AUTHOR: AYODELE, M.

TITLE: Effective and efficient estimation of distribution algorithms for permutation and scheduling problems.

YEAR: 2018

Effective and Efficient Estimation of Distribution Algorithms for Permutation and Scheduling Problems

Mayowa Ayodele

Computing and Digital Media

Robert Gordon University

A thesis submitted in partial fulfilment of the requirements of the Robert Gordon University for the degree of

Doctor of Philosophy

May, 2018

Abstract

Estimation of Distribution Algorithm (EDA) is a branch of evolutionary computation that learn a probabilistic model of good solutions. Probabilistic models are used to represent relationships between solution variables which may give useful, human-understandable insights into real-world problems. Also, developing an effective PM has been shown to significantly reduce function evaluations needed to reach good solutions. This is also useful for real-world problems because their representations are often complex needing more computation to arrive at good solutions. In particular, many real-world problems are naturally represented as permutations and have expensive evaluation functions. EDAs can, however, be computationally expensive when models are too complex. There has therefore been much recent work on developing suitable EDAs for permutation representation. EDAs can now produce state-of-the-art performance on some permutation benchmark problems. However, models are still complex and computationally expensive making them hard to apply to real-world problems.

This study investigates some limitations of EDAs in solving permutation and scheduling problems. The focus of this thesis is on addressing redundancies in the Random Key representation, preserving diversity in EDA, simplifying the complexity attributed to the use of multiple local improvement procedures and transferring knowledge from solving a benchmark project scheduling problem to a similar real-world problem.

In this thesis, we achieve state-of-the-art performance on the Permutation Flowshop Scheduling Problem benchmarks as well as significantly reducing both the computational effort required to build the probabilistic model and the number of function evaluations. We also achieve competitive results on project scheduling benchmarks. Methods adapted for solving a real-world project scheduling problem presents significant improvements.

Keywords: Estimation of Distribution Algorithm, Probabilistic Model, Random Key, Genetic Algorithm, Real-world Project Scheduling Problem, Optimisation, Gaussian Distribution, Permutation Flowshop Scheduling Problem.

I dedicate this thesis to my heavenly father who is always with me.

Acknowledgements

I would like to thank my supervisors Prof. John McCall and Dr. Olivier Regnier-Coudert for all the support given to me during my PhD. Words are not enough to appreciate them. They gave me the best supervision anyone could dream of. They are simply awesome.

I also want to thank the IDEAS institute of RGU for funding my PhD.

I say a big thanks to my family for encouraging me all through. I also appreciate my friends for supporting me through my research journey.

Contents

List of Figures

List of Tables

Abbreviations

ACO Ant Colony Optimisation

ARPD Average Relative Percentage Deviation

BPEDA Bi-Population Estimation of Distribution Algorithm

BPGA Bi-Population Genetic Algorithm

BPGA-EDA Bi-Population Genetic Algorithm and Estimation of Distribution Algorithm

CI Computational Intelligence

DE Differential Evolution

EAs Evolutionary Algorithms

EBNA Estimation of Bayesian Network Algorithm

EC Evolutionary Computation

EDA Estimation of Distribution Algorithm

EGNA Estimation of Gaussian Network Algorithm

EHBSA Edge Histogram-Based Sampling Algorithm

EP Evolutionary Programming

ES Evolutionary Strategies

ESGS Extended Schedule Generation Scheme

GA Genetic Algorithm

GM-EDA Generalised Mallows model based EDA

GP Genetic Programming

HGM-EDA Hybrid GM-EDA

ICE Induced Chromosome elements Exchanger

KTD Kendal Tau Distance

LOP Linear Ordering Problem

MIMIC Mutual Information Maximization for Input Clustering

MPBLS Multi-mode version Permutation-Based Local Search

MRCPSP Multi-Mode Resource Constrained Project Scheduling Problem

MSSGS Multi-mode Serial Schedule Generation Scheme

NHBSA Node Histogram-Based Sampling Algorithm

PBIL Population-Based Incremental Learning

PFSP Permutation Flowshop Scheduling Problem

PGM Probability Generation Mechanism

PSO Particle Swarm Optimisation

QAP Quadratic Assignment Problem

RCPSP Resource Constrained Project Scheduling Problem

REDA Recursive Estimation of Distribution Algorithm

RK Random Key

RK-EDA Random Key based Estimation of Distribution Algorithm

SGS Schedule Generation Scheme

TSP Travelling Salesman Problem

UMDA Univariate Marginal Distribution Algorithm

VNS Variable Neighbourhood Search

VRP Vehicle Routing Problem

Chapter 1

Introduction

1.1 Introduction

This chapter introduces and presents the scope of study for this thesis. It also presents the research questions as well as the summary of research publications produced during this research.

1.2 Background

In the past decades, there has been a drive to improve classical operation research methods by using meta-heuristics. A lot of industrial optimisation problems are considered to be NP-hard, making exact methods impracticable (Zlochin et al, 2004). For this reason, operations research has been one of the major application areas in the Computational Intelligence (CI) community. The community has produced many meta-heuristics that are capable of addressing complex real-world problems. Algorithms such as Genetic Algorithm (GA), Ant Colony Optimisation (ACO), Estimation of Distribution Algorithm (EDA) and Particle Swarm Optimisation (PSO) are seen as the core of CI (Kahraman et al, 2010). Other CI tools include fuzzy systems, artificial immune systems and artificial neural networks (Engelbrecht, 2007). Regarding impact, these algorithms have the potential of providing great benefits to industries as they have been noted for their ability to solve complex problems found in transportation and logistics, design and manufacturing as well as engineering. The extent to which the CI community is providing these benefits has however been of recent concern. Actual benefits gained in industries are not seen as commensurate with the research effort on CI techniques (Bonyadi et al, 2013; Chiong et al, 2012; Michalewicz, 2012a). The fact that there is a wide gap between academic research and industrial practice in this field has been recurrently highlighted by researchers.

One major branch of CI is Evolutionary Computation (EC) which focuses on algorithms that are based on the theory of evolution. Evolutionary Algorithms (EAs) are a subclass of EC. They consist of a population of solutions that change dynamically due to the emergence and elimination of individuals, and they also have the notion of fitness where fitter solutions are more likely to survive (Chiong et al, 2012). Finally,

EAs have the concept of reproduction where individuals produce similar offspring. Based on these defining factors, algorithms such as GA, Genetic Programming (GP), Evolutionary Strategies (ES) and Evolutionary Programming (EP) are considered to be direct types of EAs. Other similar algorithms such as EDAs and Differential Evolution (DE) are also considered EAs.

The genetic algorithm whose creation dates back to the 1960s is the most mature of EAs and has therefore been most widely studied. The GA is a population-based search algorithm that is the original prototype of an EA (Chiong et al, 2012). GAs require some experience of setting parameters to make the best of them. Their disadvantage lies in their inability to sufficiently exploit problem knowledge (Larrañaga and Lozano, 2002). As an improvement, EDA was created to make better use of problem knowledge through the use of probabilistic models.

EDAs which are also referred to as probabilistic model building genetic algorithms replace the crossover and mutation step of the GA with learning a probabilistic model of good quality solutions. This helps to drive the search towards promising regions of the search space (Zlochin et al, 2004). EDAs are also able to exploit problem knowledge and are amenable to the use of instance data to drive their model-building process, making them better suited for real-world problems.

Several variants of EDAs have been proposed for various problem representations. EDAs were initially designed for the binary representation, and much more were soon proposed for vector and continuous representations (Larrañaga and Lozano, 2002). However, EDAs have moved more slowly into permutations because of the difficulty of deciding how to construct and sample probabilistic models for this representation (Ceberio et al, 2012). A recurring problem is the need to respect mutual exclusivity in permutations. Many permutation EDAs require an additional procedure to ensure only valid permutations are produced. Their probabilistic models are also often large, growing exponentially with the size of a problem. However, permutation problems are sometimes solved using alternative representations such as Random Key (RK). These naturally translate into valid permutations. Also, EDAs based on RK representation often use probabilistic models of smaller dimension growing linearly with the size of the problem. However, in the RK representation, multiple genotypes generate identical phenotypes. This redundancy associated with the RK representation makes it less effective. Previous research has shown that EDAs based on RK representation exhibit poorer performance than those using other representations when applied to common permutation problems (Ceberio et al, 2012). Despite their poor performance, RK representations are amenable to lighter-weight, less computationally intensive model-building and sampling. Research continues to investigate how to design effective and efficient EDAs for permutation problems. Redundancy elimination techniques that can overcome the poor performance of RK representations are therefore of potential interest and significance.

Presently, advanced permutation EDAs such as the Generalised Mallows model based EDA (GM-EDA) exhibit the problem of premature convergence (Ceberio et al, 2014a). There has therefore been some work on improving diversity in permutation EDAs by using local search. An example is the state-of-the-art EDA, Hybrid GM-EDA (HGM-EDA)(Ceberio et al, 2014a) which is a hybrid of GM-EDA and Variable

Neighbourhood Search (VNS). Restart mechanisms are also sometimes used to escape local optima (Ceberio et al, 2014b). The introduction of diversity preserving methods in EDAs remains an area of interest in the community.

Scheduling problems are some of the most are frequently studied optimisation problems. They are usually composed of a set of tasks, a set of available resources and some constraints and are seen in many real-world situations such as timetabling, staff rostering, equipment maintenance scheduling etc. (Hart et al, 2005). Scheduling problems are often naturally represented as permutations. Although not all scheduling problems are permutation problems and vice versa, the difficulty of EDAs in the permutation domain affects their performance on scheduling problems.

Scheduling problems are considered even more complex when they are also categorised as multi-component problems. These are problems that consist of multiple interacting sub-problems. Examples are the vehicle routing problems that are dependent on the container loading problem (Rizzoli et al, 2007) and certain project scheduling problems which consist of activity scheduling as well as mode assignment problems (Voß and Witt, 2007). Researchers have argued that many real-world optimisation problems are multi-component problems. There has therefore been a drive towards solving this class of problems in an attempt to bridge the theory to practice gap that exists in the field (Bonyadi et al, 2013). Existing algorithms for solving multi-component benchmarks are however highly complex. They combine meta-heuristics with many improvement procedures and local search methods (Van Peteghem and Vanhoucke, 2011). Each of these procedures contributes to the overall computational cost. By encapsulating learning in an explicit probabilistic model, EDAs are less noisy than classical EAs. EDAs may therefore be able to approximate the optimum more quickly and effectively than a GA, thus reducing or eliminating the need for local search. However, premature convergence also exists in EDAs when the variance of iteratively re-sampled distributions becomes very small. Therefore, appropriate control of model parameters is included in this investigation.

1.3 Research Questions

This thesis focuses on answering the following research questions

- In what way can the problem of redundancy in RK be addressed?
- In what way can diversity be controlled in EDAs designed to solve permutation problems?
- How can we reduce the need for local improvement procedures in multi-component scheduling problems?
- How can we transfer knowledge from solving a multi-component test problem to a similar real-world problem?

1.4 Published Works and Thesis Overview

A brief background to this study has been presented in this chapter. During this research, some research publications were produced and contribute to some of the thesis chapters. The publications are listed as follows in ascending order of publication date.

1. Ayodele, M., McCall, J. and Regnier-Coudert, O., 2015, July. Probabilistic Model Enhanced Genetic Algorithm for Multi-Mode Resource Constrained Project Scheduling Problem. In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (pp. 745-746). ACM.

2. Ayodele, M., McCall, J. and Regnier-Coudert, O., 2016, July. BPGA-EDA for the multi-mode resource constrained project scheduling problem. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 3417-3424). IEEE.

3. Ayodele, M., McCall, J. and Regnier-Coudert, O., 2016, September. RK-EDA: A novel random key based estimation of distribution algorithm. In International Conference on Parallel Problem Solving from Nature (pp. 849-858). Springer International Publishing.

4. Ayodele, M., McCall, J., Regnier-Coudert, O. and Bowie, L., 2017. A random key based estimation of distribution algorithm for the permutation flowshop scheduling problem. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 2364-2371). IEEE.

5. Ayodele, M., McCall, J. and Regnier-Coudert, O., 2017. Estimation of distribution algorithms for the multi-mode resource constrained project scheduling problem. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 1579-1586). IEEE.

The rest of this thesis is structured as follows.

Chapter 2, Literature Survey. This chapter presents real-world applications of EDAs and ACO which are often studied together under model-based search algorithms. Some well-studied permutation and scheduling problems are also presented. Furthermore, a review of leading approaches to solving these problems is presented.

Chapter 3, RK-EDA: A Novel Random Key Based Estimation of Distribution Algorithm. This chapter proposes a novel RK-EDA for solving permutation problems. RK-EDA eliminates redundancies in the RK representation by normalising the genes in a solution. It also uses a cooling scheme to control exploration and exploitation. In this chapter, RK-EDA is applied to a range of common permutation problems; Permutation Flowshop Scheduling Problem (PFSP), Quadratic Assignment Problem (QAP), Linear Ordering Problem (LOP) and Travelling Salesman Problem (TSP) benchmark instances and particularly showed highly competitive performance on the PFSP. This chapter is developed from the study presented in the 3^{rd} paper (Ayodele et al, 2016b) listed above.

Chapter 4, RK-EDA for the Permutation Flowshop Scheduling Problem. A more focused detail of the application of RK-EDA for the PFSP is presented in this chapter. Parameters are tuned specifically for this problem leading to improved results and new best-known solutions on the largest problem instances. In this chapter, the rate of exploration/exploitation at each generation is estimated by measuring the probability of swaps in a permutation. An experimental evaluation of the exploration/exploitation is also presented showing similar results. With this approach, it is possible to determine the expected exploration/exploitation rate during a run of the algorithm without executing the algorithm. The 4^{th} paper (Ayodele et al, 2017b) was used to develop this chapter. A better set of parameters were however found after the paper was published and new results are used in the chapter.

Chapter 5, Estimation of Distribution Algorithms for the Multi-mode Resource Constrained Project Scheduling Problem. The Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP) is a multi-component problem that consists of two optimisation problems; one is permutation-based while the other is represented as a set of integers. It is common practice to embed many local search and improvement procedures in meta-heuristics when solving this problem. The aim of this chapter is to reduce the need for local improvements by introducing EDAs. This chapter adapts RK-EDA for solving the Resource Constrained Project Scheduling Problem (RCPSP), which shares similar properties with the permutation component of the MRCPSP. BPGA-EDA which combines GA and Population-Based Incremental Learning (PBIL) is also proposed in this chapter. Although RK-EDA struggled to produce competitive on the RCPSP, BPGA-EDA improved one of the most competitive methods of solving the MRCPSP. This chapter also presents the BPEDA which is made up of RK-EDA and PBIL. BPEDA was able to achieve highly competitive results with fewer improvement procedures. This demonstrates that highly competitive results can be achieved even while reducing the use of local improvement procedures. This chapter is developed from the 1^{st}, 2^{nd} and 5^{th} papers (Ayodele et al, 2015, 2016a, 2017a) listed above.

Chapter 6, EDA for Real-World Project Scheduling Problem. In this chapter, methods of solving the RCPSP and MRCPSP are adapted for solving a real-world project scheduling problem. The work takes business procedures into consideration. The proposed algorithms are shown to outperform the industrial software, Primavera.

Chapter 7, Conclusions and Future Work. This chapter concludes the thesis and presents directions for further research. Limitations of approaches used throughout this thesis are also discussed.

Chapter 2

Literature Review

2.1 Introduction

In the previous chapter, the main research themes of this thesis were introduced. A literature review investigating these research themes is presented in this chapter. One of the most mature algorithms in CI is the GA. An adaptation of this algorithm is the EDA also known as the probabilistic model building GA. EDAs have the benefit of capturing knowledge about a search space making them better suited for solving complex problems. ACOs are also often studied under the same category as EDAs because of their use of probabilistic model known as pheromone model. These model-based search algorithms can be fundamental in bridging the theory-to-practice gap in the application of EAs. In this study, a review of real-world applications of EDAs as well as ACOs is presented. This review motivates the focus on scheduling and permutation problems.

Some of the most frequently studied permutation and/or scheduling problems in literature are PFSP, QAP, TSP, and LOP. Another scheduling problem which has been well-studied in the last few decades is the RCPSP as well as its variant MRCPSP. In this chapter, we formally define these problems and present a review of EDAs applied to solving them. Based on this review, the need for a more efficient and effective EDA is motivated.

The rest of this chapter is structured as follows. In Section 2.2, we identify factors that make real-world problems considered complex. Section 2.2.1 presents common probabilistic models. Section 2.3 introduced the EDA and ACO. Section 2.4 presents the real-world applications of ACO and EDA. Section 2.5 formally defines common permutation and scheduling problems while Section 2.6 presents a review of EDAs proposed for solving these problems. Section 2.7 presents a discussion on the research focus of this thesis.

2.2 Modelling Real-World Problems

Real-world industrial problems are considered "complex" because they are:

- large: it is computationally infeasible to compute every possible solutions (Chiong et al, 2012)
- multi-objective: optimal decision relies on two or more objectives which may even be conflicting (Hart et al, 2005)
- highly constrained: the search for optimality ceases to be the major problem but is replaced by the search for feasibility (Chiong et al, 2012)
- uncertain: real-world problems change very often, the objective function can vary significantly with time or may be noisy. Also, applications designed to suit a user may soon become obsolete when company rules change (Chiong et al, 2012; Michalewicz and Fogel, 2000)
- characterised by multiple dependencies: features/elements of real-world processes are often highly interrelated (Fischer et al, 2011; Koller and Friedman, 2009).

Although EAs present an alternative to classical operational research methods, there is a theory-to-practice gap in their applications. Methods developed based on artificial test problems are often not directly applicable to real-world problems. This is due to oversimplification of constraints (Michalewicz, 2012b). Many artificial test problems are considered unrealistic because complexity has been characterised purely in terms of size. They neglect the fact that real-world optimisation problems are usually compound problems (consisting of 2 or more sub-problems) (Bonyadi et al, 2013). Although there has been a research focus on creating more realistic benchmark problems, many problem sets are still considered far from reality (Michalewicz, 2012b).

Another focus area of the community is multi-objective and constrained optimisation problems (Deb, 2001). In more recent years however, there has been research interests in dynamic and multi-component optimisation problems. Dynamic problems require a solution approach to model uncertainties in optimisation problems. However, multi-component problems consists of sub-problems that cannot be solved in isolation. Modelling interdependencies between components of problems using probabilistic models has been a successful research area (Larrañaga and Lozano, 2002). Probabilistic model building EAs, such as EDAs, perform competitively on many complex problems (Hauschild and Pelikan, 2011). They are particularly considered well suited for data-driven approaches. This work therefore focuses on problem solving using probabilistic models.

2.2.1 Probabilistic Models

Probabilistic models are formal techniques used for capturing dependencies that exist between the parameters of a process (Baluja and Davies, 1998). They can be used for encoding uncertain knowledge (Larrañaga and Lozano, 2002) and are considered better at giving the reflection of a domain than purely hand constructed models (Koller

and Friedman, 2009). Probabilistic Models can also be useful for interpreting uncertain data (Jordan et al, 2004). The advantage of probabilistic models in making uncertainty explicit makes them able to depict reality more faithfully (Koller and Friedman, 2009). In literature, Bayesian and Gaussian networks are the most extensively studied Probabilistic Models (Larrañaga and Moral, 2011). Markov Network is another probabilistic model that has been widely used (Larrañaga and Lozano, 2002).

2.2.1.1 Bayesian Network

Bayesian network has particularly been widely used by AI researchers for representing conditional (in)dependence that exists within a system/process (Heckerman et al, 1995; Zhang and Wu, 2012).

Shakya and Santana (2012) defines a Bayesian network as a pair (B, Θ). B and Θ respectively denote the structure of the model and the set of parameters of the model. B is a Directed Acyclic Graph (DAG) where each node represents a variable and the edges represent the conditional dependencies. A set of nodes Π_i is a parent of X_i if there are edges from each variable in Π_i to X_i. $\Theta = \{ p(x_1|\Pi_1), p(x_2|\Pi_2), \cdots, p(x_n|\Pi_n) \}$ is the set of conditional probabilities. $p(x_n|\Pi_n)$ is used to denote the set of probabilities associate with $X_i = x_i$ given the different configurations of its parent variables Π_i.

Summarily, given a set of variables $X = \{X_1, X_2, \cdots, X_n\}$, a joint probability distribution $p(x)$ for a Bayesian network is represented as shown in eq. (2.1)

$$p(x) = \prod_{i=1}^{n} p(x_i|\Pi_i) \tag{2.1}$$

2.2.1.2 Gaussian Network

In Gaussian Network, each variable $X_i \in \mathbf{X}$ is continuous and each local density function is the linear regression model (Larrañaga and Lozano, 2002). A Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ over a vector of n random variables $X = \{X_1, X_2, \cdots, X_n\}$ is defined by two parameters. The first is μ which is the vector of means for each variable. When all variables of a network are assumed to be independent, Σ is a vector of variances across the distribution. Otherwise, Σ is a positive-definite and symmetric nn covariance matrix (Shakya and Santana, 2012).

2.2.1.3 Markov Network

A Markov network is a pair (G, Ψ) where G and Ψ respectively denote the structure of the model and the set of parameters of the network. G is an undirected graph where each node represents a variable and the edges represent the conditional dependencies between variables. In a Markov network, relationship between nodes are seen as a neighbourhood relationship rather than a parenthood relationship. $\boldsymbol{N} = \{N_1, N_2, \cdots, N_n\}$ denotes the neighbourhood system on G, where N_i is the set of neighbouring nodes of node X_i.

Markov networks are usually represented as shown in eq. (2.2). A feature $f_i(x)$ is a real-valued function of the state and w_i are weights. T is the temperature and $Z = \sum_{x \in \Omega} \prod_{i=1}^{m} \Psi(c_i)$ is the normalisation constant. Here, Ω is the set of all possible combinations of variables in X. $\Psi(c_i)$ is a potential function on clique $C_i \in X$ (Shakya and Santana, 2012).

$$p(x) = \frac{e^{-\sum_{i=1}^{m} w_i f_i(x)/T)}}{Z} \tag{2.2}$$

2.3 Model-Based Search Algorithms

The classic GA was not designed to explicitly learn dependencies between properties of a problem (Larrañaga and Lozano, 2002). As noted in Chapter 1, EDA was adapted from the GA such that it drives its search through the explicit use of probabilistic models rather than crossover. The focus of this study is on EDAs. However, EDAs are often classified alongside ACOs as model-based search algorithms (Zlochin et al, 2004). Also, ACO has a variant (hypercube-cube framework) that is equivalent to a variant of the EDA (PBIL) (Blum and Dorigo, 2004). This section therefore describes EDAs and ACOs.

2.3.1 Estimation of Distribution Algorithms

EDAs like other EAs are stochastic and population-based. They explore the search space by building and sampling explicit probabilistic models of promising candidate solutions (Hauschild and Pelikan, 2011; Pelikan et al, 2006). This explicit use of probabilistic models in optimisation provides a better understanding of the problem domain and helps drive the search towards regions of good solutions. It also makes EDAs outperform other types of meta-heuristics on many problems (Hauschild and Pelikan, 2011). EDAs repeatedly follow these sequence of steps; select a set of promising solutions from a population of individuals, learn the joint probability distribution of the selected individuals and sample from this probability distribution (Larrañaga and Lozano, 2002).

There are many types of EDAs and they can be grouped by the way they model the interrelations between problem variables. Some EDAs are univariate ignoring all interactions between problem variables. They assume that the problem variables are all independent (Hauschild and Pelikan, 2011). Examples of univariate EDAs are Bit-based simulated crossover (Syswerda, 1992), Population-Based Incremental Learning (PBIL) (Baluja, 1994), Univariate Marginal Distribution Algorithm (UMDA) (Pelikan and Mühlenbein, 1998), compact Genetic Algorithm (cGA) (Harik et al, 1999) and Genepool Optimal Mixing Evolutionary Algorithm (GOMEA) (Bosman and Thierens, 2013). There are also bivariate EDAs that captures pair-wise interactions between the variable of a problem. Examples are Mutual Information Maximization for Input Clustering (MIMIC) (De Bonet et al, 1997), Combining Optimizers with Mutual Information Trees (COMIT) (Baluja and Davies, 1997) and Bivariate Marginal Distribution Algorithm (BMDA) (Pelikan and Mühlenbein, 1999). Finally, there are

multivariate EDAs. These are EDAs that are able to capture multiple problem dependencies such as Affinity propagation EDA (Aff-EDA) (Santana et al, 2010), Bayesian Optimization Algorithm (BOA) (Pelikan, 2005a), Hierarchical Bayesian Optimization Algorithm (hBOA) (Pelikan, 2005b), Dependency Structure Matrix Genetic Algorithm (DSMGA) (Yu et al, 2003), Distribution Estimation Using Markov network (DEUM) (Shakya and McCall, 2007), Estimation of Bayesian Network Algorithm (EBNA) (Etxeberria and Larranaga, 1999), Extended Compact Genetic Algorithm (ECGA) (Harik, 1999), Factorised Distribution Algorithm (FDA) (Mühlenbein and Mahnig, 1999), Learning Factorized Distribution Algorithm (LFDA) (Mühlenbein and Mahnig, 1999) and Markovianity based optimization algorithm(MOA) (Shakya and Santana, 2008).

EDAs can also be grouped according to the problem representations they are designed for. The three main categories are EDAs for discrete variables, real-valued vectors and permutations (Hauschild and Pelikan, 2011).

EDAs for discrete variables work on a fixed-length solution string with a finite cardinality. A probabilistic model is used to capture the probabilities of different values of the variables of a problem. Two frequently used discrete variable EDAs are UMDA and PBIL.

EDAs for real-valued vectors work on a solution with infinite cardinality. It is therefore not possible to enumerate the probabilities of variables' values. The Univariate Marginal Distribution Algorithm for continuous domain (UMDA$_c$) (Larranaga et al, 1999), Estimation of Gaussian Networks Algorithm (EGNA) (Larranaga et al, 1999) and real-coded Bayesian optimization algorithm (rBOA) (Salinas-Gutiérrez et al, 2009) are types of real-valued EDAs.

Of particular interest in this study are EDAs for permutations which are popular for many real-world problems and in particular for scheduling problems (Hauschild and Pelikan, 2011). These EDAs often need to capture two types of constraints which are absolute positions as well as mutual exclusivity constraints in solutions. EDAs for permutation problems have particularly been of recent research interest in the community (Ceberio et al, 2012). Some examples of permutation based EDAs are Generalized Mallows Estimation of Distribution Algorithm (GM-EDA) (Ceberio et al, 2014a), Node Histogram Based Sampling Algorithms (NHBSA) and Edge Histogram Based Sampling Algorithms (EHBSA) (Tsutsui, 2002).

2.3.2 Ant Colony Optimisation

ACO, is a population-based and model-based search algorithm inspired by the natural behaviour of ants when foraging for food. Real ants leaves pheromone on the ground to attract other members of the colony to a favourable path. A similar approach is used by the ACO. It uses a particular type of probabilistic model (called pheromone model) whose structure is made up of some stochastic procedures called artificial ants (Zlochin et al, 2004). Each artificial ant builds a solution to the given problem using knowledge of the search space obtained from other ants. ACO repeatedly constructs candidate solutions using a pheromone model and modifies the pheromone values such that there is bias towards high quality solutions.

The original type of ACO proposed was the Ant System (AS) (Dorigo et al, 1996). Some other frequently used ACOs are Ant Colony System (ACS) (Montemanni et al, 2005) and MIN-MAX Ant System (MMAS) (Stützle and Hoos, 2000)(Dorigo et al, 2006). Other types of ACO include Hyper-Cube Framework for ACO (Blum et al, 2001), Continuous Orthogonal Ant Colony (Hu et al, 2008) and Recursive ACO (Gupta et al, 2012).

ACO is known for solving complex combinatorial optimization problems like TSP and QAP (Dorigo et al, 1999). They have been widely applied to many real-world industrial problems such as the vehicle routing problem (Donati et al, 2008; Gambardella et al, 2003; González-Barbosa et al, 2010; Hämmerle and Ankerl, 2013; Nechita et al, 2008; Pellegrini et al, 2007; Rizzoli et al, 2004, 2007).

2.4 Real-world applications of EDAs and ACOs

For a research to be classified as a real-world application, it must be based on real-world data (Chiong et al, 2012). Also, a bi-directional flow of knowledge between researchers and industrial experts is an important step to translating academic ideas to what works in practice (DEste and Patel, 2007). Furthermore, the evaluation stage is another important step to implementing an algorithm in practice. Many flaws may not be discovered until algorithms are tested by experts from the industry. Finally, evaluation helps to reveal important business rules (Rizzoli et al, 2007).

We have identified the benefits of model building EAs in solving real-world problems and also established the similarities between EDAs and ACOs. This section therefore presents a review of both algorithms in practice using the following factors.

- Real-world data has been used
- There has been expert involvement
- There has been industrial evaluation
- Algorithm has been implemented in an industry

Each reviewed research work satisfies at least one of the attributes listed above.

2.4.1 History Matching Problems

The History matching problem is an inverse problem where observed reservoir behaviour is used to estimate certain parameters that caused the behaviour (Oliver and Chen, 2011). It involves the calibration of a reservoir model to reproduce historic observation data as well as production optimisation and forecasting (Abdollahzadeh et al, 2013). History matching is a time consuming exercise taking several months to do manually (Abdollahzadeh et al, 2013). It is also often full of uncertainties. The size of the problem and the uncertainty factor makes it a complex problem to solve. EDA has been applied to the history matching problem in (Abdollahzadeh et al, 2013) and ACO in (Hajizadeh et al, 2011).

In (Abdollahzadeh et al, 2013), BOA (an EDA) was applied to the problem using a real oilfield in the North Sea. The performance of BOA on the problem was compared to that of a commercialised GA. BOA was seen to produce better results in terms of lower misfit within a shorter period of time. EDA is able to better exploit problem-specific knowledge based on historic data to achieve better solutions. The use of real data was reported as well as interactions with industrial experts. No expert evaluation or industrial implementation was however reported in the study.

Another EDA applied to history matching problem is the PBIL (Petrovska and Carter, 2006). Two real-world fields were considered in the study using fifteen years production history. Company details were kept anonymous in the study. PBIL was shown to outperform the standard GA on this problem. Industrial involvement can be deduced but no industrial assessment or implementation was reported.

Hajizadeh et al (2011) applied ACO to the history matching problem and uncertainty quantification of reservoir models. The authors considered the real-world case of the Teal south reservoir in the gulf of Mexico. Producing a lot of history matched models is considered very important as it will help to better quantify the level of uncertainty. ACO is able to produce multiple valid models. This research relies on historic data for estimating the level of mismatch associated with solutions. Real-world data has been used and interaction with the oil and gas industry can be inferred. Industrial validation and implementation are not reported.

2.4.2 Routing Problems

The Vehicle Routing Problem (VRP) is one of the common application areas of model-based search algorithms. This problem entails scheduling vehicles to move items from one place to another. Researchers have classified VRPs based on features such as the number of pick up points, whether or not there is a heterogeneous or homogeneous fleet, whether or not there are strict or flexible time windows and whether or not jobs can be assigned to vehicles after they have taken off (Braekers et al, 2016). This has however led to a large number of variants, some of which are peculiar to certain authors. Examples are VRP with Pick-up and Delivery (VRPPD) (Rizzoli et al, 2004), VRP with time window (VRPTW) (Rizzoli et al, 2007), time dependent VRP (TDVRP), (Donati et al, 2008), dynamic VRP (Rizzoli et al, 2004), static VRP (Gambardella et al, 2003) and rich VRP (Pellegrini et al, 2007; Regnier-Coudert et al, 2016).

Other routing problems presented in this section are the delivery problem in Ochiai and Kanoh (2014) and the crane routing problem in (Hirsch et al, 2012).

The VRP described by Rizzoli et al (2007) entails distributing items from a depot to over 600 stores using heterogeneous vehicles. A real-world constraint that depicts that all jobs must be completed within a single day while respecting each customer's time window is imposed. Other real-world constraints such as the unsuitability of certain vehicles for some tasks or locations reduce the space of feasible solutions. To meet the objectives of this VRP which are to reduce the number of vehicles and time taken to complete tasks (Bräysy and Gendreau, 2005), this problem was solved using Multi Ant Colony System (MACS) which is a variant of the ACS. This algorithm

involves two interacting colonies. An ant colony called ACS-VEI attempts to minimise the number of vehicles and the other ant colony called ACS-TIME attempts to minimise the time taken (Rizzoli et al, 2007). This problem is multi-objective (minimise time and number of vehicles) and large (entails scheduling a fleet of over a hundred vehicles to service 600 stores on a daily basis). It can also be seen as highly constrained (time-window constraint, vehicle compatibility constraint). The ability of ACO to correctly model the relationship that exists in the problem is a strength of model-based search algorithm. The solution approach also involved embedding business rules. In (Rizzoli et al, 2007), the first set of solutions presented to the planners were considered infeasible due to their non-adherence to a business rule which dictates that schedules should be generated according to regions rather than considering all locations together. This was corrected, and the final application accepted even when results produced were considered worse compared to the first by the researchers. The use of real-world data, expert involvement, industrial evaluation and implementation are reported in (Rizzoli et al, 2007).

Nechita et al (2008) applied MACS to a VRP with objectives of minimising distance and the number of circuits. This study originated from a distributor of dietary products in Romania. The VRP is characterised by strict time windows. The fact that environmental changes such as weather can give preference to an objective over another at a given moment created a need for the problem to be modelled in a non-static way. The need for handling unexpected changes in route (e.g due to weather) emerged from working on this real problem. Proposing a solution that satisfies these constraints led to a ten percent savings on transportation in a year. This shows that model-based search algorithms can not only replace existing methods but has the capacity of improving them.

The VRP described in (Rizzoli et al, 2004) involves more than one pick-up point and jobs can take over a day to be completed due to the mode of operation of the case-studied logistics operator in Italy. These pick-up and delivery activities are done using a homogeneous fleet of vehicles. Similar to the VRP described in (Nechita et al, 2008), time windows need to be respected. Here, a large number of customers reported to be between 1,000 and 1,500 are served at a time. Also, drivers are constrained by a particular number of hours of work per day and all pick-ups as a matter of policy must be done before delivery starts. The real-world difficulty here includes the need to handle the uncertainty related to traffic congestion as well as loading and unloading space. This real-world problem has a business rule that depicts that the number of cities (not customers) to visit in a tour must not be more than six. Solutions not conforming to this standard are considered infeasible. Furthermore, the loading and unloading time are considered to be fixed which is reported as being imposed by the client. Different from the objective reported in theory and other practical problems, the aim here is to improve the routing efficiency. This is to reduce pollution and traffic congestion due to commercial transport. MACS algorithm was also used to tackle this problem differently due to the different objectives. The implementation of this algorithm is reported to produce a substantial improvement regarding tour efficiency to that of the human planners. The human planners experienced a decline in efficiency as problem size increases while the implemented algorithm produces better efficiency

for larger problems.

The VRP presented in (Donati et al, 2008) pays particular attention to the effective modelling of the time taken to execute tasks. Here, there is the need to handle the uncertainty associated with the traffic situations, especially in urban regions. The fact that there is a hard constraint on time window has made this aspect very important. MACS has been used to solve this problem using real data from the Padua logistic district in the Veneto region of Italy (Donati et al, 2008). The case study involves servicing 60 customers with 10 trucks. This work sheds more light on the fact that solutions to real-world problems need to be data-driven to be indeed feasible as factors like traffic situation can vary a lot which can easily make estimates unreasonable. A higher degree of infeasibility is reported with a higher level of time-dependency. Although real-world data was used, no industrial involvement or real-world implementation was however reported.

Gambardella et al (2003) analysed a VRP from both static and dynamic contexts. Static means that all jobs are known before departure while dynamic means that additional jobs can be assigned to vehicles after they have taken off. Gambardella et al (2003) presented two software tools (*AntRoute* and *DyvOil*) based on the ACO. *AntRoute* is developed to handle the static VRP while *DyvOil* is designed to handle both dynamic and static scenarios. Creating such tools were made possible by engaging with some real-world companies. A fuel oil distribution company and a supermarket supply chain company were noted. They were able to identify the common needs of these companies and develop software tools that can handle them. These needs include forecasting customer requests based on previous orders and handling the dynamism attached to placing orders at different hours of the day. Pina Petroli, a fuel oil distribution company is said to have recorded an increase in the number of successful calls (customer calls leading to order) from one out of four to one out of two which is credited to the use of *DyvOil*. We see this success can be attached to dealing with real situations, engaging with the industry, getting feedback and successful implementation in at least one company. The advantage of making solution data-driven is also identified (Gambardella et al, 2003).

Pellegrini et al (2007) describes the VRP from the context of the problem faced by a distributor of food products in Italy. Here, customers have strict but many time windows which make this VRP more flexible. There is a company rule that imposes a maximum duration to tours as drivers are paid cheaper for the first eight hours and higher thereafter. Similar to the VRP in (Rizzoli et al, 2007), there are more than a type of vehicle. Additionally, this particular real-world problem is characterised by multiple time windows (i.e. having a series of time slots for pick-up/delivery rather than a fixed time slot). This problem was solved using Multiple Ant Colony System (MACS) and Multiple MAX-MIN Ant System (M-MMAS) working hand in hand (Pellegrini et al, 2007). The choice of ACO was based on the complexity of the problem structure as well as its ability to solve multi-objective problems (Pellegrini et al, 2007). Real data was used in this study, and there is evidence of industrial involvement in this study. There is however neither any evidence of industrial evaluation nor implementation.

Ochiai and Kanoh (2014) presented the application of ACO to a real-world delivery

problem.This problem originates from a home delivery service in Tokyo, Japan and is formulated as a TSP. Historical data has been used in this study. The study, however, does not suggest the involvement of industrial experts at any stage. It also does not indicate that this work has been implemented for use in a real-world industry.

ACO has been applied to the crane routing problem experienced by a roof-tile producing company (Hirsch et al, 2012). The crane routing problem in this case study involves minimising the sum of working times of two cranes (used to move tile batches and load carriers). There is also the aim of balancing the workload distribution between both cranes. The authors describe this problem as a state-dependent problem as the operation (path) of one crane is dependent on the other. To handle this, they used two communicating ants to represent the cranes. The path of each ant, therefore, corresponds to the path of the cranes, therefore, ensuring feasibility regarding avoiding collisions. The proposed method is able to give solutions within a reasonable amount of computation time. Hirsch et al (2012) recorded success in their application of ACO to this problem as they were able to produce feasible results for the company involved. We can infer that the researchers used real data, engaged with the industry, had their methods tested by industrial experts and implemented their approach

2.4.3 Job Shop Scheduling Problems

Blażewicz et al (1996) described Job Shop Scheduling Problem (JSSP) as consisting of different machines that work on jobs. The jobs need to follow a sequence of operation by the machines. This problem is that of specifying the job sequence on the machines with the aim of minimising the makespan, the total time for completing all jobs. (Blażewicz et al, 1996). When solving this problem in practice, there are often other objectives to consider in addition to or different from minimising the makespan (Hart et al, 2005). An example is tardiness. This section reviews the applications of model-based search algorithms to JSSP instances. They are the applications of EDAs to JSSPs in (Zhang and Wu, 2012), (Salhi et al, 2007) and (Wu and Huang, 2013) as well as applications of ACO to JSSPs in (Gagné et al, 2006), (Montgomery et al, 2006) and (Chica et al, 2011).

EDA was applied to a JSSP faced by a large scale speed reducer manufacturer in China (Zhang and Wu, 2012). The need to base the criteria of JSSP on due-date-related performance rather than merely using the makespan was identified, the problem was therefore addressed from this perspective. Objective functions that are based on due dates are seen to be more practical hence the recommendation of the total weighted tardiness criterion. Furthermore, the authors identified the need to deal with due dates of products and consider products which must be delivered in a batch dynamically from the experience of working with a real-world company. This is a complex problem where due dates have to be quoted in an optimal way. A closer date may give the company a better chance of obtaining an order (i.e. more business at the expense of the company's credibility) while a longer date may give the company enough time reducing the possibility of failing on time (i.e. more credibility less chance of obtaining orders) (Zhang and Wu, 2012). Striking a right balance between these facts require an excellent modelling tool. EDA was used because of its ability to

model the relationship between decision variables. The probabilistic model of the EDA is based on the Bayesian network. To improve the performance of this model-based search algorithm, EDA was hybridized with a local search algorithm (Zhang and Wu, 2012). The EDA proposed in this research is called due date assignment EDA. There is evidence of the use of real data as well as industrial interaction, but no expert evaluation or industrial implementation reported.

EDA was applied to a Hybrid Flow Shop (HFS) problem which is a particular case of the JSSP in (Salhi et al, 2007). HFS is an NP-hard problem commonly experienced in manufacturing firms; it involves processing a set of jobs through a series of stages (Linn and Zhang, 1999). The aim is to minimise a cost function. The distinct feature of HFS is that there is a strict order for all operations required by each job. This problem experienced by a cardboard box company involves processing a set of jobs while taking into consideration the fact that the processing time and the set-up time of the machines vary from one to another. Realistic timing can be produced using previous data. A new type of EDA with guided mutation (EDA-GM) was proposed and used to solve this problem. The type of probabilistic model used is however not mentioned. The authors identified that this research lacked enough data to assess the performance of the proposed heuristic confidently. Although data used originated from a real-world problem, there is no report of any form of industrial involvement or implementation.

Another application of EDA to the JSSP is seen in the case study of a steel industry (Wu and Huang, 2013). This problem is called a single-machine deteriorating jobs scheduling problem which entails real-world factors causing the time required for the same job to differ. The fact that it is often assumed that the time required to process a job is always the same was faulted in this study. In this considered steel industry, the processing time of a job will not always be the same. Also, as noted by Zhang and Wu (2012), the need to accommodate varying due-date was also identified hence the choice of a model-based search algorithm, particularly EDA. This is important because due-dates are usually negotiable and not static like most theoretical previous works assumed. Similar to (Zhang and Wu, 2012), EDA's ability to learn problem structure recurs as the reason for the choice of algorithm. This research work also re-emphasises the need to base the objective function of JSSP on due-date-related performance rather than merely using the makespan (Wu and Huang, 2013; Zhang and Wu, 2012). The Improved EDA (IEDA) is proposed and applied to the JSSP. They recorded feasible results using this approach. Data from a real situation has been used in this research. However, there is no record of any form of interaction with the industry.

The car sequencing problem as presented in (Gagné et al, 2006) is classified as a JSSP and has been solved by an ACO. This problem is seen in an automobile assembly line where the car passes through three main stages of production which are body construction, paint and assembly. The construction and assembly shops have capacity constraints, and there is a need to minimise colour changes in the paint shop. The incorporation of more realistic constraints as well as solving this problem as a multi-objective problem is said to have significantly increased the complexity of the problem. The authors, however, recorded better solution structures and objective function

values when compared to the method used in practice. This was a collaborative research with Groupe Renault and there is evidence of the use of real data, industrial involvement and evaluation. There was, however, no evidence that this became a useful tool for the company.

The JSSP faced by a printing company involves scheduling printing jobs on 18 machines at 7 work centres (Montgomery et al, 2006). Two MAX-MIN ant systems were used to solve this JSSP problem using a month data consisting of 549 operations divided into 159 jobs. Here, execution times of tasks are uncertain. Also, there are no fixed due dates but rather promised dates making the problem less contained by time. The success of ACO in solving other complex real-world problems prompted the authors to try the algorithm on this class of problem. Data was collected from the printing company for the purpose of trying the ACO on the real-world problem. There is however no evidence to suggest industrial evaluation or implementation.

ACO was applied to the JSSP in Chica et al (2011). It was identified that assembly line balancing problems needed to be modelled as a multi-objective problem. This is based on the experience of working on a real-world Nissan assembling problem in Barcelona, Spain. This problem is characterised by two objectives which are minimising the number of stations as well as their area for a given cycle time. It was explained that it is often impossible to combine the objectives into a single objective or to work based on the seemingly most important objective. This collaborative research described in (Chica et al, 2011) has taken some real-world factors into considerations such as the fact that experts should not be presented with too many results but a few that may be of interest to them. The authors also noted the importance of incorporation domain expert knowledge. This study is based on the use of real-world data, industrial involvement and expert feedback. Implementation of this research in the Nissan plant was not reported.

2.4.4 Nurse Scheduling Problem

Nurse scheduling is a complex problem in certain medical practices, the fact that there are a limited number of nurses to cater for patients adequately pose a real challenge (Aickelin and Li, 2007). EDA and ACO have respectively been applied to a nurse scheduling problem originating from a hospital in the United Kingdom (Aickelin and Li, 2007; Aickelin et al, 2006) and a hospital in the United States (Gutjahr and Rauner, 2007).

From the case study of a hospital in the UK (Aickelin and Li, 2007), nurses are usually of different grades, prefer different shifts and have contracted hours. In the cited work, it is important to have a mix of grades of nurses at every given time slot. Also, the schedule should be fair (i.e. an even distribution of nurses). The authors identified the fact that many previous works have suffered from the problem of oversimplification thereby making such works inapplicable in real-world settings. The novel EDA presented by these authors was based on the Bayesian network. They particularly used this approach to enable them to build schedules using flexible rules rather than fixed ones (Aickelin and Li, 2007; Aickelin et al, 2006). Aickelin and Li (2007) obtained feasible solutions using this approach. This is a real-world

problem, there has been interaction with the hospital for a proper understanding of the constraints as well as access to real data. No industrial evaluation or implementation was reported.

Gutjahr and Rauner (2007) applied ACO to the nurse scheduling problem based on a real-world situation in a hospital in Vienna, United States. This research was motivated by the shortage of nurses in hospitals which was associated with poor distribution and utilisation of nurses. This is said to have been caused by a decrease in supply (nurses) and an increase in demand (patients). Similar to Aickelin and Li (2007), they identified that a lot of previous research suffered from problem over-simplification. ACO is seen to be a suitable meta-heuristic because of its ability to construct feasible solutions to highly constrained problems. ACO is able to handle the dynamic nature of this problem. The use of real-world data and engagement with the industry are identified, there is however no report of industrial assessment or implementation.

2.4.5 Other Scheduling Problems

More scheduling problems are considered in this section. An application of EDA to a vehicle charging problem (Su and Chow, 2012) as well as applications of ACOs to a power plant maintenance scheduling problem (Foong et al, 2008) and Aluminium casting scheduling (Gravel et al, 2002) are presented.

Su and Chow (2012) considered the large-scale plug-in hybrid electric vehicle charging problem. This is a real-world problem which considers the need to charge electric vehicles at a municipal parking station. A univariate EDA was used to allocate electrical energy to electric vehicles connected to a grid. The authors considered realistic constraints such as remaining battery capacity, charging time and energy prices. The data used in this research are based on the historical data of an office parking deck in the city of Livermore, CA. The authors explained that EDA provides many advantages over other optimisation techniques including GA. A notable one is that the execution time of running the EDA on large scale problems did not increase exponentially and is better than other optimisation techniques. This is not at the expense of the quality of solutions. They, therefore, recommend EDA for similar large-scale problems. There is no report of expert involvement, evaluation or real-world implementation.

Foong et al (2008) applied ACO to the power plant maintenance scheduling problem in an energy production company. This problem involves determining the optimal maintenance/servicing time for the power generating machines such that the system reliability is maximised. The proposed algorithm relies on data relating to the power demands and storage inflows. Results produced are considered better than those produced by the traditional methods. This study shows that methods developed for theoretically formulated problems can be useful for real-world problems. Foong et al (2008) modified methods developed in their previous research (Foong et al, 2005) to be able to fit to the real case. The use of real-world data and industrial engagement is reported but an assessment from industry experts or industrial implementation cannot be inferred.

Gravel et al (2002) applied ACO to the problem of scheduling continuous casting of aluminium. This is a problem faced in an Alcan aluminium foundry in Québec. Customers place orders of differing alloy types, dimensions and quantity. It is essential to meet customer demands in a timely fashion. However, changing from one alloy type to another is a time-consuming exercise of draining and cleaning to avoid contamination. Also, having a break in production is not desirable. This is because a costly task of shutting down and cleaning is required for operations to resume. The complexity of this problem can be associated with many objectives. These are maintaining continuous operations, minimising changes due to switching to a different alloy specification and respecting the turn around time. The problem here is to determine the processing sequence for a group of jobs. A multi-objective ACO is therefore proposed for this problem. The authors had previously successfully applied a GA to this problem (Gravel et al, 2000) in the same company and lead to deployment within the company. The ACO was then selected. This is because the ACO provided better quality results within a shorter computation time when compared to the GA. The use of real data, engagement with the industry and industrial verification and implementation can be deduced from this research.

2.4.6 Other Applications

Other application areas of EDAs are hospital resource management for a hospital in Netherlands (Hutzschenreuter et al, 2009), Magnetic Resonance Imaging (MRI) magnet design (Yuan et al, 2005), Frequency Assignment Problem of a GSM network (Chaves-Gonzalez et al, 2008) and design and rehabilitation of a water distribution system. ACO was applied to an industrial layout problem in (Hani et al, 2007).

Hospital resource management is another area where EDA has been successfully applied. Similar to (Aickelin and Li, 2007; Aickelin et al, 2006), staff scheduling is a significant part of hospital resource management. However, Hutzschenreuter et al (2009) takes additional resource such as beds into consideration. This involves moving beds from wards that need it less to those that need it more while respecting the number of beds to nurse ratio. The fact that a nurse is often assigned to two beds creates a constraint on the number of beds that can be moved, it has to be an even number. This problem is modelled as a dynamic optimisation problem. There are conflicting objectives of minimising the cost of resources and maximising their services. A combination of approaches which are Standard Deviation Ratio (SDR), Adaptive Variance Scaling (AVS), and Multi-objective mixture-based Iterated Density Estimation Evolutionary Algorithm (MIDEA) called SDR-AVS-MIDEA was applied to the hospital resource management problem. This approach was originally formulated by Bosman and Thierens (2007). Hutzschenreuter et al (2009) were able to get feasible results using this approach. This algorithm is seen to be ideal for this kind of complex multi-objective problem because of its ability to learn problem structure. This entails searching for all (in)dependencies and learning new relationships among problem-specific parameters. The use of realistic data, as well as expert involvement and feedback, were reported. There is, however, no report of implementation in the medical industry.

EDA has also been applied to MRI magnet design (Yuan et al, 2005) which is a useful application in the medical industry. The objective is to reduce the patient's perception of claustrophobia by shortening the MRI magnet design. Also for safety reasons, it is vital to minimise the field outside the MRI system. An EDA based on a Gaussian multivariate probabilistic model named EDAmvg was applied to this problem. There is a constraint on the construction of the MRI stating that coils that make up the MRI cannot overlap or be too close to each other.This factor makes it complex to achieve the somehow conflicting aim of minimising the size of the MRI. Again, EDA's ability to learn complex problem structure surfaces as the reason for the choice of algorithm. Other factors considered important in this study are problem characterisation, dependence capture and using problem-specific knowledge. This study does not suggest the involvement of industry experts or real-world implementation.

A univariate EDA, PBIL was used to solve a Frequency Assignment Problem of a GSM network (Chaves-Gonzalez et al, 2008). This is a real-world GSM network from Denver City, United States. The problem aims to minimise interference from assigning frequencies to transceivers. In this study, the use of the distributed island model applied to PBIL produced better results than the ones obtained with the sequential version. This work was a first attempt at applying PBIL to this problem. The use of real-world data was reported. Expert involvement, expert evaluation or implementation was not reported.

EDA has also been applied to the design and rehabilitation of a water distribution systems (Olsson et al, 2009). This problem involves making a decision that relates to creating a new design or rehabilitating of an existing water distribution network. This problem is full of dependencies and the choice of EDA is based on its ability to perform linkage learning. The authors applied hBOA and UMDA. Although hBOA is a multivariate EDA, the UMDA was reported to produce better results in a realistic computational time. Real-world data was used, but no expert involvement or evaluation was reported. Authors do not report any implementation of method in the industry.

ACO was applied to the industrial layout problem in (Hani et al, 2007). This problem is a type of facility layout problem which seeks to minimise cost and maximise production flow by creating a good configuration of resources in a given facility. This particular industrial layout problem arises from a train maintenance facility consisting of six locations. This problem was modelled as a QAP. Given that QAP is one of the problems the ACO is well suited for, the authors recorded an improvement when compared with the existing mode of operation. In this problem, it was possible to determine the best solution using exact methods. It was therefore noted that the ACO was able to produce the optimal solution to this problem within an acceptable computation time. The authors used real-world data and had some interactions with the industry but did not report any form of industrial verification nor implementation.

2.4.7 Summary

Table 2.1: Classification of real-world applications of model-based search algorithms

Problem	Scheduling Problem	Permutation Component
VRP		✓
TSP		✓
JSSP	✓	✓
Crane Scheduling	✓	✓
Maintenance Scheduling	✓	✓
Aluminum Casting Scheduling	✓	✓
History Matching		
Industrial Layout Problem		✓
Nurse Scheduling	✓	✓
Resource Management		
MRI magnet design	✓	✓
History Matching		
Electric Vehicle Charging	✓	✓
Frequency Assignment Problem		
Water Distribution Problem		

The use of local search is a common concept in both ACO and EDA applications. Montgomery (2007) explains that there is an unavoidable interaction between ACO and local search as the pheromone model is updated based on the local performance. This is consistent with the work of Rizzoli et al (2004) that also emphasises the strong relationship between ACO and local search methods. EDA applications have also used local search to improve their performance (Aickelin and Li, 2007; Aickelin et al, 2006; Zhang and Wu, 2012).

As shown in Table 2.1, 67% of the reviewed applications use the permutation representation, 70% of these are also scheduling problems. The overlap between scheduling and permutation problems is expected as many scheduling problems are naturally represented as permutations. Also, the fact that most of the reviewed problems are scheduling and/or permutation-based problems stems from the wide application areas of these problem classes (Chiong et al, 2012).

In Table 2.2, only about 26% of the reviewed papers reach the stage where their algorithms are integrated and used by a company. For the ACO applications, 7 out of 16 reached this final stage while none of the EDA applications did. This may be attributed to the fact that EDAs had not been sufficiently adapted for the permutation domain. They have therefore for many years struggled to be competitive on permutation problems (Ceberio et al, 2012). EDAs however have the advantage of using explicit probabilistic models which makes it easier to predict movement in a search space (Larrañaga and Lozano, 2002). In general, ACOs use implicit probabilistic models and consequently difficult to predict movement of population in a search space. Ceberio (2014) showed that EDAs can produce better quality solutions, if sufficiently adapted to solve permutation and scheduling problems. We will therefore focus on EDAs, identifying factors that can improve their performance

Table 2.2: Stages of real-world applications of model-based search algorithms

ACO Applications	Real data	Exp. Involv.	Ind. Eval.	Impl.
Rizzoli et al (2007)	✓	✓	✓	✓
Nechita et al (2008)	✓	✓	✓	✓
Rizzoli et al (2004)	✓	✓	✓	✓
Donati et al (2008)	✓			
Gambardella et al (2003)	✓	✓	✓	✓
Pellegrini et al (2007)	✓	✓		
Gagné et al (2006)	✓	✓	✓	
Montgomery et al (2006)	✓	✓		
Chica et al (2011)	✓	✓	✓	
Gutjahr and Rauner (2007)	✓	✓		
Hirsch et al (2012)	✓	✓	✓	✓
Foong et al (2008)	✓	✓		
Gravel et al (2002)	✓	✓	✓	✓
Hajizadeh et al (2011)	✓	✓		
Hani et al (2007)	✓	✓	✓	✓
Ochiai and Kanoh (2014)	✓			
EDA Applications				
Aickelin and Li (2007)	✓	✓		
Hutzschenreuter et al (2009)	✓	✓	✓	
Zhang and Wu (2012)	✓	✓		
Wu and Huang (2013)	✓			
Salhi et al (2007)	✓			
Yuan et al (2005)	✓	✓		
Abdollahzadeh et al (2013)	✓	✓		
Su and Chow (2012)	✓			
Chaves-Gonzalez et al (2008)	✓			
Olsson et al (2009)	✓			

on permutation and scheduling problems.

Although many of the reviewed papers use the permutation representation, some of them use more than one representation. An example is the VRP. The routing part of the problem can be solved using the permutation representation while the process of assigning jobs to fleets could be solved using the integer/vector representation. Problems that require the optimisation of interconnected sub-problems are referred to as multi-component problems. Previous studies suggest that many approaches based on theoretically formulated problems are not directly applicable in the real-world due to oversimplification. (Michalewicz, 2012a) identified the fact that many real-world problems are multi-component problems but many benchmarks are not formulated this way. This motivated recent research on more realistic benchmark problem sets. Some examples are the MRCPSP (Van Peteghem and Vanhoucke, 2014), Travelling Thief Problem (Bonyadi et al, 2013; Polyakovskiy et al, 2014) and Workforce Scheduling and Routing Problem (Castillo-Salazar et al, 2016; Hart et al, 2014). Competitions on multi-component problems have been recurrent in the two largest annual conferences on EAs, "Genetic and Evolutionary Computation Conference" [1] and "IEEE Congress on Evolutionary Computation" [2] [3], in the last five years. To understand how EDAs solve multi-component problems, we will review EDAs applied to MRCPSP later in this chapter. This thesis will therefore explore EDAs in the context of representations that have complexities motivated by real-world problems.

2.5 Common Permutation and Scheduling Problems

In the previous section, the close relationship between permutation and scheduling problems was established. This is consistent with the research of Chiong et al (2012), which shows that many real-world optimisation problems are scheduling problems and are often represented as permutations. It was also noted in (Knjazew and Goldberg, 2000) that there are many commercially interesting applications within the permutation and scheduling domain.

To assess the performance of algorithms, it is common practice to compare results based on benchmark problem sets. Many benchmarks studied in academic research are real-world motivated. Although they are often not perfect fits for real industrial problems, they are useful for comparing the relative performance of algorithms. The review in the previous section also suggests that real-world problems can be solved by adopting approaches proposed for artificial test sets. This section reviews some well-known permutation and scheduling problems which are PFSP, QAP, TSP, LOP, RCPSP and MRCPSP and are formally defined in this section. PFSP, QAP, TSP and LOP are considered the most common theoretically studied permutation problems (Ceberio et al, 2012). RCPSP and its variant MRCPSP have also been well-studied

[1] http://gecco-2017.sigevo.org/index.html/HomePage
[2] https://cs.adelaide.edu.au/ optlog/CEC2014Comp/
[3] https://cs.adelaide.edu.au/ optlog/CEC2015Comp/

because of its real-world relevance. MRCPSP has been of interest because of its multi-component characteristic. Bonyadi et al (2013) explains that most real-world optimisation problems are a combination of more than one optimisation problem. Bonyadi et al (2013) also motivates the need for the community to drive a focus towards these problems. These problems are considered more difficult to solve as the component problems cannot be solved in isolation, i.e. the solutions of one sub-problem affect the other(s). In this thesis, a real-world project scheduling problem which can be modelled as RCPSP/MRCPSP is used as a case study. This also motivates the study of RCPSP and MRCPSP.

This section presents a formal definition of these common permutation and scheduling problems. It also presents common benchmarks for these problems.

2.5.1 Permutation Flowshop Scheduling Problem

The PFSP consists of a set of jobs indexed 1, ..., n to be scheduled on a set of m machines. Each job has m operations to be performed by all m machines where the j^{th} operation of each job must be performed by machine j. A job can have its j^{th} operation performed once its $j-1^{th}$ operation has been completed by machine $j-1$ and machine j is available. The objective of this problem is to find a sequence of jobs that minimises the sum of times that each job remains on the flowshop known as the Total Flow Time (TFT). Another common objective function for this problem is the minimisation of the makespan. Both objective functions are considered in this thesis.

The TFT of an ordering of jobs π is formally defined as follows.

$$F(\pi) = \sum_{i=0}^{n} C_{\pi(i),m} \tag{2.3}$$

$$C_{\pi(i),j} = \begin{cases} P_{\pi(i),j} & i = j = 1 \\ P_{\pi(i),j} + C_{\pi(i-1),j} & i > 1,\ j = 1 \\ P_{\pi(i),j} + C_{\pi(i),j-1} & i = 1,\ j > 1 \\ P_{\pi(i),j} + Max(C_{\pi(i-1),j}, C_{\pi(i),j-1}) & i > 1,\ j > 1 \end{cases} \tag{2.4}$$

The makespan of an ordering of jobs π is formally defined as follows.

$$F(\pi) = C_{\pi(n),m} \tag{2.5}$$

$$C_{\pi(i),j} = max\left\{C_{\pi(i-1),j}, C_{\pi(i),j-1}\right\} + P_{\pi(i),j} \tag{2.6}$$

In eq. (2.3), $C_{\pi(i),m}$ denotes the completion time of a job ranked $\pi(i)$ on machine m and is calculated as shown in eq. (2.4). Similarly, $C_{\pi(n),m}$ in eq. (2.5) denotes the completion time of a job ranked last $\pi(n)$ on machine m and is calculated as shown in eq. (2.6).

In eq. (2.4) and eq. (2.6), the processing time required to perform a job ranked $\pi(i)$ on machine j is denoted by $P_{\pi(i),j}$.

2.5.2 Quadratic Assignment Problem

Given a set of locations, a set of facilities and an associated cost function, the QAP entails assigning each facility to a location such that the cost is minimised. The cost is calculated based on the distance between locations and the flow between facilities. Typically, there are two square matrices of size n: H and D. $h_{a,b}$ is the flow between facilities a and b . d_{l_a,l_b} is the distance between locations l_a and l_b.

The QAP can be formerly defined as follows.

$$min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} h_{a,b} \times d_{l_a,l_b} \right\} \tag{2.7}$$

2.5.3 Travelling Salesman Problem

Given a set of cities $c = c_1, c_2, ..., c_n$, the objective is to find the shortest path between all these cities. Solutions should involve a single visit to each city and a return to the start city.

The TSP can be formerly defined as follows.

$$min \left\{ \sum_{i=2}^{n} d_{c_{i-1},c_i} + d_{c_n,c_1} \right\} \tag{2.8}$$

In (2.8), d_{c_{i-1},c_i} is the distance between cities c_{i-1} and c_i.

2.5.4 Linear Ordering Problem

Given a square matrix B of size n, the aim is to find a permutation of the rows and columns of D such that the sum of the super-diagonal entries is maximised.

The LOP can be formerly defined as follows.

$$max \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} B_{\omega_i \omega_j} \right\} \tag{2.9}$$

In (2.9), ω_i and ω_j are respectively the row and column indexes at positions i and j of solution ω.

2.5.5 Resource Constrained Project Scheduling Problem

The RCPSP is formerly defined as follows:

A project consists of a set of n activities. Every activity i is labelled from 1,...,n. Activity $i, i \in [2, n]$ has a set of predecessors $Pred_i$ which suggests that activity i cannot be performed until every predecessor $h, h \in Pred_i$ has been completed. Given that there are A renewable resources, each renewable resource r, $r \in [1, |A|]$ is available per period of time. The maximum per period availability of r is denoted by αmax_r. Each activity i requires a set of renewable resources $(\alpha_{i,1}, ..., \alpha_{i,|A|})$, and an associated duration/execution time t_i.

The aim of the RCPSP is to order activities subject to precedence constraints such that makespan is minimised.

We formulate the RCPSP as follow.

Minimise ft_n subject to:

$$\forall\, i \in [1,n],\ st_i \geq ft_h\ \forall\, h \in Pred_i \tag{2.10}$$

Let C_p be the set of activities being executed during time period [p-1,p], then

$$\textstyle\sum_{i \in C_p} \alpha_{i,r} \leq \alpha max_r\ \forall\, r, r \in [1,|A|], \forall\, p \tag{2.11}$$

We denote the start and finish times of activity i by st_i and ft_i respectively. The precedence and resource constraints are respectively presented in (2.10) and (2.11).

2.5.6 Multi-Mode Resource Constrained Project Scheduling Problem

MRCPSP is a generalisation of the well-known RCPSP. In addition to activity scheduling of the RCPSP, the MRCPSP also consists of the mode assignment problem. MRCPSP is formerly defined as follows:

A project consists of a set of n activities. Every activity i is labelled from 1,...,n. Activity $i, i \in [2,n]$ has a set of predecessors $Pred_i$ which suggests that activity i cannot be performed until every predecessor $h, h \in Pred_i$ has been completed. Activity i must be performed in a mode $k \in [1, m_i]$, where m_i is the number of possible modes of i. Given that there are A renewable resources, each renewable resource r, $r \in [1,|A|]$ is available per period of time. The maximum per period availability of r is denoted by αmax_r. Apart from renewable resources, there are also B non-renewable resources that cannot be renewed but available for the entire project duration. The overall availability of the non-renewable resource l, $l \in [1,|B|]$ is denoted by βmax_l. Each mode of execution k of an activity i is composed of an integer vector of renewable resources $(\alpha_{i,k,1}, ..., \alpha_{i,k,|A|})$, an integer vector of non-renewable resources $(\beta_{i,k,1}, ..., \beta_{i,k,|B|})$ and the associated duration/execution time $t_{i,k}$.

The aim of the MRCPSP is to select exactly one mode of execution for each activity subject to resource and precedence constraints. This is such that makespan is minimised. We formulate the MRCPSP as follow.

Minimise ft_n subject to:

$$\forall\, i \in [1,n],\ st_i \geq ft_h\ \forall\, h \in Pred_i \tag{2.12}$$

Let C_p be the set of activities being executed during time period [p-1,p], then

$$\textstyle\sum_{i \in C_p} \alpha_{i,k_i,r} \leq \alpha max_r\ \forall\, r, r \in [1,|A|], \forall\, p \tag{2.13}$$

$$\sum_{i=1}^{n} \beta_{i,k_i,l} \leq \beta max_l\ \forall\, l, l \in [1,|B|] \tag{2.14}$$

The start and finish times of activity i are denoted by st_i and ft_i respectively. The precedence constraint is presented in (2.12) while the renewable and non-renewable resource constraints are respectively presented in (2.13) and (2.14). In (2.13) and (2.14), k_i is the allocated mode of i and can only be one of the predefined modes of i. Also, $\alpha_{i,k_i,r}$ and $\beta_{i,k_i,l}$ are respectively the amount of renewable resource r and non-renewable resource l required by activity i performed in mode k_i.

2.5.7 Benchmarks

Taillard (1993) provides benchmark for some scheduling problems such as PFSP, JSSP and open shop scheduling problems.

1. **PFSP:** Taillard's benchmark problems (Taillard, 1993) are the conventional problem sets used to evaluate the performance of algorithms on the PFSP. The benchmark consists of varying number of jobs and machines denoted by $n \times m$, where n is the number of jobs to be scheduled on m machines.

   The following are the available problem sets.

   - Size 20: 20×5, 20×10, 20×20,
   - Size 50: 50×5, 50×10, 50×20,
   - Size 100: 100×5, 100×10, 100×20,
   - Size 200: 200×10, 200×20 and
   - Size 500: 500×20

   Furthermore, each set consists of 10 problem instances.

   A similar benchmark set was created in (Ceberio et al, 2014a) because the authors wanted to access performance between 200 and 500 jobs. They therefore created instances with 250, 300, 350, 400 and 450 jobs using 10 and 20 machines. Only the Taillard instances[4] are however used in this thesis.

2. **QAP:** Some of the most widely studied QAP instances are the Taillard's benchmark problems[5]. Problem categories such as 'uniform', 'asymmetrical', 'Burkard and Offerman' as well as 'Nugent Vollmann and Ruml' were used to compare EDAs in (Ceberio et al, 2012). The *tai15a, tai15b, tai40a* and *tai40b* from the Taillard's 'uniform' and 'asymmetrical' groups were used to compare algorithms in (Regnier-Coudert and McCall, 2014). These problem instances are used in this thesis.

3. **TSP:** Reinelt (1995) presents common TSP instances. These instances are available from the TSPLIB[6]. In this thesis, the *bays29, berlin52, dantzig42*

[4]Éric Taillards web page. http://mistic.heig-vd.ch/taillard/problemes. dir/ordonnancement.dir/ordonnancement.html

[5]Éric Taillards web page. http://mistic.heig-vd.ch/taillard/problemes. dir/qap.dir/qap.html

[6]TSPLIB. http://www2.iwr.uni-heidelberg.de/groups/comopt/software /TSPLIB95/tsp

and *fri26* problem instances are used as done in (Ceberio et al, 2012; Regnier-Coudert and McCall, 2014).

4. **LOP:** The conventional problem sets for LOPs are presented in (Martí et al, 2012) and are obtainable from LOLIB[7]. Some of the common instances are *t65b11, be75np* and *be75oi* which are also used in (Ceberio et al, 2012; Regnier-Coudert and McCall, 2014).

5. **RCPSP:** RCPSP instances are available from the PSPLIB[8]. This library consists of four problem sets; J30, J60, J90 and J120 and are respectively composed on 30, 60, 90 and 120 non-dummy activities to be scheduled. Each project has two additional activities which are the dummy start and finish activities. The start activity has no predecessor while the finish activity has no successor, they also have no resource requirements. J30-90 consists of 480 problem instances while J120 consists of 600 instances. Only J30, J90 and J120 are considered in this thesis are they are the frequent choice in existing research (Debels and Vanhoucke, 2005; Fang and Wang, 2012; Wang and Fang, 2012b)

6. **MRCPSP:** The PSPLIB[9] also consists of MRCPSP problem sets which are J10, J12, J14, J16, J18, J20 and J30. These problems sets respectively consist of 10, 12, 14, 16, 18, 20 and 30 non-dummy activities to be scheduled. They also respectively contain 536, 547, 551, 550, 552, 554 and 552 feasible instances. More recently, certain disadvantages have been identified with the PSPLIB problem sets. One is the fact that modes of execution can be eliminated by executing the conventional preprocessing technique, simplifying the problems significantly. Also, some of the instances do not have any feasible solution. To avoid these disadvantages, MMLIB[10] problem sets which also consist of larger problem instances were created (Van Peteghem and Vanhoucke, 2014). The MMLIB50 and MMLIB100 which are made up of 540 instances and respectively require the scheduling of 50 and 100 activities are considered in this thesis. All PSPLIB instances are also used in this thesis.

2.5.8 Performance Measure

For common permutation and scheduling problems, the conventional measure of performance is the Average Relative Percentage Deviation (ARPD) from optimal where an optimum or a standard best solution is known. This is based on a maximum number of fitness evaluations or schedule generations.

$$ARPD = (\sum_{i=0}^{n} \frac{(Algorithm_i - Best) \times 100}{Best})/n \tag{2.15}$$

[7]https://www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/iomat/
[8]http://www.om-db.wi.tum.de/psplib/getdata.cgi?mode=sm
[9]http://www.om-db.wi.tum.de/psplib/getdata.cgi?mode=mm
[10]http://mmlib.eu

In eq. (2.15), n is the number of runs and $Algorithm_i$ is the result derived by an algorithm at the i^{th} run. $Best$ is the optimal or best known solution for a given problem instance.

2.6 EDAs for Permutation and Scheduling Problems

This section presents a characterisation of EDAs applied for solving the common permutation problems; LOP, QAP, PFSP and TSP. It also presents a review of EDAs applied for solving project scheduling problems; RCPSP and MRCPSP.

2.6.1 EDAs for Common Permutation Problems

Modelling the space of permutations is generally considered a difficult task and has been a challenging area for EDAs (Ceberio et al, 2012). This was attributed to the fact that a large proportion of EDAs were based on concepts borrowed from the integer and continuous optimisation domains rather than using characteristics of the permutation space. However, a few EDAs are specifically designed for solving permutation problems taking into consideration the unique properties of permutations. Ceberio et al (2012) categorised EDAs for solving permutation problems into integer-based, continuous and permutation-specific.

When solving integer problems, many genes can have the same value. EDAs adapted to the permutation domain from the integer domain therefore require a method to ensure each gene has a unique value. This is often done by changing the sampling strategy or introducing a repair mechanism (Pelikan et al, 2007). Such mechanisms may however disrupt the structure already learnt by the EDA. Also, the probabilistic models of integer-based EDAs are often large. A classic example of an integer-based EDA is the Univariate Marginal Distribution Algorithm (UMDA) (Larrañaga et al, 2000). Based on the more promising solutions of a population, this algorithm generates the probability of each value being in each position. For a problem of size n, the size of its probabilistic model is therefore $n \times n$. In general, the time complexity of learning and building its model grows to the square of size of the problem ($O(n^2)$). Other examples of EDAs adapted from the integer domain are dependency tree EDA (Pelikan et al, 2007), Estimation of Bayesian Network Algorithm (EBNA) (Bengoetxea et al, 2002), and Mutual Information Maximization for Input Clustering (MIMIC) (Bengoetxea et al, 2002).

EDAs adapted from the continuous domain often use the RK representation. These EDAs require a de-codification step to convert the real-valued solution into a permutation. RKs have an advantage over the integer-based representation as they always produce valid permutations. EDAs that use the RK representation also use a relatively smaller models, the time complexity of which grows linearly with the size of a problem. An example is UMDA$_c$ (Lozano and Mendiburu, 2002) which uses a Gaussian distribution. It saves the mean of all genes as well as the associated variance

values. The size of the probabilistic model is therefore $2n$. Its overall time complexity for learning and sampling its model grows linearly with the size of the problem. Irrespective of the benefits of RK based EDAs, they have struggled to produce competitive results. The representation contains some inherent redundancies resulting from several RKs producing the same ordering thereby introducing plateaux to the search space (Bosman and Thierens, 2001; Pelikan et al, 2007). Also, this variability in the values that represent the same priority across solutions of a population limits the information captured by the probabilistic model. Other examples of EDAs adapted from the continuous domain are Estimation of Gaussian Network Algorithm (EGNA) (Lozano and Mendiburu, 2002) and MIMIC$_c$ (Larrañaga et al, 2000).

Furthermore, Recursive Estimation of Distribution Algorithm (REDA) (Romero and Larrañaga, 2009) and IDEA-Induced Chromosome elements Exchanger (ICE) (Bosman and Thierens, 2001) that are adapted from the continuous domain have been categorised as permutation-specific EDAs (Ceberio et al, 2012). REDA uses the triangulation of Bayesian network approach and focuses on model efficiency by modelling subset nodes of a problem. IDEA-ICE uses a specialised crossover operator which tries to preserve building blocks in the probabilistic model learnt. These algorithms perform better than other algorithms adapted from the continuous domain. However, they are not as competitive as other permutation-specific EDAs. These algorithms fail to address the redundancies introduced by the RK representation.

Some of the EDAs originally designed for permutation use histogram models such as the Node Histogram-Based Sampling Algorithm (NHBSA) (Tsutsui et al, 2006) and Edge Histogram-Based Sampling Algorithm (EHBSA) (Tsutsui, 2002), Plackett-Luce model (Ceberio et al, 2013), Factoradic-based Model (Regnier-Coudert and McCall, 2014) or Mallows Model such as GM-EDA and HGM-EDA (Ceberio et al, 2014a). These models which are more specific to permutations have shown better performances. GM-EDA is one of the most successful algorithms in this category. The process of learning its probabilistic model entails calculating the average of each gene as well as the dispersion. This process is considered analogous to that of the Gaussian distribution in continuous EDAs (Ceberio et al, 2014a). However, the average is not sufficient in this case. The average is converted to a unique permutation using an iterative process of selecting the gene with the minimum value that has not been selected yet. In GM-EDA, the time complexity of learning the model is considered less substantial compared to that of sampling it which is estimated to be $O(n^2)$. Also, GM-EDA is not able to maintain diversity in its population of solutions but relies on the use of restart mechanisms. It particularly presents competitive results only when hybridised with local search (HGM-EDA).

The challenges of solving problems naturally represented as permutations by Estimation of Distribution Algorithms (EDAs) therefore remains a focus of interest in the community. The focus of the research in this thesis will be on eliminating redundancy in the RK representation and introducing a diversity preservation technique.

2.6.2 EDAs for Project Scheduling

In this section, we will be reviewing existing EDAs applied for solving the RCPSP and MRCPSP. It is however important to present some of the conventional procedures used when applying meta-heuristics to project scheduling problems.

The Schedule Generation Scheme (SGS) is a conventional procedure for most heuristic solution to a project scheduling problem. This is a step-wise procedure that builds a schedule from a schedule representation by activity incrementation (serial) or time incrementation (parallel) (Kolisch and Hartmann, 1999). The parallel SGS is however less common because it is sometimes unable to reach optimal (Kolisch, 1996). The SGS was proposed for the RCPSP but has been extended to the MRCPSP and is the conventional approach for generating schedules in both problems.

It is common practice to improve the SGS with the forward-backward improvement procedure when solving the RCPSP. The forward-backward improvement procedure consists of an iterative process of executing the forward pass and backward pass. The forward pass is applied to a backward scheduled solution while the backward pass is applied to a forward scheduled solution. The forward pass sorts all scheduled activities in ascending order of their start times. Each activity is then rescheduled at the earliest possible start time. Conversely, the backward pass sorts activities in descending order of finish times. Each activity is then rescheduled such that they finish at the latest possible time. The aim of this procedure is to improve the makepsan of a solution.

To the best of our knowledge, there is only one application of EDA to the RCPSP, outwit the work presented in this thesis. Wang and Fang (2012b) proposed HEDA which combines EDA with a local search. This algorithm uses the serial SGS improved by the forward-backward procedure. Even with the improvement procedures, the EDA struggle to produce competitive results.

It is common practice to use even more improvement procedures when solving the MRCPSP. They can be categorised into *initial improvement* and *schedule improvement*. The *initial improvement* procedures are only executed once before an algorithm is applied to an MRCPSP instance while *schedule improvement* procedures are applied to each solution through out the run of an algorithm. The *schedule improvement* procedures therefore require more computation.

The pre-processing procedure of Sprecher et al (1997) is a conventional *initial improvement* method, which reduces the search space by eliminating redundant and inefficient modes of execution. Another common *initial improvement* procedure is the feasibility improvement procedure which is executed on the initial population of solutions. The feasibility improvement procedure swaps the mode of execution of a randomly selected activity to that which improves the feasibility of the mode solution. This process is repeated until feasibility is achieved or a maximum number of evaluations is reached.

The most frequently used *schedule improvement* procedure is the forward-backward improvement which is adapted from that of the RCPSP. It however performs an additional step of changing the mode of activities to that which improves the start time (forward pass) or finish time (backward pass) of an activity. The forward-backward

improvement ensures that the makespan of a schedule is improved. It is however the most time consuming schedule improvement procedure because of the number of iterative steps. A more efficient *schedule improvement* procedure is the Extended Schedule Generation Scheme (ESGS) (Van Peteghem and Vanhoucke, 2010) which attempts to improve feasibility as well as the finish time of a randomly selected activity. Unlike the forward-backward schedule improvement, the ESGS does not try to improve every solution neither does it guarantee an improvement in makespan. It is however less time consuming.

Other improvement procedures are critical path improvement and work content improvement methods of Van Peteghem and Vanhoucke (2011). In general, algorithms that use more improvement procedures to solve the MRCPSP perform better. However, since our research interest is in EDAs, we focus on improvement procedures that are used in EDAs.

There are only two applications of EDA to the MRCPSP to the best of our knowledge which are presented in (Wang and Fang, 2012a) and (Soliman and Elgendi, 2014). The latter is an improvement of the former.

These EDAs use two probabilistic models to generate an MRCPSP solution. For a problem consisting of n activities and m modes of execution, the probabilistic model for generating activity solutions is of size $n \times n$ while that of mode solutions is of size $n \times m$. Since the probabilistic model for generating activity solution is adapted from the integer domain, the algorithm requires a procedure to ensure only valid permutations are produced. The permutation-based Probability Generation Mechanism (PGM) is used to ensure that that each activity appears in an ordering only once (Wang and Fang, 2012a). This method iteratively recalculates probabilities based on a set of eligible activities.

Furthermore, existing applications of EDAs to the MRCPSP use the improved serial SGS, Multi-mode Serial Schedule Generation Scheme (MSSGS). The MSSGS includes a procedure for tackling infeasibility as well as improving the finish times of activities in a solution. The method randomly selects the mode of an activity and changes it to that which improves its feasibility if the solution is infeasible or improves the finish time of an activity if the solution is feasible. It does this by changing its mode to that which improves its finish time without delaying the finish time of other activities. Checking each mode of execution of an activity in a bid to improve the solution imposes a significant addition to the computational cost of evaluating a solution. In addition, these EDAs also use the local search method called Multi-mode version Permutation-Based Local Search (MPBLS). This method attempts to make local improvements to the best solutions found at each generation. This is done by changing the mode of randomly selected activities as well as its priority.

In addition to all these methods, the EDA in Soliman and Elgendi (2014) introduced a random walk local search. The use of random walk is the difference between the two EDAs. For a selected activity, the local search selects a mode with the minimum resource infeasibility in infeasible solutions or mode with minimum duration in resource feasible solutions. Adding this extra local search method improves the quality of solutions produced by the former.

The reviewed algorithms rely on many improvement procedures. Each improve-

ment procedure incurs additional computational cost. For EDAs to efficiently solve this multi-component problem, the number of improvement procedures need to be reduced. This thesis will investigate methods of improving performance while reducing the number of improvement procedures.

2.7 Conclusion

This chapter presents a review of real-world applications of EDAs. Since ACOs and EDAs are often studied together within the context of model-based search algorithms, real-world applications of ACOs are also presented. This study shows that many real-world applications of ACOs and EDAs are scheduling problems and use the permutation representation. Thereafter, this chapter motivates the focus on EDAs. Common probabilistic models used in EDAs such as Gaussian Network, Bayesian Network and Markov Network are explained.

EDAs have been considered to have difficulty in modelling permutation spaces in (Ceberio et al, 2012). We therefore investigated further the reasons for this difficulty. We explored the benefits of EDAs that use the RK representation to solve permutation problem. They use smaller models and also always produce valid permutations. However, such EDAs suffer from the redundancies in the RK representation. Also, EDAs for solving permutation problems have no mechanism of preserving diversity except when hybridised with local search. There is therefore a need for a diversity preservation method in EDA. Also, algorithms have become increasingly complex especially when solving more complex problems like multi-component problems. This is seen in the review presented in Section 2.6.2 where it is shown that EDAs rely on many local improvement procedures to solve the project scheduling problems. Since many real-world problems are multi-component problems, EDAs must be able to handle this complexity. It is evident that EDAs need to be further investigated for solving permutation and multi-component scheduling problems.

This thesis will therefore investigate efficient and effective strategies for the EDA within the context of permutation and scheduling problems. We will focus on eliminating redundancies in the RK representation and improving diversity in EDAs. We will also focus on reducing the number of local improvement procedures needed to solve the multi-component problem, MRCPSP. Proposed approach for common permutation and scheduling problems will be adapted for a real-world project scheduling problem.

Chapter 3

RK-EDA: A Novel Random Key Based Estimation of Distribution Algorithm

3.1 Introduction

EDAs for permutation problems have been a focus of research in recent years. Ceberio et al (2012) identified that EDAs adapted from other domains do not perform well on permutation problems. New permutation-specific models have been designed since then but require iterative procedures where the range of valid values to sample in the model depends on previously generated values. Complexity of these permutation-specific models is often the main issue. The need to design EDAs based on simpler models remains a problem in the community. Permutation problems are sometimes solved using alternative representations such as RK. These naturally translate into valid permutations. Also, EDAs based on RK representation often use probabilistic models of smaller dimension growing linearly with the size of the problem. However, in the RK representation, multiple genotypes generate identical phenotypes.

Furthermore, permutation EDAs suffer from the problem of premature convergence. In previous research, techniques such as restart and hybridisation with local search have been used to avoid this problem (Ceberio et al, 2014b).

This chapter answers the first two research questions of this thesis which are; "In what way can the problem of redundancy in RK be addressed?" and "In what way can diversity be controlled in EDAs designed to solve permutation problems?". In this research, the redundancy attributed to the RK representation is eliminated by rescaling solutions to a common canonical value set before modelling. A cooling scheme is also proposed to balance exploration and exploitation. These methods differentiate the proposed RK-EDA from existing EDAs applied to permutation problems. RK-EDA outperforms existing EDAs that use RKs on common permutation test problems: PFSP, QAP, LOP, and TSP. RK-EDA is also competitive with state-of-the-art algorithms, especially on the PFSP.

The rest of this chapter is described as follows. Section 3.2 describes the novel al-

gorithm, RK-EDA. Section 3.3 presents the experimental design. Section 3.4 presents and discusses results. Section 3.5 concludes this chapter.

3.2 RK-EDA

In this section, the algorithmic details of RK-EDA are presented as Alg. 1. RK-EDA requires the initialisation of three parameters which are initial variance v, truncation size b and population size ps. It is common practice to compare algorithms using a maximum number of fitness evaluations (fe). We estimate the number of generations gen by dividing fe by ps.

Algorithm 1 RK-EDA

1: Initialise v, b, ps and fe
2: Generate initial population P of size ps
3: $gen = fe \backslash ps$
4: **for** $g = 1$ to gen **do**
5: Evaluate individuals in P
6: Rescale $\{rk_1, \cdots, rk_n\}$ of individuals in P
7: Select best $b < ps$ solutions to form S
8: Calculate $\mu = \{\mu_1, \mu_2, \cdots, \mu_n\}$
9: $c = 1 - \frac{g}{gen}$
10: $v_g = v * c$
11: Set $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$ as v_g
12: $M = N(\mu, \sigma)$
13: $P_{new} = \emptyset$
14: **repeat**
15: Sample M to generate offspring *off*
16: Add *off* to P_{new}
17: **until** $|P_{new}| = ps$
18: $P = P_{new}$
19: **end for**

A population P of RKs is randomly generated, evaluated and rescaled. For the rescaling procedure to be carried out, the RKs are converted to ranks as illustrated in Fig. 3.1 for the RKs [0.12, 0.57, 0.23, 0.25, 0.99]. The ranks are then rescaled to values in interval [0,1]. This is done by setting $rescaledRK_i = \frac{rank_i - 1}{n-1}$ where $rescaledRK_i$ and $rank_i$ are respectively the rescaled RK and rank of gene i, and n is the problem size. Thus, all values of $rescaledRK$ belong to the canonical set of n values $\{0, 1/n-1, \cdots, i/n-1, \cdots, 1\}$. Continuing with our example, a distinct but equivalent set of RKs [0.01, 0.06, 0.03, 0.04, 0.2] will have the same rescaled RK values. This approach eliminates redundancy improving information captured by the probabilistic model.

Once rescaled and evaluated, the best b solutions in P are selected to generate a population of promising solutions S. The truncation size b is set as a fraction of

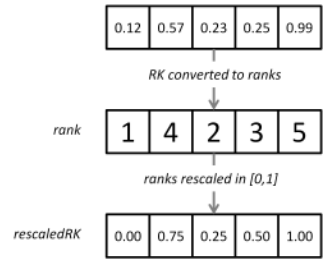

Figure 3.1: RK rescaling

ps. The mean of all RKs at each index $\{1, \cdots, n\}$ is computed from S. μ is a vector containing the mean values for each index ($\{\mu_1, \mu_2, \cdots, \mu_n\}$).

Furthermore, the cooling rate c is calculated with respect to the algorithm's current generation such that its value is higher at the start of the search and low at the end. The rate c is used to calculate the generational variance v_g. Multiplying c with v to form v_g makes it possible to achieve higher exploration at the start of the run and more exploitation as g increases.

The probabilistic model M is defined as a normal distribution $N(\mu, \sigma)$ where μ and σ are vectors of size n. σ contains identical values σ_g. We therefore only save $n + 1$ values at each generation.

An offspring solution *off* is generated by sampling M. Each gene off_i is generated by sampling $N(\mu_i, \sigma_g)$. *off* is repeatedly added to the offspring population P_{new} until its size equals ps. At the end of each generation, P_{new} completely replaces the parent population P. Note that a solution value off_i obtained by sampling M may be outside the $[0, 1]$ interval. This does not prevent ranking or rescaling.

3.3 Experimental Settings

To assess the performance of RK-EDA and compare with existing EDAs for permutation problems, we select the most frequently used permutation problem instances. These instances are presented in this section. We also present the parameter settings and experimental approach.

3.3.1 Test Problems

In Chapter 2.5.7, we presented standard benchmarks for permutation problems. Ceberio et al (2012) used instances of these benchmarks to compare EDAs in their review. Some of these instances were also used by Regnier-Coudert and McCall (2014). In this chapter, we used instances common to both research. TSP and LOP instances are respectively from the TSPLIB (Reinelt, 1991) and LOLIB (Reinelt, 2002) while the PFSP and QAP instances are from the well-known Taillard's benchmarks (Taillard, 1993). However, the PFSP instances common to the reviewed algorithms are all of the smallest dimension ($n = 20$). We therefore added four larger instances ($n = 50, 100$) of the PFSP to gain better insight into the scalability of RK-EDA.

The list of problem instances used in this chapter is presented as follows.

- TSP: *bays29, berlin52, dantzig42* and *fri26*[1]
- PFSP: *tai20-5-0, tai20-5-1, tai20-10-0* and *tai20-10-1* (smaller instances) *tai50-10-0, tai50-10-1, tai100-20-0* and *tai100-20-1* (larger instances)[2]
- QAP: *tai15a, tai15b, tai40a* and *tai40b*[3]
- LOP: *t65b11, be75np* and *be75oi*[4]

3.3.2 Parameter Setting

To be able to understand the parameter settings that suit RK-EDA, a range of values were explored, but different parameters were found suitable for different problem classes and sizes. To be able to make a fair comparison between RK-EDA and the considered algorithms, the same set of parameters is used across all problems as done in the review of Ceberio et al (2012). The set of parameters used for RK-EDA is shown in Table 3.1. Based on preliminary tests, these parameters produce relatively good quality solutions across all problem classes and instances.

Table 3.1: Parameter Values for RK-EDA

Parameters	**Values**
Population size (ps)	50
Truncation size (b)	$0.1*ps$
Variance (v)	$(1/(\pi log_{10} n))^2$
Stopping criteria	$1000n^2\ fe$
Maximum number of generations (gen)	$20n^2$
Number of runs	10

3.3.3 Experimental Approach

We present problem definitions and objective functions for the TSP, PFSP, QAP and LOP in Section 2.5. Note that we presented two objective functions for the PFSP which are makespan and total flow time. We however use the makespan in this chapter. This enables us to compare using the same objective as the reviewed methods.

The results used in this study are obtained from the review of Ceberio et al (2012). Results of the algorithm presented after the review such as the Factoradics

[1]TSPLIB. http://www2.iwr.uni-heidelberg.de/groups/comopt/software /TSPLIB95/tsp

[2]Éric Taillards web page. http://mistic.heig-vd.ch/taillard/problemes. dir/ordonnancement.dir/ordonnancement.html

[3]Éric Taillards web page. http://mistic.heig-vd.ch/taillard/problemes. dir/qap.dir/qap.html

[4]https://www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/iomat/

are obtained from Regnier-Coudert and McCall (2014). We, therefore, use the same performance measure, number of runs and stopping criteria. *ARPD* presented in Eq. (2.15), is the normalised difference between the result obtained by an algorithm and the best-known solution. *ARPD* is the most frequently used measure of performance and has been used by the algorithms considered in this chapter. Also, results for these algorithms are based on 10 runs and $1000n^2$ maximum number of fitness evaluations.

To compare results, we rank algorithms according to their performances on each problem instance. These ranks are then averaged by problem type.

3.4 Results and Discussion

In this section, we do an empirical evaluation of RK-EDA, comparing it to other EDAs applied to TSP, PFSP, QAP and LOP problem instances.

Tables 3.2, 3.3, 3.4, 3.5 and 3.6 respectively present ARPDs for algorithms applied to instances of the TSP, PFSP (smaller instances), PFSP (larger instances), QAP and LOP. In these tables, the best ARPD for each problem instance, as well as ARPDs that are not significantly different from it, are presented in bold. The student t-test with 95% confidence interval is used to measure statistical significance in these tables. The best-known fitnesses for instances of each problem are presented in the last row of each table. omeGA which is a GA was included in the review of EDAs in (Ceberio et al, 2012) to show the relative performance of EDAs in general compared to a GA. We also include the results of omeGA in this chapter.

As shown in Table 3.2, EHBSA$_{WT}$ has the best ARPD on all the TSP problem instances while EHBSA$_{WO}$ also presents the best result on *fri26*. UMDA$_c$ and EGNA$_{ee}$ also present best ARPD on *dantzig42*.

In Table 3.3, RK-EDA presents the best ARPD on *tai20-5-0* and *tai20-5-1*. EHBSA$_{WT}$ also presents the best ARPD on *tai20-10-0* and its ARPD is not significantly different from the best on *tai20-5-1* and *tai20-10-1*. NHBSA$_{WT}$ presents the best ARPD on *tai20-10-1* and its ARPD is not significantly different from the best on *tai20-10-0*. As shown in Table 3.4, RK-EDA presents the best ARPD on all the larger instances of PFSP (*tai50-10-0*, *tai50-10-1*, *tai100-20-0* and *tai100-20-1*) while the ARPD of EHBSA$_{WT}$ on *tai50-10-0* is not significantly different from that of RK-EDA. We do not present results for factoradics in Table 3.4 because the authors do not present results for these instances.

For the QAP results presented in Table 3.5, NHBSA$_{WT}$ presents the best ARPD on three of the instances (*tai15a*, *tai15b* and *tai40b*). RK-EDA also presents ARPD that is not significantly difference from that of NHBSA$_{WT}$ on *tai40b* while NHBSA$_{WO}$ presents the best ARPD on *tai40a*.

In Table 3.6, EHBSA$_{WT}$ presents the best ARPD on *t65b11xx* and *be75np* while NHBSA$_{WT}$ presents the best ARPD on *be75oi*. Note that, unlike TSP, PFSP and QAP, LOP is a maximisation problem.

In Table 3.7, an average rank is calculated for each algorithm according to their relative performance on each problem instance of TSP, PFSP$_s$, QAP, LOP and PFSP$_l$. Results from Tables 3.2, 3.3, 3.5, 3.6 and 3.4 are respectively used to generate the

Table 3.2: Travelling Salesman Problem

Algorithms	bays29	berlin52	fri26	dantzig42
RK-EDA proposed	2041.5	8404.6	949.5	771.2
Factoradics (Regnier-Coudert and McCall, 2014)	2387.4	11440.7	982.3	805.3
UMDA (Larrañaga et al, 2000)	2324.5	10059.9	1085.2	949.0
MIMIC (Bengoetxea et al, 2002)	2467.6	10921.3	1156.1	1034.9
EBNA$_{BIC}$ (Bengoetxea et al, 2002)	2467.5	9893.8	1094.9	935.1
TREE (Pelikan et al, 2007)	2818.8	12846.8	1280.8	1206.9
UMDA$_c$ (Lozano and Mendiburu, 2002)	4305.1	20460.2	1076.5	**699.0**
EGNA$_{ee}$ (Lozano and Mendiburu, 2002)	4235.3	18728.1	1024.9	**699.0**
IDEA-ICE (Bosman and Thierens, 2001)	3269.1	15813.6	1130.7	947.7
EHBSA$_{WT}$ (Tsutsui, 2002)	**2020.0**	**7542.0**	**937.0**	**699.0**
EHBSA$_{WO}$ (Tsutsui, 2002)	2022.5	7584.6	**937.0**	701.8
NHBSA$_{WT}$ (Tsutsui et al, 2006)	2321.8	11729.7	1041.5	1052.9
NHBSA$_{WO}$ (Tsutsui et al, 2006)	2177.8	9195.8	1048.3	854.0
REDA$_{UMDA}$ (Romero and Larrañaga, 2009)	4064.9	21875.9	1118.4	1750.0
REDA$_{MIMIC}$ (Romero and Larrañaga, 2009)	3316.1	15850.3	1023.6	705.0
omeGA (Knjazew and Goldberg, 2000)	3353.5	18624.0	1461.4	1393.6
Best Known (Ceberio et al, 2012)	*2020.0*	*7542.0*	*937.0*	*699.0*

average ranks presented in columns TSP, PFSP$_s$, QAP, LOP and PFSP$_l$ of Table 3.7 .

In Table 3.7, PFSP$_s$ and PFSP$_l$ respectively denote smaller and larger instances of the PFSP. Columns TSP, PFSP$_s$, QAP, LOP and PFSP$_l$ show the average ranks of algorithms on instances of their respective problem classes. Also, Column ALL is the average rank of algorithms on all instances of TSP, PFSP$_s$, QAP and LOP. Note that instances of PFSP$_l$ are not used to calculate overall ranks. This is done to eliminate bias towards performance on PFSP, especially because RK-EDA ranks highest on PFSP$_l$. Moreover, one of the reviewed algorithms was not applied to instances of PFSP$_l$. It will, therefore, be impossible to generate an overall rank for that algorithm using results on instances of PFSP$_l$.

Table 3.3: Permutation Flowshop Scheduling Problem (Smaller Instances)

Algorithms	tai20-5-0	tai20-5-1	tai20-10-0	tai20-10-1
RK-EDA (proposed)	**1278.1**	**1359.5**	1602.9	1685.2
Factoradics (Regnier-Coudert and McCall, 2014)	1291.7	1364.4	1630.4	1723.9
UMDA (Larrañaga et al, 2000)	1292.8	1372.1	1621.0	1713.7
MIMIC (Bengoetxea et al, 2002)	1297.8	1368.1	1626.0	1715.5
EBNA$_{BIC}$ (Bengoetxea et al, 2002)	1292.3	1378.0	1630.4	1705.0
TREE (Pelikan et al, 2007)	1299.0	1374.7	1659.9	1752.8
UMDA$_c$ (Lozano and Mendiburu, 2002)	1337.8	1412.0	1816.6	1889.6
EGNA$_{ee}$ (Lozano and Mendiburu, 2002)	1330.1	1400.1	1803.8	1876.7
IDEA-ICE (Bosman and Thierens, 2001)	1303.1	1371.2	1677.8	1760.5
EHBSA$_{WT}$ (Tsutsui, 2002)	1281.8	**1359.7**	**1590.4**	**1673.6**
EHBSA$_{WO}$ (Tsutsui, 2002)	1296.0	1365.7	1606.0	1710.2
NHBSA$_{WT}$ (Tsutsui et al, 2006)	1294.2	1362.7	**1591.2**	**1672.5**
NHBSA$_{WO}$ (Tsutsui et al, 2006)	1297.0	1363.2	1599.5	1678.3
REDA$_{UMDA}$ (Romero and Larrañaga, 2009)	1297.0	1375.5	1675.6	1764.5
REDA$_{MIMIC}$ (Romero and Larrañaga, 2009)	1313.6	1409.7	1706.6	1805.5
omeGA (Knjazew and Goldberg, 2000)	1310.4	1372.7	1690.0	1763.3
Best Known (Ceberio et al, 2012)	*1278.0*	*1359.0*	*1582.0*	*1659.0*

Although, Table 3.7 is generated based on relatively fewer instances compared to the review of Ceberio et al (2012), the ranks of algorithms are similar. EHBSA$_{WT}$ and NHBSA$_{WT}$ were the best performing algorithms in (Ceberio et al, 2012). A similar result is depicted by the overall rank of these algorithms in Table 3.7. EHBSA$_{WT}$ ranks 1^{st} while RK-EDA ranks 2^{nd} with NHBSA$_{WT}$.

It was observed that the RK based EDAs such as REDA$_{UMDA}$, REDA$_{MIMIC}$, EGNA$_{ee}$, UMDA$_c$ as well as the RK based GA (OmeGA) are ranked least in Table 3.7 which is also similar to the conclusion in the review of Ceberio et al (2012). RK-EDA outperforms all the RK based algorithms.

Furthermore, the performance of RK-EDA varies with different classes of problems. It produced competitive results on the PFSP, ranking 2^{nd} on PFSP$_s$ and 1^{st} on PFSP$_l$. It also produced competitive results for the TSP and LOP but less competitive performance on the QAP. This may be attributed to the fact that parameters that suit other problem classes are not particularly suitable for the search space pre-

Table 3.4: Permutation Flowshop Scheduling Problem (larger Instances)

Algorithms	tai50-10-0	tai50-10-1	tai100-20-0	tai100-20-1
RK-EDA (proposed)	**3090.7**	**2937.6**	**6386.4**	**6338.6**
UMDA (Larrañaga et al, 2000)	3151.6	3014.3	6907.5	6886.7
MIMIC (Bengoetxea et al, 2002)	3123.6	3011.0	6485.4	6441.6
EBNA$_{BIC}$ (Bengoetxea et al, 2002)	3182.6	2999.7	6917.0	6895.8
TREE (Pelikan et al, 2007)	3233.0	3106.3	6760.7	6715.0
UMDA$_c$ (Lozano and Mendiburu, 2002)	3517.1	3392.1	7400.6	7369.3
EGNA$_{ee}$ (Lozano and Mendiburu, 2002)	3486.7	3392.4	7352.0	7275.2
IDEA-ICE (Bosman and Thierens, 2001)	3245.2	3130.4	6869.6	6896.9
EHBSA$_{WT}$ (Tsutsui, 2002)	**3095.8**	2967.9	6605.3	6585.6
EHBSA$_{WO}$ (Tsutsui, 2002)	3265.5	3124.2	6992.1	6989.6
NHBSA$_{WT}$ (Tsutsui et al, 2006)	3103.0	2978.2	6584.7	6543.9
NHBSA$_{WO}$ (Tsutsui et al, 2006)	3126.0	3009.8	6643.1	6575.9
REDA$_{UMDA}$ (Romero and Larrañaga, 2009)	3336.7	3194.4	7381.2	7407.0
REDA$_{MIMIC}$ (Romero and Larrañaga, 2009)	3378.4	3339.2	7370.6	7392.3
omeGA (Knjazew and Goldberg, 2000)	3487.0	3440.0	7524.8	7526.5
Best Known (Ceberio et al, 2012)	*2991.0*	*2867.0*	*6106.0*	*6183.0*

sented by the QAP. Also, the operation of RK-EDA may not be particularly suited for the QAP because solving this problem requires the preservation of relative order which is not explicitly done by RK-EDA.

The results presented show that RK-EDA is competitive with leading EDAs on common permutation problems. More detailed results of the RK-EDA summarised according to the minimum, maximum, average and standard deviation values are presented in Table A.8.

3.5 Conclusions

EDAs based on RKs have previously been considered the poorest of permutation-based EDAs (Ceberio et al, 2012). One of the problems posed by RKs is attributed to representational redundancy (Pelikan et al, 2007). In this chapter, a novel RK based EDA (RK-EDA) is proposed which addresses this redundancy by rescaling RKs. This

Table 3.5: Quadratic Assignment Problem

Algorithms	tai15a	tai15b	tai40a	tai40b
RK-EDA (proposed)	404616.6	52088443.6	3391139.0	**652079961.9**
Factoradics (Regnier-Coudert and McCall, 2014)	399889.0	52002721.0	3327464.0	699677162.0
UMDA (Larrañaga et al, 2000)	403520.2	51949250.0	3268661.0	687946100.0
MIMIC (Bengoetxea et al, 2002)	404128.2	51975200.0	3296989.0	699671200.0
$EBNA_{BIC}$ (Bengoetxea et al, 2002)	397903.0	51971170.0	3261389.0	674194300.0
TREE (Pelikan et al, 2007)	404677.8	52033070.0	3383855.0	703873000.0
$UMDA_c$ (Lozano and Mendiburu, 2002)	437695.4	52952320.0	3592347.0	958721300.0
$EGNA_{ee}$ (Lozano and Mendiburu, 2002)	424760.2	52412710.0	3569396.0	894976600.0
IDEA-ICE (Bosman and Thierens, 2001)	421157.8	52217565.2	3469006.2	750040138.1
$EHBSA_{WT}$ (Tsutsui, 2002)	397816.8	51917837.8	3375198.4	664187682.6
$EHBSA_{WO}$ (Tsutsui, 2002)	418679.2	52322201.9	3551799.2	699043048.0
$NHBSA_{WT}$ (Tsutsui et al, 2006)	**389737.4**	**51765268.0**	3311038.4	**649637801.9**
$NHBSA_{WO}$ (Tsutsui et al, 2006)	396383.6	51845196.3	**3235093.8**	677273159.5
$REDA_{UMDA}$ (Romero and Larrañaga, 2009)	426467.8	52397560.0	3500793.0	868630000.0
$REDA_{MIMIC}$ (Romero and Larrañaga, 2009)	434397.2	52447330.0	3558663.0	886682900.0
omeGA (Knjazew and Goldberg, 2000)	430513.4	52416541.2	3600130.6	904728560.3
Best Known (Ceberio et al, 2012)	*388214.0*	*51765268.0*	*3139370.0*	*637250948.0*

approach improves the information captured by the probabilistic model. Furthermore, RK-EDA uses a cooling scheme to manage the rate of exploration/exploitation of the search space so that there is better exploration at the start of the algorithm and better exploitation of already found good pattern as the search progresses.

Learning a probability structure is considered the most expensive operation in EDAs (Bosman and Thierens, 2001). Here, a simple model which only stores the mean of solutions in a selected population is presented. This is computationally efficient compared to larger models used by the leading EDAs that grow exponentially with the size of a problem. Although RK-EDA uses simple procedures, it produces very competitive results. It outperforms other reviewed continuous EDAs. It is also competitive with the best EDAs.

RK-EDA's least competitive performance is observed on the QAP. This can be attributed RK-EDA's inability to explicitly preserve relative order.

Table 3.6: Linear Ordering Problem

Algorithms	t65b11xx	be75np	be75oi
RK-EDA (proposed)	356028.2	716644.3	111012.3
Factoradics (Regnier-Coudert and McCall, 2014)	350134.0	709328.0	110323.0
UMDA (Larrañaga et al, 2000)	347664.7	713228.8	110227.0
MIMIC (Bengoetxea et al, 2002)	349367.5	710296.2	110430.0
$EBNA_{BIC}$ (Bengoetxea et al, 2002)	347066.5	711958.9	110055.5
TREE (Pelikan et al, 2007)	337591.3	693237.8	108584.7
$UMDA_c$ (Lozano and Mendiburu, 2002)	285721.2	528949.0	91976.1
$EGNA_{ee}$ (Lozano and Mendiburu, 2002)	293023.9	528798.3	89905.6
IDEA-ICE (Bosman and Thierens, 2001)	338262.6	694163.3	107582.0
$EHBSA_{WT}$ (Tsutsui, 2002)	**356357.4**	**716816.1**	110997.1
$EHBSA_{WO}$ (Tsutsui, 2002)	318403.9	644297.8	105619.4
$NHBSA_{WT}$ (Tsutsui et al, 2006)	355904.9	716799.8	**111138.6**
$NHBSA_{WO}$ (Tsutsui et al, 2006)	355235.0	716395.8	110963.8
$REDA_{UMDA}$ (Romero and Larrañaga, 2009)	332107.8	649867.6	103679.3
$REDA_{MIMIC}$ (Romero and Larrañaga, 2009)	324124.2	541302.8	90079.3
omeGA (Knjazew and Goldberg, 2000)	293642.6	501186.0	96929.2
Best Known (Ceberio et al, 2012)	*411733.0*	*790966.0*	*118159.0*

RK-EDA's most competitive performance is seen on PFSP. Its performance on PFSP gets more competitive as the problem size increases presenting the best results on the largest of the considered PFSP instances. The next chapter will, therefore, focus on a wider range of PFSP instances.

Table 3.7: Average Ranks of Algorithms

Algorithms	**TSP**	**PFSP**$_s$	**QAP**	**LOP**	**ALL**	**PFSP**$_l$
EHBSA$_{WT}$ (Tsutsui, 2002)	1.00	1.75	4.00	2.00	2.13	3.25
RK-EDA (proposed)	3.75	2.50	7.00	2.25	4.00	1.00
NHBSA$_{WT}$ (Tsutsui et al, 2006)	8.50	3.00	2.00	1.75	4.00	3.00
NHBSA$_{WO}$ (Tsutsui et al, 2006)	6.00	4.50	2.50	4.25	4.27	4.75
Factoradics (Regnier-Coudert and McCall, 2014)	6.50	6.25	6.75	7.00	6.47	-
UMDA (Larrañaga et al, 2000)	8.25	6.75	4.75	6.25	6.53	7.00
EBNA$_{BIC}$ (Bengoetxea et al, 2002)	8.25	7.50	3.75	6.50	6.67	7.00
EHBSA$_{WO}$ (Tsutsui, 2002)	2.25	6.00	10.00	10.75	7.27	9.75
MIMIC (Bengoetxea et al, 2002)	10.50	8.00	6.25	6.50	7.80	3.50
TREE (Pelikan et al, 2007)	12.25	10.50	8.75	9.75	10.33	7.00
IDEA-ICE (Bosman and Thierens, 2001)	11.25	10.75	10.50	9.75	10.53	8.75
REDA$_{UMDA}$ (Romero and Larrañaga, 2009)	14.50	11.00	12.00	11.50	12.27	12.25
REDA$_{MIMIC}$ (Romero and Larrañaga, 2009)	8.50	14.25	14.00	13.25	12.47	12.25
EGNA$_{ee}$ (Lozano and Mendiburu, 2002)	9.00	14.75	13.25	15.00	12.93	12.25
omeGA (Knjazew and Goldberg, 2000)	14.25	12.00	14.75	14.50	13.80	14.75
UMDA$_c$ (Lozano and Mendiburu, 2002)	10.25	16.00	15.75	15.00	14.13	13.50

Chapter 4

Application of RK-EDA for the Permutation Flowshop Scheduling Problem

4.1 Introduction

The previous chapter introduced RK-EDA and presented results on some common permutation problems. Although each permutation problem examined in the previous chapter has unique features, the same set of algorithm parameters was used across all problems. While we have explored the robustness of RKEDA's performance under a fixed set of parameters, we have not fully explored the capability of the algorithm. For this reason and following promising results on the Permutation Flowshop Scheduling Problem (PFSP), this chapter presents an analysis of RK-EDA for PFSP. As is conventional in recent research, the focus is on optimising the Total Flow Time (TFT).

In Chapter 3, we started to answer the research question "In what way can exploration and exploitation be controlled in EDAs designed to solve permutation problems?". This chapter focuses on showing the effect of the proposed cooling scheme in RK-EDA. We show that RK-EDA can preserve diversity better than one of the leading EDAs, GM-EDA.

Experiments show that RK-EDA outperforms other permutation-based EDAs on instances of larger dimensions. It also outperforms other algorithms, presenting new best-known solutions on the largest problem instances.

4.2 RK-EDA: Analysis of Initial Variance

Several EDAs for permutation problems including the leading EDAs for this domain use large models. These models can grow up to the square of the size of the input ($O(n^2)$). RK-EDA, however, uses a model that increases linearly, its model grows according to the order of the magnitude of the input ($O(n)$). This in itself is a significant gain for the RK-EDA.

As shown in the previous chapter, RK-EDA requires standard EDA parameters such as population size and truncation size. Also, a parameter which is unique to RK-EDA is the initial variance. The initial variance determines the initial exploration rate of the algorithm. This value should be tuned carefully with respect to the size of a problem.

The behaviour of the RK-EDA can be examined through the search by measuring the probability of a job moving from its current position/changing its rank. Figure 4.1 shows the probability of a swap between jobs that are k position(s) apart in a problem of size 500. Based on variance value 0.0025, the percentage probabilities when k is 1, 10, 20, 30, 40 and 50 are respectively 48%, 41%, 34%, 27%, 21% and 15%. These values reduce as the search progresses as shown in Figure 4.1.

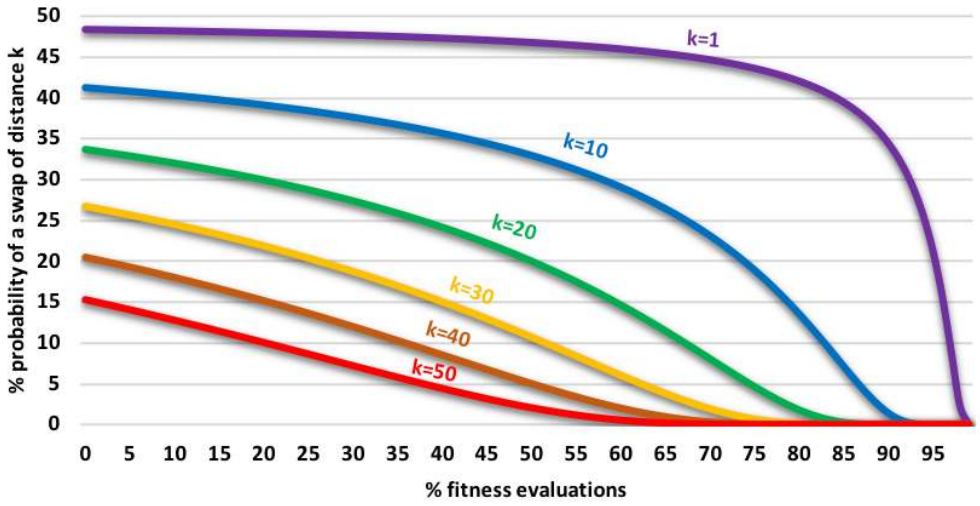

Figure 4.1: Percentage probability of a swap between k positions: problem size = 500 and initial variance = 0.0025

Due the rescaling procedure of RK-EDA ($rescaledRK_i = \frac{rank_i - 1}{n-1}$) as shown in the previous chapter, the granularity of RK values in a solution is inversely proportional to the problem size. Consequently, the difference between the RKs of consecutive ranks will reduce as the problem size grows.

Figure 4.2 shows the probability of a swap based on a variance value of 0.0025 across problem instances of sizes 20-500. For problem sizes 20, 50, 100, 200 and 500, the initial percentage probability of a swap between two consecutive ranks will respectively be 15%, 34%, 42%, 46% and 48%.

Previous studies applying the GM-EDA (Ceberio et al, 2014a) and DEP (Santucci et al, 2016) to the PFSP experienced an unexpected behaviour on problem instances with the largest number of jobs (500). These algorithms suffered from premature convergence suggesting that more exploration is needed to solve this problem effectively. These existing algorithms, therefore, used restart mechanism as well as local search procedures to achieve good quality solutions. This study leveraged on this information to ensure RK-EDA scales better to problems of larger dimensions without

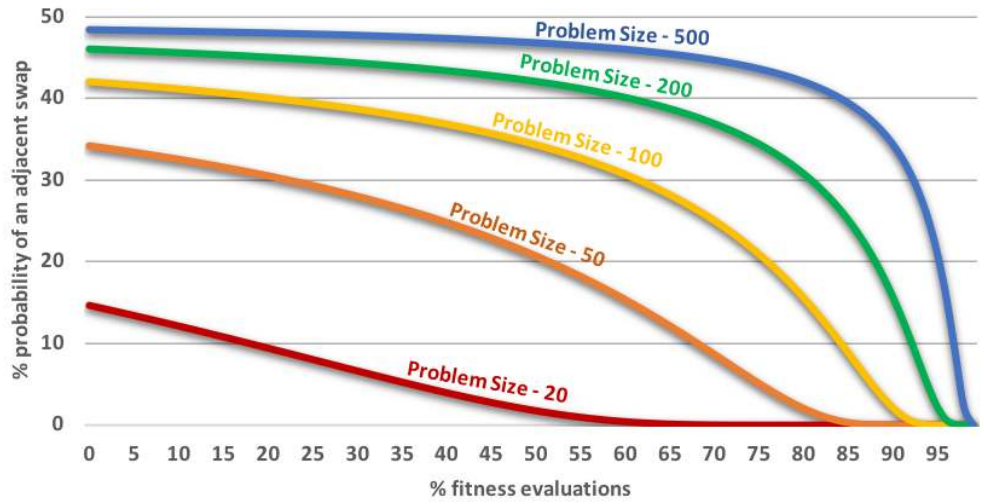

Figure 4.2: Percentage probability of an adjacent swap: varying problem sizes and initial variance = 0.0025

requiring additional methods such as local search. This chapter does not consider restart mechanisms either.

4.3 Experimental Settings

This section presents the problem sets considered in this chapter as well as the parameter settings for RK-EDA.

To be able to assess the performance of RK-EDA on different sizes of the PFSP, it is applied to problems of sizes 20-500 from the Taillard's benchmark problems Taillard (1993).

The following problem sets were considered.

1. Size 20: 20×5, 20×10, 20×20,
2. Size 50: 50×5, 50×10, 50×20,
3. Size 100: 100×5, 100×10, 100×20,
4. Size 200: 200×10, 200×20 and
5. Size 500: 500×20

Problem sets of sizes 20-100 are categorised based on 5, 10 and 20 machines while problem sets of size 200 are categorised by 10 and 20 machines only. Finally, all problem instances of size 500 are based on the use of 20 machines only.

Recent studies on PFSP (Ceberio et al, 2014a; Santucci et al, 2016) use a defined number of fitness evaluations as stopping criteria. The number of fitness evaluations is set according to the problem sizes as presented in Table 4.1.

Table 4.1: Stopping Criteria: Number of Fitness Evaluations

Problem Sizes	**Fitness Evaluations**
20×05	182,224,100
20×10	224,784,800
20×20	256,896,400
50×05	220,712,150
50×10	256,208,100
50×20	275,954,150
100×5	235,879,800
100×10	266,211,000
100×20	283,040,000
200×10	272,515,500
200×20	287,728,850
500×20	260,316,750

RK-EDA varies its exploration-exploitation balance according to the evaluation budget. For this reason, it can present competitive results even with a relatively low budget. To be able to parametrise the algorithm, RK-EDA is set to use only 20% of the conventionally used budget.

The range of population size, truncation size and variance values used to parametrise RK-EDA are presented in Table 4.2. Although population size $10n$ was used in the previous chapter, smaller values (n, $2n$ and $3n$) are considered in this chapter. This will allow the algorithm run for more generations. We try truncation values that are 10%, 25% and 40% of the population size. Initial variance values are varied between 0.0025 and 0.0400.

Due to constraints on computational cost, problems sets of sizes 20-100 are used to parametrise the algorithm.

Table 4.2: Parameter Settings for RK-EDA

Parameters	**Values**
Population Size (p_s)	n, $2n$, $3n$
Truncation Size (t_s)	$0.1{*}p_s$, $0.25{*}p_s$, $0.4{*}p_s$
Variance (σ)	0.0025, 0.0100, 0.0225, 0.0400
Stopping Criteria	20% FEs, 100% FEs

4.3.1 Preliminary Results

Tables 4.3, 4.4 and 4.5 respectively present results for tuning the population size, truncation size and initial variance. The values in these tables are the ARPD from the best-known solution for each problem instance based on 20 runs of RK-EDA. The lowest ARPD, as well as ARPD that are not significantly different, are presented in

Table 4.3: Parameter Settings: Population Sizes

Problem Instances	n	$2n$	$3n$
tai20_5_0	**14039**	14046	14073
tai20_5_1	**15160**	15170	15185
tai20_10_0	**20957**	20966	20979
tai20_10_1	**22493**	**22482**	**22530**
tai20_20_0	**33652**	**33670**	**33676**
tai20_20_1	**31592**	**31593**	**31595**
tai50_5_0	65571	**65466**	**65536**
tai50_5_1	**68990**	**69067**	69152
tai50_10_0	89412	**89130**	89198
tai50_10_1	**84547**	**84561**	**84676**
tai50_20_0	128039	**127729**	**127850**
tai50_20_1	120974	**120662**	**120671**
tai100_5_0	**256650**	**256608**	**256699**
tai100_5_1	**245498**	245697	245686
tai100_10_0	**303820**	**303688**	**303731**
tai100_10_1	**279378**	**279439**	280008
tai100_20_0	**373070**	**372743**	**372725**
tai100_20_1	380043	**379185**	**379019**

Table 4.4: Parameter Settings: Truncation Sizes

Problem Instances	10%	25%	40%
tai20_5_0	**14039**	14044	14068
tai20_5_1	**15160**	15197	15224
tai20_10_0	**20957**	20979	20995
tai20_10_1	**22493**	22609	22674
tai20_20_0	**33652**	**33701**	**33721**
tai20_20_1	**31592**	**31593**	**31603**
tai50_5_0	**65571**	**65568**	65702
tai50_5_1	**68990**	69182	69299
tai50_10_0	**89412**	**89349**	89590
tai50_10_1	**84547**	84923	84957
tai50_20_0	**128039**	**127963**	**127962**
tai50_20_1	**120974**	**120773**	**120942**
tai100_5_0	**256650**	256866	257206
tai100_5_1	**245498**	246032	246682
tai100_10_0	**303820**	**303990**	**304123**
tai100_10_1	**279378**	280155	280607
tai100_20_0	**373070**	**372996**	**373347**
tai100_20_1	380043	**379448**	380727

bold. The student t-test with 95% confidence interval is used to measure statistical significance.

While the population size is tuned, the truncation size is set to 10% of the population size and the variance set to 0.0225. Table 4.3 presents the ARPD values for population sizes n, $2n$ and $3n$ where n is the problem size. The performance of population sizes n and $2n$ are particularly similar. We however selected n as this allows RK-EDA to run for even more generations.

To tune the truncation size, the population size is set to n, and the variance value is not changed from 0.0225. Table 4.4 presents the ARPD for each problem instance using truncation sizes equal to 10%, 25% and 40% of the population size. The results show that 10% presents the best ARPD or results that are not significantly different from the best on all but one of the instances. The truncation size is therefore set to 10% of the population size.

Table 4.5: Parameter Settings: Initial Variance Values

Problem Instances	0.0025	0.0100	0.0225	0.0400
tai20_5_0	**14036**	**14038**	**14039**	14042
tai20_5_1	**15153**	15160	15160	15166
tai20_10_0	**20957**	**20962**	**20957**	**20964**
tai20_10_1	**22446**	**22464**	22493	22504
tai20_20_0	**33623**	**33629**	33652	33666
tai20_20_1	**31587**	**31590**	**31592**	**31592**
tai50_5_0	**65412**	65540	65571	65553
tai50_5_1	**68810**	68944	68990	68968
tai50_10_0	**89262**	**89242**	**89412**	89417
tai50_10_1	**84442**	**84472**	**84547**	84645
tai50_20_0	**128005**	**128076**	**128039**	128261
tai50_20_1	**120689**	121013	120974	121100
tai100_5_0	**256365**	**256506**	256650	256791
tai100_5_1	**245118**	245458	245498	245495
tai100_10_0	**303079**	**303770**	303820	304344
tai100_10_1	**279254**	**279413**	**279378**	280037
tai100_20_0	**372604**	**372735**	**373070**	373360
tai100_20_1	**379089**	**379545**	380043	380242

The initial variance value is then tuned while the population size is set to n and truncation size is a tenth of the population size. Table 4.5 presents the ARPD from the best known solution for each instance using variance values 0.0025, 0.0100, 0.0225 and 0.0400. The lowest variance value 0.0025 presents the best ARPD or ARPD that is not significantly different from the best. The initial variance value is therefore set to 0.0025 for subsequent experiments in this chapter.

The parameter settings that will be used in this chapter are therefore population size n, truncation size $0.1n$ and variance value 0.0025.

4.4 Results and Discussion

In this section, we present the results obtained by RK-EDA on the selected PFSP problem instances. RK-EDA is compared with leading permutation-based EDAs as well as leading algorithms on PFSP. Finally, we investigate whether mechanisms inherent to RK-EDA such as its cooling scheme can maintain diversity throughout the search without requiring the use of LS. We use the Kendal Tau Distance (KTD) as a measure of diversity to compare RK-EDA with one of the leading permutation EDAs, GM-EDA.

4.4.1 Comparing RK-EDA with stand-alone EDAs

In this section, RK-EDA is compared with GM-EDA, NHBSA and EHBSA which are the leading stand-alone EDAs applied to the PFSP (Ceberio et al, 2014a). Table 4.6 presents the ARPD based on 20 runs of each of these algorithms. Results for GM-EDA, NHBSA and EHBSA are obtained from (Ceberio et al, 2014a). Note however that RK-EDA was only executed for 20% of the fitness evaluations used by GM-EDA, NHBSA and EHBSA. This is because RK-EDA can present competitive results even

with fewer evaluations as it can adjust its search pattern according to the fitness budget.

Table 4.6: ARPD: Comparing RK-EDA with other EDAs

Problem Size	Best Known	GM-EDA	EHBSA	NHBSA	RK-EDA	Problem Size	Best Known	GM-EDA	EHBSA	NHBSA	RK-EDA
20 X 5	14033	0.18	**0.00**	**0.00**	0.02	100 X 5	253605	**0.87**	1.08	1.75	1.09
	15151	0.48	**0.00**	**0.00**	0.01		242579	1.08	**0.88**	1.83	1.05
	13301	0.50	0.02	**0.00**	0.03		238075	**0.85**	**0.79**	1.53	0.94
	15447	0.43	0.03	**0.01**	**0.00**		227889	0.78	**0.69**	1.53	0.88
	13529	0.21	**0.00**	**0.00**	0.04		240589	**0.80**	**0.76**	1.56	0.95
	13123	0.08	**0.01**	**0.00**	**0.00**		232689	**0.90**	1.03	1.84	1.06
	13548	0.79	**0.00**	**0.00**	0.36		240669	1.00	**0.57**	1.47	1.07
	13948	0.18	0.03	**0.00**	0.01		231064	**1.06**	**1.05**	1.94	1.24
	14295	0.18	0.11	**0.00**	0.11		248039	**1.05**	1.18	1.93	1.20
	12943	0.46	**0.00**	**0.00**	0.07		243258	1.00	**0.57**	1.58	1.02
20 X 10	20911	0.45	**0.00**	**0.00**	0.22	100 X 10	299101	1.80	2.80	2.60	**1.33**
	22440	0.54	**0.00**	**0.00**	**0.03**		274566	2.08	3.17	3.03	**1.71**
	19833	0.31	**0.00**	**0.00**	0.12		288543	**1.74**	2.87	2.69	**1.63**
	18710	0.75	0.06	**0.02**	0.31		301552	2.08	3.03	2.78	**1.60**
	18641	0.35	0.06	**0.02**	0.07		284722	1.95	2.88	2.91	**1.61**
	19245	0.77	**0.02**	0.10	**0.02**		270483	1.83	3.45	2.89	**1.36**
	18363	0.47	0.05	**0.02**	0.12		280257	1.65	2.45	2.45	**1.39**
	20241	0.47	**0.00**	**0.00**	0.19		291231	2.03	3.24	2.83	**1.56**
	20330	0.27	0.02	**0.00**	**0.00**		302624	**1.76**	2.70	2.48	**1.67**
	21320	0.24	0.01	**0.00**	0.01		291705	**1.68**	3.07	2.93	**1.56**
20 X 20	33623	0.65	0.46	**0.00**	**0.00**	100 X 20	366438	2.26	4.55	3.12	**1.68**
	31587	0.28	0.01	**0.00**	**0.00**		373138	2.04	4.13	3.17	**1.59**
	33920	**0.04**	**0.00**	**0.00**	**0.00**		371206	1.99	3.92	2.86	**1.57**
	31661	0.28	**0.00**	**0.00**	0.09		373574	1.92	4.20	3.19	**1.59**
	34557	0.26	**0.00**	**0.00**	0.05		369850	1.93	4.27	3.17	**1.61**
	32564	0.30	**0.00**	**0.00**	**0.00**		372752	2.17	4.22	3.10	**1.71**
	32922	0.61	**0.03**	**0.00**	0.32		373447	2.19	4.62	3.35	**1.80**
	32412	0.52	0.08	**0.02**	0.28		385456	1.96	4.03	2.92	**1.45**
	33600	0.56	**0.03**	**0.00**	1.03		375352	2.01	3.93	3.10	**1.48**
	32262	0.41	0.03	**0.00**	**0.01**		379899	2.05	4.01	3.08	**1.71**
50 X 5	64802	0.79	**0.03**	0.79	0.94	200 X 10	1047541	1.20	5.03	4.06	**0.75**
	68058	0.94	**0.04**	0.92	1.10		1035783	1.49	5.89	4.65	**0.93**
	63162	1.34	**0.33**	1.28	1.35		1045706	1.30	5.32	4.34	**0.82**
	68226	1.27	**0.20**	1.15	1.28		1029580	1.38	5.31	4.18	**1.00**
	69392	0.89	**0.08**	0.78	1.02		1036464	1.37	5.62	4.16	**0.76**
	66841	0.82	**0.14**	0.95	0.91		1006650	1.39	6.46	5.01	**0.92**
	66253	0.96	**0.00**	0.87	1.15		1052786	1.23	6.49	5.10	**0.79**
	64359	0.97	**0.08**	0.86	1.16		1044961	1.39	5.66	4.47	**0.85**
	62981	0.81	**0.11**	0.87	1.11		1023315	1.29	6.28	4.81	**0.80**
	68811	1.06	**0.26**	1.10	1.35		1029198	1.48	6.32	4.92	**0.84**
50 X 10	87204	2.11	**0.72**	1.86	2.36	200 X 20	1225282	1.72	6.26	5.24	**0.90**
	82820	2.45	**1.03**	1.97	1.96		1239246	1.66	6.98	5.59	**1.04**
	79987	1.84	**0.48**	1.61	1.60		1263134	1.57	5.88	5.27	**0.84**
	86545	1.87	**0.72**	1.51	1.54		1233443	1.73	7.01	5.80	**1.03**
	86424	2.05	**0.89**	1.73	1.92		1220117	1.93	7.28	5.65	**1.01**
	86637	1.55	**0.45**	1.45	1.51		1223238	1.69	7.24	5.41	**0.85**
	88866	1.97	**0.77**	1.63	1.72		1237116	1.66	7.48	5.64	**0.91**
	86820	2.04	**0.89**	1.73	1.63		1238975	1.72	6.74	5.45	**0.96**
	85526	2.10	**0.78**	1.76	1.73		1225186	1.80	7.59	5.93	**0.98**
	87998	2.09	**0.76**	1.99	1.89		1244200	1.68	6.55	5.46	**0.82**
50 X 20	125831	1.76	**1.20**	1.75	1.73	500 X 20	**6638306**	10.05	9.68	8.39	**0.49**
	119247	1.59	**0.98**	1.75	1.21		**6764798**	9.63	9.25	7.91	**0.55**
	116459	2.24	**1.30**	1.90	1.88		**6692427**	9.35	9.58	8.14	**0.48**
	120712	1.92	**1.07**	1.76	1.53		**6725985**	9.74	9.22	7.97	**0.47**
	118184	2.30	**1.32**	1.92	1.76		**6686734**	9.84	9.48	8.08	**0.62**
	120703	1.78	**1.13**	1.67	1.39		**6687549**	9.62	10.03	8.36	**0.52**
	122962	2.10	**1.46**	1.85	1.94		**6635167**	10.36	9.97	8.44	**0.54**
	122489	2.24	**1.69**	1.96	**1.72**		**6713812**	9.53	9.79	8.30	**0.55**
	121872	1.79	**1.16**	1.76	1.49		**6654590**	9.77	9.41	8.13	**0.43**
	124064	1.95	**1.17**	1.71	1.76		**6695956**	9.67	9.59	8.31	**0.47**

Table 4.6 presents the ARPD for GM-EDA, EHBSA, NHBSA and RK-EDA based on 20 runs. The best ARPD of the four algorithms is presented in bold while results that are not significantly different are also presented in bold. The student t-test is also used based on 95% confidence interval.

Table 4.7: ARPD: RK-EDA and leading algorithms for PFSP (20 X 5 - 50 X 20)

n x m	best	AGA	DEP	HGM-EDA	IGA	ILS	RK-EDA
20 × 5	14033	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	15151	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	13301	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	15447	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	13529	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	13123	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	13548	**0.00**	**0.00**	**0.00**	0.02	**0.00**	0.08
	13948	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	14295	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**
	12943	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
20 × 10	20911	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.17
	22440	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	19833	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02
	18710	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.04
	18641	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.01
	19245	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	18363	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02
	20241	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.10
	20330	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	21320	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
20 × 20	33623	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	31587	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	33920	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	31661	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**
	34557	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02
	32564	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	32922	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.13
	32412	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.14
	33600	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.51
	32262	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50 × 5	64802	**0.05**	0.06	0.12	0.08	**0.05**	0.87
	68058	**0.06**	0.09	0.13	0.14	0.13	1.01
	63162	**0.19**	0.21	0.38	0.24	0.24	1.18
	68226	**0.17**	**0.13**	0.22	0.26	**0.13**	1.12
	69392	**0.09**	0.09	0.15	0.14	0.16	0.91
	66841	0.10	**0.04**	0.18	0.16	0.19	0.91
	66253	**0.03**	**0.03**	0.08	0.10	0.09	0.96
	64359	**0.05**	**0.05**	0.23	0.32	0.15	0.98
	62981	0.09	**0.05**	0.14	0.23	0.19	0.93
	68811	0.20	**0.15**	0.34	0.33	0.22	1.09
50 × 10	87204	0.33	**0.18**	0.39	0.40	0.44	1.81
	82820	**0.22**	0.30	0.60	0.39	0.59	1.55
	79987	**0.23**	**0.22**	0.36	0.40	**0.32**	1.40
	86545	**0.21**	**0.16**	0.36	0.31	**0.26**	1.23
	86424	**0.17**	0.28	0.41	0.35	**0.24**	1.54
	86637	**0.13**	**0.11**	0.29	0.35	**0.17**	1.11
	88866	**0.25**	0.42	0.48	**0.27**	**0.28**	1.53
	86820	0.19	**0.01**	0.36	0.44	0.43	1.28
	85526	**0.29**	**0.28**	0.42	0.49	0.44	1.46
	87998	**0.18**	0.51	0.54	0.45	0.39	1.89
50 × 20	125831	**0.10**	**0.14**	0.39	0.41	0.33	1.57
	119247	**0.05**	**0.07**	0.23	0.15	0.14	0.98
	116459	**0.19**	**0.28**	0.44	0.36	0.33	1.68
	120712	**0.22**	0.34	0.34	0.36	0.35	1.40
	118184	**0.40**	**0.39**	0.52	0.60	**0.48**	1.39
	120703	**0.19**	**0.16**	0.35	0.47	0.34	1.11
	122962	**0.38**	**0.36**	0.47	0.47	0.52	1.55
	122489	**0.16**	**0.14**	0.55	0.45	0.41	1.40
	121872	**0.16**	**0.12**	0.37	0.41	0.34	1.12
	124064	**0.23**	**0.29**	0.42	0.64	0.35	1.36

As shown in Table 4.6, the EHBSA presents the best ARPD for most of the instances of size 20 (20 × 5, 20 × 10 and 20 × 20) while the NHBSA presents the

Table 4.8: ARPD: RK-EDA and leading algorithms for PFSP (100 X 5 - 500 X 20)

n x m	best	AGA	DEP	HGM-EDA	IGA	ILS	RK-EDA
100 × 5	253605	0.29	**0.05**	0.23	0.46	0.48	0.97
	242579	0.30	**0.05**	0.35	0.82	0.80	0.88
	238075	0.22	**0.07**	0.26	0.29	0.34	0.82
	227889	0.17	**0.06**	0.20	0.32	0.45	0.78
	240589	0.21	**0.02**	0.23	0.45	0.56	0.79
	232689	0.32	**0.06**	0.28	0.47	0.37	0.94
	240669	**0.15**	0.25	0.34	0.34	0.47	0.90
	231064	0.29	**0.07**	0.35	0.81	0.56	1.05
	248039	0.40	**0.09**	0.38	0.67	0.66	1.11
	243258	0.19	**0.07**	0.28	0.70	0.51	0.90
100 × 10	299101	0.43	**0.16**	0.44	0.56	0.42	1.14
	274566	0.60	**0.28**	0.69	0.85	0.54	1.30
	288543	0.37	**0.18**	0.38	0.88	0.82	1.25
	301552	0.50	**0.18**	0.53	0.96	0.88	1.32
	284722	0.61	**0.22**	0.54	0.64	0.73	1.47
	270483	0.42	**0.19**	0.45	0.63	0.52	1.18
	280257	0.37	**0.25**	0.40	0.41	0.46	1.20
	291231	0.49	**0.27**	0.61	0.72	0.69	1.34
	302624	0.36	**0.20**	0.41	0.66	0.57	1.36
	291705	0.48	**0.06**	0.50	0.45	0.47	1.35
100 × 20	366438	0.80	**0.37**	0.67	0.71	0.79	1.36
	373138	0.55	**0.25**	0.58	0.73	0.54	1.44
	371206	0.53	**0.27**	0.41	0.49	0.38	1.29
	373574	0.60	**0.26**	0.45	0.49	0.50	1.38
	369850	0.59	**0.21**	0.48	0.50	0.57	1.32
	372752	0.51	**0.30**	0.42	0.56	0.56	1.41
	373447	0.70	**0.33**	0.63	0.90	0.73	1.49
	385456	0.46	**0.20**	0.43	0.64	0.48	1.29
	375352	0.62	**0.41**	0.52	0.54	0.44	1.39
	379899	**0.48**	**0.46**	**0.49**	**0.48**	**0.44**	1.42
200 × 10	1047541	0.49	0.22	**0.19**	**0.15**	0.29	0.52
	1035783	0.94	**0.15**	0.32	0.65	0.71	0.70
	1045706	0.66	**0.15**	0.32	0.48	0.55	0.63
	1029580	0.77	**0.12**	0.45	0.48	0.51	0.74
	1036464	0.68	**0.13**	0.19	0.39	0.42	0.49
	1006650	0.50	**0.23**	**0.19**	0.55	0.67	0.67
	1052786	0.95	**0.10**	0.24	0.58	0.61	0.54
	1044961	0.62	**0.11**	0.25	0.34	0.35	0.65
	1023315	0.81	**0.24**	**0.28**	0.86	0.93	0.58
	1029198	0.97	**0.25**	**0.39**	0.78	0.87	0.62
200 × 20	1225282	0.76	**0.20**	**0.39**	**0.40**	**0.30**	0.58
	1239246	1.07	**0.21**	**0.54**	0.74	0.64	0.71
	1263134	1.08	**0.26**	**0.48**	**0.51**	**0.44**	**0.52**
	1233443	1.25	**0.24**	**0.58**	1.03	0.83	0.80
	1220117	1.12	**0.17**	**0.53**	0.78	0.80	0.72
	1223238	1.17	**0.19**	**0.46**	0.86	0.77	0.60
	1237116	1.03	**0.15**	**0.64**	0.81	**0.70**	**0.67**
	1238975	1.25	**0.19**	0.51	0.72	0.81	0.69
	1225186	1.44	**0.14**	**0.59**	0.87	0.88	0.75
	1244200	1.16	**0.11**	0.52	0.55	0.48	0.61
500 × 20	**6638306**	1.16	2.07	3.09	0.96	0.88	**0.17**
	6764798	1.21	1.62	2.92	1.02	0.90	**0.24**
	6692427	1.06	1.90	2.88	0.77	0.77	**0.16**
	6725985	1.17	1.75	2.81	0.81	0.77	**0.14**
	6686734	1.42	1.77	2.97	0.67	0.66	**0.14**
	6687549	1.14	1.27	3.11	0.71	0.67	**0.21**
	6635167	1.38	2.05	3.18	0.92	0.88	**0.21**
	6713812	1.15	1.57	2.94	1.03	1.12	**0.21**
	6654590	1.14	1.96	2.92	0.83	0.80	**0.11**
	6695956	1.26	1.94	3.09	1.00	1.04	**0.17**

best results on most of the instances of size 50 (50 × 5, 50 × 10 and 50 × 20). RK-EDA was able to present the best ARPD or results that are not significantly different from the best ARPD on some of the size 20 and 50 instances. GM-EDA

was able to achieve ARPD that is not significantly different from the best ARPD on only one instance of the 20 × 20 category and none of the other size 20/50 instances. For problem size 100, GM-EDA and EHBSA present the best ARPD on most of the instances of the 100 × 5 category. RK-EDA however presents best ARPD on the 100 × 10 and 100 × 20 categories. RK-EDA also presents the best ARPD on all instances of size 200 and 500.

The results presented in Table 4.6 show that RK-EDA scales better to larger problems than the other EDAs. Although RK-EDA was executed with just 20% of the fitness evaluations used by other algorithms, it significantly outperforms other EDAs on problems of larger dimensions. This may be attributed to the use of cooling scheme in RK-EDA which helps it to maintain better exploration-exploitation balance.

Although GM-EDA also seems to have the same characteristic of scaling better to larger problems, its performance becomes the worst on the largest problem set (500 × 20). RK-EDA's best performance is seen on the largest instances where it presents results that are at least 7.7% better than other EDAs.

4.4.2 Comparing RK-EDA with Leading Algorithms

The leading algorithms applied to the PFSP as presented in (Santucci et al, 2016) are HGM-EDA, AGA, ILS, IGA and DEP. In contrast to the previous section, RK-EDA is compared with these algorithms using on the same number of fitness evaluations presented in Table 4.1. Tables 4.7 and 4.7 present the ARPDs obtained by these algorithms based on 20 runs. Table 4.7 presents results for problem sizes 20 - 50 while Table 4.7 presents results for sizes 100 - 500. The best ARPD of these algorithms for each problem instance is presented in bold. Based on student t-test and 95% confidence interval, ARPDs that are not significantly different from the best ARPD are also presented in bold.

As shown in Table 4.7, AGA, DEP, HGM-EDA and ILS are all able to reach best known ARPD on all problem instances of categories 20 × 5, 20 × 10 and 20 × 20. IGA is also able to reach the best-known solution on most of the instances but one instance of the 20 × 5 category. RK-EDA, however, performs significantly worse on a third of instances of size 20.

For problem instances consisting of 50-200 jobs, DEP presents significantly better ARPD compared to other algorithms on most of the problem instances. AGA, HGM-EDA and ILS also present the best ARPD on a few of these instances. However, for problem instances made up of 500 jobs, RK-EDA presents not only best ARPDs but also new best-known TFTs for all the problem instances.

Detailed results of RK-EDA based on 100% of the conventionally used number of fitness evaluations are presented in Tables A.10, A.11 and A.12 of the Appendix.

4.4.3 Diversity in RK-EDA

Since RK-EDA does not use restart mechanisms or LS methods, it is important to ensure that the algorithm allows the search to both explore and exploit the search space. In section 4.2, we analysed the probability of an adjacent swap. We showed

experimentally that the search converged too quickly for the smaller problems. In this section, we do a measure of similarity of actual solutions during the run of RK-EDA. The diversity of this population is measured using the KTD Fligner and Verducci (1988). KTD represents the number of adjacent transposition that needs to be performed to transform a permutation into another one.

Pairwise KTD between two permutations π_1 and π_2 is calculated following eq. 4.1. The elements i and j belong to the set P of unordered pairs of elements obtained from π_1 and π_2. $K_{ij}(\pi_1, \pi_2)$ equals either 0 if i and j are in the same order, or 1 if i and j are in a different order in π_1 and π_2

$$KTD(\pi_1, \pi_2) = \sum_{\{i,j\} \in P} K_{ij}(\pi_1, \pi_2) \tag{4.1}$$

We calculate the KTD for all pairwise comparison from all solution pairs in a population at each generation. These are summed up and averaged by the number of pairwise comparisons. To normalise the average KTD, we divide it by $n(n-1)/2$ where n is the problem size. This way, a value of 1 indicates maximum disagreement while 0 indicates two identical permutations.

In this section, diversity in GM-EDA is compared to RK-EDA using three problem instances of different sizes. Figures 4.3, 4.4 and 4.5 respectively shows the KTD during a run of both algorithms on tai20_5_0, tai50_5_0 and tai100_5_0.

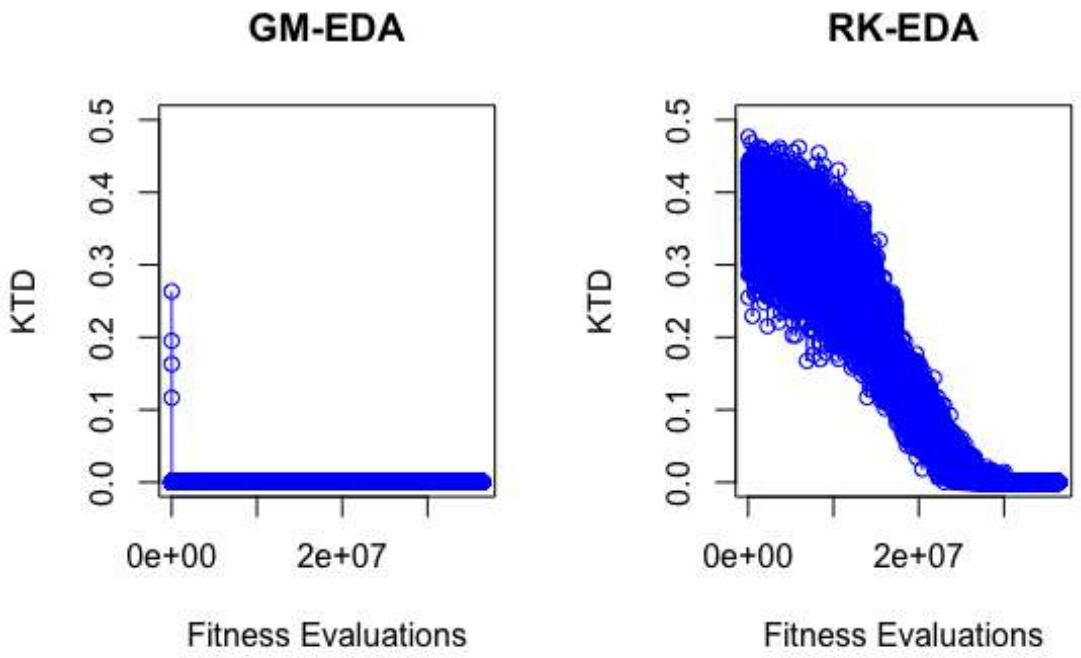

Figure 4.3: Measure of KTD on tai20_5_0

In Figures 4.3, 4.4 and 4.5, RK-EDA and GM-EDA explored least on the smallest and most of the largest of the three problems. GM-EDA, however, converged much

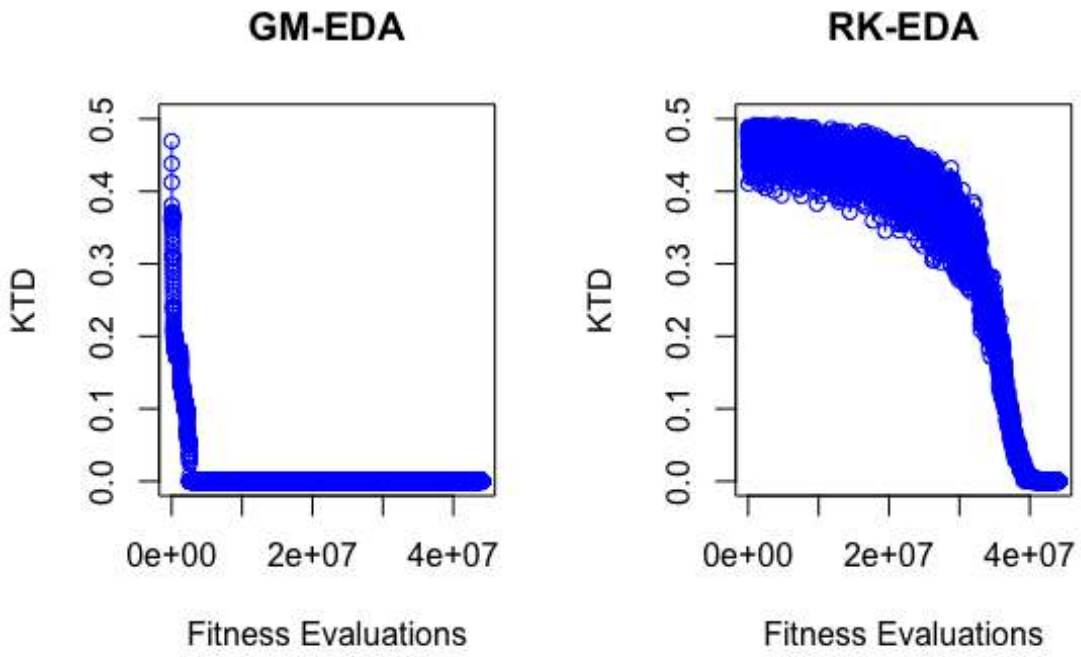

Figure 4.4: Measure of KTD on tai50_5_0

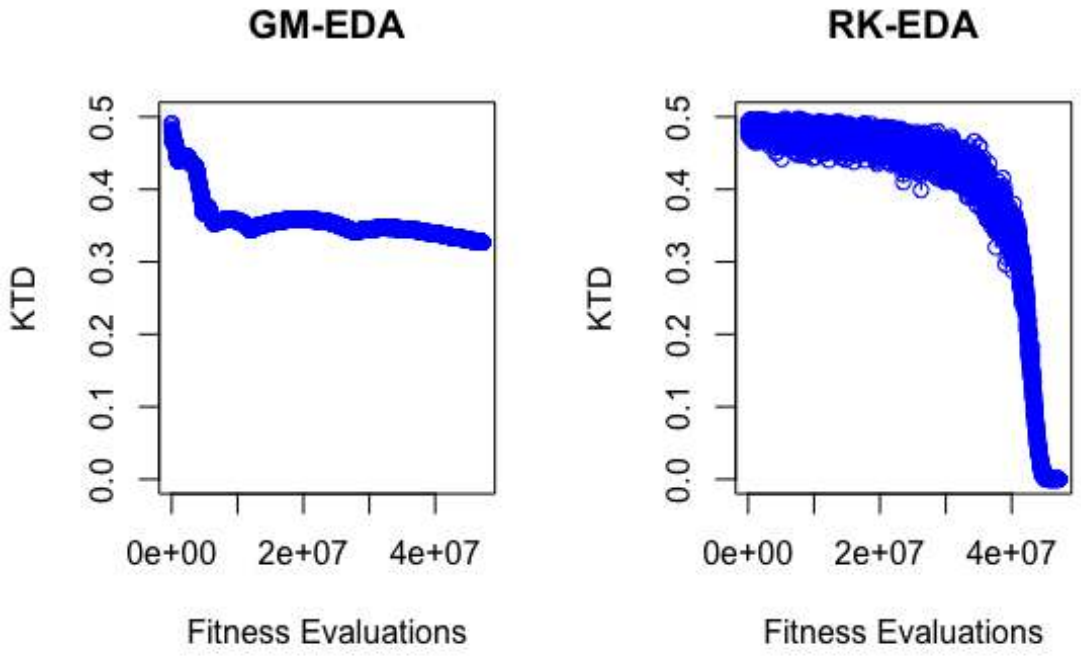

Figure 4.5: Measure of KTD on tai100_5_0

quicker than RK-EDA on tai20_5_0 and tai50_5_0. In Figure 4.5, GM-EDA maintained a high diversity all through the run while RK-EDA was able to maintain high exploration at the start and more exploitation closer to the end of the run.

4.5 Conclusions

In this chapter, RK-EDA which uses a lightweight univariate model based on RKs is further adapted for the PFSP. The lightweight characteristic of the model is of particular importance considering that learning a probability structure is considered the most expensive operation in EDAs (Bosman and Thierens, 2001). One of the key features of RK-EDA is its cooling scheme which manages the variance used in the model. Based on the cooling scheme which adapts to the number of fitness evaluations, the algorithms can present competitive results when given less budget. With only a fifth of the conventionally applied number of fitness evaluations, RK-EDA outperformed other EDAs on problems of larger dimensions. RK-EDA also outperforms state-of-the-art algorithms on the largest PFSP instances, presenting new best-known solutions.

Furthermore, diversity computed experimentally confirmed that RK-EDA avoids premature convergence in larger problems and exhibits both exploration and exploitation behaviours during the search.

Chapter 5

Estimation of Distribution Algorithms for the RCPSP and MRCPSP

5.1 Introduction

The MRCPSP is a multi-component problem consisting of *activity scheduling* and *mode assignment* problems. Due to the interactions between these sub-problems, they cannot be solved in isolation. A similar problem, which is only made up of the *activity scheduling* problem, is the RCPSP. *Activity scheduling* requires that precedence constraints are respected in addition to generating a valid permutation. To achieve good quality RCPSP solutions, it is conventional to use local search procedures to improve the performance of meta-heuristics.

This chapter adapts RK-EDA to solve the RCPSP, providing more insight into the *activity scheduling* problem. To gain more insight into the *mode assignment* problem, this chapter also proposes the BPGA-EDA which focuses on solving the *mode assignment* component of the MRCPSP with an EDA. Finally, the approaches are brought together in BPEDA, which uses RK-EDA for *activity scheduling* and the mechanisms of BPGA-EDA for *mode assignment*.

This chapter addresses the research question "How can we reduce the need for local improvement procedures in multi-component scheduling problems?". Using MRCPSP as a case study, this chapter explains how adapting EDAs to components of the MRCPSP led to competitive results with fewer improvement procedures.

The rest of the chapter is structured as follows. Section 5.2 describes problem instances of the RCPSP and MRCPSP. Section 5.3 presents the base method, BPGA, used in this chapter. Section 5.4 presents RK-EDA adapted for solving the RCPSP. Section 5.5 presents BPGA-EDA applied to solve the MRCPSP. BPEDA is presented in Section 5.6. In Section 5.7, BPEDA is compared with existing methods of solving the MRCPSP. This chapter is concluded in Section 5.8.

5.2 Problem Instances

5.2.1 RCPSP Instance

RCPSP is formally defined in Chapter 2.5.5. An example of an RCPSP instance with 30 activities to be scheduled is presented in Table 5.1. Activities 1 and 32 are respectively the dummy start and finish activities. Unlike MRCPSP, there is only one mode of execution for each activity. Furthermore, each activity requires four different renewable resources R1-R4 for the given duration. Each non-dummy activity also has at least one successor as presented in the last column of Table 5.1.

The precedence constraint requires that an activity cannot be performed before any activity defined as its predecessor. Also, each activity can only be performed if there are resources to perform it. These resources are available per period of time as shown in the last row of Table 5.1. Therefore, assigning start and finish times to an activity of a project will depend on the latest finish time of the predecessors and the earliest available time of all the resources required to perform that activity. The aim is to assign start and finish times to all activities of the project such that the makespan is minimised.

5.2.2 MRCPSP Instance

MRCPSP is formally defined in Chapter 2.5.6. An example of an MRCPSP instance with 10 activities to be scheduled is presented in Table 5.2. Activities 1 and 12 are respectively the dummy start and finish activities. There are three modes of execution for each non-dummy activity. Furthermore, each activity requires two different renewable resources (R1 and R2) as well as non-renewable resources (N1 and N2) for the given duration. Each non-dummy activity also has at least one successor as presented in the last column of Table 5.2.

The precedence constraint requires that an activity cannot be performed before any activity defined as its predecessor. Also, each activity can only be performed if there are enough resources R1 and R2 to perform it. Also, each activity can only be performed in one mode of execution. R1 and R2 are available per period of time while the availabilities of N1 and N2 are for the entire project duration. The sum of N1 and N2 for all mode selections must therefore be less than their respective availabilities. The availabilities of these resources are shown in the last row of Table 5.2. Also, assigning start and finish times to an activity of a project will depend on the latest finish time of the predecessors and the earliest available time of all the resources required to perform that activity. The aim is to assign start and finish times to all activities of the project such that the makespan is minimised.

5.3 The BPGA

BPGA is one of the leading approaches of solving the RCPSP (Debels and Vanhoucke, 2005) as well as the MRCPSP (Van Peteghem and Vanhoucke, 2010). BPGA also uses relatively simpler approaches compared to other leading algorithms. The BPGA is

Table 5.1: RCPSP Instance

Activities	Mode	Duration	R1	R2	R3	R4	Successors
1	1	0	0	0	0	0	2, 3, 4
2	1	8	4	0	0	0	6, 11, 15
3	1	4	10	0	0	0	7, 8, 13
4	1	6	0	0	0	3	5, 9, 10
5	1	3	3	0	0	0	20
6	1	8	0	0	0	8	30
7	1	5	4	0	0	0	27
8	1	9	0	1	0	0	12, 19, 27
9	1	2	6	0	0	0	14
10	1	7	0	0	0	1	16, 25
11	1	9	0	5	0	0	20, 26
12	1	2	0	7	0	0	14
13	1	6	4	0	0	0	17, 18
14	1	3	0	8	0	0	17
15	1	9	3	0	0	0	25
16	1	10	0	0	0	5	21, 22
17	1	6	0	0	0	8	22
18	1	5	0	0	0	7	20, 22
19	1	3	0	1	0	0	24, 29
20	1	7	0	10	0	0	23, 25
21	1	2	0	0	0	6	28
22	1	7	2	0	0	0	23
23	1	2	3	0	0	0	24
24	1	3	0	9	0	0	30
25	1	3	4	0	0	0	30
26	1	7	0	0	4	0	31
27	1	8	0	0	0	7	28
28	1	3	0	8	0	0	31
29	1	7	0	7	0	0	32
30	1	2	0	7	0	0	32
31	1	2	0	0	2	0	32
32	1	0	0	0	0	0	

Resource availabilities:	R1	R2	R3	R4
	12	13	4	12

used as a base method in this study. We attempt to improve the BPGA by introducing EDAs. This section therefore describes the key features of the BPGA. Note however that some of these methods are only peculiar to the MRCPSP.

5.3.1 Representation

BPGA uses RKs to represent its *activity scheduling* solutions. RK value (RK_i) serves as a priority value for activity i. An activity with a lower priority value is considered before that with a higher value.

The *mode assignment* solution is represented as a string of integers. Mode k_i defines the mode of execution of an activity i.

Table 5.2: MRCPSP Instance

Activities	Mode	Duration	R1	R2	N1	N2	Successors
1	1	0	0	0	0	0	2, 3, 4
2	1	3	6	0	9	0	5, 6
	2	9	5	0	0	8	
	3	10	0	6	0	6	
3	1	1	0	4	0	8	10, 11
	2	1	7	0	0	8	
	3	5	0	4	0	5	
4	1	3	10	0	0	7	9
	2	5	7	0	2	0	
	3	8	6	0	0	7	
5	1	4	0	9	8	0	7, 8
	2	6	2	0	0	7	
	3	10	0	5	0	5	
6	1	2	2	0	8	0	10, 11
	2	4	0	8	5	0	
	3	6	2	0	0	1	
7	1	3	5	0	10	0	9, 10
	2	6	0	7	10	0	
	3	8	5	0	0	10	
8	1	4	6	0	0	1	9
	2	10	3	0	10	0	
	3	10	4	0	0	1	
9	1	2	2	0	6	0	12
	2	7	1	0	0	8	
	3	10	1	0	0	7	
10	1	1	4	0	4	0	12
	2	1	0	2	0	8	
	3	9	4	0	0	5	
11	1	6	0	2	0	10	12
	2	9	0	1	0	9	
	3	10	0	1	0	7	
12	1	0	0	0	0	0	
Resource availabilities:			R1	R2	N1	N2	
			9	4	29	40	

5.3.2 Bi-Population

The BPGA uses two populations of solutions. One is a population of left justified schedules where activities are scheduled as early as possible. The other is a population of right justified schedules where activities are scheduled as late as possible. To encourage improvement in the scheduling process, parents from one population produce the offspring for the other and vice versa. The concept of scheduling activities forward and then backwards has been considered beneficial in previous research (Lova et al, 2009; Wang and Fang, 2012a). It was however used as a schedule improvement approach which iteratively schedules activities forward and backward until no further improvement can be made. Van Peteghem and Vanhoucke (2010) use the bi-population approach to improve search in a relatively more efficient way.

5.3.3 Preprocessing

The preprocessing procedure of (Sprecher et al, 1997) reduces the search space of feasible solutions by eliminating non-executable and inefficient modes as well as redundant resources.

In a problem with A renewable and B non-renewable resources, a mode k_i is redundant if there exists another mode k_i' with:

$$t_{i,k_i'} \leq t_{i,k_i} \tag{5.1}$$

$$\alpha_{i,k_i',r} \leq \alpha_{i,k_i,r} \; \forall \; r \in [1, |A|] \tag{5.2}$$

$$\beta_{i,k_i',l} \leq \beta_{i,k_i,l} \; \forall \; l \in [1, |B|] \tag{5.3}$$

In eqs. (5.1), (5.2) and (5.3), $t_{i,k_i'}$ and t_{i,k_i} are execution times, $\alpha_{i,k_i',r}$ and $\alpha_{i,k_i,r}$ are renewable resources while $\beta_{i,k_i',r}$ and $\beta_{i,k_i,r}$ are non-renewable resources of activity i respectively performed in the allocated modes k_i' and k_i.

A non-renewable resource l is redundant if:

$$\sum_{i=1}^{n} \beta max_{i,l} \leq \beta max_l \tag{5.4}$$

In eq. (5.4), the maximum non-renewable resource request of l for activity i; $\beta max_{i,l} = max\{\beta_{i,k,l} | k = 1, ..., m_i\}$. $\beta_{i,k,l}$ is the amount of non-renewable resource l required by activity i performed in mode k. βmax_l is the maximum availability of non-renewable resource l.

A mode k is non-executable with respect to non-renewable resource l and renewable resource r if:

$$\sum_{j=1 \; j \neq i}^{n} \beta min_{j,l} + \beta_{i,k,l} > \beta max_l \tag{5.5}$$

$$\alpha_{i,k,r} > \alpha max_r \tag{5.6}$$

In eq. (5.5), the minimum non-renewable resource request of l for activity i is denoted by $\beta min_{i,l} = min\{\beta_{i,k,l} | k = 1, ..., m_i\}$. In eq. (5.6), $\alpha_{i,k,r}$ is the amount of renewable resource r required by activity i performed in mode k while αmax_r is the maximum availability of renewable resource r.

Note that this is a conventional procedure and is peculiar to the MRCPSP.

5.3.4 Improvement of Initial Population

Improvement of initial population is also peculiar to the MRCPSP and refers to the method of improving the quality of mode solutions in the initial population (Lova et al, 2009; Van Peteghem and Vanhoucke, 2010, 2011) and presented in Alg. 2.

Algorithm 2 Improving Feasibility of Mode Solutions

```
1: compute ERR(μ) of randomly generated mode solution μ
2: set count to 0;
3: while ERR(μ) > 0 and count < maxIteration do
4:    select an activity i at random
5:    set new mode solution μ' equal to existing mode solution μ
6:    select a new mode of execution k_i' different from k_i
7:    update mode solution μ' with k_i'
8:    compute ERR(μ')
9:    if ERR(μ') ⩽ ERR(μ) then
10:      set μ = μ'
11:      set ERR(μ) = ERR(μ')
12:   end if
13:   increment count
14: end while
```

The Excess Resource Requirement ERR of mode solution μ is calculated as follows.

$$ERR(\mu) = \sum_{x=1}^{|B|}(max(0, \sum_{i=1}^{n} \beta_{i,k_i,l} - \beta max_l)) \tag{5.7}$$

$ERR(\mu)$ is set to 0, if mode solution μ is feasible. Otherwise, it is set to the difference between the sum of requirements of each non-renewable resources l, for each activity i in its allocated mode m_i, and the maximum availability of l; βmax_l.

The improvement of initial population procedure attempts to improve feasibility by changing the mode of execution k_i of a randomly selected activity i. $k_i{}'$ and μ' respectively denote the new mode of execution of i and a new mode solution. If $ERR(\mu')$ is less than that of the existing mode solution $ERR(\mu)$, then the new mode solution μ' is adopted else the existing mode solution μ is unchanged. This procedure is executed until feasibility is reached or the maximum number of iterations $maxIteration$ is reached. For the BPGA, $maxIteration$ is set to 4 times the problem size.

This procedure is executed on the first population of mode solutions.

5.3.5 Mode Improvement

The mode improvement method is only used when solving the MRCPSP. When the mode improvement method is embedded into the well-known SGS, it is referred to as the ESGS. This is done to improve the quality of mode solutions generated. The mode improvement procedure, as presented in Alg. 3, attempts to improve feasibility as well as the finish time of an activity i.

For an activity i, $|m_i|$ is the maximum number of modes while k_i denotes its existing mode of execution. Mode solution μ' is generated by replacing k_i in μ with $k_i{}'$. The $ERRs$ of the existing mode solution μ and the new mode solution μ' are

Algorithm 3 Mode Improvement Method

```
for j = 1 to |m_i| do
    set new mode solution μ' to existing mode solution μ
    compute ERR(μ)
    if k_i ≠ j then
        set new mode k_i' = j
        update mode solution μ' with k_i'
        compute ERR(μ')
        if ERR(μ') ⩽ ERR(μ) then
            compute f_i'
            if f_i' < f_i then
                μ = μ'
                ERR(μ) = ERR(μ')
                f_i = f_i'
            end if
        end if
    end if
end for
```

calculated. If $ERR(\mu)$ is not higher than $ERR(\mu')$, the procedure executes the next stage. This stage compares the finish times of i using μ' and μ and are respectively denoted by $f_i{}'$ and f_i. If $f_i{}'$ is less than f_i, mode solution μ, $ERR(\mu)$ and f_i are respectively replaced μ', $ERR(\mu')$ and $f_i{}'$.

The mode improvement method encourages activities to run in parallel thereby increasing the possibility of an improvement in makespan (Van Peteghem and Vanhoucke, 2010). In the BPGA, the mode improvement method is applied randomly to 30% of the activities.

5.3.6 Fitness Computation

Fitness is the measure of the quality of a solution. It is important to assign values that discriminate between the quality of solutions in a population.

For the RCPSP, the makespan is directly used as the fitness. In MRCPSP, however, the makespan is not a good discriminatory factor because non-renewable resource infeasible solutions will eliminate feasible solutions in the search.

A common approach in previous study is to penalise infeasible solutions. Hartmann (2001) proposed a fitness function as shown in Eq. (5.8). In this equation, $mak(x)$ is the makespan of a solution x while T is the upper bound of the project's makespan. T is calculated by adding the maximum durations of all activities of a project. $ERR(\mu)$ is calculated as shown in Eq. (5.7). The fitness is set to $mak(x)$ when the solution is feasible with respect to non-renewable resource constraint or penalised according to how much excess non-renewable resource is required by the *mode* solution. With this approach, the fitness of an infeasible solution is always worse than a feasible one.

$$f(x) = \begin{cases} mak(x) & if\ x\ is\ feasible \\ T + ERR(\mu) & otherwise \end{cases} \tag{5.8}$$

Two disadvantages were identified by Alcaraz et al (2003). The first disadvantage is that the makespan of an infeasible solution is not considered. Consequently, solutions with identical $ERR(\mu)$ but different makespan will have the same fitness. The second disadvantage is that the upper bound will be much higher than any makespan. Adding the $ERR(\mu)$ to the already high value gives infeasible solutions almost no chance of survival. Alcaraz et al (2003) therefore proposed the fitness function shown in Eq. (5.9) which was adopted by (Lova et al, 2009; Van Peteghem and Vanhoucke, 2011) and used in the BPGA.

$$f(x) = \begin{cases} mak(x) & if\ x\ is\ feasible \\ UBmak(g) + mak(x) - LB\_CP + ERR(x) & otherwise \end{cases} \tag{5.9}$$

In Eq. (5.9), the Upper Bound makespan at a generation g; $UBmak(g)$ is the maximum makespan of feasible solutions in the population. LB_CP is, however, a Lower Bound Critical Path which is calculated using the lowest duration of each activity. The critical path determines the shortest possible duration required to perform all activities of a project. Note that LB_CP is calculated only once while $UBmak(g)$ is calculated at each generation. The makespan of each infeasible solution x is penalised by adding $UBmak(g)$ and $ERR(x)$ as well as subtracting LB_CP. Unlike the fitness function of Hartmann (2001), the makespan of an infeasible solution contributes to its fitness. Also, the fitness of infeasible solutions will generally be better than that of Hartmann (2001). Note that the solutions will still be poor enough not to eliminate feasible solutions in a population.

5.3.7 Parameters

The set of parameters used by BPGA are summarised in Table 5.3. Van Peteghem and Vanhoucke (2010) found the one-point crossover to be better than other crossover types. Although each solution of a population is selected as the first parent, the second parent is selected by two-tournament selection. Furthermore, Van Peteghem and Vanhoucke (2010) found that the population size was inversely proportional to the number of activities n. They use an improvement rate of 0.3 where only 30% of activities have their modes of execution improved. The mutation rate of *activity* and *mode* solutions (p_{mut_act} and p_{mut_mod}) are respectively set to 0.04 and 0.02. In the BPGA, preliminary algorithm comparison is done with 1000 number of schedules as stopping criteria while more detailed experiments are done using 5000 schedules.

5.4 The RK-EDA for RCPSP

Table 5.3: Parameter Settings of the BPGA

Parameter	**Values**
Crossover type	one-point
Selection	tournament
Population size (ps)	$e^{3.552+\frac{22.72}{n}}$
Mode improvement rate	0.3
p_{mut_act}	4%
p_{mut_mod}	2%
Number of evaluations	$1000, 5000$
Number of Runs	1

In this section, we present RK-EDA which uses the bi-population procedure of the BPGA. We also present the parameters used and compare results with previous approaches to solving the RCPSP.

5.4.1 Workflow of RK-EDA for RCPSP

Alg. 4 presents a bi-population RK-EDA for the RCPSP. Line 1 of the algorithm initialises the population of left-justified schedules POP_L with randomly generated solutions. Each solution of POP_L is then evaluated in line 2. As shown in line 3, the probabilistic model of right-justified schedules M_R is generated based on the best b solutions in POP_L. Similarly, the probabilistic model of left-justified schedules M_L is generated based on the best b solutions in POP_R (line 16). Note that the makespan is used as the fitness in this study. Solutions with the lowest makespan in POP_L and POP_R are respectively denoted by $best_L$ and $best_R$. For the first iteration, $x = 1$ of POP_L/POP_R, the best solution from the other population ($best_R/best_L$) is retained as the first offspring. $best_L$ is converted into a right-justified schedule before it is added to POP_R (line 5) while $best_R$ is converted to a left-justified schedule before it is added to POP_L (line. 15). For subsequent iterations $x \in [2, n]$, we generate new offspring solutions $RChild/LChild$ by sampling M_R/M_L (line 10/21). The parent solution POP_{R_x}/POP_{L_x} is replaced by $RChild/LChild$ (line 13/24). Once the stopping criteria is satisfied, the overall best solution from both populations is returned.

5.4.2 Experimental Settings

In this section, we present the problem sets and parameter settings used in this chapter. An experimental justification for the selection of parameters is also presented.

5.4.2.1 Problem Set and Performance Criteria

Although the PSPLIB for RCPSP consists of J30, J60, J90 and J120. The most frequently used problem sets are J30, J60 and J120. These problems respectively consist of 480, 480 and 600 problem instances. J30, J60 and J120 are composed of 30,

Algorithm 4 RK-EDA for RCPSP

generate initial population POP_L
evaluate POP_L as left-justified schedules
build probabilistic model M_R from $|POP_L|$
repeat
  set $best_L$ as best solution in POP_L
  for $x = 1$ to $|POP_L|$ **do**
    if $x = 1$ **then**
      set $RChild = best_L$
    else
      sample M_R to produce $RChild$
    end if
    evaluate $RChild$ as a right justified schedule
    set POP_{R_x} as $RChild$
  end for
  set $best_R$ as best solution in POP_R
  build probabilistic model M_L from POP_R
  for $i = 1$ to $|POP_R|$ **do**
    if $i = 1$ **then**
      set $LChild = best_R$
    else
      sample M_L to produce $LChild$
    end if
    evaluate $LChild$ as a left justified schedule
    set POP_{L_x} as $LChild$
  end for
  build probabilistic model M_R from POP_L
until stopping criteria is satisfied
return overall best solution

60 and 120 non-dummy activities to be scheduled. Each project has two additional activities which are the dummy start and finish activities. The start activity has no predecessor while the finish activity has no successor, they also have no resource requirements. This study does not consider the J90 because most previous approaches do not present results for this problem set.

The most frequently used performance measure for the RCPSP is the APD (presented in Eq. (2.15)). In J30, the optimal values presented represents the best-known makespan while the Critical Path-Based Lower Bound (CPBLB) is used for J60 and J120. CPBLB is calculated by relaxing the resource constraints; it is only based on precedence and duration of activities.

5.4.2.2 RK-EDA: Parameter Settings and Preliminary Results

Although previous studies are based on only one run, we present results averaged over 10 runs. We use the student t-test and a confidence interval of 95% to measure statistical significance during parameter tuning. However, we are unable to test for statistical significance when RK-EDA is compared with other algorithms because previous approaches present results based on one run only.

To be able to parametrise RK-EDA for the RCPSP, a range of population sizes and variance values are examined. An attempt is made to improve each parameter based on the best set of parameters used in the previous chapter. Population sizes 30, 60 and 120 were used across the three problem sets while setting the truncation size to 10% and variance value to 0.0025. The truncation size and variance value are parameters that showed good performance in the previous chapter. As shown in Table 5.4, a population size of 60 performed best across all three problems.

Table 5.4: Accessing a range of population sizes

Problems sets	**30**	**60**	**120**
J30	0.63 (0.04)	**0.55 (0.04)**	0.56 (0.03)
J60	12.24 (0.07)	**12.09 (0.03)**	12.25 (0.04)
J120	36.85 (0.04)	**36.53 (0.06)**	37.00 (0.06)

While the population size is set to 60 and variance again set to 0.0025, truncation sizes 10%, 25% and 40% of the population size were applied to all problems. Results are presented in Table 5.5

Table 5.5: Accessing a range of truncation values

Problems sets	**10%**	**25%**	**40%**
J30	**0.55 (0.04)**	0.59 (0.02)	0.64 (0.03)
J60	**12.09 (0.03)**	12.17 (0.05)	12.26(0.03)
J120	36.53 (0.06)	**36.29 (0.04)**	36.44 (0.06)

Although the 10% truncation size was the best for J30 and J60, this is not the

case for J120 where truncation size of 25% presented the best results. These values are used for further experiments in this study.

Furthermore, variance values 0.0025, 0.0100, 0.0225 and 0.0400 were applied to the three problem sets. These values are taken from the range that produced promising solutions in the previous two chapters.

Table 5.6: Accessing a range of variance values

Problems sets	**0.0025**	**0.0100**	**0.0225**	**0.0400**
J30	0.55 (0.04)	0.33(0.04)	0.35 (0.03)	**0.33 (0.02)**
J60	**12.09 (0.03)**	12.36 (0.04)	12.16 (0.05)	12.59 (0.03)
J120	**36.53 (0.06)**	38.06 (0.06)	37.39 (0.07)	38.62 (0.07)

For J60 and J120, increasing the variance value from 0.0025 did not improve the performance of the algorithm. However, the variance of 0.0400 improved the performance of the algorithm on J30.

Based on the results presented in Tables 5.4, 5.5 and 5.6, the final set of parameters for RK-EDA is presented in Table 5.7. Although preliminary experiments were based on 1000 fitness evaluations, final results will be compared based on 5000 fitness evaluations. Note that it is common practice to use different parameters for different problem sets.

Table 5.7: Parameter Values for RK-EDA

Parameters	**J30**	**J60**	**J120**
Population Size (p_s)	60	60	60
Truncation Size (b)	0.1 * p_s	0.1 * p_s	0.25 * p_s
Variance (σ)	0.0400	0.0025	0.0025
Stopping Criteria	5000 fitness evaluations		
Number of Runs	10 runs		

5.4.3 Results

In this section, detailed results for RK-EDA on the problem sets are presented. RK-EDA is also compared with existing methods.

5.4.3.1 RK-EDA Results

Results of RK-EDA based on ten runs on J30, J60 and J90 are presented in Table 5.8. We present the minimum, maximum, mean and standard deviation APD for each problem set in columns min, max, avg and stdev.

Table 5.8: RK-EDA Results

Problem sets	min	max	avg	stdev
J30	0.17	0.21	0.19	0.01
J60	11.59	11.71	11.65	0.04
J120	33.88	34.03	33.97	0.05

5.4.3.2 Comparing RK-EDA with Existing Algorithms

Some of the leading approaches of solving RCPSP are ACOSS, which combines a local search strategy, Ant Colony Optimization (ACO), and a Scatter Search (SS) (Chen et al, 2010), Multi-Agent Optimisation Algorithm (MAOA) (Zheng and Wang, 2015), Shuffled Frog Leaping Algorithm (SFLA) (Fang and Wang, 2012), GA (Debels and Vanhoucke, 2005, 2007; Valls et al, 2008) and Scatter Search (SS) (Debels et al, 2006; Ranjbar et al, 2009). Some more recent methods are the local search procedure in (Chand et al, 2017) and GA in (Goncharov and Leonov, 2017). Other popularly cited research are EDA (Wang and Fang, 2012b), PSO (Chen, 2011), GA (Alcaraz and Maroto, 2001; Hartmann, 2002), SA (Bouleimen and Lecocq, 2003) and other heuristics Kolisch and Drexl (1996); Tormos and Lova (2003).

Table 5.9: APD after 5000 schedules

Algorithms	**J30**	**J60**	**J120**
DBGA (Debels and Vanhoucke, 2007)	0.04	10.95	**32.18**
BPGA (Debels and Vanhoucke, 2005)	0.06	11.00	32.34
ACOSS (Chen et al, 2010)	0.06	10.98	32.48
GA (Goncharov and Leonov, 2017)	-	10.87	32.51
GA (Valls et al, 2008)	0.06	11.10	32.54
MAOA (Zheng and Wang, 2015)	0.06	**10.84**	32.64
SS (Debels et al, 2006)	0.11	11.10	33.10
SFLA (Fang and Wang, 2012)	0.21	10.87	33.20
SS (Ranjbar et al, 2009)	**0.03**	11.07	33.24
EDA (Wang and Fang, 2012b)	0.14	11.43	33.61
PSO (Chen, 2011)	0.14	11.43	33.88
RK-EDA	0.19	11.65	33.97
Other heuristic (Chand et al, 2017)	0.07	11.25	34.04
other heuristic (Tormos and Lova, 2003)	0.14	11.72	35.30
GA (Hartmann, 2002)	0.22	11.70	35.39
other heuristic (Kolisch and Drexl, 1996)	0.15	11.82	35.62
GA (Alcaraz and Maroto, 2001)	0.12	11.86	36.57
SA (Bouleimen and Lecocq, 2003)	0.23	11.90	37.68

Table 5.9 presents the APD of each algorithm on J30, J60 and J120. Results are sorted according to performance on the largest problem sets (J120). Missing results are represented with "-", this is when authors have not applied their algorithm to

a given problem set. This is seen in (Goncharov and Leonov, 2017) where results are only presented for J60 and J120. Also, the lowest APD for each problem set is presented in bold. We are unable to do statistical significance tests because existing algorithms do not report variance across many runs.

The SS of Ranjbar et al (2009) presents the lowest APD on J30 while MAOA (Zheng and Wang, 2015) presents the lowest APD on J60. The DBGA presents the lowest APD on J120. Although RK-EDA is not one of the leading methods in this study, the difference between its results and other leading algorithms is not more than 1.79%. Also, RK-EDA presents a relatively simple technique requiring no local search or other improvement methods which are conventionally used when applying algorithms to the RCPSP.

The rest of this chapter will focus on solving the MRCPSP.

5.5 The BPGA-EDA for MRCPSP

BPGA-EDA replaces the mechanism of generating mode assignment solutions in BPGA with an EDA. BPGA-EDA generates new mode assignment solutions by sampling a probabilistic model rather than using crossover. This section presents the probabilistic model for generating mode solutions and the overall workflow of BPGA-EDA

5.5.1 Probabilistic Model for Mode generation

The probability matrix of *mode* solutions PM_{mod} is generated using a population S made up of the best b solutions of a given population.

$$PM_{mod} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nm} \end{pmatrix} \tag{5.10}$$

In PM_{mod}, each row, $p_{i1} + \cdots + p_{im} = 1$. The probability that activity i will be performed in mode k; p_{ik} is calculated as $\frac{count(i,k)}{b}$ where $count(i,k)$ denotes the number of solutions in S where activity i is performed in mode k. For example, if the truncation size b is set as 10 and amongst the selected solutions, activity 2 was executed in mode 1: six times, mode 2: four times and no occurrence of mode 3. The probability values p_{21}, p_{22} and p_{23} will be 0.6, 0.4 and 0 respectively.

Note that modes of activities that are impossible or have been eliminated during the preprocessing stage will always have probability scores equal to 0. For this reason, unlike existing EDAs applied to MRCPSP, BPGA-EDA does not initialise its probabilistic model with equal probabilities. With this approach, inefficient and non-executable modes will have no chance of being sampled at any generation.

In this study, the Sum of Durations SUD proposed in (Van Peteghem and Van-

houcke, 2011) is adopted for initialising PM_{mod}.

$$SUD = \sum_{i=1}^{n} t_{i,k_i} \tag{5.11}$$

As shown in Eq. (5.11), SUD is the sum of the execution times t_{i,k_i} of all activity i in their assigned mode of execution k_i. SUD is a measure of mode solutions only and has been reported to have a strong correlation with fitness (Van Peteghem and Vanhoucke, 2011).

To create a bias towards modes of execution with short execution times, the initial population is ranked using SUD rather than fitness. Truncation is applied, and the selected solutions are used to initialise the probability matrix PM_{mod}. The probability matrix of subsequent generations is however generated using fitness.

Since BPGA-EDA uses a Population-Based Incremental Learning (PBIL) style, the model uses a learning rate lr as shown in Eq. (5.12).

$$p_{ik}(g) = (lr * p_{ik}(g)) + ((1 - lr) * p_{ik}(g - 1)) \tag{5.12}$$

In eq. (5.12), $p_{ik}(g)$ and $p_{ik}(g-1)$ are p_{ik} values at generations g and $g-1$ respectively. The probabilistic model is updated at the end of each generation until the stopping criteria is met.

5.5.2 BPGA-EDA Workflow

BPGA-EDA starts by executing the conventional pre-processing procedure described in Section 5.3.3.

The BPGA-EDA is formally defined in Alg. 5. The algorithm uses two populations of solutions. The population of left-justified schedules is denoted by POP_L while that of right-justified schedules is denoted by POP_R. Left-justified schedules are generated using the well known (forward) SGS while right-justified schedules are generated with the backward SGS. SGS schedules each activity as soon as its predecessors have been scheduled and there are enough resources to execute it. In backward SGS, each activity is scheduled when all its successors have been performed, and there are enough resources to perform it.

The algorithm starts by generating *activity* and *mode* solutions for POP_L at random. The *mode* solutions are further improved using the improvement of initial population technique presented in Section 5.3.4. Each solution in POP_L is then evaluated.

BPGA-EDA uses an EDA to generate new mode solutions. The best b solutions are selected from a population POP_L to form S. Population S is used to generate a probabilistic model of *mode* solutions PM_{mod} as described in Section 5.5.1. Probabilistic models of *mode* solutions for left justified schedules and right justified schedules are respectively denoted by LPM_{mod} and RPM_{mod}. These models are initialised with PM_{mod}. Subsequently, LPM_{mod} is updated using POP_L while RPM_{mod} is updated using POP_R. New mode solutions for POP_L and POP_R are respectively generated by sampling LPM_{mod} and RPM_{mod}.

Activity solutions are generated by crossover as done by the BPGA. Each individual i in POP_L or POP_R is selected as the first parent ($parent_1$) while the second parent ($parent_2$) is selected by tournament selection. To generate a new population of solutions which replaces the old (POP_L or POP_R), the activity solution of each $parent_1$ and a selected $parent_2$ are crossed over. The one-point crossover is used in this study in the same way as the BPGA. Note that POP_L is used to generate the *activity* solutions for POP_R and vice versa.

In BPGA-EDA, parameter: $\rho \in [0, 1]$ is introduced to determine the rate at which EDA will be applied for generating *mode* solutions. We use the following notation BPGA-EDA$_\rho$ to express the type of BPGA-EDA used. BPGA-EDA$_0$ is equivalent to the BPGA (i.e. when $\rho = 0$, EDA is not used), but BPGA-EDA$_1$ indicates that all mode solutions are generated by the EDA. Where the EDA component of the BPGA-EDA is not invoked, *mode* solutions are generated by crossover just like the *activity* solutions. Note that r is a random number between 0 and 1.

Once offspring solutions are generated, the respective *activity* and *mode* solutions go through mutation at specified rates p_{mut_act} and p_{mut_mod}. For *activity* solutions, a randomly selected activity takes on a randomly generated priority while in *mode* solutions, the mode of a randomly selected activity is changed to another possible mode of execution. Although two offspring are generated at each iteration of solution generation, the best offspring from POP_L or POP_R replaces individual i in POP_R or POP_L. The best solutions in POP_L or POP_R are however not replaced except the new offspring is better. Once the algorithm reaches optimal or exhausts the maximum number of schedules, the overall best solution is returned.

5.5.3 Analysis of EDA for Mode Assignment

Since the focus of BPGA-EDA study is on generating high-quality mode solutions, it is important for the proposed algorithm to be able to generate non-renewable feasible solutions. To do this, a measure of complexity relating to the ease of generating a *mode* feasible solution is proposed.

5.5.3.1 Measure of Complexity

There are many measures of complexity in literature, some relate to *activity* solutions while others relate to *mode* solutions. Three complexity measures that relate to *mode* solution are resource factor, resource strength and resource constrainedness (Kolisch and Sprecher, 1997). These measures alongside network complexity were used to generate problem instances of the PSPLIB. The resource strength and resource constrainedness are considered the most widely used in literature (Van Peteghem and Vanhoucke, 2011). Resource strength was however used by Van Peteghem and Vanhoucke (2011) because there was no standard formula for the resource constrainedness.

Resource strength is the average measure of resources requested per activity. It is a measure of the scarceness of a resource but not limited to non-renewable resources. This measure has been used for generating problem instances rather than sampling from existing problem sets (Van Peteghem and Vanhoucke, 2011, 2014). Van Pe-

Algorithm 5 BPGA-EDA Workflow

execute preprocessing procedure (see Section 5.3.3)
generate initial population POP_L
execute improvement of initial population (see Section 5.3.4)
apply SGS/ESGS to each solution in POP_L.
calculate SUD and fitness for solutions in POP_L (see eqs. (5.11) and (5.9))
repeat
  if $\rho > 0$ **then**
    select best $b < |POP_L|$ solutions using SUD to create S.
    build probabilistic model PM_{mod} from S (see Section 5.5.1)
    initialise probabilistic models LPM_{mod} and RPM_{mod} with PM_{mod}
  end if
  for $i = 1$ to $|POP_L|$ **do**
    set individual i in POP_L as $parent_1$
    generate $parent_2$ by tournament selection from POP_L
    if $r < \rho$ **then**
      perform crossover to generate two offspring *activity* solutions
      sample LPM_{mod} to produce two offspring *mode* solutions
    else
      perform crossover to generate two offspring solutions
    end if
    perform *activity* mutation with probability p_{mut_act}
    perform *mode* mutation with probability p_{mut_mode}
    apply backward SGS/ESGS and compute fitness of the offspring
    update POP_R with the best offspring
  end for
  for $i = 1$ to $|POP_R|$ **do**
    set individual i in POP_R as $parent_1$
    generate $parent_2$ by tournament selection from POP_R
    if $r < \rho$ **then**
      perform crossover to generate two offspring *activity* solutions
      sample RPM_{mod} to produce two offspring *mode* solutions
    else
      perform crossover to generate two offspring solutions
    end if
    perform *activity* mutation with probability P_{mut_act}
    perform *mode* mutation with probability P_{mut_mode}
    apply SGS and compute fitness of the offspring
    update POP_L with the best offspring
  end for
  if $\rho > 0$ **then**
    update LPM_{mod} using POP_L
    update RPM_{mod} using POP_R
  end if
until stopping criteria satisfied
return overall best solution

teghem and Vanhoucke (2014) noted that the feasibility of a problem could not be ensured even with low resource strength. This suggests that low resource strength will not particularly correlate to the ease of generating non-renewable resource feasible solutions. For this reason, the Relative non-renewable Resource Availability (RRA) is proposed in this chapter. This is a measure that relates to the ease of generating non-renewable resource feasible solutions. Since problem instances become simplified after preprocessing, the RRA is calculated after preprocessing. This means that instances with redundant non-renewable resources will have zero complexity scores. This is because it is impossible to generate infeasible solutions for these instances.

To generate the RRA score, the following formula is used.

$$RRA = Max\left(\frac{\sum_{i=1}^{n}\frac{\sum_{k=1}^{m_i}\beta_{i,k,1}}{|m_i|}}{\beta max_1}, \cdots, \frac{\sum_{i=1}^{n}\frac{\sum_{k=1}^{m_i}\beta_{i,k,|B|}}{|m_i|}}{\beta max_{|B|}}\right) \tag{5.13}$$

For a *mode* solution to be feasible, the sum of each non-renewable resource utilisation for all activities must be less than the maximum availability of that resource. RRA measure the ratio of utilisation to the availability of non-renewable resources. The higher the RRA, the more difficult it is to generate feasible solutions. In Eq. (5.13), the amount of non-renewable resource l required by activity i performed in mode k, $\beta_{i,k,l}$ is averaged across all possible modes of execution of each activity. The result is divided by the maximum availability of l, βmax_l. This is done for all $l \in |B|$ and the highest value is returned as the RRA of each problem instance. This way, the complexity of a problem is defined by its most constrained non-renewable resource.

5.5.3.2 Feasibility of Mode Solutions: Comparing BPGA with BPGA-EDA

In this section, the relative performance of EDA compared to GA for generating non-renewable resource feasible *mode* solutions is assessed. Some preliminary tests are performed to show the rate of feasible solutions produced in a generation of BPGA and BPGA-EDA. In Figure 5.1, the GA is based on crossover while EDA is based on sampling a probabilistic model (based on fitness). The charts on the left show the RRA across the problem instances of J10, J20 and J30. The charts on the right show the feasibility rate of *mode* solutions generated by BPGA and BPGA-EDA across the same problem instances. Note that problems are ordered in increasing order of RRA. Also, feasibility score is between 0 and 1, score 0 means that there is no feasible solution in the population while 1 means all the solutions are feasible.

Figure 5.1 shows that both algorithms produce less feasible solutions as the complexity increases. However, EDA generates a relatively higher number of feasible solutions for more complex problems than the GA.

5.5.4 Experimental Settings

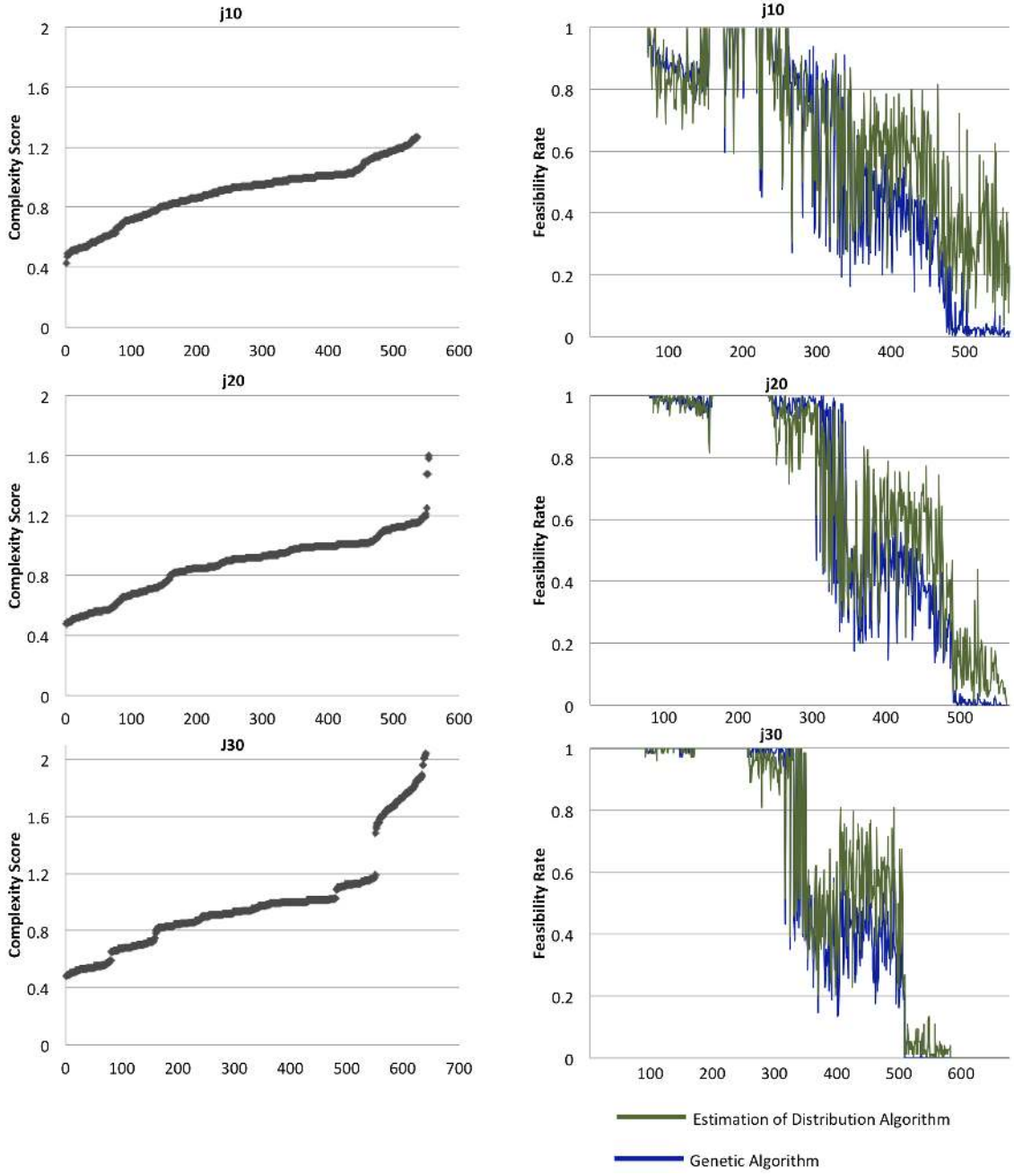

Figure 5.1: Feasibility of BPGA compared to BPGA-EDA

In this section, we present the experimental justification and parameter settings for BPGA-EDA. The conventionally used J10, J20 and J30 problem sets of the PSPLIB described in Chapter 2 are used to assess the BPGA-EDA.

5.5.4.1 Stopping Criterion and Performance Measure

In the applications of meta-heuristics to the MRCPSP, the conventional stopping criterion is the number of schedules generated. For ease of comparison, the same criterion is used in this chapter. Note that a schedule refers to a single time (start and finish) assignment for each activity of a project. However, some local search methods like the makespan improvement (Lova et al, 2009; Van Peteghem and Vanhoucke, 2010) may require more than one time assignment for an activity. To cater for this, Lova et al (2009) calculate the number of schedules by dividing the number of times the activities of a project have been assigned a start time by the total number of activities. This implies that each change in the start time of an activity contributes to a fraction of a schedule. For instance, if each activity of a project has been assigned a feasible start time twice, the number of schedules will be equal to 2. This method of calculating number of schedules is also used in (Van Peteghem and Vanhoucke, 2010, 2014). The same method is also adopted in this chapter.

Furthermore, the most common performance measure is the average percentage deviation from optimal (APD) which is presented in Eq. (2.15). Where there are no optimal values, the critical path based lower bound (CPBLB) is used instead of the optimal. The CPBLB is estimated using the critical path based on the modes with the least durations. We average the APD across all instances of each problem set.

5.5.4.2 Parameter Settings

Like the BPGA, most previous algorithms are evaluated based on only one run per problem instance. This limits the computational cost associated with running experiments. However, meta-heuristics are non-deterministic and results may vary from one run to another. It is therefore important to measure variance in algorithm performance. The fact that algorithms behave differently across instances of the same problem set has been established in 5.1. For this reason, there is a need to repeat runs so that the variance can be captured and used to asses the algorithms. However, parameter tuning based on repeated runs is computationally expensive. To do this efficiently, this section proposes a method of selecting instances for parameter tuning.

5.5.4.3 Variance in Results

To show the claim regarding the variance between different runs of the BPGA, results based on ten runs on J10 are generated. The APD values varied between 0.018 and 0.096 as shown in Figure 5.2. To make the most meaningful comparisons, the average performance of BPGA-EDA with variance alongside best performance over several runs are reported in this chapter.

5.5.4.4 Problem Instance Selection Approach

Considering the need for repeated runs and the number of instances in the problem sets, the computational cost of experiments grows very quickly. To be able to make comparisons based on several runs with limited computational cost, we make use of

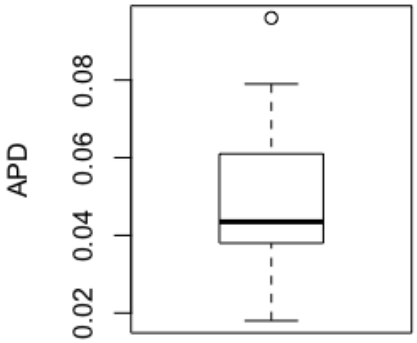

Figure 5.2: Results for BPGA on J10 - APD

smaller number of instances. To ensure that the selected instances are representative of the problem sets, we sort them by RRA and select uniformly across the distribution.

To generate x selected instances, the aim is to divide the instances in a problem set into x complexity groups and select the instance with the highest RRA from each group.

Algorithm 6 Sample Set Generation Mechanism

order problem set of size d in ascending order of RRA
initialise the required number of instances x
create an array $dataGroupSizes$ of size x
$leftOver = d\%x$
for $i = 1$ to d **do**
    $dataGroupSizes[i] = d/x$
    if $leftOver > 0$ **then**
        add 1 to $dataGroupSizes_i$
        subtract 1 from $leftOver$
    end if
end for
define a variable j=0
for $i = 1$ to x **do**
    add $dataGroupSizes[i]$ to j
    select the j^{th} element of the problem set
end for

As shown in Alg. 6, the size $dataGroupSizes$ of each group is calculated as the integer division of the number of instances d in a problem set and x. We distribute the remainder $leftOver$ over the earlier groups. This is so that groups of easier

problems are the ones that get the bigger group sizes where x does not divide d without remainders. After setting the *dataGroupSizes*, instances in a problem set are ordered in increasing order of RRA. Problems are sampled according to j which is a cumulative value of *dataGroupSizes*. This way, only the last problems of each group (hardest) are sampled.

For example, 20 instances are sampled from J10. Figure 5.3 shows the complexity distribution of J10, where the dots indicates the selected instances.

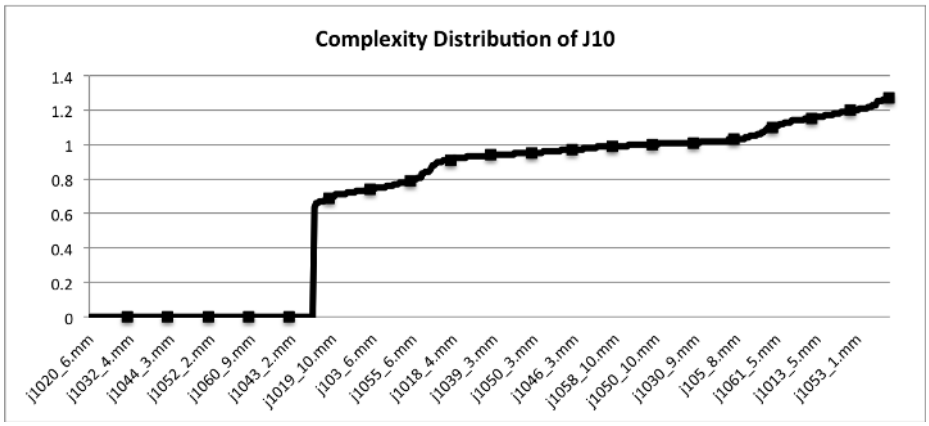

Figure 5.3: Sampling along the complexity distribution of J10

Only 20 instances are selected from the J10, J20 and J30 to parametrise the BPGA-EDA. This makes it possible to examine a wider range of parameters than would have been possible if all problem instances are used. Selected instances are presented in Table A.7.

5.5.4.5 BPGA-EDA: Parameter Settings and Preliminary Results

BPGA-EDA uses most of the recommended parameters of BPGA as presented in Table 5.3. The same population size, crossover type, selection method, mode improvement rate and mutation rates are used. However, results are based on ten runs and 5000 schedules as the stopping criteria. Although we note that this may not be the best set of parameters for the EDA aspect of BPGA-EDA, we retain them for simplicity of parameter tuning.

Furthermore, since EDAs and GAs traverse search spaces in different ways, there has been research on combining both algorithms (Pena et al, 2004; Zhang et al, 2004). We introduced a variable ρ which defines how often solutions are generated with the EDA. Apart from generating all *mode* solutions with EDA ($\rho = 1.0$), we considered combining crossover (GA) and sampling of the probabilistic model (EDA) equally for the generation of *mode* solutions ($\rho = 0.5$). Note that $\rho = 0$ is equivalent to BPGA. Although more values of ρ can be tried, it is out of the scope of this study.

Apart from ρ, BPGA-EDA requires two additional parameters which are b and lr. We vary b between 10% and 50% using a step size of 10 and lr from 0.1 to 1.0 using a step size of 0.1.

BPGA-EDA is applied to sample sets from J10, J20 and J30. The APD values are averaged across 10 runs. The Friedman ranking test (Theodorsson-Norheim, 1987) is used to select the set of parameters used in this chapter. Average ranks of parameters are presented in Tables A.1 to A.6. It was observed that b set to 10% of ps was consistent amongst the best-ranked parameters. However, lr varied a lot but generally greater than or equal to 0.4. Table 5.10 shows the b and lr for BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$.

Table 5.10: BPGA-EDA Parameters based on ESGS- $b(\%ps)/lr$

Problem Sets	BPGA-EDA$_{0.5}$	BPGA-EDA$_1$
J10	10/0.9	10/1.0
J20	10/0.6	10/0.4
J30	10/0.7	10/0.5

Since the aim of this study is to access performance of EDA on the mode component of the problem. We also compare without the use of mode improvement method (i.e. standard SGS). This will help to access the independent performance of the algorithms. As shown in Tables A.1 to A.6, we observe that the algorithm is more sensitive to parameters when the standard SGS is used. As shown in Table 5.11, the lr for BPGA-EDA$_1$ remains the same but requires less selective pressure as the problem size increases. BPGA-EDA$_{0.5}$, also requires higher b values for $J20$ and $J30$. Higher lr are required for $J10$ and $J20$ but lower for $J30$. Note that many parameters are often ranked the same and we have used the number of schedules required to reach optimal as a discriminating criteria.

Table 5.11: BPGA-EDA Parameters based on SGS - $b(\%ps)/lr$

Problem Sets	BPGA-EDA$_{0.5}$	BPGA-EDA$_1$
J10	10/1.0	10/1.0
J20	20/0.8	20/0.4
J30	20/0.5	30/0.5

5.5.5 Results

In this section, results from comparing the BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$ with our implementation of the BPGA are presented. It was shown that there are variations in multiple runs of the BPGA but results for only one run is reported in literature. For this reason, we use our implementation of the BPGA so that comparison can be made based on average and standard deviations based on several runs. Also, for a fairer assessment, comparison has been made based using the same algorithm with just an additional parameter ρ to determine how much of EDA is used for *mode* generation. All algorithms are therefore based on same conditions and implementation.

In Tables 5.12 and 5.13, we present the average of APD and the standard deviation (in brackets) for ten runs of our implementation of the BPGA, BPGA-EDA$_{0.5}$ and

Table 5.12: Results based on SGS - average APD (Standard deviation) of ten runs

Problem sets	BPGA	BPGA-EDA$_{0.5}$	BPGA-EDA$_1$
J10	0.61 (0.08)	**0.19 (0.05)**	**0.20 (0.04)**
J20	2.34 (0.05)	**1.21 (0.06)**	**1.60 (0.07)**
J30	17.89 (0.18)	**14.67 (0.06)**	**15.06 (0.07)**

Table 5.13: Results based on ESGS - average APD (Standard deviation) of ten runs

Problem sets	BPGA	BPGA-EDA$_{0.5}$	BPGA-EDA$_1$
J10	0.05 (0.02)	**0.03 (0.02)**	**0.03 (0.02)**
J20	0.88 (0.04)	**0.53 (0.04)**	**0.69 (0.04)**
J30	14.41 (0.05)	**13.68 (0.03)**	**13.87 (0.07)**

BPGA-EDA$_1$ on the J10, J20 and J30. Results that are statistically better than the BPGA are displayed in bold. In this section, the student t-test with confidence interval 95% is used to measure statistical significance.

Comparison Based on the SGS: Table 5.12 shows results using the SGS for schedule generation. The BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$ have significantly lower APD than the BPGA on all the problem sets: J10, J20 and J30. Using EDA for *mode* generation significantly improves the performance of the BPGA.

Comparison Based on the ESGS: Table 5.13 shows results that are based on the ESGS. The BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$ produce statistically lower APD than the BPGA on J10, J20 and J30. Again, a significant improvement is achieved by using EDA for *mode* generation.

Impact of ESGS: SGS extended by the local search method (mode improvement) not only improves results produced by BPGA but also improves the results of BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$. Although there is a clear advantage of hybridising EDA with BPGA (BPGA-EDA), the mode improvement local search method cannot be eliminated by applying the BPGA-EDA without compromising the quality of results produced. The results in Tables 5.12 and 5.13 therefore also show that the BPGA-EDA$_{0.5}$ and BPGA-EDA$_1$ without mode improvement are worse than the BPGA with mode improvement.

BPGA-EDA$_{0.5}$ and BPGA-EDA$_{1.0}$: BPGA-EDA$_{0.5}$ produces significantly better APD than the BPGA-EDA$_{1.0}$ on the J20 and J30 but not significantly better on the J10. This is true when the SGS or the ESGS is used. In general, the BPGA-EDA$_{0.5}$ performs better than the BPGA-EDA$_{1.0}$ as shown in Tables 5.12 and 5.13. The competitive performance of the hybridised GA and EDA has been attributed to the exploration ability of the GA and the exploitation ability of EDA (Pena et al, 2004; Zhang et al, 2004).

For the purpose of comparison with existing published values, Table 5.14 shows the BPGA-EDA's best of ten runs as well as the published value of the BPGA.

The results in Table 5.14 are similar and it is not clear which approach is better than which. We assert that results averaged over several runs provide a fairer assessment of the algorithms.

Table 5.14: Results based on ESGS - average % deviation from optimum - best of ten runs

Problem sets	BPGA	BPGA-EDA$_{0.5}$	BPGA-EDA$_1$
J10	0.01	0.01	0.00
J20	0.57	0.46	0.62
J30	13.75	13.73	13.61

Following these promising results, a full EDA approach to solving the MRCPSP is presented in Section 5.6.

5.6 The BPEDA for MRCPSP

BPEDA combines RK-EDA for generating activity solution presented in Section 5.4 with the mechanism of generating *mode assignment* solutions in Section 5.5. This section presents the algorithmic details of BPEDA and the experimental settings used in this chapter.

5.6.1 Workflow for the BPEDA

BPEDA also uses the bi-population approach (POP_L and POP_R). As shown in Alg. 7, we start by executing the conventional preprocessing procedure of Sprecher et al (1997) to eliminate non-executable and inefficient modes as well as redundant non-renewable resources. The preprocessing procedure is presented in Section 5.3.3. After this, POP_L is randomly generated. The conventional feasibility improvement of initial mode solutions (Van Peteghem and Vanhoucke, 2010) is executed to improve solutions in the initial POP_L. This procedure reduces the number of resource infeasible solutions in the initial population. This is done by changing the modes of randomly selected activities to those that reduce the infeasibility with respect to non-renewable resources. The solutions in this population are then evaluated using the SGS.

A selected population S which contains the best b (truncation size) solutions in POP_L is generated. Probabilistic models of activity solutions and mode solutions, PM_{act} and PM_{mod} are generated based on POP_L. Note that PM_{act} is generated based on the procedures of RK-EDA presented in Alg. 1 while PM_{mod} is generated as described in Section 5.5.1.

Probabilistic models LPM_{mod} and RPM_{mod} which are respectively used for generating mode solutions in POP_L and POP_R are both initialised with PM_{mod}. Probabilistic model RPM_{act} for generating activity solutions in POP_R is initialised with PM_{act}. Probabilistic model LPM_{act} for generating activity solutions in POP_L is however generated based on b most promising solutions in POP_R. In a similar way, POP_R is updated based on b most promising solutions in POP_L.

At each generation, the solution that generates the best schedule in POP_L, $best_L$ is rescheduled as a right-justified solution and set as an offspring of POP_R. Similarly, the best solution in POP_R, $best_R$ is rescheduled as a left justified solution and set as an

Algorithm 7 Proposed EDA for the MRCPSP

execute preprocessing procedure
generate initial population POP_L
evaluate POP_L with SGS/ESGS
select best $b < |POP_L|$ solutions to form S.
build probabilistic models PM_{act} and PM_{mod} from S
initialise $LPM_{mod} = RPM_{mod} = PM_{mod}$
initialise $RPM_{act} = PM_{act}$
repeat
  set $best_L$ as best solution in POP_L
  for $i = 1$ to $|POP_L|$ **do**
    if $i = 1$ **then**
      set POP_{R_i} as the genome of $best_L$
    else
      sample RPM_{act} and RPM_{mod} to produce POP_{R_i}
      apply backward SGS to POP_{R_i}
    end if
    update POP_R with POP_{R_i}
  end for
  build probabilistic model LPM_{act} from POP_R
  Update RPM_{mod} with POP_R
  set $best_R$ as best solution in POP_R
  for $i = 1$ to $|POP_R|$ **do**
    if $i = 1$ **then**
      set POP_{L_i} as the genome of $best_R$
    else
      sample LPM_{act} and LPM_{mod} to produce POP_{L_i}
      apply SGS/ESGS to POP_{L_i}
    end if
    update POP_L with POP_{L_i}
  end for
  build probabilistic model RPM_{act} from POP_L
  update LPM_{mod} with POP_L
until stopping criteria satisfied
return overall best solution

offspring of POP_L. To produce the remaining population of right/left justified schedules, RPM_{act} and RPM_{mod} / LPM_{act} and LPM_{mod} are sampled to produce POP_{R_i} / POP_{L_i}. POP_{R_i} / POP_{L_i} is evaluated using backward SGS/ forward SGS where activities are scheduled as late as possible/as early as possible within resource and precedence feasibility. Also, while the forward SGS schedules activities in increasing order of their RKs, backward SGS schedules in decreasing order of RKs.

Also, once each solution has been scheduled, the RKs of that solution is updated to respect the order in which the activities were performed. This is because the order depicted by the RKs would not be the same as the order of execution because of resource and precedence constraints. We therefore respectively rank the activities of solutions in POP_L and POP_R by start and finish times. To fulfil the normalisation step of the RK-EDA, the RK of each activity in the jth rank is set to $\frac{j-1}{n}$ where n is the number of activities in the project

5.6.2 Experimental Settings

This section presents the approach of evaluating the quality of solutions produced by the BPEDA. This includes the problem sets, parameter settings and the experimental approach.

5.6.2.1 Problem Sets

In addition to J10, J20 and J30 problem sets used in Section 5.5, we also included other problem sets in the PSPLIB; J12, J14, J16 and J18. Comparison with existing EDAs is done based on all instances of the PSPLIB.

We observed that BPGA-EDA scaled better to larger problems and so did RK-EDA. We therefore also introduced larger problems from the MMLIB which are MMLIB50 and MMLIB100 (Van Peteghem and Vanhoucke, 2014). As presented in Section 2.5.7, MMLIB50 and MMLIB100 respectively consist of 50 and 100 activities to be scheduled.

5.6.2.2 Parameter settings

RK-EDA which is used for generating activity solutions and the integer based EDA used for generating mode solutions require two parameters in common which are population size ps and truncation size b . The integer based EDA, in addition, requires a learning rate lr while RK-EDA requires a variance parameter σ. The parameters for BPEDA are presented in Table 5.15.

Values presented in Table 5.15 are derived based on preliminary tests. These tests reveal that different ps and σ values are required for the MMLIB and PSPLIB problem sets. This may be attributed to the difference in the formulation of both libraries.

We have set the limit on the number of evaluations to 5000 as this is the most frequently used stopping criteria (Van Peteghem and Vanhoucke, 2014).

Table 5.15: Parameter Settings

Parameter	**PSPLIB**	**MMLIB**
Population Size (ps)	$\frac{3000}{n}$	100
Truncation Size (b)	$0.1 \times ps$	$0.1 \times ps$
Variance (σ)	Minimum of 0.2 and $\frac{3}{n}$	0.05
Mode improvement rate	$0.0, 0.2$	$0.0, 0.2$
Learning rate	0.8	0.8
Number of evaluations	5000	5000
Number of Runs	10	10

5.6.2.3 Experimental Approach

The most common performance measure, using number of schedules as stopping criteria, is the APD presented in Eq. (2.15). In Eq. (2.15), $Best$ is the fitness of the best solution generated by an algorithm. The value of *optimal* is either the reported optimal value for J10 and J20 or CPBLB for J30, MMLIB50 and MMLIB100. This is common practice in previous research especially the review in (Van Peteghem and Vanhoucke, 2014). As noted in Chapter 3, CPBLB is used because the optimal values of certain problem instances are unknown.

Furthermore, local search methods have been reported to significantly improve the performance of meta-heuristics (Van Peteghem and Vanhoucke, 2014). The proposed approach also considers the mode improvement method which is one of the most efficient improvement methods shown to improve the performance of most leading algorithms. These include the SS, BPGA and BPGA-EDA. Since the proposed approach uses a similar approach as the BPGA and BPGA-EDA, we will be comparing directly with these algorithms with and without the use of the mode improvement local search method.

The mode improvement rate relates to the number of activities that are improved when scheduling a project. Based on preliminary results, 0.2 mode improvement rate gave the best results. Using 0.2 rather than 0.3 used in Chapter 3 implies that 10% fewer activities will be going through the improvement procedure. We also considered applying ESGS and SGS at every other generation ($e = 0.5$). This further reduces additional computation that may be attributed to the ESGS. In ESGS, each possible start time assignment of an activity counts as a fraction of a schedule evaluation, calculating the ERR each time can, however, be considered additional computation. Setting e to 0 or 1 respectively denotes that SGS or ESGS is used at every generation.

In literature, it is common practice to run algorithms only once across all problem sets of the PSPLIB/MMLIB. It was however shown in Chapter 3 that several runs are needed to capture the true performance of non-deterministic algorithms applied to these problem sets. In this chapter, results are averaged over ten runs.

5.7 Results and Discussion

In this section, BPEDA is compared with BPGA, BPGA-EDA, existing EDAs and other leading algorithms. The student t-test with 95% confidence interval is used to measure statistical significance.

5.7.1 Comparing BPEDA with BPGA-EDA and BPGA

To be able to compare based on several runs for results based on the standard SGS and ESGS, results for the BPGA are obtained from experiments in Chapter 3.

In Table 5.16, we present the APD averaged over ten runs (alongside the standard deviation) for the BPGA, BPGA-EDA and BPEDA. These results are based on the standard SGS.

Table 5.16: Results based on SGS - average APD (Standard deviation)

Problem sets	BPGA	BPGA-EDA	BPEDA
J10	0.61 (0.08)	**0.20 (0.04)**	**0.19 (0.04)**
J20	2.34 (0.05)	1.60 (0.07)	**1.05 (0.08)**
J30	17.89 (0.18)	15.06 (0.07)	**14.52 (0.07)**

In Table 5.16, the best results as well as results that are not significantly different from the best are presented in bold. BPEDA is significantly better than the BPGA on J10, J20 and J30 problem sets. The performance of BPEDA is also better than BPGA-EDA with no statistical difference on J10 but significantly better results on J20 and J30.

The performance of BPEDA in comparison with BPGA-EDA show that RK-EDA can produce better quality of activity solutions than GA. Since the Proposed EDA uses similar procedures as the BPGA and BPGA-EDA, Its relative performance also shows that EDA can outperform the GA on both components of the MRCPSP.

Since previous research has shown that the mode improvement method can significantly improve the performance of BPGA and BPGA-EDA, we also compare performance based on the use of this improvement method. Table 5.17 presents the APD (and Standard deviation) averaged across ten runs of the algorithms based on ESGS (i.e. e =1).

Table 5.17: Results based on ESGS - average APD (Standard deviation)

Problem sets	BPGA	BPGA-EDA	BPEDA
J10	0.05 (0.02)	**0.03 (0.02)**	0.06 (0.02)
J20	0.88 (0.04)	0.69 (0.04)	**0.64 (0.04)**
J30	14.41 (0.05)	13.87 (0.07)	**13.66 (0.07)**

Based on Table 5.17, there is no statistical difference between the performance of BPEDA and BPGA on J10, but BPEDA performs significantly better than the BPGA on J20 and J30. BPEDA is however significantly worse than the BPGA-EDA on J10 but significantly better on J20 and J30.

5.7.2 Comparing the proposed EDA with existing EDAs

Since existing EDAs use an improved SGS (MSSGS), we compare them with BPEDA based on ESGS. Table 5.18 presents the result of the EDAs in (Wang and Fang, 2012a) and (Soliman and Elgendi, 2014) as well as BPEDA.

Table 5.18: Results comparing BPEDA with other EDAs: average APD

Problem Sets	BPEDA	EDA^w	EDA^s
J10	**0.06**	0.12	0.09
J12	0.17	0.14	**0.12**
J14	**0.28**	0.43	0.36
J16	**0.40**	0.59	0.42
J18	**0.48**	0.90	0.85
J20	**0.64**	1.28	1.09
J30	**13.95**	15.55	-

In Table 5.18, EDA^w is used to represented the EDA of Wang and Fang (2012a) while EDA^s denotes the EDA of Soliman and Elgendi (2014)

Although the results of the EDA in (Soliman and Elgendi, 2014) are based on several runs, information about variance are however not presented. Similarly, there is no information on variance provided in (Wang and Fang, 2012a). We are therefore unable to test for statistical significance. Apart from J12, BPEDA shows better performance than the EDA in (Wang and Fang, 2012a) or (Soliman and Elgendi, 2014).

This is a competitive performance by BPEDA considering the fact that it uses fewer improvement methods and also a lighter weight model. The probabilistic model used by this EDA (of size $n + 1$) for activity solutions is much smaller than that of existing EDAs (of size $n \times n$)

5.7.3 Comparing the proposed EDA with leading algorithms

Having established the competitive performance of BPEDA with BPGA, BPGA-EDA and existing EDAs, this section focuses on comparing BPEDA with other leading algorithms.

As previously noted, results presented based on applications of meta-heuristics to the MRCPSP are often based on a single run. This includes the review in (Van Peteghem and Vanhoucke, 2014). However, to be able to compare with other algorithms on the MMLIB datasets as well as PSPLIB, most results are retrieved from (Van Peteghem and Vanhoucke, 2014). Some more recent algorithms not captured in the review such as algorithms presented in (Soliman and Elgendi, 2014), (Geiger, 2016) and (Vanhoucke and Coelho, 2016) are however retrieved from the authors' papers. Apart from results presented in (Soliman and Elgendi, 2014) which are averaged across many runs, the results for other algorithms are based on a single run as presented by the authors are used. Note that results of the BPGA in Table 5.17 is therefore different from Table 5.19, as the former is based on our implementation and based on multiple runs. Some more literature on MRCPSP exist, but we have not been able

to compare with them because they have presented results based on different criteria such as CPU time.

Table 5.19: Results comparing BPEDA with leading algorithms: average APD

Algorithms	Improved SGS	J10	J20	J30	MMLIB50	MMLIB100
MMHGA (Lova et al, 2009)	✓	0.04	0.89	14.58	28.59	31.01
DE (Damak et al, 2009)	×	0.74	1.62	15.43	32.46	36.87
BPGA (Van Peteghem and Vanhoucke, 2010)	✓	**0.01**	**0.57**	13.75	27.12	29.55
GA (Elloumi and Fortemps, 2010)	×	0.12	1.51	16.16	32.47	40.22
GA (Coelho and Vanhoucke, 2011)	✓	0.07	0.80	14.44	-	-
EDA (Wang and Fang, 2012a)	✓	0.09	1.28	15.55	31.95	38.55
EDA (Soliman and Elgendi, 2014)	✓	0.09	1.09	-	-	-
LS (Geiger, 2016)	✓	-	-	-	33.02	44.11
GA (Vanhoucke and Coelho, 2016)	✓	0.07	0.94	14.62	29.42	34.60
BPGA-EDA	✓	0.03	0.69	13.87	27.96	31.38
BPEDA ($e = 0.0$)	×	0.19	1.05	14.52	28.27	28.94
BPEDA ($e = 0.5$)	✓	0.09	0.71	**13.65**	**26.20**	**27.47**
BPEDA ($e = 1.0$)	✓	0.06	0.64	13.66	26.52	28.46
SS (Van Peteghem and Vanhoucke, 2011)	✓	0.00	0.32	13.66	25.45	26.51

In Table 5.19, results based on improved SGS such as ESGS, MSSGS, MM-FBI are appended a "✓" while those that are based on standard SGS are appended a "×". Also, missing results or problem set for which an algorithm has not been able to attain feasibility for its instances are represented by "-". Although we show SS on the table, we do not directly compare the proposed method with it. This is because we have shown that it contains many other evaluations of complete solutions in addition to the SGS, which inhibits fair comparison.

We present results for BPEDA based on the standard SGS ($e = 0.0$), standard SGS/ESGS and at every other generation ($e = 0.5$) and ESGS ($e = 1.0$) only. Of the three configurations, $e = 0.5$ presents the best results, especially for larger problems. The $e = 1.0$ configuration, however, presents the best results on smaller problems. We show that the performance of the proposed EDA approach when $e = 0.5$ is the most competitive on the largest problem, MMLIB100. This indicates that larger problem requires less of the improvement procedure. This may be because the mode improvement only encourages more activities to run in parallel (Van Peteghem and Vanhoucke, 2010) but does not guarantee an improvement in makespan. The mode improvement method helps to explore the search space better in smaller problems but do not scale well to larger problems. Moreover, as shown in (Van Peteghem and Vanhoucke, 2011), certain problem instances benefits more from some improvement techniques than others.

Furthermore, compared to the algorithms that do not use any schedule improvement procedure (GA in (Elloumi and Fortemps, 2010) and DE in (Damak et al, 2009)), the BPEDA without schedule improvement ($e = 0$) presents better results.

Summarily, BPGA presents the best results on J10 and J20. However, BPEDA when $e = 0.5$ presents the best performance on J30, MMLIB50 and MMLIB100. This is consistent with the behaviour of RK-EDA on common permutation problems scaling better to larger problems.

5.8 Conclusion

Since RCPSP shares common characteristics with the *Activity Scheduling* sub-problem of the MRCPSP, RK-EDA is adapted to solve the RCPSP. Although RK-EDA presented worse results compared to the leading approaches, it produced results better than some of the existing approaches with a much simpler approach.

This chapter also proposed a hybrid of the state of the art GA; BPGA and an EDA. The proposed BPGA-EDA introduces a probabilistic model into the BPGA for generating *mode* solutions. Experiments comparing the BPGA-EDA with the BPGA show that hybridisation with EDA produces significant performance improvements. BPGA-EDA is also shown to scale better to larger problem sets than the BPGA. We conclude that the EDA is better suited for the generation of modes than the GA.

In this chapter, BPEDA was also proposed. BPEDA uses RK-EDA to solve the *activity scheduling* sub-problem and the same approach as BPGA-EDA to solve the *mode assignment* sub-problem. BP-EDA presented even further improvement when compared to the BPGA-EDA.

BPEDA showed better performance than existing EDAs on most of the problem sets used in this chapter. It also uses a lighter weight model for generating activity solutions. Additional procedures for achieving mutual exclusivity in activity solutions which is common in existing EDAs is also not used by the BPEDA. This makes BPEDA more effective and efficient than existing EDAs.

BPEDA is also compared with state-of-the-art approaches of solving the MRCPSP and shown to be competitive with these approaches. Its performance is most competitive on the larger problem sets (MMLIB50 and MMLIB100).

In this study, however, we have not been able to test for statistical significance when comparing with most of the previous methods because results for only one run were presented.

Chapter 6

Estimation of Distribution Algorithm for Real-World Project Scheduling

6.1 Introduction

The previous three chapters are on the applications of Estimation of Distribution Algorithms (EDAs) to benchmark problems. Although many of these problems are motivated by real-world problems, they are often not perfect fits for industrial problems Chiong et al (2012). Comparison with other meta-heuristics based on benchmark problems only gives an idea of how a designed approach performs in the research community but not necessarily in the industry. It is therefore important to assess the performance of the proposed algorithms on an actual real-world problem.

In this chapter, we answer the research question "How can we transfer knowledge from solving a multi-component test problem to a similar real-world problem?". A project scheduling problem experienced in a construction company is used as a case study in this study. This problem has been formulated as an RCPSP as well as MRCPSP. RK-EDA and BPEDA presented in Chapter 5 are respectively applied to the RCPSP and MRCPSP formulations. The results produced are compared to that of Primavera software, the software currently used in the case-studied construction company.

The rest of the chapter is structured as follows. The formulation of the case study is presented in Section 6.2. Section 6.3 presents the industrial solution approach as well as the proposed solution approach. The experimental settings are presented in Section 6.4. Results are presented in 6.5 while conclusions are presented in Section 6.6

6.2 A Real-World Project Scheduling Problem Case Study

The case-studied project scheduling problem is from Saudi Aramco, a petroleum and natural gas company. Saudi Aramco is a state-owned oil company of Saudi Arabia with subsidiaries and affiliates in many countries of the world such as China, United States of American and the United Kingdom. Saudi Aramco is one of the world's largest oil producers, and its operations also include exploration, production, refining, chemicals, distribution and marketing (Demirbas et al, 2016).

The case studied project scheduling problem originated from the company's branch in Abu Dhabi which executes projects of various sizes. Some projects consist of tens of activities while others consist of thousands of activities. Furthermore, certain projects may take several months/years to execute; it is, therefore, essential to minimise the duration of projects. Unlike many industries that require results in real-time, this company can spare more time to get good quality results.

As noted in the RCPSP/MRCPSP formulation, a project consists of a series of activities with precedence relationships between them. In this case study, design documents are generated for one of the company's projects. However, the design of some documents must be approved before the design of other documents can proceed. Also, the design of each document requires the completion of four activities as shown in Figure 6.1.

Figure 6.1: Document Design Process

The *development* activities (i.e. *first stage development* and *second stage development*) are performed by *developers* while *review* activities (i.e. *first stage review* and *second stage review*) are performed by *reviewers*. The company outsources for *developers* and therefore assumes they are unlimited. However, *reviewers* are staff of the company. There are often a limited number of *reviewers* at a given time. Also, a *reviewer* is either given 20 working days to complete a *review* activity or 10 working days if two *reviewers* are assigned to the same *review* activity. This duration is based on company policy and will be given to *reviewers* irrespective of how quickly they are able to perform the activity.

Since the design of each document requires four stages, the problem size is $4d$ where d is the number of documents to be prepared. The considered problem consists of 45 documents and is presented in Tables A.13 - A.17 of the Appendix. Table 6.1 is a sample of the problem instance. As shown in this table, *development* activities are denoted by *DOCx-y* while *review* activities are denoted by *DOCxR-y*, x represents the document number while y is the development/review stage (1 or 2). *AA001* and *AA002* respectively represent the dummy start and finish activities.

Table 6.1: Project Scheduling Problem

Activity ID	Document Reference	Original Duration	Predecessors	Successors
AA001	START	0d		DOC1-1, ⋯, DOC38-1
DOCUMENT 01				
DOC1-1	DOCUMENT 1 FIRST STAGE	30d	AA001	DOC1R-1
DOC1-2	DOCUMENT 1 SECOND STAGE	10d	DOC1R-1	DOC1R-2
DOC1R-1	DOCUMENT 1 FIRST STAGE REVIEW	10d	DOC1-1	DOC1-2, DOC4-1, DOC7-1
DOC1R-2	DOCUMENT 1 SECOND STAGE REVIEW	10d	DOC1-2	DOC4-2, DOC7-2
	⋮			
DOCUMENT 45				
DOC45-1	DOCUMENT 45 FIRST STAGE	70d	DOC44R-1	DOC45-2
DOC45-2	DOCUMENT 45 SECOND STAGE	40d	DOC45-1, DOC44R-2	DOC45R-1
DOC45R-1	DOCUMENT 45 FIRST STAGE REVIEW	10d	DOC45-2	DOC45R-2
DOC45R-2	DOCUMENT 45 SECOND STAGE REVIEW	10d	DOC45R-1	AA002
AA002	COMPLETION	0d	DOC3R-2,⋯, DOC45R-2	

Although theoretical problems are often not perfect fits for real-world problems, they however often serve as a guide to solving them (Chiong et al, 2012). As previously expressed in Chapter 2, MRCPSP is a generalisation of the RCPSP. The difference is that MRCPSP requires the choice of a mode of execution for each activity while RCPSP has only one mode of execution. In this case study, the *development* activities have only one mode of execution. However, the *review* activities have more than one mode of execution. In the case studied company, there are only two *reviewers* and there are three modes of execution; the first *reviewer*, the second *reviewer* or both *reviewers* may execute a *review* activity. Each *review* activity can, therefore, be performed in modes 1, 2 or 3 while each *development* activity has only one mode of execution which is represented by mode 0. This problem can be formulated as an MRCPSP.

In the industrial solution approach however, an assumption that both *reviewers* combine to perform every *review* activity is made. Since this problem also does not consider non-renewable resources, this makes the problem simply an instance of RCPSP.

In this chapter, we consider the RCPSP and MRCPSP formulations of this problem and are respectively solved with RK-EDA and BPEDA. We compare both approaches as well as compare with results from Primavera.

6.3 Solution Approach

6.3.1 Industrial Solution Approach

In the case study company, project schedules are generated using the industrial software, **Primavera** [1]. This is a project management software which uses a deterministic approach to build project schedules. It uses a resource levelling approach where an attribute of the project is used to resolve resource conflicts. When there are insufficient resources to execute all activities required to be performed at a given time, this attribute serves as a way to prioritise activities.

To solve the considered problem instance, an assumption that each *review* activity is always performed by both *reviewers* was made, making the problem an instance

[1] https://www.oracle.com/uk/applications/primavera/index.html

of RCPSP. Furthermore, the resources are levelled in ascending as well as descending order of activity IDs. The activity ID serves as a priority definer determining which activity gets a resource when there are conflicting resource needs.

6.3.2 EDA Solution Approach

In this chapter, we apply RK-EDA proposed for RCPSP in the previous chapter to the RCPSP formulation of the case study without modifications. BPEDA for MRCPSP also proposed in Chapter 5 is applied to the MRCPSP formulation but with some changes.

Some procedures of BPEDA are not applicable to the case study and were therefore removed. One such method is the mode improvement because there is no non-renewable resource infeasibility considered in this problem. Also, each mode of execution requires the same resource to duration ratio meaning jobs can not necessarily be done faster by changing the mode of execution. Furthermore, the preprocessing procedure is also not executed because there are no redundant, inefficient or non-executable modes in the considered problem. In Chapter 5, the fitness of a feasible solution is set to its makespan while the makespan of a non-renewable resource infeasible solution is penalised. However, since there are no non-renewable resources in this case study, the makespan is directly used as fitness. Other aspects of Alg. 7 are retained for the MRCPSP formulation.

6.4 Experimental Settings

In this chapter, the parameters used are adapted from Chapter 5 as presented in Table 6.2. Given that the problem size is 180, the parameters for the largest problem set in the previous chapter is used. We use the same set of parameters for the RCPSP and MRCPSP formulations. We use a population size of 100 and 10% of it as the truncation size. The variance value and number of fitness evaluation are respectively set to 0.05 and 5000. Since the BPEDA requires an additional parameter which is learning rate, value 0.8 is applied when solving the MRCPSP formulation. Individual runs of each algorithm are repeated 20 times.

Table 6.2: Parameter Values for RK-EDA and BPEDA

Parameters	**RK-EDA**	**BPEDA**
Learning Rate (lr)	–	0.8
Population Size (p_s)	100	100
Truncation Size (t_s)	10	10
Variance (σ)	0.05	0.05
Maximum Fitness Evaluations ($MaxFEs$)	5000	5000
Number of Runs	20	20

6.5 Results and Analysis

6.5.1 Comparing the RCPSP formulation with the MRCPSP formulation

Table 6.3: Results Comparing RCPSP and MRCPSP Formulations

	RCPSP	MRCPSP
Best makespan	920	920
Worst makespan	920	930
Median makespan	920	920
Number of evaluations for best run	100	401
Number of evaluations for worst run	211	5000
Average (stdev) number of evaluations	124 (37)	1471 (1077)

Based on the results presented in Table 6.3, solving the RCPSP formulation of the problem required only a fraction of the number of fitness evaluations of the MRCPSP formulation. This is expected as the RCPSP formulation simplifies the problem reducing the number of possible solutions. Also, the best and median makespan are the same. However, the worst makespan of the RCPSP is 10 days better than the MRCPSP.

To test for significant difference, we use a non-parametric test in this chapter. This is because the D'Agostino-Pearson normality test shows that the data is not normally distributed. The Wilcoxon matched-pairs signed-rank test and a confidence interval of 95% is used. This test shows that there is no significant difference in the performance of RK-EDA and BPEDA on the case-studied problem. This can be attributed to the fact that each *reviewer* is allocated the same duration to complete a review task. Also, assigning a task to both *reviewers* reduces the duration by half. It is therefore more efficient to assign two *reviewers* to all activities which is consistent with the assumption made for the industrial approach.

Although there is no significant difference between the RCPSP and MRCPSP formulations, formulating the problem as an MRCPSP may however make the solution approach more flexible. We respectively show the schedules for the RCPSP and MRCPSP on the first three documents in Figures 6.2 and 6.3. The full project schedule for the RCPSP formulation is presented in Figures B.1 - B.3 while that of the MRCPSP is also presented in Figures B.1 - B.3 of the Appendix. From Figure 6.3, we see how the algorithm schedules different review activities based on different modes. This may be of benefit if a reviewer is not available for a given period. It is also easily adaptable to company policies which may increase/reduce the number of *reviewers* or vary the number of working days required by individual *reviewers*.

6.5.2 Comparing Primavera Solution with EDA

As shown in Table 6.4, when resources are levelled in ascending order of activity ID, a makespan of 1120 days was obtained. However, when executed in descending order of activity ID, a lower makespan of 955 days was achieved by the *Primavera* software.

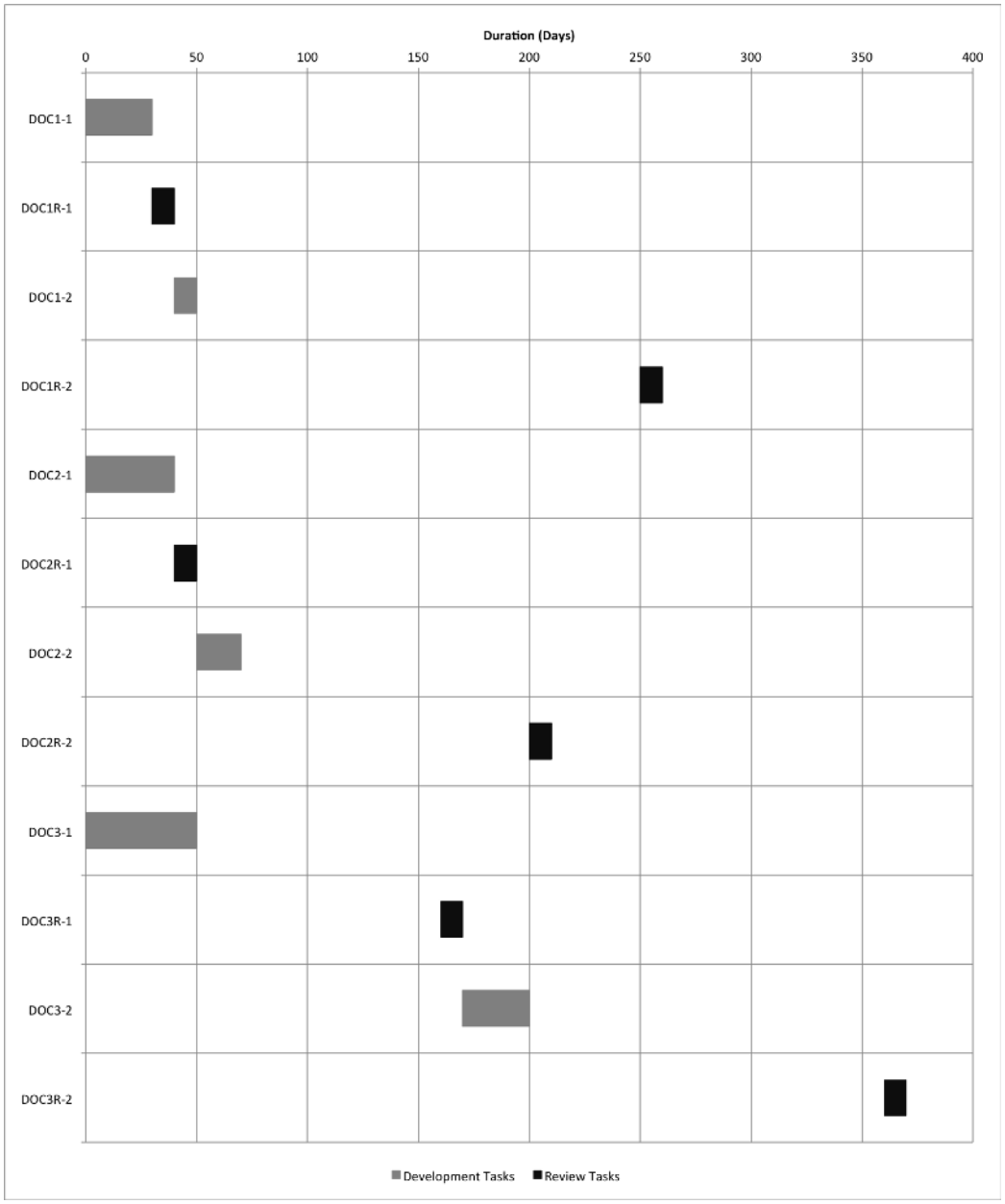

Figure 6.2: Schedule by RK-EDA: DOCs 1-3

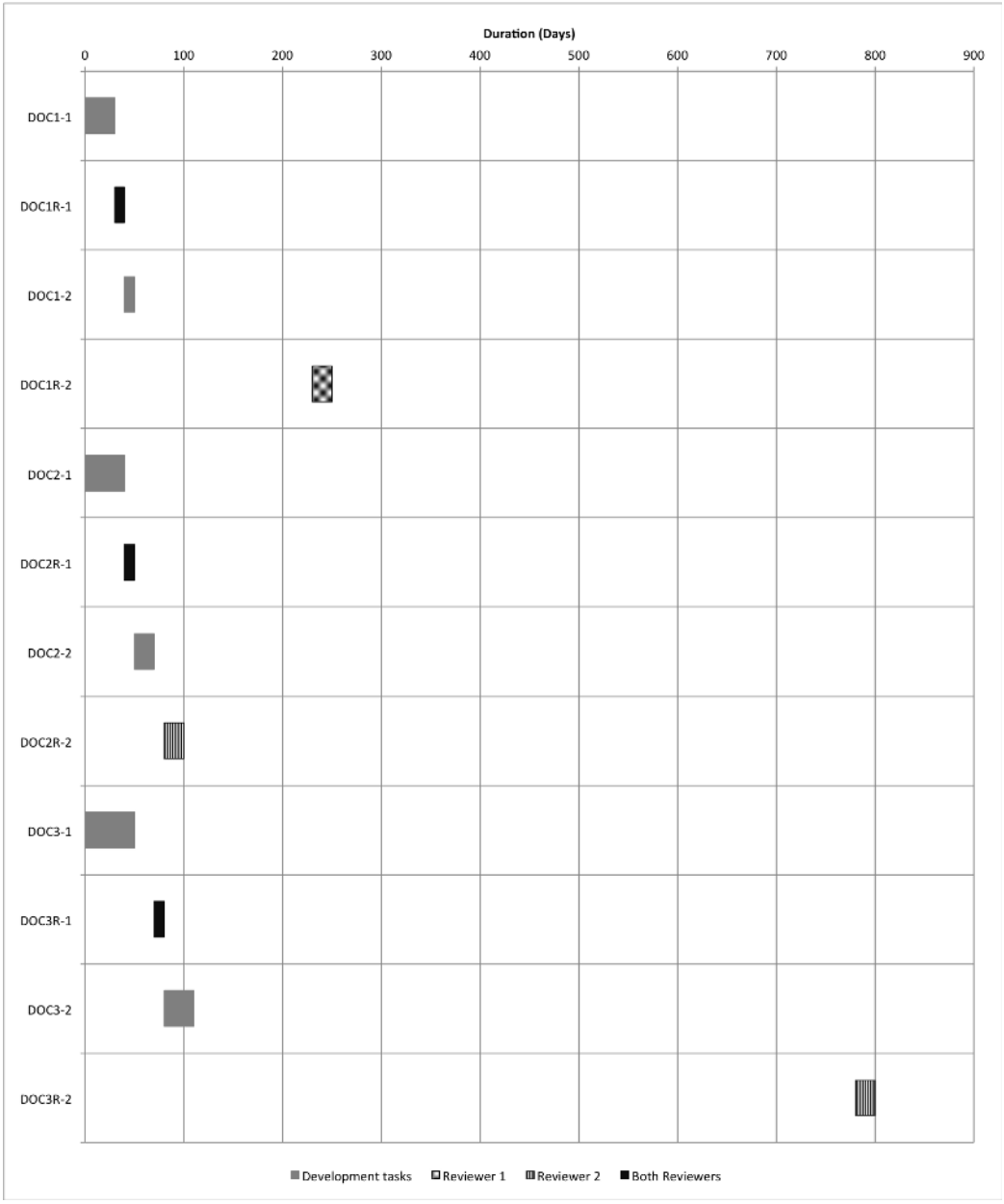

Figure 6.3: Schedule by BPEDA: DOCs 1-3

Table 6.4: Results: Makespan (days)

Primavera (Asc)	**Primavera** (Desc)	**EDA**
1120	955	920

Since the results from *Primavera* are deterministic, each run of the software will produce the same result. Using the Wilcoxon matched-pairs signed-rank test and a confidence interval of 95%, the results of EDA(BPEDA/RK-EDA) are significantly better than *Primavera*. The saving for the industry is to reduce the duration of engaging staff for that project by at least 2%.

6.6 Conclusions

This study benefits from a problem which has been pre-formulated such that it can be solved with the Primavera software. The EDA proposed for the RCPSP/MRCPSP was therefore directly applicable with only minor changes. These changes include excluding non-renewable resources which are not presented as part of this problem.

Although the case study problem is neither a perfect fit to the RCPSP nor the MRCPSP, understanding this theoretically formulated problem gave more insight to solving the real-world problem. In this chapter, EDA is shown to outperform the industrial software (Primavera) on the case studied problem because it has already been optimised for solving benchmark problems. The EDA also present more opportunities for flexibility in modes of executing activities of a project. This can be especially important in handling changes such as new availability of staff or temporary unavailability of staff.

We found that business rules play an important role in the formulation of a problem. The assumption that each *Reviewer* takes exactly the same duration, makes the RCPSP a more suitable formation for the case-study problem.

In this chapter, we compare based on makespan rather than ARPD which was used in previous chapter. This is because there are not many methods to compare with, there is no known optimal and the interest of the company is in the number of days that can be saved by using an EDA.

In summary, we have been able to achieve a proof-of-concept real-world application. The result of this research has however not been used in the industry. To achieve this, further effort would be needed to package the algorithm in a form that could be easily integrated into the business processes of the company.

Chapter 7

Conclusion and Further Work

7.1 Introduction

This thesis contributes to a number of research areas within the field of evolutionary computation in general and EDA in particular. Those contributions are presented in this chapter. Also, some limitations of this study are presented. Finally, this chapter presents a prospectus for possible future work.

7.2 Research Questions Revisited

This section presents answers to the research questions presented in Chapter 1.

- *In what way can the problem of redundancy in RK be addressed?*

  In this thesis, we identified the problem of redundancy in the RK representation where more than one genotype generates the same phenotype. This problem is solved by rescaling solutions to a common canonical value set. With this approach, each genotype generates a unique phenotype. This approach was used in the proposed RK-EDA which significantly outperforms other EDAs that use the RK representation. Results are presented in Chapter 3

- *In what way can diversity be controlled in EDAs designed to solve permutation problems?*

  This research has also addressed the problem of premature convergence that exists in EDAs. This is inevitable when the variance of iteratively re-sampled distributions becomes very small. In this study, the variance of the distribution is pre-defined and controlled using a cooling scheme. This enables the EDA to control diversity in the search leading to improved performance. The proposed cooling scheme used by RK-EDA was shown to preserve diversity better than the state-of-the-art EDA, GM-EDA. Detailed analysis of this is presented in Chapter 4

- *How can we reduce the need for local improvement procedures in multi-component scheduling problems?*

We have also shown that multi-component problems are often more difficult to solve. To handle the complexity of this class of problems, meta-heuristics have relied on many local improvement procedures. This has led to many complex algorithms consisting of many parts and requiring the tuning of many parameters. In this study, we investigated how the introduction of EDAs can reduce the need for local improvement procedures. Since EDAs encapsulate learning in an explicit probabilistic model, they can identify and preserve patterns better than other classical EAs like the GA. We show that EDA was able to produce more solutions that respect constraints than the classical GA. Also, algorithms are sometimes not sufficiently adapted for solving a problem. In Chapter 5, each component of the MRCPSP was independently examined leading to a solution approach that produced better quality solutions with fewer improvement procedures.

- *How can we transfer knowledge from solving a multi-component test problem to a similar real-world problem?*

  The EA community has identified the fact that real-world problems can be significantly different from benchmark problems. Methods designed for benchmarks are sometimes considered inapplicable for solving real-world problems. In this study, we show how methods proposed for solving RCPSP and MRCPSP benchmarks can be adapted for solving a real-world project scheduling problem. A difference between the benchmarks and the case-study presented in this thesis are business-specific rules. This has also been identified as a major factor in previous study. Apart from business rules, we see that the performance measureARPD used in previous chapters in not useful when solving the real-world problem.This is because there is no known optimal and the interest is in the number of days that can be saved. Results produced in Chapter 6 show that the proposed methods perform significantly better than one of the standard industrial software for project scheduling, Primavera.

7.3 Summary of Contributions and Analysis of Limitations

The following are the main contributions of this thesis.

This thesis proposed a novel RK based EDA (RK-EDA) which was adapted for solving common permutation and scheduling problems. These problems are PFSP, TSP, QAP, LOP and RCPSP. The performance of RK-EDA was less competitive on QAP and RCPSP. When solving the QAP, it is particularly important to preserve relative order. Also, the RCPSP requires that precedence is always respected. Since RK-EDA is a univariate EDA, it is not able to explicitly define relative order or precedence relationships. However, RK-EDA was particularly competitive on the PFSP, presenting new best-known solutions on the largest problem instances. RK-EDA was able to perform well on the PFSP because the algorithm has a mechanism

for controlling diversity in a population. Previous approaches to solving the PFSP had suffered from premature convergence.

This thesis also presents the BPGA-EDA which combines a GA with an EDA to solve the MRCPSP. The hybrid algorithm was shown to be better than the stand alone GA. As an improvement to the BPGA-EDA, BPEDA was proposed to solve the MRCPSP. BPEDA's performed significantly better than one of the leading methods of solving the MRCPSP, BPGA. It also performs better than BPGA-EDA. However, it is common practice to present results for only one run in previous studies. It was therefore impossible to determine the statistical significance of the difference in performance of BPEDA and results presented in literature.

In Chapter 2, we categorised real-world applications using four factors. These are the use of real-world data, expert involvement, industrial evaluation and real-world implementation. In Chapter 6, we were able to obtain realistic data. We also have an industrial contact who provided all the necessary background information and business rules relating to the case study. Furthermore, the results produced was presented to the industrial expert who validated them. Although there is potential for further work with the case-study company, we did not attain the stage of real-world implementation during the period of this research. To be usable by the company, further effort would be needed to package the algorithm in a form that could be easily integrated into their business processes.

The work presented resulted in four conference papers and a workshop paper in highly-ranked conferences in EC.

7.4 Directions for Further Research

7.4.1 Investigation of Cooling Scheme in RK-EDA

RK-EDA performs competitively with leading algorithms applied to permutation and scheduling problems. However, investigations on the cooling scheme presented in Chapter 4 shows that RK-EDA converges to similar solutions too quickly. More detailed analysis of cooling schemes in RK-EDA is recommended for further studies. Alternative cooling schemes such as logarithmic schemes may further improve the performance of the algorithm.

7.4.2 Multivariate BPEDA for MRCPSP

In this thesis, BPEDA was proposed for the MRCPSP. As shown in Chapter 5, BPEDA combines RK-EDA with an integer-based EDA to solve the MRCPSP. However, BPEDA does not model the interactions between the two sub-problems of the MRCPSP. A multivariate approach may further improve the performance of the algorithm.

7.4.3 Preserving Relative Order in RK-EDA

RK-EDA does not have an explicit mechanism for preserving relative order. The QAP amongst other permutation problems is sensitive to the preservation of relative order. RK-EDA's relatively poor performance on QAP compared to other common permutation problems can be attributed to this fact. A multivariate RK-EDA may perform better on the QAP.

7.4.4 Efficient EAs

Hybridising evolutionary algorithms with local search and improvement procedures has been a common trend in the community. In this thesis, we see that these methods perform well on problems of lower dimensions but do not scale well to problems of larger dimensions. This thesis shows that progress can be achieved by sufficiently adapting algorithms to a problem domain rather than improving algorithms by including more complex operations.

7.5 General Conclusions

The work presented in this thesis covers many areas in the application of EDAs to permutation and scheduling problems. RK-EDA presents competitive results and even best-known results on some of the permutation problem instances. This shows that EDAs can outperform many leading algorithms on permutation and scheduling problems if sufficiently adapted to the problem domain. The introduction of a rescaling approach and a cooling scheme moves random key EDAs from being the poorest for permutation problems to be one of the best EDAs. RK-EDA also outperforms other leading evolutionary algorithms on the PFSP.

Also, we show that EDA can outperform even software that is currently used in the industry. Applying RK-EDA to an actual real-world problem confirms the applicability of EDA in practice. Based on the factors listed in Section 2.4, we were able to use real-world data, ensure the problem is understood from the industrial perspective and results were also verified by an industrial expert. However, we did not achieve the final attribute listed which is real-world implementation. Although there is a prospect for real-world implementation, we did not get to this stage during the period of this research.

Furthermore, there has been a recent trend of using many local search and improvement procedures in evolutionary algorithms to achieve better results at the expense of computation and simplicity of procedures. We observe that these methods do well on smaller problems but often do not scale well to larger problems. In this thesis, results produced by the proposed algorithms emphasise the importance of focusing on simpler methods that scale well to larger problems. Proposing less complex methods that effectively and efficiently solve larger problems is a major step towards bridging the theory-to-practice gap in the community.

Appendix A

Tables

Table A.1: Average Ranks of Parameters for J10: ρ=1

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-34, lr-1.0	21.250	b-101, lr-0.7	23.200
b-68, lr-0.7	21.350	b-101, lr-0.9	23.200
b-68, lr-0.9	21.375	b-135, lr-1.0	23.200
b-34, lr-0.6	21.400	b-34, lr-1.0	23.200
b-101, lr-1.0	21.700	b-68, lr-1.0	23.200
b-101, lr-0.7	21.775	b-135, lr-0.7	23.825
b-68, lr-0.8	21.975	b-68, lr-0.5	23.825
b-34, lr-0.4	22.200	b-68, lr-0.9	23.825
b-101, lr-0.9	22.275	b-169, lr-1.0	24.450
b-34, lr-0.8	22.500	b-101, lr-0.6	24.575
b-34, lr-0.5	22.525	b-135, lr-0.6	24.575
b-68, lr-1.0	22.550	b-68, lr-0.6	24.575
b-135, lr-1.0	22.750	b-68, lr-0.8	24.575
b-34, lr-0.9	22.925	b-101, lr-0.5	24.700
b-101, lr-0.8	23.075	b-135, lr-0.9	24.700
b-101, lr-0.6	23.225	b-34, lr-0.4	24.700
b-68, lr-0.6	23.325	b-34, lr-0.6	24.700
b-68, lr-0.4	23.375	b-34, lr-0.7	24.700
b-34, lr-0.7	23.450	b-34, lr-0.8	24.700
b-68, lr-0.3	23.625	b-68, lr-0.7	24.700
b-68, lr-0.2	24.325	b-101, lr-0.2	25.025
b-34, lr-0.3	24.425	b-169, lr-0.6	25.100
b-135, lr-0.7	24.525	b-169, lr-0.8	25.100
b-68, lr-0.1	24.675	b-34, lr-0.5	25.250
b-169, lr-0.9	24.725	b-34, lr-0.9	25.425
b-101, lr-0.5	24.850	b-101, lr-0.4	25.450
b-135, lr-0.9	24.975	b-101, lr-0.8	25.450
b-135, lr-0.8	25.350	b-101, lr-1.0	25.450
b-101, lr-0.4	25.525	b-135, lr-0.8	25.450
b-68, lr-0.5	25.650	b-34, lr-0.2	25.450
b-135, lr-0.6	26.100	b-68, lr-0.3	25.775
b-169, lr-0.8	26.200	b-169, lr-0.3	25.800
b-34, lr-0.2	26.275	b-169, lr-0.9	25.825
b-169, lr-1.0	26.625	b-101, lr-0.3	25.975
b-34, lr-0.1	27.225	b-135, lr-0.4	25.975
b-101, lr-0.3	27.250	b-169, lr-0.5	25.975
b-169, lr-0.7	28.025	b-34, lr-0.1	26.275
b-135, lr-0.5	28.475	b-169, lr-0.7	26.275
b-101, lr-0.1	28.975	b-68, lr-0.2	26.325
b-169, lr-0.5	29.650	b-68, lr-0.4	26.525
b-135, lr-0.3	29.675	b-34, lr-0.3	26.625
b-135, lr-0.4	29.800	b-101, lr-0.1	26.675
b-135, lr-0.1	29.800	b-68, lr-0.1	26.925
b-169, lr-0.4	29.875	b-169, lr-0.4	27.350
b-169, lr-0.6	29.975	b-135, lr-0.3	27.500
b-135, lr-0.2	30.225	b-135, lr-0.1	27.700
b-101, lr-0.2	30.275	b-135, lr-0.2	27.700
b-169, lr-0.1	30.400	b-135, lr-0.5	27.775
b-169, lr-0.3	30.675	b-169, lr-0.2	28.075
b-169, lr-0.2	31.850	b-169, lr-0.1	31.675

Table A.2: Average Ranks of Parameters for J10: ρ=0.5

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-34, lr-1.0	19.675	b-101, lr-0.2	25.000
b-34, lr-0.8	20.050	b-101, lr-0.3	25.000
b-34, lr-0.7	20.150	b-101, lr-0.4	25.000
b-68, lr-1.0	20.225	b-101, lr-0.6	25.000
b-34, lr-0.4	20.350	b-101, lr-0.7	25.000
b-34, lr-0.5	20.525	b-101, lr-0.8	25.000
b-34, lr-0.9	20.525	b-101, lr-0.9	25.000
b-34, lr-0.6	20.675	b-101, lr-1.0	25.000
b-34, lr-0.3	21.050	b-135, lr-0.1	25.000
b-68, lr-0.8	21.150	b-135, lr-0.4	25.000
b-68, lr-0.7	21.175	b-135, lr-0.6	25.000
b-101, lr-1.0	21.600	b-135, lr-0.7	25.000
b-68, lr-0.9	22.250	b-135, lr-0.8	25.000
b-101, lr-0.7	22.450	b-135, lr-0.9	25.000
b-68, lr-0.6	22.775	b-135, lr-1.0	25.000
b-101, lr-0.9	22.925	b-169, lr-0.1	25.000
b-101, lr-0.8	23.100	b-169, lr-0.5	25.000
b-68, lr-0.5	23.550	b-169, lr-0.6	25.000
b-101, lr-0.6	24.150	b-169, lr-0.7	25.000
b-135, lr-1.0	24.150	b-169, lr-0.8	25.000
b-135, lr-0.8	24.150	b-169, lr-0.9	25.000
b-101, lr-0.5	24.725	b-169, lr-1.0	25.000
b-34, lr-0.2	25.075	b-34, lr-0.1	25.000
b-68, lr-0.4	25.175	b-34, lr-0.3	25.000
b-135, lr-0.7	25.325	b-34, lr-0.5	25.000
b-135, lr-0.9	25.575	b-34, lr-0.6	25.000
b-34, lr-0.1	25.625	b-34, lr-0.8	25.000
b-101, lr-0.4	26.175	b-34, lr-0.9	25.000
b-169, lr-1.0	26.200	b-34, lr-1.0	25.000
b-169, lr-0.9	26.625	b-68, lr-0.3	25.000
b-135, lr-0.5	26.800	b-68, lr-0.4	25.000
b-68, lr-0.3	27.375	b-68, lr-0.5	25.000
b-68, lr-0.2	27.475	b-68, lr-0.8	25.000
b-135, lr-0.6	27.700	b-68, lr-0.9	25.000
b-101, lr-0.3	27.950	b-68, lr-1.0	25.000
b-135, lr-0.3	28.000	b-101, lr-0.5	26.200
b-135, lr-0.4	28.100	b-34, lr-0.7	26.200
b-68, lr-0.1	28.100	b-68, lr-0.6	26.200
b-169, lr-0.8	28.275	b-101, lr-0.1	26.250
b-169, lr-0.6	28.300	b-135, lr-0.2	26.250
b-169, lr-0.5	29.125	b-135, lr-0.3	26.250
b-169, lr-0.7	30.150	b-135, lr-0.5	26.250
b-101, lr-0.2	30.300	b-68, lr-0.1	26.250
b-135, lr-0.2	30.550	b-68, lr-0.7	26.250
b-169, lr-0.1	30.675	b-34, lr-0.2	26.425
b-101, lr-0.1	30.800	b-169, lr-0.2	27.450
b-169, lr-0.3	30.800	b-169, lr-0.4	27.450
b-135, lr-0.1	31.625	b-68, lr-0.2	27.450
b-169, lr-0.4	32.425	b-169, lr-0.3	27.500
b-169, lr-0.2	33.325	b-34, lr-0.4	27.625

Table A.3: Average Ranks of Parameters for J20: ρ=1

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-11, lr-0.4	18.600	b-22, lr-0.7	19.725
b-22, lr-0.4	18.675	b-22, lr-1.0	20.400
b-22, lr-0.6	18.850	b-11, lr-0.8	20.725
b-22, lr-0.8	19.875	b-11, lr-0.7	21.150
b-22, lr-0.5	19.900	b-11, lr-0.6	21.200
b-33, lr-0.6	20.150	b-22, lr-0.9	21.450
b-22, lr-0.7	20.325	b-22, lr-0.4	21.725
b-11, lr-0.5	21.150	b-22, lr-0.8	21.775
b-44, lr-0.8	21.375	b-11, lr-1.0	22.150
b-33, lr-0.9	21.375	b-11, lr-0.9	22.175
b-33, lr-0.8	21.575	b-33, lr-1.0	22.200
b-22, lr-0.9	21.600	b-11, lr-0.5	22.250
b-11, lr-0.7	21.875	b-33, lr-0.7	22.375
b-11, lr-0.3	22.025	b-44, lr-0.6	22.450
b-11, lr-0.6	22.025	b-11, lr-0.4	22.625
b-33, lr-0.7	22.250	b-22, lr-0.5	22.675
b-33, lr-0.5	22.375	b-33, lr-0.9	22.775
b-44, lr-0.9	22.675	b-22, lr-0.6	23.125
b-22, lr-1.0	22.800	b-44, lr-0.9	23.250
b-44, lr-0.7	22.850	b-44, lr-1.0	23.600
b-11, lr-0.8	23.000	b-33, lr-0.8	23.750
b-44, lr-1.0	23.175	b-11, lr-0.3	23.800
b-33, lr-0.4	23.200	b-55, lr-1.0	23.850
b-55, lr-0.8	23.325	b-22, lr-0.3	23.900
b-55, lr-0.9	24.025	b-55, lr-0.8	24.050
b-33, lr-1.0	24.150	b-44, lr-0.8	24.550
b-11, lr-0.2	24.150	b-44, lr-0.7	24.725
b-22, lr-0.3	24.250	b-33, lr-0.5	25.525
b-44, lr-0.6	24.350	b-33, lr-0.6	25.550
b-11, lr-1.0	24.400	b-11, lr-0.2	25.575
b-11, lr-0.9	25.425	b-55, lr-0.9	25.925
b-55, lr-0.6	25.800	b-44, lr-0.5	26.250
b-44, lr-0.5	25.800	b-33, lr-0.4	26.950
b-55, lr-1.0	26.325	b-44, lr-0.4	27.450
b-22, lr-0.2	26.625	b-55, lr-0.5	27.525
b-55, lr-0.7	26.950	b-33, lr-0.3	27.875
b-33, lr-0.3	26.975	b-55, lr-0.6	28.250
b-44, lr-0.4	28.500	b-55, lr-0.7	28.500
b-55, lr-0.5	28.675	b-44, lr-0.3	28.900
b-55, lr-0.4	29.475	b-22, lr-0.2	28.975
b-11, lr-0.1	30.425	b-33, lr-0.2	29.200
b-33, lr-0.2	31.375	b-22, lr-0.1	30.075
b-44, lr-0.3	32.300	b-55, lr-0.4	30.150
b-44, lr-0.2	33.125	b-11, lr-0.1	30.150
b-55, lr-0.3	33.200	b-55, lr-0.3	30.225
b-22, lr-0.1	34.150	b-44, lr-0.2	31.075
b-33, lr-0.1	34.925	b-33, lr-0.1	32.100
b-55, lr-0.2	36.850	b-44, lr-0.1	32.600
b-44, lr-0.1	37.725	b-55, lr-0.2	33.050
b-55, lr-0.1	40.025	b-55, lr-0.1	34.700

Table A.4: Average Ranks of Parameters for J20: ρ=0.5

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-22, lr-0.8	18.575	b-11, lr-0.6	20.975
b-22, lr-0.4	18.650	b-33, lr-0.7	21.450
b-33, lr-0.7	19.050	b-22, lr-0.4	21.675
b-33, lr-0.6	19.350	b-22, lr-0.8	21.725
b-22, lr-0.5	19.375	b-11, lr-0.8	21.850
b-11, lr-0.4	19.450	b-11, lr-0.3	21.925
b-22, lr-0.7	19.850	b-22, lr-0.7	22.350
b-33, lr-0.8	19.975	b-33, lr-1.0	22.425
b-11, lr-0.6	21.050	b-11, lr-0.4	22.700
b-22, lr-0.6	21.400	b-11, lr-1.0	23.225
b-11, lr-0.5	21.425	b-22, lr-0.5	23.350
b-33, lr-0.5	21.425	b-11, lr-0.7	23.375
b-44, lr-0.8	21.450	b-44, lr-0.9	23.475
b-22, lr-0.9	21.500	b-33, lr-0.6	23.625
b-11, lr-0.8	22.250	b-44, lr-0.6	23.675
b-11, lr-0.3	22.350	b-55, lr-1.0	23.750
b-11, lr-0.7	22.375	b-22, lr-0.9	24.100
b-11, lr-1.0	22.425	b-11, lr-0.5	24.125
b-33, lr-0.4	22.475	b-33, lr-0.8	24.250
b-44, lr-0.9	23.200	b-22, lr-1.0	24.300
b-44, lr-0.6	23.500	b-11, lr-0.9	24.375
b-55, lr-0.9	23.625	b-55, lr-0.7	24.450
b-22, lr-0.3	23.875	b-44, lr-0.8	24.675
b-33, lr-1.0	24.125	b-11, lr-0.2	24.675
b-33, lr-0.9	24.175	b-44, lr-0.7	24.825
b-44, lr-0.7	24.275	b-33, lr-0.5	24.850
b-22, lr-1.0	24.600	b-44, lr-1.0	24.850
b-44, lr-1.0	24.800	b-22, lr-0.6	24.925
b-55, lr-0.8	24.975	b-33, lr-0.9	25.175
b-33, lr-0.3	25.325	b-55, lr-0.8	25.275
b-55, lr-0.7	25.575	b-33, lr-0.4	25.325
b-44, lr-0.5	25.925	b-55, lr-0.9	25.425
b-11, lr-0.2	26.200	b-44, lr-0.4	25.575
b-55, lr-0.6	26.300	b-33, lr-0.3	26.175
b-11, lr-0.9	26.825	b-22, lr-0.3	26.525
b-55, lr-1.0	27.000	b-44, lr-0.5	27.000
b-44, lr-0.4	27.750	b-55, lr-0.6	27.450
b-22, lr-0.2	27.925	b-55, lr-0.4	27.700
b-33, lr-0.2	29.150	b-55, lr-0.3	27.925
b-55, lr-0.5	29.200	b-44, lr-0.3	28.150
b-44, lr-0.3	30.025	b-22, lr-0.2	28.300
b-55, lr-0.4	30.600	b-33, lr-0.2	28.450
b-22, lr-0.1	31.575	b-44, lr-0.2	28.725
b-55, lr-0.3	32.375	b-55, lr-0.5	29.075
b-11, lr-0.1	32.750	b-11, lr-0.1	29.200
b-44, lr-0.2	33.025	b-55, lr-0.2	29.575
b-55, lr-0.2	34.875	b-55, lr-0.1	30.700
b-44, lr-0.1	37.675	b-22, lr-0.1	31.025
b-55, lr-0.1	38.975	b-33, lr-0.1	32.950
b-33, lr-0.1	40.400	b-44, lr-0.1	33.350

Table A.5: Average Ranks of Parameters for J30: ρ=1

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-22, lr-0.7	16.775	b-7, lr-0.5	18.025
b-22, lr-0.6	16.800	b-15, lr-0.9	18.325
b-15, lr-0.6	17.575	b-7, lr-0.6	18.875
b-22, lr-0.9	17.700	b-15, lr-1.0	18.925
b-30, lr-0.7	17.750	b-7, lr-0.7	18.975
b-15, lr-0.7	17.825	b-15, lr-0.8	19.050
b-15, lr-0.9	19.275	b-7, lr-1.0	19.150
b-15, lr-0.5	19.350	b-15, lr-0.7	19.225
b-30, lr-0.9	19.575	b-7, lr-0.8	19.600
b-30, lr-0.8	19.950	b-22, lr-0.9	20.150
b-37, lr-0.9	19.950	b-7, lr-0.9	20.300
b-37, lr-0.8	20.475	b-7, lr-0.4	20.600
b-15, lr-0.4	20.550	b-22, lr-0.8	20.825
b-22, lr-0.8	20.700	b-15, lr-0.6	20.875
b-15, lr-0.8	20.700	b-22, lr-0.7	21.175
b-7, lr-0.6	20.950	b-22, lr-1.0	22.025
b-30, lr-0.6	21.150	b-15, lr-0.5	22.275
b-15, lr-1.0	21.200	b-30, lr-0.9	22.275
b-30, lr-1.0	21.475	b-22, lr-0.6	22.450
b-30, lr-0.5	21.625	b-30, lr-1.0	22.675
b-7, lr-0.5	21.800	b-15, lr-0.4	22.700
b-22, lr-0.5	21.800	b-30, lr-0.8	22.975
b-37, lr-1.0	22.375	b-7, lr-0.3	23.225
b-7, lr-0.4	22.550	b-30, lr-0.7	24.125
b-37, lr-0.7	22.875	b-37, lr-0.8	24.300
b-22, lr-0.4	23.125	b-22, lr-0.5	24.725
b-22, lr-1.0	23.200	b-37, lr-1.0	24.950
b-37, lr-0.6	23.800	b-37, lr-0.9	25.075
b-15, lr-0.3	24.275	b-37, lr-0.7	25.825
b-7, lr-0.3	25.425	b-15, lr-0.3	25.950
b-37, lr-0.5	25.950	b-30, lr-0.6	26.075
b-22, lr-0.3	26.300	b-22, lr-0.4	27.200
b-30, lr-0.4	26.700	b-30, lr-0.5	27.325
b-7, lr-0.7	28.025	b-37, lr-0.6	27.425
b-7, lr-0.8	28.250	b-7, lr-0.2	28.150
b-7, lr-0.2	28.700	b-30, lr-0.4	28.175
b-7, lr-0.9	29.875	b-37, lr-0.5	28.650
b-15, lr-0.2	30.175	b-22, lr-0.3	29.150
b-37, lr-0.4	31.300	b-37, lr-0.4	29.675
b-30, lr-0.3	31.500	b-30, lr-0.3	30.025
b-37, lr-0.3	32.975	b-15, lr-0.2	30.375
b-7, lr-1.0	33.000	b-22, lr-0.2	32.000
b-22, lr-0.2	33.725	b-37, lr-0.3	32.150
b-7, lr-0.1	33.825	b-30, lr-0.2	33.275
b-15, lr-0.1	36.125	b-37, lr-0.2	34.525
b-30, lr-0.2	36.725	b-7, lr-0.1	34.975
b-37, lr-0.2	37.525	b-22, lr-0.1	35.250
b-22, lr-0.1	38.250	b-15, lr-0.1	36.175
b-30, lr-0.1	41.500	b-30, lr-0.1	37.275
b-37, lr-0.1	42.000	b-37, lr-0.1	37.525

Table A.6: Average Ranks of Parameters for J30: ρ=0.5

Parameter (SGS)	Ranking	Parameter (ESGS)	Ranking
b-22, lr-0.5	16.275	b-7, lr-0.7	18.650
b-30, lr-1.0	17.625	b-7, lr-0.9	18.725
b-22, lr-0.8	17.750	b-7, lr-1.0	18.900
b-15, lr-0.6	18.425	b-15, lr-0.9	19.300
b-30, lr-0.6	18.775	b-22, lr-0.9	19.450
b-22, lr-0.7	19.975	b-7, lr-0.8	19.725
b-15, lr-0.5	20.125	b-7, lr-0.5	20.125
b-15, lr-0.4	20.225	b-7, lr-0.4	20.750
b-30, lr-0.5	20.375	b-7, lr-0.6	20.875
b-22, lr-0.9	20.475	b-15, lr-0.8	21.175
b-37, lr-0.9	20.725	b-22, lr-1.0	21.300
b-37, lr-0.7	20.750	b-15, lr-1.0	21.600
b-22, lr-1.0	20.750	b-22, lr-0.7	22.025
b-15, lr-0.7	20.850	b-15, lr-0.6	22.175
b-37, lr-0.8	21.000	b-15, lr-0.7	22.300
b-22, lr-0.6	21.875	b-30, lr-1.0	22.400
b-30, lr-0.9	22.175	b-30, lr-0.9	23.300
b-30, lr-0.7	22.350	b-22, lr-0.8	23.350
b-30, lr-0.8	22.500	b-30, lr-0.7	23.675
b-15, lr-0.9	22.550	b-30, lr-0.8	24.175
b-7, lr-0.6	22.600	b-15, lr-0.5	24.275
b-7, lr-0.4	23.275	b-15, lr-0.3	24.675
b-22, lr-0.4	23.400	b-7, lr-0.3	24.825
b-7, lr-0.5	23.475	b-37, lr-0.9	24.900
b-30, lr-0.4	23.775	b-30, lr-0.6	24.925
b-15, lr-0.8	24.050	b-22, lr-0.6	25.000
b-37, lr-0.5	24.350	b-22, lr-0.5	25.175
b-15, lr-0.3	24.375	b-15, lr-0.4	25.275
b-37, lr-0.6	24.675	b-37, lr-0.8	25.575
b-37, lr-1.0	24.975	b-37, lr-0.6	26.100
b-22, lr-0.3	24.975	b-22, lr-0.3	26.450
b-7, lr-0.7	25.100	b-22, lr-0.4	26.675
b-7, lr-0.8	25.675	b-37, lr-1.0	26.800
b-30, lr-0.3	26.500	b-7, lr-0.2	27.175
b-15, lr-0.2	27.125	b-30, lr-0.5	27.375
b-7, lr-0.3	27.650	b-37, lr-0.7	27.600
b-15, lr-1.0	27.775	b-37, lr-0.5	28.100
b-37, lr-0.4	27.775	b-30, lr-0.4	28.775
b-7, lr-0.2	28.725	b-37, lr-0.4	29.000
b-22, lr-0.2	30.025	b-15, lr-0.2	29.775
b-37, lr-0.3	31.100	b-37, lr-0.3	29.850
b-7, lr-0.9	31.350	b-30, lr-0.3	30.200
b-7, lr-1.0	32.250	b-30, lr-0.2	31.475
b-30, lr-0.2	33.650	b-22, lr-0.2	31.725
b-37, lr-0.2	34.100	b-37, lr-0.2	32.350
b-15, lr-0.1	34.900	b-7, lr-0.1	32.525
b-7, lr-0.1	37.600	b-15, lr-0.1	32.775
b-22, lr-0.1	37.925	b-37, lr-0.1	33.475
b-30, lr-0.1	41.925	b-30, lr-0.1	33.525
b-37, lr-0.1	44.375	b-22, lr-0.1	34.675

Table A.7: Selected Problem Instances

J10	J20	J30
j1020_6.mm	j2027_7.mm	j3027_7.mm
j1032_4.mm	j2030_5.mm	j3030_5.mm
j1044_3.mm	j2057_3.mm	j3057_3.mm
j1052_2.mm	j2060_10.mm	j3060_10.mm
j1060_9.mm	j2062_9.mm	j3062_9.mm
j1043_2.mm	j2018_10.mm	j3020_10.mm
j1019_10.mm	j2023_6.mm	j3024_5.mm
j103_6.mm	j2021_9.mm	j3024_8.mm
j1055_6.mm	j2055_7.mm	j3054_8.mm
j1018_4.mm	j2051_10.mm	j3049_10.mm
j1039_3.mm	j2055_9.mm	j3055_3.mm
j1050_3.mm	j209_5.mm	j309_10.mm
j1046_3.mm	j2015_4.mm	j3016_2.mm
j1058_10.mm	j2042_8.mm	j3014_9.mm
j1050_10.mm	j2015_10.mm	j3045_2.mm
j1030_9.mm	j2048_6.mm	j3042_8.mm
j105_8.mm	j2011_4.mm	j3011_2.mm
j1061_5.mm	j2035_3.mm	j3033_2.mm
j1013_5.mm	j2033_6.mm	j3040_8.mm
j1053_1.mm	j203_2.mm	j308_6.mm

Table A.8: Average Performance of RK-EDA on Benchmark Problems

Problems	**Instances**	**Minimum**	**Maximum**	**Mean**	**Stdev**
TSP	bays29	2020.0	2091.0	2041.5	21.3
	berlin52	8207.0	8742.0	8404.6	164.0
	dantzig42	729.0	824.0	771.2	35.6
	fri26	937.0	968.0	949.5	11.9
$PFSP_s$	tai20-5-0	1278.0	1279.0	1278.1	0.3
	tai20-5-1	1359.0	1360.0	1359.5	0.5
	tai20-10-0	1586.0	1618.0	1602.9	11.1
	tai20-10-1	1680.0	1691.0	1685.2	3.2
$PFSP_l$	tai50-10-0	3046.0	3119.0	3090.7	24.2
	tai50-10-1	2923.0	2964.0	2937.6	14.9
	tai100-20-0	6344.0	6424.0	6386.4	21.0
	tai100-10-1	6291.0	6381.0	6338.6	27.2
QAP	tai15a	393496.0	412072.0	404616.6	5350.2
	tai15b	51968294.0	52238818.0	52088443.6	72876.7
	tai40a	3353650.0	3418792.0	3391139.0	20951.9
	tai40b	642257062.0	659424886.0	652079961.9	4690584.3
LOP	t65b11	355180.0	356311.0	356028.2	295.6
	be75np	716221.0	716930.0	716644.3	249.8
	be75oi	110928.0	111156.0	111012.3	77.8

Table A.9: Comparing Average ARPD of RK-EDA and other Algorithms.

Problem Size	AGA	VNS4	GM-EDA	VNS	HGM-EDA	ILS	IGA	DEP	EHBSA	NHBSA	RK-EDA
20X5	0.00	0.00	0.35	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.07
20X10	0.00	0.00	0.46	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.11
20X20	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.18
50X5	0.10	0.93	0.98	0.21	0.20	0.15	0.20	0.09	0.13	0.96	1.14
50X10	0.22	0.97	2.01	0.43	0.42	0.36	0.39	0.25	0.75	1.72	1.79
50X20	0.21	0.71	1.97	0.43	0.41	0.36	0.43	0.23	1.25	1.80	1.64
100X5	0.25	1.49	0.94	0.37	0.29	0.52	0.53	0.08	0.86	1.70	1.05
100X10	0.46	1.66	1.86	0.73	0.49	0.61	0.67	0.20	2.97	2.76	1.54
100X20	0.58	1.47	2.05	0.78	0.51	0.54	0.60	0.31	4.19	3.11	1.62
200X10	0.74	1.54	1.35	1.04	0.28	0.59	0.53	0.17	5.84	4.57	0.85
200X20	1.13	1.70	1.72	1.43	0.52	0.67	0.73	0.19	6.90	5.54	0.93
500X20	0.88	1.08	9.40	2.71	2.65	0.52	0.54	1.46	9.24	7.85	0.18

Table A.10: ARPD: RK-EDA for PFSP (20 X 5 - 50 X 5)

jobs	machines	instance	min	max	avg	std
20	5	1	14033	14034	14033	0
20	5	2	15151	15151	15151	0
20	5	3	13301	13313	13302	3
20	5	4	15447	15447	15447	0
20	5	5	13529	13529	13529	0
20	5	6	13123	13123	13123	0
20	5	7	13548	13583	13559	8
20	5	8	13948	13948	13948	0
20	5	9	14295	14315	14297	6
20	5	10	12943	12943	12943	0
20	10	1	20911	20958	20947	18
20	10	2	22440	22440	22440	0
20	10	3	19833	19850	19836	5
20	10	4	18710	18755	18718	14
20	10	5	18641	18644	18642	1
20	10	6	19245	19249	19245	1
20	10	7	18363	18396	18367	9
20	10	8	20241	20315	20260	21
20	10	9	20330	20330	20330	0
20	10	10	21320	21323	21320	1
20	20	1	33623	33623	33623	0
20	20	2	31587	31587	31587	0
20	20	3	33920	33920	33920	0
20	20	4	31661	31698	31663	8
20	20	5	34557	34586	34565	11
20	20	6	32564	32564	32564	0
20	20	7	32922	33039	32964	43
20	20	8	32412	32481	32457	20
20	20	9	33600	34175	33773	263
20	20	10	32262	32262	32262	0
50	5	1	65088	65546	65369	112
50	5	2	68521	69030	68748	133
50	5	3	63760	64129	63904	93
50	5	4	68669	69281	68989	168
50	5	5	69718	70302	70026	160
50	5	6	67306	67619	67452	82
50	5	7	66767	67211	66892	99
50	5	8	64881	65329	64988	101
50	5	9	63377	63745	63564	108
50	5	10	69343	69918	69563	131

Table A.11: ARPD: RK-EDA for PFSP (50 X 10 - 100 X 10)

jobs	machines	instance	min	max	avg	std
50	10	1	88191	89348	88781	310
50	10	2	83763	84578	84104	223
50	10	3	80698	81477	81105	228
50	10	4	87207	88215	87613	199
50	10	5	87340	88118	87759	215
50	10	6	87089	88118	87596	271
50	10	7	89777	90658	90223	248
50	10	8	87468	88460	87935	255
50	10	9	86231	87292	86775	305
50	10	10	89124	90093	89662	264
50	20	1	127185	128432	127801	318
50	20	2	119816	120847	120416	289
50	20	3	117818	119047	118416	339
50	20	4	121661	122910	122405	327
50	20	5	119323	120496	119825	345
50	20	6	121639	122496	122041	244
50	20	7	124441	125699	124874	333
50	20	8	123490	124785	124205	366
50	20	9	122625	123859	123238	341
50	20	10	125285	126534	125755	299
100	5	1	255576	256754	256060	307
100	5	2	244263	245120	244726	201
100	5	3	239639	240358	240029	178
100	5	4	229348	230246	229678	199
100	5	5	242031	242960	242484	220
100	5	6	234423	235453	234866	273
100	5	7	242077	243175	242831	253
100	5	8	233166	233959	233481	203
100	5	9	250156	251368	250801	275
100	5	10	245004	245800	245451	200
100	10	1	301544	303562	302513	462
100	10	2	277047	280182	278149	807
100	10	3	290742	293531	292153	731
100	10	4	304730	306617	305526	461
100	10	5	287353	289801	288916	544
100	10	6	272432	274526	273662	481
100	10	7	282770	284722	283611	439
100	10	8	293554	295899	295139	550
100	10	9	305733	308222	306747	658
100	10	10	294564	296656	295650	487

Table A.12: ARPD: RK-EDA for PFSP (100 X 20 - 500 X 20)

jobs	machines	instance	min	max	avg	std
100	20	1	370429	372495	371407	521
100	20	2	376789	379691	378512	737
100	20	3	374108	376894	376006	805
100	20	4	377273	379848	378724	675
100	20	5	373269	375668	374716	597
100	20	6	375902	379468	378013	948
100	20	7	378105	380267	379028	539
100	20	8	388879	392169	390445	753
100	20	9	379394	381511	380564	509
100	20	10	384306	386302	385289	582
200	10	1	1050445	1055241	1052984	1042
200	10	2	1042005	1045015	1043077	982
200	10	3	1050267	1054507	1052313	1102
200	10	4	1035560	1039071	1037174	696
200	10	5	1040108	1042448	1041536	612
200	10	6	1011959	1014775	1013372	750
200	10	7	1056734	1060129	1058500	907
200	10	8	1049162	1054427	1051739	1300
200	10	9	1026541	1031220	1029249	1202
200	10	10	1033504	1038601	1035575	1248
200	20	1	1229135	1236326	1232389	1652
200	20	2	1244509	1252292	1247999	2037
200	20	3	1267859	1273800	1269641	1749
200	20	4	1240742	1247241	1243256	1650
200	20	5	1226326	1231966	1228899	1538
200	20	6	1226790	1233815	1230572	1887
200	20	7	1242066	1249482	1245372	1798
200	20	8	1243468	1251454	1247495	1773
200	20	9	1232200	1236999	1234396	1296
200	20	10	1248854	1254938	1251788	2162
500	20	1	6638306	6662109	6649445	4933
500	20	2	6764798	6804378	6780722	9366
500	20	3	6692427	6710148	6703236	4533
500	20	4	6725985	6747612	6735550	4463
500	20	5	6686734	6712450	6696428	6407
500	20	6	6687549	6709387	6701524	6167
500	20	7	6635167	6660516	6649215	6607
500	20	8	6713812	6744287	6728245	6850
500	20	9	6654590	6672419	6661983	4147
500	20	10	6695956	6725370	6707349	5832

Table A.13: Real-World Project Scheduling Problem: Start - DOC10

Activity ID	Document Reference	Duration	Predecessors	Successors
AA001	START	0d		DOC1-1, DOC2-1, DOC3-1, DOC6-1, DOC8-1, DOC12-1, DOC14-1, DOC18-1, DOC21-1, DOC26-1, DOC28-1, DOC34-1, DOC38-1
AA002	COMPLETION	0d	DOC3R-2, DOC12R-2, DOC15R-2, DOC17R-2, DOC19R-2, DOC20R-2, DOC6R-2, DOC14R-2, DOC25R-2, DOC28R-2, DOC33R-2, DOC35R-2, DOC37R-2, DOC39R-2, DOC41R-2, DOC42R-2, DOC43R-2, DOC45R-2	
DOCUMENT 01				
DOC1-1	DOCUMENT 1 FIRST STAGE	30d	AA001	DOC1R-1
DOC1-2	DOCUMENT 1 SECOND STAGE	10d	DOC1R-1	DOC1R-2
DOC1R-1	DOCUMENT 1 FIRST STAGE REVIEW	10d	DOC1-1	DOC1-2, DOC4-1, DOC7-1
DOC1R-2	DOCUMENT 1 SECOND STAGE REVIEW	10d	DOC1-2	DOC4-2, DOC7-2
DOCUMENT 02				
DOC2-1	DOCUMENT 2 FIRST STAGE	40d	AA001	DOC2R-1
DOC2-2	DOCUMENT 2 SECOND STAGE	20d	DOC2R-1	DOC2R-2
DOC2R-1	DOCUMENT 2 FIRST STAGE REVIEW	10d	DOC2-1	DOC2-2, DOC5-1, DOC29-1
DOC2R-2	DOCUMENT 2 SECOND STAGE REVIEW	10d	DOC2-2	DOC5-2, DOC29-2
DOCUMENT 03				
DOC3-1	DOCUMENT 3 FIRST STAGE	50d	AA001	DOC3R-1
DOC3-2	DOCUMENT 3 SECOND STAGE	30d	DOC3R-1	DOC3R-2
DOC3R-1	DOCUMENT 3 FIRST STAGE REVIEW	10d	DOC3-1	DOC3-2
DOC3R-2	DOCUMENT 3 SECOND STAGE REVIEW	10d	DOC3-2	AA002
DOCUMENT 04				
DOC4-1	DOCUMENT 4 FIRST STAGE	20d	DOC1R-1	DOC4R-1
DOC4-2	DOCUMENT 4 SECOND STAGE	10d	DOC4R-1, DOC1R-2	DOC4R-2
DOC4R-1	DOCUMENT 4 FIRST STAGE REVIEW	10d	DOC4-1	DOC4-2, DOC9-1
DOC4R-2	DOCUMENT 4 SECOND STAGE REVIEW	10d	DOC4-2	DOC9-2
DOCUMENT 05				
DOC5-1	DOCUMENT 5 FIRST STAGE	60d	DOC2R-1	DOC5R-1
DOC5-2	DOCUMENT 5 SECOND STAGE	40d	DOC5R-1, DOC2R-2	DOC5R-2
DOC5R-1	DOCUMENT 5 FIRST STAGE REVIEW	10d	DOC5-1	DOC5-2, DOC10-1, DOC24-1
DOC5R-2	DOCUMENT 5 SECOND STAGE REVIEW	10d	DOC5-2	DOC10-2, DOC24-2
DOCUMENT 06				
DOC6-1	DOCUMENT 6 FIRST STAGE	40d	AA001	DOC6R-1
DOC6-2	DOCUMENT 6 SECOND STAGE	30d	DOC6R-1	DOC6R-2
DOC6R-1	DOCUMENT 6 FIRST STAGE REVIEW	10d	DOC6-1	DOC6-2
DOC6R-2	DOCUMENT 6 SECOND STAGE REVIEW	10d	DOC6-2	AA002
DOCUMENT 07				
DOC7-1	DOCUMENT 7 FIRST STAGE	50d	DOC1R-1	DOC7R-1
DOC7-2	DOCUMENT 7 SECOND STAGE	40d	DOC7R-1, DOC1R-2	DOC7R-2
DOC7R-1	DOCUMENT 7 FIRST STAGE REVIEW	10d	DOC7-1	DOC7-2, DOC11-1
DOC7R-2	DOCUMENT 7 SECOND STAGE REVIEW	10d	DOC7-2	DOC11-2
DOCUMENT 08				
DOC8-1	DOCUMENT 8 FIRST STAGE	70d	AA001	DOC8R-1
DOC8-2	DOCUMENT 8 SECOND STAGE	40d	DOC8R-1	DOC8R-2
DOC8R-1	DOCUMENT 8 FIRST STAGE REVIEW	10d	DOC8-1	DOC8-2, DOC37-1
DOC8R-2	DOCUMENT 8 SECOND STAGE REVIEW	10d	DOC8-2	DOC37-2
DOCUMENT 09				
DOC9-1	DOCUMENT 9 FIRST STAGE	30d	DOC4R-1	DOC9R-1
DOC9-2	DOCUMENT 9 SECOND STAGE	20d	DOC9R-1, DOC4R-2	DOC9R-2
DOC9R-1	DOCUMENT 9 FIRST STAGE REVIEW	10d	DOC9-1	DOC9-2, DOC13-1
DOC9R-2	DOCUMENT 9 SECOND STAGE REVIEW	10d	DOC9-2	DOC13-2
DOCUMENT 10				
DOC10-1	DOCUMENT 10 FIRST STAGE	60d	DOC5R-1	DOC10R-1
DOC10-2	DOCUMENT 10 SECOND STAGE	20d	DOC10R-1, DOC5R-2	DOC10R-2
DOC10R-1	DOCUMENT 10 FIRST STAGE REVIEW	10d	DOC10-1	DOC10-2, DOC17-1
DOC10R-2	DOCUMENT 10 SECOND STAGE REVIEW	10d	DOC10-2	DOC17-2

Table A.14: Real-World Project Scheduling Problem: DOC11 - DOC20

Activity ID	Document Reference	Duration	Predecessors	Successors
DOCUMENT 11				
DOC11-1	DOCUMENT 11 FIRST STAGE	30d	DOC7R-1	DOC11R-1
DOC11-2	DOCUMENT 11 SECOND STAGE	10d	DOC11R-1, DOC7R-2	DOC11R-2
DOC11R-1	DOCUMENT 11 FIRST STAGE REVIEW	10d	DOC11-1	DOC11-2, DOC20-1
DOC11R-2	DOCUMENT 11 SECOND STAGE REVIEW	10d	DOC11-2	DOC20-2
DOCUMENT 12				
DOC12-1	DOCUMENT 12 FIRST STAGE	20d	AA001	DOC12R-1
DOC12-2	DOCUMENT 12 SECOND STAGE	10d	DOC12R-1	DOC12R-2
DOC12R-1	DOCUMENT 12 FIRST STAGE REVIEW	10d	DOC12-1	DOC12-2
DOC12R-2	DOCUMENT 12 SECOND STAGE REVIEW	10d	DOC12-2	AA002
DOCUMENT 13				
DOC13-1	DOCUMENT 13 FIRST STAGE	50d	DOC9R-1	DOC13R-1, DOC15-1, DOC16-1
DOC13-2	DOCUMENT 13 SECOND STAGE	25d	DOC13R-1, DOC9R-2	DOC13R-2, DOC15-2
DOC13R-1	DOCUMENT 13 FIRST STAGE REVIEW	10d	DOC13-1	DOC13-2
DOC13R-2	DOCUMENT 13 SECOND STAGE REVIEW	10d	DOC13-2	DOC15-2, DOC16-2
DOCUMENT 14				
DOC14-1	DOCUMENT 14 FIRST STAGE	80d	AA001	DOC14R-1
DOC14-2	DOCUMENT 14 SECOND STAGE	60d	DOC14R-1	DOC14R-2
DOC14R-1	DOCUMENT 14 FIRST STAGE REVIEW	10d	DOC14-1	DOC14-2
DOC14R-2	DOCUMENT 14 SECOND STAGE REVIEW	10d	DOC14-2	AA002
DOCUMENT 15				
DOC15-1	DOCUMENT 15 FIRST STAGE	60d	DOC13-1	DOC15R-1
DOC15-2	DOCUMENT 15 SECOND STAGE	40d	DOC15R-1, DOC13-2, DOC13R-2	DOC15R-2
DOC15R-1	DOCUMENT 15 FIRST STAGE REVIEW	10d	DOC15-1	DOC15-2
DOC15R-2	DOCUMENT 15 SECOND STAGE REVIEW	10d	DOC15-2	AA002
DOCUMENT 16				
DOC16-1	DOCUMENT 16 FIRST STAGE	40d	DOC13-1	DOC16R-1
DOC16-2	DOCUMENT 16 SECOND STAGE	30d	DOC16R-1, DOC13R-2	DOC16R-2
DOC16R-1	DOCUMENT 16 FIRST STAGE REVIEW	10d	DOC16-1	DOC16-2, DOC19-1, DOC23-1
DOC16R-2	DOCUMENT 16 SECOND STAGE REVIEW	10d	DOC16-2	DOC19-2, DOC23-2
DOCUMENT 17				
DOC17-1	DOCUMENT 17 FIRST STAGE	40d	DOC10R-1	DOC17R-1
DOC17-2	DOCUMENT 17 SECOND STAGE	20d	DOC17R-1, DOC10R-2	DOC17R-2
DOC17R-1	DOCUMENT 17 FIRST STAGE REVIEW	10d	DOC17-1	DOC17-2
DOC17R-2	DOCUMENT 17 SECOND STAGE REVIEW	10d	DOC17-2	AA002
DOCUMENT 18				
DOC18-1	DOCUMENT 18 FIRST STAGE	50d	AA001	DOC18R-1
DOC18-2	DOCUMENT 18 SECOND STAGE	20d	DOC18R-1	DOC18R-2
DOC18R-1	DOCUMENT 18 FIRST STAGE REVIEW	10d	DOC18-1	DOC18-2, DOC22-1
DOC18R-2	DOCUMENT 18 SECOND STAGE REVIEW	10d	DOC18-2	DOC22-1
DOCUMENT 19				
DOC19-1	DOCUMENT 19 FIRST STAGE	60d	DOC16R-1	DOC19R-1
DOC19-2	DOCUMENT 19 SECOND STAGE	40d	DOC19R-1, DOC16R-2	DOC19R-2
DOC19R-1	DOCUMENT 19 FIRST STAGE REVIEW	10d	DOC19-1	DOC19-2
DOC19R-2	DOCUMENT 19 SECOND STAGE REVIEW	10d	DOC19-2	AA002
DOCUMENT 20				
DOC20-1	DOCUMENT 20 FIRST STAGE	70d	DOC11R-1	DOC20R-1
DOC20-2	DOCUMENT 20 SECOND STAGE	50d	DOC20R-1, DOC11R-2	DOC20R-2
DOC20R-1	DOCUMENT 20 FIRST STAGE REVIEW	10d	DOC20-1	DOC20-2
DOC20R-2	DOCUMENT 20 SECOND STAGE REVIEW	10d	DOC20-2	AA002

Table A.15: Real-World Project Scheduling Problem

Activity ID	Document Reference	Duration	Predecessors	Successors
DOCUMENT 21				
DOC21-1	DOCUMENT 21 FIRST STAGE	50d	AA001	DOC21-2
DOC21-2	DOCUMENT 21 SECOND STAGE	20d	DOC21-1	DOC21R-1
DOC21R-1	DOCUMENT 21 FIRST STAGE REVIEW	10d	DOC21-2	DOC21R-2, DOC36-1
DOC21R-2	DOCUMENT 21 SECOND STAGE REVIEW	10d	DOC21R-1	DOC36-2
DOCUMENT 22				
DOC22-1	DOCUMENT 22 FIRST STAGE	70d	DOC18R-1, DOC18R-2	DOC22-2
DOC22-2	DOCUMENT 22 SECOND STAGE	40d	DOC22-1	DOC22R-1
DOC22R-1	DOCUMENT 22 FIRST STAGE REVIEW	10d	DOC22-2	DOC22R-2, DOC31-1
DOC22R-2	DOCUMENT 22 SECOND STAGE REVIEW	10d	DOC22R-1	DOC31-2
DOCUMENT 23				
DOC23-1	DOCUMENT 23 FIRST STAGE	40d	DOC16R-1	DOC23-2
DOC23-2	DOCUMENT 23 SECOND STAGE	30d	DOC23-1, DOC16R-2	DOC23R-1
DOC23R-1	DOCUMENT 23 FIRST STAGE REVIEW	10d	DOC23-2	DOC23R-2, DOC25-1
DOC23R-2	DOCUMENT 23 SECOND STAGE REVIEW	10d	DOC23R-1	DOC25-2
DOCUMENT 24				
DOC24-1	DOCUMENT 24 FIRST STAGE	60d	DOC5R-1	DOC24-2
DOC24-2	DOCUMENT 24 SECOND STAGE	25d	DOC24-1, DOC5R-2	DOC24R-1
DOC24R-1	DOCUMENT 24 FIRST STAGE REVIEW	10d	DOC24-2	DOC24R-2, DOC32-1
DOC24R-2	DOCUMENT 24 SECOND STAGE REVIEW	10d	DOC24R-1	DOC32-2
DOCUMENT 25				
DOC25-1	DOCUMENT 25 FIRST STAGE	80d	DOC23R-1	DOC25-2
DOC25-2	DOCUMENT 25 SECOND STAGE	60d	DOC25-1, DOC23R-2	DOC25R-1
DOC25R-1	DOCUMENT 25 FIRST STAGE REVIEW	10d	DOC25-2	DOC25R-2
DOC25R-2	DOCUMENT 25 SECOND STAGE REVIEW	10d	DOC25R-1	AA002
DOCUMENT 26				
DOC26-1	DOCUMENT 26 FIRST STAGE	60d	AA001	DOC26-2
DOC26-2	DOCUMENT 26 SECOND STAGE	40d	DOC26-1	DOC26R-1
DOC26R-1	DOCUMENT 26 FIRST STAGE REVIEW	10d	DOC26-2	DOC26R-2, DOC27-1
DOC26R-2	DOCUMENT 26 SECOND STAGE REVIEW	10d	DOC26R-1	DOC27-2
DOCUMENT 27				
DOC27-1	DOCUMENT 27 FIRST STAGE	60d	DOC26R-1	DOC27-2
DOC27-2	DOCUMENT 27 SECOND STAGE	20d	DOC27-1, DOC26R-2	DOC27R-1
DOC27R-1	DOCUMENT 27 FIRST STAGE REVIEW	10d	DOC27-2	DOC27R-2, DOC44-1
DOC27R-2	DOCUMENT 27 SECOND STAGE REVIEW	10d	DOC27R-1	DOC44-2
DOCUMENT 28				
DOC28-1	DOCUMENT 28 FIRST STAGE	90d	AA001	DOC28-2
DOC28-2	DOCUMENT 28 SECOND STAGE	70d	DOC28-1	DOC28R-1
DOC28R-1	DOCUMENT 28 FIRST STAGE REVIEW	10d	DOC28-2	DOC28R-2
DOC28R-2	DOCUMENT 28 SECOND STAGE REVIEW	10d	DOC28R-1	AA002
DOCUMENT 29				
DOC29-1	DOCUMENT 29 FIRST STAGE	45d	DOC2R-1	DOC29-2
DOC29-2	DOCUMENT 29 SECOND STAGE	30d	DOC29-1, DOC2R-2	DOC29R-1
DOC29R-1	DOCUMENT 29 FIRST STAGE REVIEW	10d	DOC29-2	DOC29R-2, DOC30-1
DOC29R-2	DOCUMENT 29 SECOND STAGE REVIEW	10d	DOC29R-1	DOC30-2
DOCUMENT 30				
DOC30-1	DOCUMENT 30 FIRST STAGE	50d	DOC29R-1	DOC30-2
DOC30-2	DOCUMENT 30 SECOND STAGE	30d	DOC30-1, DOC29R-2	DOC30R-1
DOC30R-1	DOCUMENT 30 FIRST STAGE REVIEW	10d	DOC30-2	DOC30R-2, DOC33-1
DOC30R-2	DOCUMENT 30 SECOND STAGE REVIEW	10d	DOC30R-1	DOC33-2

Table A.16: Real-World Project Scheduling Problem: DOC21 - DOC30

Activity ID	Document Reference	Duration	Predecessors	Successors
DOCUMENT 31				
DOC31-1	DOCUMENT 31 FIRST STAGE	60d	DOC22R-1	DOC31-2
DOC31-2	DOCUMENT 31 SECOND STAGE	20d	DOC31-1, DOC22R-2	DOC31R-1
DOC31R-1	DOCUMENT 31 FIRST STAGE REVIEW	10d	DOC31-2	DOC31R-2, DOC42-1
DOC31R-2	DOCUMENT 31 SECOND STAGE REVIEW	10d	DOC31R-1	DOC42-2
DOCUMENT 32				
DOC32-1	DOCUMENT 32 FIRST STAGE	70d	DOC24R-1	DOC32-2
DOC32-2	DOCUMENT 32 SECOND STAGE	40d	DOC32-1, DOC24R-2	DOC32R-1
DOC32R-1	DOCUMENT 32 FIRST STAGE REVIEW	10d	DOC32-2	DOC32R-2, DOC43-1
DOC32R-2	DOCUMENT 32 SECOND STAGE REVIEW	10d	DOC32R-1	DOC43-2
DOCUMENT 33				
DOC33-1	DOCUMENT 33 FIRST STAGE	50d	DOC30R-1	DOC33-2
DOC33-2	DOCUMENT 33 SECOND STAGE	25d	DOC33-1, DOC30R-2	DOC33R-1
DOC33R-1	DOCUMENT 33 FIRST STAGE REVIEW	10d	DOC33-2	DOC33R-2
DOC33R-2	DOCUMENT 33 SECOND STAGE REVIEW	10d	DOC33R-1	AA002
DOCUMENT 34				
DOC34-1	DOCUMENT 34 FIRST STAGE	50d	AA001	DOC34-2
DOC34-2	DOCUMENT 34 SECOND STAGE	20d	DOC34-1	DOC34R-1
DOC34R-1	DOCUMENT 34 FIRST STAGE REVIEW	10d	DOC34-2	DOC34R-2, DOC35-1
DOC34R-2	DOCUMENT 34 SECOND STAGE REVIEW	10d	DOC34R-1	DOC35-2
DOCUMENT 35				
DOC35-1	DOCUMENT 35 FIRST STAGE	80d	DOC34R-1	DOC35-2
DOC35-2	DOCUMENT 35 SECOND STAGE	60d	DOC35-1, DOC34R-2	DOC35R-1
DOC35R-1	DOCUMENT 35 FIRST STAGE REVIEW	10d	DOC35-2	DOC35R-2
DOC35R-2	DOCUMENT 35 SECOND STAGE REVIEW	10d	DOC35R-1	AA002
DOCUMENT 36				
DOC36-1	DOCUMENT 36 FIRST STAGE	50d	DOC21R-1	DOC36-2
DOC36-2	DOCUMENT 36 SECOND STAGE	20d	DOC36-1, DOC21R-2	DOC36R-1
DOC36R-1	DOCUMENT 36 FIRST STAGE REVIEW	10d	DOC36-2	DOC36R-2, DOC39-1
DOC36R-2	DOCUMENT 36 SECOND STAGE REVIEW	10d	DOC36R-1	DOC39-2
DOCUMENT 37				
DOC37-1	DOCUMENT 37 FIRST STAGE	70d	DOC8R-1	DOC37-2
DOC37-2	DOCUMENT 37 SECOND STAGE	20d	DOC37-1, DOC8R-2	DOC37R-1
DOC37R-1	DOCUMENT 37 FIRST STAGE REVIEW	10d	DOC37-2	DOC37R-2
DOC37R-2	DOCUMENT 37 SECOND STAGE REVIEW	10d	DOC37R-1	AA002
DOCUMENT 38				
DOC38-1	DOCUMENT 38 FIRST STAGE	80d	AA001	DOC38-2
DOC38-2	DOCUMENT 38 SECOND STAGE	60d	DOC38-1	DOC38R-1
DOC38R-1	DOCUMENT 38 FIRST STAGE REVIEW	10d	DOC38-2	DOC38R-2, DOC40-1
DOC38R-2	DOCUMENT 38 SECOND STAGE REVIEW	10d	DOC38R-1	DOC40-2
DOCUMENT 39				
DOC39-1	DOCUMENT 39 FIRST STAGE	60d	DOC36R-1	DOC39-2
DOC39-2	DOCUMENT 39 SECOND STAGE	40d	DOC39-1, DOC36R-2	DOC39R-1
DOC39R-1	DOCUMENT 39 FIRST STAGE REVIEW	10d	DOC39-2	DOC39R-2
DOC39R-2	DOCUMENT 39 SECOND STAGE REVIEW	10d	DOC39R-1	AA002
DOCUMENT 40				
DOC40-1	DOCUMENT 40 FIRST STAGE	70d	DOC38R-1	DOC40-2
DOC40-2	DOCUMENT 40 SECOND STAGE	40d	DOC40-1, DOC38R-2	DOC40R-1
DOC40R-1	DOCUMENT 40 FIRST STAGE REVIEW	10d	DOC40-2	DOC40R-2, DOC41-1
DOC40R-2	DOCUMENT 40 SECOND STAGE REVIEW	10d	DOC40R-1	DOC41-2

Table A.17: Real-World Project Scheduling Problem: DOC41 - DOC45

Activity ID	Document Reference	Duration	Predecessors	Successors
DOCUMENT 41				
DOC41-1	DOCUMENT 41 FIRST STAGE	70d	DOC40R-1	DOC41-2
DOC41-2	DOCUMENT 41 SECOND STAGE	40d	DOC41-1, DOC40R-2	DOC41R-1
DOC41R-1	DOCUMENT 41 FIRST STAGE REVIEW	10d	DOC41-2	DOC41R-2
DOC41R-2	DOCUMENT 41 SECOND STAGE REVIEW	10d	DOC41R-1	AA002
DOCUMENT 42				
DOC42-1	DOCUMENT 42 FIRST STAGE	60d	DOC31R-1	DOC42-2
DOC42-2	DOCUMENT 42 SECOND STAGE	30d	DOC42-1, DOC31R-2	DOC42R-1
DOC42R-1	DOCUMENT 42 FIRST STAGE REVIEW	10d	DOC42-2	DOC42R-2
DOC42R-2	DOCUMENT 42 SECOND STAGE REVIEW	10d	DOC42R-1	AA002
DOCUMENT 43				
DOC43-1	DOCUMENT 43 FIRST STAGE	70d	DOC32R-1	DOC43-2
DOC43-2	DOCUMENT 43 SECOND STAGE	40d	DOC43-1, DOC32R-2	DOC43R-1
DOC43R-1	DOCUMENT 43 FIRST STAGE REVIEW	10d	DOC43-2	DOC43R-2
DOC43R-2	DOCUMENT 43 SECOND STAGE REVIEW	10d	DOC43R-1	AA002
DOCUMENT 44				
DOC44-1	DOCUMENT 44 FIRST STAGE	60d	DOC27R-1	DOC44-2
DOC44-2	DOCUMENT 44 SECOND STAGE	20d	DOC44-1, DOC27R-2	DOC44R-1
DOC44R-1	DOCUMENT 44 FIRST STAGE REVIEW	10d	DOC44-2	DOC44R-2, DOC45-1
DOC44R-2	DOCUMENT 44 SECOND STAGE REVIEW	10d	DOC44R-1	DOC45-2
DOCUMENT 45				
DOC45-1	DOCUMENT 45 FIRST STAGE	70d	DOC44R-1	DOC45-2
DOC45-2	DOCUMENT 45 SECOND STAGE	40d	DOC45-1, DOC44R-2	DOC45R-1
DOC45R-1	DOCUMENT 45 FIRST STAGE REVIEW	10d	DOC45-2	DOC45R-2
DOC45R-2	DOCUMENT 45 SECOND STAGE REVIEW	10d	DOC45R-1	AA002

Appendix B

Figures

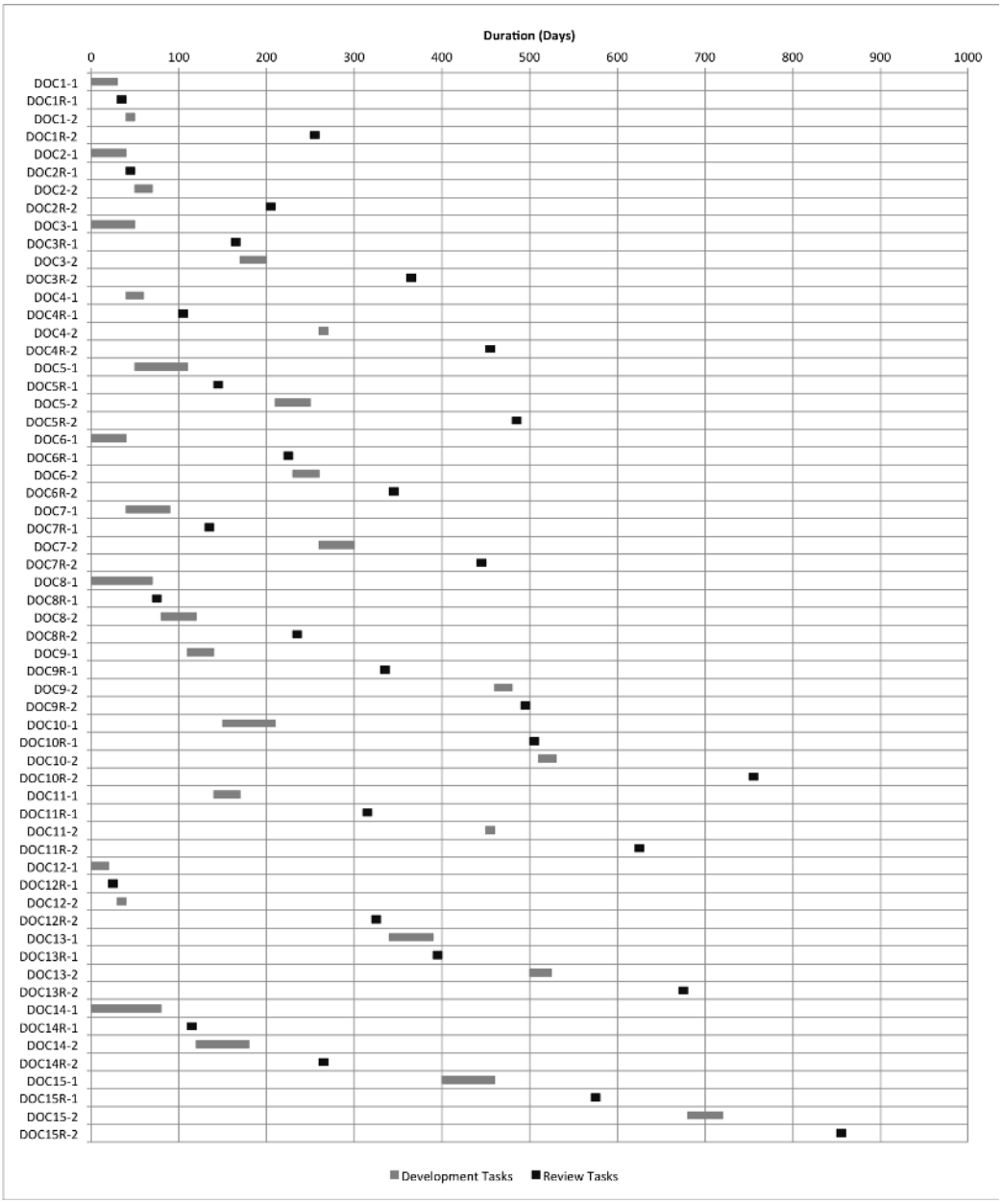

Figure B.1: RCPSP Schedule: DOCs 1-15

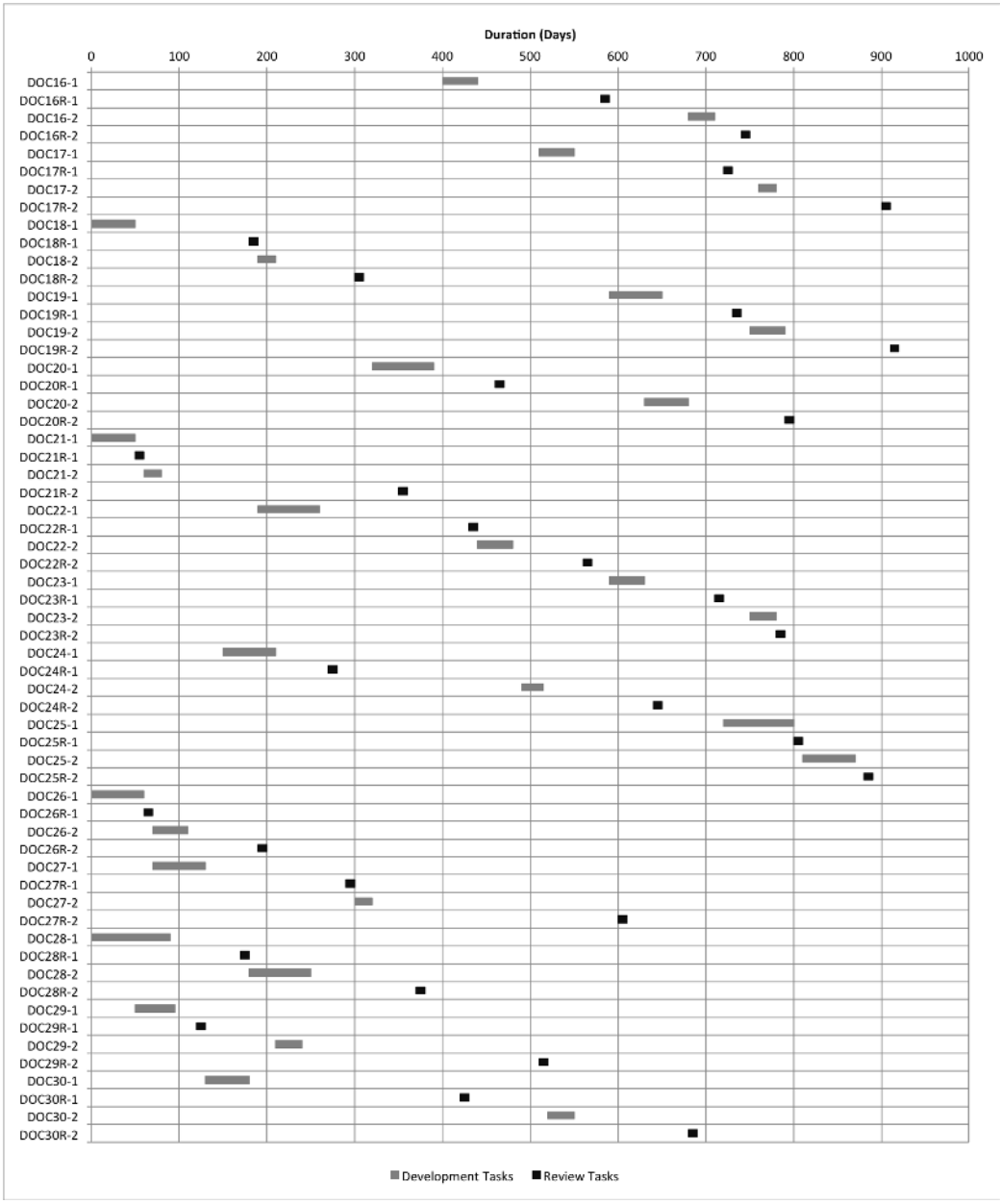

Figure B.2: RCPSP Schedule: DOCs 16-30

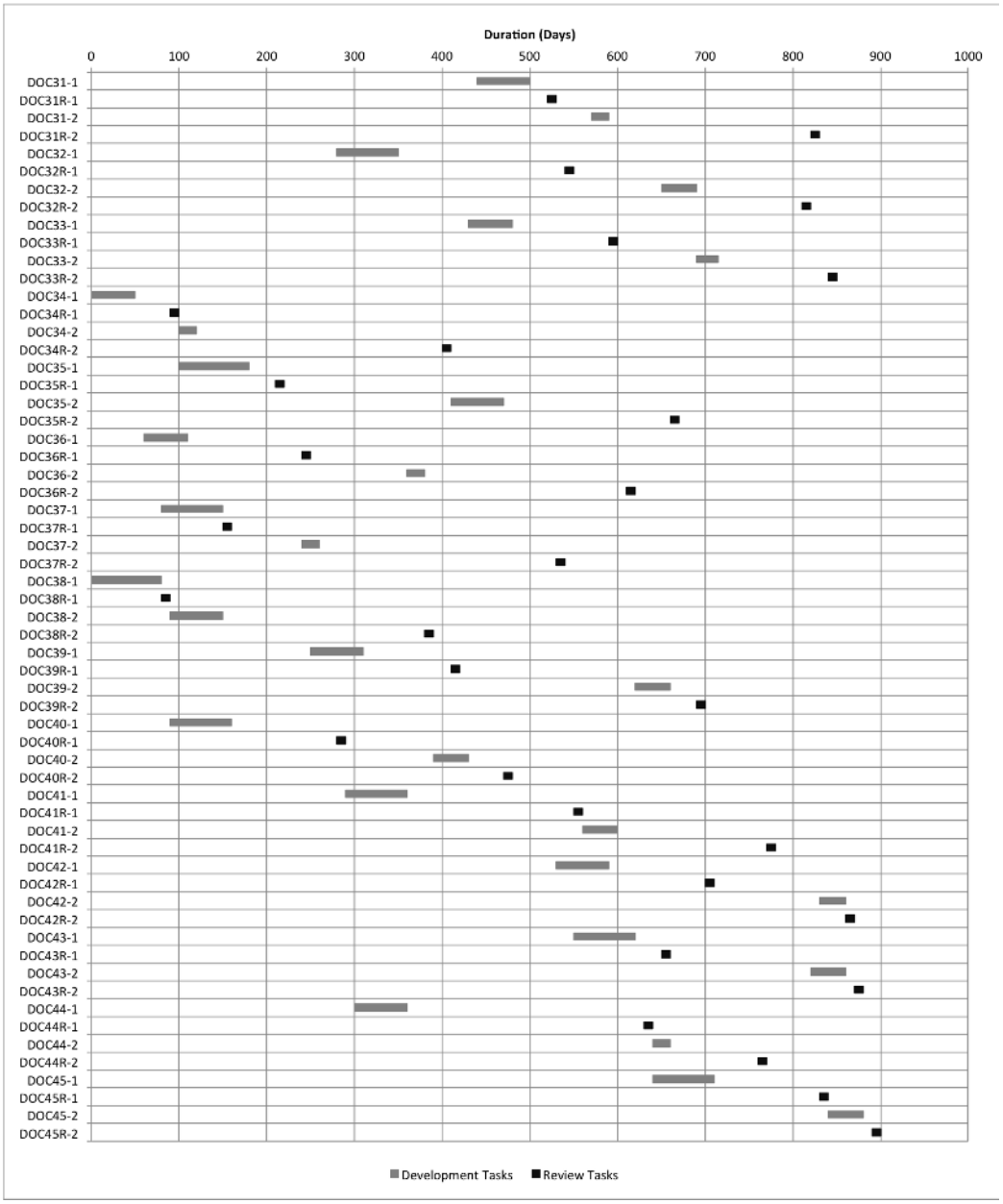

Figure B.3: RCPSP Schedule: DOCs 31-45

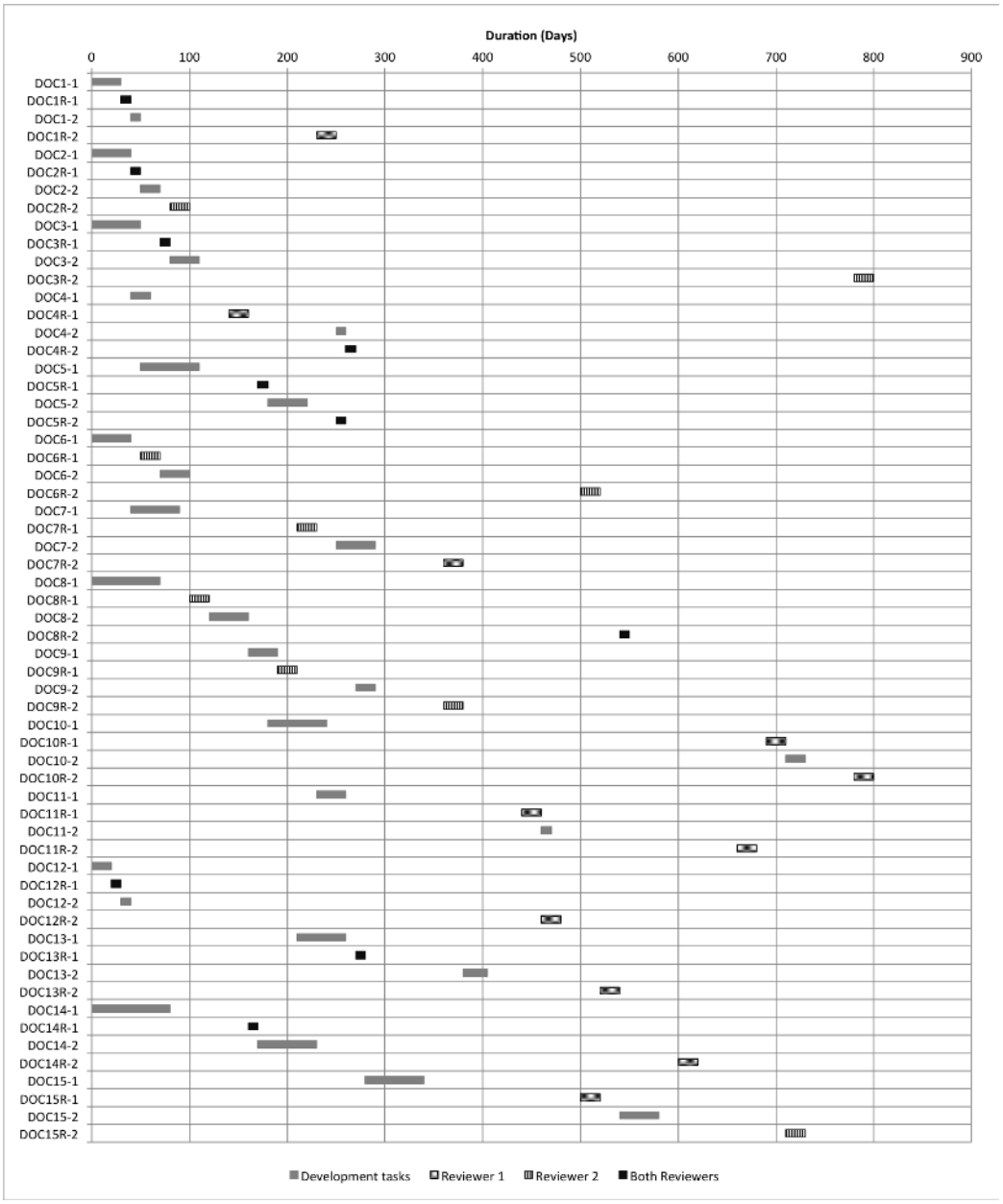

Figure B.4: MRCPSP Schedule: DOCs 1-15

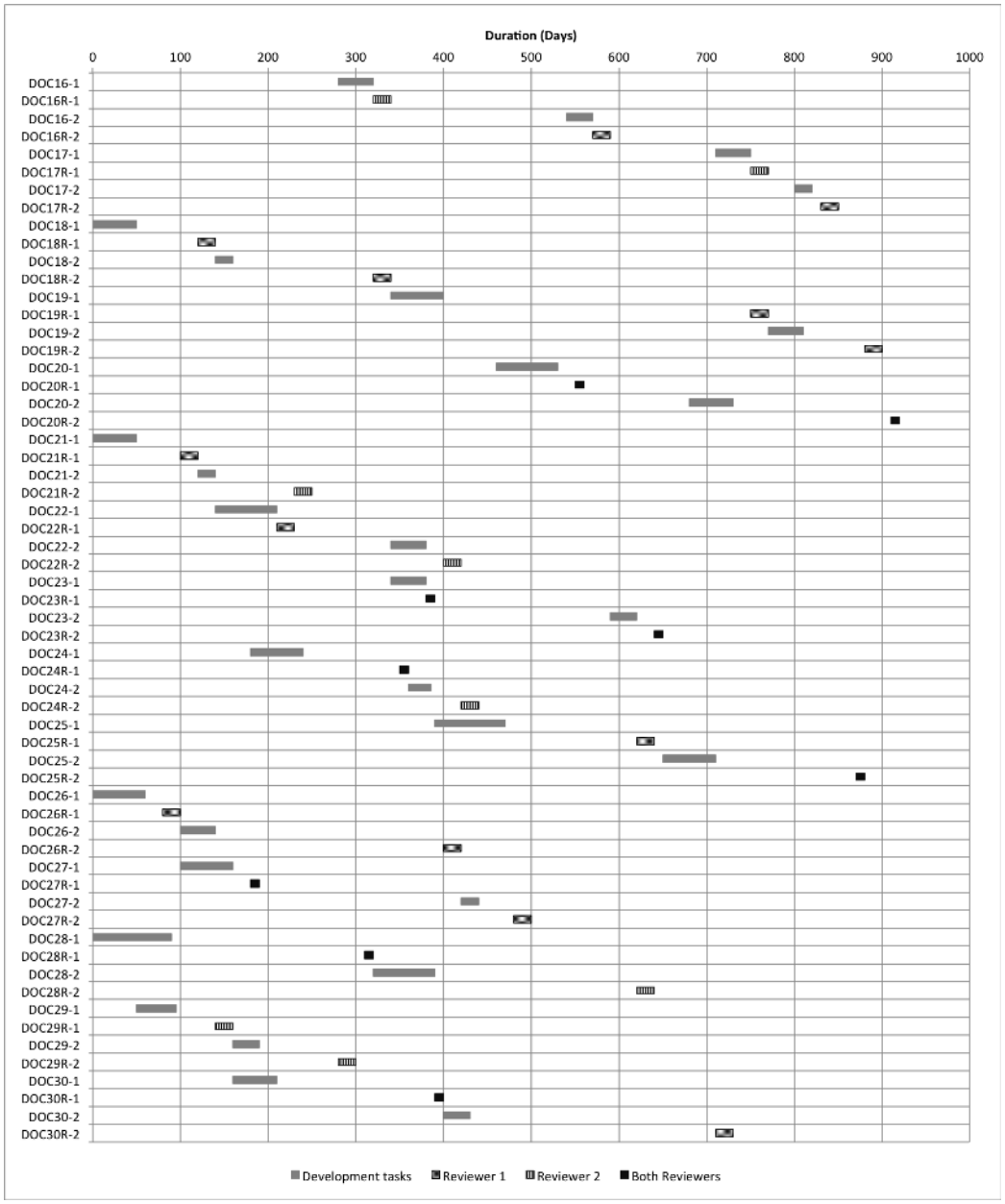

Figure B.5: MRCPSP Schedule: DOCs 16-30

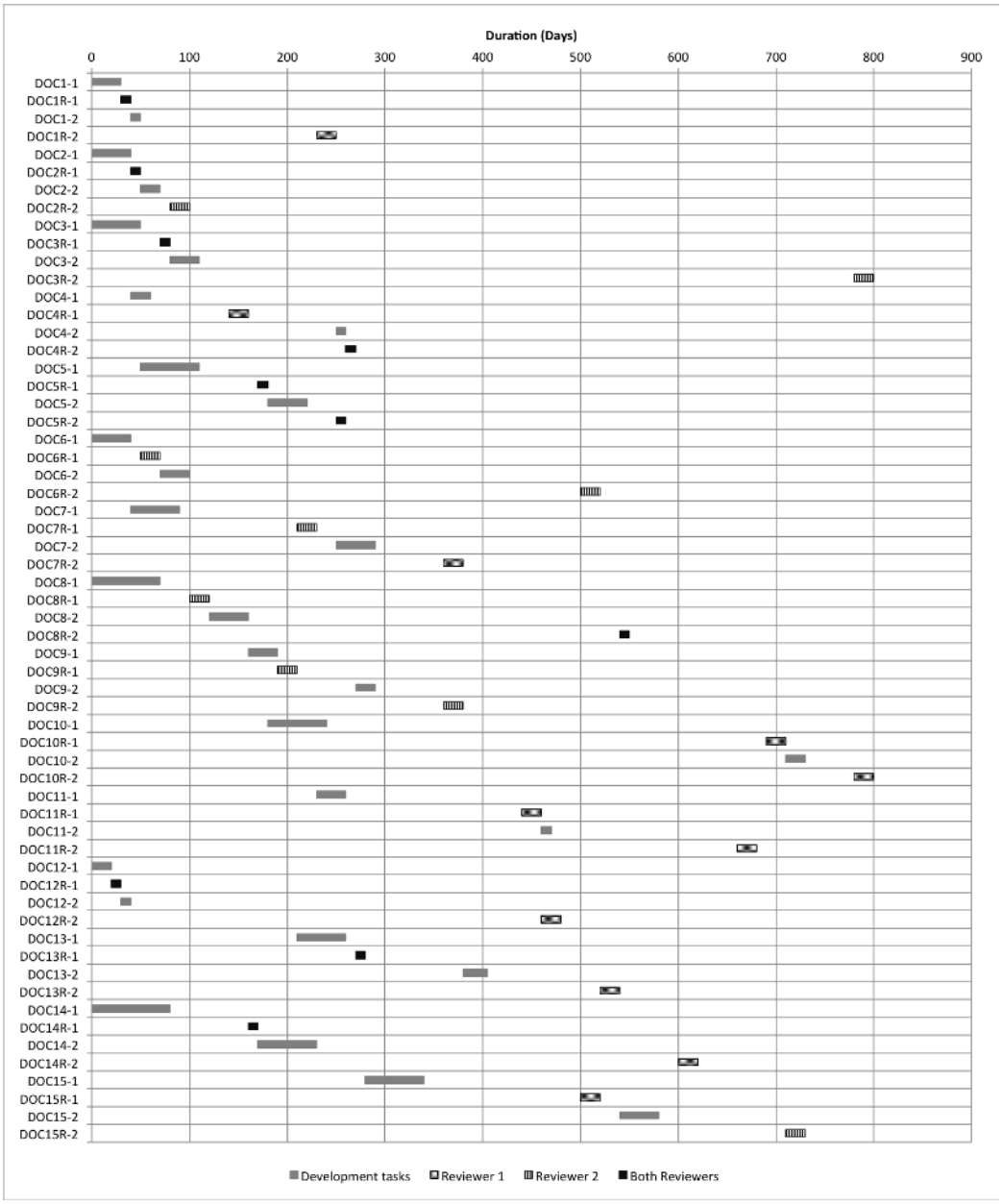

Figure B.6: MRCPSP Schedule: DOCs 31-45

Bibliography

Abdollahzadeh A, Reynolds A, Christie M, Corne DW, Williams GJ, Davies BJ, et al (2013) Estimation of distribution algorithms applied to history matching. SPE Journal 18(03):508–517

Aickelin U, Li J (2007) An estimation of distribution algorithm for nurse scheduling. Annals of Operations Research 155(1):289–309

Aickelin U, Burke EK, Li J (2006) An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. Journal of the Operational Research Society 58(12):1574–1585

Alcaraz J, Maroto C (2001) A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research 102(1-4):83–109

Alcaraz J, Maroto C, Ruiz R (2003) Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54(6):614–626

Ayodele M, McCall J, Regnier-Coudert O (2015) Probabilistic model enhanced genetic algorithm for multi-mode resource constrained project scheduling problem. In: Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference, ACM, pp 745–746

Ayodele M, McCall J, Regnier-Coudert O (2016a) BPGA-EDA for the multi-mode resource constrained project scheduling problem. In: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, pp 3417–3424

Ayodele M, McCall J, Regnier-Coudert O (2016b) RK-EDA: A novel random key based estimation of distribution algorithm. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 849–858

Ayodele M, McCall J, Regnier-Coudert O (2017a) Estimation of distribution algorithms for the multi-mode resource constrained project scheduling problem. In: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, pp 1579–1586

Ayodele M, McCall J, Regnier-Coudert O, Bowie L (2017b) A random key based estimation of distribution algorithm for the permutation flowshop scheduling problem. In: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, pp 2364–2371

Baluja S (1994) Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Tech. rep., DTIC Document

Baluja S, Davies S (1997) Combining multiple optimization runs with optimal dependency trees. Tech. rep., DTIC Document

Baluja S, Davies S (1998) Fast probabilistic modeling for combinatorial optimization. In: AAAI/IAAI, pp 469–476

Bengoetxea E, Larrañaga P, Bloch I, Perchant A, Boeres C (2002) Inexact graph matching by means of estimation of distribution algorithms. Pattern Recognition 35(12):2867–2880

Blażewicz J, Domschke W, Pesch E (1996) The job shop scheduling problem: Conventional and new solution techniques. European journal of operational research 93(1):1–33

Blum C, Dorigo M (2004) The hyper-cube framework for ant colony optimization. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34(2):1161–1172

Blum C, Roli A, Dorigo M (2001) Hc–aco: The hyper-cube framework for ant colony optimization. In: Proceedings of MIC, vol 2, pp 399–403

Bonyadi MR, Barone L, Michalewicz Z (2013) The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico

Bosman PA, Thierens D (2001) Crossing the road to efficient ideas for permutation problems. In: Proceedings of the 6th annual conference on Genetic and evolutionary computation, ACM, pp 219–226

Bosman PA, Thierens D (2007) Adaptive variance scaling in continuous multi-objective estimation-of-distribution algorithms. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, pp 500–507

Bosman PA, Thierens D (2013) More concise and robust linkage learning by filtering and combining linkage hierarchies. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, ACM, pp 359–366

Bouleimen K, Lecocq H (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 149(2):268–281

Braekers K, Ramaekers K, Van Nieuwenhuyse I (2016) The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99:300–313

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation science 39(1):119–139

Castillo-Salazar JA, Landa-Silva D, Qu R (2016) Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research 239(1):39–67

Ceberio J (2014) Solving permutation problems with estimation of distribution algorithms and extensions thereof. PhD thesis, PhD thesis, Faculty of Computer Science, University of the Basque Country

Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2012) A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Progress in Artificial Intelligence 1(1):103–117

Ceberio J, Mendiburu A, Lozano JA (2013) The plackett-luce ranking model on permutation-based optimization problems. In: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, pp 494–501

Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2014a) A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. Evolutionary Computation, IEEE Transactions on 18(2):286–300

Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2014b) Extending distance-based ranking models in estimation of distribution algorithms. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, pp 2459–2466

Chand S, Singh HK, Ray T (2017) A heuristic algorithm for solving resource constrained project scheduling problems. In: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, pp 225–232

Chaves-Gonzalez JM, Dominguez-Gonzalez D, Vega-Rodriguez MA, Gomez-Pulido JA, Sanchez-Perez JM (2008) Parallelizing pbil for solving a real-world frequency assignment problem in gsm networks. In: Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on, IEEE, pp 391–398

Chen RM (2011) Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. Expert Systems with Applications 38(6):7102–7111

Chen W, Shi Yj, Teng Hf, Lan Xp, Hu Lc (2010) An efficient hybrid algorithm for resource-constrained project scheduling. Information Sciences 180(6):1031–1039

Chica M, Cordón O, Damas S, Bautista J (2011) Including different kinds of preferences in a multi-objective ant algorithm for time and space assembly line balancing on different nissan scenarios. Expert Systems with Applications 38(1):709–720

Chiong R, Weise T, Michalewicz Z (2012) Variants of evolutionary algorithms for real-world applications. Springer

Coelho J, Vanhoucke M (2011) Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. European Journal of Operational Research 213(1):73–82

Damak N, Jarboui B, Siarry P, Loukil T (2009) Differential evolution for solving multi-mode resource-constrained project scheduling problems. Computers & Operations Research 36(9):2653–2659

De Bonet JS, Isbell CL, Viola P, et al (1997) Mimic: Finding optima by estimating probability densities. Advances in neural information processing systems pp 424–430

Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. John Wiley & Sons

Debels D, Vanhoucke M (2005) A bi-population based genetic algorithm for the resource-constrained project scheduling problem. In: International Conference on Computational Science and Its Applications, Springer, pp 378–387

Debels D, Vanhoucke M (2007) A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. Operations Research 55(3):457–469

Debels D, De Reyck B, Leus R, Vanhoucke M (2006) A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research 169(2):638–653

Demirbas A, Alsulami H, Nizami AS (2016) The natural gas potential of saudi arabia. Energy Sources, Part A: Recovery, Utilization, and Environmental Effects 38(18):2635–2642

Donati AV, Montemanni R, Casagrande N, Rizzoli AE, Gambardella LM (2008) Time dependent vehicle routing problem with a multi ant colony system. European journal of operational research 185(3):1174–1191

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26(1):29–41

Dorigo M, Di Caro G, Gambardella LM (1999) Ant algorithms for discrete optimization. Artificial life 5(2):137–172

Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. Computational Intelligence Magazine, IEEE 1(4):28–39

DEste P, Patel P (2007) University–industry linkages in the uk: What are the factors underlying the variety of interactions with industry? Research policy 36(9):1295–1313

Elloumi S, Fortemps P (2010) A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. European Journal of Operational Research 205(1):31–41

Engelbrecht AP (2007) Computational intelligence: an introduction. John Wiley & Sons

Etxeberria R, Larranaga P (1999) Global optimization using bayesian networks. In: Second Symposium on Artificial Intelligence (CIMAF-99), pp 332–339

Fang C, Wang L (2012) An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. Computers & Operations Research 39(5):890–901

Fischer A, Greiff S, Funke J (2011) The process of solving complex problems. Journal of Problem Solving 4(1):19–42

Fligner MA, Verducci JS (1988) Multistage ranking models. Journal of the American Statistical association 83(403):892–901

Foong WK, Maier HR, Simpson AR (2005) Ant colony optimization for power plant maintenance scheduling optimization. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM, pp 249–256

Foong WK, Simpson AR, Maier HR, Stolp S (2008) Ant colony optimization for power plant maintenance scheduling optimizationa five-station hydropower system. Annals of Operations Research 159(1):433–450

Gagné C, Gravel M, Price WL (2006) Solving real car sequencing problems with ant colony optimization. European Journal of Operational Research 174(3):1427–1448

Gambardella LM, Rizzoli AE, Oliverio F, Casagrande N, Donati AV, Montemanni R, Lucibello E (2003) Ant colony optimization for vehicle routing in advanced logistics systems. In: Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003), pp 3–9

Geiger MJ (2016) A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. European Journal of Operational Research

Goncharov E, Leonov V (2017) Genetic algorithm for the resource-constrained project scheduling problem. Automation and Remote Control 78(6):1101–1114

González-Barbosa JJ, Delgado-Orta JF, Cruz-Reyes L, Fraire-Huacuja HJ, Ramirez-Saldivar A (2010) Comparative analysis of hybrid techniques for an ant colony system algorithm applied to solve a real-world transportation problem. In: Soft Computing for Recognition Based on Biometrics, Springer, pp 365–385

Gravel M, Price WL, Gagné C (2000) Scheduling jobs in an alcan aluminium foundry using a genetic algorithm. International Journal of Production Research 38(13):3031–3041

Gravel M, Price WL, Gagné C (2002) Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. European Journal of Operational Research 143(1):218–229

Gupta DK, Arora Y, Singh UK, Gupta JP (2012) Recursive ant colony optimization for estimation of parameters of a function. In: Recent Advances in Information Technology (RAIT), 2012 1st International Conference on, IEEE, pp 448–454

Gutjahr WJ, Rauner MS (2007) An aco algorithm for a dynamic regional nurse-scheduling problem in austria. Computers & Operations Research 34(3):642–666

Hajizadeh Y, Christie M, Demyanov V (2011) Ant colony optimization for history matching and uncertainty quantification of reservoir models. Journal of Petroleum Science and Engineering 77(1):78–92

Hämmerle A, Ankerl M (2013) Solving a vehicle routing problem with ant colony optimisation and stochastic ranking. In: Computer Aided Systems Theory-EUROCAST 2013, Springer, pp 259–266

Hani Y, Amodeo L, Yalaoui F, Chen H (2007) Ant colony optimization for solving an industrial layout problem. European Journal of Operational Research 183(2):633–642

Harik G (1999) Linkage learning via probabilistic modeling in the ecga. Urbana 51(61):801

Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. Evolutionary Computation, IEEE Transactions on 3(4):287–297

Hart E, Ross P, Corne D (2005) Evolutionary scheduling: A review. Genetic Programming and Evolvable Machines 6(2):191–220

Hart E, Sim K, Urquhart N (2014) A real-world employee scheduling and routing application. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, pp 1239–1242

Hartmann S (2001) Project scheduling with multiple modes: a genetic algorithm. Annals of Operations Research 102(1-4):111–135

Hartmann S (2002) A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics (NRL) 49(5):433–448

Hauschild M, Pelikan M (2011) An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1(3):111–128

Heckerman D, Geiger D, Chickering DM (1995) Learning bayesian networks: The combination of knowledge and statistical data. Machine learning 20(3):197–243

Hirsch P, Palfi A, Gronalt M (2012) Solving a time constrained two-crane routing problem for material handling with an ant colony optimisation approach: an application in the roof-tile industry. International Journal of Production Research 50(20):6005–6021

Hu XM, Zhang J, Li Y (2008) Orthogonal methods based ant colony search for solving continuous optimization problems. Journal of computer science and technology 23(1):2–18

Hutzschenreuter AK, Bosman PA, La Poutré H (2009) Evolutionary multiobjective optimization for dynamic hospital resource management. In: Evolutionary Multi-criterion Optimization, Springer, pp 320–334

Jordan MI, et al (2004) Graphical models. Statistical Science 19(1):140–155

Kahraman C, Kaya I, Çinar D (2010) Computational intelligence: Past, today, and future. In: Computational Intelligence in Complex Decision Systems, Springer, pp 1–46

Knjazew D, Goldberg DE (2000) Omega-ordering messy ga: Solving permutation problems with the fast messy genetic algorithm and random keys. In: Proceedings of Genetic and Evolutionary Computation Conference, pp 181–188

Kolisch R (1996) Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research 90(2):320–333

Kolisch R, Drexl A (1996) Adaptive search for solving hard project scheduling problems. Naval Research Logistics (NRL) 43(1):23–40

Kolisch R, Hartmann S (1999) Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. Springer

Kolisch R, Sprecher A (1997) PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. European Journal of Operational Research 96(1):205–216

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT press

Larrañaga P, Lozano JA (2002) Estimation of distribution algorithms: A new tool for evolutionary computation, vol 2. Springer

Larrañaga P, Moral S (2011) Probabilistic graphical models in artificial intelligence. Applied soft computing 11(2):1511–1528

Larranaga P, Etxeberria R, Lozano J, Pena J, Pe J, et al (1999) Optimization by learning and simulation of bayesian and gaussian networks. Tech. rep., Department of Computer Science and Artificial Intelligence, University of the Basque Country

Larrañaga P, Etxeberria R, Lozano JA, Peña JM (2000) Optimization in Continuous Domains by Learning and Simulation of Gaussian Networks. In: Workshop in Optimization by Building and using Probabilistic Models, Las Vegas, Nevada, USA, A Workshop withing the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000, pp 201–204

Linn R, Zhang W (1999) Hybrid flow shop scheduling: a survey. Computers & Industrial Engineering 37(1):57–61

Lova A, Tormos P, Cervantes M, Barber F (2009) An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. International Journal of Production Economics 117(2):302–316

Lozano J, Mendiburu A (2002) Estimation of distribution algorithms applied to the job shop scheduling problem: Some preliminary research. In: Estimation of Distribution Algorithms, Springer, pp 231–242

Martí R, Reinelt G, Duarte A (2012) A benchmark library and a comparison of heuristic methods for the linear ordering problem. Computational optimization and applications 51(3):1297–1317

Michalewicz Z (2012a) Quo vadis, evolutionary computation? In: Advances in Computational Intelligence, Springer, pp 98–121

Michalewicz Z (2012b) Ubiquity symposium: Evolutionary computation and the processes of life: the emperor is naked: evolutionary algorithms for real-world applications. Ubiquity 2012(November):3

Michalewicz Z, Fogel DB (2000) How to solve it: Modern Heuristics. Springer New York

Montemanni R, Gambardella LM, Rizzoli AE, Donati AV (2005) Ant colony system for a dynamic vehicle routing problem. Journal of Combinatorial Optimization 10(4):327–343

Montgomery J (2007) Alternative solution representations for the job shop scheduling problem in ant colony optimisation. In: Progress in Artificial Life, Springer, pp 1–12

Montgomery J, Fayad C, Petrovic S (2006) Solution representation for job shop scheduling problems in ant colony optimisation. In: Ant Colony Optimization and Swarm Intelligence, Springer, pp 484–491

Mühlenbein H, Mahnig T (1999) Fda-a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolutionary computation 7(4):353–376

Nechita E, Crisan GC, Talmaciu M (2008) Cooperative ant colonies for vehicle routing problem with time windows. a case study in the distribution of dietary products. In: Proceedings of the 12th World Multiconference on Systemics, Cybernetics and Informatics (WMSCI 2008) Orlando, USA, vol 5

Ochiai J, Kanoh H (2014) Hybrid ant colony optimization for real-world delivery problems based on real time and predicted traffic in wide area road network. In: Fourth International conference on Computer Science and Information Technology-CCSIT, vol 4

Oliver DS, Chen Y (2011) Recent progress on reservoir history matching: a review. Computational Geosciences 15(1):185–221

Olsson RJ, Kapelan Z, Savic DA (2009) Probabilistic building block identification for the optimal design and rehabilitation of water distribution systems. Journal of Hydroinformatics 11(2):89–105

Pelikan M (2005a) Bayesian optimization algorithm. In: Hierarchical Bayesian Optimization Algorithm, Springer, pp 31–48

Pelikan M (2005b) Hierarchical Bayesian optimization algorithm. Springer

Pelikan M, Mühlenbein H (1998) Marginal distributions in evolutionary algorithms. In: Proceedings of the International Conference on Genetic Algorithms Mendel, Citeseer, vol 98, pp 90–95

Pelikan M, Mühlenbein H (1999) The bivariate marginal distribution algorithm. In: Advances in Soft Computing, Springer, pp 521–535

Pelikan M, Sastry K, Cantú-Paz E (2006) Scalable optimization via probabilistic modeling: From algorithms to applications, vol 33. Springer

Pelikan M, Tsutsui S, Kalapala R (2007) Dependency trees, permutations, and quadratic assignment problem. In: Genetic And Evolutionary Computation Conference: Proceedings of the 9 th annual conference on Genetic and evolutionary computation, vol 7, pp 629–629

Pellegrini P, Favaretto D, Moretti E (2007) Multiple ant colony optimization for a rich vehicle routing problem: a case study. In: Knowledge-Based Intelligent Information and Engineering Systems, Springer, pp 627–634

Pena J, Robles V, Larrañaga P, Herves V, Rosales F, Pérez MS (2004) Ga-eda: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: Innovations in Applied Artificial Intelligence, Springer, pp 361–371

Petrovska I, Carter J (2006) Estimation of distribution algorithms for history matching. In: 10th European Conference on the Mathematics of Oil Recovery

Polyakovskiy S, Bonyadi MR, Wagner M, Michalewicz Z, Neumann F (2014) A comprehensive benchmark set and heuristics for the traveling thief problem. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, pp 477–484

Ranjbar M, De Reyck B, Kianfar F (2009) A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. European Journal of Operational Research 193(1):35–48

Regnier-Coudert O, McCall J (2014) Factoradic representation for permutation optimisation. In: Parallel Problem Solving from Nature, PPSN XIII, Springer, pp 332–341

Regnier-Coudert O, McCall J, Ayodele M, Anderson S (2016) Truck and trailer scheduling in a real world, dynamic and heterogeneous context. Transportation Research Part E: Logistics and Transportation Review 93:389–408

Reinelt G (1991) Tspliba traveling salesman problem library. ORSA journal on computing 3(4):376–384

Reinelt G (1995) Tsplib95. Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg

Reinelt G (2002) Linear ordering library (lolib). University of Heidelberg, Dec

Rizzoli A, Oliverio F, Montemanni R, Gambardella LM (2004) Ant colony optimisation for vehicle routing problems: from theory to applications. Reports of Istituto Dalle Molle di Studi sull'Intelligenza Arti?ciale 9(1):1–50

Rizzoli AE, Montemanni R, Lucibello E, Gambardella LM (2007) Ant colony optimization for real-world vehicle routing problems. Swarm Intelligence 1(2):135–151

Romero T, Larrañaga P (2009) Triangulation of bayesian networks with recursive estimation of distribution algorithms. International Journal of Approximate Reasoning 50(3):472–484

Salhi A, Rodríguez JAV, Zhang Q (2007) An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, pp 570–576

Salinas-Gutiérrez R, Aguirre AH, Diharce ERV (2009) Using copulas in estimation of distribution algorithms. In: MICAI, Springer, vol 9, pp 658–668

Santana R, Larranaga P, Lozano JA (2010) Learning factorizations in estimation of distribution algorithms using affinity propagation. Evolutionary Computation 18(4):515–546

Santucci V, Baioletti M, Milani A (2016) Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. IEEE Transactions on Evolutionary Computation 20(5):682–694

Shakya S, McCall J (2007) Optimization by estimation of distribution with deum framework based on markov random fields. International Journal of Automation and Computing 4(3):262–272

Shakya S, Santana R (2008) An eda based on local markov property and gibbs sampling. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM, pp 475–476

Shakya S, Santana R (2012) Markov Networks in Evolutionary Computation. Springer

Soliman OS, Elgendi EA (2014) A hybrid estimation of distribution algorithm with random walk local search for multi-mode resource-constrained project scheduling problems. arXiv preprint arXiv:14025645

Sprecher A, Hartmann S, Drexl A (1997) An exact algorithm for project scheduling with multiple modes. Operations-Research-Spektrum 19(3):195–203

Stützle T, Hoos HH (2000) Max–min ant system. Future generation computer systems 16(8):889–914

Su W, Chow MY (2012) Performance evaluation of an eda-based large-scale plug-in hybrid electric vehicle charging algorithm. IEEE Transactions on Smart Grid 3(1):308–315

Syswerda G (1992) Simulated crossover in genetic algorithms. In: FOGA, pp 239–255

Taillard E (1993) Benchmarks for basic scheduling problems. european journal of operational research 64(2):278–285

Theodorsson-Norheim E (1987) Friedman and quade tests: Basic computer program to perform nonparametric two-way analysis of variance and multiple comparisons on ranks of several related samples. Computers in biology and medicine 17(2):85–99

Tormos P, Lova A (2003) An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 41(5):1071–1086

Tsutsui S (2002) Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In: Parallel Problem Solving from Nature, PPSN VII, Springer, pp 224–233

Tsutsui S, Pelikan M, Goldberg DE (2006) Node histogram vs. edge histogram: a comparison of pmbgas in permutation domains. MEDAL Report

Valls V, Ballestin F, Quintanilla S (2008) A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 185(2):495–508

Van Peteghem V, Vanhoucke M (2010) A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. European Journal of Operational Research 201(2):409–418

Van Peteghem V, Vanhoucke M (2011) Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. Journal of Heuristics 17(6):705–728

Van Peteghem V, Vanhoucke M (2014) An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. European Journal of Operational Research 235(1):62–72

Vanhoucke M, Coelho J (2016) An approach using SAT solvers for the RCPSP with logical constraints. European Journal of Operational Research 249(2):577–591

Voß S, Witt A (2007) Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. International journal of production economics 105(2):445–458

Wang L, Fang C (2012a) An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. Computers & Operations Research 39(2):449–460

Wang L, Fang C (2012b) A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem. Expert Systems with Applications 39(3):2451–2460

Wu HP, Huang M (2013) Improved estimation of distribution algorithm for the problem of single-machine scheduling with deteriorating jobs and different due dates. Computational and Applied Mathematics pp 1–17

Yu TL, Goldberg DE, Yassine A, Chen YP (2003) Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In: GECCO, pp 1620–1621

Yuan B, Gallagher M, Crozier S (2005) Mri magnet design: search space analysis, edas and a real-world problem with significant dependencies. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM, pp 2141–2148

Zhang Q, Sun J, Tsang E, Ford J (2004) Hybrid estimation of distribution algorithm for global optimization. Engineering computations 21(1):91–107

Zhang R, Wu C (2012) A hybrid local search algorithm for scheduling real-world job shops with batch-wise pending due dates. Engineering Applications of Artificial Intelligence 25(2):209–221

Zheng Xl, Wang L (2015) A multi-agent optimization algorithm for resource constrained project scheduling problem. Expert Systems with Applications 42(15):6039–6049

Zlochin M, Birattari M, Meuleau N, Dorigo M (2004) Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research 131(1-4):373–395