refer to those of [53] and [140].

A considerable number of exact methods, heuristics and meta-heuristics has been proposed in the literature. For the former case it is possible to mention a few works proving lower bounds on known benchmark instances for CARP, obtained through greedy approaches as in [8], and OR techniques such as the cutting plane approach of [7], dynamic programming as in [5], [87], [6] and [138]. The bounds are computed on the most common benchmark datasets for CARP, which are *gdb* [25], *Val* [8], *egl* [28] which comprises two sets of instances, and *BMVC* [9] which includes the four sets of instances named *C,D,E,F*. The *egl* and *BMVC* are of particular interest as they represent real maps. Although dated, these benchmark are still being used in the CARP literature, although the attention of the researchers is now focusing on the largest instances (e.g the *egl* dataset).

A few approximation bounds are known for CARP. In particular, Frederickson' algorithm [37], which was proposed in 1979 for the Rural Postman Problem and that is based on Christophides's approximation algorithm for the Travelling Salesman Problem [19], provides a $\frac{3}{2}$ approximation bound. Analogously, Jansen proposed another approximation algorithm for the General Routing Problem having the same bound in 1992 [63]. The tightest approximation bound is that shown in a work from 2008 by Wøhlk [139], who presented a novel Approximation Algorithm for CARP having $\frac{7}{2} - \frac{3}{W}$ bound, where W corresponds to the vehicle capacity. Several variants of the problem have been proposed to reflect the presence of further constraints or dynamic scenarios. Mei et al. also proposed an approach for the multi objective CARP in [94], where the second objective consists of minimizing the cost of the longest route. Xing et al. [144] developed a hybrid algorithm based on the ACO meta-heuristic for the Extended CARP, which introduces several constraints, such as a maximum service time, penalties for turns and a variable amount and position of depots. A further work was carried out by Xing et al. [145], who proposed an evolutionary approach for the Multi-Depot CARP. CARPTW, conversely, is the version of the problem having specific time windows there the service of each task should start and finish, object of analysis of [118]. PCARP, on the other hand, is a periodic version of CARP where tasks need to be visited several times in different periods. A evolutionary based approach for this problem is that included in [95]. The reader can find an extended list of all the existing variants in [140].

2.2.1 Heuristics For CARP

Several heuristics were proposed for CARP in the past decades. Along with the classic constructive approaches such as Augment-Insert [108], Augment-Merge [45], Path-Scanning [44] it is possible to mention also post-optimization heuristics such as Ulusoy's splitting method [130] or those described in [54]. Although they are not among the most competitive strategies any more, these heuristics still prove to be extremely relevant in the modern research, as some of those are still being adopted in hybrid meta-heuristic as domain specific procedures [128], during the initialization phase or as post-optimization routines.

2.2.2 Meta-heuristics for CARP

The panorama of meta-heuristics and population based approaches for CARP is particularly rich. Early works are those of Lacomme et al., that proposed the use of a Genetic Algorithm [69], a Memetic Algorithm [70] as well as an ant-based approach [71]. Chu et al. proposed a scatter search in 2006 [20] while Beullens et al. proposed a Guided Local Search [9] in 2003 using a heuristic based on the evaluation of the potential mutations of each solution. Hertz and Mittaz [56] adopted a Variable Neighbourhood Search in 2001, which combines a set of novel mutation operators.

Other approaches suggesting the use of Simulated Annealing are those of [137] and [28]. Another popular heuristic for CARP is Tabu Search, which has been proposed in several occasions [12] [55], [48] and [92] and that continues to be relevant also in some of the most recent approaches [18]. Memetic Algorithms are among the most competitive techniques in the CARP literature. The first proposed approach by Lacomme in 2004[70], named LMA, was extended by Mei et al in 2009 [93]. The authors update the local search of the algorithm introducing a new cost function that is a combination of both fitness and constraint violation. In 2009, Tang et al. proposed a novel Memetic Algorithm [128] named MAENS, building on the work carried out previously. MAENS, which will be object of more in depth analysis in section 3.2, combines the use of the Sequence Based Crossover operator [113], with a novel Local Search operator and with the Stochastic Ranking selection operator [120]. MAENS's local search is in turn an iterative local search which combines the use of three different move operators, the novel cost function combining both fitness and violation introduced in [93], with a long-step operator called Merge-Split, which makes use of the classic heuristics Path-Scanning and Ulusoy's grand tour[130]. Improvements for MAENS were proposed by Fu et al. in 2010 [38] suggesting a strategy to select the most suitable routes that are modified during the Merge-Split step in Local Search, and by Chen [17], which proposed a decomposition strategy to reduce the computational cost of the Local Search. It is also worth mentioning the work by Feng et al., proposed in 2010 [32], which consists of a self-adaptive memetic algorithm influencing the local search probability. This work is also relevant as it is the only one, to the best of our knowledge, prior to the work carried out in this thesis, that defines a diversity measure between CARP instances. The measure adopted in this work, however, lacks formality and accuracy as it does not take into account the relationship of consecutiveness between tasks.

A different strategy was proposed to deal with Large Scale CARP instances, as the previous approaches would incur into a prohibitive computational cost. It is possible to mention the work by Mei et al. in [89] and [91], based on the use of the Cooperative Co-evolution framework [112].
Some recent and alternative approaches for CARP are those of Wang et al. [134] based on the development of a novel Local Search heuristic focusing on the reduction of the distance between tasks, the Clonal Selection approach by Shang et al. [123], and the extremely competitive hybrid approach by Chen et al. [18], integrating two different local search operators within a genetic algorithm., Finally, it is also worth mentioning in this section the work carried out by Feng et al. [31] on CARP, which however will be object of a more accurate analysis in section 2.4.

2.2.3 Discussion

There is a great body of work that has been carried out on CARP in the recent years, but this has mostly focused on the refinement of the previous algorithms, while approaches considering Parameter Setting (either online or offline) are lacking, with the exception the self-adaptive memetic algorithm of [32]. Another aspect is that relative to the Crossover Operators that are being adopted in CARP, which are mutated from the Vehicle Routing literature. There is no evidence of a study being conducted on CARP to propose alternative recombination mechanisms and how these adapt to different instances. As a consequence, it is reasonable to claim that there is no consensus on which crossover operator is the most successful for the CARP benchmark sets. In the interest of improving the approximation ability of existing meta-heuristics, it would be fruitful to perform a study to propose novel crossover operator techniques.
A further aspect is relative to the absence of a formal and accurate distance measure in CARP literature. This is undoubtedly an extremely useful piece of information that could

be integrated well in the context of an Adaptive Operator, as a performance indicator over the diversity of the population. Therefore, it would be beneficial for the CARP research area to define a formal distance metric between solutions, that can be effectively used to monitor and maintain the diversity of the population.

2.3 Parameter Setting and Adaptive Operator Selection

Parameter Setting is an important area of research in the Evolutionary Computation field. An a-priori identification of the optimal configuration of the parameters can be a time-consuming and impractical task. This is particularly true when such task is performed through classical iterative approaches. Alternative methods have been developed to deal with Automatic Parameter Configuration, also called Static Parameter Configuration, such as [10], [60], [152], [149] or [76] to name a few. Not only can these strategies reduce the amount of time required to find the best configuration, but they are also useful to perform a sensitivity analysis of the parameters, and, ultimately, to design algorithms with an increased level of generality. Another promising line of research follows the idea of predicting the performances of the algorithm when using different parameter settings through the use of learning strategies. Portfolio selection strategies, algorithm selection, and sometimes hyper-heuristics are terms that are often used in the literature to describe a similar scenario [46], [15], [39]. The analysis and the definition of novel predictive measures is in this case a fundamental aspect in order to be able to classify problem instances into classes associated to different hardness levels, and ultimately to choose the most appropriate strategy for each. Several approaches have consequently started investigating the landscape of the problem in order to recognize patterns and common traits [101], [84].

In a scenario where the amount of resources is limited, when the set of instances to be tackled is unknown or in a dynamic context, it is not possible to rely on static parameter configuration. The research area of Parameter Control focuses on the develoment of approaches based on the online selection of the best parameter configuration. Adaptation schemes have been applied successfully on multiple and heterogeneous components of the meta-heuristics adopting various learning strategies in countless occasions. In [29], an EA using Temporal Difference to modify the parameters on-the-fly is shown to outperform a benchmark EA on a rugged landscape. A similar approach is adopted on a on a (1+1)-evolution strategy in [100]. The authors stress the importance of choosing a proper reward measure for the successful application of the adaptive system. In [121] reinforcement learning is used to select among a suite of search operators, and the reward is computed across the results achieved by the whole population. In [142] and [102] parameter adaptation techniques are used to maintain the population diversity. In [126] the authors study the effect of offline and online information in the context of the Quadratic Assignment Problem. Dynamic schemes have also been adopted to determine what is the best algorithm to use among a suite of possible choices, as in [110].

The problem of identifying the most suitable variation operator among several, also known as Adaptive Operator Selection (AOS), occupies an undoubtedly important place in the Parameter Setting literature [66], [58]. Adaptive Operator Selection can be divided into two sub-tasks: the Credit Assignment (CA) mechanism, used to evaluate the performance of the operators, and the Operator Selection (OS) Rule, necessary to determine the most suitable operator using the information provided by the CA mechanism.

The majority of the Credit Assignment approaches in the literature are based on the evaluation of the fitness of the offspring generated by the operator, which is compared either to the current best solution [24], to the median fitness [66] or to the parents' fitness [4]. A different strategy evaluating both fitness and diversity of the offspring was proposed in [88]. The reward has been mostly considered as the value assessed during the last evaluation (*Instantaneous* reward), as the average reward over a window of the last N evaluations (*Average* reward), and as the biggest improvement achieved over a window of the last N evaluations (*Extreme* reward)[35], [24], [4] and [135].

The Credit Assignment mechanism is coupled with Operator Selection rules such as Probability Matching [43], Adaptive Pursuit [129] or Multi Armed Bandit solvers (MAB) [23]. The Operator Selection scenario can, in fact, be seen as a dynamic version of the game theory Multi Armed Bandit problem. In this case, each operator represents an arm with an unknown reward probability. The objective is therefore that of selecting during each time-step the arm that maximises the probability of obtaining the highest reward. MAB solvers for the AOS case make use of the Upper Confidence Bound rule [2], combined with mechanisms to detect changes in the performances of the operators. Fialho et al. suggest in [22] the use of the Page-Hinckley test [57]. Several experiments have shown how MAB-based approaches outperform the former techniques. The technique has been adopted mostly to choose among mutation operators and is tested in simple problems, such as, Royal Road and Long K-Path Problem. A further variation considers the use of a sliding window of the last n recorded performances of each operator, to limit the calculation of the reward measure to such values [36].

In [47] the dynamic Multi Armed Bandit technique is adapted to numerical optimization and used alongside a Differential Evolution algorithm. More recently, the use of this technique has been extended to the multi-objective case, as in [74]. Among the most recent contributions, [85] discusses the adoption of a hyper-heuristic in the context of the Capacitated Vehicle Routing Problem.

Arguably, Multi Armed Bandit solvers represent the predominating strategy in AOS. The success of such techniques might be explained by the innovation introduced with mechanisms that are being used to detect abrupt changes of the performances of the operators during the search. In fact, it is reasonable to expect that the best operator during a specific period of the search might not be the best afterwards. The use of techniques such as the Page Hinkley [57] statistical test for change point detection, or the use of sliding windows have been in fact introduced to limit the influence of the past performances of the operator in a much refined way. There are however a great number of strategies that can be adopted from different research areas, such as Online Learning, that could be successfully used for the purpose. In fact, there is a strong analogy between the notion of Concept Drift in Online Learning and the detection of changes in the best operator.

Online Learning approaches tackle Concept Drift in two different manners. In the first case, algorithms such as [103] or [41] perform a test to determine the presence of concept drift. When Concept Drift is detected, the learner is discarded and replaced by a new one. This approach has the disadvantage of not considering the information prior to the detection of the Concept Drift, which might still be relevant to a certain extent. In the second case, the system is composed by an ensamble of learners, which can be discarded when their prediction performance is poor and replaced. In this case, the detection of concept drift happens more gradually as the system requires more time to react to it, by discarding the old learners and replacing them with new ones. Examples of these techniques are the Dynamic Weighted Majority algorithm [68] or Concept Drift Committee [125]. The reader can find a detailed review of the state of the art in Online Learning algorithms in presence of Concept Drift in [98].

2.3.1 Alternative Measures to Fitness

As mentioned previously, one of the objectives of this thesis is to investigate the adoption of measures alternative to the fitness of the offspring. By introducing new and different information in the system, in fact, the algorithm can improve its ability to read different contexts where one particular measure might fail or not be useful. An example is that of a landscape with a great degree of neutrality, where the use of a measure based on the fitness of the individuals might actually be ineffective.

The adoption of alternative measures has recently been the object of investigation by some of the works in the literature. The introduction and the integration of multiple types of information as Credit Reward measures is one of the novelty introduced by *Compass* technique [88], which evaluates the performance of each operator combining both the fitness of its offspring and its diversity.

Beyond the classic approaches based on fitness or diversity, there is a great deal of information that can still be extracted and taken into account, through the use of Fitness Landscape Analysis (FLA) techniques. FLA techniques have been traditionally designed and adopted to characterize the hardness of problems. The reader can refer to the survey of Malan and Engelbrecht [83] for a complete analysis of FLA techniques and taxonomy. Only recently, several authors have started shifting the scope of these techniques towards different directions, such as that of investigating the relationship between algorithmic performances and problem characteristics [83].

Vanneschi in [131] makes use of the Negative Slope Coefficient technique to perform an offline optimization of the most effective Genetic Programming configuration. Ochoa et al. [104], the authors analyze the neutrality and the convexity of the landscape to improve the design of a constructive hyper-heuristic for Timetabling Problems. Another example is that of the Dispersion Metric technique of [82], which was designed to extract some information that can help to gain a better understanding of the behaviour of the CMA-ES heuristic. The study conducted in [109] analyzes the relationship between parameter configurations and performances of hybrid meta-heuristics using NK landscapes. The authors of [146] perform a Fitness Landscape Analysis of the Delay Constrained Least-Cost Multicast Routing Problem, using fitness distance correlation analysis and autocorrelation analysis. The results of such analysis are used to design a novel Iterated

Local Search Approach for the aforementioned problem. The authors of [124] discuss the adoption of an evolvability metric [132] to replace fitness evaluation in the context of Adaptive Operator Selection on the Onemax, Royal Staircase and Multiple Knapsack problems. More recently, the authors of [61] predicted the best strategies for the Graph Coloring Problem using data mining techniques.

Selecting the right Fitness Landscape Analysis technique, for the right problem and during a specific moment of the search, can be a difficult trivial task. Problems are known to respond to the various Fitness Landscape Analysis techniques in different ways. Moreover, it has been proved that there is no such thing as a Fitness Lanscape Technique that can be used universally to characterize every NP-Hard problem difficulty [52]. None of the studies previously mentioned considers the use of a multiple set of measures. Including more and different types of information does not only provide ulterior knowledge, but can also be a good strategy to improve the adaptability of the AOS system to different scenarios. Moreover, the majority of such studies focuses on the investigation of the relationship between algorithmic performances and landscape characteristics in an offline manner. This excludes all the dynamics that are only present in online scenarios. For instance, different areas of the landscapes that are explored during diverse moments of the search might respond differently to various evolutionary operators. In a case like that, multiple types of information might be more suitable in the different periods of the search.

Finally, there are very few empirical studies relative to the application of such techniques to real world problems. In particular, none of such studies focuses of the analysis of the landscape of the Capacitated Arc Routing Problem and how such information can improve the approximation ability of the algorithms designed to tackle this particular problem.

2.3.2 Discussion

The analysis of the literature reveals that most of the research in Adaptive Operator Selection investigates mainly the selection of mutation operators. Works focusing on the use of AOS techniques choosing crossover operators from an ensemble are therefore scarce and only limited to toy problems and / or considering simple evolutionary algorithms. A second aspect that is worth mentioning is relative to the fact that the large majority of Credit Assignment techniques rely on the evaluation of the fitness of the offspring to measure the performances of the operators, with a few exceptions such as [88]. However, recent works have provided strong indications that Fitness Landscape Analysis techniques can be beneficial for the offline design of algorithms. The introduction of new sources of information can potentially endow the algorithm with more indications to detect the best operators also in an online fashion.

It is also worth noting that the crossdisciplinary research area between Machine Learning and Evolutionary Algorithms is mostly oriented towards the use of EAs for learning purposes. Approaches considering the use of Machine Learning algorithms to aid Evolutionary Algorithms in their optimization tasks are much less frequent, and mostly adopted to perform an offline analysis. Hybridizing ML as Evolutionary Operators within metaheuristics is a path not frequently taken and that can be reasonably expected to lead to good results [151]. Finally, it is possible to notice that there is a lack of works focusing on the use of Adaptive techniques on complex combinatorial optimization problems such as routing problems, either in Parameter Setting or in the Hyper-heuristic domain [16], [15].

2.4 Knowledge Incorporation

In a book from 2005 collecting works on this topic [65], Jin gave a broad definition of Knowledge Incorporation in the context of Evolutionary Computation as the "Incorporation of a priori knowledge, such as expert knowledge, meta-heuristics and human preferences, as well as domain knowledge acquired during evolutionary search, into evolutionary algorithms". Despite representing a promising direction, Knowledge Incorporation is as a matter of fact no more than an umbrella term that incorporates and embraces pieces of works belonging to different research areas and that lacks a formal recognition as a proper research area. Alternative terms that are often used in the literature to represent a similar concept are Knowledge Reuse, Memory Inoculation, Lifelong Learning Systems or memory-based systems. Jin [65] categorizes several pieces of works into five groups, based on the purpose of the Knowledge Incorporation techniques. In particular, he distinguishes between techniques incorporating knowledge in representation, initialization, mutation or crossover operators, techniques incorporating knowledge in selection and reproduction processes, or in the fitness function. Moreover, he distinguishes also approaches based on human-computer interaction, (e.g. Interactive Evolutionary Computation) and those works integrating human-preferences in multi-objective optimization. The reader can find several examples belonging to each of these categories are in [65].

Knowledge Incorporation techniques can also be characterized by distinguishing the scope and visibility of the shared information. In it is possible to distinguish between:

1. Knowledge Incorporation whose scope lies within the search on a single Problem Instance (e.g. Memetic or Cultural Approaches);

2. Knowledge Incorporation across different instances of the same optimization problem (e.g. techniques based on Intra Domain Knowledge Incorporation);

3. Knowledge Incorporation between instances of different problems (e.g. techniques considering Inter Domain Knowledge Incorporation).

This classification is transversal with the categorization introduced by Jin, as it focuses more on the visibility of the pieces of knowledge rather than the specific algorithmic feature they are applied to.

2.4.1 Knowledge Incorporation Within the Search

To the first category belong all those approaches using nuggets of information that is being shared through the rest the search. This information is often domain knowledge, either provided offline through a prior analysis, or through human-computer interaction in the shape of expert knowledge or user preferences, or in an online fashion through the use of domain specific heuristics. A first example is the Memetic Algorithm meta-heuristic [99]. Memetic algorithms are, in fact, designed to combine a Genetic Algorithm with a domain expert heuristic which generates *memes* of information that are shared through the rest of the search. Analogously, all those hybrid approaches combining an evolutionary algorithm and a domain expert heuristic, such as those reviewed in [11], can be considered part of this category.

Cultural Algorithms [119], on the other hand, have been designed to maintain a separated memory, called *Belief Space*, where favourable traits are stored, along with mechanisms that allow the current population to access it and to integrate it. A further example is constituted by the algorithms using partial restart strategies [86]. In this case, a partial re-initialization of the population is triggered when premature convergence is detected. A large majority of the population is then discarded and replaced by novel individuals, but few of those are kept from those belonging to the previous generation, in order to share the favourable traits within the new population.

2.4.2 Intra Domain Knowledge Incorporation

The works belonging to this category are based on the assumption that some of the information that has been obtained through the application of an optimization algorithm on a specific problem instance can still be relevant for other instances. This assumption can be considered valid when there are instances with extremely similar characteristics.

However, fragments of information, or partial solution, might be possibly shared even between problem instances with considerably different traits.

The earliest works in this area stem from the context of dynamic optimization. In fact, it is immediate to see a strong analogy between two different discrete times of a dynamic optimization problem and two problem instances of static optimization which share specific characteristics, such as their problem size, or overlapping topology (in the case of routing problem).

Ramsey and Grefenstette proposed in 1993 [116] a Case-Based Reasoning (CBR) approach for the *Anytime Learning* (Dynamic Optimization) scenario.

The CBR strategy aims at solving new problems by means of stored solutions of past similar problems. In order to accomplish this, it is necessary to a) store such solutions in a dedicated memory, and b) to provide the system with a similarity measure that is capable of identifying which one is the problem encountered in the past that is the closest to the current one. CBR was also adopted to model and explain the transmission of knowledge between similar solutions in Genetic Algorithms in [80], and later in [117]. Since its adoption in the early 90's the CBR has become, de facto, the standard technique adopted in this research field.

The idea of sharing the knowledge between different problem instances is discussed in [75]. In this initial work, as well as in his following, the authors adopts a CBR strategy to maintain a memory of past solutions that are injected during the initialization phase of the algorithm when a *similar* instance is tackled. The technique is applied for different combinatorial problems, such as the Traveling Salesman Problem, Combinatorial Logic Circuits, Strike Force Asset Allocation and Job Shop Scheduling. [79]. In each of these works, the results show that inoculating solutions during the initialization phase, or during the search, leads to better solutions than a GA with random initialization. The inoculation of past solutions during the initialization has also been proven to be effective in dynamic contexts, as in [148], in learning scenarios, as in [64], [27].

Several limitations arise from the use of the CBR. Most of the works existing in the literature investigating this approach ultimately draw the same conclusions, confirming in particular that:

1. Identifying a similarity measure between problem instances is not an easy task. Arguably, similarity between instances depends on the solution representation and ultimately on the problem definition. The majority of the works adopting the CBR approach rely on the solution similarity to measure how different two problems can be. As a consequence, only problem instances having the same size are considered in this case;

2. To prove the effectiveness of the algorithm, the benchmark set are artificially created in order to obtain instances with the same size and with extremely similar structures. Thus, the applicability of this technique is strongly limited to those very few cases where extremely similar instances (both in terms of their structure and in problem size) exist in the benchmark set [1];

3. Several works confirm that the adoption of this technique is beneficial only when there is a strong similarity between problem instances [127], and that its usage can lead to premature convergence [111] ;

4. Since it is not easy to adapt such technique to the existing benchmark sets, experimental comparisons are carried out against non state-of-the-art EAs. It is not clear whether its adoption would improve the optimization ability of much more complex and performing approaches.

In a much more recent work [33], Feng et al. introduced a new technique based on the concept of *memes* of information, applied to the Capacitated Arc Routing Problem (CARP) and Capacitated Vehicle Routing Problem (CVRP) cases. Here a meme is designed by the authors to be a transformation matrix M which induces a clustering of the tasks of a problem instance into specific groups, by transforming their distance to match the one found in an optimal solution during the previous execution of a EA. A meme is created therefore by matching the distance matrix of a problem with a matrix showing the clustering of the tasks in an optimal solution. The meme M is obtained by maximizing the dependency between the two matrices by means of the Hilbert-Schmidt Independence Criterion (HSIC) [49]. Memes are then stored in a pool, called *Society of Memes*. The second phase of the algorithm is the *meme selection*, which deals with selecting the memes from the memory that better fit the new problem instance. This process is carried out measuring the statistical dependence between meems and the distance matrix of the new problem instance, again through the use of the HSIC criterion. A final meme M_t is obtained as a linear combination of selected memes, having weights proportional to the calculated statistical dependence. The final step of the algorithm requires the application of the meme to the new instance and the clustering of the tasks into the routes induced by it. In order to verify the validity of this technique, the paradigm was applied to the ILMA [93] algorithm with improved optimization performance.

This work is extremely relevant for several reasons. Firstly, it overcomes the problem of defining a domain related distance measure by use of the domain independent HSIC dependency criterion. Secondly, this technique allows the sharing of information between instances having different characteristics and introduces the possibility of multiple pieces of information contributing to the next instance. Finally, it is probably the first work producing evidence in favour of the use of Knowledge Incorporation to improve existing and consolidated meta-heuristics.

The different works carried out in this area show some common traits.

- The whole body of work carried out in this context has been focusing exclusively to improve the solution quality during the initialization phase, by inoculating solutions stored or extracted from memory. Early attempts have shown that the impact of this technique can be modest [127] since the improvement deteriorates in a few generations because of the work of the evolutionary operators that drive the search towards better areas. Further experiments carried out in more recent works confirm that operating the inoculation during the execution of the algorithm can improve the performances [78]. This is particularly evident when such a technique is applied in hybrid meta-heuristics that combine a population based search with domain expert heuristics. No study has explored the idea of knowledge reuse in any of the several phases of the meta-heuristics (selection, mutation or crossover operators, local search);

- Incorporating prior knowledge raises concerns relative to the risk of premature convergences, and requires the introduction of further mechanisms to balance the exploration ability of the meta-heuristic. The majority of the works report the existence of such scenario, and adopt simple strategies such as limiting the amount of inoculated solutions to a certain percentage of the population. None of these studies proposed a mechanism to mitigate and control the effect of the knowledge reuse on the population diversity;

- The Case Based Reasoning approach does not take into account the possibility of having multiple sources of information that can share knowledge with new instances. With the exception of the work by Feng et al. [33], all approaches in the literature introduce solutions from a single source of information, following the CBR paradigm;

- As repeatedly noted by several authors, inoculating solutions in the initial population works under the premise that the current and the past problem, from which the solution has been extracted, are considerably similar. In other words, it can be concluded that CBR does not generalize well, and when in presence of a considerably different instance, there is a concrete risk of introducing useless information that might delay the optimization ability of the algorithm. It is reasonable to expect that the information should be shared with the new instance only when a certain degree of a similarity has been reached. No study has examined this aspect;

- The results of the algorithm depend on the sequence of instances presented to the algorithm. No study has examined the possibility of introducing a thorough mechanism to analyze the robustness of the results achieved, regardless of the order of the sequences;

- The work of [33] is based on the use of the distance between tasks in CARP instances to generate the transformation matrix. The association between tasks is taken from an optimal solution extracted from an evolved population. This strategy is however ignoring the great amount of information that is contained in an evolved population. This choice can potentially lead to premature convergence. Depending on the structure of the landscape, the final population can have several optimal solutions completely different from each other. The information contained in close to optimal solutions can also still be relevant. Thus, it is reasonable to claim that the information extracted from evolved population should include and could benefit from much more information;

- The transformation matrix of [33] is based on the correlation between task distance and the clustering of an optimal solution. Although this correlation definitely exists, it is extremely unlikely that it can, alone, explain completely the final routing of an optimal solution, and it is most certainly not the only information that can be used. Expanding the system to include more than one type of measurement can contribute to improve its performances by defining a more general and robust system that can be more easily adapted to other domains;

- Finally, it is worth mentioning how Machine Learning as well as Dynamic Optimization are very relevant for this research area, and works such as [13] or [147] can in fact contribute notably to the development of novel approaches. Studies in this area should be looking at these fields in order to draw inspiration and new ideas.

2.4.3 Inter Domain Knowledge Incorporation

Following a very recent and promising trend, the research is focusing on the sharing of the information across different domains (e.g. different combinatorial problems). Feng et al. extended the approach developed on CARP and CVRP, introduced in the previous section, by developing a mechanism to transfer *memes* between the two different domains [31]. The authors also show the benefit of the adoption of this technique in a real world scenario, the Package Collection/Delivery Problem, proving the effectiveness of the application of such technique also in a stochastic and dynamic context. Gupta et al. [50] recently proposed a new paradigm called *Evolutionary Multitasking* which tackles the inter-domain knowledge incorporation task by optimizing a set of different NP-Hard problems at the same time. The work, inspired by the *multifactorial inheritance* mechanism that regulates the transmission of traits to the offspring, shows that the indirect interaction between a set of problems being optimized at the same time is beneficial, as good traits manage to transfer across different domains. This is shown by realizing an algorithm that was effective when tackling sets of Continuous Optimization Problems, Discrete Problems, as well as a mixture of Continuous and Discrete problems.

2.4.4 Discussion

Not much work has been carried out to investigate the use of Knowledge Incorporation in the context of Evolutionary Computation. The difficulty of tracing work on this topic is also due to the fact that there is not a formal recognition for this research topic, as it happens for other cases. As a consequence, most of the works that could be relevant for this case are not immediately accessible as they are scattered in different areas.

In order to produce general and robust systems, it is important to design lifelong learning mechanism that are as general as possible, consider multiple sources of information and can be easily adapted to different domains. The vast majority of the works conducted on Knowledge Incorporation in the intra-domain context relies heavily on the use of the Case-Based-Reasoning (CBR) paradigm for problem solving. Although effective, CBR has several limitations, including poor generalization capability, which is however a fundamental requirement when designing a lifelong learning system. It also requires the definition of a domain dependent similarity measure, and limits the application of the technique to the cases where instances share specific characteristics (e.g. problem size). With the exception of the recent work of Feng et al. [33], no work has been exploring the possibility of learning from more than one of the several sources stored in memory. It would appear to be beneficial to go beyond the CBR paradigm to explore and integrate concepts and ideas from different areas such as Machine Learning.

Introducing more and heterogeneous sources of information, which is one of the main ideas that motivate this thesis, can contribute to increase the generality of the system and to improve its capacity of adaptation when facing a completely new instance. Therefore, in the context of the Capacitated Arc Routing Problem, it appears to be beneficial to extend the considerations made in [33] by considering the optimal structures found in the whole population, and by extracting different types of information from the problem instances other than distances between tasks.

Finally, a different aspect is relative to the purpose of the knowledge reuse. Literature lacks works investigating the knowledge reuse in contexts other than the initialization phase, where other components of meta-heuristics (e.g. mutation, crossover or selection operators) could be easily integrated with such information.

2.5 Summary

This thesis addressed the problem of improving the approximation ability of state-of-the-art meta-heuristics for the Capacitated Arc Routing Problem, by means of the use of adaptive systems that learn and perform decisions based on the use of multiple and heterogeneous sources of information. The chapter provides an overview of three different research areas that have been investigated during the course of this thesis, and where the ideas that motivate this thesis could be applied.

The review of the literature has shown that the amount of research that has been conducted for the case study of this thesis, the Capacitated Arc Routing Problem (CARP), has been focusing mostly on the development of novel meta-heuristics and domain related heuristics. Not many contributions have been provided to investigate the use of alternative crossover operators for this problem and their impact, in a single operator scenario, or when used simultaneously in an adaptive fashion. Very few works have been investigating the importance of balancing the diversity of the population in CARP meta-heuristics. Chapter 3 illustrates the introduction of four novel crossover operators for CARP, the definition of a formal diversity metric between CARP solutions and provides the first application of an Adaptive Operator Selection (AOS) technique for Crossover Operators in the context of CARP, and, to the best of our knowledge, one of the few in the general area of Combinatorial Optimization.

In the context of Parameter Setting, and more specifically Adaptive Operator Selection (AOS), the literature focuses mainly on the evaluation of the fitness of the offspring generated by each operator, when choosing a way to evaluate their performances. Not many works have been focusing on adopting an AOS strategy to select from a suite of Crossover Operators. On top of that, traditional approaches do not consider the possibility of performing the choice based on more than one reward mechanisms. Chapter 4 tries to contribute to the field by introducing a novel Adaptive Operator Selection mechanism based on the use of a set of measures, adapted from the Fitness Landscape Analysis area, and on the use of an Online Learning algorithm, to perform a selection based on the previously mentioned set of measures.

The analysis of the background research in the area of Knowledge Incorporation, provides a novel classification of the techniques belonging to this area, based on the scope and visibility of the information that is being shared. The techniques belonging to the first group are those algorithms that share information throughout the search, either by the use of domain expert heuristics or by using explicit ad-hoc structures to maintain a memory of the most relevant traits encountered so far. In the second group, the knowledge is shared between different instances of the same problem. The dominating trend is, in this case, the use of the Case Based Reasoning (CBR) paradigm, where any new problem instance encountered is injected with information coming from the most similar problem instance that has been solved in the past. it has been shown that the adoption of the CBR strategy to inoculate knowledge in evolutionary algorithms has several limitations. In fact, (a) its applicability is limited only between those instances sharing specific characteristics. (b) it does not generalize well and (c) requires the challenging definition of a similarity measure between instances. It has also been shown how the totality of the techniques in this area investigate only the possibility of sharing the information by mean of inoculating information during the initialization phase. Chapter 4 introduces a novel learning system that (a) replaces the use of CBR with an Ensemble Learning method, based on the definition of (b) a set of 34 problem features that can be used to understand and (c) explain the solution quality extracted from an evolved population of solutions.

CHAPTER 3

Diversity-Driven Selection of Multiple Crossover Operators

The art of progress is to preserve order amid change and to preserve change amid order.

Alfred North Whitehead
An Essay on Cosmogology

3.1 Introduction

For NP-Hard problems strongly connected with real world applications such as the Capacitated Arc Routing Problem (CARP), it is important that algorithms that we design are reliable enough to return good solutions, when not optimal, in as many cases as possible. However, as we have seen in the previous chapter, the work on the reinforcement of the average results of the algorithms for CARP has been lacking. The aim of this chapter is that of improving the approximation ability of a state-of-the-art algorithm for CARP through the use of the following techniques:

- the use of a novel distance measure between CARP solutions;
- a diversity-driven stochastic ranking operator;
- a set of new recombination operators for the problem;
    - Greedy Sequence Based Crossover (GSBX);
    - Greedy Route Crossover (GRX);
    - Pivot Based Crossover (PBX);
    - Shortest Path Based Crossover (SPBX).
- an Adaptive Operator Selection strategy which uses a new Credit Assignment technique based on the aforementioned ranking operator.

The rest of the chapter is organized as follows. Section 3.2 introduces the MAENS algorithm, a popular and competitive algorithm for CARP which represents the case study for this work. Section 3.3 presents a novel diversity measure for CARP solutions and a diversity-driven ranking operator. Section 3.4 introduces a suite of crossover operators for the problem and the Adaptive Operator Selection technique adopted. Section 3.5 shows the performance of the MAENS algorithm when using the proposed techniques.

3.2 MAENS

The Memetic Algorithm with Extended Neighbourhood Search (MAENS) [128] is one of the most competitive and efficient algorithms in the context of the CARP. Its pseudo-code is provided in figure 3.4. It is possible to divide it into four phases: initialization (lines 1-7), crossover (lines 10-11), local search (line 14) and stochastic ranking (line 26).

During the initialization phase, solutions are generated through the use of the Path-Scanning procedure [45]. The pseudo-code of the Path-Scanning procedure is provided in algorithm 3.1. A solution is generated by iteratively adding unserviced tasks to an empty route. Once the route cannot have any more tasks without violating the capacity constraint, it is added to the new solution (lines 3-7). The tasks to include in the route are selected based on a set of rules. If there is only one task that can be included in the route without violating the capacity constraint, this is included in the route (line 10). If more than one tasks satisfy this condition, the algorithm uses one of the five rules shown in lines 13-18 to select the next task to include. This process is repeated until all tasks have been included in a route. The procedure generates in this way 5 different solutions (according to the different rules). The solution with the smallest total service cost is then returned.

In the main cycle, couples of parent solutions are randomly selected to generate an offspring using the Sequence Based Crossover (SBX) operator [113]. The pseudo-code of the SBX operator is provided in algorithm 3.2. Two routes are randomly extracted by each parent solution and randomly split in two parts (lines 1-6). Two parts coming from different routes are combined and inserted in the new solution (line 7), which is then repaired in case of repeated or unserviced tasks.

A local search is therefore performed on the neighbourhood of the solution, with a probability *lsprob*, using three move operators, namely Single Insertion (one task is moved to another route), Double Insertion (two consecutive tasks are moved to another route) and Swap (two tasks of different routes exchange their places). The best solution is then improved through the Merge-Split operator, which applies the Path Scanning procedure [45] and the Ulusoy Splitting tour [130] to the tasks of two randomly selected routes. The local search procedure is completed by another iterative search using the three move operators. The pseudo-code of the Merge-Split operator is provided in algorithm 9. The behaviour of the Merge-Split operator is the following. A number of p routes are

Algorithm 3.1: Path Scanning Heuristic

```
1  Initialize a empty solution;
2  $S$ = set of required tasks;
3  while $S$ is not empty do
4      Initialize an empty Route $R$;
5      while $Capacity(R) < C$ do
6          $S_c$ = unserviced tasks that do not violate the capacity constraint if added to
             $R$;
7          if $|S_c| = 0$ then
8              add $R$ to the new solution and initialize a new route;
9          end
10         else if $|S_c| = 1$ then
11             add $t \in S_c$ to $R$;
12         end
13         else
14             Find the unserviced tasks that minimize one of the following rules;
15             1) maximize the distance from the head of task to the depot;
16             2) minimize the distance from the head of task to the depot;
17             3) maximize $dem(t)/sc(t)$;
18             4) minimize $dem(t)/sc(t)$;
19             5) use rule 1) if the vehicle is less than half full, otherwise use rule 2).;
20         end
21     end
22 end
23 Return the solution among the 5 generated that has the minimum cost.
```

Algorithm 3.2: SBX Crossover

```
1  $S_1$ and $S_2$ parent solutions;
2  $S_{\text{new}} = S_1$ ;
3  randomly extract $r_1$ from $S_1$;
4  randomly extract $r_2$ from $S_2$;
5  randomly split $r_1$ in $r_{1a}$ and $r_{1b}$;
6  randomly split $r_2$ in $r_{2a}$ and $r_{2b}$;
7  combine $r_{1a}$ and $r_{2b}$ and replace $r_1$ in $S_{\text{new}}$ with the new route;
8  repair $S_{\text{new}}$ if there are missing or repeated tasks;
9  return $S_{\text{new}}$;
```

randomly extracted from the initial solution S, and removed from it (line 1). A set of five new solutions are generated using the Path Scanning procedure (line 2). The solution with the smallest cost undergoes the Ulusoy Splitting tour, which is capable of finding the optimal splitting into routes based on the initial sequence of tasks provided by the solution (line 3). This procedure is repeated as long as the new solution has a smaller service cost than the initial one (cycle 4-8).

Algorithm 3.3: Merge Split Operator

```
1 Randomly select $p$ routes from $S$ and remove them from $S$;
2 $S_{ps}$ = PathScanning($S$);
3 $S_u$ = Ulusoy($S_{ps}$);
4 while $sc(S_u) < sc(S)$ do
5     replace $S$ with $S_u$;
6     Randomly select $p$ routes from $S$ and remove them from $S$;
7     $S_{ps}$ = PathScanning($S$);
8     $S_u$ = Ulusoy($S_{ps}$);
9 end
```

The offspring are then compared to the solutions of the previous generation and sorted using the Stochastic Ranking procedure [120].

3.2.1 Analysis of the algorithm

The algorithmic features of MAENS [128] were analyzed in order to identify what might be the possible drawbacks that affect the robustness of its results. As a memetic algorithm, it is equipped with a local search operator which greatly improves its capacity of exploiting the good solutions found during the search. This exploitation ability might not be balanced enough by an efficient exploration as the algorithm does not have any mechanism that maintains the diversity of the population. The algorithm makes use of a single heuristic, the Path-Scanning [45], in both the initialization phase and within the local search. This might affect the ability of the algorithm to generate new routes during

Algorithm 3.4: MAENS Algorithm

```
1  Initialise a population pop of size equal to *psize* ;
2  while (pop not full OR attempts < ubtrial) do
3      generate a new individual p;
4      if (p is not a clone) then
5          add p to pop;
6      end
7  end
8  while ($t < G_m$) do
9      for ($i = 0$ to psize) do
10         Randomly select $p_1$, and $p_2$ from pop$^t$;
11         Generate $s_x^i$ = SBX($p_1$, $p_2$);
12         Extract a random value $r$;
13         if ($r < P_{ls}$) then
14             generate $s_{ls}^i$ = LocalSearch($s_x^i, p$);
15             else
16                 $s_{ls}^i = s_x^i$;
17             end
18         end
19         if ($s_{ls}^i$ is a clone of a parent) then
20             replace parent in pop$^t$ with $s_{ls}^i$;
21         end
22         else if ($s_{ls}^i$ not a clone) then
23             add to pop$^t$;
24         end
25     end
26     pop$^{t+1}$ = Stochastic Ranking(pop$^t$);
27     $t = t + 1$
28 end
29 return best individual in population pop$^{G_m}$
```

this phase. Therefore, in order to contrast the exploitation ability of the algorithm, A control over the diversity of the population is embedded in the ranking operator. Moreover the exploration capability of the recombination move is increased by replacing the SBX with a suite of different operators.

3.3 A Distance Measure for the CARP

In order to be able to evaluate the performance of the algorithm in terms of exploration ability, it is necessary to define a distance measure between solutions. This section identifies three possible approaches to define such a measure are proposes the adoption of a novel distance metric for CARP.

3.3.1 Measuring the Average Diversity of the Population

A first approach is to consider routes as clusters of tasks and consequently to choose an index that is commonly used in the data clustering context. Two non trivial issues affect this approach: it does not take into account the task service order and it has a high computational cost ($O(n^2)$).

One could choose a different approach by considering routes as strings and to use similarity indexes taken from this field, such as the Levenshtein distance [73]. Although this approach is more precise than the previous one, it is still computationally non trivial ($O(n^2)$) and it has the issue of depending on the chosen representation.

A third approach is that of considering the relationship of consecutiveness between edges as items of the dataset (e.g. if task t_2 follows task t_1 the couple (t_1, t_2) is an item of the dataset) and using a similarity measure between sets. The task service order is therefore split into couples of tasks. This approach has a linear computational time ($O(n)$) and it achieves a higher precision, if compared to the classic clustering approach, by taking into account the task service order. Such a measure has been successfully used in the context of multi-objective Vehicle Routing Problem as in [42] where the number of overlapping arcs is measured through the use of the Jaccard Index [62].

This measure is extended by developing a metric that exploits the task service order and that can be used to infer information from the population such as the average pairwise diversity.

3.3.2 A Revised Distance Measure Based on Neighbouring Tasks

Algorithm 3.5: Diversity Measure for CARP

```
1  find $p_1(t)$ and $n_1(t) \forall t$ in $S_1$;
2  find $p_2(t)$ and $n_2(t) \forall t$ in $S_2$;
3  for (each task t) do
4      if (task t is served in both solutions) then
5          if ( $p_1(t) = p_2(t)$ ) then
6              increase by one;
7          end
8          if ( $n_1(t) = n_2(t)$ ) then
9              increase by one;
10         end
11     end
12 end
13 divide the obtained value by $2N$;
```

Similarly to the aforementioned measure, we exploit the relation of consecutiveness between tasks inside the routes. However, the similarity measure adopted in this work is not based on the number of shared arcs between two tasks, but on the number of tasks which share their previous or next arcs. This approach maintains a linear computational cost and takes into account the service order, but has the advantage of always having consistent measurements as the number of tasks is a constant value. The distance between two solutions S_1 and S_2 is described in Algorithm 3.5.

Since a CARP solution is represented by a sequence of tasks t, split into different routes, we can define $p_i(t)$ and $n_i(t)$ as two functions that return respectively the previous and the next tasks of task t in the sequence of solution S_i, regardless of the route. A task t has a perfect correspondence in both solutions if its previous and next tasks match. In the most extreme cases, for two solutions S_1 and S_2, the value of the distance measure will be equal to 1 if $p_1(t) = p_2(t)$ and $n_1(t) = n_2(t)$ for each task t, and will be equal to 0 if $(p_1(t) \neq p_2(t)$ and $n_1(t) \neq n_2(t)$ for every possible case. In the former case, the two solutions are identical, as there is a full correspondence between the $p_i(t)$ and the $n_i(t)$ of both solutions for each task t, while in the latter case the two solutions are completely different. It is important to point out that while the order in which the tasks in each route are serviced is considered, the order of the routes is not. Therefore two solutions are still identical if they have perfectly corresponding routes even if permuted in a different order. In all the other cases, when the correspondence is partial, the diversity measure will consequently assume values within the range $[0, 1]$. We also point out that the $2N$ possible values achievable are uniformly distributed in the $[0 - 1]$ space, where N is equal to the number of requested arcs in the CARP instance. This allows the calculation of average similarities within the population.

On the Difference With the Jaccard Index Measure

To provide an example of the way the distance is computed, and to clarify how it differs from the Jaccard Index measure of [42], let us consider two solutions p and q:

$$p = [(t_1, t_2, t_3), (t_4, t_5, t_6), (t_7, t_8)]$$

and

$$q = [(St_2, t_3, t_1), (t_4, t_6, t_7, t_8, t_5)]$$

and remembering that the Jaccard Index formula is:

$$\text{jaccard}(p, q) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} y_{ijp} \times y_{ijq}}{\sum_{i=0}^{n} \sum_{j=0}^{n} \text{sign}(y_{ijp} + y_{ijq})} \tag{1}$$

where $y_{ijp} \times y_{ijq}$ is 1 only if the arc ij is present in both solutions p and q and $\text{sign}(y_{ijp} + y_{ijq})$ is 1 for every arc that is present in either solution. It is immediate to see that the intersection of the set of arcs of the two solutions is composed by 2 arcs, respectively (t_2, t_3) and (t_7, t_8). The value of $\text{jaccard}(p, q)$ will therefore be:

$$\text{jaccard}(p, q) = \frac{2}{9}$$

as there are a total of 9 distinct arcs in the two solutions.

Table 3.1 shows the calculations the compute the diversity between the two solutions p and q using the distance measure proposed in this thesis. Using the formula included in section 3.3, the result will be:

$$\text{diversity}(p, q) = \frac{5}{16}$$

3.3.3 Diversity-Driven Stochastic Ranking

This work adopts the diversity measure between CARP solutions to define a new ranking operator with the aim to preserve the diversity of the population. In order to do that, this information is embedded in the stochastic ranking operator [120], which has also been used in the MAENS algorithm [128].

The pseudo-code of the Diversity-Driven Stochastic Ranking is that of Algorithm 3.6. The main difference with the original algorithm is in line 4. While in the stochastic ranking the comparison based on the fitness value of the solutions is performed either

Table 3.1: Calculation of the distance measure. For each task (column *task*), the table shows its predecessor (column *pred*) and successor (column *successor*) for the two solutions p and q shown in 13. Last column (*count*) shows whether one or both tasks relationships are matching.

	p		q		
task	predecessor	successor	predecessor	successor	count
t_1	t_0	t_2	t_3	t_0	0
t_2	t_1	$\mathbf{t_3}$	t_0	$\mathbf{t_3}$	1
t_3	$\mathbf{t_2}$	t_0	$\mathbf{t_2}$	t_1	1
t_4	$\mathbf{t_0}$	t_5	$\mathbf{t_0}$	t_6	1
t_5	t_4	t_6	t_8	t_0	0
t_6	t_5	t_0	t_4	t_7	0
t_7	t_0	$\mathbf{t_8}$	t_6	$\mathbf{t_8}$	1
t_8	$\mathbf{t_7}$	t_0	$\mathbf{t_7}$	t_5	1

Algorithm 3.6: Diversity-Driven Stochastic Ranking

```
[!htbp] Assign a random order to the solutions of the population **for** *(size of the population)* **do**
    **for** *(each pair of consecutive individuals p and q in the population)* **do**
        extract a random value $r$ in the range $[0 - 1]$;
        **if** $(d_{AVG}(p) < d_{AVG}(q)$ *OR* $r < R)$ **then**
            **if** *(fitness(p)>fitness(q))* **then**
                switch position of $p$ and $q$;
            **end**
        **end**
        **else**
            **if** *(fitness(p)>fitness(q))* **then**
                switch position of $p$ and $q$;
            **end**
        **end**
    **end**
**end**
```

when both solutions have no constraint violation or with a probability R, which lies in a range between 0.40 and 0.50 according to the authors of the original paper[120]. Even in this case the same value adopted in the MAENS algorithm is maintained, to prevent to obtain results that are affected by aspects that are not the ones proposed in this work. In this version the comparison is performed when the average distance of the solution p from all the rest of the population, called d_{AVG} is less than that of solution q. Therefore, we consider the value d_{AVG} as a *crowding distance* which is supposed to bias the search towards the solutions which lie in areas of the search space which have not been thoroughly exploited. As this is the first attempt to use such a measure, the work might be extended by considering either only the n closest solutions, or the solutions with a distance over a certain threshold, or by considering the distance of each solution from the *centroid* individual of the population.

3.4 Operator Selection

As previously mentioned, the use of a single heuristic to generate routes might limit the exploration capacity of the algorithm and consequently its ability to escape local optima. We propose the use of a suite of crossover operators under the assumption that it might lead to a more robust performance of the algorithm.

This offers the advantage to use each heuristic in those instances where they perform better. As this is not known a priori, it is necessary to define an operator-selection strategy which is able to address the problem of selecting the best performing crossover operator for each instance. Besides, the use of the proper combination of heuristics might improve further the performance of the single use of each one of them, as different heuristics might be the best in different phases of the search.

3.4.1 Crossover Operators

We have defined four new crossover operators for the CARP problem, denoted as GSBX, GRX, PBX, SPBX, which are described in this section.

Greedy Sequence Based Crossover (GSBX)

The first operator combines a greedy selection with the SBX [113] operator. The greedy choice replaces the random selection of routes using the following rule that supports the selection of those routes which might have been less efficiently filled. The pseudo-code of the GSBX algorithm is provided in Algorithm 3.7. The probability of selecting a route r from a parent solutions is proportional to the value of

$$score(r) = [1 - \frac{C - load(r)}{C}] \tag{2}$$

where C is the maximum capacity of a vehicle. It is easy to see that the probability of being extracted will be higher if the route is emptier (lines 3-7). Once the operator has extracted one route from each parent solution, respectively r_1 and r_2 (lines 9-10), a new route is obtained by performing the one point crossover on the two routes (line 11). The offspring is generated by replacing the newly generated route in one of the parents and by repairing the solution in presence of missing tasks or if any task has been serviced more than once (line 12).

Greedy Route Crossover (GRX)

The GRX operator adapts the concept of the GPX operator for the Graph Colouring Problem [40] in the context of the CARP. The pseudo-code of the algorithm is included in Algorithm 3.8. Given an initial randomly selected parent solution, the algorithm will identify the best route to include in the offspring by mean of the same greedy-rule used in the GSBX operator. (lines 4-9).

Algorithm 3.7: GSBX Crossover

```
1  $S_1$ and $S_2$ parent solutions;
2  $S_{\text{new}} = S_1$;
3  for (each route $r_i$ in $S_1$) do
4      compute score($r_i$)=$1 - \frac{C - load(r_i)}{C}$
5  end
6  for (each route $r_j$ in $S_2$) do
7      compute score($r_i$)=$1 - \frac{C - load(r_i)}{C}$
8  end
9  extract $r_1$ from $S_1$ with p($r_1$) = score($r_i$);
10 extract $r_2$ from $S_2$ with p($r_2$) = score($r_j$);
11 combine $r_1$ and $r_2$ and replace $r_1$ in $S_{\text{new}}$ with the new route;
12 repair $S_{\text{new}}$ if there are missing or repeated tasks;
13 return $S_{\text{new}}$;
```

The route is then copied in the offspring and its tasks are removed from parent solution routes (lines 10-12). The algorithm proceeds to repeat this operation selecting alternatively both parent solutions with a round robin criterion, until all remaining routes R in both parent solutions have load(R) $< C/2$ (the cycle condition in line 3). The remaining routes are then randomly selected and combined to form new routes (lines 14-17) or inserted (when $|R| = 1$) in the existing ones in the positions which minimize the total service cost of the solution (lines 18-21).

Pivot Based Crossover (PBX)

The PBX operator ranks a list of tasks in a similar way to the Augment-Insert [108] heuristic, although it differs from it as the tasks are ranked according to the load of their outward and return paths, instead of their total service cost.

The pseudo-code of the Augment-Insert procedure is provided in 3.9. Augment-Insert builds a new route by identifying the unserviced task that minimizes the value sc(head(t), tail(t)) + sc(tail(t), head(t)) (line 2). All requested tasks in this path are then added to the route in the same sequence (line 3). The route is then completed by inserting tasks

Algorithm 3.8: GRX Crossover

```
1  $S_1$ and $S_2$ parent solutions;
2  $S_{\text{new}} = \{\emptyset\}$;
3  while ($\exists r \in S_1$ or $r \in S_2$ such that $load(r) > C/2$) do
4      select alternatively $S_1$ and $S_2$;
5      foreach (route $r$ in current parent solution) do
6          compute score($r$)=$1 - \frac{C - load(r)}{C}$
7      end
8      find $r^*$ with highest score;
9      $S_{\text{new}} = S_{\text{new}} \cup r^*$;
10     foreach (task $t \in r^*$) do
11         remove $t$ from $S_1$ and $S_2$;
12     end
13 end
14 foreach (route $r \in S_1$ or $r \in S_2$ not yet included in $S_{new}$) do
15     try to insert in an existing route minimizing the total cost;
16     otherwise insert into $S_{\text{new}}$;
17 end
18 foreach (task $t$ not yet included in $S_{new}$) do
19     try to insert in an existing route minimizing the total cost;
20     otherwise insert into an empty route in $S_{\text{new}}$;
21 end
22 return $S_{\text{new}}$;
```

that minimize the increment of the service cost, provided that the capacity constraint is satisfied (cycle in lines 5-7).

Algorithm 3.9: Augment-Insert

```
1 initialize an empty route $R$ ;
2 sort the tasks in decreasing order based on the value sc(head(t),tail(t)) +
    sc(tail(t),head(t));
3 $t^*$ = task with the smallest sc(head(t),tail(t)) + sc(tail(t),head(t));
4 insert all tasks crossed in the shortest path of $t^*$;
5 while $Capacity(R) < C$ do
6     add the unserviced task that minimizes the increment of the service cost ;
7 end
8 return the route $R$.
```

The pseudo-code of the PBX crossover is included in algorithm 3.10. A route from each parent individual is randomly selected and their tasks are inserted in a unordered list L (lines 2-4). A function $load_L(path)$ returns the total load of all the tasks in L that belong to a certain path. A score is then assigned to every task t in L, corresponding to the larger $load_L$ between its return and outward path costs (line 5-7). The selected subpath is copied into a new route r_{new} (line 18), which is completed by inserting the tasks in L that minimize the total cost of the route and that do not violate the capacity constraint (lines 19-21). Remaining tasks in L are either inserted in the existing solution or in a new route (lines 22-25).

Shortest Path Based Crossover (SPBX)

The SPBX applies the idea of the PBX operator to a couple of tasks (t_A, t_B). Rather than identifying a *pivot*, SPBX identifies a pivot path. The pseudo-code of the SPBX algorithm is provided in Algorithm 3.11. The function $SP_L(t_a, t_b)$, returns the total service costs of the tasks that lie along the shortest path between tasks t_a and t_b, and that belong to the set L. The algorithm will select the couple (t_i, t_j) with a probability proportional to its

Algorithm 3.10: PBX Crossover

```
$S_1$ and $S_2$ parent solutions;
extract $r_1$ and $r_2$ with uniform probability from $S_1$ or $S_2$;
$S_{\text{new}} = S_1 - r_1$;
$L = \{\emptyset\}$;
$L = L \cup r_1$;
$L = L \cup r_2$;
foreach ( task $t \in L$) do
    if ($load_L(path(t, depot) > load_L(path(depot, t))$ then
        score$(t)$ = $load_L(path(t, \text{depot})$;
        pivotPath$(t)$ = $path(t, \text{depot})$;
    end
    else
        score$(t)$ = $load_L(path(\text{depot}, t)$;
        pivotPath$(t)$ = $path(\text{depot}, t)$;
    end
end
extract pivot task $t^*$ with $p(t) \propto$ score$(t)$;
$r_{\text{new}}$ = pivotPath$(t^*)$;
while $\exists t$ can be inserted in $r_{new}$ and $L$ not empty do
    insert the task $t$ in $L$ that induces the minimum increment of the service cost;
end
foreach (task $t$ in $L$ not yet included in $S_{new}$) do
    try to insert in an existing route minimizing the total cost;
    otherwise insert into an empty route in $S_{\text{new}}$;
end
add $r_{\text{new}}$ to $S_{\text{new}}$;
return $S_{\text{new}}$;
```

score, which is calculated with the formula:

$$score(t_i, t_j) = SP_L(t_A, t_B) - SP_L(\text{depot}, t_i) + SP_L(t_j, \text{depot}) \qquad (3)$$

The formula assigns an higher score to the couples having larger SP_L and shorter distance from and to the depot (lines 7-11). The *pivot* path is copied into a new route r_{new} (line 12), which is completed by inserting the tasks in L that minimize the total cost of the route and that do not violate the capacity constraint (lines 13-15). Remaining tasks in L are either inserted in the existing solution or in a new route (lines 16-19).

Algorithm 3.11: SPBX Crossover

```
1  $S_1$ and $S_2$ parent solutions;
2  extract $r_1$ and $r_2$ with uniform probability from $S_1$ or $S_2$;
3  $S_{\text{new}} = S_1 - r_1$;
4  $L = \{\emptyset\}$;
5  $L = L \cup r_1$;
6  $L = L \cup r_2$;
7  foreach ( couple of tasks $(t_i, t_j) \in L, i \neq j$) do
8      score$(t_i, t_j) = SP_L(t_i, t_j) - SP_L(\text{depot}, t_i) + SP_L(t_j, \text{depot})$;
9      pivotPath$(t_i, t_j) = path(t_i, t_j)$;
10 end
11 extract pivot path$(t_i^*, t_j^*)$ with $p((t_i, t_j)) \propto$ (roulette wheel) score$((t_i, t_j))$;
12 $r_{\text{new}}$ = pivotPath$(t_i, t_j)$;
13 while $\exists t$ can be inserted in $r_{new}$ and $L$ not empty do
14     insert the task $t$ in $L$ that induces the minimum increment of the service cost;
15 end
16 foreach (task $t$ in $L$ not yet included in $S_{new}$) do
17     try to insert in an existing route minimizing the total cost;
18     otherwise insert into an empty route in $S_{\text{new}}$;
19 end
20 add $r_{\text{new}}$ to $S_{\text{new}}$;
21 return $S_{\text{new}}$;
```

3.4.2 Adaptive Operator Selection

We have chosen to adopt a Multi Armed Based (MAB) based technique, called dynamic MAB (dMAB), first proposed in [22], and we have combined it with a new credit assignment technique based on the use of the diversity-driven ranking operator that we have previously introduced. The choice of dMAB is merely dictated by the fact that it is one of the most promising techniques in the area of AOS, as in this stage we are not interested into identifying the most efficient operator selection technique. Comparisons of the performance of dMAB with respect other techniques, which have not been performed in this context, can be found in dMAB original paper [22].

Dynamic Multi-Armed Bandit

The dMAB adapts the classic Multi-Armed Bandit scenario to a dynamic context where the reward probability of each arm is not independent and is not fixed. To address the dynamic context problem, the classical Upper Confidence Bound (UCB1) [2] algorithm is combined with a Page Hinkley test [57], to identify the change of reward probabilities. More information can be found in the original paper [22].

Proportional Reward

We propose a new credit assignment mechanism, named Proportional Reward (PR), exploiting the selection operated by the diversity-driven ranking operator. We assign a reward r which is proportional to the number of offspring generated by the selected operator that will survive to the next generation after the stochastic ranking operator has been applied. Therefore, $r = 0$ when none of the individuals generated by the selected operator has survived the ranking process and $r = 1$ when only individuals generated by it have been chosen by the ranking operator.

The PR can be formally represented with the following formula:

$$PR(i)^t = \frac{|x_i : x_i \in \text{parent}^{t+1}|}{|\text{parent}^{t+1}|} \tag{4}$$

where i refers to the i-th operator, x_i is an individual obtained through the use of operator i, t is the t-th generation and parent^{t+1} is the parent population at the $t+1$ generation. If more than one operator is used during the same generation, the PR can be calculated in the following way:

$$PR(i)^t = \frac{|x_i : x_i \in \text{parent}^{t+1}|}{|\text{offspring}_i^t|} \tag{5}$$

where offspring$_i^t$ is the set of individuals generated using the operator i during the t-th generation.

Such a technique shows several advantages with respect to classic fitness-based credit assignment ones:

- it takes into account the fitness of the solutions, their similarity and the violation of the constraints;
- the adaptive operation selection process does not require domain knowledge;
- the reward values are always normalized and there is no need to derive a scaling factor;
- unlike fitness-based techniques, it is not affected by the convergence speed of the algorithm.

3.5 Experimental Studies

We have tested the results of the original algorithm against several versions that we have labelled *MAENSD*, *MAENSM* and *MAENS**, which are respectively the versions of the algorithm adopting the diversity-driven stochastic ranking operator, the one using the proposed AOS strategy and the combination of both techniques.

Table 3.2: Parameter values for the MAENS* algorithm

Name	Description	Value
psize	population size	30
ubtrial	maximum attempts to generate each initial solution	50
opsize	offspring generated during each generation	6*psize
P_{ls}	probability of performing the local search	0.2
p	number of routes selected during MergeSplit	2
G_m	maximum generations	500
SR_r	probability of sorting solutions according to their fitness	0.45
σ	tolerance factor for Page-Hinkley test	0.05
λ	change threshold for Page-Hinkley test	1.25

The comparison has been carried out on four benchmark test sets, namely *gdb* [25] (23 instances), *val* [8] (34 instances), *egl* [28] (24 instances), and *BMVC al.* [9], which is composed of four groups of 25 instances, namely C,D,E and F.

Table 3.3: A summary of the results of the four algorithms. Each column shows the number of instances where each algorithm achieved a better average fitness (W), performed equally (D) or worse (L) than MAENS, as well as the mean approximation ratio across the whole dataset (ε)

	MAENS	MAENSD	MAENSM	MAENS*
W	–	85/181	79/181	92/181
D	–	77/181	74/181	76/181
L	–	19/181	28/181	13/181
ε	.0195	.0158	.0164	.0151

The Wilcoxon Signed-Rank test [136] has been used to perform a statistical hypotesis test between MAENS and each of the proposed versions. The test has been conducted using the *R* software environment [115]. In each case, the test has rejected the null hypothesis that the results of the two algorithms were not significantly different. Table 3.4 reports the details of such tests.

Table 3.2 shows the parameters used to execute the algorithm for 30 independent trials on each of the 181 instances. It was not possible to perform a paired test for each group of results achieved on every instance, as the results of the single executions of the MAENS algorithm were not available for comparison. The algorithm has not been through a process of parameter configuration, and the parameters present in the original version of the algorithm (first 7 parameters included in table 3.2) have kept their original values. This strategy has been adopted throughout this thesis to prevent the results to be affected by a different parameter configuration rather than by the proposed contributions described in this work. New parameters such as those necessary for the Page-Hinckley test have been identified through a few test-and-trial attempts, using the set of benchmark instances as a training set and might not be the optimal ones.

Table 3.4: Results of the statistical tests on the *MAENSM*, *MAENSD* and *MAENS** algorithm, as described in A.1. The columns show the V statistic computed with the Wilcoxon Signed-Rank Test and the p-value obtained

	MAENSD	MAENSM	MAENS*
V	4683	4917	5238.5
p-values	2.426e-10	2.945e-10	4.164e-15

The results obtained through this experiment are reported in tables 3.5, 3.6, 3.8, 3.9, 3.10, 3.11. A more detailed description of the statistical analysis performed is available in A.1. In terms of mean average fitness, both MAENSD and MAENSM manage to outperform the original algorithm in 85 and 79 instances, and lose the comparison only in 19 and 28 cases. The results of their combined version MAENS* confirm how the combination of the two techniques has a constructive effect on the algorithm, achieving a better average fitness in 92 instances and losing in only 13. The algorithms were also compared in terms of their average approximation-ratio. MAENS ratio of 0.0195 has been reduced to 0.0158 and 0.0164 in the cases of MAENSD and MAENSM and to 0.0151 in the case of MAENS*.

Table 3.5: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the *gdb* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table provides the average fitness of the best solution, its standard deviation and the best result achieved in columns *avg*, *std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$	best	avg	std	best	avg	std	best	avg	std	best	avg	std	best
gdb1	12	22	22	316	316	0.00	316	316	0.00	316	316	0.00	316	316	0.00	316
gdb2	12	26	26	339	339	0.00	339	339	0.00	339	339	0.00	339	339	0.00	339
gdb3	12	22	22	275	275	0.00	275	275	0.00	275	275	0.00	275	275	0.00	275
gdb4	11	19	19	287	287	0.00	287	287	0.00	287	287	0.00	287	287	0.00	287
gdb5	13	26	26	377	377	0.00	377	377	0.00	377	377	0.00	377	377	0.00	377
gdb6	12	22	22	298	298	0.00	298	298	0.00	298	298	0.00	298	298	0.00	298
gdb7	12	22	22	325	325	0.00	325	325	0.00	325	325	0.00	325	325	0.00	325
gdb8	27	46	46	348	348.7	1.00	348	348.07	0.36	348	348.4	0.80	348	348	0.00	348
gdb9	27	51	51	303	303	0.00	303	303.03	0.18	303	303.3	0.69	303	303	0.00	303
gdb10	12	25	25	275	275	0.00	275	275	0.00	275	275	0.00	275	275	0.00	275
gdb11	22	45	45	395	395	0.00	395	395	0.00	395	395	0.00	395	395	0.00	395
gdb12	13	23	23	458	458	0.00	458	458	0.00	458	458	0.00	458	458	0.00	458
gdb13	10	28	28	536	536	0.00	536	536	0.00	536	536	0.00	536	536	0.00	536
gdb14	7	21	21	100	100	0.00	100	100	0.00	100	100	0.00	100	100	0.00	100
gdb15	7	21	21	58	58	0.00	58	58	0.00	58	58	0.00	58	58	0.00	58
gdb16	8	28	28	127	127	0.00	127	127	0.00	127	127	0.00	127	127	0.00	127
gdb17	8	28	28	91	91	0.00	91	91	0.00	91	91	0.00	91	91	0.00	91
gdb18	9	36	36	164	164	0.00	164	164	0.00	164	164	0.00	164	164	0.00	164
gdb19	8	11	11	55	55	0.00	55	55	0.00	55	55	0.00	55	55	0.00	55
gdb20	11	22	22	121	121	0.00	121	121	0.00	121	121	0.00	121	121	0.00	121
gdb21	11	33	33	156	156	0.00	156	156	0.00	156	156	0.00	156	156	0.00	156
gdb22	11	44	44	200	200	0.00	200	200	0.00	200	200	0.00	200	200	0.00	200
gdb23	11	55	55	233	233	0.00	233	233	0.00	233	233	0.00	233	233	0.00	233

With regard to the runtime, MAENS* is essentially comparable to its original version. The additional computational cost introduced by the calculation of the average diversity of the solutions is balanced by the improved convergence speed, while the AOS does not add any noticeable effort in the algorithm, whose computational cost is still largely dominated by the local search procedure.

3.6 Summary and Discussion

The contributions of this chapter are:

Table 3.6: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the *val* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$\lvert V\rvert$, $\lvert R$, $\lvert E\rvert$, *best*. For each algorithm, the table provides the average fitness of the best solution, its standard deviation and the best result achieved in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\lvert V\rvert$	$\lvert R\rvert$	$\lvert E\rvert$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
val1A	24	39	39	173	173	0.00	173	173	0.00	173	173	0.00	173	173	0.00	173
val1B	24	39	39	173	173	0.00	173	173	0.00	173	173	0.00	173	173	0.00	173
val1C	24	39	39	245	245	0.00	245	254.7	1.16	253	245	0.00	245	255.13	1.25	253
val2A	24	34	34	227	227	0.00	227	227	0.00	227	227	0.00	227	227	0.00	227
val2B	24	34	34	259	259	0.00	259	259	0.00	259	259	0.00	259	259	0.00	259
val2C	24	34	34	457	457.2	1.1	457	457	0.00	457	457	0.00	457	457	0.00	457
val3A	24	35	35	81	81	0.00	81	81	0.00	81	81	0.00	81	81	0.00	81
val3B	24	35	35	87	87	0.00	87	87	0.00	87	87	0.00	87	87	0.00	87
val3C	24	35	35	138	138	0.00	138	138	0.00	138	138	0.00	138	138	0.00	138
val4A	41	69	69	400	400	0.00	400	400	0.00	400	400	0.00	400	400	0.00	400
val4B	41	69	69	412	412	0.00	412	412	0.00	412	412	0.00	412	412	0.00	412
val4C	41	69	69	428	431.1	3.1	428	428.53	1.26	428	431.37	2.73	428	428.93	1.69	428
val4D	41	69	69	526	532.9	3.3	530	530.73	1.75	530	532.53	3.03	530	530.13	0.72	530
val5A	34	65	65	423	423	0.00	423	423	0.00	423	423	0.00	423	423	0.00	423
val5B	34	65	65	446	446	0.00	446	446	0.00	446	446	0.00	446	446	0.00	446
val5C	34	65	65	473	474	0.00	474	474	0.00	474	474	0.00	474	474	0.00	474
val5D	34	65	65	573	582.9	2.2	577	583.97	2.65	579	585.8	3.65	577	583.13	2.06	579
val6A	31	50	50	223	223	0.00	223	223	0.00	223	223	0.00	223	223	0.00	223
val6B	31	50	50	233	233	0.00	233	233	0.00	233	233	0.00	233	233	0.00	233
val6C	31	50	50	317	317	0.00	317	317	0.00	317	317	0.00	317	317	0.00	317
val7A	40	66	66	279	279	0.00	279	279	0.00	279	279	0.00	279	279	0.00	279
val7B	40	66	66	283	283	0.00	283	283	0.00	283	283	0.00	283	283	0.00	283
val7C	40	66	66	334	334	0.00	334	334	0.00	334	334	0.00	334	334	0.00	334
val8A	30	63	63	386	386	0.00	386	386	0.00	386	386	0.00	386	386	0.00	386
val8B	30	63	63	395	395	0.00	395	395	0.00	395	395	0.00	395	395	0.00	395
val8C	30	63	63	518	525.9	1.7	521	525.67	1.74	521	526.9	2.65	522	524.63	1.76	521
val9A	50	92	92	323	323	0.00	323	323.1	0.3	323	323.13	0.34	323	323.03	0.18	323
val9B	50	92	92	326	326	0.00	326	326	0.00	326	326	0.00	326	326	0.00	326
val9C	50	92	92	332	332	0.00	332	332	0.00	332	332	0.00	332	332	0.00	332
val9D	50	92	92	385	391	0.00	391	391	0.00	391	391.37	1.05	391	391	0.00	391
val10A	50	97	97	428	428	0.00	428	428.2	0.4	428	428.23	0.5	428	428.1	0.3	428
val10B	50	97	97	436	436	0.00	436	436.13	0.34	436	436.13	0.34	436	436.13	0.34	436
val10C	50	97	97	446	446	0.00	446	446.07	0.25	446	446.07	0.25	446	446	0.00	446
val10D	50	97	97	525	533.6	1.5	531	531.5	1.28	530	532.8	1.33	530	530.6	1.23	528

- a novel distance metric between CARP solutions, with linear computational cost, which can be used to analyze the average diversity of a population of CARP solution in reasonable time;

Table 3.7: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the *egl* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table includes the average fitness of the best solution, its standard deviation and the best result achieved in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
E1-A	77	51	98	3548	3548	0.00	3548	3548	0.00	3548	3548	0.00	3548	3548	0.00	3548
E1-B	77	51	98	4498	4516.5	17.6	4498	4505.77	11.06	4498	4512.13	12.22	4498	4501.2	8.33	4498
E1-C	77	51	98	5566	5601.6	9.9	5595	5595	0.00	5595	5598.6	7.2	5595	5595	0.00	5595
E2-A	77	72	98	5018	5018	0.00	5018	5018	0.00	5018	5018	0.00	5018	5018	0.00	5018
E2-B	77	72	98	6305	6341.4	12	6317	6325.6	8.28	6317	6330.87	11.25	6317	6323.67	9.58	6317
E2-C	77	72	98	8243	8355.7	35.9	8335	8336.47	2.19	8335	8335.67	1.49	8335	8335.13	0.72	8335
E3-A	77	87	98	5898	5898.8	2.9	5898	5898	0.00	5898	5898	0.00	5898	5898	0.00	5898
E3-B	77	87	98	7704	7802.9	27.3	7775	7784.2	9.26	7777	7791.83	14.68	7775	7780.43	5.91	7775
E3-C	77	87	98	10163	10321.9	18	10292	10313.43	15.1	10292	10314.4	19.42	10292	10317.6	18.45	10292
E4-A	77	98	98	6408	6475.2	10.3	6456	6464.63	3.04	6460	6466.47	4.54	6460	6462.5	3.04	6450
E4-B	77	98	98	8884	9023	18.7	8998	9015.57	17.78	8992	9002	13.89	8988	9022.5	16.39	8988
E4-C	77	98	98	11427	11645.8	46.7	11561	11597.23	26.69	11550	11649.37	57.56	11577	11592.53	32.82	11538
S1-A	140	75	190	5018	5039.8	35.9	5018	5018	0.00	5018	5018	0.00	5018	5018	0.00	5018
S1-B	140	75	190	6384	6433.4	8.6	6388	6404.53	18.09	6388	6410.17	21.83	6388	6399.9	16.38	6388
S1-C	140	75	190	8493	8518.3	1.5	8518	8518	0.00	8518	8518	0.00	8518	8518	0.00	8518
S2-A	140	147	190	9824	9959.2	34.6	9895	9945.03	24.44	9903	9950.27	31.96	9892	9931.63	26.62	9889
S2-B	140	147	190	12968	13231.6	63.2	13147	13192.73	36.58	13123	13217.43	49.52	13145	13179.07	26.11	13124
S2-C	140	147	190	16353	16509.8	51.8	16430	16515.2	52.04	16446	16535.1	60.57	16442	16510.1	43.05	16430
S3-A	140	159	190	10143	10312.7	26.5	10257	10294.67	26.45	10220	10304.97	29.36	10250	10282.63	29.41	10221
S3-B	140	159	190	13616	13876.6	67.8	13749	13864.73	61.41	13754	13874.1	91.84	13697	13820.13	57.75	13736
S3-C	140	159	190	17100	17305.8	41.4	17207	17312.67	35.96	17235	17315.2	36.27	17222	17289.73	42.75	17220
S4-A	140	190	190	12143	12419.2	33.2	12341	12423.93	28.16	12336	12403.97	44.51	12304	12400.87	47.91	12283
S4-B	140	190	190	16093	16441.2	38.1	16337	16462.07	38.9	16351	16465.9	46.32	16336	16421.17	50.46	16325
S4-C	140	190	190	20375	20767.2	74.6	20538	21072.87	139.28	20836	20769.53	85.08	20563	21047.97	174.66	20758

- a diversity-driven stochastic ranking operator, exploiting the aforementioned notion of distance metric between CARP solutions, which dynamically balances the exploration and exploitation tendencies of the algorithm;
- the definition of four novel crossover operators for CARP;
- a novel Credit Assignment strategy, called Proportional Reward, based on the survival ability of the offspring generated by the operators;
- the design of the MAENS* algorithm, which introduces the aforementioned ideas in the MAENS algorithm for CARP.

The results of the work described in this chapter provide an initial answer to the

Table 3.8: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the group C of *BMCV* CARP dataset. For each problem instance the table includes the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table shows the average fitness of the best solution, its standard deviation and the best result achieved, respectively in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
C1	69	79	98	1590	1707.00	23.90	1660	1676.67	23.25	1660	1684.33	24.49	1660	1671.67	19.38	1660
C2	48	53	66	1095	1095.70	3.70	1095	1095.00	0.00	1095	1095.83	1.86	1095	1095.00	0.00	1095
C3	46	51	64	875	927.80	3.90	925	925.00	0.00	925	928.17	4.56	925	925.00	0.00	925
C4	60	72	84	1285	1342.70	4.50	1340	1340.00	0.00	1340	1340.00	0.00	1340	1340.00	0.00	1340
C5	56	65	79	2410	2522.30	30.00	2470	2473.50	6.34	2470	2479.00	16.35	2470	2471.00	2.00	2470
C6	38	51	55	855	907.50	3.40	895	902.50	5.12	895	902.50	5.59	895	902.00	4.58	895
C7	54	52	70	1735	1795.00	0.00	1795	1795.00	0.00	1795	1795.00	0.00	1795	1795.00	0.00	1795
C8	66	63	88	1640	1732.30	4.30	1730	1730.00	0.00	1730	1730.00	0.00	1730	1730.00	0.00	1730
C9	76	97	117	1775	1852.80	21.50	1820	1831.83	17.20	1820	1847.50	23.69	1820	1830.00	16.73	1820
C10	60	55	82	2190	2317.80	43.80	2270	2274.83	8.80	2270	2279.33	11.81	2270	2272.17	6.54	2270
C11	83	94	118	1725	1853.70	34.10	1815	1820.83	13.61	1815	1827.67	18.43	1815	1816.33	3.14	1805
C12	62	72	88	1510	1610.00	0.00	1610	1610.00	0.00	1610	1610.00	0.00	1610	1610.00	0.00	1610
C13	40	52	60	1050	1122.00	21.40	1110	1110.00	0.00	1110	1110.00	0.00	1110	1110.00	0.00	1110
C14	58	57	79	1620	1687.30	10.80	1680	1680.67	2.49	1680	1682.00	4.00	1680	1680.67	2.49	1680
C15	97	107	140	1765	1896.50	16.30	1860	1873.83	10.54	1860	1879.50	15.78	1860	1866.17	7.38	1860
C16	32	32	42	580	585.20	0.90	585	585.00	0.00	585	585.67	2.81	585	585.00	0.00	585
C17	43	42	56	1590	1618.30	17.80	1610	1610.00	0.00	1610	1610.00	0.00	1610	1610.00	0.00	1610
C18	93	121	133	2315	2411.70	18.90	2390	2405.17	6.89	2390	2407.50	11.01	2385	2403.67	7.74	2385
C19	62	61	84	1345	1425.70	19.10	1395	1398.67	2.21	1395	1398.67	2.21	1395	1396.83	2.41	1395
C20	45	53	64	665	668.50	6.70	665	665.00	0.00	665	665.00	0.00	665	665.00	0.00	665
C21	60	76	84	1705	1725.20	0.90	1725	1725.00	0.00	1725	1725.00	0.00	1725	1725.00	0.00	1725
C22	56	43	76	1070	1070.00	0.00	1070	1070.00	0.00	1070	1070.00	0.00	1070	1070.00	0.00	1070
C23	78	92	109	1620	1724.30	30.80	1690	1690.00	0.00	1690	1691.00	3.96	1690	1690.00	0.00	1690
C24	77	84	115	1330	1368.50	6.20	1360	1360.33	1.25	1360	1364.00	5.23	1360	1360.83	2.61	1360
C25	37	38	50	905	907.00	7.60	905	905.00	0.00	905	905.00	0.00	905	905.00	0.00	905

research questions relative to the effectiveness of the use of online technique to improve the robustness of EAs for CARP. Moreover, this study can be considered significant for several aspects:

- The work introduces a set of instruments that can be easily used to develop new online operators and to gain a deeper understanding of the behaviour of the EAs for CARP. In particular, the CARP diversity measure will play a central role in the rest of this thesis, as it will prove to be an extremely useful instrument to monitor the behaviour of the algorithm;

- The definition of the Proportional Reward introduces a discussion relative to the

Table 3.9: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the group D of *BMCV* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table includes the average fitness of the best solution, its standard deviation and the best result achieved, respectively in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
D1	69	79	98	725	745.00	0.00	745	741.67	6.75	725	742.83	5.11	725	742.83	4.02	725
D2	48	53	66	480	480.00	0.00	480	480.00	0.00	480	480.00	0.00	480	480.00	0.00	480
D3	46	51	64	415	415.20	0.90	415	415.00	0.00	415	415.00	0.00	415	415.00	0.00	415
D4	60	72	84	615	616.00	3.80	615	615.00	0.00	615	615.00	0.00	615	615.00	0.00	615
D5	56	65	79	1040	1040.00	0.00	1040	1040.00	0.00	1040	1040.00	0.00	1040	1040.00	0.00	1040
D6	38	51	55	485	493.00	15.60	485	485.00	0.00	485	485.00	0.00	485	485.00	0.00	485
D7	54	52	70	735	847.30	17.70	835	835.33	1.80	835	837.00	4.00	835	835.00	0.00	835
D8	66	63	88	615	704.20	15.50	685	690.00	5.00	685	689.00	4.90	685	685.67	2.49	685
D9	76	97	117	680	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680
D10	60	55	82	900	910.00	0.00	910	910.00	0.00	910	910.00	0.00	910	910.00	0.00	910
D11	83	94	118	920	935.20	6.20	920	935.83	2.27	935	937.67	5.28	935	936.50	3.91	935
D12	62	72	88	680	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680
D13	40	52	60	690	691.00	2.00	690	690.00	0.00	690	690.00	0.00	690	690.00	0.00	690
D14	58	57	79	920	931.00	3.10	930	931.00	3.00	930	931.33	3.40	930	930.67	2.49	930
D15	97	107	140	910	919.00	3.10	910	920.00	0.00	920	919.50	1.98	910	919.33	2.13	910
D16	32	32	42	170	170.00	0.00	170	170.00	0.00	170	170.00	0.00	170	170.00	0.00	170
D17	43	42	56	675	675.00	0.00	675	675.00	0.00	675	675.00	0.00	675	675.00	0.00	675
D18	93	121	133	930	934.20	8.70	930	932.83	5.11	930	936.00	7.57	930	930.33	1.80	930
D19	62	61	84	650	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680	680.00	0.00	680
D20	45	53	64	415	415.20	0.90	415	415.00	0.00	415	415.00	0.00	415	415.00	0.00	415
D21	60	76	84	695	834.20	18.80	810	815.00	4.28	805	815.50	4.54	810	814.83	5.24	805
D22	56	43	76	690	690.00	0.00	690	690.00	0.00	690	690.00	0.00	690	690.00	0.00	690
D23	78	92	109	715	748.20	8.40	735	771.67	10.83	750	767.00	9.88	745	769.83	12.28	740
D24	77	84	115	620	683.50	19.30	670	670.00	0.00	670	671.00	5.39	670	670.00	0.00	670
D25	37	38	50	410	410.00	0.00	410	410.00	0.00	410	410.00	0.00	410	410.00	0.00	410

reward measures used in literature, which is the object of study of chapter 4;

- The results produce evidence to support the use of Adaptive Operator Selection techniques for crossover operators, for which very few works in literature exist;
- The results of a experimental comparison between MAENS* and MAENS show that the former outperformed the original algorithm in terms of average fitness and produced more robust and reliable results, obtaining state-of-the-art performances in this research area;
- Some of the contributions of this work, such as the Proportional Reward and the

Table 3.10: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the group E of *BMCV* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table includes the average fitness of the best solution, its standard deviation and the best result achieved, respectively in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
E1	73	85	105	1855	1967.80	35.00	1935	1939.17	4.67	1935	1943.00	7.14	1935	1936.17	2.11	1935
E2	58	58	81	1580	1615.50	14.20	1610	1610.00	0.00	1610	1610.00	0.00	1610	1610.00	0.00	1610
E3	46	47	61	750	752.00	4.10	750	750.00	0.00	750	750.00	0.00	750	750.00	0.00	750
E4	70	77	99	1580	1684.30	21.10	1610	1613.00	9.36	1610	1645.33	32.27	1610	1610.33	1.80	1610
E5	68	61	94	2130	2228.70	49.00	2160	2191.33	21.17	2160	2207.67	22.65	2160	2181.67	22.34	2160
E6	49	43	66	670	670.00	0.00	670	670.00	0.00	670	670.00	0.00	670	670.00	0.00	670
E7	73	50	94	1780	1900.00	0.00	1900	1900.00	0.00	1900	1900.00	0.00	1900	1900.00	0.00	1900
E8	74	59	98	2080	2150.50	1.50	2150	2150.00	0.00	2150	2151.50	2.93	2150	2150.00	0.00	2150
E9	93	103	141	2160	2327.70	38.20	2235	2258.83	24.18	2225	2279.50	30.99	2225	2252.00	21.16	2230
E10	56	49	76	1690	1691.50	5.70	1690	1690.00	0.00	1690	1690.00	0.00	1690	1690.00	0.00	1690
E11	80	94	113	1810	1932.00	44.50	1850	1865.33	18.53	1850	1869.83	29.54	1850	1857.00	13.52	1835
E12	74	67	103	1580	1764.30	17.90	1710	1725.83	15.17	1710	1739.67	20.61	1710	1717.33	13.15	1695
E13	49	52	73	1300	1335.30	22.20	1325	1325.00	0.00	1325	1325.00	0.00	1325	1325.00	0.00	1325
E14	53	55	72	1780	1817.00	10.90	1810	1810.00	0.00	1810	1812.00	4.58	1810	1810.00	0.00	1810
E15	85	107	126	1555	1617.80	13.00	1595	1606.67	5.53	1595	1613.33	8.10	1600	1602.50	6.68	1590
E16	60	54	80	1785	1825.00	0.00	1825	1825.00	0.00	1825	1825.67	2.13	1825	1825.00	0.00	1825
E17	38	36	50	1290	1294.30	6.30	1290	1290.00	0.00	1290	1290.00	0.00	1290	1290.00	0.00	1290
E18	78	88	110	1600	1612.30	6.30	1610	1610.50	1.98	1610	1612.50	4.03	1610	1610.17	0.90	1610
E19	77	66	103	1400	1437.00	3.10	1435	1443.67	3.40	1435	1436.00	3.00	1435	1442.67	4.23	1435
E20	56	63	80	950	990.00	0.00	990	990.00	0.00	990	990.00	0.00	990	990.00	0.00	990
E21	57	72	82	1700	1755.50	22.90	1705	1707.33	2.49	1705	1711.67	9.78	1705	1708.00	2.45	1705
E22	54	44	73	1155	1187.50	6.30	1185	1185.00	0.00	1185	1185.00	0.00	1185	1185.00	0.00	1185
E23	93	89	130	1395	1469.00	13.10	1435	1437.50	3.10	1435	1441.17	7.49	1435	1435.50	1.50	1435
E24	97	86	142	1695	1822.20	26.30	1785	1786.50	2.93	1785	1788.33	4.89	1785	1785.83	2.61	1785
E25	26	28	35	655	655.00	0.00	655	655.00	0.00	655	655.00	0.00	655	655.00	0.00	655

Diversity-Driven Stochastic Ranking are domain independent and can be easily adapted for different scenarios.

Table 3.11: Comparison of the experimental results of the four algorithms MAENS, MAENSM, MAENSD, MAENS* on the group F of *BMCV* CARP dataset. For each problem instance the table shows the instance name, the number of vertices, required edges and total edges as well as the best known results in literature respectively in columns *instance*,$|V|$, $|R$, $|E|$, *best*. For each algorithm, the table includes the average fitness of the best solution, its standard deviation and the best result achieved, respectively in columns *avg*,*std* and *best*

instance					MAENS			MAENSD			MAENSM			MAENS*		
	$\|V\|$	$\|R\|$	$\|E\|$		avg	std	best	avg	std	best	avg	std	best	avg	std	best
F1	73	85	105	1065	1071.00	7.90	1065	1072.17	2.79	1065	1072.17	3.34	1065	1071.83	2.73	1065
F2	58	58	81	920	920.00	0.00	920	920.00	0.00	920	920.00	0.00	920	920.00	0.00	920
F3	46	47	61	400	400.00	0.00	400	400.00	0.00	400	400.00	0.00	400	400.00	0.00	400
F4	70	77	99	930	963.50	8.40	940	954.33	4.61	940	955.67	4.42	940	953.67	3.64	940
F5	68	61	94	1180	1180.30	1.30	1180	1180.00	0.00	1180	1180.00	0.00	1180	1180.00	0.00	1180
F6	49	43	66	490	490.00	0.00	490	490.00	0.00	490	490.00	0.00	490	490.00	0.00	490
F7	73	50	94	1080	1090.70	24.50	1080	1103.33	42.30	1080	1080.67	3.59	1080	1080.00	0.00	1080
F8	74	59	98	1135	1145.00	0.00	1145	1145.00	0.00	1145	1145.00	0.00	1145	1145.00	0.00	1145
F9	93	103	141	1145	1197.80	27.60	1145	1168.50	7.54	1160	1171.33	11.90	1155	1161.00	11.79	1145
F10	56	49	76	1010	1010.00	0.00	1010	1010.00	0.00	1010	1010.00	0.00	1010	1010.00	0.00	1010
F11	80	94	113	1015	1037.50	16.30	1015	1028.67	12.84	1015	1030.67	12.70	1015	1030.00	11.11	1015
F12	74	67	103	900	939.50	33.00	910	928.00	26.26	910	930.50	28.65	910	925.00	23.42	910
F13	49	52	73	835	835.00	0.00	835	835.00	0.00	835	835.00	0.00	835	835.00	0.00	835
F14	53	55	72	1025	1065.50	14.40	1025	1036.33	11.69	1025	1049.67	16.68	1025	1037.33	12.23	1025
F15	85	107	126	945	951.70	9.90	945	945.00	0.00	945	945.00	0.00	945	945.00	0.00	945
F16	60	54	80	775	775.00	0.00	775	775.00	0.00	775	775.00	0.00	775	775.00	0.00	775
F17	38	36	50	605	605.00	0.00	605	605.00	0.00	605	605.00	0.00	605	605.00	0.00	605
F18	78	88	110	835	861.20	23.30	850	850.67	2.49	850	851.00	3.00	850	850.67	2.49	850
F19	77	66	103	685	725.00	0.00	725	741.67	11.28	725	726.83	3.76	725	735.17	9.35	725
F20	56	63	80	610	614.80	0.90	610	610.83	1.86	610	611.83	2.41	610	611.00	2.00	610
F21	57	72	82	905	905.00	0.00	905	905.00	0.00	905	905.00	0.00	905	905.00	0.00	905
F22	54	44	73	790	790.00	0.00	790	790.00	0.00	790	790.00	0.00	790	790.00	0.00	790
F23	93	89	130	705	736.30	14.30	725	725.00	0.00	725	725.00	0.00	725	725.00	0.00	725
F24	97	86	142	975	1001.30	17.10	975	1006.00	8.21	980	1002.17	11.38	980	999.33	8.63	980
F25	26	28	35	430	430.00	0.00	430	430.00	0.00	430	430.00	0.00	430	430.00	0.00	430

CHAPTER 4

Dynamic Selection of Evolutionary Operators Based on Online Learning and Fitness Landscape Analysis

We are the children of our landscape;
it dictates behavior and even thought
in the measure to which we are
responsive to it.

Lawrence Durrell, *Justine*

4.1 Introduction

Several of the most popular Credit Assignment (CA) strategies existing in literature rely exclusively on the mere evaluation of the fitness of the offspring. However, the information provided by the fitness at a single generation may not be sufficient to assess the usefulness of an operator (e.g. in a landscape with a high degree of neutrality). The purpose of this chapter is to develop a new dynamic CA mechanism which considers a suite of measures, and that can be adopted also as an Operator Selection Rule. The Memetic Algorithm with

Extended Neighbourhood Search (MAENS*) introduced in chapter 3 is considered as a case study of this research, as it already utilizes an Adaptive Operator Selection scenario which allows the study of other AOS approaches and provides a term of comparison with alternative techniques. Although the hyper-heuristic proposed in this work is applied to the Capacitated Arc Routing Problem, it would be possible to adapt it to different NP-Hard problems by replacing the low level heuristics and by identifying the best Fitness Landscape Analysis metrics that better describe the specific landscapes of the different NP-Hard problem.

The contributions of this work include:

- An ensemble of four different online Fitness Landscape Analysis techniques, performed during the execution of the MAENS* algorithm in order to give a more accurate description of the current population (RQ3);
- A Credit Assignment technique based on the use of an online learning algorithm to predict the reward of the most suitable operator (RQ4);
- Two different Reward Measures are studied: one based on the survival ability of the offspring and another one based on the analysis of their diversity.

The results of the experiments carried out show that the proposed approach is able to produce results with comparable solution quality to a state-of-the-art strategy and reveal how in some cases the presence of a set of measures have a beneficial effect on the optimization ability of the AOS.

The rest of the chapter is organized as follows. Section 4.2 describes the novel Reward Measures and Operator Selection Rules investigated in this work. Section 4.3 describes the ensemble of Fitness Landscape Techniques used in conjunction with the CA mechanism of the MAENS* algorithm. Section 4.4 describes the online Learning algorithm that has been used and adapted for the CA system. Section 4.5 presents the proposed MAENS*-II algorithm. Section 4.6 describes the experiments that have been carried out to verify the assumptions of this research and their results, while Section 4.7 extends the discussion with further comments and analysis.

4.2 Adaptive Operator Selection

As previously mentioned, AOS is conventionally composed of two different sub-tasks, the Credit Assignment and the Operator Selection. For the former part, the use of two different Reward Measures is suggested, named Proportional Reward (PR) and Diversity Based Reward (DBR). For the latter, the chapter proposes the study the performance of two different strategies: a simple Instantaneous Reward (IR) approach and a Concurrent Strategy (CS) based approach.

4.2.1 Credit Assignment

The choice of the proper Credit Assignment Strategy can be fundamental for the performance of the algorithm. As one objective of this work is to evaluate more than just the fitness of the individuals, two different strategies that involve the evaluation of different measurements are adopted. The first one, named Proportional Reward, already described in section 3.4, together with a Multi Armed Bandit approach. the second credit reward consists of a novel measure based on the evaluation of the diversity of the offspring, named Diversity Based Reward.

Proportional Reward (PR)

PR is a measure of the survival ability of the offspring generated by each crossover operator. A reward r is assigned, where $r \in [0, 1]$ corresponds to the percentage of the solutions that have survived the selection phase of the algorithm, and are going to become the parent population for its generation. The use of this technique is a way to entrust the algorithm itself for the evaluation of the offspring. In the case of MAENS*, the offspring able to survive the ranking phase are evaluated according to their fitness value, the amount of violation of the constraints and the average pairwise diversity from the other individuals of the population. The performance of the crossover operator is in this case evaluated at the end of the generation: rather than evaluating the individuals as soon as they are generated, PR evaluates their performance in a longer period of time (e.g. an iteration).

Diversity Based Reward (DBR)

In the case of the DBR, the approach is opposite to that of the PR, as the crossover offspring is evaluated as soon as they have been generated. As one purpose of the crossover operator is that of introducing diversity in the population through the exploration of new areas of the landscape, a measure of the diversity introduced by the offspring is adopted. In particular, for each operator, it is required to measure how distant the offspring are from the parent population, and how wide is the area explored. Therefore, it is possible to define a parent distance measure

$$P_d(x) = \frac{d(x, p_1) + d(x, p_2)}{2} \tag{1}$$

as the average distance from the offspring x to its parents p_1 and p_2 and it is possible to consequently compute the average parent distance for operator i, $P_d(i)$, by averaging the $P_d(x)$ of all the offspring generated by such operator. To measure the distance between individuals, the distance measure for CARP introduced in 3.3 is adopted.

It also possible to define the *coverage* measure of the operator i

$$C_m(i) = \frac{\sum_i \sum_j d(x_a, x_b)}{N_i^2} \tag{2}$$

as the pairwise average distance between any pair of individuals x_a and x_b that have been generated by it, where N_i is the number of individuals generated through the use of operator i. The DBR of the i-th operator can be computed in the following way:

$$DBR(i) = P_d(i) * C_m(i) \tag{3}$$

Similar to Compass [88], this Credit Assignment technique considers the diversity of the offspring as a criterion to evaluate the performance of the operators. However, there a several differences between such approaches. First, the Compass approach addresses the evaluation of both the fitness and the diversity while DBR only considers the diversity, being focused on the evaluation of crossover operators exclusively. Secondly, Compass makes use of the Hamming distance entropy as in [72] to measure the population diversity, while DBR deals with both the average pairwise distance of the offspring as well as the distance from the parent population using the CARP based diversity measure included in algorithm 3.5.

4.2.2 Operator Selection Rule (OSR)

The second step of the AOS process is the Operator Selection Rule. The OSR, given the information gained through the use of the Credit Assignment mechanism, needs to decide what is the most suitable operator and how to use it. A first problem in this context is that of balancing the exploration of all the operators against the exploitation of the most useful one. In other words, while using the operator that has performed the best so far, one wants to verify whether there is another operator that can do better. A second aspect is that of identifying changes during the execution. As the search goes on, the operator that has performed the best so far might not necessarily be the best one afterwards. It is therefore necessary to balance how much of the “history” relative to each operator one most consider to perform the selection.

In this work two different approaches for the OSR are considered, namely a single operator based approach named Instantaneous Reward and a Reinforcement Learning-inspired one called Concurrent Strategy.

Instantaneous Reward (IR)

In the IR approach, the offspring is produced through the use of only one crossover operator per generation. As offspring and parent populations are merged in an unique population, it is still possible to evaluate all the crossover operators who have generated a solution that is still present in the population. The operator to use in the next generation $(t+1)$ is consequently selected as the operator op_i that has obtained the largest reward in the current generation (t):

$$op_i^{t+1} = \max_i(RW(op_i)^t), op_i \in \text{operators} \tag{4}$$

given $RW()$ as a reward measure. Those operators having produced more “extreme” improvements (e.g. discovered new optima) with respect to the others, will have a more favourable evaluation that will last for more generations, even when they have not been selected for the current generation.

The information relative to the previous performances of the operator, except for the last iteration, is discarded. IR is therefore designed to be more sensitive to changes, having a bias on the performance of the operators during the previous generation. Finally, the adoption of such approach has the potential risk of eliminating completely an operator from the competition if none of its offspring are present in the current population.

Concurrent Strategy (CS)

One of the disadvantages of adopting the Instantaneous Reward strategy is that it is not possible to identify changes in the environment when only one operator is used. A different approach, therefore, is that of allowing the use of all the operators during all the generations. Such approach, named CS, aims to maximize the gain obtained by using the best performing operator, and thus allowing the generation of a larger fraction of the offspring by it, while the remaining part is still generated by the other operators. The CS is similar in its behaviour to the Adaptive Pursuit (AP) approach [129] in its intent to maintain a minimum percentage of the solutions to be generated by the less performing operators. The formula to assign the Selection Rate to each operator i, is the following:

$$SR_i = SR_{\min} + (1 - n \times SR_{\min}) \frac{e^{RW(op_i)^t/\psi}}{\sum_{j=1}^{n} e^{RW(op_j)^t/\psi}} \tag{5}$$

where $SR_{\min}$ is the minimum selection rate, n is the number of operators, $RW(op_i)$ is the reward calculated for the operator op_i during the generation t, and ψ is a control parameter that regulates how quickly the system reacts to the changes in the environment. In this case, $n = 4$ since four operators are available, and randomly selected using a roulette wheel approach, to create the offspring during each generation.

4.3 Online Fitness Landscape Analysis

The existing Fitness Landscape Analysis (FLA) techniques have been analysed with the purpose to identify those that can be used in the CARP context. Such selection has been driven by both the necessity to reduce the computational effort by exploiting some calculations that are already performed by the algorithm, and the necessity to identify measures able to "capture" different features of the landscape. Four FLA techniques have been identified, consisting of one evolvability measure, two neutrality measures and one fitness distribution measure, to describe different features of the landscape and without much increasing the computational effort. The computation of such techniques is based on the evaluation of the neighbourhood of each solution. Such neighbourhoods are already generated through the initial iteration of the local search operator of the MAENS algorithm, by using of the three different move operators involved in this process (Single Insertion, Double Insertion, Swap Insertion). The FLA techniques are employed during each generation, and their results are used as input features of an online learning algorithm to predict the value of one of the two Reward Measures introduced in section 4.2.1, in order to create a more accurate and informative “snapshot” of the current population which eventually might lead to a better selection of the crossover operator. A final remark is necessary about the constraints handling and how it affects the fitness of the individuals. The landscape in which MAENS* operates is that of solutions which may potentially violate the capacity constraints of the vehicles. Therefore, the following fitness function is considered, adopted from [128]:

$$f(S) = TC(S) + \lambda * TV(S) \tag{6}$$

where λ is an adaptive parameter depending on the cost, on the violation and on the best feasible solution found so far, $TC(S)$ is the total cost of the solution and $TV(S)$ its total violation.

The rest of this section will introduce the four FLA techniques that have been considered in this work and how they are integrated in the MAENS* algorithm.

4.3.1 Accumulated Escape Probability

The Accumulated Escape Probability [81] is a metric that aims to measure the evolvability, which can be defined as the capacity of the solutions to evolve into better solutions. The Accumulated Escape Probability is obtained by averaging the mean escape rate [96] (the proportion of solutions with equal or better fitness in the neighbourhood) of each fitness level with the formula:

$$aep = \frac{\sum_{f_j \in F} P_j}{L}, \text{ where } F = f_0, f_1, ..., f_{L-1} \tag{7}$$

where f_i is a fitness level (subset of all the solutions with fitness equal to a value f_i), P_j is the average Escape Rate of all samples belonging to the f_j fitness level and L is the number of possible fitness levels. Being the mean value of a set of probabilities, the aep will be 0 when the instance is hard and higher (up to 1) otherwise. The calculation of the aep requires the analysis of the neighbourhood of each solution in order to identify how many individuals have a equal or better fitness than the original individual. An analysis of the evolvability of the solutions which have been selected for the local search is therefore performed. Since the calculation of the neighbourhood of each solution corresponds to the first step of the local search, no significant additional cost is required to compute the aep.

4.3.2 Dispersion Metric

The analysis of the distribution of the solutions within the landscape can be sometimes used to understand more about the difficulty that a "jump" between fitness levels requires and to gain some information on the global structure of the landscape. In this context, the Dispersion Metric (dm) [82] is a technique to obtain information about the global structure of the landscape, by measuring the dispersion of good solutions. Ideally, if good solutions are very close it would be possible to have a single funnel structure. If, on the contrary, solutions get more distant when their fitness improves, the landscape might be more like a multi funnel structure. The analysis can be described as follows:

1. A sample S of solutions is taken from the search space;
2. the best S_{best} solutions are selected from S (using a threshold value);
3. the average pairwise distances in S ($\overline{d}(S)$) and in S_{best}($\overline{d}(S_{best})$) are calculated using the CARP diversity measure shown in Figure 3.5;
4. the dm is obtained as the difference between $\overline{d}(S_{best})$ and $\overline{d}(S)$.

The calculation of the pairwise distance between all the individuals of the sample is already performed during the diversity-based Stochastic Ranking of MAENS* by using the distance measure shown in figure 3.5, and therefore requires no additional cost. Thus, the dm can computed on the set of all the *psize*$*$*offset* individuals created during each generation of MAENS*. Finally, it is possible to rely on the ranking performed by the diversity-based Stochastic Ranking operator and choose these solutions as the subset of the best ones.

4.3.3 Average Neutrality Ratio and $\Delta-$fitness

Neutrality is the study of the width, distribution and frequency of neutral structures within a landscape (e.g. plateaus, ridges). A set of several neutrality measures was defined in [133]. Among these, the following two are selected:

1. average neutrality ratio ($\overline{r}$): can be obtained by averaging the neutrality ratio (e.g. the number of solutions with equal fitness) of each individual with respect to its neighbourhood;

2. average $\Delta-$fitness of neutral network ($\Delta(\overline{f})$): can be defined as the average fitness gain after one mutation step of each individual belonging to a neutral network.

In the same fashion as in the case of the aep, the computational effort of this technique can be absorbed by the generation of the neighbourhood of the initial solution during the local search.

4.4 Online learning

The AOS model followed by MAENS* is that of the Multi Armed Bandit approach, where the UCB1 [2] algorithm is used to balance the exploration and exploitation of the crossover operators and the Page-Hinckley [57] test is used to detect when a different operator has become the most suitable.

In this work, the adoption of a different model is proposed. The abrupt and scarcely predictable changes of the most suitable operator which might happen during the search show many similarities to the notion of concept drift [122] [97] in machine learning. Thus, in such a context, it might be possible to adopt an online learning algorithm capable of (a) predicting a reward for each operator using the online Fitness Landscape Analysis measures and (b) tracking the changes of the environment, relying only on a limited number of training instances. The learning problem can be defined more formally in the following way. At a given generation of the EA, the FLA metrics are computed $(fla_1, fla_2, fla_3, fla_4)$ as well as the reward of each operator $(RW(op_i))$. Tuples $(fla_1, fla_2, fla_3, fla_4, RW(op_i))$ are then used as training examples for the online learning algorithm, where $(fla_1, fla_2, fla_3, fla_4)$ are the input features and $(RW(op_i))$ is the target output.

The Dynamic Weighted Majority (DWM) [68] algorithm is then employed as an online learning algorithm, which has proved to be one of the most effective techniques in the task of tracking concept drifts. The DWM algorithm, shown in algorithm 4.1, can be described as follows. A set of learners (called experts) are used to classify the incoming instances $\{\overrightarrow{x}, y\}$, where $\overrightarrow{x}$ is the vector of n input features and y is the output feature (lines 1-3). Each expert e_j has its own weight w_j, and operates a classification λ of the instance. The global prediction is identified as the prediction with the largest sum of weights (line 9). All the experts which have failed to classify correctly the instance have their weights reduced by a β factor (lines 4-6). Moreover, for every p instances, all the experts with a weight below a certain threshold θ are deleted (lines 10-15) and a new expert e^new is created if the global prediction is wrong (lines 16-18). All the surviving experts are then trained with the new instance (lines 20-22).

4.4.1 DWM for the Regression Task

As the DWM algorithm was originally conceived for classification it is necessary to adapt and modify some of its mechanism for the regression task of predicting the reward of a given operator based on the FLA techniques. The pseudo-code of the DWM algorithm is included in algorithm 4.1, while the pseudo code of DWM for the regression task (rDWM) is given in algorithm 4.2. The modifications introduced are:

1. The global prediction σ_i is obtained by calculating the weighted average of all predictions (line 10);

2. A prediction is considered correct if its difference from the output feature is less than a threshold τ (lines 5-6);

3. a new expert is created if the difference between the global prediction and the output feature is less than a t factor (lines 17-18);

4. A window containing the last n instances wTS is introduced in order to train the new experts upon creation (line 2).

4.5 MAENS*-II

The revised version of the algorithm adopting the rDWM as an AOS mechanism, named MAENS*-II, is provided in algorithm 4.3. A set of four (one for each crossover operator) rDWM instances are created upon initialization of the algorithm (line 2). During each generation, one new training example is created for each rDWM instance by using the current set of FLA metrics as input features, and the reward associated to the operator as the output feature (lines 10, 13-14) obtained with a given Credit Reward strategy. The set of four rDWM instances are then used to predict the reward of each operator (line 4).

Algorithm 4.1: Dynamic Weighted Majority

```
for (each instance $\{\overrightarrow{x}_i, y_i\}$) do
    for (each expert $e^j$) do
        $\lambda^j$ =classify($e^j$, $\overrightarrow{x}_i$);
        if ($|\lambda_i^j \neq y_i$) then
            $w_j = \beta * w_j$;
        end
    end
    normalize weights;
    $\sigma_i$= select class with largest sum of weights;
    if ($p$ mod $i = 0$) then
        for (each expert $e^j$) do
            if ($w_j < \theta$) then
                delete expert $e^j$;
            end
        end
        if ($\sigma_i \neq y_i$) then
            create new expert $e^new$;
        end
    end
    for (each expert $e^j$) do
        train($e^j$, $\overrightarrow{x}_i$);
    end
end
```

Finally, an Operator Selection Rule is adopted to choose the operators to use during each generation.

Three different versions of the MAENS*-II algorithm were implemented employing the two different techniques for the Operator Selection Rule introduced in section 4.2.2 as well as the two different Credit Assignment mechanism presented in section 4.2.1. All the algorithms implemented consider the *Weka* [51] implementation of REPTrees as base learners for the DWM algorithm. Table 4.1 summarizes a list of the different versions of the algorithm and a description of their components. It is worth noting that the combination of the DBR strategy and the Instantaneous Reward was not considered, as the strategy of measuring the reward of the crossover offspring and the use of only one

Algorithm 4.2: Dynamic Weighted Majority for the regression task

```
for (each instance $\{\overrightarrow{x}_i, y_i\}$) do
    update wTS($\overrightarrow{x}_i$);
    for (each expert $e^j$) do
        $\lambda^j$ =predict($e^j$, $\overrightarrow{x}_i$);
        if ( $|\lambda_i^j - y_i| > tau$ ) then
            $w_j = \beta * w_j$;
        end
    end
    normalize weights;
    $\sigma_i$= global prediction
    if ($p$ mod $i = 0$) then
        for (each expert $e^j$) do
            if ($w_j < \theta$) then
                delete expert;
            end
        end
        if ( $|\sigma_i - y_i| > t$ ) then
            create new expert and train using wTS;
        end
    end
    for (each expert $e^j$) do
        train($e^j$, $\overrightarrow{x}_i$);
    end
end
```

operator during each generation would lead to the its exclusive use for the whole execution of the algorithm.

4.5.1 Improvements on Local Search Efficiency

One of the most effective features of MAENS [128] is its Local Search, which, however, has a high computational cost - the algorithm spends around 95% of its runtime performing this operation. Although the proposed modifications to the original MAENS algorithm, as explained in section 4.3, cause no significant increment of the runtime of the algorithm, it is possible to introduce a fast implementation of MAENS local search, which helped

```
Algorithm 4.3: MAENS*-II algorithm
1  initialize a population $pop$ of $psize$ individuals;
2  initialize four rDWM$_i$ instances;
3  initialize four rewards $rw_i$;
4  $t = 0$;
5  while $(t < G_m)$ do
6      choose the crossover operator $op_i$ with largest $rw_i$;
7      generate a population $pop_t$of $opsize$ individuals, choosing the parents from
        $pop_t \cup pop_{t-1}$;
8      generate $pop_{ls}(i)$ for each individual $pop_t(i)$ with probability $= P_{ls}$;
9      if $(pop_{ls}(i)$ is better than $pop_t(i))$ then
10         replace $pop_t i$;
11     end
12     calculate $Y^t = \{aep,\bar{r},\Delta(\bar{f}),dm\}$;
13     use d-Stochastic Ranking and overwrite $pop_t$ for each $op_i$ do
14         $out_i = CreditAssignment(op_i)$;
15     end
16     $rw_i =$ rDWM$_i(Y^t, out_i)$;
17 end
18 return best individual from $pop_{G_m}$;
```

Table 4.1: List of MAENS*-II combinations. Each row represents a different combination of one Operator Selection Rules and one Reward Measure strategies.

combination	Operator Selection Rule	Reward Measure
a	Concurrent Strategy	Diversity Based Reward
b	Concurrent Strategy	Proportional Reward
c	Instantaneous Reward	Proportional Reward

reducing effectively the runtime without incurring into extra memory consumption. The approach is similar to the one introduced in [150] for the Vehicle Routing Problem, but without relying on the use of memory.

The approach can be summarized by the following points:

1. Every individual a in the neighbourhood of a solution x is represented as a *move* M, where M stores the information relative to the move operator op_i such that $op_i(x) = a$, the tasks involved in the move, and the variations in terms of fitness

and violation of the constraints w.r.t. the values of the initial solution;

2. The set of moves representing the whole neighbourhood is split into $V = |R| * |R|$ subsets, where $|R|$ is the number of routes of the initial solution. Each subset contains the moves relative to the move operators applied to the tasks belonging to the routes R_i and R_j;

3. During the first iteration of the local search, the whole neighbourhood of moves is produced. A storage array of size V is kept to store the best solution of each subset;

4. The best move in the neighbourhood is identified with a computational time of $O(M)$. If the best move belongs to the subset relative to the routes R_i and R_j, the positions in the storage relative to the combination of either routes are set to null;

5. In the following iterations, the local search produces only the moves involving either the route R_i or R_j or both. The positions of the storage relative to such moves are consequently updated.

After the first iteration, the number of subsets to be evaluated is therefore decreased from $|R|^2$ to $2|R| + 1$, resulting in a significant reduction in terms of size of the neighbourhood that is necessary to evaluate during each local search iteration. It is worth mentioning that the use of the *move* notation itself reduces the cost of evaluating the fitness and the violation of one individual from $O(n)$ to $O(k)$, where n is the number of tasks and k is equal either to 7 or 8 (depending on the move operator considered).

4.6 Experimental Studies

A set of experiments were designed to understand the behaviour of MAENS*-II. As a first step, an oracle based on the Proportional Reward was implemented with the purpose of analysing a set of CARP instances in order to obtain optimal crossover operator selection rates and to analyze them. The oracle can be briefly described as follows. Four different populations are obtained during each generation by using each crossover operator. All the individuals of the four generations are merged into a single population which is sorted using the MAENS* ranking operator. The Proportional Reward mechanism is therefore used to assess the best operator. The results achieved by the oracle show that the predictions operated by the dMAB are not optimal, as better results can be achieved. Besides, the results of the oracle should be considered “optimal” only when the Proportional Reward strategy is considered, because they might not necessarily be optimal when in presence of a set of multiple measures, as in the case of MAENS*-II, or when a different credit assignment strategy is considered.

Two different datasets are considered for the experiments. The first one, named Group A, is composed of instances taken from the known benchmark test sets egl [28], BMVC *C, D, E, F* [9] and val [8]. The second group (Group B) corresponds to the large scale CARP instances of the dataset EGL-G [12]. The characteristics relative to each instance, in terms of number of vertices, number of edges, number of required edges and best fitness value found in literature are included in table 4.2.

An example of a solution for the *D07* problem instance produced by one of the variants of the MAENS*-II algorithm is provided in figure 4.1, to clarify what kind of results this algorithm produces.

The set of parameters adopted in all the MAENS*-II algorithm variants, included in table 4.3, were identified by a series of test-and-trial attempts and might not correspond to the best configuration. Even in this case, the strategy of maintaining the same parameter setting of the MAENS and MAENS* is necessary to prevent having results that might be caused by other factors (e.g. different parameter configuration), and due to the enormous computational cost that a parameter optimization would require for algorithms with this many parameters as it is the case of MAENS, MAENS* and all its variants.

Table 4.2: Characteristics of the instances of groups A (top part) and B (bottom part). For each instance the table provides the number of vertices of the graph ($|V|$), the number of required edges or tasks ($|R|$), the number of edges of the graph ($|E|$) and the best known solution in literature (BK)

instance	$\|V\|$	$\|R\|$	$\|E\|$	BK	instance	$\|V\|$	$\|R\|$	$\|E\|$	BK
C1	69	79	98	1590	e3-C	77	87	98	10163
C10	60	55	82	2190	e4-A	77	98	98	6408
C11	83	94	118	1725	e4-B	77	98	98	8884
C15	97	107	140	1765	e4-C	77	98	98	11427
C18	93	121	133	2315	S1-B	140	75	190	6384
C5	56	65	79	2410	S2-A	140	147	190	9824
C6	38	51	55	855	S2-B	140	147	190	12968
C9	76	97	117	1775	s2-C	140	147	190	16353
D1	69	79	98	725	s3-A	140	159	190	10143
D11	83	94	118	920	s3-B	140	159	190	13616
D21	60	76	84	695	s3-C	140	159	190	17100
D23	78	92	109	715	s4-A	140	190	190	12143
D7	54	52	70	735	s4-B	140	190	190	16093
D8	66	63	88	615	s4-C	140	190	190	20375
E1	73	85	105	1855	F1	73	85	105	1065
E11	80	94	113	1810	F11	80	94	113	1015
E12	74	67	103	1580	F12	74	67	103	900
E15	85	107	126	1555	F14	53	55	72	1025
E19	77	66	103	1400	F19	77	66	103	685
E21	57	72	82	1700	F24	97	86	142	975
E23	93	89	130	1395	F4	70	77	99	930
E5	68	61	94	2130	F7	73	50	94	1080
E9	93	103	141	2160	F9	93	103	141	1145
e1-B	77	51	98	4498	val4D	41	69	69	526
e2-B	77	72	98	6305	val5D	34	65	65	573
e3-B	77	87	98	7704	val8C	30	63	63	518
					val10D	50	97	97	525

instance	$\|V\|$	$\|R\|$	$\|E\|$	BK	instance	$\|V\|$	$\|R\|$	$\|E\|$	BK
EGL-G1-A	255	347	375	970495	EGL-G2-A	255	375	375	1061103
EGL-G1-B	255	347	375	1085097	EGL-G2-B	255	375	375	1173286
EGL-G1-C	255	347	375	1201030	EGL-G2-C	255	375	375	1295036
EGL-G1-D	255	347	375	1325317	EGL-G2-D	255	375	375	1430267
EGL-G1-E	255	347	375	1461469	EGL-G2-E	255	375	375	1557159

These parameters can be identified in table 4.3. All the final results were obtained by averaging the output of 30 independent runs.

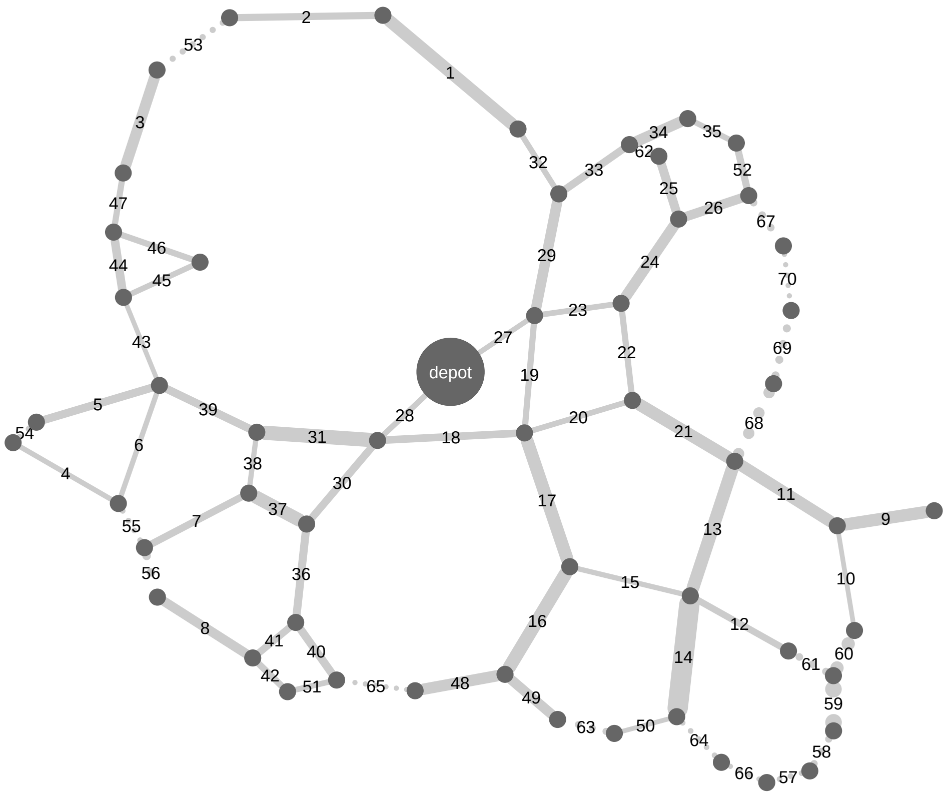
Table 4.3: Parameters of the MAENS*-II algorithms. The upper part of the table includes the set of parameters and the respective values that are shared with the MAENS* algorithm. The bottom part of the table shows the new parameters introduced for the MAENS*-II algorithm.

MAENS* parameters		
Name	Description	Value
psize	population size	30
ubtrial	maximum attempts to generate a solution	50
opsize	size of the offspring during each generation	6*psize
parent selection	crossover parent selection strategy	random selection
P_{ls}	probability of performing the local search	0.2
pMS	routes selected during MergeSplit	2
G_{max}	maximum generations	500
SR_{r1}	probability of sorting solutions using diversity	0.25
SR_{r2}	probability of sorting solutions using fitness	0.70
MAENS*-II parameters		
Name	Description	Value
p	expert removal period	5
β	decrease factor for expert weights	0.75
τ	expert weight reduction threshold	0.05
θ	threshold for expert removal	0.05
t	threshold for expert creation	0.10
ψ	control parameter for concurrent strategy	0.002

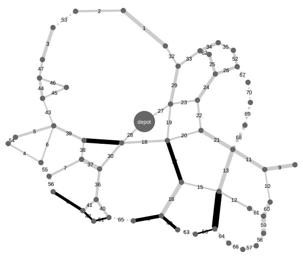
4.6.1 Single Operator Scenario

In order to understand the improvement achievable by MAENS*-II, the algorithm was executed on the two benchmark sets considering each of the four available crossover operators. The results of such experiments for group A are included in table 4.4. For each single operator MAENS* version, the results show the average fitness over 30 independent runs, the standard deviation and the fitness of the best individual found. The last column, named *best*, shows the results of what an “optimal” adaptive operator selection would achieve (picking the best results out of the four achieved). In the second to last row (named #), the table provides the number of instances with statistically different results according to the results of a Wilcoxon Rank-Sum test with Holm-Bonferroni correction

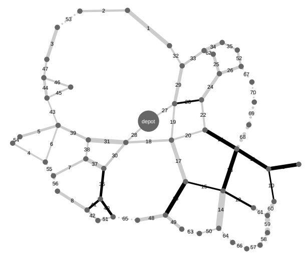
Figure 4.1: The network relative to the *D07* instance (a). The cost of serving each edge is proportional to the thickness of the line. Non required edges can be identified by dotted lines. In figures b, c, d and e the four routes that compose a solution for this problem instance, generated by the MAENS*-IIa algorithm. The edges served during each route have been highlighted in black.



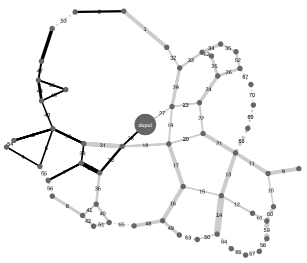
(a) *D07* instance network



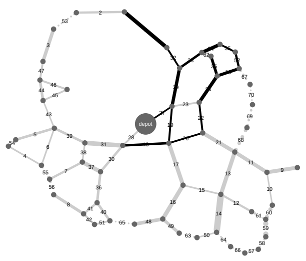
(b) route 1



(c) route 2



(d) route 3



(e) route 4

Table 4.4: Experimental results on group A using a single crossover operator. The first column shows the instance name (*inst*). For each operator (one among GSBX,GRX,PBX,SPBX) the table includes the average fitness of the best solution (*avg*), the standard deviation (*std*) and the best solution (*best*). Last column (*best*) shows the best *avg* result among the four crossover operators

	GSBX			GRX			PBX			SPBX			optimal AOS
inst	avg	std	best	avg	std	best	avg	std	best	avg	std	best	avg
C01	4175.33	26.80	4150	4159.00	14.80	4150	4151.67	4.53	4150	4153.17	7.13	4150	4151.67
C05	5373.75	15.29	5365	5366.00	2.00	5365	5366.50	2.29	5365	5367.50	2.50	5365	5366.00
C06	2545.75	4.82	2535	2537.50	4.61	2535	2541.00	4.90	2535	2540.33	4.99	2535	2537.50
C09	5274.92	19.27	5260	5281.17	19.57	5260	5262.83	9.97	5260	5263.33	9.52	5260	5262.83
C10	4709.42	16.26	4700	4709.00	12.21	4700	4712.33	10.14	4700	4703.33	7.45	4700	4703.33
C11	4648.58	16.33	4640	4643.17	3.29	4640	4641.33	2.21	4640	4641.50	3.20	4630	4641.33
C15	4964.42	16.43	4940	4946.83	10.76	4940	4946.50	5.19	4940	4946.33	4.27	4940	4946.33
C18	5639.83	9.83	5620	5642.17	6.28	5620	5635.00	8.37	5620	5640.17	9.26	5620	5635.00
D01	3225.00	9.04	3215	3235.00	0.00	3235	3230.83	5.01	3215	3229.67	6.45	3215	3225.00
D07	3115.83	2.76	3115	3115.33	1.80	3115	3115.00	0.00	3115	3115.67	2.49	3115	3115.00
D08	3052.83	4.12	3045	3058.00	10.38	3045	3045.67	2.49	3045	3047.67	4.42	3045	3045.67
D11	3761.08	3.17	3760	3760.33	3.14	3755	3760.83	2.27	3755	3760.17	3.76	3745	3760.17
D21	3058.33	3.25	3050	3056.67	2.98	3050	3059.83	2.41	3050	3060.00	2.24	3055	3056.67
D23	3171.17	11.74	3140	3158.17	8.51	3145	3187.17	10.30	3155	3177.50	10.47	3145	3158.17
E01	4916.83	5.84	4910	4910.33	1.25	4910	4912.00	3.32	4910	4912.67	4.23	4910	4910.33
E05	4623.33	21.67	4585	4623.33	27.34	4585	4607.33	19.09	4585	4608.67	18.39	4585	4607.33
E09	5855.33	25.00	5820	5838.83	23.55	5810	5836.50	19.88	5815	5832.67	17.83	5810	5832.67
E11	4697.25	24.91	4660	4671.67	4.35	4670	4675.00	11.25	4670	4678.33	12.67	4670	4671.67
E12	4228.50	17.28	4190	4209.33	19.09	4190	4200.67	11.95	4190	4201.83	9.79	4195	4200.67
E15	4220.67	7.04	4205	4215.00	6.06	4210	4221.50	5.19	4210	4220.00	6.19	4210	4215.00
E19	3244.17	2.76	3235	3239.33	4.96	3235	3244.67	1.80	3235	3243.83	3.08	3235	3239.33
E21	3733.50	2.29	3730	3731.33	2.21	3730	3734.50	1.50	3730	3734.00	2.00	3730	3731.33
E23	3718.83	5.87	3715	3715.17	1.57	3710	3715.33	1.80	3710	3717.33	2.81	3710	3715.17
egl-e1-B	4512.47	12.04	4498	4504.87	10.94	4498	4503.03	8.78	4498	4501.77	8.41	4498	4501.77
egl-e2-B	6328.65	11.75	6317	6321.93	8.68	6317	6322.60	6.03	6317	6327.10	10.19	6317	6321.93
egl-e3-B	7792.07	15.71	7775	7780.90	9.45	7775	7784.97	8.83	7777	7782.57	4.96	7777	7780.90
egl-e3-C	10328.18	19.82	10292	10324.73	16.07	10305	10315.87	17.93	10292	10310.67	16.24	10292	10310.67
egl-e4-A	6464.97	5.39	6444	6464.23	4.26	6461	6463.47	3.00	6456	6463.90	1.83	6461	6463.47
egl-e4-B	9021.28	17.37	8988	9059.70	25.38	8988	9024.40	15.16	8998	9013.10	14.09	8988	9013.10
egl-e4-C	12032.60	1047.53	11559	11593.13	22.87	11554	11586.40	18.91	11539	11584.97	25.66	11543	11584.97
egl-s1-B	6415.70	21.28	6388	6405.50	19.41	6388	6393.17	2.48	6388	6399.43	13.95	6388	6393.17
egl-s2-A	9942.62	26.54	9895	9949.67	24.62	9890	9929.37	23.69	9889	9939.50	27.44	9889	9929.37
egl-s2-B	13201.76	35.16	13144	13244.63	90.51	13137	13163.57	30.84	13103	13181.60	29.65	13122	13163.57
egl-s2-C	16500.12	42.51	16430	16480.80	39.54	16430	16456.77	17.46	16430	16462.37	27.46	16430	16456.77
egl-s3-A	10298.59	31.43	10221	10305.10	45.56	10242	10284.60	25.31	10233	10300.73	25.47	10253	10284.60
egl-s3-B	13847.47	60.89	13713	13906.90	50.92	13771	13792.63	40.81	13713	13815.87	61.71	13707	13792.63
egl-s3-C	17317.86	38.59	17209	17290.47	41.79	17197	17292.90	39.24	17242	17287.70	37.00	17221	17287.70
egl-s4-A	12409.36	41.76	12296	12438.10	33.13	12389	12367.40	38.27	12315	12399.70	34.21	12316	12367.40
egl-s4-B	16448.97	43.34	16316	16499.40	45.08	16430	16384.27	43.71	16292	16405.10	40.22	16329	16384.27
egl-s4-C	25200.21	2151.47	20781	22201.00	143.44	21792	20801.63	103.60	20601	20796.70	112.85	20584	20796.70
F01	4046.81	3.04	4040	4047.00	3.32	4040	4047.50	3.10	4040	4046.67	3.50	4040	4046.67
F04	3499.47	5.10	3485	3500.17	3.76	3495	3498.83	5.43	3485	3500.50	5.06	3485	3498.83
F07	3347.73	33.33	3335	3338.33	17.95	3335	3338.33	17.95	3335	3355.00	40.00	3335	3338.33
F09	4750.71	9.95	4730	4743.00	7.48	4730	4751.67	12.20	4730	4753.83	10.70	4740	4743.00
F11	3850.38	13.65	3835	3845.83	11.91	3835	3839.83	6.89	3835	3844.83	10.99	3835	3839.83
F12	3416.14	28.91	3395	3450.33	21.21	3395	3423.67	22.10	3395	3410.17	16.71	3395	3410.17
F14	3339.29	11.03	3330	3349.33	18.25	3330	3349.00	6.88	3340	3359.33	14.13	3330	3339.29
F19	2553.97	9.38	2525	2531.17	10.46	2525	2565.33	9.21	2545	2564.83	12.88	2525	2531.17
F24	3234.38	11.30	3215	3230.17	13.01	3210	3245.50	7.89	3225	3244.17	6.20	3220	3230.17
val10D	531.47	1.79	528	532.73	1.26	530	530.73	1.18	528	530.50	1.52	528	530.50
val4D	532.03	3.01	530	531.63	2.44	530	530.27	0.85	530	530.00	0.00	530	530.00
val5D	583.37	1.83	579	586.80	3.53	578	581.87	2.38	575	583.60	2.42	579	581.87
val8C	524.37	1.96	521	526.80	1.08	525	524.50	1.84	521	525.00	1.83	521	524.37

at the significance level of 0.05. The row at the bottom (named W) shows the number of comparison won against the other algorithms. A Wilcoxon Rank-Sum test was performed on the results achieved on every instance by each pair of algorithms, with Holm-Bonferroni correction to deal with the multiple comparisons. The results across all the problem instances were then compared using the Wilcoxon Signed-Rank test. Each problem instance with comparable results was treated as paired results and therefore omitted from the test. The results of such test, subject to the Holm-Bonferroni correction, are included in table 4.6.

For Group A, it is possible to notice how there is a great number of instances for which the four versions achieve statistically different results. The only exception is represented by the comparison between the *PBX* and the *SPBX* based versions, for which there is a limited number of statistically different instances (7); in all the other cases, the statistically different instances are at least 24. The GSBX-based operator seem to be the one performing the worst, losing the comparison to most of the instances, while the other three operators achieve the best results in a similar number of times. The statistical difference between the results of the GSBX-version and the other three versions is also confirmed by the results of the Wilcoxon Signed-Rank test. None of the four single operator based versions of MAENS* algorithm is able to perform as good as the results achieved by the "optimal" AOS strategy (in the *best* column), as testified by the results of the Wilcoxon Signed-Rank test included in the *pBest* row in table 4.4.

The results of the comparison for the CARP instances belonging to the group B are included in table 4.5. The table shows the results of the four different versions of the algorithm, based on the use of one of the four crossover operators available. Analogously to the previous, the table presents the average fitness (*best*), the standard deviation (*std*) and the best result found (*best*) for each algorithm. The results show how the best results are achieved always by the *PBX* and *SPBX* based versions of the algorithm (providing

Table 4.5: Experimental results on Group B using alternatively each crossover operator. The results of the four versions of the algorithm are split in two different rows. The first column shows the instance name (*inst*). For each operator (one among GSBX,GRX,PBX,spxb) the table includes the average fitness of the best solution (*avg*), the standard deviation (*std*), the best solution (*best*). In the rows at the bottom, the number of comparisons (#) against every operator with statistically different results. The following row shows the number of comparisons where the algorithm achieves a better average fitness.

	GSBX			GRX		
inst	avg	std	best	avg	std	best
EGL-G1-A	983408.50	3866.28	974054	979095.97	4718.81	968747
EGL-G1-B	1091041.60	5539.83	1081857	1094405.97	7413.40	1079590
EGL-G1-C	1213921.77	5924.27	1198728	1212862.87	7906.85	1198393
EGL-G1-D	1337645.60	6620.72	1322885	1336158.93	6527.20	1326125
EGL-G1-E	1473433.10	5394.11	1461472	1472399.20	6338.61	1461155
EGL-G2-A	1107790.40	5057.72	1099946	1138050.43	10215.79	1110914
EGL-G2-B	1225440.43	5982.44	1214762	1255569.68	16865.83	1224099
EGL-G2-C	1359221.30	3798.77	1349981	1404364.32	10734.40	1369046
EGL-G2-D	1497934.97	6544.95	1486595	1563310.77	5689.30	1552126
EGL-G2-E	1641472.67	8022.86	1626564	1713877.53	8561.59	1687159
	GRX	PBX	SPBX	GSBX	PBX	SPBX
#	6	7	7	6	8	7
W	5	0	0	1	0	0
	PBX			SPBX		
inst	avg	std	best	avg	std	best
EGL-G1-A	980746.30	5681.29	970911	**978411.33**	4308.05	969682
EGL-G1-B	1087682.37	5542.29	1074857	**1087255.93**	4986.75	1079899
EGL-G1-C	**1208196.23**	5138.15	1198557	1210928.20	5972.77	1202072
EGL-G1-D	**1330286.83**	6554.80	1321271	1333503.57	6436.41	1324605
EGL-G1-E	**1462940.17**	5293.51	1452158	1467270.13	5003.86	1458893
EGL-G2-A	**1104884.20**	3252.86	1099756	1105976.47	3662.89	1098458
EGL-G2-B	1221379.87	3586.68	1213622	**1220895.90**	4180.51	1212440
EGL-G2-C	**1351635.77**	5294.39	1343015	1354111.13	5087.04	1343399
EGL-G2-D	**1490662.50**	4435.48	1484014	1492414.43	4495.21	1484208
EGL-G2-E	1635578.87	4851.51	1623322	**1634917.43**	5199.54	1623417
	GSBX	GRX	SPBX	GSBX	GRX	PBX
#	7	8	1	7	7	1
W	7	8	1	7	7	0

better results on all statistically different instances against the other two versions). The *GRX* based version, in contrast, is the one that performs the worst (losing the comparison 5 times out of 6 against the *GSBX* based version and on all the statistically different instances for the other two versions).

In both datasets, SPBX and PBX operators appear to be the operators whose usage leads to the best results. This can be explained by the fact that such operators can introduce a fair amount of diversity in the offspring as one or more routes are built from scratch. On the other hand, they maintain the good traits of the parents copying the routes that are not affected by the recombination. The GSBX operator, in contrast, might not introduce much diversity in the offspring as the new routes are a combination of the subroutes of the parents. Therefore, despite being the least disruptive operator, on the long term it produces a minor contribution than SPBX and PBX. The GRX operator, on the other hand, has a larger disruptive capacity as only the best routes are preserved in the offspring. In the context of large instances with a great number of routes as in the case of dataset B, therefore, this operator might introduce an excessive level of exploration and consequently perform worse than the others.

A further experiment was conducted to analyse the behaviour of the four crossover operators. A population of 10000 solutions was generated using the initialization operator. Each operator was then used to generate a population of 10000 solutions, using a random parent selection mechanism. Table 4.7 reports the number of instances for which the operators achieved the worst results in terms of fitness, violation, average pairwise distance of the offspring population and average distance from the parents. This experiment has been repeated for both datasets.

The results show that in dataset A the operator GSBX achieves the worst results in the largest number of instances for each of the characteristics analysed. This is coherent with the results achieved by the four evolutionary algorithms. On the other hand, for dataset B, GRX is the worst algorithm for both fitness and violation and the second worst for both the diversity measures, which reflects the behaviour of the algorithm. It is however worth specifying that these results refer to the behaviour of the operators with a population of low quality solutions (as they are generated through the use of the initialization operator) and might not necessarily reflect the behaviour of the crossover operators during the most advanced phases of the search.

Table 4.6: Number of comparisons (#) against every operator with statistically different results of table 4.4. The following row (*W*) shows the number of comparisons where the algorithm achieves a better average fitness. In the last rows (*p* and *pBest*), the p-values relative to the test performed between each single based version against the column of the best results included in table 4.4

	GRX	PBX	SPBX	GSBX	PBX	SPBX	GSBX	GRX	SPBX	GSBX	GRX	PBX
#	26	31	24	26	26	27	31	26	7	24	27	7
W	8	6	5	18	8	13	25	18	3	19	14	4
p	0.126	0.0004	0.0021	0.126	0.1738	0.3524	0.0004	0.1738	-	0.0021	0.3524	-
	GSBX			GRX			PBX			SPBX		
pBest	< 0.001			0.002			0.005			0.007		

Table 4.7: Summary of the performances of the four crossover operators for datasets A and B. The table shows the number of instances for which each of the four crossover operators (GSBX, GRX, PBX, SPBX) achieved the worst results in terms of fitness, violation, average pairwise diversity and distance from parents. The values are obtained averaging the statistics of four different populations of size 10000 generated using each crossover respectively from the same parent population of 10000 individuals.

Dataset A	GSBX	GRX	PBX	SPBX
Fitness	18	14	14	12
Violation	17	12	13	16
Diversity	17	16	11	16
Parent Distance	16	12	13	15
Dataset B	GSBX	GRX	PBX	SPBX
Fitness	3	3	3	1
Violation	3	6	1	0
Diversity	4	3	0	3
Parent Distance	0	3	4	3

4.6.2 Operator Selection Rules and Reward Measures: a Comparison

The performance of the algorithm using different Operator Selection Rules and Reward Measures is shown in tables 4.8 and 4.9, respectively for the groups A and B. The results of the three combinations introduced are included in table 4.1, along with the optimal result considering the best performance of the single-operator versions (in the last column, named *best*). In table 4.8, the results of the statistical tests show how the three versions of the algorithm achieve statistically different results only on a limited subset of the instances (at most 7 between MAENS*-IIa and MAENS*-IIc). The versions achieving the best results appear to be the ones adopting the Concurrent Strategy as an OSR (MAENS*-IIa and MAENS*-IIb). The two versions achieve extremely similar results (differing only in 3 instances), while the version using the Instantaneous Reward (MAENS*-IIc) differs from the other two respectively in 7 and 5 instances, and loses the comparison in the majority of the cases. In contrast, MAENS*-IIc is the variant that differs the least w.r.t. the best results achieved by the single-operator versions of the algorithm (only 6 statistically different instances, while the other two versions differ in 8 ad 9 instances).

Although such results show small differences between the performances of the algorithms when adopting one OSR rather than the other, it is possible to see how the Concurrent Strategy appears to perform slightly better. This might be explained by several factors. First, the use of more than one crossover operator might introduce higher diversity in the whole offspring population. Secondly, the capacity of monitoring and verifying the performance of all the crossover operators might be important to detect changes in the environment. With regards to the Reward Measure adopted, the two approaches achieved similar results. This could be interpreted by similar importance of the requirements

Table 4.8: Experimental results on the instances of Group A for the three algorithms *MAENS*-IIa*, *MAENS*-IIb* and *MAENS*-IIc*. The first column shows the instance name (*inst*). For each version of the algorithm tested the table includes the average fitness of the best solution (*avg*), the standard deviation (*std*), the best solution (*best*)

	MAENS*-IIa			MAENS*-IIb			MAENS*-IIc			best
inst	avg	std	best	avg	std	best	avg	std	best	avg
C01	4159.50	17.53	4150	4156.50	12.66	4150	4160.00	17.75	4150	4151.67
C05	5367.17	3.34	5365	5366.83	2.41	5365	5369.00	5.83	5365	5366.00
C06	2540.00	5.00	2535	2539.67	4.99	2535	2541.00	4.90	2535	2537.50
C09	5268.00	16.00	5260	5261.67	7.23	5260	5264.00	10.12	5260	5262.83
C10	4707.33	9.64	4700	4704.17	8.37	4700	4705.17	9.53	4700	4703.33
C11	4641.33	3.14	4630	4641.50	2.29	4640	4642.33	2.49	4640	4641.33
C15	4944.67	7.74	4940	4945.50	4.72	4940	4946.00	7.57	4940	4946.33
C18	5641.17	9.04	5620	5637.50	8.73	5620	5637.76	9.27	5620	5635.00
D01	3232.83	3.10	3225	3231.17	6.67	3215	3232.17	5.11	3215	3225.00
D07	3115.00	0.00	3115	3115.00	0.00	3115	3115.00	0.00	3115	3115.00
D08	3045.33	1.82	3045	3047.67	4.42	3045	3047.67	4.42	3045	3045.67
D11	3760.33	2.25	3755	3760.50	1.50	3760	3761.72	3.48	3755	3760.17
D21	3058.33	3.24	3050	3058.33	3.25	3050	3059.00	4.16	3050	3056.67
D23	3172.67	9.66	3150	3162.50	13.34	3135	3162.67	7.39	3150	3158.17
E01	4911.00	2.41	4910	4911.00	2.00	4910	4911.50	2.93	4910	4910.33
E05	4611.83	25.35	4585	4601.17	18.74	4585	4615.00	26.08	4585	4607.33
E09	5830.67	18.35	5810	5832.17	19.39	5810	5834.17	21.64	5810	5832.67
E11	4674.33	10.70	4670	4672.33	5.12	4670	4678.00	15.03	4660	4671.67
E12	4201.00	7.39	4195	4205.33	12.24	4195	4207.50	14.59	4180	4200.67
E15	4218.00	6.98	4210	4217.83	5.73	4205	4219.33	5.59	4210	4215.00
E19	3243.33	3.78	3235	3242.00	4.58	3235	3242.00	4.58	3235	3239.33
E21	3733.50	2.31	3730	3733.67	2.21	3730	3733.27	2.39	3730	3731.33
E23	3715.83	3.23	3710	3716.83	2.73	3715	3715.50	1.98	3710	3715.17
e1-B	4503.60	9.87	4498	4499.67	6.26	4498	4504.79	10.42	4498	4501.77
e2-B	6324.67	10.28	6317	6321.63	5.84	6317	6323.86	9.41	6317	6321.93
e3-B	7783.77	9.12	7775	7782.87	8.35	7777	7786.55	10.73	7777	7780.90
e3-C	10312.23	15.45	10292	10314.80	20.03	10292	10318.31	19.15	10292	10310.67
e4-A	6463.87	3.30	6454	6463.07	2.02	6461	6463.83	5.07	6446	6463.47
e4-B	9029.27	16.82	9000	9026.63	16.17	9000	9021.10	17.84	8990	9013.10
e4-C	11589.13	24.44	11540	11586.80	27.09	11536	11621.28	72.42	11555	11584.97
s1-B	6402.53	18.33	6388	6401.80	16.88	6388	6397.59	12.70	6388	6393.17
s2-A	9931.93	25.17	9889	9928.37	24.06	9889	9934.80	29.49	9881	9929.37
s2-B	13171.97	24.27	13122	13170.97	31.06	13107	13171.41	29.10	13123	13163.57
s2-C	16478.50	34.87	16425	16492.30	39.99	16442	16505.97	51.89	16434	16456.77
s3-A	10282.67	32.08	10221	10288.47	28.39	10221	10290.67	25.78	10251	10284.60
s3-B	13814.90	58.66	13722	13818.63	73.32	13717	13821.50	47.04	13747	13792.63
s3-C	17287.27	37.04	17205	17288.43	31.12	17223	17309.87	37.46	17221	17287.70
s4-A	12404.20	35.59	12301	12388.17	37.70	12304	12388.59	41.42	12316	12367.40
s4-B	16399.90	50.38	16305	16427.13	51.61	16278	16437.60	54.52	16281	16384.27
s4-C	20847.80	134.55	20603	20912.60	266.38	20565	21037.24	223.95	20648	20796.70
F01	4047.00	2.79	4040	4045.50	3.25	4040	4047.59	3.32	4040	4046.67
F04	3498.67	3.45	3485	3498.17	4.74	3485	3499.00	4.16	3485	3498.83
F07	3348.33	30.45	3335	3345.00	30.00	3335	3338.34	17.95	3335	3338.33
F09	4749.50	12.21	4730	4746.67	10.43	4730	4748.45	8.48	4730	4743.00
F11	3843.67	11.25	3835	3843.17	11.58	3835	3847.07	12.35	3835	3839.83
F12	3404.67	11.93	3395	3408.00	17.45	3395	3416.83	26.94	3395	3410.17
F14	3342.50	9.62	3330	3343.33	16.35	3330	3339.50	11.86	3330	3339.29
F19	2541.67	9.59	2525	2533.17	6.39	2525	2532.50	9.64	2525	2531.17
F24	3235.33	10.33	3215	3232.50	9.46	3215	3233.83	10.38	3210	3230.17
val10D	530.77	0.94	528	530.70	1.39	528	531.13	1.26	529	530.50
val4D	530.73	2.25	530	530.13	0.43	530	530.53	1.65	530	530.00
val5D	582.67	2.67	577	582.97	2.07	579	581.93	2.83	577	581.87
val8C	524.40	2.12	521	524.73	1.57	522	524.73	2.00	521	524.37

Table 4.9: Experimental results on the instances of Group B for MAENS*-IIa and MAENS*-IIb. The first column shows the instance name (*inst*). The second column shows the fitness of the best known (BK) solution for each instance[87]. For each version of the algorithm tested the table includes the average fitness of the best solution (*avg*), the standard deviation (*std*), the best solution (*best*)

		MAENS*-IIa			MAENS*-IIb		
inst	BK	avg	std	best	avg	std	best
EGL-G1-A	970495	978636.00	5267.70	964014	978127.07	5330.95	968157
EGL-G1-B	1085097	1086113.80	4709.52	1075069	1088504.40	5597.17	1076011
EGL-G1-C	1201030	1209512.20	5983.41	1197057	1208264.80	6225.46	1196975
EGL-G1-D	1325317	1331918.77	5702.86	1322682	1331367.00	5963.31	1318679
EGL-G1-E	1461469	1466771.97	6458.51	1451314	1465321.17	4890.69	1455995
EGL-G2-A	1061103	1107519.20	3744.93	1099674	1107461.13	2864.67	1101083
EGL-G2-B	1173286	1220912.47	4687.58	1213516	1220423.67	4751.97	1213237
EGL-G2-C	1295036	1356660.60	4883.30	1346969	1352307.90	5182.32	1338497
EGL-G2-D	1430267	1493163.27	5173.56	1482470	1494645.93	5294.38	1486269
EGL-G2-E	1557159	1635756.30	5750.90	1622468	1636974.10	6235.37	1626530

that the two measures try to satisfy (diversity and survival ability of the offspring). The balance might be different when tackling larger CARP instances, as in the case of those in Group B, where the exploration ability of the operator might have a bigger impact on the performance of the algorithm.

The results achieved by MAENS*-IIa and MAENS*-IIb on Group B are included in table 4.9. The two algorithms show a comparable result on 9 instances out of 10, with the only statistically different result according to the Wilcoxon Rank-Sum test being that of the instance EGL-G2-C, with a p-value of 0.0004. The similarity of the results achieved by the two different versions of the algorithm in both datasets can be explained by the fact that the use of the FLA metrics makes the algorithm more robust with respect to the Reward Measure considered. Further experiments with more aggressive Credit Assignment strategies might reveal more differences between the adoption of the two different Reward Measures.

Finally, a comparison with a version the algorithm selecting one crossover operator randomly during each generation is provided. The results of such algorithm are included in table 4.10, in the rightmost column named *random*. At the bottom it is possible to see the number of statistically different instances according to the Wilcoxon Rank-Sum test with a level of significance of 0.05 ((line #) with respect to the three algorithms MAENS*-IIa, MAENS*-IIb and MAENS*-IIc, along with the number of times the random algorithm has won the comparison (line W). It is worth noting that the random algorithm achieves a fairly good performance, as it achieves statistically comparable results with the proposed techniques for several instances. This result could be explained by several factors, including a positive interaction between the crossover operators that have been considered in this case study, the fact that none of the crossover operators performs extremely poorly, and their limited number. It is reasonable to expect that when a much larger number of options is considered, and when some of these are extremely poor, the performances of a random operator selection would deteriorate considerably.

4.6.3 Effectiveness of the FLA measures

An experiment was designed to understand whether the use of the online FLA techniques has a beneficial effect on both the optimization ability and the prediction capacity of the algorithm. Therefore, MAENS*-IIc was compared to MAENS*-IIrw on the Group A dataset, a version of the algorithm which only makes use of the Proportional Reward measure as an input feature of the learning algorithm, without considering the values provided by the FLA techniques. In this context, the results achieved by the algorithm are the main interest of this work, but it is more interesting to verify that the results are significantly different or not and prove, as a consequence, a certain suitability of the rDWM algorithm to the presence of the FLA measures. The results of such algorithm are included in table 4.10 in the column *MAENS*-IIrw*. A Wilcoxon Ranked-Sum test was performed against the results achieved by MAENS*-IIc. The two algorithms produced

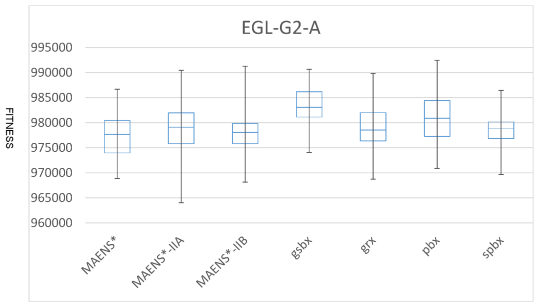
Table 4.10: Experimental results on the instances of Group A relative to *MAENS**, *MAENS*-IIrw*, the *oracle*, and MAENS* with random selection. The first column shows the instance name (*inst*). For each version of the algorithm tested the table includes the average fitness of the best solution (*avg*), the standard deviation (*std*), the best solution (*best*).

	MAENS*			MAENS*IIrw			oracle			random		
inst	avg	std	best	avg	std	best	avg	std	best	avg	std	best
C01	4161.67	19.38	4150	4158.67	13.16	4150	4155.33	13.03	4150	4164.67	18.48	4150
C05	5366	10.95	5365	5378.33	18.36	5365	5365	0.00	5365	5366.17	2.11	5365
C06	2542	25.1	2535	2545.17	3.98	2535	2536.67	3.73	2535	2541.33	4.82	2535
C09	5270	91.65	5260	5280.33	20.41	5260	5269	15.08	5260	5272.83	17.01	5260
C10	4702.17	6.54	4700	4707.33	11.53	4700	4700.67	3.59	4700	4702.17	6.54	4700
C11	4641.33	3.14	4630	4657	27.37	4640	4640.17	2.41	4630	4642.33	2.49	4640
C15	4946.17	7.38	4940	4964.17	15.76	4940	4947	9.27	4940	4946.17	3.80	4940
C18	5638.67	7.74	5620	5642.17	6.91	5625	5636.17	7.82	5625	5638.33	10.03	5620
D01	3232.83	4.02	3215	3224.83	8.99	3215	3229.5	6.87	3215	3231.17	5.87	3215
D07	3115	0.00	3115	3116.33	3.4	3115	3115	0.00	3115	3115	0.00	3115
D08	3045.67	2.49	3045	3052	4.58	3045	3045.67	2.49	3045	3046.33	3.40	3045
D11	3761.5	3.91	3760	3762.67	6.42	3745	3759.67	4.99	3745	3760.17	2.41	3750
D21	3059.83	5.24	3050	3063.67	11.47	3055	3055.17	3.98	3050	3058.17	3.02	3050
D23	3164.83	12.28	3135	3167.83	12.23	3140	3153.17	8.51	3135	3165.83	14.55	3140
E01	4911.17	2.11	4910	4916	6.11	4910	4910.5	1.5	4910	4910.83	2.27	4910
E05	4606.67	22.34	4585	4621.5	21.57	4585	4612	24.17	4585	4605.83	22.10	4585
E09	5837	21.16	5815	5851.33	25.26	5815	5835.83	21.26	5815	5840.17	22.49	5810
E11	4677	13.52	4655	4698	25.68	4670	4673.83	7.71	4665	4678.83	12.23	4670
E12	4202.33	13.15	4180	4226	17.63	4195	4204.5	11.5	4190	4203.67	11.32	4190
E15	4217.5	6.68	4205	4223.67	5.91	4210	4214.5	6.24	4205	4217.67	6.02	4210
E19	3242.67	4.23	3235	3244.67	1.8	3235	3238.33	4.71	3235	3242	4.58	3235
E21	3733	2.45	3730	3732.67	2.49	3730	3730.67	1.7	3730	3733.17	2.41	3730
E23	3715.5	1.5	3715	3720.5	7.34	3715	3714	2	3710	3716	2.71	3710
egl-e1-B	4501.2	8.33	4498	4509.17	11.68	4498	4502.6	8.5	4498	4500.80	7.44	4498
egl-e2-B	6323.67	9.58	6317	6329.83	13.35	6317	6320.37	6.36	6317	6324.17	8.96	6317
egl-e3-B	7780.43	5.91	7775	7790.47	11.23	7777	7783.93	11.61	7775	7778.43	3.22	7775
egl-e3-C	10317.6	18.45	10292	10323.6	20.38	10292	10316.63	18.86	10292	10313.07	15.52	10292
egl-e4-A	6462.5	3.04	6450	6464.07	5.39	6446	6462.77	2.58	6456	6462.13	5.20	6446
egl-e4-B	9022.5	16.39	8988	9023.47	16.23	8992	9011.2	11.79	8993	9032.60	15.95	8999
egl-e4-C	11592.53	32.82	11538	11602.8	31.64	11550	11610.13	41.31	11554	11593.50	21.02	11555
egl-s1-B	6399.9	16.38	6388	6407.3	19.35	6388	6399.7	14.5	6388	6399.87	15.42	6388
egl-s2-A	9931.63	26.62	9889	9943.43	32.78	9889	9928.37	27.01	9885	9933.97	29.35	9889
egl-s2-B	13179.07	26.11	13124	13217.13	44.41	13159	13179.2	29.61	13124	13181.57	32.72	13125
egl-s2-C	16510.1	43.05	16430	16516.03	46.02	16430	16498	41.64	16433	16512.27	39.21	16442
egl-s3-A	10282.63	29.41	10221	10293.87	29.07	10242	10276.5	26.39	10221	10288.63	28.17	10243
egl-s3-B	13820.13	57.75	13736	13874.37	59.29	13736	13823.37	60.51	13750	13818.93	62.17	13714
egl-s3-C	17289.73	42.75	17220	17325.9	46.56	17237	17296.1	33.42	17249	17286.87	32.29	17215
egl-s4-A	12400.87	47.91	12283	12403.37	47.36	12316	12382.93	41.71	12304	12407.07	31.77	12305
egl-s4-B	16421.17	50.46	16325	16454.3	42.73	16351	16414.67	47.18	16344	16435.90	33.33	16334
egl-s4-C	21047.97	174.66	20758	21065.8	166.32	20702	21117.7	327.1	20745	20955.50	181.04	20611
F01	4046.83	2.73	4040	4046.43	2.54	4040	4044.5	3.73	4040	4046.17	3.34	4040
F04	3498.67	3.64	3485	3499.67	5.31	3485	3496.17	3.8	3485	3499	4.16	3485
F07	3335	0.00	3335	3345	30	3335	3345	30	3335	3341.67	24.94	3335
F09	4746	11.79	4730	4750.53	12.34	4730	4742	8.12	4730	4746.67	10.83	4730
F11	3850	11.11	3835	3846.97	13.56	3835	3841	6.88	3835	3851.50	12.12	3835
F12	3410	23.42	3395	3425.7	32.32	3395	3402.33	13.09	3395	3412.17	16.82	3395
F14	3342.33	12.23	3330	3340.83	13.17	3330	3338	13.52	3330	3344.33	12.16	3330
F19	2535.17	9.35	2525	2537.67	8.73	2525	2526.67	3.73	2525	2544.67	12.78	2525
F24	3234.33	8.63	3215	3232	9.36	3215	3225.5	11.28	3210	3239.33	9.64	3220
val10D	530.6	1.23	528	532.13	1.65	530	529.9	1.08	528	531.03	1.35	528
val4D	530.13	0.72	530	530.77	2.04	530	530.23	0.76	530	530.37	1.17	530
val5D	583.13	2.06	579	583.97	3.42	577	581.93	2.69	577	583.57	2.29	579
val8C	524.63	1.76	521	524.23	2.49	521	523.37	1.76	521	524.93	1.77	521

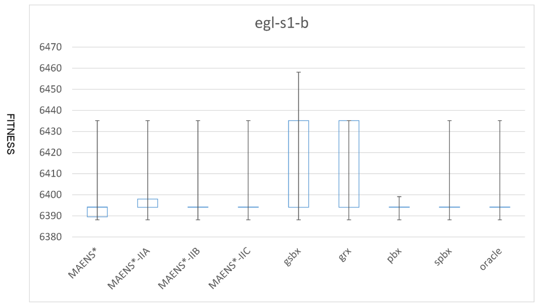
Table 4.11: Number of comparisons against every operator with statistically different results. The bottom row shows the number of comparisons where the algorithm achieves a better average fitness

		MAENS*-IIa	MAENS*II-b	MAENS*IIc
MAENS*	#	4	2	4
	W	0	0	2
MAENS*-IIrw	#	-	-	36
	W	-	-	3
Oracle	#	19	-	-
	W	16	-	-
Random	#	8	8	4
	W	0	0	0

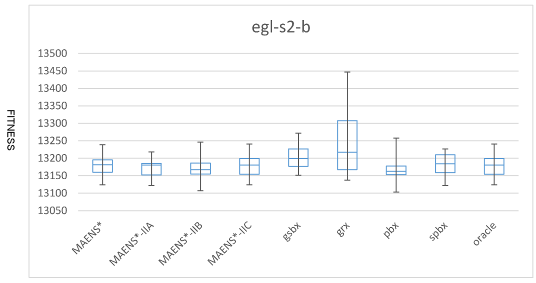
Figure 4.2: Box plots relating to the four instances *EGL-G2-A* (a), *egl-s1-b* (b), *egl-s2-b* (c), *egl-e4-c* (d). The boxes refer to the first quartile, median and third quartile. Whiskers show minimum and maximum values over a sample size of 30 best fitness values relative to the independent runs of each algorithm reported in tables 4.4, 4.5, 4.10, 4.8, 4.9 and 4.13 .



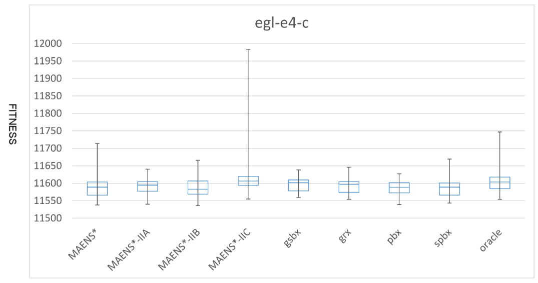
(a) Fitness distribution for *EGL-G2-A* instance



(b) Fitness distribution for *egl-s1-b* instance



(c) Fitness distribution for *egl-s2-b* instance



(d) Fitness distribution for *egl-e4-c* instance statistically different results on 36 instances out of 53. MAENS*-IIrw achieved better results only on 3 instances, losing the comparison on 33. A Wilcoxon signed-rank test was consequently applied across the problem instances, which confirmed that the two algorithms produce significantly different results (respectively $W_{stat} = 26$ with $p < 0.05$ and $W_{stat} = 54.5$ sample size: 42). This can be interpreted as a signal that the rDWM is concretely affected by the FLA measures, which influence (in a beneficial way) the decisions made by the algorithm.

Table 4.12: In the rows at the bottom, the number of comparisons against every operator with statistically different results. The following row shows the number of comparisons where the algorithm achieves a better average fitness.

	MAENS*-IIb	MAENS*-IIc	best
#	3	7	9
W	1	5	0
	MAENS*-IIa	MAENS*-IIc	best
#	3	5	8
W	2	5	0
	MAENS*-IIa	MAENS*-IIb	best
#	7	5	6
W	2	0	0

4.6.4 Comparison with the state-of-the-art

The second research question in the introduction of this chapter focuses on the performance of the proposed approach with respect to the existing ones. Therefore, the MAENS*-II variants that make use of the Proportional Reward (a and b) were tested against the oracle. All the three variants were also compared against the results achieved by their base algorithm MAENS*. The results achieved by the oracle and by the MAENS* algorithm for Group A are included in table 4.10, in columns *MAENS** and *oracle*. In the bottom rows, the results of the set of statistical tests as described in A.1.

The results of the test show how the number of statistically different results is small

(4 for MAENS*-IIa and MAENS*-IIc and 2 for MAENS*-IIb). In these few instances, MAENS*-IIa and MAENS*-b perform better than MAENS*, while MAENS*-II wins the comparison in half of the instances (2 out of 4). The online learning system is therefore able to achieve results comparable to those achieved by the bandit solver.

The comparison with the oracle shows that MAENS*-IIa and MAENS*-IIb are able to achieve comparable results in most cases. In most of the instances with statistically different results, the oracle was able to perform better. It is worth noting that in a small number of instances the algorithm using the FLA measures was able to produce better results than the oracle. This is some evidence that, if the oracle represents a "lower bound" for the results that is possible to achieve using the Proportional Reward, the use of more than one measures (as in this case) can help the algorithm to achieve results beyond these bounds.

Finally, the results achieved by MAENS*-IIa and MAENS*-IIb, included in table 4.9, are compared against four state-of-the-art algorithms, whose results are included in table 4.13. The results of MAENS*, of MAENS-RDG [90] and VND [91] and an algorithm combining Iterate Local Search and Variable Neighbourhood Descent [87] are included. It is possible to notice how MAENS*-IIa and MAENS*-IIb, as well as MAENS*, outperform all the other algorithms in terms of solution quality for the first 5 instances of group B 4.9. MAENS*-IIa, MAENS*-IIb and MAENS* produce a new best known solution for all of these instances, with MAENS*-IIa achieving the best ones on the first two instances (G1-A and G1-B), MAENS*-IIb on instances G1-C and G1-D and MAENS* finding the best one on the instance G1-E. In all these instances MAENS*-IIa and MAENS*-IIb achieve also the best average fitness in four cases. For the following 5 instances, the best results are achieved by either MAENS-RDG or VND. In all these cases, MAENS* is outperformed by both MAENS*-IIa and MAENS*-IIb. These results can be explained by the fact that their base algorithm, MAENS*, is already performing well for these in-

Table 4.13: Comparison with some state-of-the art approaches for CARP. The first column shows the instance name (*inst*). For each algorithm the table includes the average fitness of the best solution (*avg*) and the best solution (*best*). The results are compared to those achieved by MAENS*-IIa and MAENS*-IIb included in table 4.9. The algorithms considered are MAENS*[21], MAENS-RDG[90], VND [91] and ILS-RVND[87]. No statistical test was carried out due to the partial availability of the results of the compared algorithms.

	MAENS*		MAENS-RDG		VNS		ILS-RVND	
inst	avg	best	avg	best	avg	best	avg	best
EGL-G1-A	977754.4	968897	1007223.0	1000575	10007393.0	997055	1014930.9	1004864
EGL-G1-B	1088706.5	1079793	1124751.0	1111971	1122077.0	1114120	1143221.7	1129937
EGL-G1-C	1209058.8	1195902	1251718.0	1243779	1253789.0	1243808	1270100.3	1262888
EGL-G1-D	1331595.8	1323397	1383619.0	1371443	1383997.0	1373480	1409811.9	1398958
EGL-G1-E	1469455.4	1449542	1524393.0	1512584	1525994.0	1517772	1556138.5	1543804
EGL-G2-A	1107363.0	1101559	1108916.0	1096027	1105870.0	1098454	1126561.0	1115339
EGL-G2-B	1223132.3	1213769	1222183.0	1213617	1220012.0	1211759	1237741.8	1226645
EGL-G2-C	1354725.3	1345587	1353118.0	1344148	1351845.0	1344184	1376931.6	1371004
EGL-G2-D	1495089.7	1486646	1489723.0	1481181	1489500.0	1481045	1520794.3	1509990
EGL-G2-E	1636140.6	1630656	1630132.0	1618955	1630048.0	1616119	1664230.2	1659217

stances. However, both variants MAENS*-IIa and MAENS*-IIb managed to outperform MAENS* in most of the instances. It is important to note that the runtime (not considered in this work) of these algorithms is not comparable to those of the decomposition based approaches, which manage to find these results in a fraction of the time required by MAENS*-IIa and MAENS*-IIb.

The behaviour of the algorithms can be analyzed also in terms of the fitness distribution of its solutions. Figure 4.2 shows the box plot relative to three representative instances belonging to Group A (*egl-e4-C*,*egl-s1-B*,*egl-s2-B*) and one instance of Group B (*EGL-G2-A*). In the case of the *EGL-G1-A* instance, it is possible to notice how SPBX and GRX are the crossover operators whose usage leads to the distributions with the lowest median. The distribution of the three AOS considered in this case (MAENS*, MAENS*-IIa and MAENS*-IIb) are centred around the same median value, although MAENS*-IIa is capable of producing solutions of considerably better quality (bottom whisker) which translate into new minima for this instance.

For the *egl-s1-B* instance (figure 4.2.b), the behaviour of the algorithms is quite similar, as in most of the cases the distributions lie around the same median. When considering the results of the versions of the algorithm using each crossover operator, GSBX, GRX and SPBX show a much wider distribution of their results although in the first two cases a large number of solutions are equal to the median value, while PBX results are much less spread. The different AOS strategies achieve overall comparable results. This instance represents an example of non optimal behaviour as none of the AOS strategies considered has managed to match that of the best crossover operator (GRX).

For *egl-s2-b* (figure 4.2.c), PBX is the operator that achieves the best results, while GRX performs the worst. MAENS*-IIb manages to achieve the same solution quality and similar median to PBX. This is also confirmed by the larger selection rate given to the PBX operator (figure 4.9).

In the case of *egl-e4-c* (figure 4.2.d), PBX and SPBX distributions have a similar median and similar quartiles performing the best among the four crossover operators. Among the AOS strategies, MAENS*-IIb solutions are distributed around a similar median but more spread.

4.6.5 Prediction Ability

In order to understand the behaviours of the algorithms, and to gain a deeper understanding of the selection mechanisms, a comparison of the selection rates of the four different crossover operators is included in figures 4.3 to 4.15. The plots refer to the selection rates relative to the instances *egl-s1-B*, *egl-s2-B* and *EGL-G2-A*.

The y-axis in the figure refers to the Selection Rate (SR) of each crossover operators, where a SR of 0 means that the operator is not selected and a SR equal to 1 means that only that operator is selected. The x-axis corresponds to the average fitness of the

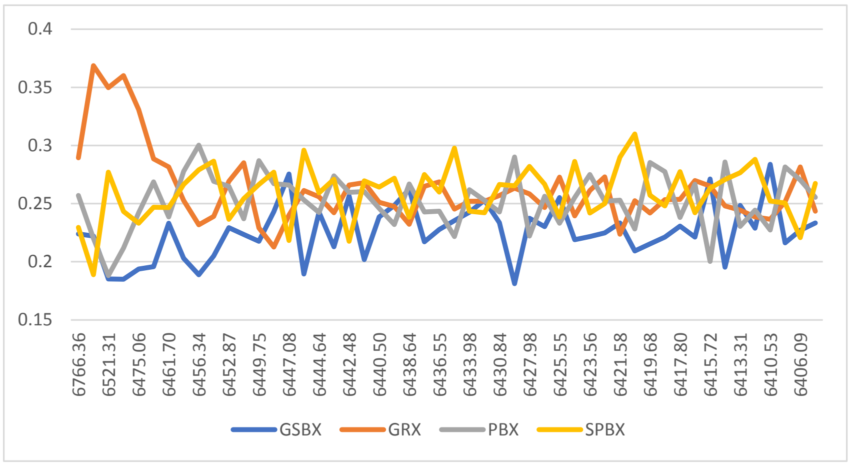


Figure 4.3: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIa on s1-B

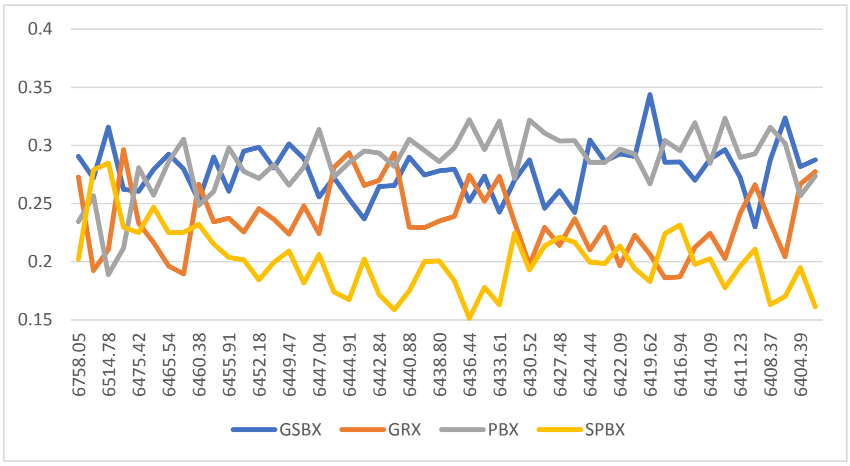


Figure 4.4: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIb on s1-B

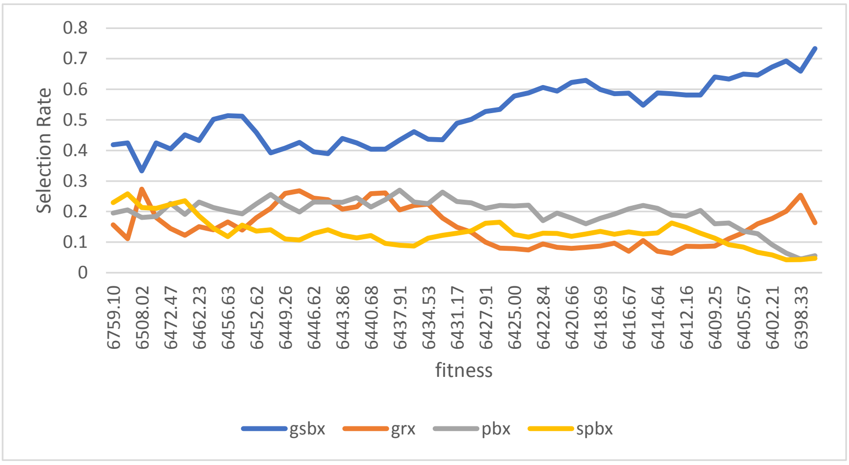


Figure 4.5: Selection Rates for of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) MAENS*-IIc on s1-B

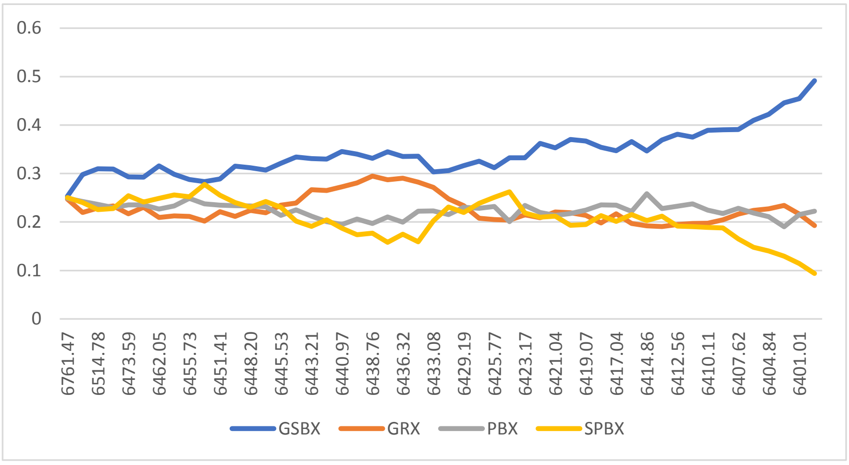


Figure 4.6: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS* on s1-B

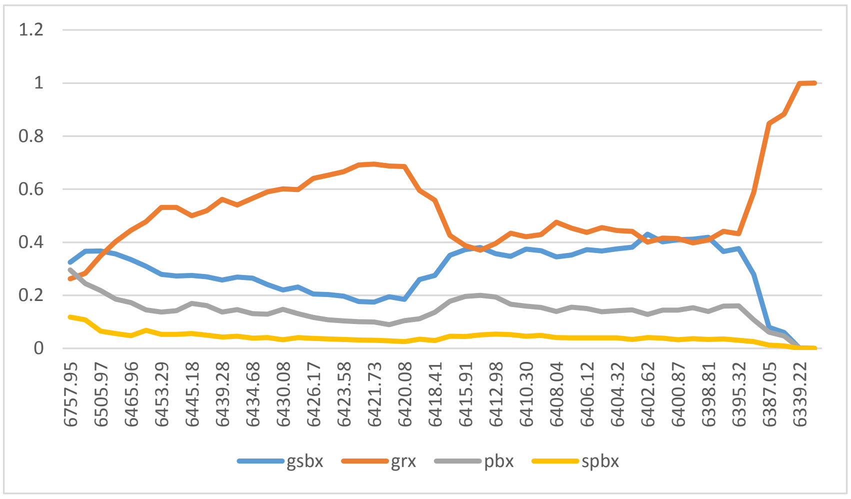


Figure 4.7: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for Oracle on s1-B

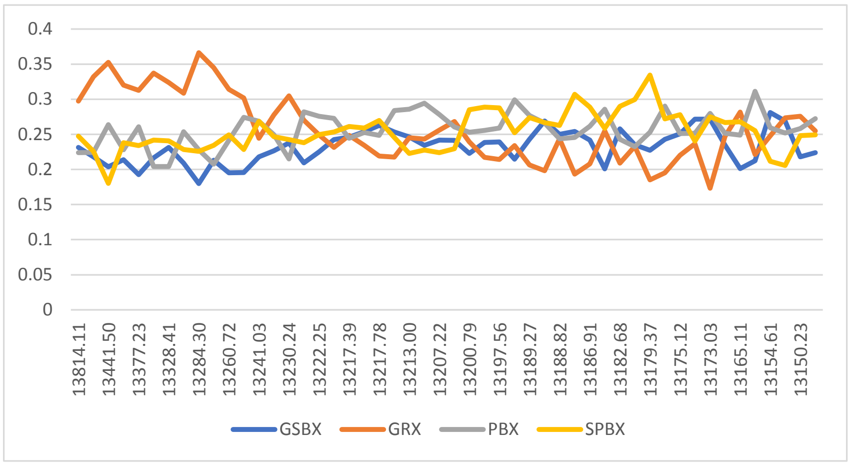


Figure 4.8: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIa on s2-B

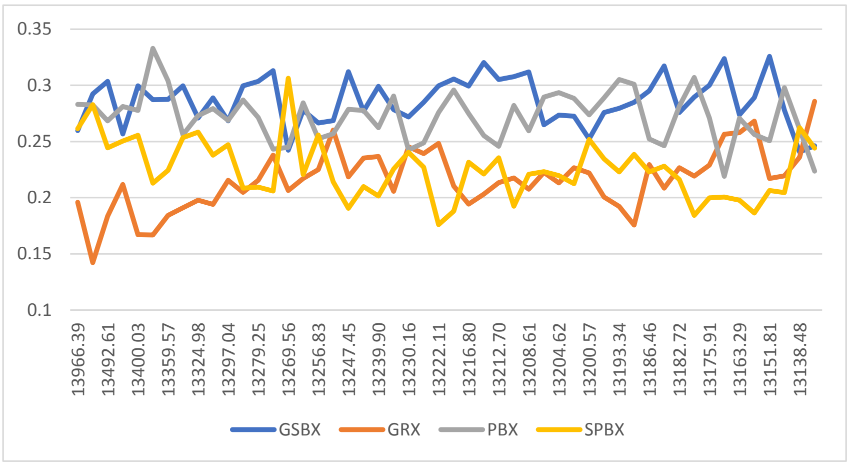


Figure 4.9: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIb on s2-B

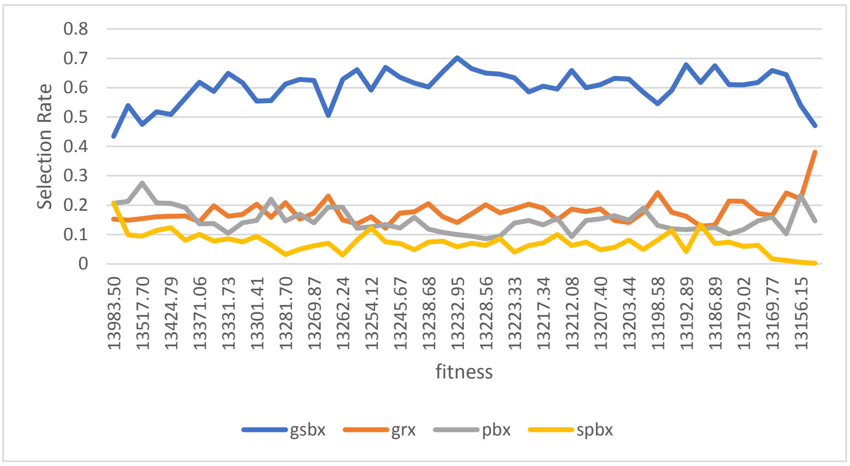


Figure 4.10: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIc on s2-B

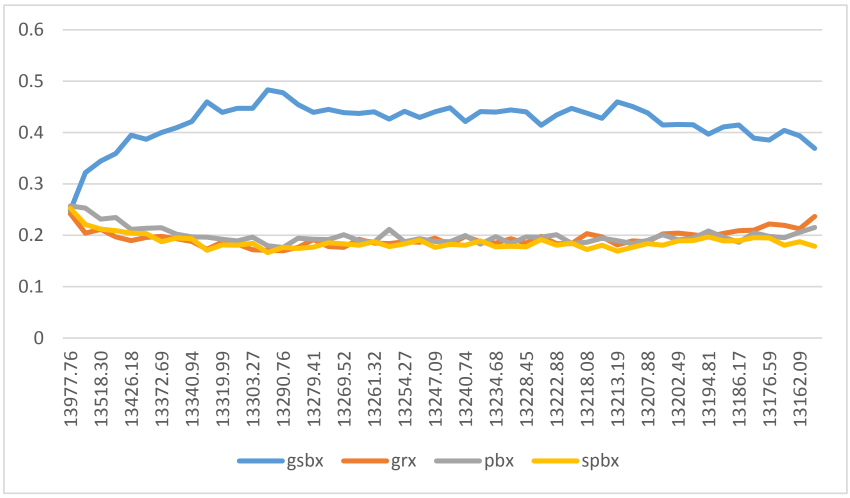


Figure 4.11: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS* on s2-B

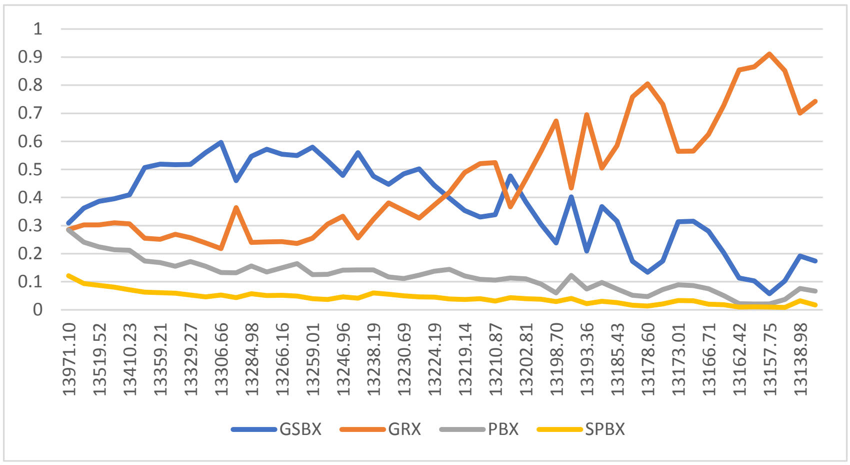


Figure 4.12: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for Oracle on s2-B

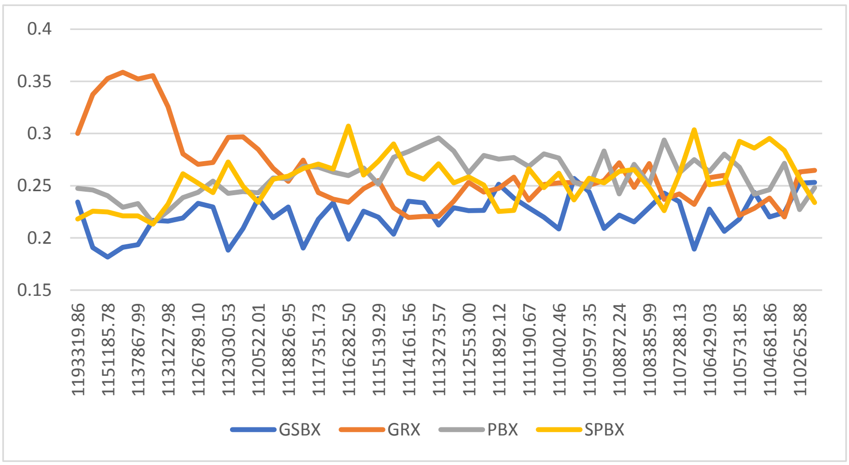


Figure 4.13: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIa on G2-A

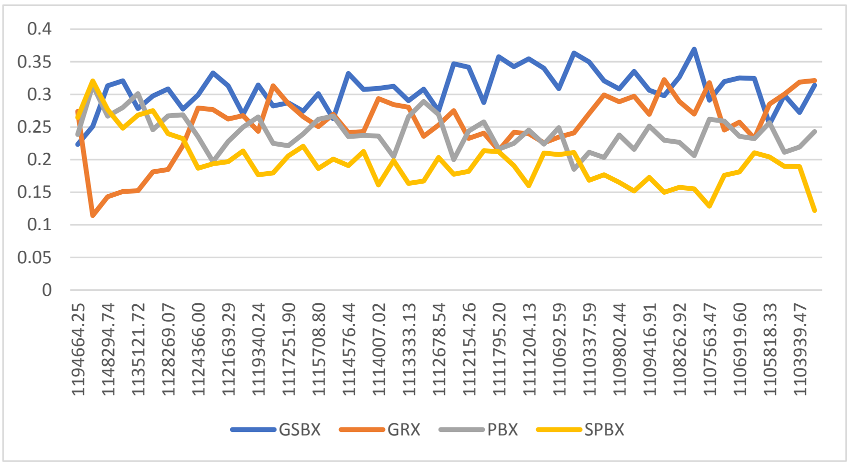


Figure 4.14: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS*-IIb on G2-A

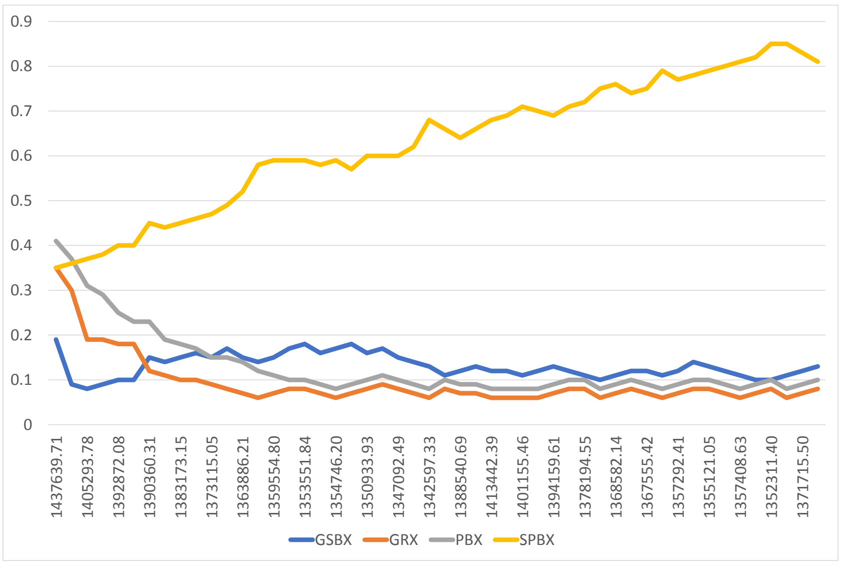


Figure 4.15: Selection Rates of the four different crossover operators (*GSBX*, *GRX*, *PBX*, *SPBX*) for MAENS* on G2-A

population discretised into 50 intervals. The focus of this study, is therefore, how the SR of the four operator changes while the search is carried out and the average fitness of the population decreases.

In the first instance, *egl-s1-B* (figures 4.3,4.4,4.5,4.6 and 4.7), it is possible to notice three phases in the oracle prediction (figure 4.7). A first phase where the GRX operator is preferred over the others, an intermediate phase where the GRX and GSBX operators have nearly equal selection rates and a last phase characterized by a rise of the selection rate of the GRX operator which reaches 1 in the last moments of the search.

Both MAENS* (figure 4.6) and MAENS*-IIc (figure 4.5) award the GSBX operator with the highest selection rate for the whole search, missing the prediction of the change in the environment made by the soracle. It is possible to see, however, how MAENS*-IIc increases the selection rate of GSBX more rapidly than MAENS*.

The SR of both MAENS*-IIa (figure 4.3) and MAENS*-IIb (figure 4.4) show different changes during the search, proving that the CS is more successful in predicting such events. In particular, MAENS*-IIb acknowledges the operators GSBX and PBX as the most useful ones during the search. It is worth remembering that MAENS*-IIa, makes use of a different Reward Measure and, therefore, is not comparable to the prediction made by the oracle. In this case, MAENS*-IIa, after an initial epoch of dominance of the operator *GRX*, shows an alternation of moments where the three operators *GRX*, *PBX* and *SPBX* show the highest selection rates.

On the second instance (figures 4.8,4.9,4.10,4.11,4.12), the oracle identifies a change in the environment halfway through the search (figure 4.12). The concept drift is not detected by either MAENS* (figure 4.11) or MAENS*-IIc (figure 4.10), which, however shows an higher exploitation of the GSBX operator. MAENS*-IIb (figure 4.9) identifies the operators *GSBX* and *PBX* as the most successful ones; even in this case the change detected by the oracle is not present. As for the previous instance, MAENS*-IIa (figure 4.8) shows different moments where the three operators *GRX*, *PBX* and *SPBX* achieve the highest SR. The lowest SR for *GSBX* seems to indicate that this operator, for this version of the algorithm, is probably the one that introduces the least diversity in the population.

For the large CARP instance EGL-G2-A, the behaviour of MAENS* (figure 4.15) shows a predominance of operator *GSBX* over the other ones. MAENS*-IIb (figure 4.14) shows a similar behaviour to that of MAENS*, identifying the *GSBX* operator as the one with the best performance during almost the whole search. Finally, MAENS*-IIa shows again an initial period of higher performance for the *GRX* operator, followed by an alternation of the *PBX* and *SPBX* operators (figure 4.13). The occurrence of this initial period of higher performance for the *GRX* operator seems to suggest that this operator is introducing the highest diversity in the initial part of the search, when the solutions are not extremely good.

The results of these experiments show that failing to detect a change in the environment does not necessarily translate into a worst performance of the algorithm and vice versa. This is confirmed by the fact that the algorithms produce good results despite the different selection rates. The relationship between the prediction ability of the algorithms and their results is therefore quite complex. There are several factors that influence its behaviour and that should be considered in order to fully grasp this mechanism, such as the interaction between the different operators, the performance of the single operators and the variation of the selection rates.

4.7 Summary and Discussion

The contributions of this chapter are:

- A study of the existing Fitness Landscape Analysis techniques, in order to identify those that are more suitable to be adopted within the MAENS* algorithm, and that can more usefully provide a representation of a population;
- The definition of a novel Credit Assignment technique, combining the use of an online learning algorithm that is used to predict the performance of each crossover operator;
- An investigation of the combined use of such Credit Assignment techniques with alternative Reward Measures, respectively the Proportional Reward and the Diversity Based Reward.

The work discussed in this chapter provides evidence to support the use of online learning strategies within EAs. Some of the results and the aspects of this work can be found of particular interest. In particular:

- It represents one of the first works involving FLA techniques in the AOS scenario. In fact, at the time this work was carried out, the literature on Adaptive Operator

Selection was lacking works focusing on alternative ways to measure the reward of operators that would not involve the use of the fitness of the offspring or their diversity;

- It is one of the few works on Adaptive Operator Selection applied on a real world problem. While the studies on AOS are conducted using toy problems and simplified versions of EAs, the current work deals with a real world problem (CARP) and is studied in what, at the time of the preparation of the present work, could be considered as a state-of-the-art meta-heuristic for this combinatorial problem;
- As for the previous chapter, this work provides evidence of the effectiveness of the use of AOS strategies in the context of crossover operators.

CHAPTER 5

Knowledge Incorporation Through Task Affinity Prediction

We shall not cease from exploration,
and the end of all our exploring will
be to arrive where we started and
know the place for the first time.

T. S. Eliot, *Little Gidding*

5.1 Introduction

The work of this chapter proposes a novel Knowledge Incorporation technique based on the use of a ensemble learning based classifier that is capable of predicting some domain information, named *task affinity*, for the Capacitated Arc Routing Problem, using a set of features of the CARP instances, in order to enhance the performance of a meta-heuristic for such problem. A novel version of the MAENS* algorithm, which represents the case study of this work, named MAENS*-III illustrates the use of this technique in the context of CARP.

The contributions of this work include:

1. The definition of the *task affinity*, a CARP related information that, if predicted correctly, can be effectively used to boost the performances of an EA, and that can be easily converted to similar problems in the routing family and to other combinatorial problems;

2. The definition of a set of 34 CARP related features, extracted from the problem instances, that are used to train the learning system in order to predict the *task affinity*;

3. The creation of a Affinity Based Local Search that is used to improve the average affinity of solutions and that is used to inject the information predicted by the learning system.

Experimental analysis was conducted using MAENS*-III on a known benchmark dataset and provides empirical evidence of the effectiveness of such an approach in several instances of the dataset. The chapter is structured in the following way. In Section 5.2 the fundamental concept of *Task Affinity* is introduced, upon which the work is based. Section 5.3 introduces the techniques developed to generate a dataset necessary to train the learning system that predicts the affinity for new instances. Section 5.4 shows how the predicted affinity is used within the algorithm to improve its performances. In section 5.5 an experimental analysis using a known benchmark dataset is presented. Finally, section 5.6 includes includes some conclusive remarks and discussion.

5.2 Task Affinity within CARP Instances

This section introduces the concept of *task affinity* between pairs of tasks within a CARP instance. Along with the definitions, the strategies adopted to calculate and update this information throughout the search are discussed.

5.2.1 Task Affinity

The affinity aff(a, b) between two tasks a and b belonging to a CARP instance I can be defined as the likelihood to find a and b within the same route in the set of the optimal solutions for I. It is important to notice that for the calculation of the affinity, the relationship of consecutiveness between the two tasks is not taken into account, although a similar analysis might be performed to predict the likelihood that a follows b in an optimal solution. As computing the affinity would require the knowledge of the whole fitness landscape, it is more practical to resort to approximate this value by estimating it from the information that is in possess of the algorithm.

Given a population pop^t, at the *t-th* generation of a search performed by a meta-heuristic, it is possible to produce an estimate of aff(a, b) by computing a statistic based on the solutions contained in pop^t [1].

The calculation of aff(a, b) depends on two factors:

1. Frequency: how many times the pair (a, b) is contained in the same route within the solutions in *pop*;
2. Quality: the quality of the solutions having this pair in the same route and the quality of solutions where they are not.

The fitness is measured in order to evaluate the quality of a solution, although extended studies might be performed to investigate the importance of other factors such as the diversity between each solution and the rest of the population, and whether the whole population should be taken into account, as it is done in this study, instead of a subset

[1]In the rest of the chapter, the apex notation relative to the generation is dropped to improve the readability, as it should be clear that the affinity that is being referred to is the one estimated during the search and not the authentic one, except for the case where the use of such notation is necessary to distinguish the values of the affinity in different moments of the search.

of the best ones. Given a solution x belonging to the population *pop* the score function can be defined as:

$$\text{score(x)} : \frac{1}{f(x)} \tag{1}$$

where $f(x)$ is the value of the fitness function for the individual x. Since CARP is a minimization problem, an higher score is assigned to the solutions whose fitness is smaller. The value of the score function is then normalized within the range [0...1] using the formula

$$S(x) : \frac{\text{score(x)}}{\sum_{(y \in \text{pop})} \text{score(y)}} \tag{2}$$

aff(a, b) is finally computed using the formula:

$$\text{aff}(a, b) = \sum_x S(x) : R^x(a) == R^x(b) \tag{3}$$

which computes a score depending on both the quality of the solutions where the pair of tasks (a, b) is present in the same route as well as the number of solutions that agree on this. It is easy to notice how aff(a, b) is necessarily going to be a value within the range [0...1], where 0 corresponds to the case where none of the solutions of the population have the two tasks a and b in the same route and 1 is equivalent to the opposite case (all solutions agree on having such pair in the same route).

5.2.2 Computing the Affinity Matrix

The *Affinity Matrix* of a population *pop* is a matrix A of size $N \times N$ where $A(a, b)$ is equal to aff(a, b) and N is the number of required edges (or tasks) of the CARP instance I. It is worth noting that *pop* $\simeq A$, since A contains the same information contained in the individuals of *pop* although coded using a different representation. The matrix A can be obtained by computing the value of aff(a, b) for each pair of required tasks (a, b) in the instance I. In this context, the following two assumptions are made:

1. The most useful affinity matrix (the one approximating with higher accuracy the authentic affinity matrix of the landscape) is most likely the one contained in a population of extremely evolved solutions (whether optimal or suboptimal ones);
2. The information obtained in earlier stages of the search might still be useful.

In view of these considerations, it seems reasonable to adopt a strategy that consists of evolving an affinity matrix by updating its values throughout the whole search. Such matrix will reflect the information contained in the last population of the search as well as some information acquired during the previous generations.

Update Rule

A new affinity matrix tA^{t+1} can be computed at each time step $t + 1$. It is reasonable not to replace the old affinity matrix A^t with the new one, but to compute a combination of the two through a reinforcement learning approach. Therefore, two weights α^t and α^{t+1} are used, such that $\alpha^t + \alpha^{t+1} = 1$, which determine how much the affinity matrix is supposed to rely on the previous information and how much on the one that has been just computed. The updated affinity matrix is therefore obtained in the following way:

$$A(a, b) = A(a, b)^t \times \alpha^t + A(a, b)^{t+1} \times \alpha^{t+1} \tag{4}$$

In a context where the affinity is just learned and not used to modify the search and taking into account that CARP is a minimization problem (smaller fitness corresponds to better solutions) it seems convenient to compute the values of α^t and α^{t+1} considering the average fitness of the two populations pop^t and pop^{t+1}, such that

$$\alpha^t = \frac{f(pop^{t+1})}{f(pop^t) + f(pop^{t+1})} \tag{5}$$

and

$$\alpha^{t+1} = \frac{f(pop^t)}{f(pop^{t+1}) + f(pop^t)} \tag{6}$$

5.3 Learning from CARP Features

This section introduces the strategy developed to extract the information from the problem instances in order to generate a training instance for each pair of tasks of a CARP network[1]. Moreover, it is shown how this data is analyzed and pre-processed and the steps necessary to generate a dataset that is used to learn the relationship between every pair of tasks and their *affinity*.

5.3.1 CARP Features

In order to understand whether the affinity between a pair of tasks a and b can be predicted, the learning system considers a set of 34 different input features. Table 5.1 shows the list of such features and a description. The 34 features are opportunely normalized in order to favour data integration between different datasets. The features can be distinguished between static features related to the Network (the first 14 rows in table 5.1) which are descriptive measures obtained by analyzing the problem instance and that therefore assume the same value for each pair of tasks belonging to the same problem instance. This information is most likely to be useful for scaling purposes when values belonging to different problems instances will be compared. A detailed description of such features is not provided in this context, as they are of easier comprehension given the problem definition.

The remaining 20 features, called "Pair" related instances, assume a specific value for each pair of tasks considered. A first group of features has been extracted from the

[1]In the rest of the chapter, the term *problem instance*, or *CARP instance*, will be used to refer to the instance of a combinatorial optimization problem, while the term *training instance* will refer to the training data used in a Machine Learning context

information that is immediately available from the problem instance description (features 17-20 and 28-29). Several measures are directly derived from the computation of the distance between the two tasks or between each task and the Depot (measures 15-16, 21-22, 25-27 and 33). A final group of instances are inferred by analyzing the pair of tasks in relation with the rest of the graph. In particular, measures 23-24 and 30, compute the length of a path intended as the number of edges that are crossed. Measure 31 is computed in the following way. Firstly, the average number of tasks per route (measure 14) is computed, called *taskroute*. The s-th closest task to a and b such that s is equal to $taskroute + 1$ can be then derived. In other words, s is the first task that would be excluded if each task needs to be clustered with its closest *taskroute* tasks (without considering the demand constraint). The threshold values τ_i and τ_j are respectively the distances from a and b from their respective s_i and s_j. The measure is therefore computed averaging the ratios between the distance of the two tasks and their threshold τ. Measure 32, called *ncloser*, computes the number of tasks that are closer to a with respect to b and vice versa, and averages these two quantities. Finally, measure 33 computes the ratio between the measure *ncloser* and the average number of tasks that are clustered in a route *taskroute*.

In a classical batch learning context, it would be useful to reduce the number of instances to the ones that best explain the affinity of tasks, in order to decrease the computational cost and obtain simpler models with increased generality. However, it seems reasonable to compute and maintain such a large set of features due to the heterogeneous nature of the data, which is presented to the algorithm in an online fashion, and that is extracted from different sources (problem instances). This prevents the possibility and decreases the necessity to perform an offline analysis of the data in order to determine which is the best subset of features. It is possible however to rely on the machine learning algorithm to perform some sort of feature selection in order to increase the generality of the whole learning system.

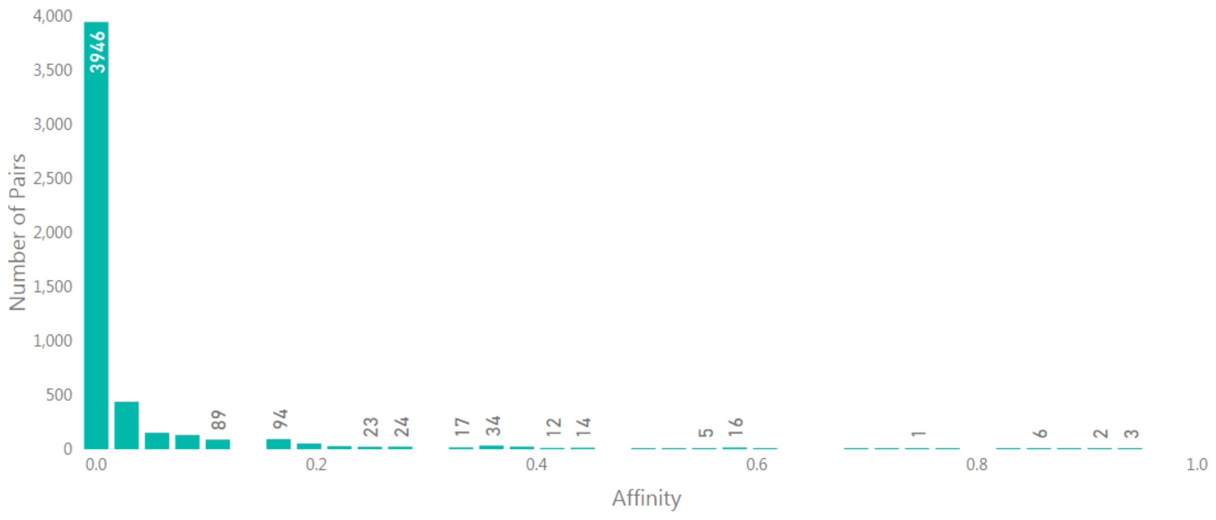
Table 5.1: Input Features. Each line provides an identification number, the name and the description of the feature.

Nr	Title	Description
1	Cap	Vehicle Capacity
2	NVeh	Number of vehicles
3	avgSC	Average Service Cost
4	avgD	Average parwise distance between two tasks (in Service Cost)
5	avgConsec	Average number of consecutive tasks
6	avgDtoDep	Average distance between any task and the depot
7	avgDem	Average Demand
8	stdSC	Standard deviation of Service Cost
9	stdDem	Standard deviation of Demand
10	ntasks	Number of tasks (required edges)
11	notreq	Number of non required edges
12	nvert	Number of vertices
13	density	Graph density
14	taskroute	Average number of tasks per route
15	d(a,b)	Distance between a and b
16	d(a,b)	Distance between b and a
17	dem(a)	Demand of a
18	dem(b)	Demand of b
19	SC(a)	Service cost of a
20	SC(b)	Service cost of b
21	d(a, Dep)	Distance from a to the Depot
22	d(b, Dep)	Distance from b to the Depot
23	n(a, Dep)	Tasks crossed in the path between a and the depot
24	n(b, Dep)	Tasks crossed in the path between b and the depot
25	d(a,b, Dep)	d(a,b)+d(b, Dep)-d(a, Dep)
26	d(b,a, Dep)	d(b,a)+d(a, Dep)-d(b, Dep)
27	sumD	d(a,b)+d(a,b)
28	sumDem	dem(a)+dem(b)
29	sumSC	SC(a)+SC(b)
30	sumN	n(a, Dep)+n(b, Dep)
31	distThreshold	Ratio between sumD and threshold τ[1]
32	ncloser/ntasks	Number of tasks closest to a than b
33	avgDto	$(d(a, b) + d(b, a))/avgD$
34	ncloser/taskroute	Ratio between ncloser and taskroute
35	affinity	Output Feature

5.3.2 Numeral to Class Conversion

It is useful to have a look at how the affinity values are distributed in the range $[0, 1]$ to gain a deeper understanding and consequently design a better learning system. Figure 5.1 shows an histogram relative to the affinity matrix of the *egl-e1-B* CARP instance, obtained during the execution of the MAENS* algorithm. It is possible to notice how the data is extremely skewed, as a large portion of the affinity connections are understandably equal to 0 or very close to it, while very few of those are distributed in the rest of the interval. Designing a regression model that is capable of detecting the outliers (high affinity connections) with great accuracy seems to be an extremely difficult task.

Figure 5.1: Affinity distribution for the egl e1-B instance



The ratio between the size of the three classes is highly dependent from the landscape of the specific instance and from the outcome of a single execution of the algorithm. If the population from which the affinity matrix is extracted is composed by solutions that are very close to each other, because they are in the same basin of attraction, the number of high affinity connections will be higher. As a consequence, the values contained in the AM itself might not reflect accurately the real affinity matrix of the algorithm (computed from the set of all the optimal solutions of the landscape). Ideally the algorithm should be able to reproduce the real affinity matrix and not just an approximation of it. For this reason it seems reasonable the choice of approximating the prediction of the affinity into three different classes of relationships. This choice introduces generality as relies less on the values of the affinity matrix and can be used to reduce the problems of the skewness previously described. The whole range [0-1] of the affinity values is divided into three different classes:

1. No Affinity (affinity equal to 0);
2. Low Affinity (affinity larger than 0 and smaller than a value τ);
3. High Affinity (affinity larger than τ).

Where τ is a threshold to distinguish LOW affinity from HIGH affinity pairs. In this case the assumption made is that there isn't a significant loss of performances by the approximation given by the classification and that actually the use of a classification based strategy might contribute to introduce more generality. Experimental analysis can easily prove whether this assumption holds.

On the Distinction between NO and LOW Affinity Instances

Operating a clear distinction between affinity connections of each different class is fundamental to avoid providing false information to the classifier. With regards to this, a further problem arises when it is necessary to distinguish between the affinity connections belonging to the NO-AFFINITY and LOW-AFFINITY classes. This is due to the fact that during the update process, an affinity connection that is present in the initial population of the algorithm and that never occurs in the remaining generations, will decrease its value approaching 0 but will never reach this value. Classifying this as a LOW-AFFINITY relationship might be misleading. For this reason, a transformation can be applied to the values of the affinity matrix by converting to 0 all those affinity matrix whose value is less than a threshold θ. All the experiments described in this chapter have been carried out with $\theta = 0.0001$.

Threshold Detection for Classification: a Clustering Based Approach

The choice of the threshold τ appears to be important for the dynamic of the algorithm, as a wrong value can have important consequences on the performance of the algorithm, due to the mislabelling of training instances. It is therefore essential to investigate more on the optimal value to choose. There are several factors that can influence the choice of this value. An aspect to consider is the degree of convergence that a population can reach, which can cause the affinity to shift towards larger values. This can be ascribed to several factors such as the shape of the landscape (whether it is multi modal or not), its ruggedness as well as the optimization algorithms itself in the way diversity within the population is handled, or its specific parameter configuration. Last but not least, the stochastic component of the EAs can lead the same algorithm with the same configuration to produce affinity matrices with different degrees of convergence, due to the exploration of different areas of the landscape and/or to the discovery of a solution with a much better quality.

Although understanding how these aspects influence the convergence of the algorithm can be useful, tracking down all these issues can be an extremely difficult task. A possible approach to deal with this issue is that of grouping the affinity values in the three different classes using a clustering algorithm. Such task is performed through the use of the Expectation Maximization algorithm.

Class Imbalance

The skewness of the data, after the process of labelling into the three different affinity classes, causes the dataset to have a great amount of imbalance between the three classes. In particular, the number of samples belonging to the NO Affinity class will be much higher than the others. This will prevent the model to predict correctly the LOW and HIGH affinity instances as it will be much safer to classify as NO affinity in order to reduce the classification error. To deal with this problem, an uniform resampling of the minority classes is performed in a way that the final amount of instances belonging to each class is matched. The resampling is performed once the AM is obtained at the end of the execution of the algorithm on every instance.

Error Types

A final remark is relative to the misclassification errors and how they affect the outcome of the algorithm. It is reasonable to expect that classifying a NO AFFINITY into a HIGH AFFINITY pair (or vice versa) has a higher negative impact on the performances of the algorithm than classifying a LOW AFFINITY into any of the other two classes. In order to reduce the risk of this type of error, the learning system makes use of the Weka *CostSensitiveClassifier*, which combines a base classifier with a cost matrix that is used to assign a weight to the training instances according to the misclassification cost. The cost matrix CM that is adopted is:

$$\mathbf{CM} = \begin{array}{c} \\ N \\ L \\ H \end{array} \begin{array}{c} \begin{array}{ccc} N & L & H \end{array} \\ \begin{pmatrix} 0 & .25 & 3 \\ .25 & 0 & .25 \\ 10 & .25 & 0 \end{pmatrix} \end{array} \tag{7}$$

where each value corresponds to the misclassification cost of classifying class i (row) into class j (column). Labels assigned to rows and columns N,L and H refer respectively to the three affinity classes NO, LOW and HIGH. The value of CM is obviously equal to 0 where the classification is correct (CM_{NN}, CM_{LL} and CM_{HH}). A smaller cost of 0.25 is assigned to the low severity misclassifications (CM_{HL}, CM_{NL}, CM_{LH},and CM_{NL} and a larger cost to the high severity ones (CM_{HN} and CM_{NH}). The asymmetry between these two costs is due to the much smaller cardinality of the HIGH affinity class.

5.3.3 Learning Strategy

Once the data from the problem instance has been extracted, pre processed and converted to a dataset as previously described, it is possible to train a model for each problem instance encountered, in order to build a collection of models corresponding to all the problem instances encountered in the past. For this reason, it looks reasonable to adopt a strategy based on an ensemble of learners, in order to perform a prediction of the affinity between the pair of tasks of a new problem instance. The reasons behind this choice are several. Firstly, this scenario appears to be particularly fit for the use of an ensemble technique, and the learning system is provided with an increasing number of models, as ensemble methods are commonly used in online learning. Secondly, as mentioned in the research questions, it is preferable to avoid the use of a domain related distance measure to establish which of the encountered cases is the closest to the current one. In fact, the ensemble strategy that is being adopted is completely independent from the domain and can immediately reused in a different context. Furthermore, it is reasonable to expect that when none of the previously encountered cases is not similar enough to the current one, the use of an ensemble strategy can improve the results. Ensemble methods are in fact known to perform good in the task of reducing the classification errors. By assigning a weight to each classifier of the ensemble it is possible to reduce the risk of misclassification and to improve the performances of the learner. The rest of the section introduces the base learners used to generate single instance models as well as the ensemble strategy that is adopted to perform the classification task.

Training a Single Instance Model using Random Forests

Since it is likely the data will vary considerably between different problem instances, it is important to adopt a model with a great degree of generality that reduces the risk of overfitting. Moreover, it is beneficial for the learning system to reduce the number of features in the model. A further condition to improve the performance of ensemble models is that each base learner should possibly be trained on different data and or with different features. Finally, it would be useful to reduce the number of parameters of the model itself, as it can be expected the best configuration to vary across different instances. Performing a parameter optimization analysis, besides being computationally expensive, might not be effective in a online learning context.

Therefore, the learning system adopts a Random Forest Classifier [14] as its base learner. The choice of this learner is due to several reasons. Firstly, since Random Forests are a collection of Random Trees, they can perform a feature selection by preventing the trees to grow beyond a certain depth. Therefore, only the features that explain better the data are kept in the trees. Secondly, the randomness introduced by the use of such technique can be beneficial for the performance of the ensemble learning system. Finally, Random Forests proved to perform reasonably well in a very reasonable time. Table 5.2 shows some experimental evidence that supports our choice. The performance of a Random Forest model trained with two different configurations is evaluated. Performances are evaluated using the True Positive Rate (TPR), which measures the percentage of correctly classified instances for each class. The first configuration is a forest trained using 100 Random Trees. The depth of each tree is set to 0, meaning that there is no limit to the maximum depth of the tree. The second configuration limits the number of Trees to 10 and their depth to 5. The configurations are tested on two different datasets, obtained after the execution of the MAENS* algorithm on the *egl-e1-B* and *egl-s4-A* CARP instances. These represent respectively the instances with the smallest and the largest number of vertices in the egl benchmark dataset, and therefore can provide an indication of the scaling capability of this system.

The *egl-e1-B* instance is composed of 12480 training instances after resampling. It is immediate to see how both configurations are capable of obtaining a perfect or close to perfect classification for the *HIGH* affinity instances, and a fairly good classification for the instances belonging to the *NO* affinity class. The amount of time required to perform this task is however considerably different. In fact, the second configuration runs only for a fraction of the time required by the 100 trees version. By looking at the largest instance, *egl-s4-A*, it is possible to understand the scaling performances of the system in terms of runtime. In fact, the runtime of the classifier grows approximately with a factor of 2 with the size of the new instance. In terms of classifier performance, the slower version achieves an higher TPRate in all the classes. In particular, there is a significant difference for the *LOW* affinity class, where the faster classifier fails to predict correctly more than the 2% of the instances, while the slow one predicts correctly half of the instances.

This experimental analysis suggests that the use of a random forests with just 10 base learners and with a maximum depth of its tree equal to 5 can be enough to train base learners that could explain well each instance with regards to the *NO* and *HIGH* affinity classes, without a dramatic loss in terms of performance. Moreover, the adoption of such a configuration contributes to reduce the overfitting of the models, and consequently sustains and improves the generality of the learning system. Finally, the reduced execution time required by the classifier to train each model, which is almost neglectable in the case of the fast configuration, reduces the necessary overhead required by the introduction of this learning phase in the system.

Classification using Ensemble Models

Once the classifier system has been updated with a new single instance model, a new dataset is generated by extracting the input features from a new problem instance. In order to identify the best strategy to classify such dataset, it is necessary to take into account that the models included in the ensemble have been trained with different datasets and might focus on different features. It is therefore necessary to adopt a strategy that considers the

Table 5.2: Training a Random Forest on the egl-e1-B CARP instance with two different configurations. For every configuration the table reports the number of base learners (column *learners*) and their maximum depth(column *depth*). The models are obtained after the execution of the MAENS* algorithm + resampling. Evaluation is performed through 10-folds cross-validation on the training test. Values reported in the table show the True Positive Rate for the three classes, respectively in columns *NO*, *LOW*, *HIGH*, as well as the runtime (column *runtime*).

Dataset	Size	Learners	Depth	NO	LOW	HIGH	Runtime
e1-B	12480	100	0	0.873	0.010	1.000	1.01
e1-B	12480	10	5	0.694	0.000	.998	0.07
s4-A	220480	100	0	0.959	0.486	1.000	38.75
s4-A	220480	10	5	0.820	0.025	1.000	2.62

variety of the models included in the ensemble. For this reason, the learning system has been developed to adopt an approach similar to the Dynamic Classifier System(DCS) of [143]. The behaviour of the algorithm can be summarized in the following steps:

1. A weight, corresponding to the confidence of the classifier, is assigned to every classifier included in the ensemble;

2. While in DCS the confidence of a classifier is obtained by measuring how much the current instance fits the data that has been used to train it, the learning system builds a *confidence measure* by computing the Class Probabilities (the probability of belonging to each class according to the classifier);

3. The Class Probability of each classifier is averaged to build a global Class Probability;

4. The classification is performed by assigning the class that has the largest Global Class Probability;

5. To prevent the creation of further parameters, an affinity value equal to 0 for the *NO-AFFINITY* pairs, equal to 0.25 for the *LOW-AFFINITY* ones and equal to 0.75

for the *HIGH AFFINITY* connections. These values have been chosen through the observation of the average affinity of the three classes. Further studies might reveal better affinity values to use or dynamic assignment strategies to compute such values at runtime.

5.4 Using Predicted Affinity in New Problem Instances

Once the AM for the current problem instance has been predicted, it is possible to inject this information in the algorithm such that the search is driven towards the connections with the largest affinity. There are multiple ways to use effectively this information. This section, analyzes just one of such possibilities, relative to the definition of an Affinity-Based Local Search (ALS). A discussion relative to alternative ways to use the information contained in the AM within the MAENS* algorithm, and more in general within EAs, is included in section 5.6.

5.4.1 Affinity-Based Local Search

Given an affinity matrix A, it is possible to define a local search operator that improves the average affinity of a given solution. Once a neighbourhood of the possible moves is generated, the local search computes the variation of the average affinity of the initial solution produced by each solution in the neighbourhood. It is possible to define the average affinity of the solution x as:

$$\overline{\mathrm{aff}}(x) = \frac{\sum_k \sum_{a,b \in R_k} A(a,b)}{\sum_k \dfrac{size(R_k) * (size(R_k) - 1)}{2}} \tag{8}$$

where R_k is the k-th route in x and $size(R_k)$ returns the number of tasks served within R_k, and $A(a,b)$ is the affinity between the two tasks a and b. It is immediate to see that applying a move operator to a solution induces a variation in the average affinity of the solution. This variation will be positive if the average affinity of the solution has increased, and negative otherwise.

The affinity-based local search can therefore be summarized by the following steps:

1. Generate a neighbourhood of the initial solution x using one or a set of move operators;
2. Assign a score to each solution y in the neighbourhood, using the formula:

$$score(y) = \Delta\overline{\text{aff(y)}} \tag{9}$$

   where $\Delta\overline{\text{aff}}(y)$ is equal to $\overline{\text{aff}(x)} - \overline{\text{aff}(y)}$;

3. Identify the solution y with the largest score;
4. Replace x with y;
5. repeat the process on the new solution until no move exists with a positive score.

5.4.2 Discussion: Effective Usage of ALS

Balancing the use of the ALS is a key factor for the successful use of this technique. There are several considerations to make:

1. The introduction of an extra operator translates into an increment of the computational time of the algorithm, when the termination criterion is a fixed number of generation as in the case of MAENS. This is particularly relevant when a local search operator is considered, such as the ALS. As one of the possible ways to improve the results of the algorithm is that of decreasing its runtime while achieving comparable solution quality, it is important to reduce the impact of this operator to a minimum;

2. Driving the search towards specific areas of the landscape can be harmful as it can lead to premature convergence, decrease the capability of the algorithm to escape local optima and, ultimately, lead to poor results. Once again it is important to balance the dosage of the ALS to reduce this risk;

3. It is reasonable to expect that the ALS will be more useful in the initial part of the search, as it will help the algorithm to focus towards safer areas of the landscape;

4. As it is realistic to expect the prediction to be not always accurate, it is necessary that the algorithm is capable of detecting and reacting to bad predictions by decreasing the affinity matrix of poor connections.

Given these considerations, the probability to adopt the ALS during each generation is decreased as the algorithm proceeds in its search. The formula to calculate such probability is:

$$p(t) = \frac{k}{k+t+1} \tag{10}$$

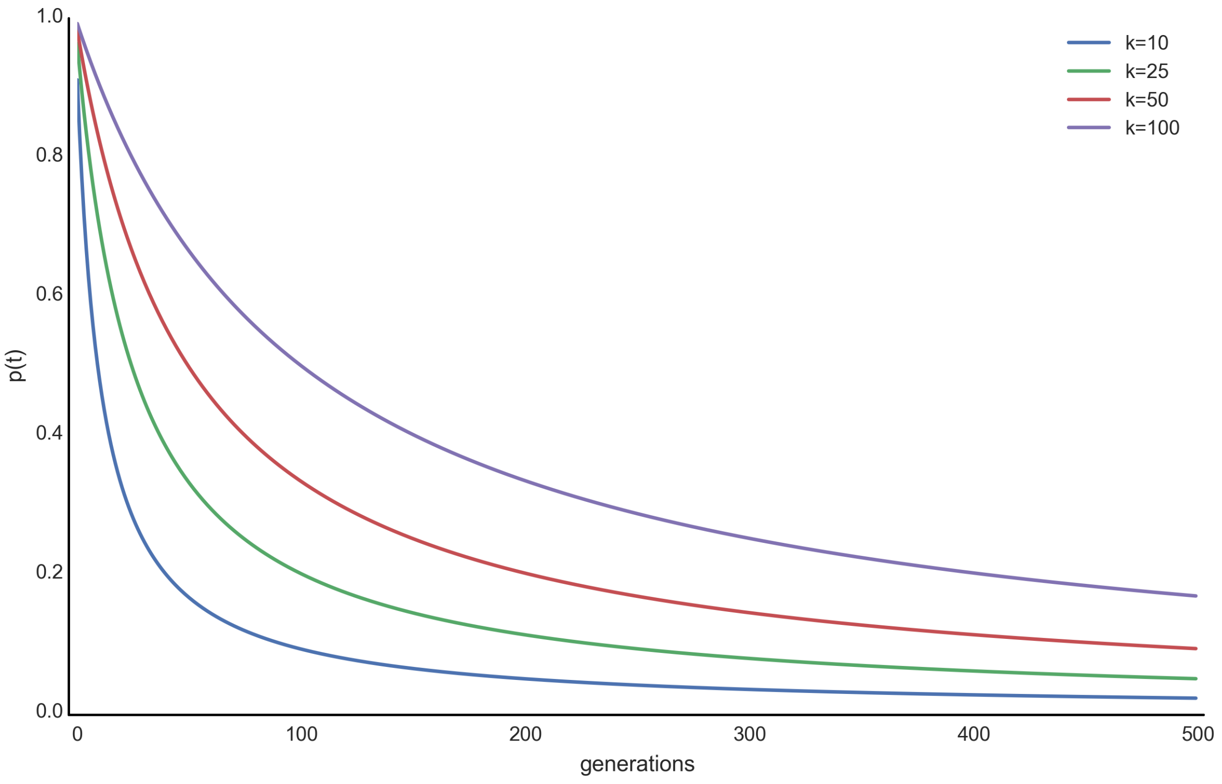
where t is the current timestep or generation and k is a parameter to the determine the steepness of the function. The behaviour of the function is illustrated in figure 5.2, where the probability of performing the ALS is calculated when k assumes respectively the values of 10, 25, 50 and 100. These are also the values that have been considered in our experimental test.

This probability function is also useful for the calculation of the affinity matrix that is performed during the search. As explained in section 5.2.2, the algorithm requires the use of two weights associated to the *update rule* necessary to modify the affinity matrix after every generation. In the MAENS* algorithm, this is easily performed by considering the average fitness of the population of two consecutive generations. If however it is necessary to take into account the predicted affinity matrix, it is necessary to choose a different strategy to determine the weights of the update rule. Moreover, it is reasonable to trust more the prediction during the initial phases of the search and less otherwise. For these reasons, this very probability function is adopted (with or without a different value of k) to determine the weight of the predicted affinity matrix on the computation of the AM for the next generation. The adoption of this probability function translates into the generation of two novel parameters:

1. k_U *update rate*: the value of k that regulates the $p(t)$ function and determines the weight of the predicted affinity matrix when the AM for the next generation is computed. Since this value is always within the range $[0, 1]$, it is easy to compute the weight of the affinity matrix relative to the current population as $1 - p(t)$;

2. k_L *ALS probability*: it represents how fast the probability of using the ALS during the search decreases, and corresponds to the value of k in the $p(t)$ function.

Figure 5.2: Variation of the values of $p(t)$ with k is 10, 25, 50 or 100



5.4.3 MAENS*-III

The MAENS*-III algorithm, described in this chapter, combines the CARP meta-heuristic MAENS*[21], the use of the Affinity-Based Local Search and the learning system to extract the information, train the models and predict the affinity of the new CARP instances.

The execution of the algorithm can be divided in two phases. In the first *training* phase, the algorithm has the objective to gather information in order to train its learning system. For this reason, the use of the ALS is inhibited, as the learning system is not "mature" enough to be used to predict the AM. The algorithm, can be summarized by the following steps:

1. The algorithm is presented with a CARP instance. A set of 34 features, described in section 5.3.1, is extracted from the instance, for each pair of its required tasks;

2. If the algorithm is not in its training phase, MAENS*-III will predict the task affinity between each pair of tasks of the current CARP instance, using the ensemble strategy discussed in section 5.3.3;

3. The MAENS* algorithm is now used on the specific instance. If the algorithm is in its training phase, the use of the ALS is inhibited. Otherwise, the algorithm will adopt the ALS and will update the AM during its execution, according to the strategy described in section 5.4.2. At the end of the run, a final Affinity Matrix A^i is extracted;

4. A training set is generated by combining the 34 input features previously computed for each pair of tasks, and the respective task affinity extracted from A^i, corresponding to the output feature;

5. Extremely small affinity values are converted to 0, as described in section 5.3.2;

6. The EM algorithm is used to cluster the training instances into the three classes with labels *NO*, *LOW* and *HIGH*, as described in section 5.3.2;

7. Resampling is operated on the training set, to prevent imbalancing between classes, as discussed in section 5.3.2;

8. A Random Forest Classifier with Cost Sensitive Matrix is trained using this training set, as detailed in sections 5.3.3 and 5.3.2;

9. The new model is then added to the ensemble of the existing models. The system is now ready to tackle a new instance.

5.5 Experimental Analysis

In order to answer the research questions relating to this chapter, an experimental analysis is performed using the known benchmark dataset for CARP *egl*[28], and considering the MAENS* algorithm as the case study.

5.5.1 Simulating Misclassification

This first experiment has been designed to answer to the following research questions:

1. Does the *affinity* provide good enough information to improve the performance of MAENS*?

2. What are the results of the algorithm when in presence of misclassification, and to what extent such results can be considered useful?

3. What is the effect of misclassifying for a particular class? Is it reasonable to claim that misclassifying a LOW Affinity pair is a less severe error than misclassifying a NO or HIGH affinity pair?

The investigation of these research questions is carried out through the use of a simulated environment, where the algorithm is provided with an artificial Affinity Matrix. Such Affinity Matrix is obtained through the execution of the MAENS* algorithm on the very same instance and is therefore not predicted. Once the AM is created in this way, it is possible to artificially introduce misclassification to test the behaviour of the algorithm under different circumstances. In particular, the following scenarios are simulated:

1. To understand whether the use of the Affinity Matrix can be effective to boost the performance of the algorithm, with what degree of misclassification, and whether the conversion between real values and class labels incurs a loss of information, the algorithm is presented with an artificial AM with correct classification equal to $100\%, 75\%$ or 50% for the three classes;

2. To understand whether the misclassification on the LOW affinity class has a smaller influence on the results of the algorithm, the algorithm is tested with an artificial AM with 0% misclassification for NO and HIGH affinity classes and with 100% for the LOW affinity class.

The results of these simulations have been obtained on a subset of the *egl* CARP benchmark, and are summarized in table 5.3

No Misclassification

In the first experiment, the algorithm is assessed using an artificial AM which predicts correctly (using the information obtained from a previous execution of the MAENS* algorithm) the affinity relationship between tasks in every pair and for all classes. The experiment has been performed setting the value of the two parameters k_U and k_L to 25. Different parameters settings might be more effective for this specific case. The results of MAENS*-III, included in table 5.9, in the column labelled $[1.0, 1.0, 1.0]$ show that the algorithm outperforms MAENS* in almost all the instances considered. In particular, it

Table 5.3: Comparison of the results achieved by MAENS* and by MAENS*-III with simulated AM having different degrees of misclassification. The versions of the algorithms are compared in terms of average fitness of the best solution found over 30 independent runs and runtime. In both cases average, standard deviation and best value are provided. Statistically significant results are highlighted in bold

instance		1.00,1.00,1.00			.75,.75,.75			.50,.50,.50			1.00,0.00,1.00		
		avg	std	best	avg	std	best	avg	std	best	avg	std	best
s2-a	fitness	9921.30	22.21	9889.00	9928.60	22.76	9895.00	9928.63	32.52	9881.00	9919.57	21.16	9889.00
	runtime	699952.77	453189.15	5992.00	757024.10	543242.02	41630.00	1044553.83	366715.55	304357.00	866551.17	444088.37	26501.00
s2-c	fitness	16465.67	40.67	16426.00	16470.93	34.54	16430.00	16480.30	40.13	16430.00	16473.67	31.28	16430.00
	runtime	447026.43	299605.76	14579.00	539210.67	260561.19	111681.00	772163.07	220671.77	276017.00	408692.90	332968.96	21347.00
e4-b	fitness	9027.67	13.89	9001.00	9030.30	12.37	9010.00	9033.80	14.25	9005.00	9028.33	14.36	9000.00
	runtime	534474.03	223145.80	144558.00	507836.50	227708.58	148435.00	530554.00	218697.88	76819.00	521874.50	208379.19	40889.00
s4-a	fitness	12369.10	41.71	12308.00	12360.90	50.88	12257.00	12392.60	27.87	12316.00	12379.40	46.52	12300.00
	runtime	1015580.93	542892.46	55008.00	1156450.10	419691.78	449533.00	1509743.33	338513.58	671865.00	1025350.67	528719.79	26513.00
e1-c	fitness	5595.80	3.09	5595.00	5595.47	2.56	5595.00	5595.00	0.00	5595.00	5595.60	3.29	5595.00
	runtime	57625.20	58097.28	8216.00	35410.53	30252.94	2109.00	43996.13	37138.78	5902.00	49889.03	43587.78	596.00

is immediate to notice how the runtime of the algorithm has decreased considerably for all except 5 of the instances and has achieved comparable results in the rest, with an average runtime decrease of 68% over the original runtime, and with average runtime for each instance ranging in the interval between 11% and 113%.

Along with a reduced execution time, the algorithm is capable of improving the optimization results of MAENS* in six of the instances considered (*s2-a*, *s2-c*, *s3-a*, *s4-a*, *s4-b*, and *s4-c*) and to achieve comparable results in the remaining ones. It is important to notice that in the case of the instances *e2-b*, *e2-c*, *e3-c*, the algorithms MAENS* and MAENS*-III achieve comparable fitness because they both reach results that are very close to the optimum or to the best known solution in literature. Finally, it is also worth mentioning how MAENS*-III outperforms MAENS* in terms of the fitness of the best solution found across the 30 different executions in all the 6 instances with statistically different results.

25% **Misclassification Rate**

In this second case, the algorithm is assessed using an artificial AM which predicts correctly (using the information obtained from a previous execution of the MAENS* algorithm) the affinity relationship between tasks in the 75% of the cases. The experiment is performed on just 5 of the instances where the version with perfect classification achieved statistically significant results. This result is simulated in the three different classes. Misclassification is attributed to the other classes with equal probability. The analysis of the results of MAENS*-III, which are included in table 5.3, in the column labelled $[.75, .75, .75]$ show that the algorithm outperforms MAENS* in all the instances considered. In particular, it is immediate to notice how the the performances of the algorithm are considerably similar to the version with perfect misclassification. The algorithm performs well for the instances *s4-A* and *e1-C*, where it achieves an improvement respectively for the optimization in the former instance and for the runtime in the latter. Moreover, the algorithm manages to achieve comparable but slightly worse results in the remaining instances, with the exception of instance *s2-A*, where the fitness is considerably worse. Such results seem to indicate that it is not necessary to obtain a perfect classification and that a certain amount of misclassification can still lead to a certain amount of improvement.

50% **Misclassification Rate**

In this third experiment, the algorithm is assessed using an artificial AM which predicts correctly (using the information obtained from a previous execution of the MAENS* algorithm) the affinity relationship between tasks in 50% of the cases. This result is simulated in the three different classes and misclassification is again attributed to the other classes with equal probability, on the same group of 5 instances. The analysis of the results of MAENS*-III, which are included in table 5.3, in the column labelled $[.50, .50, .50]$ show that the performances of MAENS*-III are worse than the other versions, in all the instances considered. In particular, it is immediate to notice how both the runtime and the fitness of the algorithm are always larger than that of MAENS*, with the exception of the results of *e1-c*. The results of this experiment suggest that a 50% is probably not enough to improve the performances of the algorithm, either in terms of its optimization ability and in terms of its runtime.

100% **Misclassification Rate in Low Affinity Class**

In this last experiment, the algorithm is assessed using an artificial AM which predicts correctly (using the information obtained from a previous execution of the MAENS* algorithm) the affinity relationship between tasks in every pair and for the *HIGH* and *NO* classes, and misclassifies all the pairs contained in the class *LOW*. As in the previous cases, the misclassification is handled reassigning the pairs to the other two classes with equal probability. The results of this experiment, conducted on the same set of 5 instances, are included in table 5.3, in the column labelled $[1.0, 0, 1.0]$. In this case, the algorithm is capable of achieving results that are comparable to the version with perfect classification in all of the instances. This is valid for both types of performances that are analyzed. The analysis of such results suggests that the *LOW* affinity class has a poor influence on the effectiveness of the algorithm, and that for this reason it is recommendable to design a classifier that focuses more on the correct prediction *NO* and *HIGH* affinity pairs.

Analysis and Discussion

The analysis of the results previously described shows some interesting properties of the algorithm and confirms some of the hypotheses of this work. In particular:

1. The first important result of this set of experiments is the confirmation that, provided that the algorithm is capable of predicting the affinity with a certain amount of accuracy, MAENS*-III can be effective to improve the results of MAENS*. Moreover, this confirms that the Affinity-Based Local search is indeed successful in its aim of reusing the information contained in the AM to boost the performances of the algorithm;

2. A second aspect relates to the conversion of the affinity values into class labels. The fact that the algorithm is capable of achieving considerably good results when the classification is correct, suggests the idea that the loss of information is neglectable and that the most significant task is rather that of assigning the correct class label;

3. The values that have been assigned to generate the AM are respectively 0 for the *NO* Affinity class, .25 for the *LOW* Affinity class, and .75 for the *HIGH* Affinity one. Such values have been chosen as they appear to be in the range of values observed in the AM generated by the algorithm, and seem to be robust enough for the purpose. Alternative ways to rebuild the affinity values might actually improve the performances of the algorithm;

4. The performance of MAENS*-III with 25% of misclassification rate and with 50% shows that the algorithm is capable of achieving remarkable results in some instances with a misclassification rate within this range;

5. It is clear to see that when the classification is not perfect, the algorithm has to use resources to fix the wrong affinity values. This is particularly evident in the 50% case, where the runtime of the algorithm is considerably higher;

6. The results achieved by the version of the algorithm that misclassifies LOW affinity pairs can be explained by the fact that the algorithm requires less resources to fix a LOW affinity pair that has been attributed to one of the other two classes, as the numerical difference between their median values is smaller. It also supports the decision to adopt the Cost Sensitive Classifier to weight the misclassification errors;

7. Finally, such results have been obtained with the two parameters k_U and k_L set to 25. These value were not obtained through a parameter optimization process, and for this reason might not be the best combination. It is reasonable to expect that larger values of the parameters might lead to better results when the misclassification is perfect and might not perform too well in the other cases. The best parameter setting must be chosen considering the "real" classification ability of the algorithm.

5.5.2 Assessing MAENS*-III

This experiment has the purpose to analyze the behaviour of MAENS*-III algorithm and to compare its results to those achieved by MAENS*. In particular, the experiment wants to answer to the following research questions:

1. Does the prediction ability of MAENS*-III improve as more and more models are added in memory?

2. What is the performance of MAENS*-III in terms of runtime with respect to MAENS*?

3. What is a good value for the k_U and k_L parameters?

Since the introduction of the memory component in MAENS*-III transforms this EA in what is in fact a stateful system, it seems reasonable and useful for visualization to consider the sequence of results of the algorithm on every problem instance as the values of a time series. The behaviour of the algorithm is tested when the two parameters for the learning rate and the ALS probability, k_U and k_L, are considered. Figure 5.3 shows the behaviour of the algorithm when the parameters are set respectively to 50 and 10. The top plot shows the variation of the True Positives (TP) Rate, which can be defined as the ratio between the number of instances correctly classified over the total number of instances, during the execution of the algorithm over the 24 instances of the *egl* dataset. Each line shows the average TP rate achieved for the three classes HIGH, LOW and NO affinity, and a 95% confidence interval. The middle plot, labelled *fitness*, shows the variation of the best fitness found by the algorithm compared to those achieved through an execution of the MAENS* algorithm on the same instance. Results are normalized in order to allow the comparison across different problem instances, using the formula:

$$A(I,r) = \frac{\frac{1}{N}\sum_{r=1}^{N} A(I,r))}{\frac{1}{N}\sum_{j=1}^{N} \text{MAENS}^*(I,j))} \tag{11}$$

where $A(I,r)$ corresponds to the best result achieved by the algorithm A (either MAENS* or any version of MAENS*-III) on problem instance I at its $r-th$ independent execution, and $\text{MAENS}^*(I,j)$ is the best result achieved by MAENS* on problem instance I at the $j-th$ independent execution. The horizontal line associated with MAENS* indicates the mean result achieved by the algorithm, along with a 95% confidence interval. Since CARP is a minimization problem, it is immediate to see how values larger than 1 correspond to a worse performance and viceversa. Finally, the bottom plot, labelled *runtime*, shows the same comparison in terms of the average runtime achieved by each algorithm on every problem instance. In both the *fitness* and *runtime* plots, a significance test was performed to verify whether the two groups of results are statistically significant, using the Wilcoxon Rank Sum Test[136], with a 0.05 significance level. Instances with statistically significant results according to this test are recognizable by the presence of a dot marker. It is worth noting that at the beginning of the execution, since there are no models stored in memory that can be used to create a prediction for the initial instance (*egl-e1-A* in this case), the MAENS* algorithm is used, whose AM computed at the end of its execution will be the first training set used to train a single instance model.

Prediction Ability

The analysis of the results of the experiment described in 5.3 provide some evidence to answer to some of the research questions indicated at the beginning of this section. For the specific sequence of instances considered in this experiment, it is possible to notice that the algorithm starts with a poor TP rate relative to the HIGH affinity class (less than 0.2 in the first predicted affinity matrix). However, this value tends to increase quickly and to reach 0.6 after just five models. Conversely, the TP rate of the LOW affinity class quickly converges to 0, due to the effect of the Sensitive Cost Matrix in the classifying system. The TP rate of the NO affinity class shows an overall tendency to increase, with a value that oscillates around 0.8 after half of the instances have been encountered. The TP rate of the HIGH affinity class is the one that appears to be more sensitive to the introduction of a novel instance considerably different from the ones encountered so far. This might explain the drop in the TP rate for the HIGH affinity class once the second group of instances (starting with the letter s) is encountered. Nevertheless, the algorithm is indeed capable of partially predicting the correct affinity of the *egl-s1-A* instance using the information learned in the previous ones. The TP rate of the HIGH affinity goes above

0.60 after a few instances, and reaches values higher than 0.8 for the prediction performed in the last instances of the sequence. It appears reasonable to claim that the algorithm is indeed learning from the past experience and that it is also capable to transfer and some of the information learned on considerably different instances to the new ones.

Parameter Sensitivity, Fitness and Runtime Analysis

In order to analyze the sensitivity of the algorithm towards the parameters k_U and k_L, a further experiment was designed considering the values of those two parameters within the range $[10, 25, 50, 100]$. The results of such experiments are reported in the tables 5.4, 5.5, 5.6 and 5.7. The analysis of the results achieved across the whole set of experiments shows several interesting aspects. In particular:

- In terms of fitness of the best solution, the number of instances with statistically significant differences between MAENS* and MAENS*-III is limited to 7 instances out of 18 in the largest case, considering that the remaining 6 instances of the *egl* dataset are considered *easy* instances where the best known result is achieved by both algorithms during every execution;

- The best results are achieved with the combinations $k_L = 50, k_U = 10$ and $k_L = 10, k_U = 50$, where 4 instances have statistically significant results with better mean;

- In terms of runtime, it is predictably noticeable that the use of MAENS*-III translates into an increment of the runtime of the algorithm. This is particularly evident when the parameter k_L grows larger than 10, since the only instances with statistically significant execution time have, in effect, a larger average runtime;

- The summarized results of table 5.8 confirm that an effective dosage of the ALS is critical to the success of the algorithm, as discussed in detail in section 5.4.2. This task appears to be extremely complex, as the behaviour of the algorithm can be affected by a plethora of factors, including the topology of the CARP instance, the initial random seed, the choice of the classifier and its configuration, the sequence of instances encountered so far, as well as the interplay between the parameters k_U and k_L;

- In terms of runtime, it appears to be safe to limit the parameter k_U to values equal or lower than 10. Between the two parameters, this is the one that seem to influence more rapidly the runtime of the algorithm. This is particularly evident from the results of table 5.4.2, where the number of instances having a runtime statistically longer than MAENS* is extremely correlated with the increment of the value of k_U, with the only exception of the combination $k_L = 10$ and $k_U = 50$, where there only one of the instances shows a worst performance;

- The results discussed in this section have been obtained without the use of a *training phase*, where the MAENS*-III is adopted only after a certain amount of instances have been tackled. The experiments described in section 5.5.3 point out what could be the best amount of training instances regardless of the specific instance order. If these were to be considered, the amount of instances with worst result would be much less, as they are mostly distributed among the instances that belong to the initial part of the sequence;

- Due to the high computational cost required, the experiments have been performed on a cluster of machines. For this reason, the results relative to the execution time of the algorithm might not be extremely accurate. This is particularly noticeable considering the extremely elevated standard deviation associated with runtime of the algorithm;

- It is possible to notice how the algorithm performs better on certain instances, regardless of parameter combination considered. Instances where this is visible are *s4-C*, *s4-A*, *s2-A*, *s2-C*. In this case, since the affinity matrix is effective to boost the performance of the algorithm, MAENS*-III is capable of achieving best results in most of the combinations, even when the average fitness across the 30 executions is comparable to that of MAENS*. The opposite effect is also noticeable on certain instances, such as *s2-B*, where the the considerably poor performance of the algorithm across most of the combinations is reflected in the worst results in terms of best fitness.

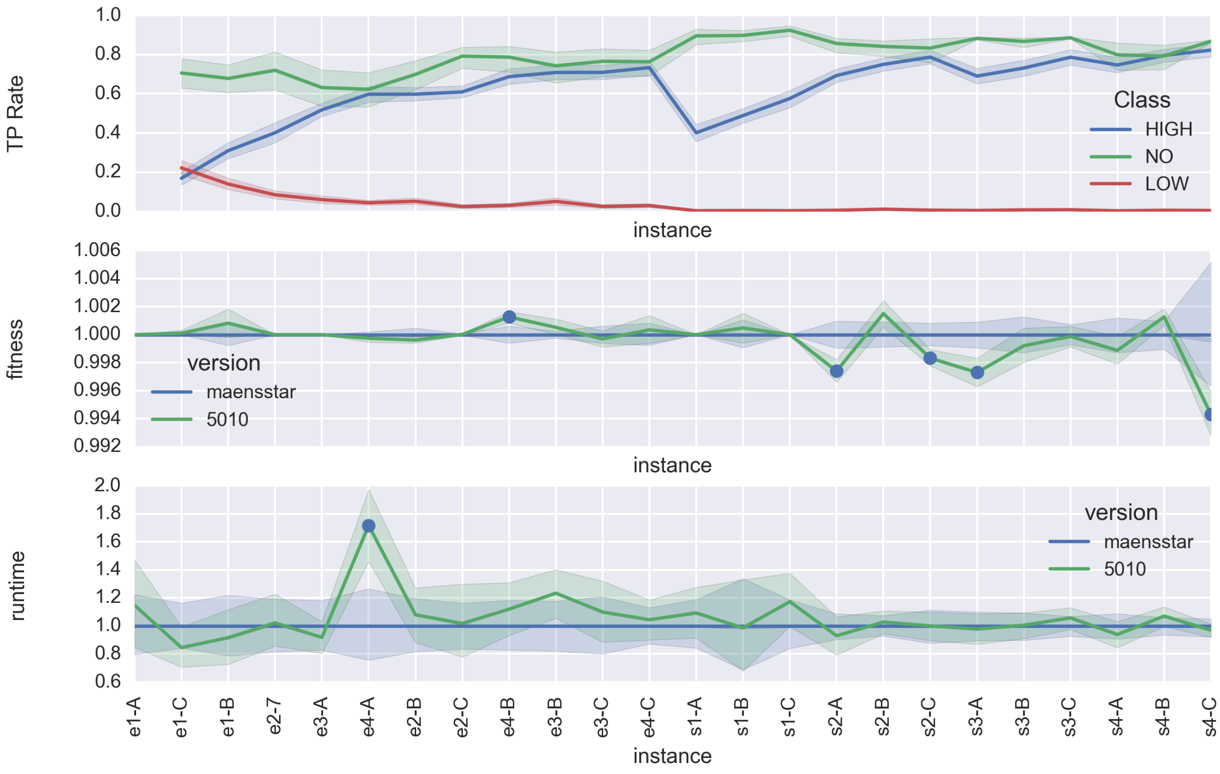
5.5.3 Randomizing the Instance Order

The motivation of this experiment are:

1. Verifying the performance of MAENS*-III regardless of the order given to the problem instances;
2. Assessing how many training instances are necessary for the algorithm to start producing good enough predictions of the Affinity Matrix.

In order to examine that, the algorithm is run with a random order for 50 independent executions, which will prevent the results to be influenced by a specific order and that will help understanding whether the system appears to be robust regardless of the order in which the problem instances are tackled. Although a sample of 50 sequences is most definitely not representative of the whole population, as there are a total of 24! possible sequences, it is still possible to extract some useful information. Figure 5.4 shows the average TP rate for the three classes across the whole benchmark set. As in the previous case, only the first problem instance of the sequence is used for training purposes, while the remaining 23 are tackled using MAENS*-III. The top plot shows the average TP for the three affinity classes, along with a 95% confidence interval (vertical bar) and the minimum and maximum values achieved for each instance (the range band). The analysis of the plot shows that the range of values that the TP is assuming is considerably large for all the instances, proving that the prediction ability of the algorithm is strongly associated with the instance sequence. Nevertheless, some trends in the data can be observed. Firstly, it is quite evident how the use of the cost sensitive matrix is limiting the prediction ability of the LOW affinity class to small values across all instances and regardless of the sequence. Moreover, by looking at the TP rate for the HIGH Affinity class, it is possible to notice how the algorithm prediction is considerably poorer in certain instances (the *s1-** group), as it had already been noticed in the experiment show in figure 5.3. It is reasonable to

Figure 5.3: Results of the MAENS*-III algorithm when the parameters k_1 and k_2 are set to the values 50 and 10. The plots show the average TP rate for the three classes of affinity connections (top plot), the comparison with MAENS* in terms of fitness (middle plot) and runtime (bottom plot). Statistically significant results are identifiable with a circle marker.



assume that this group of instances, where the algorithm prediction struggles the most, is particularly different from the rest. Nevertheless, MAENS*-III manages to achieve a TP rate above .60 in certain sequences.

The consequence of a weak prediction across most of the sequences translates into the algorithm failure to improve the performance of MAENS*, as it is possible to see in the middle and bottom plots, relative to the fitness and the runtime of the algorithm. This is understandably imputable to the lack of training data to support the prediction process. It is therefore reasonable to perform an initial phase of training before being able to trust MAENS*-III to produce statistically sound improvements.

The second research question is a direct consequence of this analysis: how many training instances are enough instances so that MAENS*-III start producing improvements or comparable results? Providing an answer to this question is not an easy task. Firstly, it is evident from the previous experiments that a certain accuracy of the prediction does not have the same consequences on all instances. Secondly, the impact of MAENS*-III can reflect either on the runtime of the algorithm or on the optimization performance, or on both. In particular, when the CARP instance that the algorithm is trying to tackle is particularly easy, the improvement is only reflected in a reduced runtime of the algorithm. Figure 5.5 shows the evolution of the average TP rate for the three classes across the whole 50 random orders. The TP rate of the NO and LOW affinity classes are quite similar across the whole set of experiments. In particular, the former remains around a value of 0.9 regardless of the instance order, while the latter follows the behaviour seen previously, rapidly decreasing its value to 0 in less than 5 instances. With regards to the HIGH affinity class, it is possible to distinguish two phases. In the first phase, which can be considered a learning phase, the TP rate grows from 0.1 to 0.5 during the course of 7 instances. In the following phase, the average TP rate assumes values in the range 0.6 - 0.7, which are kept for the rest of the sequence. It is reasonable to claim that for this particular configuration, and given the benchmark instances considered, a number of 7 instances are enough for the algorithm to converge towards a more stable result.

Figure 5.4: Results of the MAENS*-III algorithm when the two parameters k_1 and k_2 are set to the values 50 and 10 across random sequences of the problem instances. The plots show the average TP rate for the three classes of affinity connections, with a 95% confidence interval and minimum and maximum values (top plot), the comparison with MAENS* in terms of fitness (middle plot) and runtime (bottom plot). Statistically significant results are identifiable with a circle marker

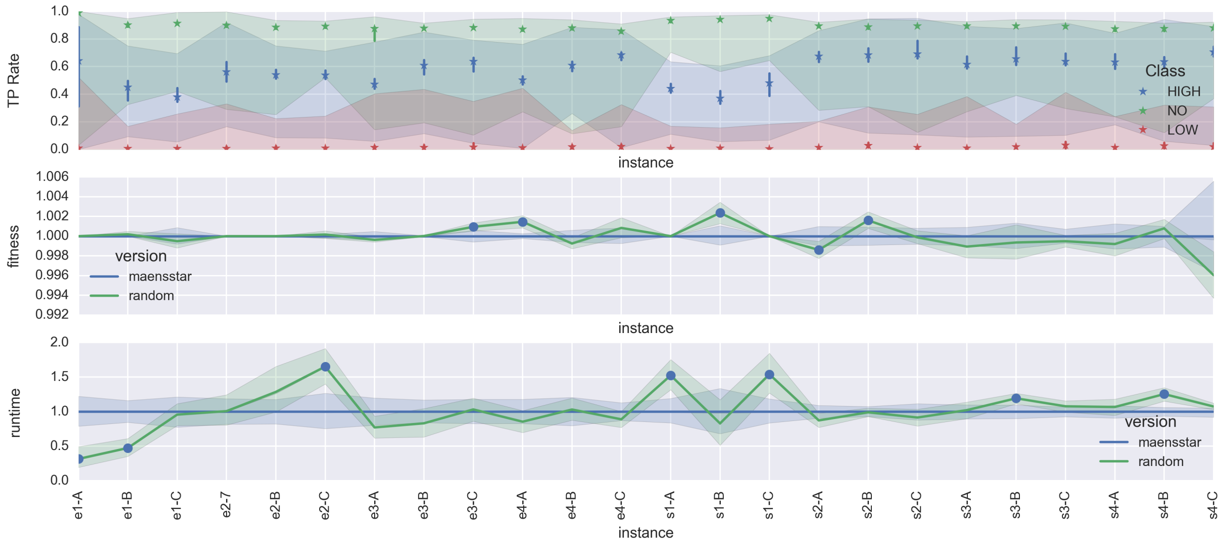
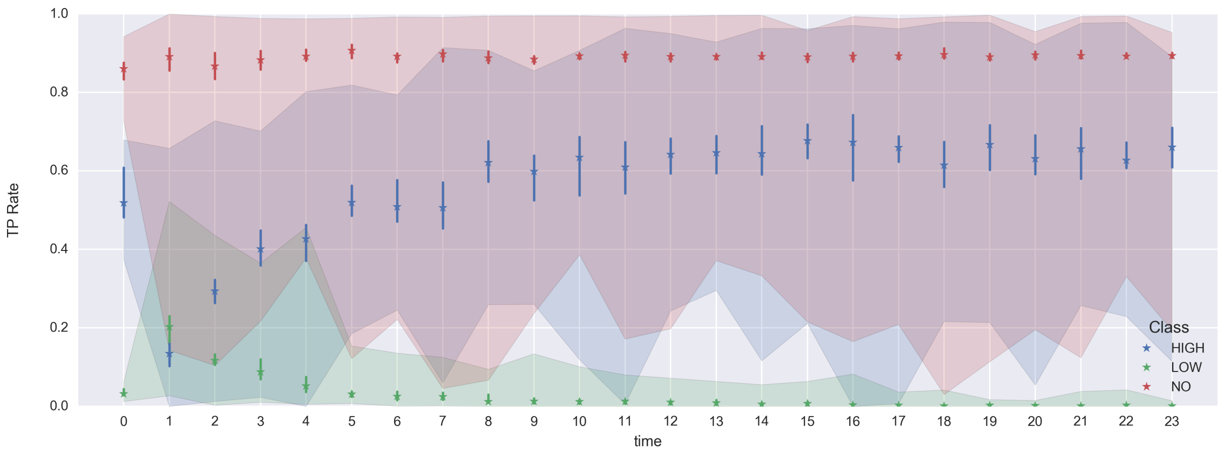


Figure 5.5: Results of the MAENS*-III algorithm when the parameters k_1 and k_2 are set to the values 50 and 10 across random sequences of the problem instances. The plots show the average TP rate for the three classes of affinity connections, with a 95% confidence interval and minimum and maximum values.



5.6 Summary and Discussion

The contributions of this chapter are:

- The definition of a set of 34 CARP features, some of which novel, to describe and measure the information contained in a CARP instance;
- The introduction of the concept of task affinity, its formal definition, and the procedures required to compute and update its values for an evolving population;
- The definition of a novel Local Search operator focusing on the improvement of the average task affinity of the solution;
- The use of a classifier to learn and predict the task affinity of new CARP instances, using the information gained through the use of the MAENS*-III algorithm for already encountered CARP instances.

As mentioned in the introduction of section 5.4, the approach that has been followed in this work is one of the possible ways to reuse the predicted affinity. Extensive experimentation with each of such alternatives would be required to determine whether this approach is the most useful and effective, which might be prohibitive in terms of time. Therefore, it is not possible to exclude that this might not be the most effective choice and that a better way to proceed exists. It is also worth mentioning that not all the work that has been carried out relatively to these research questions has been included in this thesis. In particular, an alternative research direction was followed to design a partial restart strategy using the affinity matrix as a way to reinoculate the information within the new population. This is clearly leaving some unexplored directions for future extensions of this work. In particular, some of the alternative approaches that can be considered are:

- Combining the Affinity-Based Local Search approach to the existing local search operator, where a score function considering fitness, violation and average affinity is used to choose the best individual in the neighbourhood;

- Defining an heuristic to be used within the crossover operator, in order to drive the selection of the what routes to merge based on the evaluation of the affinity improvement of the offspring;

- Influencing the parent selection process of the crossover operator, to assign a larger selection probability to those individuals with an higher average affinity;

- Reducing the execution time of the Local Search Operator, by reducing the size of the neighbourhood that is generated during each iteration, by preventing the evaluation of the neighbours induced by poor affinity-wise moves;

- Influencing the Stochastic Ranking operator to enhance the survival chances of those individuals with an higher average affinity, particularly in situations with great stagnation and poor diversity within the population.

The results of the work described in this chapter provide several answers to the research questions illustrated in this chapter. In particular, it is reasonable to claim that:

- The use of task affinity, when predicted with a good accuracy, can lead to considerable improvements in the optimization ability of the algorithm and to reduce its runtime;

- The task of predicting the task affinity in such a constantly evolving environment is non trivial and requires several steps of pre-processing of the data and the use of ensemble learning techniques to maximize its accuracy;

- In the best cases, the average misclassification rate is equal to 20%, which is an amount of error that still allows to improve the performance of the algorithm in several cases;

- The task of predicting the labels of the three classes having *HIGH*,*LOW* and*NO* affinity might be simplified to the task of distinguishing between *HIGH* and *NO* affinity. However, the presence of the *LOW* affinity class is determinant to reduce the impact of the misclassification errors.

Table 5.4: Results of MAENS*-III with k_U set to 10 and k_L assuming the values 10,25,50,100 on the *egl* CARP instance set. The table compares both fitness and runtime on 30 independent runs. Average values with a boldfaced value indicate a results statistically significant with respect to the results of MAENS*, with significance level smaller than 0.05.

		MAENS*			10,10			10,25			10,50			100,100		
instance		avg	std	best	avg	std	best	avg	std	best	avg	std	best	avg	std	best
e1-A	fitness	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00
	runtime	7.23	4.35	1.48	10.92	6.95	1.49	8.72	5.45	1.36	10.70	7.64	1.23	9.82	5.85	2.29
e1-C	fitness	5595.00	0.00	5595.00	5595.00	0.00	5595.00	5595.00	0.00	5595.00	5595.00	0.00	5595.00	5595.00	0.00	5595.00
	runtime	133.79	59.63	34.92	120.38	61.58	29.04	**105.21**	50.49	43.18	122.97	42.14	44.49	133.78	58.98	47.00
e1-B	fitness	4505.10	10.58	4498.00	4505.30	10.56	4498.00	4504.28	10.41	4498.00	4504.87	10.11	4498.00	4504.14	10.21	4498.00
	runtime	123.67	75.51	2.77	143.82	73.07	3.81	132.56	72.84	14.06	165.80	81.15	5.83	119.66	79.78	4.30
e2-7	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	20.41	10.63	2.48	**15.23**	8.09	1.48	21.31	8.96	7.62	17.77	7.99	3.14	20.81	10.36	4.96
e3-A	fitness	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00
	runtime	65.17	32.55	22.55	60.49	28.74	4.46	68.87	29.77	17.96	63.99	26.88	21.43	82.40	45.03	24.43
e4-A	fitness	6463.80	3.62	6456.00	6462.53	6.44	6444.00	6464.34	4.04	6461.00	6464.03	3.67	6456.00	6464.48	5.96	6446.00
	runtime	232.20	164.13	43.01	**351.61**	166.65	50.66	**351.09**	172.25	46.71	**314.97**	165.84	40.81	299.32	161.28	47.43
e2-B	fitness	6321.63	7.78	6317.00	6318.37	3.74	6317.00	6320.00	4.54	6317.00	6319.67	4.80	6317.00	6319.41	4.62	6317.00
	runtime	195.95	105.84	43.71	178.42	96.28	33.33	196.44	108.27	22.99	186.14	120.90	17.15	222.92	106.58	51.00
e2-C	fitness	8335.40	1.20	8335.00	8335.27	1.00	8335.00	8335.28	1.00	8335.00	8335.27	1.00	8335.00	8335.14	0.72	8335.00
	runtime	143.19	66.34	49.48	132.10	97.15	33.57	129.04	90.47	32.02	130.54	89.71	23.41	114.83	54.61	31.09
e4-B	fitness	9022.10	15.37	8994.00	**9036.57**	12.97	9009.00	**9036.66**	15.08	9005.00	**9035.77**	15.52	8999.00	**9033.93**	10.08	9012.00
	runtime	471.94	236.23	56.97	419.17	180.16	23.00	500.69	196.84	138.75	**345.08**	214.80	29.14	471.90	247.24	66.84
e3-B	fitness	7780.73	5.48	7775.00	7790.03	14.93	7775.00	7782.48	7.10	7775.00	**7790.1**	14.54	7777.00	7788.62	13.88	7777.00
	runtime	303.06	148.31	59.21	318.11	132.84	34.59	348.04	153.33	47.12	285.79	154.44	55.34	266.41	146.87	48.56
e3-C	fitness	10316.93	17.65	10292.00	10312.63	16.79	10292.00	10312.86	16.35	10292.00	10313.03	16.12	10292.00	10315.93	19.17	10292.00
	runtime	266.04	150.95	31.04	284.87	160.58	61.70	318.31	160.26	97.93	302.28	155.37	72.59	322.08	140.21	84.69
e4-C	fitness	11591.30	25.36	11556.00	11586.33	26.73	11556.00	11588.31	21.34	11547.00	11593.67	26.73	11553.00	11598.86	33.90	11556.00
	runtime	418.14	152.82	165.67	492.07	160.67	70.81	387.01	182.91	91.93	425.09	155.51	154.21	406.69	166.01	126.52
s1-A	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	70.46	34.22	34.57	**90.04**	36.31	38.23	78.18	29.35	30.59	**85.28**	30.80	31.18	**91.05**	42.25	30.84
s1-B	fitness	6403.23	17.68	6388.00	6409.00	19.83	6388.00	6409.14	19.93	6388.00	6414.03	20.18	6394.00	6405.31	18.02	6394.00
	runtime	165.33	150.45	9.69	165.59	162.78	7.45	143.34	117.44	15.12	157.51	162.02	6.11	165.43	136.27	9.20
s1-C	fitness	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00
	runtime	55.21	26.67	23.76	**88.84**	43.74	30.86	68.40	52.49	28.90	76.62	49.32	28.16	**76.65**	31.22	19.66
s2-A	fitness	9942.53	26.83	9890.00	**9909.43**	17.41	9889.00	**9904.55**	17.84	9888.00	**9916.33**	24.48	9889.00	**9916.07**	22.77	9889.00
	runtime	1094.88	282.01	407.59	953.19	349.60	311.86	981.99	330.43	285.56	1042.72	310.61	326.83	**904.9**	309.18	427.34
s2-B	fitness	13157.07	34.06	13097.00	**13173.63**	23.49	13124.00	**13180.9**	28.76	13117.00	13166.03	34.46	13096.00	**13178.76**	30.38	13113.00
	runtime	1056.05	216.66	600.67	1076.62	255.78	479.94	1100.49	253.44	661.10	1062.43	291.78	462.05	1055.13	223.24	419.02
s2-C	fitness	16491.00	37.13	16442.00	16477.33	36.15	16430.00	16472.69	26.73	16442.00	**16467.03**	26.06	16430.00	**16470.31**	39.10	16430.00
	runtime	731.91	244.61	234.81	659.53	252.24	182.78	657.33	208.77	247.12	706.74	215.34	279.52	**590.72**	222.86	225.34
s3-A	fitness	10294.30	27.14	10240.00	10282.97	33.00	10221.00	**10276.76**	26.32	10221.00	10279.47	29.64	10221.00	**10272.59**	26.90	10221.00
	runtime	1063.47	298.49	444.06	1155.64	283.99	466.89	1058.35	295.61	471.31	986.02	380.61	175.03	993.53	364.94	263.06
s3-B	fitness	13806.37	50.26	13684.00	13792.83	70.81	13695.00	13801.59	59.24	13709.00	13811.33	60.63	13692.00	13788.14	53.87	13707.00
	runtime	912.45	247.37	398.00	951.35	210.81	399.30	955.15	183.10	481.95	961.29	249.93	409.38	990.78	204.65	482.64
s3-C	fitness	17285.97	34.40	17222.00	17273.63	31.21	17209.00	17270.24	28.74	17212.00	17286.70	40.70	17212.00	17279.69	26.54	17237.00
	runtime	768.58	152.55	375.37	755.12	186.23	380.75	749.33	239.50	143.02	756.62	250.62	181.79	733.04	184.68	319.19
s4-A	fitness	12398.90	43.64	12313.00	12381.73	43.45	12276.00	12384.55	39.31	12278.00	12383.20	39.04	12301.00	12379.62	47.88	12268.00
	runtime	1308.82	332.56	562.45	1335.77	322.96	682.15	1319.67	394.84	337.17	1208.02	408.74	402.62	1340.01	333.52	553.86
s4-B	fitness	16407.23	46.59	16287.00	16410.10	42.59	16321.00	16392.07	56.59	16300.00	16398.40	49.54	16243.00	16403.59	48.07	16281.00
	runtime	1135.00	199.26	665.93	1204.76	202.63	659.93	**1261.69**	186.11	860.75	**1247.34**	213.55	740.14	**1257.65**	210.06	694.50
s4-C	fitness	20886.93	270.29	20630.00	20825.93	212.86	20616.00	20822.66	197.78	20646.00	**20770.6**	138.43	20582.00	**20763.34**	111.32	20571.00
	runtime	1169.94	222.46	49.44	1082.55	246.16	237.56	1204.99	178.96	784.03	1107.19	219.87	433.96	1095.30	197.20	701.38

Table 5.5: Results of MAENS*-III with k_U set to 25 and k_L assuming the values 10,25,50,100 on the *egl* CARP instance set. The table compares both fitness and runtime on 30 independent runs. Average values with a boldfaced value indicate a results statistically significant with respect to the results of MAENS*, with significance level smaller than 0.05.

		MAENS*			25,10			25,25			25,50			25,100		
instance		avg	std	best	avg	std	best	avg	std	best	avg	std	best	avg	std	best
e1-A	fitness	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00
	runtime	7.23	4.35	1.48	8.75	6.13	1.49	**11.72**	8.22	0.89	7.67	6.33	0.96	8.88	5.72	1.03
e1-C	fitness	5595.00	0.00	5595.00	5595.60	3.23	5595.00	5595.00	0.00	5595.00	5595.00	0.00	5595.00	5595.97	3.64	5595.00
	runtime	133.79	59.63	34.92	151.78	60.51	76.73	136.58	67.22	69.19	126.80	54.39	45.90	117.97	55.35	57.10
e1-B	fitness	4505.10	10.58	4498.00	4508.67	11.98	4498.00	4507.20	11.13	4498.00	4506.10	12.04	4498.00	4505.52	10.60	4498.00
	runtime	123.67	75.51	2.77	147.50	91.22	1.99	144.27	86.14	7.19	137.83	81.69	5.00	155.28	64.51	5.19
e2-7	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	20.41	10.63	2.48	20.72	13.59	0.49	20.59	10.52	8.77	20.62	9.34	5.90	21.68	12.53	1.18
e3-A	fitness	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00
	runtime	65.17	32.55	22.55	66.93	34.29	26.36	**97.05**	66.61	26.84	92.12	60.18	25.66	80.22	39.55	28.69
e4-A	fitness	6463.80	3.62	6456.00	**6466.73**	6.22	6446.00	6464.87	3.97	6458.00	6463.00	6.47	6446.00	6464.10	5.70	6446.00
	runtime	232.20	164.13	43.01	**413.12**	177.65	49.60	304.83	172.08	86.74	**364.79**	184.59	49.05	**354.37**	178.20	73.77
e2-B	fitness	6321.63	7.78	6317.00	6319.57	4.68	6317.00	6319.60	4.79	6317.00	6320.43	6.40	6317.00	6319.45	4.67	6317.00
	runtime	195.95	105.84	43.71	178.36	107.42	23.37	150.17	100.22	18.36	172.98	131.76	9.98	170.77	95.25	21.12
e2-C	fitness	8335.40	1.20	8335.00	8335.27	1.00	8335.00	8335.13	0.72	8335.00	8335.27	1.00	8335.00	8335.14	0.72	8335.00
	runtime	143.19	66.34	49.48	139.09	113.69	16.75	158.96	67.31	62.91	129.96	111.62	21.64	156.11	113.61	20.58
e4-B	fitness	9022.10	15.37	8994.00	**9031.37**	11.63	9005.00	**9032.0**	12.37	9003.00	**9031.3**	12.32	9000.00	**9034.62**	13.26	9003.00
	runtime	471.94	236.23	56.97	458.19	260.54	39.95	537.41	232.96	55.02	501.05	256.26	40.38	454.07	233.71	77.79
e3-B	fitness	7780.73	5.48	7775.00	**7791.07**	15.75	7775.00	**7794.9**	16.45	7777.00	**7789.87**	14.09	7777.00	**7790.41**	14.72	7775.00
	runtime	303.06	148.31	59.21	336.29	148.67	112.40	329.84	169.17	32.22	294.43	157.49	50.41	278.72	131.41	77.76
e3-C	fitness	10316.93	17.65	10292.00	10314.67	18.16	10292.00	10315.13	17.86	10292.00	10315.60	16.88	10292.00	10314.34	17.22	10292.00
	runtime	266.04	150.95	31.04	292.28	151.36	31.05	281.08	152.36	74.85	227.79	132.55	52.27	329.69	141.19	37.33
e4-C	fitness	11591.30	25.36	11556.00	11594.77	24.88	11556.00	11589.27	21.76	11556.00	11602.47	26.28	11561.00	11588.97	26.27	11544.00
	runtime	418.14	152.82	165.67	370.13	183.55	80.95	427.60	186.19	126.04	373.23	199.55	51.49	411.25	186.42	91.47
s1-A	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	70.46	34.22	34.57	**107.23**	45.95	46.30	**117.74**	63.55	39.05	**98.37**	44.92	30.63	**93.01**	41.06	31.96
s1-B	fitness	6403.23	17.68	6388.00	6408.70	19.78	6388.00	6413.00	20.69	6388.00	**6415.67**	20.70	6388.00	6413.79	20.14	6394.00
	runtime	165.33	150.45	9.69	155.28	125.67	6.00	151.25	155.56	13.46	145.93	163.87	9.41	127.97	131.04	10.50
s1-C	fitness	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00
	runtime	55.21	26.67	23.76	**91.99**	52.41	41.91	**77.39**	28.69	30.09	**100.03**	47.54	31.52	**107.56**	73.00	37.70
s2-A	fitness	9942.53	26.83	9890.00	**9920.77**	21.88	9888.00	**9913.5**	20.75	9889.00	**9920.93**	24.16	9889.00	**9912.66**	19.31	9889.00
	runtime	1094.88	282.01	407.59	1048.83	342.81	260.29	1074.89	392.95	359.98	1082.48	336.65	338.76	1014.26	387.82	316.43
s2-B	fitness	13157.07	34.06	13097.00	**13185.4**	27.84	13117.00	13169.27	31.71	13121.00	13173.83	35.29	13118.00	**13173.38**	29.64	13128.00
	runtime	1056.05	216.66	600.67	1133.53	207.91	624.21	1045.59	272.01	428.01	1113.77	253.00	434.05	987.83	316.43	267.94
s2-C	fitness	16491.00	37.13	16442.00	16474.97	30.34	16430.00	**16464.33**	27.33	16430.00	**16466.97**	16.72	16442.00	16475.72	24.56	16442.00
	runtime	731.91	244.61	234.81	664.26	192.70	237.21	653.96	242.08	346.80	752.26	232.67	327.39	675.12	222.14	210.81
s3-A	fitness	10294.30	27.14	10240.00	10285.53	23.78	10227.00	**10277.17**	30.99	10221.00	10282.23	25.94	10221.00	**10274.21**	27.18	10221.00
	runtime	1063.47	298.49	444.06	1025.82	338.40	358.58	1077.43	368.82	216.67	1062.42	350.53	389.66	952.31	322.93	291.59
s3-B	fitness	13806.37	50.26	13684.00	13799.20	57.05	13695.00	13782.77	50.65	13697.00	13808.50	65.32	13694.00	13792.79	60.77	13706.00
	runtime	912.45	247.37	398.00	997.36	267.46	278.05	989.65	250.82	262.91	996.24	298.97	394.11	957.96	234.91	412.74
s3-C	fitness	17285.97	34.40	17222.00	17294.47	37.26	17219.00	17283.60	39.51	17202.00	17299.73	32.07	17226.00	17281.14	34.37	17206.00
	runtime	768.58	152.55	375.37	790.66	214.42	317.36	828.03	144.18	463.36	822.69	195.82	304.92	**853.35**	150.60	495.99
s4-A	fitness	12398.90	43.64	12313.00	12381.97	42.90	12280.00	12388.10	41.39	12290.00	12380.27	43.12	12289.00	12375.72	48.89	12263.00
	runtime	1308.82	332.56	562.45	1343.12	356.00	560.38	1330.63	401.62	579.68	1350.43	372.94	646.94	1304.62	344.92	523.28
s4-B	fitness	16407.23	46.59	16287.00	16405.10	40.79	16334.00	16407.57	43.83	16288.00	16415.33	41.59	16277.00	16401.69	38.95	16304.00
	runtime	1135.00	199.26	665.93	**1279.31**	213.53	846.73	**1324.23**	207.33	730.67	**1315.08**	267.80	573.07	**1295.05**	210.21	912.72
s4-C	fitness	20886.93	270.29	20630.00	20812.83	184.02	20572.00	**20772.37**	121.88	20560.00	20820.37	193.15	20616.00	**20767.1**	98.99	20602.00
	runtime	1169.94	222.46	49.44	1196.83	147.08	668.36	1219.71	192.76	815.92	1158.75	212.29	547.30	1191.77	195.98	466.49

Table 5.6: Results of MAENS*-III with k_U set to 50 and k_L assuming the values 10,25,50,100 on the *egl* CARP instance set. The table compares both fitness and runtime on 30 independent runs. Average values with a boldfaced value indicate a results statistically significant with respect to the results of MAENS*, with significance level smaller than 0.05.

		MAENS*			50,10			50,25			50,50			50,100		
instance		avg	std	best	avg	std	best	avg	std	best	avg	std	best	avg	std	best
e1-A	fitness	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00
	runtime	7.23	4.35	1.48	8.30	6.25	0.76	9.99	6.30	1.17	7.32	5.86	0.85	9.92	6.98	1.22
e1-C	fitness	5595.00	0.00	5595.00	5595.60	3.23	5595.00	5595.60	3.23	5595.00	5596.00	3.00	5595.00	5595.93	3.64	5595.00
	runtime	133.79	59.63	34.92	113.11	53.40	24.26	123.96	54.99	42.77	**190.75**	96.01	33.14	150.10	64.24	61.84
e1-B	fitness	4505.10	10.58	4498.00	4508.80	12.70	4498.00	4504.43	9.79	4498.00	4505.97	10.99	4498.00	4507.57	12.25	4498.00
	runtime	123.67	75.51	2.77	113.40	67.00	8.98	**163.86**	77.61	9.78	142.40	79.66	11.14	149.02	77.69	4.25
e2-7	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	20.41	10.63	2.48	20.87	10.59	9.81	25.73	13.79	4.75	20.67	11.42	6.23	25.01	10.38	8.75
e3-A	fitness	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00
	runtime	65.17	32.55	22.55	59.85	20.56	21.36	76.53	32.73	30.97	94.14	63.04	7.22	**91.98**	44.10	29.29
e4-A	fitness	6463.80	3.62	6456.00	6462.13	4.90	6446.00	6465.17	5.34	6456.00	6465.67	5.08	6460.00	6466.60	6.29	6456.00
	runtime	232.20	164.13	43.01	**398.98**	167.35	96.52	**327.66**	186.44	55.45	**417.73**	187.62	66.66	**343.62**	199.62	40.60
e2-B	fitness	6321.63	7.78	6317.00	6319.20	3.95	6317.00	6318.57	3.89	6317.00	6318.43	3.77	6317.00	6318.50	3.91	6317.00
	runtime	195.95	105.84	43.71	211.62	107.29	31.73	176.82	103.21	8.32	211.94	101.26	23.54	181.73	107.45	33.33
e2-C	fitness	8335.40	1.20	8335.00	8335.53	1.71	8335.00	8335.13	0.72	8335.00	8335.40	1.20	8335.00	8335.67	1.49	8335.00
	runtime	143.19	66.34	49.48	145.81	104.82	44.17	163.49	108.43	37.90	165.31	117.34	16.37	148.47	102.58	31.72
e4-B	fitness	9022.10	15.37	8994.00	**9033.53**	10.64	9000.00	**9034.23**	14.27	9000.00	**9032.93**	14.03	9000.00	**9033.2**	10.42	9011.00
	runtime	471.94	236.23	56.97	528.53	249.82	51.06	485.93	213.89	74.73	440.81	216.38	91.54	553.80	227.12	34.10
e3-B	fitness	7780.73	5.48	7775.00	7784.80	12.00	7775.00	7789.40	14.15	7775.00	**7797.6**	15.86	7777.00	7788.33	14.00	7775.00
	runtime	303.06	148.31	59.21	374.08	148.17	91.88	353.86	164.61	38.28	306.50	152.57	65.88	356.74	153.05	62.83
e3-C	fitness	10316.93	17.65	10292.00	10313.80	16.75	10292.00	10313.60	15.63	10292.00	10315.80	17.84	10292.00	10316.00	16.27	10292.00
	runtime	266.04	150.95	31.04	292.52	161.33	112.79	**349.37**	142.26	96.50	**348.78**	163.59	52.66	308.20	117.55	110.41
e4-C	fitness	11591.30	25.36	11556.00	11595.27	32.44	11538.00	11601.50	24.58	11566.00	11599.13	35.24	11544.00	11589.47	26.00	11541.00
	runtime	418.14	152.82	165.67	436.82	162.62	140.61	453.53	173.59	164.50	419.62	197.46	106.99	485.43	153.96	140.61
s1-A	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	70.46	34.22	34.57	77.03	35.75	16.18	**118.41**	52.27	41.63	**113.79**	40.74	63.50	**109.79**	38.53	46.23
s1-B	fitness	6403.23	17.68	6388.00	6406.30	18.79	6394.00	6410.93	19.71	6394.00	6411.33	20.29	6388.00	6412.13	20.16	6388.00
	runtime	165.33	150.45	9.69	163.11	151.40	8.11	189.40	153.96	9.74	183.43	160.73	8.22	151.34	140.30	9.70
s1-C	fitness	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00
	runtime	55.21	26.67	23.76	64.79	29.06	28.64	**107.46**	48.10	43.64	**110.66**	57.99	35.64	**86.81**	43.13	35.90
s2-A	fitness	9942.53	26.83	9890.00	**9916.67**	22.82	9889.00	**9922.7**	23.54	9889.00	**9908.83**	15.41	9889.00	**9924.8**	23.24	9889.00
	runtime	1094.88	282.01	407.59	1018.51	412.65	169.78	1124.62	339.66	387.66	1089.05	366.86	412.08	1173.84	326.02	475.65
s2-B	fitness	13157.07	34.06	13097.00	13176.90	36.28	13124.00	13173.47	30.31	13125.00	**13181.9**	33.60	13124.00	13173.90	33.03	13121.00
	runtime	1056.05	216.66	600.67	1085.55	237.35	462.54	1077.12	281.75	529.92	1129.53	223.88	569.77	1079.73	286.39	465.57
s2-C	fitness	16491.00	37.13	16442.00	**16463.43**	26.44	16430.00	**16472.43**	29.55	16430.00	16479.77	28.63	16442.00	16480.23	31.64	16442.00
	runtime	731.91	244.61	234.81	732.39	212.38	198.36	735.86	227.01	182.63	635.91	219.11	127.18	744.59	220.10	233.06
s3-A	fitness	10294.30	27.14	10240.00	**10266.5**	29.48	10221.00	**10275.47**	26.77	10221.00	**10272.97**	26.92	10227.00	**10279.37**	27.02	10221.00
	runtime	1063.47	298.49	444.06	1040.15	321.11	322.18	1137.39	343.12	518.67	1110.78	301.57	330.77	1115.64	340.69	431.93
s3-B	fitness	13806.37	50.26	13684.00	13795.43	47.46	13710.00	13790.93	64.08	13698.00	13812.03	76.55	13708.00	13801.93	59.49	13703.00
	runtime	912.45	247.37	398.00	917.44	231.54	300.70	**1114.69**	214.71	654.06	1040.77	218.94	466.80	**1003.4**	278.43	309.73
s3-C	fitness	17285.97	34.40	17222.00	17283.80	35.98	17201.00	17280.27	26.98	17248.00	17288.63	43.60	17208.00	17282.57	37.95	17210.00
	runtime	768.58	152.55	375.37	812.97	167.16	263.81	787.59	205.46	313.57	790.15	205.73	333.26	823.49	217.50	277.79
s4-A	fitness	12398.90	43.64	12313.00	12384.97	33.60	12308.00	12387.10	39.25	12303.00	12380.50	38.99	12301.00	12404.00	28.01	12325.00
	runtime	1308.82	332.56	562.45	1229.01	342.10	467.21	**1555.66**	340.74	702.25	**1510.49**	348.35	490.86	1461.89	377.15	556.53
s4-B	fitness	16407.23	46.59	16287.00	16428.07	27.01	16383.00	16414.33	45.80	16308.00	16420.93	46.28	16298.00	16412.17	45.61	16280.00
	runtime	1135.00	199.26	665.93	1214.96	215.64	607.28	**1374.98**	260.63	706.25	**1453.65**	251.90	429.33	**1382.24**	212.97	952.22
s4-C	fitness	20886.93	270.29	20630.00	**20767.9**	91.30	20599.00	20803.07	148.56	20548.00	20826.33	115.21	20613.00	**20779.77**	149.86	20604.00
	runtime	1169.94	222.46	49.44	1134.50	157.93	784.82	1243.69	154.27	718.92	1179.52	198.67	717.64	**1220.09**	247.26	255.33

Table 5.7: Results of MAENS*-III with k_U set to 100 and k_L assuming the values 10,25,50,100 on the *egl* CARP instance set. The table compares both fitness and runtime on 30 independent runs. Average values with a boldfaced value indicate a results statistically significant with respect to the results of MAENS*, with significance level smaller than 0.05.

		MAENS*			100,10			100,25			100,50			100,100		
instance		avg	std	best	avg	std	best	avg	std	best	avg	std	best	avg	std	best
e1-A	fitness	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00	3548.00	0.00	3548.00
	runtime	7.23	4.35	1.48	8.87	7.86	1.29	**10.28**	5.56	1.09	10.08	5.62	1.40	10.54	7.15	1.72
e1-C	fitness	5595.00	0.00	5595.00	5595.93	3.64	5595.00	5595.60	3.23	5595.00	5595.60	3.23	5595.00	5595.62	3.23	5595.00
	runtime	133.79	59.63	34.92	146.89	70.12	35.21	147.69	60.22	38.98	156.90	79.98	3.10	171.99	91.01	53.86
e1-B	fitness	4505.10	10.58	4498.00	4509.03	12.21	4498.00	4505.33	11.47	4498.00	4511.13	12.30	4498.00	**4512.0**	13.04	4498.00
	runtime	123.67	75.51	2.77	**167.92**	89.39	3.95	147.21	69.65	19.79	140.33	80.78	4.40	113.50	78.57	0.42
e2-7	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	20.41	10.63	2.48	22.80	12.44	3.95	**26.93**	10.73	13.10	22.75	11.02	3.98	**30.66**	15.21	5.26
e3-A	fitness	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00	5898.00	0.00	5898.00
	runtime	65.17	32.55	22.55	78.24	38.13	20.89	**123.53**	107.70	21.80	74.18	37.82	24.85	89.21	68.78	33.47
e4-A	fitness	6463.80	3.62	6456.00	**6467.63**	7.91	6446.00	**6468.87**	6.09	6461.00	**6468.77**	7.73	6446.00	6464.03	6.92	6444.00
	runtime	232.20	164.13	43.01	**350.72**	188.01	36.08	**373.93**	200.79	78.78	284.28	214.07	32.22	**418.28**	176.24	99.85
e2-B	fitness	6321.63	7.78	6317.00	6319.13	3.97	6317.00	6318.10	3.14	6317.00	6319.23	4.82	6317.00	6320.79	5.37	6317.00
	runtime	195.95	105.84	43.71	193.16	121.89	35.50	195.58	83.30	39.14	154.14	113.96	13.50	213.61	115.89	16.45
e2-C	fitness	8335.40	1.20	8335.00	8335.40	1.20	8335.00	8335.67	1.49	8335.00	8335.40	1.20	8335.00	8335.69	1.49	8335.00
	runtime	143.19	66.34	49.48	140.47	98.10	34.23	168.50	92.09	28.08	140.36	101.32	29.22	215.11	131.32	32.98
e4-B	fitness	9022.10	15.37	8994.00	**9030.57**	12.17	9009.00	**9030.07**	13.04	9005.00	**9030.67**	10.80	9009.00	**9034.55**	10.46	9009.00
	runtime	471.94	236.23	56.97	564.89	241.77	72.54	446.06	231.03	54.45	495.33	198.24	152.38	515.54	205.79	146.10
e3-B	fitness	7780.73	5.48	7775.00	**7798.5**	16.47	7777.00	**7790.67**	15.64	7777.00	**7794.77**	14.93	7777.00	**7799.31**	14.64	7777.00
	runtime	303.06	148.31	59.21	303.41	167.47	43.36	344.45	161.20	43.16	**438.09**	153.83	99.77	**388.14**	164.30	81.11
e3-C	fitness	10316.93	17.65	10292.00	10318.90	17.13	10292.00	10310.27	14.63	10292.00	10310.60	15.19	10292.00	10315.93	14.98	10292.00
	runtime	266.04	150.95	31.04	290.13	155.69	36.31	325.90	153.23	102.31	**379.7**	146.11	164.66	330.53	142.33	102.60
e4-C	fitness	11591.30	25.36	11556.00	**11613.8**	41.48	11560.00	11594.37	27.53	11554.00	11590.97	28.80	11556.00	11607.10	48.70	11560.00
	runtime	418.14	152.82	165.67	495.25	186.44	55.15	486.97	152.38	157.84	488.02	174.17	187.40	476.61	163.03	123.59
s1-A	fitness	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00	5018.00	0.00	5018.00
	runtime	70.46	34.22	34.57	**124.19**	40.92	60.32	**129.12**	53.45	53.24	**137.84**	50.33	57.90	**142.13**	41.95	58.61
s1-B	fitness	6403.23	17.68	6388.00	6413.27	20.12	6394.00	6412.87	20.19	6394.00	6409.03	19.76	6394.00	6412.38	20.05	6394.00
	runtime	165.33	150.45	9.69	188.08	178.68	5.79	199.35	171.89	9.47	188.00	156.45	8.58	156.84	150.23	0.83
s1-C	fitness	8518.00	0.00	8518.00	8518.00	0.00	8518.00	8518.57	2.87	8518.00	8518.00	0.00	8518.00	8518.00	0.00	8518.00
	runtime	55.21	26.67	23.76	**100.76**	48.05	30.74	**140.57**	79.24	48.10	**129.2**	64.48	51.52	**133.38**	75.07	41.76
s2-A	fitness	9942.53	26.83	9890.00	**9912.37**	19.30	9889.00	**9915.03**	19.26	9889.00	**9922.57**	24.11	9889.00	**9913.45**	23.82	9889.00
	runtime	1094.88	282.01	407.59	1102.73	311.91	580.81	1183.66	395.26	419.45	**1242.7**	334.92	394.63	964.71	296.97	330.22
s2-B	fitness	13157.07	34.06	13097.00	**13183.03**	26.67	13113.00	**13194.47**	42.77	13121.00	**13190.7**	42.47	13124.00	**13182.59**	31.23	13121.00
	runtime	1056.05	216.66	600.67	1128.93	241.48	621.62	1147.87	222.53	683.08	1168.18	228.74	568.15	1143.57	226.86	687.08
s2-C	fitness	16491.00	37.13	16442.00	16480.33	24.07	16438.00	16475.80	23.04	16442.00	16476.73	30.18	16430.00	**16469.52**	21.14	16442.00
	runtime	731.91	244.61	234.81	749.54	222.18	302.41	765.40	269.05	262.12	802.64	266.88	204.80	762.85	205.90	377.22
s3-A	fitness	10294.30	27.14	10240.00	**10276.83**	22.76	10233.00	10285.37	26.59	10222.00	**10280.83**	23.82	10221.00	**10279.28**	25.09	10227.00
	runtime	1063.47	298.49	444.06	1160.30	328.58	342.41	1162.54	401.96	377.31	1176.59	341.45	451.09	1199.22	329.44	269.59
s3-B	fitness	13806.37	50.26	13684.00	13821.17	74.99	13715.00	**13782.87**	58.81	13704.00	13818.30	67.41	13708.00	13793.52	60.28	13696.00
	runtime	912.45	247.37	398.00	**1181.78**	182.56	495.00	**1148.71**	227.22	532.01	**1135.24**	249.86	583.22	**1067.08**	183.73	654.99
s3-C	fitness	17285.97	34.40	17222.00	17287.33	39.92	17208.00	17298.80	31.92	17249.00	17281.10	40.64	17207.00	17280.00	35.37	17206.00
	runtime	768.58	152.55	375.37	**844.97**	226.19	253.67	**899.53**	250.29	376.43	**856.97**	190.67	488.51	**871.39**	161.25	547.27
s4-A	fitness	12398.90	43.64	12313.00	12387.83	38.67	12299.00	12388.33	32.70	12322.00	12399.63	32.47	12319.00	12393.07	29.47	12316.00
	runtime	1308.82	332.56	562.45	**1633.03**	382.29	515.17	**1609.53**	346.64	825.64	**1513.42**	427.66	436.80	1446.54	428.34	616.82
s4-B	fitness	16407.23	46.59	16287.00	16429.90	31.94	16362.00	16429.67	33.75	16336.00	16416.97	38.52	16319.00	16419.79	36.53	16326.00
	runtime	1135.00	199.26	665.93	**1463.68**	224.51	1036.94	**1549.5**	239.99	1076.70	**1537.37**	229.76	1093.31	**1501.16**	235.86	863.49
s4-C	fitness	20886.93	270.29	20630.00	20855.30	180.12	20579.00	**20764.0**	117.83	20562.00	20804.07	139.73	20548.00	20796.55	153.64	20568.00
	runtime	1169.94	222.46	49.44	**1288.85**	155.12	952.89	**1319.41**	202.39	505.45	**1270.76**	164.84	929.97	**1252.21**	244.19	587.07

Table 5.8: Summary of the results of tables 5.4, 5.5, 5.6 and 5.7. The table includes the number of instances where each combination of the parameters k_U and k_L obtained better results (column +) or worst results (column -) according to the set of statistical tests described in section A.1, in terms of fitness (left side) and runtime (right side)

		k_L															
		fitness								runtime							
		10		25		50		100		10		25		50		100	
		+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
k_U	10	1	2	3	2	4	2	2	2	1	3	1	3	2	3	1	2
	25	1	4	2	3	3	3	4	2	0	4	0	4	0	5	0	4
	50	4	1	2	3	3	1	3	1	0	1	0	7	0	7	0	8
	100	2	5	2	4	3	4	3	4	0	9	0	10	0	9	0	10

Table 5.9: Comparison of the results achieved by MAENS* and by MAENS*-III with simulated AM having perfect classification. The two algorithms are compared in terms of average fitness of the best solution found over 30 independent runs and runtime. In both cases average, standard deviation and best value are provided. Statistically significant results are highlighted in bold

instance	MAENS*						MAENS*-III (No Misclassification)					
	fitness			runtime			fitness			runtime		
	avg	std	best	avg	std	best	avg	std	best	avg	std	best
e1-a	3548.00	0.00	3548.00	7231.77	4423.69	1483.00	3548.00	0.00	3548.00	**808.53**	886.89	30.00
e1-b	5595.00	0.00	5595.00	133788.57	60651.75	34918.00	5595.80	3.09	5595.00	**57625.20**	58097.28	8216.00
e1-c	4505.10	10.76	4498.00	123672.90	76801.77	2766.00	4502.00	9.38	4498.00	**83338.97**	74502.39	529.00
e2-a	5018.00	0.00	5018.00	20414.13	10809.28	2482.00	5018.00	0.00	5018.00	**14680.37**	11787.44	768.00
e2-b	5898.00	0.00	5898.00	65166.00	33103.44	22546.00	5898.00	0.00	5898.00	48903.77	44781.70	3413.00
e2-c	6463.80	3.68	6456.00	232204.27	166931.87	43012.00	6463.83	2.48	6461.00	177462.47	222430.90	2772.00
e3-a	6321.63	7.91	6317.00	195953.63	107649.50	43713.00	6319.50	6.39	6317.00	**185858.73**	135362.42	3198.00
e3-b	8335.40	1.22	8335.00	143191.80	67474.56	49482.00	8335.13	0.73	8335.00	**144469.40**	119525.21	3172.00
e3-c	9022.10	15.64	8994.00	471935.53	240267.82	56974.00	9027.67	13.89	9001.00	**534474.03**	223145.80	144558.00
e4-a	7780.73	5.58	7775.00	303059.00	150843.52	59213.00	7782.00	9.00	7775.00	**210735.23**	180559.12	2005.00
e4-b	10316.93	17.95	10292.00	266037.87	153529.67	31041.00	10311.73	19.19	10292.00	188377.93	173602.76	1648.00
e4-c	11591.30	25.80	11556.00	418141.50	155428.35	165674.00	11590.90	21.68	11541.00	**213335.90**	202294.85	1990.00
s1-a	5018.00	0.00	5018.00	70464.90	34805.40	34571.00	5018.00	0.00	5018.00	**55968.83**	43869.66	7050.00
s1-b	6403.23	17.99	6388.00	165325.00	153023.57	9688.00	6406.57	19.63	6388.00	**87361.50**	118200.72	244.00
s1-c	8518.00	0.00	8518.00	55213.60	27125.22	23761.00	8518.00	0.00	8518.00	**22509.60**	30638.82	250.00
s2-a	9942.53	27.29	9890.00	1094883.30	286832.21	407587.00	**9921.30**	22.21	9889.00	**699952.77**	453189.15	5992.00
s2-b	13157.07	34.65	13097.00	1056050.83	220359.33	600672.00	13156.03	30.58	13101.00	**561677.33**	419212.53	10380.00
s2-c	16491.00	37.76	16442.00	731908.17	248787.17	234810.00	**16465.67**	40.67	16426.00	**447026.43**	299605.76	14579.00
s3-a	10294.30	27.60	10240.00	1063470.10	303593.81	444059.00	**10275.03**	27.59	10221.00	**855461.00**	394155.63	55754.00
s3-b	13806.37	51.12	13684.00	912449.07	251597.46	398001.00	13796.30	45.81	13695.00	**580283.03**	385181.78	41625.00
s3-c	17285.97	34.99	17222.00	768581.23	155162.73	375371.00	17277.07	41.41	17189.00	**559173.30**	276487.02	19540.00
s4-a	12398.90	44.39	12313.00	1308822.07	338243.93	562450.00	**12369.10**	41.71	12308.00	**1015580.93**	542892.46	55008.00
s4-b	16407.23	47.39	16287.00	1135001.30	202662.51	665934.00	**16379.10**	54.14	16262.00	1019469.80	307327.25	350021.00
s4-c	20886.93	274.91	20630.00	1169938.30	226262.57	49436.00	**20778.03**	130.28	20577.00	**751836.17**	249579.83	255788.00

CHAPTER 6

Conclusion and Future Work

The problem with self-improvement is knowing when to quit

David Lee Roth

The objective of this thesis is to propose an adaptive online learning strategy to improve the approximation ability of state-of-the-art meta-heuristics for the Capacitated Arc Routing Problem (CARP). The main idea that motivates this thesis is to provide the algorithm with several and alternative sources of information, along mechanisms capable of learning such information and to operate choices based on it during the search. The thesis explores two different research areas where this approach could be followed. The former is the use of online Adaptive Operator Selection techniques to choose from a suite of crossover operators. The latter is using Knowledge Incorporation to improve the performance of the algorithms reusing the information obtained during the execution of the algorithm on past instances. The proposed approaches show that the use of online search techniques combined with the introduction of novel and different sources of information, endows the algorithms with an increased adaptivity and, ultimately, leads to an increased robustness of its results, which can be translated into a better approximation ability.

This final chapter summarizes and details the contributions of this thesis and provides directions that can be explored for future work. The contributions correspond to the answers to the research questions that have been detailed in section 1.2.

6.1 Diversity-Driven Selection of Multiple Crossover Operators

The first contribution of this thesis, included in chapter 3, aims to answer to research questions **RQ1** and **RQ2**. This is done by developing an improved version of the MAENS [128] algorithm for the CARP, called MAENS*. The main characteristics of this algorithm are:

1. it makes use of a new diversity metric between CARP solutions. A formal definition of the metric is provided;

2. a diversity-driven stochastic ranking operator, that replaces the original stochastic ranking operator, and that is used to balance the levels of exploration within the population through the measurement of the average pairwise diversity in the population;

3. the use of a state-of-the-art AOS strategy, called dynamic Multi Armed Bandit (dMAB), to select the most suitable crossover operator;

4. the definition of four novel crossover operators for CARP;

5. a novel Credit Assignment Strategy, named Proportional Reward, using the aforementioned ranking operator to define the performance of an operator based on the survival ability of the offspring.

Experimental results performed on a set of 181 CARP instances, belonging to known benchmark sets, show how MAENS* is capable of outperforming the original algorithm MAENS.

The first and foremost answer to the research question is given by the experimental results achieved by the algorithm, providing evidence of the increased performances thanks to the use of the AOS technique and the diversity measure. This work is also relevant because it provides the motivations for research questions **RQ3** and **RQ4**, and the foundations of the further experimentations described in chapter 4.

Moreover, it supports the idea of performing AOS studies not only on mutation operators, which draw most of the attention of this research field, but also on crossover operators. Finally, it provides some elements that are domain independent and that can easily be adopted within meta-heuristics designed for different NP-Hard problems, such as the Proportional Reward and the Diversity-Driven Stochastic Ranking.

6.1.1 Future Work

The work carried out so far leaves space for several possible improvements. In particular, a thorough study should be performed to identify optimal values of the parameters, which have been set through a process of test and trial, due to the considerable computational cost of performing a parameter optimization step, particularly large when algorithms with a high number of parameters are considered, as it is the case of the ones examined in this work. A second possible direction to follow is that of extending the AOS approach to different algorithmic features of MAENS, such as alternative selection mechanisms, parent selection for crossover, the selection of the routes or of the move operator within the local search. Finally, it would be interesting to conduct a deeper analysis to investigate the performances of the Proportional Reward when compared to other reward measures.

6.2 Dynamic Selection of Evolutionary Operators Based on Online Learning and Fitness Landscape Analysis

The work described in chapter 4 proposes a novel Adaptive Operator Selection mechanism which uses a set of four Fitness Landscape Analysis techniques and an Online Learning algorithm, to provide more detailed information about the current population and the search space in order to better determine the most suitable crossover operator. This work provides an answer to research questions **RQ3** and **RQ4**.

Two different Reward Measure strategies are considered, the Diversity Based Reward (DBR) and the Proportional Reward (PR), as well as two different Operator Selection Rules, namely the Instantaneous Reward (IR) and a Concurrent Approach (CA). The AOS proposed combines a set of four Fitness Landscape Analysis measures in conjunction with an Online Learning algorithm, the Dynamic Weighted Majority (DWM), to predict the most suitable crossover operator. The four FLA metrics used as inputs of our predictive model are: accumulated escape probability, dispersion metric, average neutrality ratio and average delta fitness of neutral networks. These metrics have been chosen because (1) they can be computed without much increasing the computational effort and (2) they complement each other by capturing different features of the landscapes.

Three versions of the MAENS* algorithm were implemented and tested on two different datasets of CARP instances. The results of such experiments were compared against those by state-of-the art algorithms, and against the results obtained by an oracle. Experimental analysis shows that different crossover operators behave differently during the search process, and that selecting the proper one adaptively can lead to more promising results.

The results achieved by MAENS*-II show also that this technique is able to compete with the state-of-the-art techniques and can, in some cases, exploit the multiple measures to outperform the state-of-the-art. In the dataset containing large CARP instances, MAENS*-II was able to outperform all the existing approaches in terms of average and best solution quality in half of the instances, and even discovered new lower bounds.

The experiments seem to suggest a better performance of the Concurrent Strategy over the Instantaneous Reward, and a comparable performance of the two Reward Measure Strategies.

6.2.1 Future Work

This work leaves space for interesting directions that can be explored. In particular:

- As has been noted in Chapter 4, the ability of the algorithm to detect changes in the environment is not optimal. A proper parameter optimization would be required to detect the best setting for AOS system, in order to improve its detection rate and react promptly when in presence of concept drift;

- Experimental results have shown a comparable performance when DBR and PR are being used. It might be possible and reasonable to claim that the two measure address two types of information that are equally important. For this reason, the two Reward Measures might be combined to generate a novel measure that is able to predict better both the diversity and the survival ability of the offspring;

- It would be interesting to test the behaviour of MAENS*-II when adopting an Average or Extreme Reward strategy and the use of different base learners;

- Another promising direction is that of reducing the computational cost of MAENS*-II. As mentioned earlier in chapter 4, the algorithm spends most of its time generating the neighbourhood in the local search phase. Adaptive techniques might be designed with the purpose of reducing the size of the neighbourhood generated during this step;

- Due to the improved optimization ability provided by this approach, it would be interesting to test the use of MAENS*-II as the Single Objective routine for existing decomposition based approaches;
- Finally, the use of FLA and Online learning to perform AOS can be easily adopted to be embedded in different EAs and to tackle different combinatorial optimization problems, by replacing the FLA techniques shown in this thesis with the ones that best explain the new landscape.

6.3 Knowledge Incorporation Through Task Affinity Prediction

The contributions included in chapter 4, providing an answer to the research questions **RQ3** and **RQ4**, are implemented in the MAENS*-III algorithm. They are:

1. A study over the structure of CARP instances to identify existing and novel sources of information. The chapter provides a set of 34 features that capture various aspects of the instance with different levels of granularity (instance or Network level, task level, pair of tasks level);
2. The chapter includes the study and formal definition of the concept of task affinity, which is used to capture the salient traits out of an evolved population of solutions throughout the search, through a reinforcement learning mechanism. Experimental analysis conducted with the creation of an Oracle, using an artificial predicted affinity matrix reveals how this information, if predicted with a certain accuracy, can improve the performances of the algorithm, particularly regarding its execution time;

3. The introduction of a novel Local Search operator, designed to maximize the average task affinity of a solution. The experiments conducted with the oracle have shown that despite the extra computational time required to run this operator, if the information contained in the predicted affinity matrix is accurate enough the local search is an effective way to inject the information within the solutions of the population.

4. Since the use of techniques of Knowledge Incorporation translates into a higher risk of premature convergence, as shown in chapter 2, the chapter describes a self-adaptive technique to control the influence of the predicted affinity matrix and the impact of the affinity based local search during the search. Experimental analysis through parameter optimization reveals how balancing the dosage of these two techniques is fundamental and leads to versions of the algorithm with completely different results;

5. A learning system that extracts information from past instances and from instance features in order to be able to predict the task affinity of new problem instances. Experimental analysis shows that in order to achieve good results, it is not necessary to predict correctly the LOW affinity between pairs of tasks. The results also indicate that the algorithm is capable of predicting the affinity of the tasks, on average, in the 90% of the cases when there is no affinity, and in the 80% when there is high affinity. Statistical analysis reveals how the algorithm was successful into improving the results of MAENS* with statistical significance in a subset of the benchmark instance set, and to achieve comparable results in the remaining ones.

6.3.1 Future Work

The body of work included in chapter 4 is considerably large and some of the different experiments and research direction conducted during, were, in fact, left out of this thesis. The learning system is in fact made up of several components, which act and interact among each other and investigating their behaviour, either singularly or combined, revealed to be not a trivial task. It is reasonable to expect that some of the choices that have been made might, in fact, have been done in a different and perhaps better way. This translates into a great amount of alternative directions that could be followed to both improve the current system and to extend its functionalities. Some of these are:

- An alternative research direction, not included in the current thesis, has been conducted on designing Local Search algorithm adopting a score function that considers both fitness, violation and average affinity in order to choose the best individual in the neighbourhood. Experimental analysis would be required to provide a comparison with MAENS*-III existing approach. In particular it would be interest to analyze the survival rates of the offspring generated by the local search operators, their computational cost, and ultimately the optimization ability and runtime of the two systems;

- The information stored in the affinity matrix might be used to design another AOS strategy for the crossover operators or to replace the heuristic that is currently adopted to choose the routes that are being merged, taking into account the average affinity of the tasks included in such routes, or ultimately to propose a novel crossover operator;

- The average affinity of solutions might as well be used to influence the parent selection operators of the algorithm, assigning an higher selection probability to those individuals with higher average affinity, or to select the offspring based on the same value;

- A further idea is that of using the information contained in the affinity matrix to reduce the size of the search space, by cutting out areas that are known to decrease the affinity of solutions. This would be useful particularly when the neighbourhood of the individuals is generated during the local search phase, and could lead to a considerable reduction of the runtime of the algorithm;

- Replacing the stochastic ranking operator with a more complex online learning algorithm that assigns a rank to the individuals of the population, based on the evaluation of a set of factors, including fitness, amount of violation of the constraints, average pairwise diversity, fitness landscape analysis metrics as well as average affinity of the solution;

- The combination of the 14 static instance features and the 20 dynamic features described in this chapter could be used to perform a classical FLA analysis study on the CARP scenario to identify if any of such features is particularly important to determine the hardness of a problem instance;

- A different way to intend affinity between tasks is that of restricting its definition to represent the relationship of strict consecutiveness between tasks. Extending the concept of affinity to another Combinatorial Optimization algorithm would be a trivial task, as depending on their representation, they would fall easily in either of the two categories (clustering based problems or sequence based problems);

- It is possible to notice how the affinity matrix is an alternative, compact, fast and much more expressive way to represent a population of solutions. Experiments might be conducted to design a meta-heuristic that make use of such an alternative representation as well as novel algorithmic operators that perform mutation, crossover and local search in this scenario;

- A further direction, that has been partially explored during the course of this research, but that has not been included in this dissertation, is relative to the combined use of the technique discussed in this chapter with a restart strategy. The restart strategy can be used to detect moments of stagnation during the search, and to operate a partial reinitialization of the algorithm which can adopt the AFLS to regain quickly the solution quality achieved in the earlier stages of the search. This technique could be integrated with techniques to prevent the not-so-accidental discovery of already known local minima, through the use of taboo lists of already explored local optima;

- A further research direction is that of exploring the possibility of using the information contained in the affinity matrix to perform a intra-domain transfer learning between different combinatorial problems that share the same solution representation and affinity level (clustering or consecutiveness).

APPENDIX A

Statistical Analysis

The set of statistical tests that have been performed in this thesis are described in this appendix.

A.1 Statistical Analysis

The statistical analysis conducted in this thesis has been carried out using the following procedure:

- All the averages and standard deviation values reported in the experiments have been computed using the results of 30 independent executions of each algorithm;
- To test whether the results of two algorithms on the same problem instance are statistically different, a Wilcoxon Rank Sum Test[136] (or Mann-Whitney U test) was performed. The null hypothesis is rejected if the p-value is smaller than the significance level of 0.05;
- To test whether the results of two algorithms across multiple problem instances are statistically different, a Wilcoxon Signed Rank Test[136] was performed. The null hypothesis was rejected if the p-value is smaller than the significance level of 0.05;

- When the results of multiple algorithms are compared across a dataset of problem instances, a Holm-Bonferroni[59] correction was performed, unless otherwise specified.

List of References

[1] Jason Amunrud and Bryant A Julstrom. Instance similarity and the effectiveness of case injection in a genetic algorithm for binary quadratic programming. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1395–1396. ACM, 2006. One citation in section 2.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002. 3 citations in sections 2.3, 3.4.2, and 4.4.

[3] Roberto Baldacci and Vittorio Maniezzo. Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47(1):52–60, 2006. One citation in section 2.2.

[4] Helio JC Barbosa and AM Sá. On adaptive operator probabilities in real coded genetic algorithms. In *Workshop on Advances and Trends in Artificial Intelligence for Problem Solving*, 2000. One citation in section 2.3.

[5] Enrico Bartolini, Jean-François Cordeau, and Gilbert Laporte. Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, 137(1-2):409–452, 2013. One citation in section 2.2.

[6] José M Belenguer and Enrique Benavent. A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705–728, 2003. One citation in section 2.2.

[7] José-Manuel Belenguer, Enrique Benavent, Philippe Lacomme, and Christian Prins. Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, 33(12):3363–3383, 2006. One citation in section 2.2.

[8] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. The capacitated arc routing problem: lower bounds. *Networks*, 22(7):669–690, 1992. 3 citations in sections 2.2, 3.5, and 4.6.

[9] Patrick Beullens, Luc Muyldermans, Dirk Cattrysse, and Dirk Van Oudheusden. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3):629–643, 2003. 4 citations in sections 2.2, 2.2.2, 3.5, and 4.6.

[10] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18. Morgan Kaufmann Publishers Inc., 2002. 2 citations in sections 1.1.2 and 2.3.

[11] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011. One citation in section 2.4.1.

[12] José Brandão and Richard Eglese. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112–1126, 2008. 2 citations in sections 2.2.2 and 4.6.

[13] Jürgen Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 1875–1882. IEEE, 1999. One citation in section 2.4.2.

[14] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. One citation in section 5.3.3.

[15] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013. 3 citations in sections 2.1, 2.3, and 2.3.2.

[16] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010. One citation in section 2.3.2.

[17] Xiaomeng Chen. Maens+: A divide-and-conquer based memetic algorithm for capacitated arc routing problem. In *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*, volume 1, pages 83–88. IEEE, 2011. One citation in section 2.2.2.

[18] Yuning Chen, Jin-Kao Hao, and Fred Glover. A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 253(1):25–39, 2016. One citation in section 2.2.2.

[19] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976. One citation in section 2.2.

[20] Feng Chu, Nacima Labadi, and Christian Prins. A scatter search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169(2):586–605, 2006. One citation in section 2.2.2.

[21] Pietro Consoli and Xin Yao. Diversity-driven selection of multiple crossover operators for the capacitated arc routing problem. In Christian Blum and Gabriela Ochoa, editors, *Evolutionary Computation in Combinatorial Optimisation - 14th European Conference, EvoCOP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers*, number 12 in Lecture Notes in Computer Science, pages 97–108. Springer, 2014. 3 citations in sections (document), 4.13, and 5.4.3.

[22] Luis Da Costa, Alvaro Fialho, Marc Schoenauer, Michèle Sebag, et al. Adaptive operator selection with dynamic multi-armed bandits. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 913–920, 2008. 4 citations in sections 1.2.2, 2.3, 3.4.2, and 3.4.2.

[23] Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michèle Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 913–920. ACM, 2008. One citation in section 2.3.

[24] Lawrence Davis. Adapting operator probabilities in genetic algorithms. In *International Conference on Genetic Algorithms '89*, pages 61–69, 1989. One citation in section 2.3.

[25] James Stone DeArmon. *A comparison of heuristics for the capacitated Chinese postman problem*. PhD thesis, University of Maryland, 1981. 2 citations in sections 2.2 and 3.5.

[26] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. One citation in section 1.1.3.

[27] Federico Divina and Elena Marchiori. Knowledge-based evolutionary search for inductive concept learning. In *Knowledge Incorporation in Evolutionary Computation*, pages 237–253. Springer, 2005. One citation in section 2.4.2.

[28] Richard W Eglese. Routeing winter gritting vehicles. *Discrete applied mathematics*, 48(3):231–244, 1994. 5 citations in sections 2.2, 2.2.2, 3.5, 4.6, and 5.5.

[29] AE Eiben, Mark Horvath, Wojtek Kowalczyk, and Martijn C Schut. Reinforcement learning for online control of evolutionary algorithms. In *International Workshop on Engineering Self-Organising Applications*, pages 151–160. Springer, 2006. One citation in section 2.3.

[30] Agoston Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141, 1999. 3 citations in sections (document), 1.1.2, and 1.1.

[31] Liang Feng, Yew-Soon Ong, Meng-Hiot Lim, and Ivor W Tsang. Memetic search with interdomain learning: A realization between cvrp and carp. *IEEE Transactions on Evolutionary Computation*, 19(5):644–658, 2015. 2 citations in sections 2.2.2 and 2.4.3.

[32] Liang Feng, Yew-Soon Ong, Quang Huy Nguyen, and Ah-Hwee Tan. Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7. IEEE, 2010. 3 citations in sections 1.2.1, 2.2.2, and 2.2.3.

[33] Liang Feng, Yew-Soon Ong, Ivor Wai-Hung Tsang, and Ah-Hwee Tan. An evolutionary search paradigm that learns with past experiences. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012. 2 citations in sections 2.4.2 and 2.4.4.

[34] Álvaro Fialho. *Adaptive operator selection for optimization.* PhD thesis, Université Paris Sud-Paris XI, 2010. One citation in section 1.1.2.

[35] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *Learning and Intelligent Optimization*, pages 176–190. Springer, 2009. One citation in section 2.3.

[36] Álvaro Fialho, Marc Schoenauer, and Michèle Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 779–786. ACM, 2009. 2 citations in sections 1.2.1 and 2.3.

[37] Greg N Frederickson. Approximation algorithms for some postman problems. *Journal of the ACM (JACM)*, 26(3):538–554, 1979. One citation in section 2.2.

[38] Haobo Fu, Yi Mei, Ke Tang, and Yanbo Zhu. Memetic algorithm with heuristic candidate list strategy for capacitated arc routing problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010. One citation in section 2.2.2.

[39] Alex S Fukunaga. Genetic algorithm portfolios. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 1304–1311. IEEE, 2000. One citation in section 2.3.

[40] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3(4):379–397, 1999. One citation in section 13.

[41] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004. One citation in section 2.3.

[42] Abel Garcia-Najera and John A Bullinaria. Comparison of similarity measures for the multi-objective vehicle routing problem with time windows. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 579–586. ACM, 2009. 3 citations in sections 2.2, 3.3.1, and 13.

[43] David E Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5(4):407–425, 1990. One citation in section 2.3.

[44] Bruce L Golden, James S DeArmon, and Edward K Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47–59, 1983. One citation in section 2.2.1.

[45] Bruce L Golden and Richard T Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981. 5 citations in sections 2.2, 2.2.1, 3.2, 9, and 3.2.1.

[46] Carla P Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001. One citation in section 2.3.

[47] Wenyin Gong, Álvaro Fialho, Zhihua Cai, and Hui Li. Adaptive strategy selection in differential evolution for numerical optimization: an empirical study. *Information Sciences*, 181(24):5364–5386, 2011. One citation in section 2.3.

[48] Peter Greistorfer. A tabu scatter search metaheuristic for the arc routing problem. *Computers & Industrial Engineering*, 44(2):249–266, 2003. One citation in section 2.2.2.

[49] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005. One citation in section 2.4.2.

[50] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Multifactorial evolution: toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2016. One citation in section 2.4.3.

[51] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009. One citation in section 4.5.

[52] Jun He and Xin Yao. An analysis of evolutionary algorithms for finding approximation solutions to hard optimisation problems. In *Evolutionary Computation, 2003.*

CEC'03. The 2003 Congress on, volume 3, pages 2004–2010. IEEE, 2003. 2 citations in sections 1.1.1 and 2.3.1.

[53] Alain Hertz. Recent trends in arc routing. In *Graph theory, combinatorics and algorithms*, pages 215–236. Springer, 2005. One citation in section 2.2.

[54] Alain Hertz, Gilbert Laporte, and Pierrette Nanchen Hugo. Improvement procedures for the undirected rural postman problem. *INFORMS Journal on computing*, 11(1):53–62, 1999. One citation in section 2.2.1.

[55] Alain Hertz, Gilbert Laporte, and Michel Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations research*, 48(1):129–135, 2000. One citation in section 2.2.2.

[56] Alain Hertz and Michel Mittaz. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35(4):425–434, 2001. One citation in section 2.2.2.

[57] David V Hinkley. Inference about the change-point from cumulative sum tests. *Biometrika*, 58(3):509–523, 1971. 3 citations in sections 2.3, 3.4.2, and 4.4.

[58] Robert Hinterding, Zbigniew Michalewicz, and Agoston E Eiben. Adaptation in evolutionary computation: A survey. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 65–69. IEEE, 1997. One citation in section 2.3.

[59] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979. One citation in section A.1.

[60] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009. 2 citations in sections 1.1.2 and 2.3.

[61] Nur Insani, Kate Smith-Miles, and Davaatseren Baatar. Selecting suitable solution strategies for classes of graph coloring instances using data mining. In *Information Technology and Electrical Engineering (ICITEE), 2013 International Conference on*, pages 208–215. IEEE, 2013. One citation in section 2.3.1.

[62] Paul Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901. One citation in section 3.3.1.

[63] Klaus Jansen. An approximation algorithm for the general routing problem. *Information Processing Letters*, 41(6):333–339, 1992. One citation in section 2.2.

[64] Wojciech Jaskowski, Krzysztof Krawiec, and Bartosz Wieloch. Knowledge reuse in genetic programming applied to visual learning. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1790–1797. ACM, 2007. One citation in section 2.4.2.

[65] Yaochu Jin. *Knowledge incorporation in evolutionary computation*, volume 167. Springer, 2005. 3 citations in sections 1.2.3, 2.1, and 2.4.

[66] Bryant A Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. *proceeding of: Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 15-19, 1995*, 1995. One citation in section 2.3.

[67] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015. 2 citations in sections 1.2.1 and 2.1.

[68] Jeremy Z Kolter and M Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 123–130. IEEE, 2003. 2 citations in sections 2.3 and 4.4.

[69] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Chérif. A genetic algorithm for the capacitated arc routing problem and its extensions. In *Applications of evolutionary computing*, pages 473–483. Springer, 2001. One citation in section 2.2.2.

[70] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1-4):159–185, 2004. One citation in section 2.2.2.

[71] Philippe Lacomme, Christian Prins, and Alain Tanguy. First competitive ant colony scheme for the carp. In *Ant Colony Optimization and Swarm Intelligence*, pages 426–427. Springer, 2004. One citation in section 2.2.2.

[72] Frédéric Lardeux, Frédéric Saubion, and Jin-Kao Hao. Gasat: a genetic local search algorithm for the satisfiability problem. *Evolutionary Computation*, 14(2):223–253, 2006. One citation in section 4.2.1.

[73] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966. One citation in section 3.3.1.

[74] Ke Li, Alvaro Fialho, Sam Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014. One citation in section 2.3.

[75] Xiaohua Liu and S Louis. Combining genetic algorithm and case-based reasoning for structure design. Master's thesis, University of Nevada, Reno, 1996. One citation in section 2.4.2.

[76] Manuel López-Ibánez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical report, Citeseer, 2011. 2 citations in sections 1.1.2 and 2.3.

[77] Sushil Louis and Gong Li. Augmenting genetic algorithms with memory to solve traveling salesman problems. In *Proceedings of the Joint Conference on Information Sciences*, pages 108–111, 1997. One citation in section 1.2.3.

[78] Sushil J Louis. Genetic learning for combinational logic design. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):38–43, 2005. One citation in section 2.4.2.

[79] Sushil J Louis and John McDonnell. Learning with case-injected genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4):316–328, 2004. 2 citations in sections 1.2.3 and 2.4.2.

[80] Sushil Louisy, Gary McGrawy, and Richard O Wyckoy. Cbr assisted explanation of ga results. *Computer Science*, 812:855–6486y, 1992. One citation in section 2.4.2.

[81] Guanzhou Lu, Jinlong Li, and Xin Yao. Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In *Evolutionary Computation in Combinatorial Optimization*, pages 108–117. Springer, 2011. One citation in section 4.3.1.

[82] Monte Lunacek and Darrell Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 477–484. ACM, 2006. 2 citations in sections 2.3.1 and 4.3.2.

[83] Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013. One citation in section 2.3.1.

[84] Katherine M Malan and Andries P Engelbrecht. Fitness landscape analysis for metaheuristic performance prediction. In *Recent advances in the theory and application of fitness landscapes*, pages 103–132. Springer, 2014. One citation in section 2.3.

[85] Richard J Marshall, Mark Johnston, and Mengjie Zhang. Hyper-heuristic operator selection and acceptance criteria. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 99–113. Springer, 2015. One citation in section 2.3.

[86] Rafael Martí, Mauricio GC Resende, and Celso C Ribeiro. Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, 226(1):1–8, 2013. One citation in section 2.4.1.

[87] Rafael Martinelli, Marcus Poggi, and Anand Subramanian. Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, 40(8):2145–2160, 2013. 5 citations in sections (document), 2.2, 4.9, 4.6.4, and 4.13.

[88] Jorge Maturana and Frédéric Saubion. A compass to guide genetic algorithms. In *Parallel Problem Solving from Nature–PPSN X*, pages 256–265. Springer, 2008. 4 citations in sections 2.3, 2.3.1, 2.3.2, and 4.2.1.

[89] Yi Mei, Xiaodong Li, and Xin Yao. Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation, accepted on 31 July 2013*, 2013. One citation in section 2.2.2.

[90] Yi Mei, Xiaodong Li, and Xin Yao. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *Evolutionary Computation, IEEE Transactions on*, 18(3):435–449, 2014. 3 citations in sections (document), 4.6.4, and 4.13.

[91] Yi Mei, Xiaodong Li, and Xin Yao. Variable neighborhood decomposition for large scale capacitated arc routing problem. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1313–1320. IEEE, 2014. 4 citations in sections (document), 2.2.2, 4.6.4, and 4.13.

[92] Yi Mei, Ke Tang, and Xin Yao. A global repair operator for capacitated arc routing problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(3):723–734, 2009. One citation in section 2.2.2.

[93] Yi Mei, Ke Tang, and Xin Yao. Improved memetic algorithm for capacitated arc routing problem. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1699–1706. IEEE, 2009. 2 citations in sections 2.2.2 and 2.4.2.

[94] Yi Mei, Ke Tang, and Xin Yao. Decomposition-based memetic algorithm for multi-objective capacitated arc routing problem. *Evolutionary Computation, IEEE Transactions on*, 15(2):151–165, 2011. One citation in section 2.2.

[95] Yi Mei, Ke Tang, and Xin Yao. A memetic algorithm for periodic capacitated arc routing problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(6):1654–1667, 2011. One citation in section 2.2.

[96] Peter Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325, 2004. One citation in section 4.3.1.

[97] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):730–742, 2010. One citation in section 4.4.

[98] Leandro Lei Minku. *Online ensemble learning in the presence of concept drift*. PhD thesis, University of Birmingham, 2011. One citation in section 2.3.

[99] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989. One citation in section 2.4.1.

[100] Sibylle D Muller, Nicol N Schraudolph, and Petros D Koumoutsakos. Step size adaptation in evolution strategies using reinforcement learning. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 151–156. IEEE, 2002. One citation in section 2.3.

[101] Bart Naudts and Leila Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, 2000. One citation in section 2.3.

[102] Filipe V Nepomuceno and Andries P Engelbrecht. A self-adaptive heterogeneous pso for real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 361–368. IEEE, 2013. One citation in section 2.3.

[103] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *International conference on discovery science*, pages 264–269. Springer, 2007. One citation in section 2.3.

[104] Gabriela Ochoa, Rong Qu, and Edmund K Burke. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 341–348. ACM, 2009. One citation in section 2.3.1.

[105] Pietro S Oliveto, Jun He, and Xin Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007. One citation in section 1.

[106] CS Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974. One citation in section 2.2.

[107] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998. 2 citations in sections 1 and 1.1.1.

[108] Wen Lea Pearn. Augment-insert algorithms for the capacitated arc routing problem. *Computers & Operations Research*, 18(2):189–198, 1991. 2 citations in sections 2.2.1 and 22.

[109] Martin Pelikan. Nk landscapes, problem difficulty, and hybrid evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 665–672. ACM, 2010. One citation in section 2.3.1.

[110] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800, 2010. One citation in section 2.3.

[111] E Islas Pérez, CA Coello Coello, and A Hernandez Aguirre. Extraction of design patterns from evolutionary algorithms using case-based reasoning. In *International Conference on Evolvable Systems*, pages 244–255. Springer, 2001. One citation in section 3.

[112] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from NatureŮPPSN III*, pages 249–257. Springer, 1994. One citation in section 2.2.2.

[113] Jean-Yves Potvin and Samy Bengio. The vehicle routing problem with time windows part ii: genetic search. *INFORMS journal on Computing*, 8(2):165–172, 1996. 3 citations in sections 2.2.2, 23, and 3.4.1.

[114] Adam Prugel-Bennett. Benefits of a population: five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation*, 14(4):500–517, 2010. One citation in section 2.1.

[115] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. One citation in section 3.5.

[116] Connie Loggia Ramsey and John J Grefenstette. Case-based initialization of genetic algorithms. In *ICGA*, pages 84–91, 1993. One citation in section 2.4.2.

[117] Khaled Rasheed and Haym Hirsh. Using case based learning to improve genetic algorithm based design optimization. In *ICGA*, pages 513–520, 1997. One citation in section 2.4.2.

[118] Mohamed Reghioui, Christian Prins, and Nacima Labadi. Grasp with path relinking for the capacitated arc routing problem with time windows. *Applications of evolutionary computing*, pages 722–731, 2007. One citation in section 2.2.

[119] Robert G Reynolds. An introduction to cultural algorithms. In *Proceedings of the third annual conference on evolutionary programming*, pages 131–139. Singapore, 1994. 2 citations in sections 1.2.3 and 2.4.1.

[120] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, 2000. 4 citations in sections 2.2.2, 9, 3.3.3, and 15.

[121] Yoshitaka Sakurai, Kouhei Takada, Takashi Kawabe, and Setsuo Tsuruta. A method to control parameters of evolutionary algorithms by using reinforcement learning. In *Signal-Image Technology and Internet-Based Systems (SITIS), 2010 Sixth International Conference on*, pages 74–79. IEEE, 2010. One citation in section 2.3.

[122] Jeffrey C Schlimmer and Richard H Granger. Beyond incremental processing: Tracking concept drift. In *AAAI*, pages 502–507, 1986. One citation in section 4.4.

[123] Ronghua Shang, Hongna Ma, Jia Wang, Licheng Jiao, and Rustam Stolkin. Immune clonal selection algorithm for capacitated arc routing problem. *Soft Computing*, 20(6):2177–2204, 2016. One citation in section 2.2.2.

[124] Jorge A Soria Alcaraz, Gabriela Ochoa, Martin Carpio, and Hector Puga. Evolvability metrics in adaptive operator selection. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 1327–1334. ACM, 2014. One citation in section 2.3.1.

[125] Kenneth O Stanley. Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003. One citation in section 2.3.

[126] Jianyong Sun, Qingfu Zhang, and Xin Yao. Meta-heuristic combining prior online and offline information for the quadratic assignment problem. *IEEE transactions on cybernetics*, 44(3):429–444, 2014. One citation in section 2.3.

[127] Patrick Surry and Nicholas Radcliffe. Inoculation to initialise evolutionary search. *Evolutionary Computing*, pages 269–285, 1996. 2 citations in sections 3 and 2.4.2.

[128] Ke Tang, Yi Mei, and Xin Yao. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *Evolutionary Computation, IEEE Transactions on*, 13(5):1151–1166, 2009. 9 citations in sections 1.2.1, 2.2.1, 2.2.2, 3.2, 3.2.1, 3.3.3, 4.3, 4.5.1, and 6.1.

[129] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1539–1546. ACM, 2005. 3 citations in sections 1.2.2, 2.3, and 4.2.2.

[130] Gündüz Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985. 3 citations in sections 2.2.1, 2.2.2, and 9.

[131] Leonardo Vanneschi. Investigating problem hardness of real life applications. In *Genetic Programming Theory and Practice V*, pages 107–124. Springer, 2008. One citation in section 2.3.1.

[132] Leonardo Vanneschi, Manuel Clergue, Philippe Collard, Marco Tomassini, and Sébastien Vérel. Fitness clouds and problem hardness in genetic programming. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 690–701. Springer, 2004. One citation in section 2.3.1.

[133] Leonardo Vanneschi, Yuri Pirola, and Philippe Collard. A quantitative study of neutrality in gp boolean landscapes. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 895–902. ACM, 2006. One citation in section 4.3.3.

[134] Zhurong Wang, Haiyan Jin, and Manman Tian. Rank-based memetic algorithm for capacitated arc routing problems. *Applied Soft Computing*, 37:572–584, 2015. One citation in section 2.2.2.

[135] James M Whitacre, Tuan Q Pham, and Ruhul A Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1345–1352. ACM, 2006. One citation in section 2.3.

[136] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945. 3 citations in sections 3.5, 5.5.2, and A.1.

[137] Sanne Wøhlk. *Contributions to arc routing.* Citeseer, 2006. One citation in section 2.2.2.

[138] Sanne Wøhlk. New lower bound for the capacitated arc routing problem. *Computers & Operations Research*, 33(12):3458–3472, 2006. One citation in section 2.2.

[139] Sanne Wøhlk. An approximation algorithm for the capacitated arc routing problem. *Open Operational Research Journal*, 2:8–12, 2008. One citation in section 2.2.

[140] Sanne Wøhlk. A decade of capacitated arc routing. In *The vehicle routing problem: latest advances and new challenges*, pages 29–48. Springer, 2008. One citation in section 2.2.

[141] David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995. One citation in section 1.

[142] Y-Y Wong, K-H Lee, K-S Leung, and C-W Ho. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 7(8):506–515, 2003. One citation in section 2.3.

[143] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):405–410, 1997. One citation in section 5.3.3.

[144] Li-Ning Xing, Philipp Rohlfshagen, Ying-Wu Chen, and Xin Yao. A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(4):1110–1123, 2011. One citation in section 2.2.

[145] Lining Xing, Philipp Rohlfshagen, Yingwu Chen, and Xin Yao. An evolutionary approach to the multidepot capacitated arc routing problem. *Evolutionary Computation, IEEE Transactions on*, 14(3):356–374, 2010. One citation in section 2.2.

[146] Ying Xu and Rong Qu. An iterative local search approach based on fitness landscapes analysis for the delay-constrained multicast routing problem. *Computer Communications*, 35(3):352–365, 2012. One citation in section 2.3.1.

[147] Shengxiang Yang and Xin Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, 2008. One citation in section 2.4.2.

[148] Wen-Jun Yin, Min Liu, and Cheng Wu. A genetic learning approach with case-based memory for job-shop scheduling problems. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 3, pages 1683–1687. IEEE, 2002. One citation in section 2.4.2.

[149] Zhi Yuan, Thomas Stützle, Marco A Montes de Oca, Hoong Chuin Lau, and Mauro Birattari. An analysis of post-selection in automatic configuration. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1557–1564. ACM, 2013. One citation in section 2.3.

[150] Emmanouil E Zachariadis and Chris T Kiranoudis. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, 37(12):2089–2105, 2010. One citation in section 4.5.1.

[151] Jun Zhang, Zhi-hui Zhan, Ying Lin, Ni Chen, Yue-jiao Gong, Jing-hui Zhong, Henry SH Chung, Yun Li, and Yu-hui Shi. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4):68–75, 2011. 2 citations in sections 2.1 and 2.3.2.

[152] Tiantian Zhang, Michael Georgiopoulos, and Georgios C Anagnostopoulos. S-race: A multi-objective racing algorithm. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1565–1572. ACM, 2013. One citation in section 2.3.