Search Based Software Project Management

Jian Ren

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

June 24, 2013

Abstract

This thesis investigates the application of Search Based Software Engineering (SBSE) approach in the field of Software Project Management (SPM). With SBSE approaches, a pool of candidate solutions to an SPM problem is automatically generated and gradually evolved to be increasingly more desirable. The thesis is motivated by the observation from industrial practice that it is much more helpful to the project manager to provide insightful knowledge than exact solutions. We investigate whether SBSE approaches can aid the project managers in decision making by not only providing them with desirable solutions, but also illustrating insightful "what-if" scenarios during the phases of project initiation, planning and enactment.

SBSE techniques can automatically "evolve" solutions to software requirement elicitation, project staffing and scheduling problems. However, the current state-of-the-art computer-aided software project management tools remain limited in several aspects. First, software requirement engineering is plagued by problems associated with unreliable estimates. The estimations made early are assumed to be accurate, but the projects are estimated and executed in an environment filled with uncertainties that may lead to delay or disruptions. Second, software project scheduling and staffing are two closely related problems that have been studied separately by most published research in the field of computer aided software project management, but software project managers are usually confronted with the complex trade-off and correlations of scheduling and staffing. Last, full attendance of required staff is usually assumed after the staff have been assigned to the project, but the execution of a project is subject to staff absences because of sickness and turnover, for example.

This thesis makes the following main contributions: (1) Introducing an automated SBSE approach to Sensitivity Analysis for requirement elicitation, which helps to achieve more accurate estimations by directing extra estimation effort towards those error-sensitive requirements and budgets. (2) Demonstrating that Co-evolutionary approaches can simultaneously co-evolve solutions for both work package sequencing and project team sizing. The proposed approach to these two interrelated problems yields better results than random and single-population evolutionary algorithms. (3) Presenting co-evolutionary approaches that can guide the project manager to anticipate and ameliorate the impact of staff absence. (4) The investigations of seven sets of real world data on software requirement and software project plans reveal general insights as well as exceptions of our approach in practise. (5) The establishment of a tool that implements the above concepts. These contributions support the thesis that automated SBSE tools can be beneficial to solution generation, and most importantly, insightful knowledge for decision making in the practise of software project management.

Acknowledgements

First and foremost, I would like to thank my supervisor *Prof. Mark Harman*. He is the best supervisor one could ever expect for.

I would also like to thank my second supervisor *Dr. Jens Krinke*, my early PhD mentor *Prof. Zheng Li*, and my previous external advisor *Prof. Anthony Finkelstein*.

I am grateful to the following external personnel for their generous help and valuable expertises on some specific topics: *Prof. Xin Yao* at University of Birmingham, *Prof. Massimiliano Di Penta* at University of Sannio, *Prof. Giuliano Antoniol* at École Polytechnique de Montréal, *Prof. Francisco Palomo Lozano and Prof. Inmaculada Medina Bulo* at University of Cádiz, *Prof. Günther Ruhe* at University of Calgary, and *Prof. Filomena Ferrucci and Dr. Federica Sarro* at University of Salerno.

My greatest gratitude is reserved for all my fellow colleagues in CREST (Centre for Research on Evolution, Search and Testing) whose advice, help and support are always available at those critical moments, especially to *Dr. William Langdon*, *Dr. David Clark*, *Dr. Afshin Mansouri*, *Dr. Shin Yoo*, and *Dr. Yuanyuan Zhang*. Finally and very importantly, I would like to thank the CREST admin, *Ms. Lena Hierl*, for the kindest and most professional administrative support.

This thesis is dedicated to my inspiring grandparents, my beloved parents, and my supportive family and friends.

List of Publications

Chapters 4 and 5 of this thesis have been published as:

- J. Ren, S. Yoo, M. Harman and J. Krinke, Search Based Data Sensitivity Analysis Applied to Requirement Engineering, *Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1681-1688. Cited by 17[1].

- J. Ren, M. Harman and M. Di Penta, Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling, *Proceedings of the 3rd International Symposium of Search Based Software Engineering (SS-BSE 2011)*, pages 127-141. Cited by 9[1].

The following papers have been published or submitted during the course of this PhD programme, although they do not form a part of this thesis:

- F. Ferrucci, M. Harman, J. Ren and F. Sarro, Not Going to Take this Anymore: Multi-Objective Overtime Planning for Software Engineering Projects, *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013)*, accepted on 20 November 2012.

- M. Harman, J. Krinke, I. M. Bulo, F. P. Lozano, J. Ren and S. Yoo, Empirical Evaluation of Exact Sensitivity Analysis for the Next Release Problem, *ACM Transactions on Software Engineering and Methodology*, under revision.

- A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren and Y. Zhang, A Search Based Approach to Fairness Analysis in Requirement Assignments to Aid Negotiation, Mediation and Decision Making, *Requirements Engineering Journal*, 14(4):231-245, December 2009. Cited by 24[1].

- J. Ren, Y. Zhang, A. Finkelstein, M. Harman and S. A. Mansouri, . “Fairness Analysis” in Requirements Assignments, *Proceedings of the 16th International Requirements Engineering Conference (RE’08)*, pages 115-124. Cited by 28[1].

[1] The citation counts are estimated by Google Scholar Citations. Accessed on May 2013.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis addresses the study of applying the Search Based Software Engineering (SBSE) approach to several software project management problems. In this chapter, introductions of the SBSE as well as the problems to be tackled are presented, the major contributions of this work are highlighted, and the layout of this thesis is presented.

1.1 Search-based Software Engineering

As a set of techniques to apply meta-heuristic algorithms to software engineering problems, Search Based Software Engineering (SBSE) [Harman and Jones, 2001] is becoming an increasingly popular paradigm for the study and implementation of solving software engineering problems that are highly complex and dynamic [Harman et al., 2012]. It works by using search-based algorithms to automatically generate solutions, and "evolving" them gradually to optimal or near optimal solutions.

Genetic Algorithms (GAs) [Holland, 1992] have been identified as one of the most widely used search-based algorithms for SBSE [Harman, 2007]. Genetic algorithms maintain one population of individual solutions to a specific problem. Each individual solution is represented as a chromosome that carries its genetic information (DNA) which defines how such an individual plans to solve the corresponding problem. Depending on how well one individual solves the underlining problem, a value is assigned to represent such individual's "fitness". Naturally, GA repeatedly chooses those "fitter" solutions to reproduce (Figure 1.1) new individuals. The new individuals then compete with their parent populations as well as their siblings. The "fitter" individuals survive, form the population in the current generation, and reproduce the next generation. When this iteration of selection and reproduction ends, the individuals in the last generation are expected to be the optimal or near optimal solutions. The sequence of these operations is given in Figure 1.2.

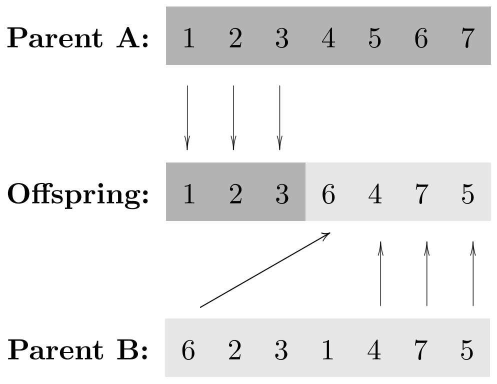


Figure 1.1: GA crossover for reproduction

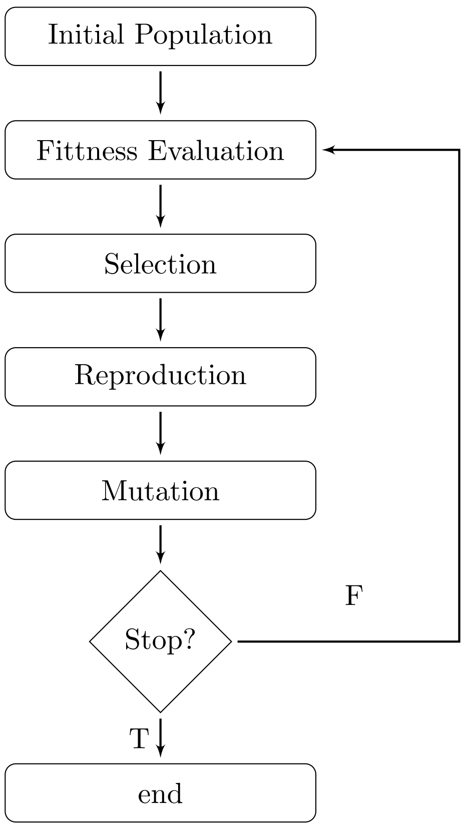


Figure 1.2: Simplified flow chart of a genetic algorithm

Co-evolutionary genetic algorithm [Hillis, 1990, Potter and Jong, 1994] is very important extension of standard GA that allows the optimisation process to "evolve" more than one population of solutions, among which they are inter-related with those in the other population. According to how the fitness of one individual might affect the fitness evaluation of other individuals, the relationship across co-evolving population can be competitive [Hillis, 1990] or cooperative [Potter and Jong, 1994].

1.2 Software Project Management

Software project scheduling and staffing problems have been tackled by heuristic algorithm by Chang et al. [Chang et al., 2001, Chang et al., 2008, Ge and Chang, 2006, Chang et al., 1998], Alba et al. [Alba and Chicano, 2005, Alba and Chicano, 2007], and many other researchers [Gueorguiev et al., 2009, Antoniol et al., 2005, Alvarez-Valdes et al., 2006, Chan et al., 1996, Hindi et al., 2002]. In fact, GA has been widely studied in tackling software project management as a project scheduling problem. However, the current state-of-the-art research demonstrate a lack of flexibility to effectively cope with the dynamic of modern software projects, such as, changing staff during the project is not allowed in the models.

From all those previous studies on software project scheduling and staffing, we have known how to optimise the schedule of a software project under ideal conditions such as accurate estimations given on: 1) accurate estimations on the effort of each individual work packages, and 2) comprehensive knowledge of resources, e.g.: provision for individual project staff.

However, in the practices of software project engineering management, the project team is assembled and staff is allocated to work packages according to a given "optimised" project schedule. This given schedule can no longer be considered to be "optimised" once the staff information become fully available, because the hired staff's skills, efficiencies and availabilities are often guaranteed to be different from the ones that were used to optimise the schedule.

Therefore, there is a need to simultaneously optimise these two closely related problems together, and we decided to investigate the co-evolutionary algorithm's advantage that enables us to study the impact of the interconnections between multiple subproblems.

In fact, unlike the work in this thesis, none of the previous work on project scheduling and staffing has used a co-evolutionary optimisation approach. More importantly, our co-evolutionary model allows us to investigate uncertainties caused by unplanned changes such as changing requirements and staff absence which can not be easily coped with by traditional standard GA.

1.3 Contributions of this Work

This thesis makes a main contribution of demonstrating that advanced SBSE techniques can be beneficial in solving software project management problems at the stages of project initiation (Chapter 4), planning (Chapter 5) and enactment (Chapter 6). The contributions of this thesis are elaborated as follows::

1. It presents an automated Sensitivity Analysis approach to identify sensitive requirements and budgets with respect to inaccurate cost estimation. The approach is based on SBSE for both single-and-multi-objective Next Release Problem formulations.

2. It introduces Cooperative Co-Evolutionary Algorithms to the software project staffing and scheduling problems for the first time. A Cooperative Co-evolutionary Algorithm is able to outperform random search and single population genetic algorithm on software project staffing and scheduling problems.

3. It presents empirical studies of four real world software projects planning data that demonstrate co-evolutionary optimisation techniques that can, not only find solutions to compensate the impact of staff absence during the project execution, but also provide various insightful knowledge that can aid project manager in making better and safer decisions.

4. It presents the establishment of a tool called Amphisbaena [1] (AMPHI- Search Based manAgEmeNt Approach). Currently, Amphisbaena provides intuitive

[1] **Amphisbaena** noun [ˌam(p)-fəs-ˈbē-nə] from: merriam-webster.com
Definition: a serpent in classical mythology having a head at each end and capable of moving in either direction
Origin: Latin, from Greek ‘amphisbaina’, from ‘amphis’ on both sides (from ‘amphi’ around) + ‘bainein’ to walk, go

visualisations for sensitivity analysis on requirement estimation, and it also provides solutions and insights of staffing and scheduling via the automated analysing process that is operated directly on the Microsoft® Project Plan (`*.mpp`) file.

In summary, the thesis proposes the following three main steps in software project management process: 1) the SBSE sensitive analysis to help on achieving more accurate estimations during requirement selection, 2) the cooperative co-evolutionary approach to help on attaining more effective project staffing and scheduling, and 3) the co-evolutionary approach to help on compensate the impact of staff absence. These contributions support the thesis that automated SBSE assistance can provide both solutions and insightful knowledge to a software project management problems across the entire software development life cycle from the project's initiation, through planing to its enactment.

1.4 Research Methodology

This research adopts the quantitative research method to systematically and empirically investigate three software project management problems. In particular, this thesis tackles the problems which cause software project managers: 1) suffering from unreliable estimations during requirement selection, 2) not being able to co-optimise staffing and scheduling when they are planning the project, and 3) being difficult to management staff absence during the enactment of a project.

In essence, the empirical experimentations are designed in answering the following three sets of research questions: 1) How does SBSE sensitivity analysis help in understanding the correlation of the key factors (i.e., cost and inaccuracy) and the revenue of the final solution, and how to understand the exceptions to the general trends? 2) Can cooperative co-evolutionary algorithm outperforms random search and conventional genetic algorithm in co-optimising the staffing and scheduling problems, and how effective and efficient? 3) How do co-evolutionary optimisation techniques reveal and compensate the impact of staff absence?

The research questions are answered by the empirical studies based on the quantitative data and statistical analysis on the result. The first set of quantitative research questions mainly ask for the significance of the correlation between two pairs of variables: {cost and impact} and {inaccuracy and impact}. Positive correlation assumptions are statistically tested against both synthetic and real-world requirement data. Spearman's rank correlation coefficient shows that strong positive correlations exist at the confidence level of 95%. The second set of quantitative research questions focus on the comparisons of the performance of random algorithm, conventional evolutionary algorithm and the cooperative co-evolutionary algorithms. The empirical studies on different algorithms are conducted on four real world software projects. The Wilcoxon Rank Sum Test on the results found that the cooperative co-evolutionary algorithm performs better than the conventional 1-population evolutionary algorithm with great statistical significance ($p \leq 1.28E - 06$). The third set of quantitative research questions mainly ask to identify the configurations of co-evolutionary algorithm that are able to find more extreme solutions than the others under the combined influence of staff absence and project complexity.

During the course of answering the research questions, a set of automated tools simulating project enactment are developed. Seven sets of real world requirement and project planning data are used to perform empirical experiments. Statistical tools are utilised to analyse the correlations between variables and the comparison of different optimisation techniques.

1.5 Layout of the Thesis

This thesis is organised as follows:

Chapter 2 - Literature Review summarises the literature in the fields of software project management, techniques for sensitivity analysis, evolutionary optimisation, search based software engineering and its applications in software project management.

Chapter 3 - Industrial Data for Evaluation describes the real world data sets used for the empirical studies in this thesis. The chapter begins by describing a set of software requirements from Motorola Inc.. The cost and revenue of each requirement are listed. The chapter then introduces each of the six sets of real world software projects' plans by: describing the industrial context, summarising the key features, and visualising the key information.

Chapter 4 - Sensitivity Analysis on Cost Estimation of Requirements Selection presents an approach to sensitivity analysis in a requirement selection problem. The approach uses search based software engineering to aid the decision maker to explore sensitivity of the cost estimates of requirements for the Next Release Problem (NRP). The chapter presents both single- and multi-objective formulation of NRP with empirical sensitivity analysis on synthetic and real-world data. Then the chapter moves on to the analysis of the empirical study in which the some intuitive assumptions are confirmed. A heat-map style visualisation tool is presented to reveal those counter-intuitive exceptions which require careful consideration.

Chapter 5 - Cooperative Co-evolutionary Job Sequencing and Team Sizing introduces an new approach to search based software project management based on Cooperative Co-evolution. The approach aims to "co-optimise" both work package scheduling and developers' team staffing problems simultaneously by applying cooperative co-evolutionary techniques to achieve early overall finish time. The chapter first introduces the models of the problems to apply the cooperative co-evolutionary approach in generating, reproducing, and eliciting desirable solutions. The solution evaluation is based on the simulation of executing such a project plan which consists of two solutions to each of the two problems. The chapter then presents the results of the empirical study using real world projects data from four different software companies. The cooperative co-evolution is demonstrated to be more efficient and effective than single population evolution and random search.

Chapter 6 - Co-evolutionary Project Planning Optimisation under Staff Absence extends the work in Chapter 5 to consider how to fully utilise advantage of co-evolutionary project planning technique to help the project manager to mitigate possible impact of staff absence. The key to analysing the impact of staff absence is first to be able to distinguish the staff by their skill, and then to simulate the absence at different stages of a project. The chapter extends the design of the problem model of staffing, as introduced in Chapter 5, to allow the representation of staff's absence in a staff availability calendar. The scheduler simulating the execution of project plan is completely redesigned to accommodate the new job assignment rules that are associated with the skills. This chapter then presents four new configurations of co-evolutionary optimisation techniques and their empirical studies on four real world software project data. The result demonstrates the co-evolutionary software project planning technique is able to provide lower and upper bound of current project's finish time, identify the dominating problem during the execution of current project, and provide useful insights on the correlations among staff absence rate, the delay caused, and the complexity of a project.

Chapter 7 - Conclusions concludes the thesis with a summary of its major contributions and proposals of future work.

Chapter 2

Literature Review

2.1 Software Project Management

Project management is a broad subject, and all of its subtopics cannot be covered in this thesis. Therefore, the thesis is focused on Software Project Management, including cost and scheduling estimation, risk management, and staff assignment optimisation.

In general, a project can be defined as a series of activities that are conducted to achieve one or more specific objectives at a specified cost and within a specified time [Hughes et al., 2004]. Essentially, a management method is a set of processes used to run a project in a controlled and, therefore, predictable fashion. In the context of software engineering, we focus on projects that develop new software, and the management activities including: planning, coordinating, measuring, monitoring, controlling, and reporting, which collectively ensure that the development and maintenance of the software is systematic, disciplined, and quantified [Abran et al., 2004,IEEE610.12-1990, 1990].

The Software Engineering Coordinating Committee, which is sponsored by the IEEE Computer Society, has developed an all–inclusive collection of knowledge within the profession of software engineering that is known as the Software Engineering Body of Knowledge (SWEBOK) [Abran et al., 2004]. SWEBOK suggests the 10 Knowledge Areas (KAs) that form the classification of the scheme of the field, i.e., Software Requirements, Software Design, Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, Software Engineering Tools and Methods, and Software Quality.

With regard to the software project management, SWEBOK specifically includes a breakdown that allows the Software Engineering Management KA to be viewed as an organisational process. As shown in Figure 2.1, the primary basis for the top-level breakdown is the process of managing a software engineering project. The software project management process is addressed in its first five subareas, and software engineering measurement is addressed in the last sub–area.

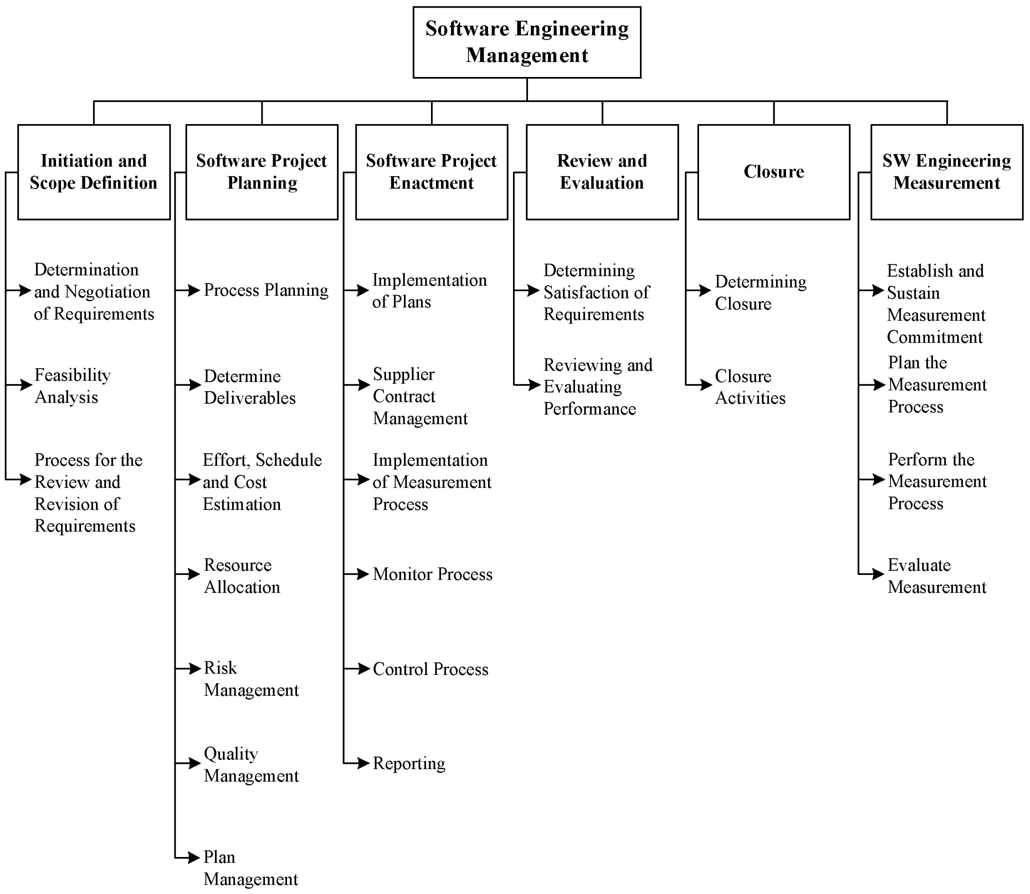

Software Engineering Management
Initiation and Scope Definition
Determination and Negotiation of Requirements
Feasibility Analysis
Process for the Review and Revision of Requirements
Software Project Planning
Process Planning
Determine Deliverables
Effort, Schedule and Cost Estimation
Resource Allocation
Risk Management
Quality Management
Plan Management
Software Project Enactment
Implementation of Plans
Supplier Contract Management
Implementation of Measurement Process
Monitor Process
Control Process
Reporting
Review and Evaluation
Determining Satisfaction of Requirements
Reviewing and Evaluating Performance
Closure
Determining Closure
Closure Activities
SW Engineering Measurement
Establish and Sustain Measurement Commitment
Plan the Measurement Process
Perform the Measurement Process
Evaluate Measurement

Figure 2.1: Breakdown of topics covered in the Software Engineering Management KA. Figure adapted from [Abran et al., 2004].

2.1.1 Software Life Cycle Process Models

Waterfall Model: The Waterfall model is the oldest and most well known software development model. Published by Royce in 1970 [Royce, 1970], it models the software development process in sequential phases and suggests that the current phase should be completed and checked for accuracy before proceeding to the next phase. By using the Waterfall model, the software project manager expects each task to be completed properly the first time it is done. However, for most software projects, the developers' knowledge and understanding of the information related to the project become clearer as the processing proceeds. Therefore, if some important details are discovered that were unknown at the beginning of the project, the Waterfall developing model requires the process to be restarted. So, the Waterfall model works best for projects for which the required information is known and static, the objectives of project are clearly defined, and there is a very low probability of any surprise.

Spiral Model: The Spiral model, which was developed by Boehm in 1988 [Boehm, 1988], was designed to overcome the Waterfall model's major weaknesses. In the Spiral model, a project starts with the development of a small set of requirements to guide the developers' effects through the whole process. Then, in each of the following iterations, the development team add additional requirements to the product based on the experience gained from the previous iterations and any new, available external knowledge that may be available about the product. This iterative approach results in a more flexible development process to that is able to adapt to changing requirements. It also reduces the risks by providing opportunities for the objectives to be refined and for the risks to be reassessed at the end of each iteration.

V–Model: The V–model was first introduced for use in Germany's federal IT projects [Sommerville, 1992]. It pays special attention to improving the communication between the developer and the customer by associating the analysis and development phases with the corresponding testing processes. This approach allows the provisions of guidelines so that both developers and customers can contribute cooperatively to the project.

Agile Methods: New software is integrated seamlessly into people's every day lives, and software development is no longer viewed as a technically demanding activity that only serious scientists can do. Quite often, the software engineers begin coding when only a small fraction of the requirements are clearly defined and well before the overall design structure of the software has been finalised. Since even the customers might not have a clear idea of what all their needs are at the beginning of the software development process, the early finalisation of the overall structure of software that is being developed is very difficult. Most importantly, the requirements often change during the course of the development process. Therefore, it is unrealistic to enforce an exact "plan" for the software when its development has just begun.

In a traditional "Plan Driven" development process, an attempt is made to plan all the requirements and changes upfront. At the beginning of the process, efforts are made to anticipate any changes in the requirements that may be needed, and, subsequently, the goal of the following management activities is to try to ensure that the project is developed according to plan, hoping that nothing goes wrong.

Agile "spirit" guides the project management to attempt to achieve "working software" in very short periods of an incremental development cycle. The "plan" is to develop the software along with its overall structure until the customer is satisfied or the resources are used up. The goal is the development of functional software that satisfies the customer, and Agile methods attempt to achieve this by emphasising the role of the day-to-day input of customers in keeping the process moving in the right direction. In general, the software system and the requirements are developed gradually as the development activities take place.

In February 2001, the Agile Manifesto [Beck et al., 2001] was published and the Agile Alliance was found to promote Agile methods including SCRUM and Extreme Programming. SCRUM [Schwaber and Beedle, 2001] is one of the earliest Agile software project management methods while Extreme Programming [Beck and Andres, 2004] is one of the most widely used software development methods [Boehm, 2006].

2.1.2 Challenges in Software Project Management

The Software Engineering Management KA [Abran et al., 2004] addresses the management of software engineering project and the measurement of software. It also identifies some of the key aspects that are specific to a software project and that complicate the effective management of such projects. Harman *et al.* [Harman et al., 2009b] identified a number of unresolved challenges in software project planning, including: lack of robustness in planning, poor estimates and lack of appropriate integration of the various processes. These problems that are specific to software project management are summarised below in items 1 through 6 and the remaining unresolved challenges are summarised in item 7 through 9:

1. Lack of appreciation for the complexity inherent in Software Engineering, particularly in relation to the impact of changing requirements.

2. Changing requirements that might be generated by the clients (customers) or by the software engineering processes themselves.

3. Iterative Developing Process: software is often built in an iterative process rather than a sequence of closed tasks.

4. Software engineering necessarily incorporates aspects of creativity and discipline, and maintaining an appropriate balance between the two is often difficult.

5. The degree of novelty and complexity of software is often extremely high.

6. There is a rapid rate of change in the underlying technology.

7. Often, the importance of robustness in planning is overlooked; it may be more important to develop plans that are robust and can accommodate changes than to develop plans that lead to early completion.

8. The unreliability of price and schedule estimates is problem that constantly plagues software project development activities— [Jø rgensen and Shepperd, 2007].

9. Appropriate integration of the process steps is often overlooked: software project management is not an activity that can be optimised in isolation. It is necessary to develop techniques to integrate management activities with other activities, such as design, testing, maintenance, or even with other engineering, such as requirement engineering.

Herroelen [Herroelen, 2005] provided a list of the 12 reasons that are often cited for the escalation of project costs and schedules. In examining that list, it is apparent that none of them is related to the technology used in the project; rather, they are more related to human factors in the project management process. Keil [Keil et al., 2003] gathered data for an information system based on a survey of 376 information system audit and control professionals in the U.S. Bryde [Bryde, 2003] conducted an empirical study of project management practices in projects in the UK. The tools that are used in project management are project tracking, time analysis, cost analysis, and resource analysis. Herroelen suggested that "proper use of project management software may well not be considered the most important driving force behind project success."

2.1.3 Industry Standards and Practices for PM

Major project management practices in the real–world are listed below:

Critical Path Method (CPM) [Kelley Jr and Walker, 1959]: Developed by DuPont Corporation, CPM is a scheduling algorithm to analyse the ordering of work packages.

Program Evaluation Review Technique (PERT) [Malcolm et al., 1959]: Invented by US Department of Defense for a US Navy Project, PERT is a method to calculate the total completion time of a project by analysing the completion time of each task involved in the project.

Work Breakdown Structure (WBS) [Jø rgensen, 2004b, Tausworthe, 1980]: Also invented by US Department of Defense, WBS is a hierarchical tree structure that contains all the subtasks that need to be done to complete the whole project.

SCRUM [Schwaber and Beedle, 2001]: Scrum is an agile software development model based on multiple small teams working in an intensive and interdependent manner.

A Guide to the Project Management Body of Knowledge (PMBOK Guide) [PMI, 2004]: Published by Project Management Institute, PMBOK is an standard of accepted project management information and practices. The latest (4th) edition was released in 2008.

Earned Value Management (EVM) : EVM is a set of techniques for measuring the project progress.

Total Cost Management (TCM) : TCM is a process for applying the skills and knowledge of cost engineering.

PRojects IN Controlled Environments (PRINCE, PRINCE2) : It provides a method for managing projects within a clearly defined framework.

Theory of Constraints (TOC), Critical Chain Project Management (CCPM) : CCPM is developed from TOC. It aims to analyse constraints on the project and manage the resources to keep the whole project on schedule.

2.1.4 Risk Management in Software Development

Boehm [Boehm, 1991, Boehm and Ross, 1989] introduced the concept of Software Risk Management into the field of Software Engineering in the 1980s. He summarised four types of sources of software risk addressed by risk management techniques:

- Potential software errors
- Overruns of budget and schedule
- Not satisfying functionality or performance requirements
- Developing a product which is hard to modify or use in other situations

Boehm offered a six–step risk management process to assess and control these sources of risk. The six steps are grouped into two categories called assessment and control as follows:

- Risk Assessment
  1. Risk Identification
  2. Risk Analysis
  3. Risk Prioritisation
- Risk Control
  4. Risk Management Planning
  5. Risk Resolution
  6. Risk Monitoring

With regards to the risk caused by software errors, an example of the Risk Reduction Leverage calculation is given in Boehm's paper [Boehm and Ross, 1989] to confirm that the investments in risk management with verification and validation (V&V) in the early stages of a software project have a higher pay-off ratio than investments that are made in testing later on. For the sake of easy identification, the most common risk items on a software project, i.e., a checklist of the "Top 10 Primary Sources of Risk on Software Projects," are summarised based on their survey of a number of experienced project managers. He identified the difficulty of making accurate estimations of the possibility of the occurrence of a risk and the associated loss identified by the checklist. This leads to an inaccurate risk assessment and, consequently, to poor risk control. Boehm mentioned some techniques to improve the estimation of the probability of the occurrence of a risk. For simplicity, the risk probabilities and losses are assessed on a relative scale of 0 to 10.

Fairley [Fairley, 1994] provided a design of the process that was more practical than Boehm's. He created a seven-step process for risk management was based on several years of work with numerous organisations to identify and overcome risk factors in software projects:

1. Identify risk factors
2. Assess risk probabilities and effects on the project
3. Develop strategies to mitigate identified risks
4. Monitor risk factors
5. Invoke a contingency plan
6. Manage the crisis
7. Recover from the crisis

An example of risk management for a project to implement a telecommunications protocol for a network gateway was proposed to demonstrate his mathematical model of the assessment of risk probabilities and effects. However, unlike Boehm, he suggested no additional tools or methods.

In the practice of Software Project Risk Management, different project managers tend to focus on different aspects of risk analysis. Moynihan [Moynihan, 1997] confirmed this even with his survey of only 14 experienced application systems developers in Ireland on the topic of "How experienced project managers assess risk." More importantly, he analysed the risk factors summarised from the survey and compared them with those risk factors identified in the software project management literature, such as Barki's Risk Variable [Barki et al., 1993] and SEI Risk Question [Carr et al., 1993]. The analysis suggested that it is unrealistic to build a single, universal risk taxonomy for use by all software development projects, which means a universal "checklist" for software project risk management does not exist. Project risk management has different taxonomies within different project contexts, just as we might expect.

Before a meaningful risk management plan can be developed, risks must be identified in advance. Despite the different aspects of the risk management process, both Boehm and Fairley started their risk management process by Risk Factors Identification. However, since a universal "checklist" does not exist and experts' risk management is highly dependent on their experience, researchers tend to seek out and use the tools or models that will help the experts analyse risks that have specific characteristics to determine their impacts on the project.

Keil et al. [Keil et al., 1998] proposed a risk categorisation framework for identifying software project risk based on two dimensions, i.e., perceived level of control, and perceived relative importance of the risk. They assembled three panels of more than 40 software project managers from all over the world to rank the importance of a common set of 11 risk factors. The project managers had little or no input concerning the risk factors that were considered to be the most important. Subsequently, the study results suggested that the software project manager should expand her or his risk assessment to include the factors over which project managers have relatively little control, such as risks relative to customer mandates, which were identified among the most important risks but which were missing entirely from Boehm's top 10 checklist of possible risk sources [Boehm and Ross, 1989].

With the help of their risk framework, Wallace and Keil subsequently investigated [Wallace and Keil, 2004] how different types of risk influence project outcomes by collecting and analysing input from more than 500 software project managers. The study showed that managing the risks related to a project's execution, scope, and requirements is critical. Project management almost always requires tradeoffs to deal with the triple constraints of scope, cost, and schedule.

A recent empirical study by Odzaly et al. [Odzaly and Des Greer, 2009] indicated that a sample of 18 experienced software project managers had good awareness of risk management but a low usage of the tools required to perform risk management on projects. The authors also sought to develop an agent-based, automatic tool to bypass the perception that the risk management process is costly, which has been confirmed as a major barrier that prevents or reduces its application in software project management practice. Gueorguiev et al. [Gueorguiev et al., 2009] introduced a search-based approach to identify and quantitatively analyse the risk associated with project scheduling.

To summarise, many research and standardisation efforts have been attempted in the field of risk analysis for software project management, but, in practice, practitioners still tend to avoid using the recommended approaches based on the facts that 1) a set of "standard" risk management processes is generally unacceptable to software project managers and 2) it is costly to do so.

Automated or semi–automated tools have become important in advancing the practice of risk management to a new level. As indicated in Section 2.5 of this review, Search Based Software Engineering (SBSE) techniques are the ideal tool to automate the risk analysis process in software project management. Within the context of SBSE, project managers have the choice of defining the objectives that they think are more important for successful completion of the current project or using the SBSE tool to determine the optimal or near–optimal candidate solutions.

2.2 Software Cost Estimation

Software cost estimation is critical for the success of software projects management. Depending on whether the final step of the estimation is a mechanical quantification step, such as a formula, there are two ways to estimate costs during the software project planning phase, as indicated in Expert Judgment–Based Estimations and Formal Model–Based Methods [Jø rgensen et al., 2009].

According to an extensive review of software estimation models and techniques conducted by Boehm in 2000 [Boehm et al., 2000], these methods can be classified into six major categories: 1) *parametric models* that use specific mathematical models to measure and calculate estimates for the development efforts, such as COCOMO and COCOMO II [Boehm, 1984, Boehm et al., 1995], SLIM [Putnam and Myers, 1992], and Check Point [Jones, 1997]; 2) *expertise–based techniques* that help practitioners provide estimates based on their knowledge and experiences, such as the Delphi method [Boehm, 1984,Helmer et al., 1966] and Work Breakdown Structure-based methods [Jø rgensen, 2004b,Tausworthe, 1980]; 3) *learning–oriented techniques* that perform the estimation by "learning" from previous experience. The "learning" process can be done manually, such as the Case Based Reasoning (CBR) approach [Aamodt and Plaza, 1994], or automatically, such as the Machine Learning approach [Goldberg, 1989]; 4) *dynamics–based models* developed by Jay Forrester [Forrester and Wright, 1961] that acknowledge the change of software project efforts or cost over the duration of the development of the system; 5) *regression–based models* that include the "Standard" regression – Ordinary Least Squares (OLS) method [Griffiths et al., 1993] and the "Robust" regression (e.g., Least–squares of Inverted balanced Relative errors (LIRS)) [Miyazaki et al., 1994], which is an improved version of OLS that alleviates the common problem of outliers in observed software engineering data and 6) *composite methods* that combine two or more techniques to accommodate the needs of different project situations.

Researchers have a strong inclination to combining their expert judgment with the results of formal methods for developing estimates of software development efforts. In a recent debate [Jø rgensen et al., 2009] with Boehm on "which is better for software development estimation: formal models or expert judgement?" Jorgensen claimed that the most important advantage of the judgment-based method is that the experts' highly specific knowledge is difficult to include in the formal models. Boehm argued that parametric models contain a significant amount of information on which factors cause software costs to change and by how much. He also argued that organisations are performing extensive sensitivity, risk, and trade–off analyses to narrow the cone of uncertainty, but these analyses are complex and time consuming.

On the other hand, industrial surveys have indicated that formal models-based methods are seldom used by software practitioners, mainly because of the high cost of implementing formal models and the insignificant benefits that result from their use [Yang et al., 2008a]. Due to the people–centric nature of software engineering, an accurate cost estimation system has yet to be developed and remains as one of the unachieved challenges in Software Engineering [Shepperd, 2007].

In addition to the nature of the "high cost" and inaccuracies of estimation techniques, project uncertainties are also of considerable importance [Jø rgensen, 2004a]. Large differences between estimated and actual effort do not necessarily indicate poor estimation skills. While it is important to analyse the degree of the uncertainties of the estimate, it might be more helpful for the decision maker to gain insight into the effects caused by the uncertainties during the cost estimation phase [Gueorguiev et al., 2009,Harman et al., 2009a].

2.3 Sensitivity Analysis

As discussed in Sections 2.1.4 and 2.2 concerning risk analysis and cost estimation, it was indicated that, due to variables that are unpredictable, software engineers are not able to predict accurately actual risks or cost. While various models and procedures have been developed in attempts to achieve more accurate predictions of the parameters, some researchers have been attempting to prevent failures in projects with a slightly different approach; they have tried to identify the parameters that are more sensitive to inaccurate estimations.

Sensitivity Analysis (SA) refers to "the determination of the contribution of individual, uncertain analysis inputs to the uncertainty in analysis results" [Helton et al., 2006]. Usually, when a system becomes increasingly complex, it becomes increasingly difficult or even impossible to derive and summarise the relationship between the inputs and outputs of the system using a mathematical model. Sensitivity analysis is a set of techniques used to analyse the relationship between the inputs and outputs of a complex system based on the input parameters and the observed outputs produced by the system. Generally, sensitivity analysis is used to identify the input parameters that most significantly influence the system's outputs.

Applications of SA are widely found in the literature for various areas, such as chemical kinetics [Sandu et al., 2003], physical science [Newman et al., 1999], environmental modelling [Hamby, 1994], telecommunications engineering [Racu et al., 2005], and financial analysis [Levine and Renelt, 1992]. A few typical usages of SA techniques have been selected and classified into the following categories for experimentations of evolutionary computation:

One–At–a–Time (OAT) methods [Saltelli et al., 2000] are the simplest of the various methods, and they are also referred to as Local Sensitivity Analysis [Saltelli et al., 2008] or Nominal Range Sensitivity Analysis [Christopher Frey and Patil, 2002]. Conceptually, these methods vary one parameter at a time repeatedly, while all of the other parameters are maintained at their fixed, baseline values. The major drawback of OAT methods is that the interactions between parameters are not taken into account in the analysis. Therefore, results from OAT methods and global sensitivity analyses can be contradictory [Thogmartin, 2010]. Several possible ways for correcting OAT methods have been proposed recently by Saltelli et al., including full factorial design methods, the regression effects method, and the elementary effects method, which they note can be used "...at no extra cost." [Saltelli and Annoni, 2010].

Factorial Design (FD) [Box et al., 1978] overcomes the major drawback of the OAT methods by selecting a given number of samples for each parameter and running the model for all combinations of the samples. In this case, the interactions between parameters are considered. However, it is not feasible to use this method to perform SA on a model with a large number of parameters because of the tremendous number of runs required. In 2006, Helton et al. [Helton et al., 2006] conducted a comprehensive survey of uncertainty and sensitivity analysis techniques using sampling-based (e.g., Monte Carlo (MC)) methods.

The Elementary Effects (EE) [Morris, 1991] Method was first introduced by Morris in 1991 to accommodate the needs for SA from models that are deterministic and complicated with a large number of input parameters, which makes "classical" SA methods impractical. The elementary effect of the ith input x_i is defined below for a given set of k inputs whose values are denoted by vector X :

$$d_i(X) = [y(x_1, x_2, \ldots, x_i + \Delta, \ldots, x_k) - y(X)]/\Delta \tag{2.1}$$

where y is the output of the model, and Δ is the simulated "error" on x_i. (Note: X here is not the basic line.) Suppose r sample values for x_i are selected; for each sample value, we have an elementary effect value for this particular input. Then the mean and standard deviation are calculated for the elementary effects for this input as follows:

$$\mu_i = \frac{1}{r}\sum_{1}^{r} d_i(X) \tag{2.2}$$

$$\sigma_i = \sqrt{\frac{1}{r-1}\sum_{1}^{r}\{d_i(X) - \mu_i\}^2} \tag{2.3}$$

A large measure of mean value, μ, indicates x_i has a large "overall" influence, while a high value of standard deviation, σ, indicates non-linear effects or that x_i interacts with other inputs. Obviously, low values of both variables indicates that input is non-influential. EE methods are considered to be a very good compromise between accuracy and efficiency, especially for sensitivity analysis of large models [Campolongo and Braddock, 1999]. Aiming at improving Morris' strategy on randomly sampling the input space, Campolongo et al. recently enhanced the strategy to gain a better spread of the input domain without increasing the number of model executions needed [Campolongo et al., 2007].

Box and Wilson (1951) first developed the Response Surface Methodology (RSM) for determining optimum conditions in a chemical investigation [Box and Wilson, 1951]. Since that time, RSM has been of value for many other fields [Bucher, 1990, Carley et al., 2004, Hill and Hunter, 1966, Isukapalli et al., 2000, Khuri and Mukhopadhyay, 2010]. The underlying idea of RSM is to try to reduce the number of computational experiments necessary to explore the input/output relationship space. It provides a way to develop and/or simplify the model itself by rigorously choosing a few points on the response surface to effectively represent all the possible points. Then, the sensitivity analysis can be determined by investigation of the response surface, i.e., inspection of the functional form of the response surface.

In summary, OAT methods and FD can be considered as "brute force" strategies for tuning the input parameters to determine the features of the output results, while the EE method attempts to sample the parameters with care. Furthermore, other methods, such as RSM, simplify the model and subsequently prioritise the input parameters. Practitioners are more likely to use the combinations of these methods mentioned above [Saltelli, 2004,Saltelli, 2005,Saltelli et al., 2004,Saltelli et al., 2008].

It is worth mentioning that Uncertainty Analysis (UA) and Sensitivity Analysis are often coupled in practice [Saltelli and Annoni, 2010]. However, the objectives of UA and SA are different. Uncertainty analysis refers to "the determination of the uncertainty in analysis results that derives from uncertainty in analysis inputs" [Helton et al., 2006] and answers the question, "How uncertain is this inference?" On the other hand, sensitivity analysis aims to answer the question, "Where is this uncertainty coming from?" [Saltelli and Annoni, 2010].

2.4 Evolutionary Computation

The inspiration for using Evolutionary Computation (EC) to solve engineering or science problems came from the model of Darwinian evolution and biological theory [Darwin, 1859]. EC uses the Darwinian concepts of evolution and the principles of natural selection to automate the process of solving problems. Possible solutions for an underlying problem are considered as "individuals". The evaluation of each individual is calculated by the fitness function. Higher fitness values indicate better quality solutions for the given problem. Each individual is encoded with the "chromosomes" on which the genetic operations (i.e., crossover and mutation) are conducted. Finally, the candidate solutions with higher fitness values survive during the process of "selection" and produce their "offspring" by means of crossover and mutation. The new population, which consists of the candidate solutions that survived and their offspring, is called a new generation. As such, under the force of evolutionary pressure, the whole population evolves better solutions gradually over generations. A brief introduction to each step of this evolutionary process is provided below.

Representation: The representation of solutions reflects how the designer understands the problem domain and the solution domain. There are two representations that are most frequently used, i.e., individuals could represent the solution for a real world optimisation problem as a binary string (GA) or individuals could represent the solution in the form of a vector of real numbers (ES). In both cases, the evaluation process is based on extracting information from these representations, which will be given to an objective function to calculate the fitness.

Initialisation: Random is most frequently used for initialisation. Because the first generation of individuals is generated randomly, all the individuals of the first generation are distributed randomly in the search space, providing diverse information about the search space. It is the easiest way to initially randomise the population, because the knowledge of search space is not necessary in the procedure of random initialisation.

Evaluation: The evaluation process is to determine the fitness value of the individuals. The objective function(s) used to calculate the fitness value represent(s) the ultimate goal of the optimisation, which should be clearly defined by the designer. The algorithm should conduct the evaluation methods, which could then be used to direct the algorithm to identify the "fittest" solution(s), subject to constraints of resources.

Selection: The selection procedure is based mainly (but not necessarily fully) on the fitness value of each individual. The individuals with higher fitness values will be more likely to survive and produce offspring. On the other hand, the individuals with less fitness values could also be selected when they have, for instance, high potentials to contribute to the diversity of the population and produce high-fitness-value offspring. The ultimate goal of selection is to maintain a population of high fitness individuals and provide an optimal set of solutions.

Genetic Operators: Mutation and Crossover (Recombination) are the two methods used to generate offspring from the current population with the hope of producing better individuals in the offspring. The process of mutation is to alter a small portion of an individual's chromosome, hoping to introduce more fit genes. The process of crossover (recombination) involves mixing the genes from two selected individuals (chromosomes), hoping to combine the best genes to produce stronger offspring.

Termination: The process of evolution could be endless iterations of "survival of the fittest". The designer must decide when to terminate the iterations and achieve the final survivals. This decision could be made based on the evaluation of the population (how good the population is) or based on the constraint of the resources that run the algorithms, i.e., the amounts of time and money that can be spent on it.

2.4.1 Evolutionary Algorithm

The Evolutionary Algorithm (EA) is a sub-set of EC. There are two prominent features that distinguish EAs from other search algorithms: 1) EAs are all population-based and 2) there are communications and information exchange among individuals or between populations [Yao, 1996]. Due to the different configuration in the implementations, four major EAs have been developed under the same concepts of evolution that were aforementioned.

2.4.1.1 Genetic Algorithm

The Genetic Algorithm (GA) refers to a model introduced and investigated by John Holland in 1975 [Holland, 1992]. GA is one of the most popular types of EAs. The representation of the individuals in a GA is normally in the form of strings of numbers. It has been used mostly to evolve solutions in the parameterised problem domain [Goldberg, 1989, Kicinger et al., 2005].

2.4.1.2 Genetic Programming

Genetic Programming (GP) [Koza and Poli, 2005, Koza, 1992] has a tree structure to represent individual candidate solutions, while it is attempting to evolve actual computer programs. With the tree-structure chromosome representation, a complex mathematical model or a program can be expressed well, and, then, the genetic operations can perform the evolution process accordingly. GP has been used to evolve actual computer programs for solving a number of computational tasks [Langdon and Poli, 2002].

2.4.1.3 Evolutionary Strategy

The Evolutionary Strategy (ES) approach was first introduced by Rechenberg [Rechenberg, 1964] and Schwefel [Schwefel, 1965]. One genetic operator, Crossover (recombination), is less commonly used in ES. On the other hand, mutation operator on each individual is guided by parameters that evolves along with individuals themselves. In such cases, the genetic operation process is able to adapt to the given problem automatically.

2.4.1.4 Evolutionary Programming

Evolutionary Programming (EP) was pioneered by L. Fogel, A. Owens, and M. Walsh in 1966 [Fogel et al., 1966] and recently refined by D. Fogel [Fogel, 1991]. There is no fixed representation or structure to distinguish EP from the other EAs. Therefore, researchers found it hard to distinguish EP from the other EAs on the theoretical level [Weise, 2009]. Detailed elaborations of the mechanism and comparisons of ES, GA, and EP can be found in the literature [Bäck and Schwefel, 1993].

2.4.1.5 Co–evolutionary Genetic Algorithm

Co-evolutionary computation research is another very important extension of standard GA. Axelord was one of the first to introduce ideas of modelling the behaviour of natural co-evolution in game theory [Axelrod and Dion, 1988, Axelrod and Hamilton, 1981]. Hills [Hillis, 1990], Husbands and Mill [Husbands and Mill, 1991], and Paredis [Paredis, 1994] were the earliest researchers to implement Co-evolutionary GA (CGA). Hills used two populations to evolve the sorting networks and test cases simultaneously. Husbands and Mill applied two populations to evolve two fractions of solutions independently for a job-shop scheduling problem. Paredis evolved the constraints and solutions in two different populations for an “n-queen” problem.

In the context of evolutionary computation, co-evolution refers to evolving more than one independent population. The feature that distinguishes the co-evolutionary GA from the standard GA is that the evaluation of individuals is based on the interactions with other individuals instead of being conducted alone. Depending on the nature of these interactions, CGAs are further classified into two types, i.e., Competitive CGA and Cooperative CGA.

2.4.2 Competitive Co–evolutionary Algorithm

Competitive CGA maintains different candidate solutions in multiple populations that simulate competing relationships, such as predator-pray. For example, in Hills’ early implementation [Hillis, 1990], the objective of one population is to evolve increasingly better sorting networks, while the objective of the other population is to evolve increasingly difficult test cases for the sorting network. In such a case, increasingly better solutions for both sorting networks and test cases are found under the pressure of selection. Most work with co-evolutionary algorithms are this type of so-called Competitive CGAs [Wiegand, 2003].

2.4.3 Cooperative Co-evolutionary Algorithm

Cooperative co-evolution [Potter, 1997,Potter and Jong, 1994] was proposed to solve large and complex problems by implementing the divide–and–conquer strategy. In the model of cooperative CGAs, the individuals from each population represent a fraction of the solution to the given problem. Candidate solutions in each population are evolved independently. The interactions only occur to obtain fitness. It was originally designed to decompose a high-dimensional problem into smaller sub-problems that could be handled by conventional evolutionary algorithms [Wiegand, 2003,Yang et al., 2008b].

2.5 Search Based Software Engineering

Meta-heuristic search techniques, such as Genetic Algorithm (GA), Simulated Annealing (SA), and Hill Climbing (HC), have proven to be good at providing "good-enough" solutions for complex problems within a reasonable time. On the other hand, the nature of software engineering makes it the ideal application for search-based optimisation [Harman, 2010b]. For these reasons, the application of search-based techniques to software engineering problems has become a burgeoning interest to many researchers in recent years.

2.5.1 Origins and Applications

One of the earliest publications concerning the use of search-based techniques to solve software engineering problems was in 1976 by Miller and Spooner [Miller and Spooner, 1976]. In 2001, Harman and Jones [Harman and Jones, 2001] coined the term "Search-Based Software Engineering" (SBSE). Since then, research on SBSE has been developed widely in various areas for solving software engineering problems. Recently, in 2009, Harman et al. also conducted an extensive survey [Harman et al., 2009b] on the application of search based techniques to problems throughout the software engineering lifecycle, such as requirement selection problems [Finkelstein et al., 2008,Finkelstein et al., 2009,Harman et al., 2009a,Zhang et al., 2007], cost estimation problems [Harman et al., 2009a], software project scheduling problems [Alba and Chicano, 2007, Di Penta et al., 2011, Di Penta et al., 2007, Herroelen, 2005], and testing [Harman and McMinn, 2010, McMinn, 2005]. It is interesting to see that, even though search-based techniques have existed for decades, research on SBSE did not gain significant attention until the last decade. The number of publications on SBSE has increased every year since 2000. The number of publications on SBSE in 2008 was about 10 times greater than it was in 2000 [Harman et al., 2009b].

The survey by Harman et al. [Harman et al., 2009b] indicated that 54% of the overall SBSE literature is concerned with applications related to software testing. Several important surveys specific to search based testing can be found in the literature [Afzal et al., 2009, Ali et al., 2010, McMinn, 2004]. On the other hand, GA, SA, and HC have been identified as the most widely used search-based algorithms for SBSE.

2.5.2 Software Project Management with Meta-heuristics

Software projects always demand a large amount of management effort for planning, scheduling, risk estimation, and monitoring. The project manager usually conducts these management activities in order to achieve specific objectives and satisfy various constraints. The highly-complex nature of these management activities justifies the need for computer-aided tools for finding the best solutions. Such tools are used for activities such as requirement selection, project scheduling, resource allocation, and cost estimation, which are difficult tasks because there are too many potential solutions to be searched exhaustively without computer assistance. On the other hand, the decisions made by project managers are unlikely to be guaranteed to be the best decisions, because the project managers have different backgrounds, training, and experience.

Existing tools, such as the Project Evaluation and Review Technique, the Critical Path Method, Gantt diagrams, and Earned Value Analysis, can be used in planning and tracking project milestones. However, the scheduling problem with recourse constraints is indeed strongly NP-hard problem, which implies that the computation time for the exact algorithms is excessive even for moderately-sized applications.

The application of search-based techniques for solving software project scheduling and staffing problems has been developed over the past decade or so [Hart et al., 2005, Kolisch and Hartmann, 2006], although the studies of these techniques were not entirely focused on software engineering. Search-based software project planning (SBSPP) may benefit from the results that are provided by the more general literature on project scheduling. Antoniol et al. [Antoniol et al., 2004] proposed an approach to help assess staffing needs based on queuing theory and stochastic simulation. Di Penta et al.. [Di Penta et al., 2011, Di Penta et al., 2007] used a search–based project staffing and scheduling approach to address the scheduling problems in the context of software projects. None of the previous works on project scheduling and staff allocation has used a cooperative co-evolutionary optimisation approach. However, the cooperative CGA additionally allows the researcher to investigate the impact of the interconnections between multiple sub-problems on the overall optimisation goal, such as the project completion time, as we are attempting to do in this thesis. More work is required to integrate other aspects of the software development process into an optimised software project management activity.

2.5.2.1 Project Scheduling Problem with Meta-heuristics

Project management techniques like Project Evaluation and Review (PERT) [Malcolm et al., 1959], and Critical Path Method (CPM) [Kelley Jr and Walker, 1959] were developed in the 1950s. These tools allow projects to be planned and analysed by networks in which major project events, denoted by nodes, are connected by project activities that denoted by arcs, so that the interrelationship between project events and activities can be visualised and then further analysed in either a probabilistic way (PERT), or deterministic way (CPM). However, these diagram–based techniques aim at analysing the time flow of the project, but they do not take into account the cost and resource constraints. This is the Resource–Constrained Project Scheduling Problem (RCPSP). The RCPSP is known as an NP–hard problem [Blazewicz et al., 1983]. The objective of RCPSP is to minimise the overall duration of the project while subject to various constraints such as resource constraints and interdependencies constraints. Various forms of RCPSPs have been intensively studied for decades. A comprehensive survey on parallel machine scheduling problems can be found here [Mokotoff, 2001].

Software project management problems are generalised as a project scheduling problem and tackled by heuristic algorithm by Chang et al. [Chang et al., 2001, Chang et al., 2008, Ge and Chang, 2006, Chang et al., 1998], Alba et al. [Alba and Chicano, 2005, Alba and Chicano, 2007], and many others researchers [Gueorguiev et al., 2009, Antoniol et al., 2005, Alvarez-Valdes et al., 2006]. In 1996, Chan et al. proposed using genetic algorithms to solve RCPSP [Chan et al., 1996] by treating the problem as one of determining the optimal ordering of scheduling activities through selection pressure and recombination. They noticed the interaction between resource availability profiles and the quality of schedule produced. Hindi et al. [Hindi et al., 2002] performed an empirical study of 2370 instances. The result showed that GA is effective of finding near–optimal solutions for RCPSP. GA has been widely studied in tackling software project management as a project scheduling problem. However, the current state–of–art researches demonstrate a lack of flexibility to effectively cope with the dynamic of modern software projects, such as, changing staff during the project is not allowed, skill set for each individual staff is normally fixed which does not consider the employees' abilities of learning new skills.

2.5.2.2 Project Resource Allocation with Meta-heuristics

The Generalised Assignment Problem (GAP) is a well–known NP–hard problem of finding the optimal assignment of n jobs to m servers in a manner that the overall cost is minimised and constraints is not violated. It can be stated as follows. Let $I = \{1, 2, ..., n\}$ be the set of jobs to be processed on servers $J = \{1, 2, ..., m\}$. For each server j, a_{ij} is the capacity absorption when job i is assigned to server j, and b_j is the available capacity of server j. Furthermore, there is a assignment cost for assigning job i to server j, which is denoted by c_{ij}. The formulation of the GAP is to minimise the total cost of the process:

$$Minimise: \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{2.4}$$

subject to

$$\sum_{i=1}^{n} a_{ij}x_{ij} \leq b_j, \quad \forall j \in J \tag{2.5}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad \forall i \in I \tag{2.6}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in I,\ \forall j \in J \tag{2.7}$$

where $x_{ij} = 1$ when job i is assigned to server j, otherwise $x_{ij} = 0$. The constraint (2.5) ensures that the servers are not overloaded, while constraint (2.6) ensures that a job is only assigned to one server. Let's define the assignment matrix X as:

$$X = \{\ x_{ij} \mid \forall i \in I,\ \forall j \in J\ \}$$

An early survey of the techniques of solving GAP can be found in [Cattrysse and Wassenhove, 1992], which was conducted by Cattrysse and Wassenhove in 1992. Ross and Soland [Ross and Soland, 1975] and Fisher et al. [Fisher et al., 1986] devised exact algorithms based on branch-and-bound techniques. Guignard and Rosenwein [Guignard and Rosenwein, 1989] proposed a method that included an improved Lagrangian dual-ascent procedure that solved GAP with more than 500 variables for the first time.

The essential objective of GAP is to find an optimal assignment matrix X, and the minimisation of total cost can be considered as the evaluation of the fitness of the candidate matrix. It is quite straightforward to apply the GAs. Wilson [Wilson, 1997] and Chu and Beasley [Chu and Beasley, 1997] published some of the earliest works on using GAs for the GAP at almost the same time. Both previous papers used simple GAs to develop a solution matrix for the GAP in two similar ways, and, because of that, it was apparent that both of these works were stimulated by Beasley and Chu's previous work on using GA for the set-covering problem [Beasley and Chu, 1996]. Wilson attempted to improve the feasibility of optimal, but infeasible, candidate solutions during the evolution process, while Chu and Beasley attempted to improve the optimality of feasible candidate solutions.

Recent applications of GA for solving GAP can be found in various areas, such as airline-crew scheduling [Levine, 1996], nurse scheduling [Aickelin and Dowsland, 2004], and scheduling students' project assignments [Harper et al., 2005]. Hajri-Gabouj [Hajri-Gabouj, 2003] developed a fuzzy, multi-objective GA to solve a multi-level GAP that is usually encountered in the clothing industry. In 2007, Majumdar and Bhunia [Majumdar and Bhunia, 2007] used interval values in the representation of an assignment matrix. They compared two forms of crossover schemes for the matrix representation of chromosomes: 1) a modified form of whole arithmetical crossover and 2) matrix binary crossover (MBX). The performance of the latter was found to be better. In 2008, Garrett and Dasgupta [Garrett and Dasgupta, 2008] proposed tools for examining the multi-objective landscape of a number of instances of the GAP using tabu search.

In summary, search based techniques have been studied as a means of solving software project management problems. However, there are still some unexplored areas, such as software project staff absenteeism. In the meantime, new techniques have been developed by evolutionary optimisation researchers. Theses new optimisation techniques, such as the cooperative co-evolutionary algorithm, have great potential to solve more complex and larger software project management problems.

Chapter 3

Industrial Data for Evaluation

A total number of seven sets of industrial data have been used in the empirical studies for this thesis. These data come from real world projects including one set of requirement data from Motorola that is studied in Chapter 4, and six sets of software project planning data that are studiedin Chapters 5 and 6. The data was previously collected by third parties' research efforts conducted in [Baker et al., 2006, Di Penta et al., 2011, Gueorguiev et al., 2009]. This chapter explains the origins and structure of the data and the method used to adapt them to assist the studies of this thesis.

3.1 Industrial Requirement Data

In the empirical studies conducted in this thesis, a set of real data from Motorola Inc. was adapted from [Baker et al., 2006]. The author did not participate in the process of collecting the data. The requirement data was available to the author in the form of a table as shown in Table 3.1. It contains 35 software features that can be implemented into the future model of a mobile phone. Each feature has its own Cost, which represents the amount of the effort and resource it takes to implement the feature. Expected revenue represents the possible revenue that each feature is expected to bring to the company. It ranges from 1 to 3, with 3 being the biggest revenue and 1 the smallest. Dependency relations between the these 35 features were very sparse, so the issue of the feature dependency was ignored in current stage of the project.

Table 3.1: Data of thirty five software features for a future model of a mobile phone from Motorola Inc.. Adapted from [Baker et al., 2006].

Feature ID	*Cost*	*Revenue*	*Feature ID*	*Cost*	*Revenue*
Feature 1	100	3	Feature 19	1000	3
Feature 2	50	3	Feature 20	120	3
Feature 3	300	3	Feature 21	300	3
Feature 4	80	3	Feature 22	50	3
Feature 5	70	3	Feature 23	10	3
Feature 6	100	3	Feature 24	30	3
Feature 7	1000	3	Feature 25	110	3
Feature 8	40	3	Feature 26	230	3
Feature 9	200	3	Feature 27	40	3
Feature 10	20	1	Feature 28	180	1
Feature 11	1100	3	Feature 29	20	3
Feature 12	10	3	Feature 30	150	3
Feature 13	500	3	Feature 31	60	3
Feature 14	10	1	Feature 32	100	1
Feature 15	10	3	Feature 33	400	3
Feature 16	10	2	Feature 34	80	2
Feature 17	20	1	Feature 35	40	1
Feature 18	200	1			

3.2 Industrial Project Plan Data

There is a total number of six sets of real world software project planning data available to us. This section aims to describe the fundamental details of each of the six projects as the followings: 1) industrial context of each project; 2) a table of summary of key features such as: number of Work Packages (WPs), required skills and staff; and 3) dependency graph that visualises the interdependency among WPs, resource and efforts required by each WP, and critical path of each project.

3.2.1 Industrial Contexts of Real Software Projects

Project A (Y2K) is a massive maintenance project for fixing the Y2K problem in a large financial computing system from a European organisation. According to its work breakdown structure, the application was decomposed into WPs loosely coupled, elementary units subject to maintenance activities. The entire system was decomposed into 84 WPs, each one composing, on average, of 300 COBOL and JCL files. Each WP was managed by a WP leader and assigned to a maintenance team. No WP dependency was documented, and thus, no constraint had to be satisfied in its scheduling. *Project A* can be considered as representative of massive maintenance projects related to highly-standardized tasks such as currency conversion, change of social security numbering, etc.

Project B (Data-intensive System) aimed at delivering the next release of a large data-intensive, multi-platform software system, written in several languages, including DB II, SQL, and .NET. The project is composed of 108 WPs for which WP inter-dependence information is available. Though a little smaller (in terms of total effort required) than *Project A*, the presence of a total of 102 dependences between the project's WPs considerably complicates the problem of project management.

Project C (SoftChoice) aimed at delivering an online selling system to provide North American businesses of all sizes with a fast and flexible way to select, acquire and manage all their hardware and software needs. The system includes the development and testing of website, search engine, and order management, etc.

Project D (QuoteToOrder) is a medium-sized project implemented in a large Canadian sales company. This project aims to add new enhancements to the supply chain of the company by allowing instant and on-demand conversion of quotes to orders. This change is both internal and customer facing, ultimately affecting every level of the organization (Web, internal development, database, sales, and customers). Most of the employees were involved eventually in training sessions.

Project E (Database) is a database upgrade procedure. It consisted of a migration from the Oracle version 9g to 10g. In addition, the upgrade affected 70% of the internal applications, since a considerable number of them relied on Oracle Forms. There were different layers of the organisation involved including DBAs, BSAs, developers and users. Furthermore, the project also included the training of the staff for the new features of the system.

Project F (SmartPrice) consists of a supply chain enhancement of medium size affecting mostly the website as well as a few internal applications. It is a customer-facing enhancement to the sales process of the same sales organisation. This feature provided more adequate pricing mechanism as well as a method for discounts, voucher use, pricing conversion, etc. The enhancement influences directly all of the revenue stream of the company and as such, extensive QA was involved. This feature affected mostly the web portion of the company's infrastructure with smaller impact on the underlying database and other internal software. The project ended with adequate employee training to ensure that the features were used properly by everyone.

3.2.2 Features and Characteristics of Projects and Visualisations

Table 3.2: Features of all six software projects

Code Name	*#WPs*	*#Dep.*	*#Required Skills*	*#Staff*	*Total Effort (person-days)*	*Note*
A	84	**N/A**	**N/A**	80	4287	No Dep., No Skills.
B	120	102	**N/A**	**N/A**	594	No Skills.
C	253	226	10	18	833	SoftChoice
D	60	57	7	9	68	QuoteToOrder
E	106	105	5	7	674	Database
F	72	71	6	13	196	SmartPrice

The data was initially provided in the format of Microsoft Project (*.mpp) files. The adaptation process includes the following steps: 1) an automated Transitive Reduction [Aho et al., 1972] is implemented to reduce the redundant dependency information, 2) the original data contains workpackages that require zero effort to represent milestones, these workpackages are eliminated, 3) those 'parental' workpackages that act as a summary of several sub-workpackages are also eliminated to ensure the accuracy on total efforts, and 4) because the eliminations of certain workpackages cause potential lost of dependency information, the dependency graph is reconstructed accordingly to ensure the same precedence of executing workpackages. This adaptation process are automated and performed by scripts mainly for the purpose of avoiding inaccuracy caused by manual processing.

The key features of these software project, posterior to the adaptation process, are summarised in Table 3.2. Two most distinguishable differences among these six industrial projects are the availabilities of the information on: 1) dependency among a project's WPs, and 2) the skills and staff required by individual WPs. Previous related research on *Project A* and *B* can be found in [Antoniol et al., 2004, Antoniol et al., 2005,Di Penta et al., 2007,Di Penta et al., 2011], and related work with *Project C* to *F* can be found in [Jose, 2008,Gueorguiev, 2008]. This thesis is the first research to utilise the information on *Required Skills* and *Staff* for each WP of Project C to F.

The *dependency information* among a project's WPs defines the precedence of which WPs can be executed. Without dependency information, all WPs in a project are considered to be independent WPs that can be executed in parallel. For example, WPs in *Project A* can be distributed to different centres without the consideration of where and when the other WPs is executed. On the other hand, for Project B to F, each project has a set of explicit dependency constraints that need to be satisfied to fulfil interdependency requirement among WPs. The explicit dependency information for *five* projects are illustrated as *solid black lines* in Figure 3.1 to 3.5 respectively for *Project B* to *F*.

The information of *skills required* by each individual WP allows us to investigate the allocations of staff and resources with the help of simulating scenarios that are more closed to the real–world case. It prevents the simulation model from becoming over-simplified. The requirement of resources for each WPs in *four* projects (Project C to F) are represented by *coloured lines* in Figure 3.2 to 3.5 respectively.

Furthermore, the degree of *normalised efforts required* of each WP is also reflected by the level of grey of each WP shown on the figure, and those WPs on *critical path* are marked with labels in red.

3.3 Limitations of Data Usage

In this thesis, all real data is secondary data collected by third parties. This section lists the key points which could lead to threats to the data validity of this thesis:

- The author did not participate in the process of collecting the data, neither had the opportunity to discuss the quality of the data in particular with the data collectors. However, the lack of knowledge on data collecting methods can be assumed to have trivial impact on the research because the research utilised only the quantitative parts of the data which were estimated by either the experts or project managers from industry.

- The studies utilise only the quantitative parts of the data, i.e.: the cost and revenue of the requirements, the duration and dependency of the WPs. These variables were estimated by either experts or project managers. There are problems caused by the potential inaccuracy on estimations.

- There is only one set of requirement dataset and six sets of project plans. It may not be valid to generalise all the observations and findings from such a relatively small number of sample.

- This thesis presents the results based on the data from real world projects. The author did not elicit contributions from the experts and managers who may provide quite different explanations on the results from the perspective of participators of the projects. It would be worthwhile to further validate the result of this research, although it is not possible to arrange such event.

- The project plans were initially provided in a form that contains redundant information to provide easier accessibility to human. Data transformation is necessary in the data adaptation process to reduce the redundant information. Although the data transformation process is repeatable via the automated scripts, the original data is not available to the public due to the prior Non-Disclosure Agreement between the CREST centre (Centre for Research on Evolution, Search and Testing) and the third parties who collected the data.

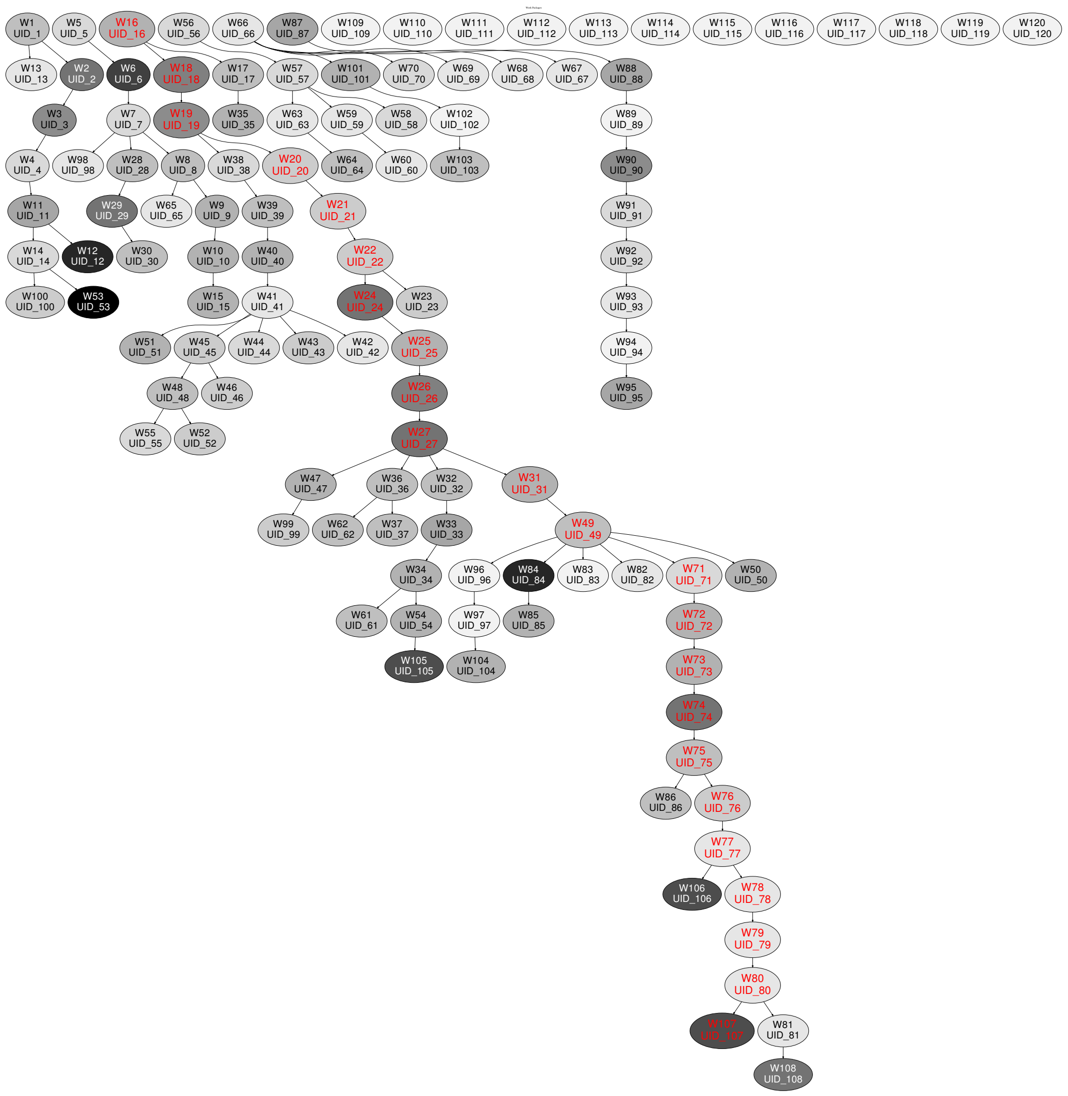


Figure 3.1: Work Package Dependency Graph of Project B. It shows the work packages, the normalised efforts (*degree of greyness*), dependency relationship among WPs (*black lines*), and the critical path (*WPs that marked with red tags: W[id] and UID_[uid]*).

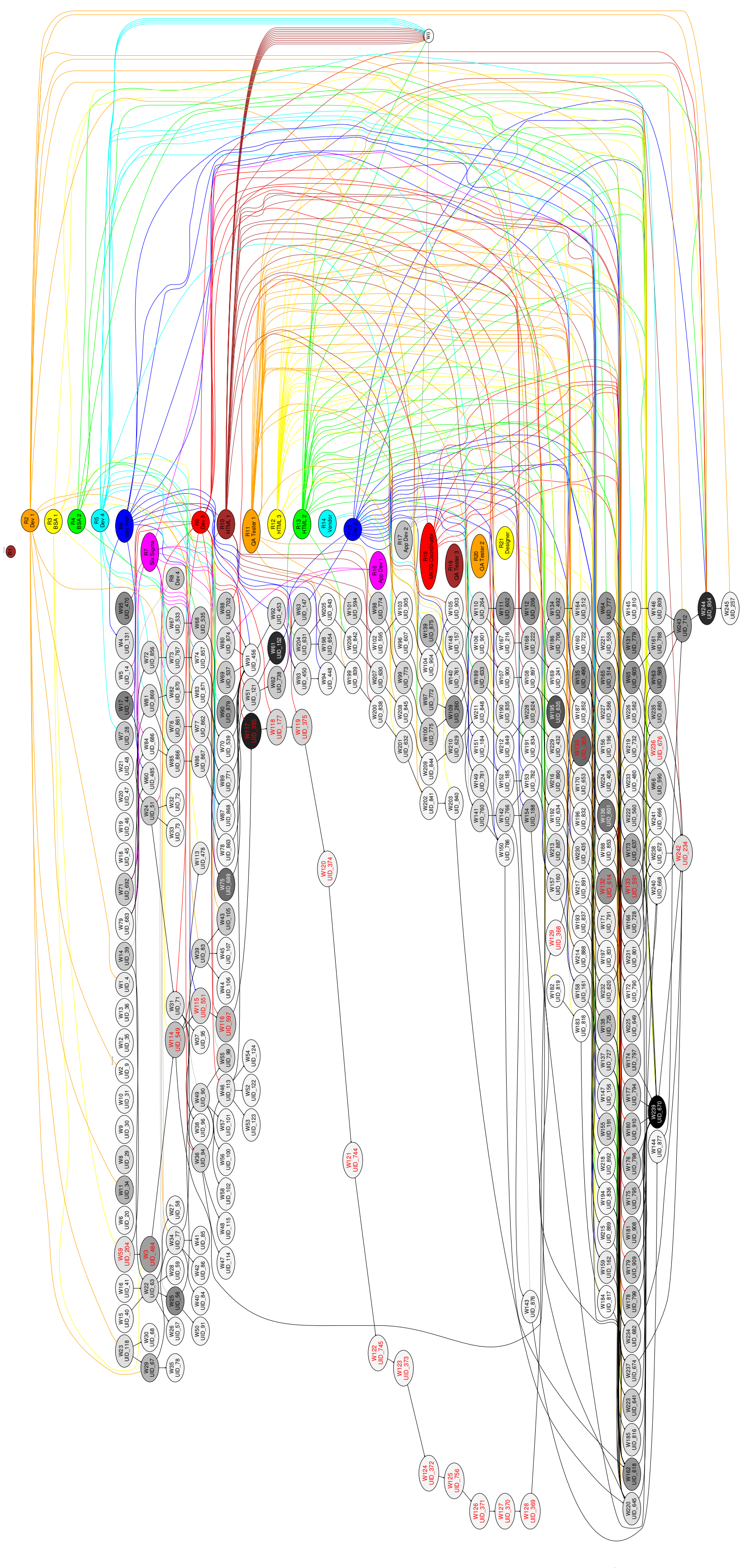


Figure 3.2: Work Package Dependency Graph and Resource Allocation Graph of Project C. In addition, it shows the resources (*coloured ellipsis*), and the corresponding WPs require such resources (*coloured lines*).

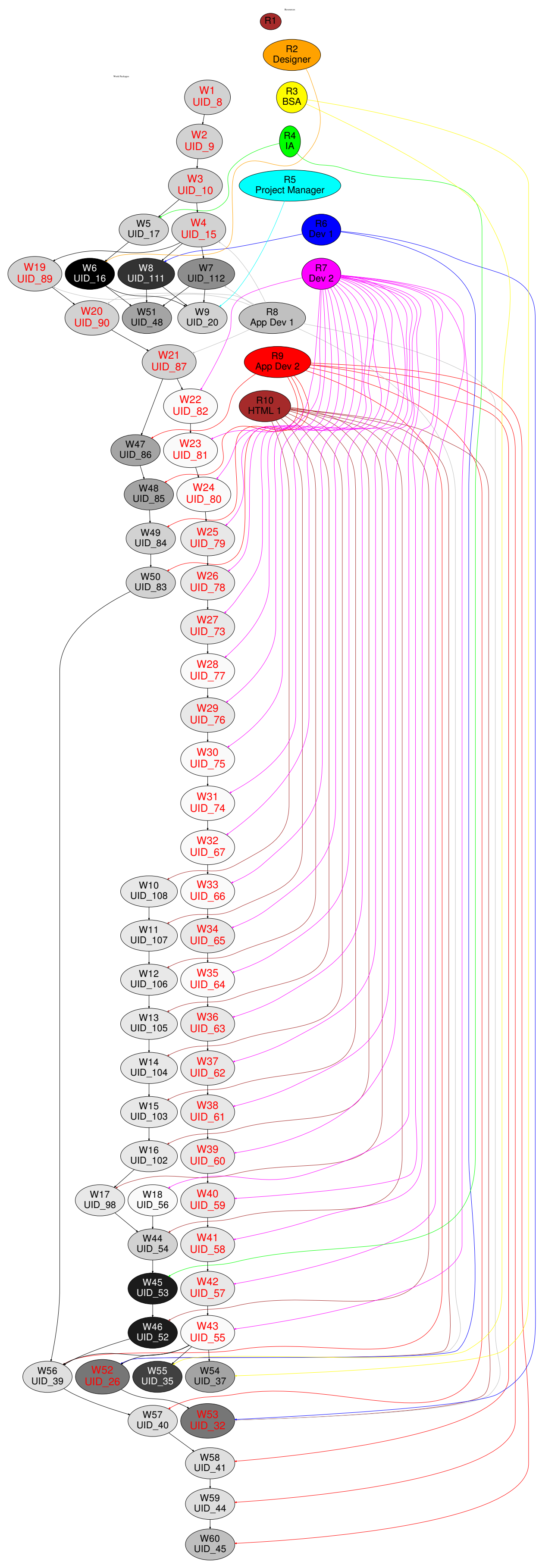


Figure 3.3: Work Package Dependency Graph and Resource Allocation Graph of Project D

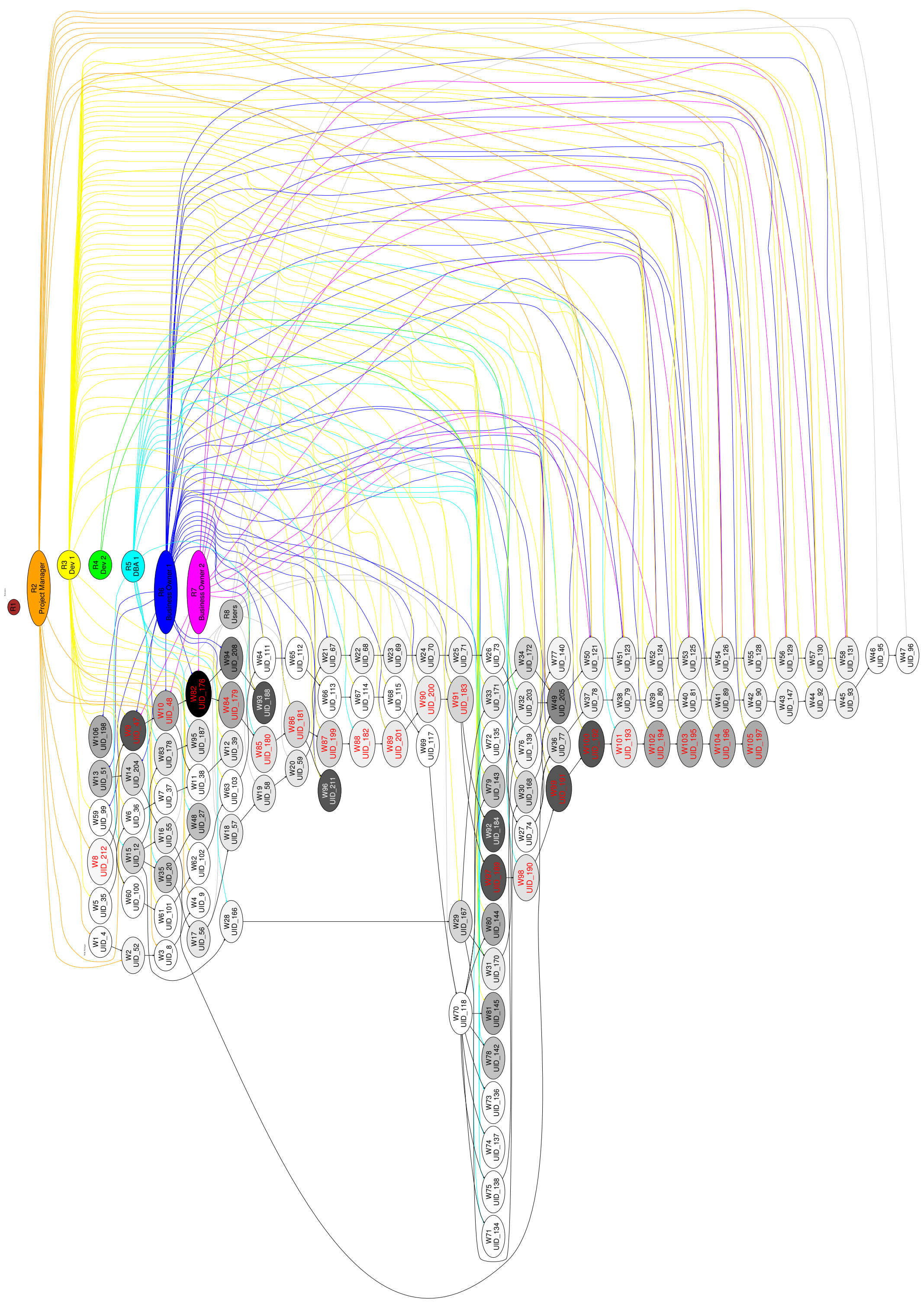


Figure 3.4: Work Package Dependency Graph and Resource Allocation Graph of Project E

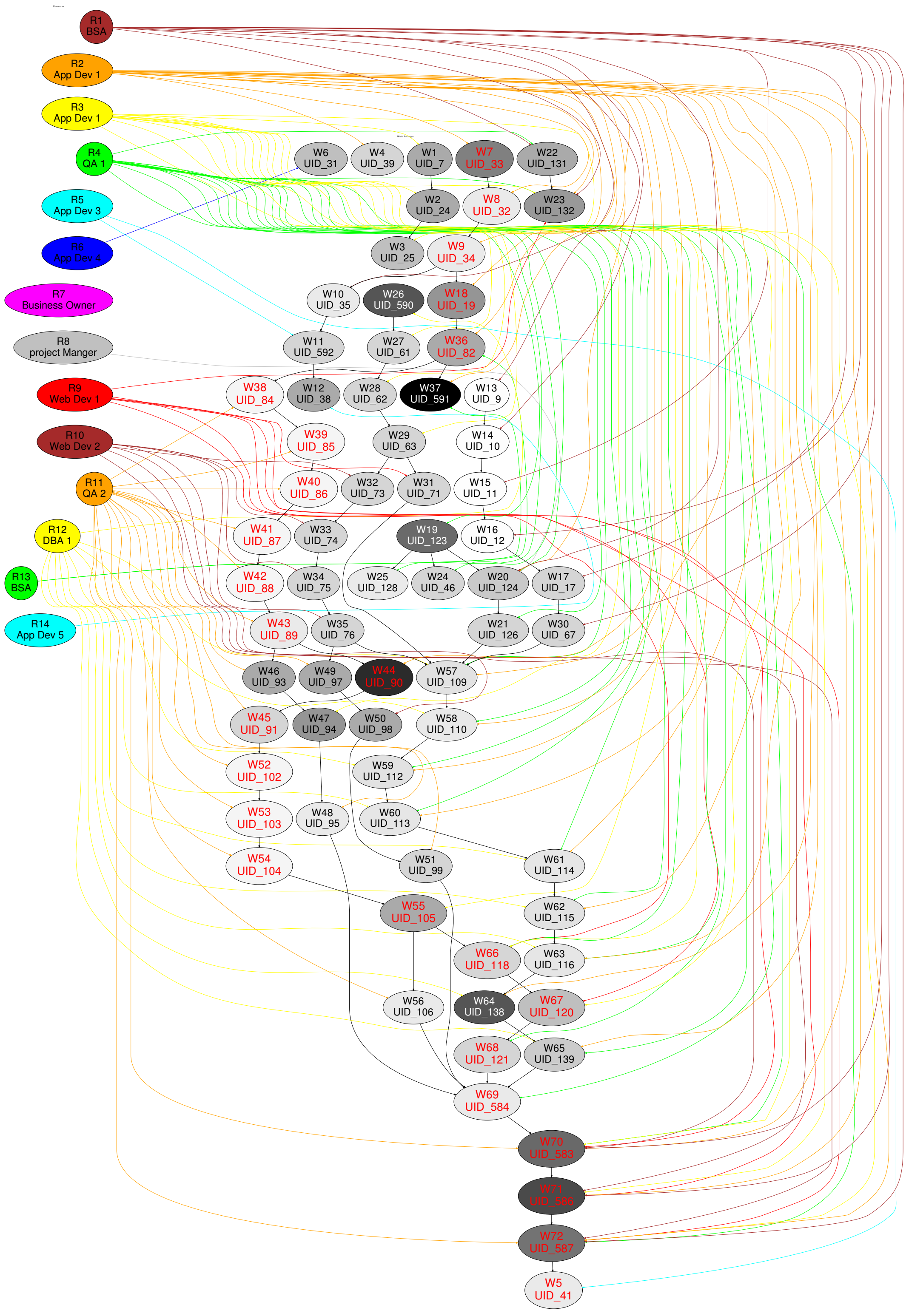


Figure 3.5: Work Package Dependency Graph and Resource Allocation Graph of Project F

Chapter 4

Sensitivity Analysis on Cost Estimation of Requirements Selection

4.1 Introduction

One of the common problems in requirements engineering is caused by the uncertainties that are inherent in the decisions made by the requirements engineer. Most of the data needed by the requirement engineer, such as expected revenue, development cost or duration, is inherently unknown at the time of requirement planning stage. The clients of the product also contribute to these uncertainties because often they do not possess clear idea about which features they want before actually see it. Naturally, the requirement engineer has to balance many complex criteria based on estimated data.

It is a well-known fact that cost estimation is a difficult and demanding activity [Armour, 2002, Boehm, 1984]. It is also widely believed that the cost estimates produced by software engineers tend to include a great degree of inaccuracy [Flyvbjerg et al., 2002, Flyvbjerg et al., 2005]. This is not due to the ineptness of the requirements engineer; it is rather because of the astonishingly wide variety of the problems that software engineering faces. Unlike other engineering disciplines, there are fewer well-understood construction approaches, which consequently causes the inaccuracy.

This work does not attempt to resolve the inaccurate cost estimate problem; it seems that the problem will remain unsolved for the foreseeable future of software engineering. Rather, the work seeks to introduce an approach to provide the requirements engineer with a decision support system guided by Search-Based Software Engineering (SBSE). The approach assesses the impact of inaccuracies of the cost estimation of each requirement on the solutions to the requirements allocation problem, known as the Next Release Problem [Bagnall et al., 2001]. The Next Release Problem is the problem of selecting the software requirements to be implemented in the next release of a product so that benefits such as customer satisfaction or revenue is maximised while all the constraints such as budget are satisfied. The decision support system aids the requirements engineer by identifying the sensitive regions in the estimated data which will lead to relatively higher impact on the selection of the requirements. This information then can be used to focus the effort on obtaining more accurate estimation of those requirements.

Each set of estimates and customer choices denotes a separate and unique optimisation problem. The structure of the data and the relationships between estimated data may create unexpected interactions between requirement estimates, which can yield a butterfly effect; a small inaccuracy in a low cost requirement can have an unexpectedly large effect of the overall decision. Because of the size of the data sets involved and the inherent complexity of the interactions between estimates, it is nearly impossible for an engineer to fully comprehend these hidden relationships without automated decision support.

The intuitive answer to the sensitivity of cost estimation problem is that, the more expensive the requirement is, the greater impact it will have on the result when estimated inaccurately. Also, similarly, we expect that higher levels of inaccuracies are most likely to cause greater impacts. This thesis indeed statistically confirms these intuitive assumptions. However, there are exceptions to these trends. These exceptions require careful cost estimation, because they can have unexpectedly high impacts on the selection of requirements. A heat-map style visualisation is generated using a search-based approach, to identify these sensitive exceptions in the data. The hot-spots on the heat map will indicate areas where a particular inaccuracy level for a particular requirement estimate can lead to high impact. The heat-map provides an intuitive visual aid to comprehend the complex interaction in the data set.

The chapter presents two different formulations of the problem. With the single-objective formulation, the requirements engineer assesses the impact of inaccuracy at a specific level on weighted customer satisfaction values. In this model, the requirements engineer knows the expected inaccuracy and seeks to identify overall budget levels and particular requirements that are sensitive to this. The second formulation is the multi-objective formulation in which the requirements engineer simply seeks to reduce estimated cost and increase estimated revenue, but does not know how inaccurate the estimates are likely to be. The single objective formulation is more appropriate for a mature organisation with a history of development and a consequent knowledge of likely levels of estimate inaccuracy. The multi objective formulation has the advantage that it can be applied without any knowledge of likely estimate inaccuracy levels.

Both formulations are applied to both synthetic and real world data. The primary contributions of the chapter are as follows:

1. To show how SBSE can be used as a technique for sensitivity analysis in requirements engineering.

2. To present two formulations of the NRP of requirements engineering and shows how SBSE can be used for both formulations, presenting an evaluation using real world data and synthetic data.

3. To show how heat-maps can be used to intuitively identify unexpectedly sensitive requirements estimates to guide the decision maker, providing insight into the structure of their estimate data.

4.2 Background

The work presents a sensitivity analysis for two different formulations of the Next Release Problem (NRP): single-objective version and multi-objective version.

4.2.1 Single-objective Next Release Problem

The single-objective formulation follows the definition of NRP by Bagnall et al. [Bagnall et al., 2001]. First, it is assumed that for an existing software system, there is a set of possible software requirements, denoted by:

$$\mathcal{R} = \{r_1, \ldots, r_n\}$$

For the sake of simplicity, it is also assumed that there is no dependency relation between those requirements. Bagnall et al. note that any instance of NRP with dependency relation can be converted to a *basic* NRP by merging the requirements that belong to dependency chains [Bagnall et al., 2001].

The cost of fulfilling this set of requirements $r_i (1 \leq i \leq n)$ is denoted by:

$$Cost = \{cost_1, \ldots, cost_n\}$$

The expected revenue of every possible requirement is denoted by:

$$Revenue = \{revenue_1, \ldots, revenue_n\}$$

The decision problem form of NRP is the question of finding the optimal subset(s) of requirements to maximise the total revenue and minimise the cost of development. The decision vector, $\overrightarrow{X}$, is represented by:

$$\overrightarrow{X} =< x_1, \ldots, x_n >$$

where ith element of $\overrightarrow{X}$ is 1 if the ith requirement is to be implemented and 0 if it is not. Now, given an instance of the decision vector, $\overrightarrow{X_1}$, its fitness, $F(\overrightarrow{X_1})$, is the sum of expected revenues for the requirements to be implemented by $\overrightarrow{X_1}$:

$$F(\overrightarrow{X_1}) = \sum_{i=1}^{n} revenue_i \cdot x_i$$

Similarly, the cost of implementing a set of requirements represented by $\overrightarrow{X_1}$ is:

$$cost(\overrightarrow{X_1}) = \sum_{i=1}^{n} cost_i \cdot x_i$$

Given a budget of b, the single-objective NRP is a problem of finding a decision vector $\overrightarrow{X}$ such that $F(\overrightarrow{X})$ is maximised while satisfying $F(\overrightarrow{X}) \leq b$:

$$\text{Maximise } \sum_{i=1}^{n} \mathit{revenue}_i \cdot x_i$$

$$\text{while subject to} \sum_{i=1}^{n} \mathit{cost}_i \cdot x_i \leq b$$

4.2.2 Multi-objective Next Release Problem

The multi-objective Next Release Problem (MONRP) is a multi-objective optimisation version of NRP. In multi-objective optimisation problems, there are multiple objectives expressed in fitness functions, which are often in conflict with each other [Coello, 2000]. In case of MONRP, it can be said that the expected revenue and the development cost of a product are in conflict with each other.

The multi-objective formulation is defined following Zhang et al. [Zhang et al., 2007]. Unlike the single-objective formulation, the cost is no longer a constraint. In multi-objective formulation, the development cost is minimised while the expected revenue is maximised.

$$\text{Maximise } \sum_{i=1}^{n} \mathit{revenue}_i \cdot x_i, \quad \mathit{and}$$

$$\text{Minimise } \sum_{i=1}^{n} \mathit{cost}_i \cdot x_i$$

In multi-objective optimisation, a solution A is said to dominate a solution B if and only if A is at least equal to B in all objectives, and excels B in at least one objective. This is called Pareto-optimality. As a result, a solution of a multi-objective optimisation problem is expressed in a Pareto-front, which is a set of multiple solutions that do not dominate each other.

4.3 Sensitivity Analysis in NRP

Since the models used in the empirical studies are small enough to be solved quickly, a *brute force* approach is implemented for sensitivity analysis: simply modify the initial input data and run the algorithm repeatedly to see how the result changes. Figure 4.1 illustrates the difference between general optimisation process and sensitivity analysis. The cost of each requirement is modified to simulate the inaccurate estimation. This data is then fed into a meta-heuristic optimisation algorithm designed for NRP, which will produce an alternative solution. The impact is then evaluated by measuring the distance between the original solution and the alternative solution.

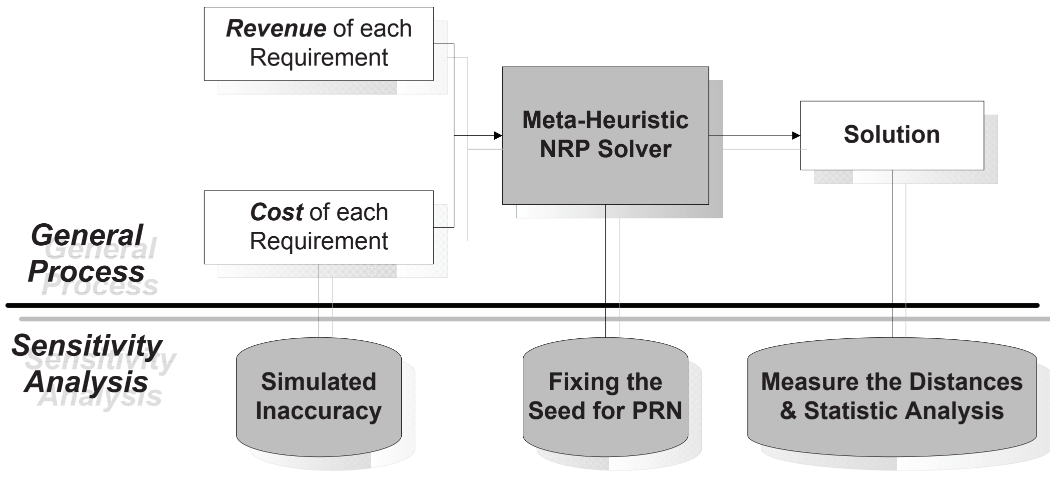


Figure 4.1: Sensitivity Analysis Flow Chart

There are two critical elements that are required in order to simulate *what-if* scenarios in which a particular estimation is inaccurate. First, the algorithm used to solve NRP has to be deterministic, otherwise it is impossible to determine whether the observed change in the result is due to the inaccurate estimation or the randomness of the algorithm.

In most (if not all) multi-objective evolutionary algorithms, *Pseudo Random Number(PRN)* are used in the procedure of evolutionary calculation. For instance, pseudo random number is used in generating the initial population, selecting the bits in candidates to perform mutation and crossover. Due to the inherent randomness of those evolutionary algorithms, and the fact that it can not guarantee the global optimum, every 'run' of the implementation would provide a different result even if the input data are identical. In order to perform the sensitivity analysis, we need to distinguish the difference between the indeterminacy of the algorithm itself and the changes caused by the modification on the input data.

We introduce fixed seed for *PRN* to provide a identical sequence of PRN for each execution of the implementation. With the identical sequence of PRN, we can ensure that the change on the result is caused by the change on the input data only.

The second element required by sensitivity analysis is a method that can measure the changes brought in by the error in a quantitative manner. If it is not possible to express the changes in quantitative forms, it would also be impossible to compare the criticality of errors. The actual method of measurement is specific to the definition and representation of the problem.

With the single-objective formulation, we evaluate the difference between two decision vectors by their Hamming distance. This is possible because the greedy algorithm produces a single solution to an instance of NRP problem. However, NSGA-II produces not a single solution, but a set of solutions that form Pareto Frontier. Therefore, the difference should be measured between two sets of solutions (two Pareto fronts), not two different solutions. In order to measure the distance between two sets of solutions, the Generation Distance is used [Van Veldhuizen, 1999]. It is based on the calculations of *Euclidean Distance* of the solutions on two fronts.

To calculate the distance between two fronts (f_a, f_b) of two different executions of optimisation, we define (A_1, A_2, ..., A_n) to denote the n solutions belonging to front f_a, while (B_1, B_2, ..., B_m) denote the m solutions belonging to front f_b, where n and m are the numbers of solutions contained by each front respectively.

First, we need to calculate the distance from one solution A_i on front f_a to front f_b.

The distance from solution $A_i(x_i, y_i)$ to solution $B_j(x_j, y_j)$ is the Euclidean distance between coordinate values normalised to [0,1]. Distance between A_i and B_j is defined as:

$$Dis(A_i, B_j) = \pm\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

The distance from one particular point A_i to f_b is considered as the same distance from A_i to its geometrically closest point on front f_b. Distance between A_i and front f_b is defined as:

$$Dis(A_i, f_b) = Dis(A_i, B_j)$$

where B_j is the closest point to A_i on front f_b.

The distance from front f_a to f_b is then calculated as the mean value of the distance from every point on f_a to f_b.

$$Dis^*(f_a, f_b) = \frac{\sum_{i=1}^{n} Dis(A_i, f_b)}{n}$$

where n is the number of optimal solutions on front f_a.

Finally, in order to achieve fair contributions from both fronts to the distance calculation, we develop the formulation to calculate the distance between two Pareto fronts f_a and f_b as below:

$$Distance(f_a, f_b) = \frac{Dis^*(f_a, f_b) + Dis^*(f_b, f_a)}{2}$$

4.4 SA Experimental Set Up

The argument of data sensitivity problem is based on the assumption that some of the estimated quantitative data may contain some errors. The amount of the actual error will be known only afterwards. However, it is possible to measure the repercussions of the potential errors by trying out various *what-if* scenarios. If an introduction of a certain deliberate error to a specific part of data creates large amount of change in the final solution, it would be safe to say that the specific part of data is highly sensitive to an error. With this knowledge, the decision maker can manage the potential risks more efficiently, as well as focusing on elaborating the estimation of more sensitive data.

In case of NRP, the most important scenarios are the cases when the costs of some requirements are based on wrong estimation. The decision maker would want to know which requirement will create the most significant change in the final solution if there is an error in the estimation of its development cost. Therefore, the scenarios in this case will be different versions of the data, each containing a requirement with modified development cost. The alternative solution will be a subset of requirements selected based on the modified data. If the alternative solution is radically different from the original solution, it indicates that the introduced error brings in a significant change. If this process is repeated for each requirement with the same margin of error, it is possible to identify the requirement that is most sensitive to the same level of inaccuracy.

The intuitive answer to the cost sensitivity analysis problem is that the more expensive a requirement is, the bigger impact it will have if its cost is estimated inaccurately. Similarly, it can be said that the more inaccurate the estimation is, the bigger impact it will have on the result of NRP. We hereby call this the *Positive Correlation Assumption*, (PCA). More specifically, we denote the first assumption (between cost and impact) by PCA-1, and the second assumption (between inaccuracy and impact) by PCA-2. These assumptions are statistically tested against both synthetic and real-world requirement data. For this, the empirical studies utilise the greedy algorithm and NSGA-II to single- and multi-objective formulations of NRP with deliberate errors in the data set.

4.4.1 Greedy Algorithm

Greedy algorithm is known to be efficient and effective for 0-1 knapsack problem, which is the basis of NRP. It is constructive in nature and start with an empty set of selected requirement. At each iteration, a requirement is added to the set until no further additions can be made without exceeding the given budget. The choice of which requirement to select at each iteration is guided by the fitness value.

First of all, all the requirements are sorted according their fitness value (expected revenue). All those requirements with the highest fitness value will then be selected into the solution vector until the budget bound has been reached. Algorithm 1 shows the pseudo-code of the greedy algorithm used.

```
input  : N:number of requirements; cost; budget
output: solution; currentCost
Sort the requirements in the order of descending revenue
for $i \leftarrow 1$ to $N$ do
    if $currentCost + cost(i) \leq budget$ then
        $currentCost \leftarrow currentCost + cost(i)$;
        $solution(i) \leftarrow 1$;
    end
end
```

Algorithm 1: Greedy Algorithm

4.4.2 NSGA-II

The recent implementation of NSGA-II [Deb et al., 2002] from Zhang et al. [Zhang et al., 2007] is modified to be applied to Motorola's data set. Initially, a random parent population P_0 is created. The population size is N. The population is sorted using the non-dominated relations. Each solution is assigned a fitness value equal to its non-domination level. Binary tournament selection, crossover, and mutation operators are used to create the offspring population Q_0 of size N. Then the NSGA-II procedure goes to the main loop which is described in Algorithm 2. Maximising the overall revenue and minimising the overall cost of each solution are considered as the two objectives for NSGA-II.

```
while not stopping rule do
    Let $R_t = P_t \cup Q_t$;
    Let $F$ = fast-non-dominated-sort($R_t$);
    Let $P_{t+1} = \emptyset$ and $i = 1$;
    while $|P_{t+1}| + |F_i| \leqslant N$ do
        Apply crowding-distance-assignment($F_i$);
        Let $P_{t+1} = P_{t+1} \cup F_i$;
        Let $i = i + 1$;
    end
    $Sort(F_i, \prec n)$;
    Let $P_{t+1} = P_{t+1} \cup F_i[1 : (N\text{-}|P_{t+1}|)]$;
    Let $Q_{t+1}$ = make-new-pop($P_{t+1}$);
    Let t = t + 1;
end
```

Algorithm 2: NSGA-II Algorithm

4.4.3 Requirement Data

We used two sets of synthetically generated data as well as a set of real-world requirements data obtained from a large telecommunications company. The first synthetic data is generated purely randomly, i.e. there is no correlation between the cost of a requirement and its expected revenue, which is connected to its fitness value in the optimisation problem. The second set is generated so that the cost of a requirement has a positive correlation with its expected revenue. Both of the two sets of synthetic data contain 30 requirements. The cost and revenue for each requirement are generated using the uniform distribution over the interval of (1, 1500) and (1, 10) respectively. Comparing the results from these two synthetic data set allows us to test the statistical significance of PCA.

The real-world requirement data is obtained from a large telecommunications company. It originally contained 40 different features that are interference-free, i.e. any combination of which can be implemented into a single product. However, 5 features that represent the core functionality of the product were woven by dependencies between themselves, and it was decided that they will always be included in the final selection of requirements. This left us 35 features with so sparse dependency relationship that it could be ignored.

4.4.4 Evaluation

We modify the cost of each requirement using 21 different Percentage Increase in Actual Cost (PIAC) values ranging from -50% to 50% with steps of 5%. A positive PIAC value means that the actual cost has increased compared to the estimated cost, which means an underestimation. A negative PIAC value means that the actual cost has decreased compared to the estimated cost, which means an overestimation. The Euclidean distance between result from modified data and the original data is used to quantify the difference observed in these multiple executions. Spearman's rank correlation coefficient is used to test PCA and analyse how the changes on result relates to the modifications of initial data.

4.4.5 Research Questions

We present the following research questions. **RQ1** and **RQ2** concern the statistical significance of PCA.

RQ1: Does the sensitivity analysis confirm PCA-1, i.e. the correlation between the cost of a requirement and its impact on NRP with statistical significance?

RQ2: Does the sensitivity analysis confirm PCA-2, i.e. the correlation between the level of inaccuracy and its impact on NRP with statistical significance?

RQ1 and **RQ2** is quantitatively answered using Spearman's rank correlation analysis in Section 4.5. The third research question inherently requires qualitative analysis.

RQ3: Is there any exception to the general trend observed by PCA?

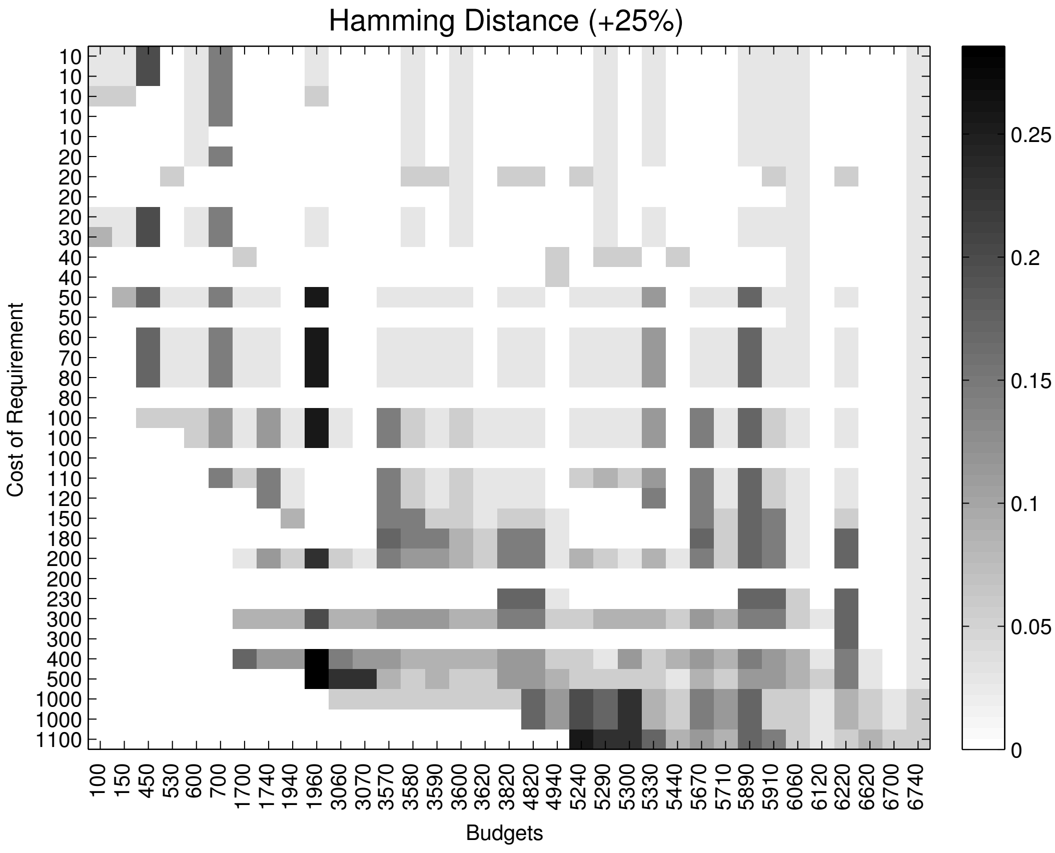
RQ3 is answered by analysing the heat-map visualisation in Section 4.5.

4.5 SA Results and Analysis

4.5.1 Result From Single-Objective Formulation

Figure 4.2 and 4.3 shows four heat-map visualisations from the results of sensitivity analysis on Motorola's data set, using single-objective formulation of NRP. The x-axis corresponds to different instances of NRP, sorted in the ascending order of the budget assigned to each instance. The y-axis corresponds to different requirements, sorted in the ascending order of their estimated cost. The two heat-maps on Figure 4.2 show the Hamming distance between the original greedy algorithm solutions and the alternative solutions with PIAC value of $\pm 25\%$, i.e. the underestimate or overestimate error by 25% margin. Similarly, the two on Figure 4.3 show the results with PIAC value of $\pm 50\%$, i.e. the underestimate overestimate error by 50% margin. The darker the colour presented, the bigger hamming distance is represented.

The heat-map reveals the complex interaction between the budget and the revenue and cost of each requirement. A single requirement shows varying levels of sensitiveness depending on the combination of the budget and the margin of error. However, some straightforward patterns can be easily observed. First, errors on expensive requirements do not have any impact on smaller budgets if the original estimated cost and modified cost are both larger than the given budget, of which the fact is reflected by the white area in the left lower corner of all four heat-maps. Second, when comparing the PIAC value of four heat-maps, the bigger PIAC value tend to bring more impact on the results. Third, when comparing the cost of each requirement, more expensive requirements tend to have bigger impact on the results. On the other hand, some cheaper requirements do not have any impact on the result if and only if their cost is overestimated (with PIAC=$-25\%, -50\%$), in which cases the amount of errors is relatively too small to free enough space on given budget for a more expensive requirement to be filled in. Another interesting observation is that some requirements do not have any impact on Hamming distance across all budget values.



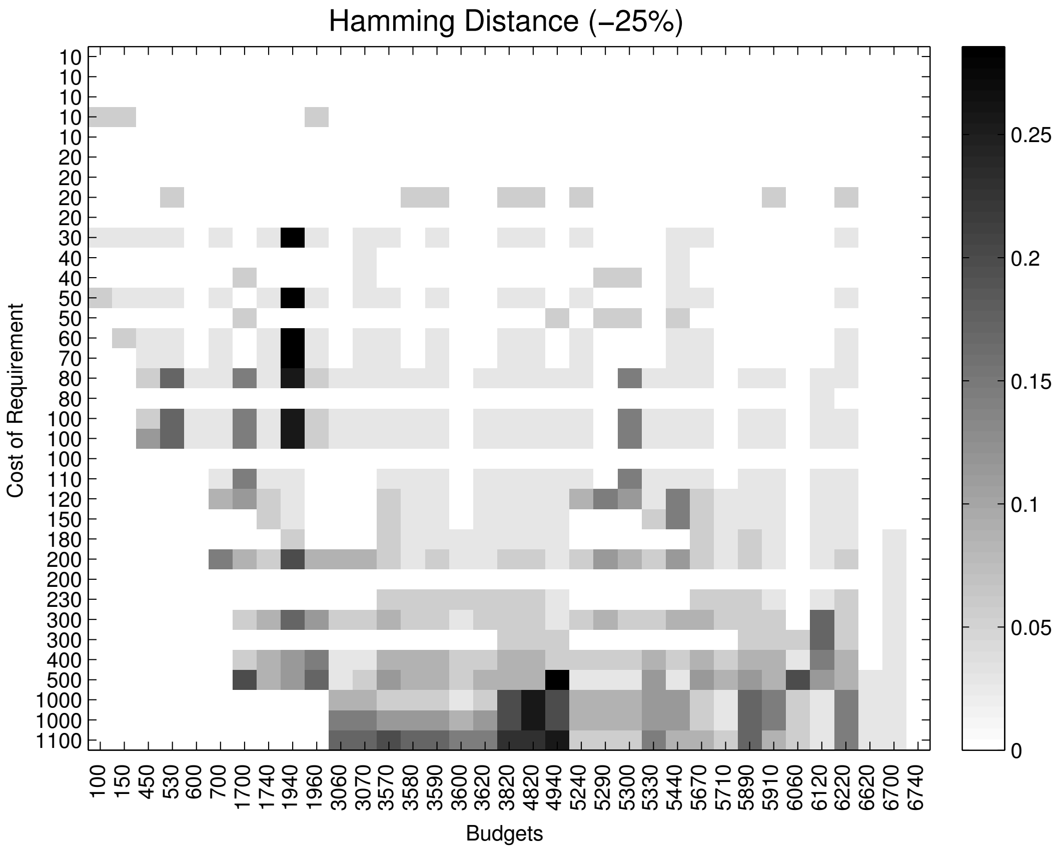
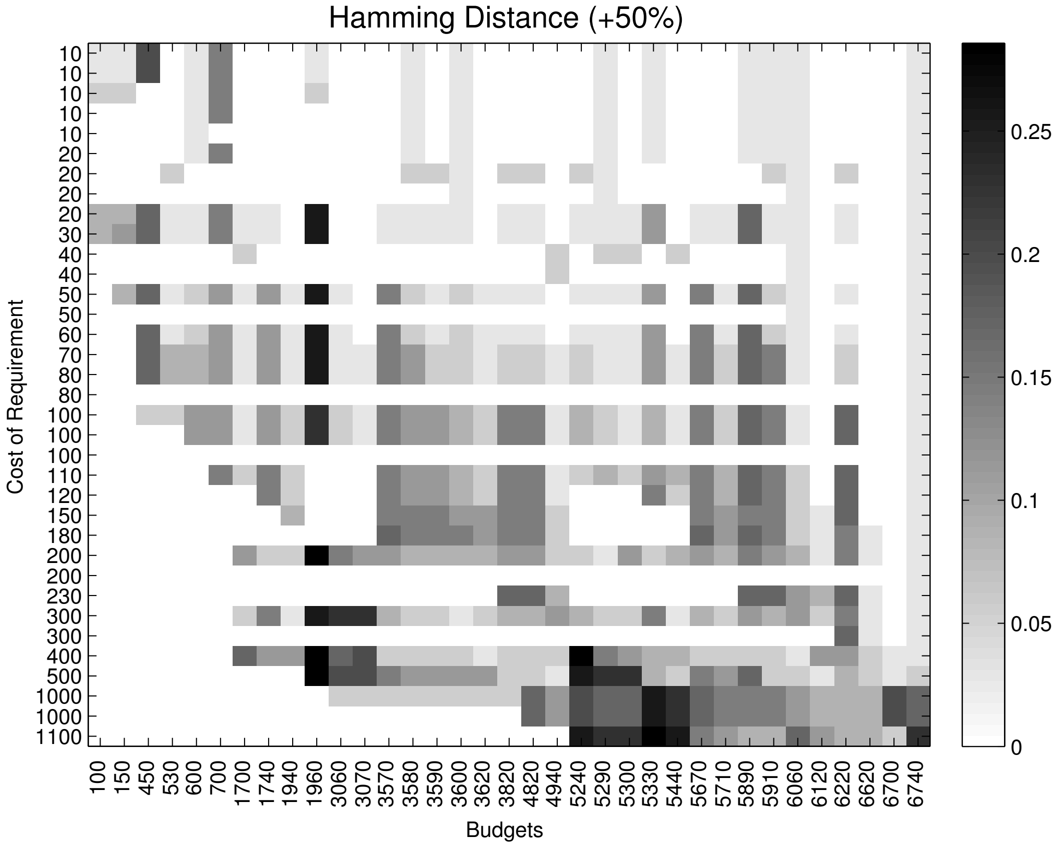


Figure 4.2: Hamming distance from the original solution to the solution obtained by the greedy algorithm with PIAC value of ±25%.



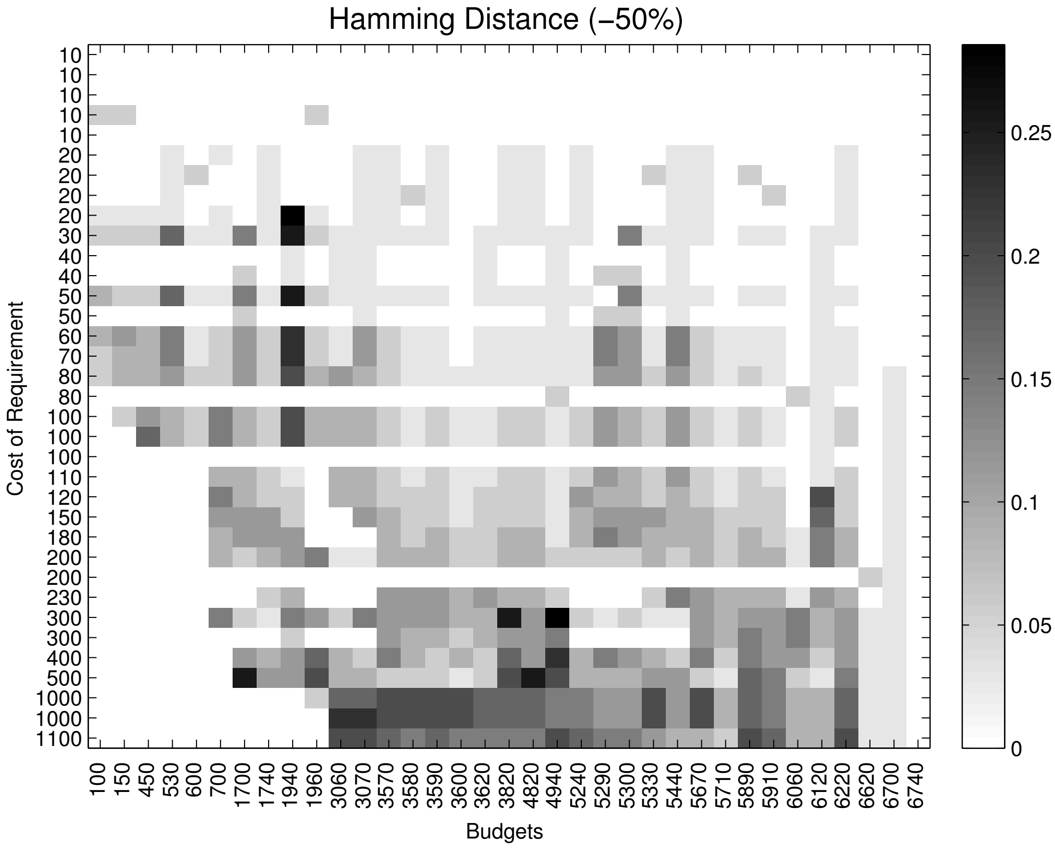


Figure 4.3: Hamming distance from the original solution to the solution obtained by the greedy algorithm with PIAC value of ±50%.

However, due to the existence of budget constraints, it is not possible to visualise the trend with respect to the cost, the expected revenue, and the PIAC value at the same time. For this, we turn to the multi-objective formulation; since the Euclidean distance between two Pareto fronts incorporate differences in both the revenue and the cost, we can observe the trend between PIAC value and its impact.

4.5.2 Result From Multi-Objective Formulation

Figure 4.4 shows the heat-map visualisation generated from the sensitivity analysis for the MONRP formulation. The x-axis represents different PIAC values, ranging from -50% (overestimation) to 50% (underestimation). The y-axis represents different requirements, sorted by their development cost. By cross-referencing x-axis and y-axis, it is possible to observe how much impact it makes to underestimate or overestimate the cost of a specific requirement by the given degree of error. The darker the colour is, the bigger impact the particular error has.

A few trends can be easily observed. First, one of the dominant trends across all three data sets is that the distance between the original and inaccurate Pareto front increases as PIAC value increases. Second, when comparing the cost of those requirements, more expensive requirements tend to have bigger impact on the results. These two observations are statistically tested in Section 4.5.3.

However, there are a few exceptions to the general trend. Certain requirements almost consistently have significant impacts on the result. For example, the second requirement in the real-world data set consistently produces Euclidean distance of 0.026 from PIAC value of -5% to -50%. This consistency provides two interesting insights into the real-world data set. First, this particular requirement brings about significant impact on the result even when its cost is reduced only by 5% (PIAC $= -5\%$). Second, and more interestingly, further reduction in its cost still produces the same level of impact up to reduction of 50% (PIAC $= -50\%$). This is due to the fact that the particular requirement has the lowest cost and lowest expected revenue among the requirements in the data set. It is possible to conclude that the threshold for overestimation of this particular requirement is 5%. Any overestimation that is larger than the threshold value would mean that the final solution will be different from the original solution.

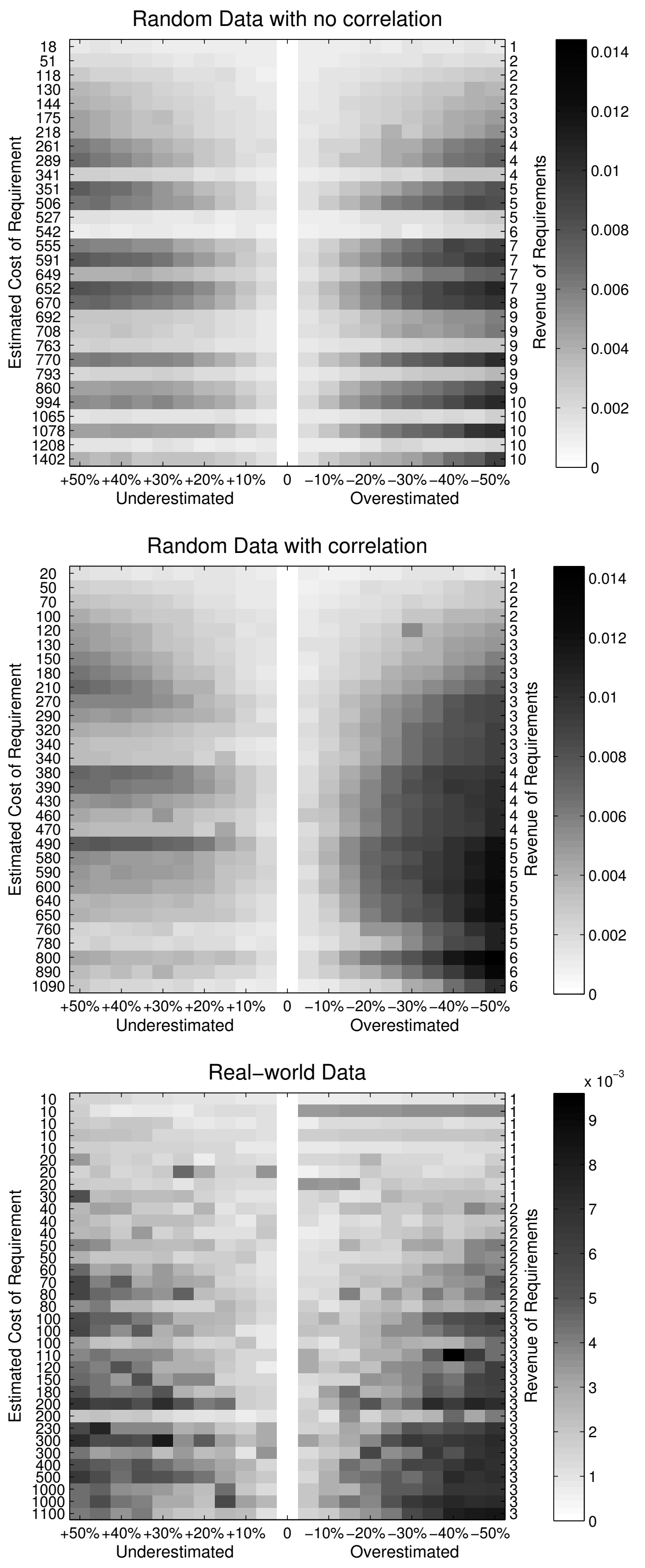


Figure 4.4: Euclidean distance between original estimated Pareto-front and actual Pareto-front by different PIAC values.

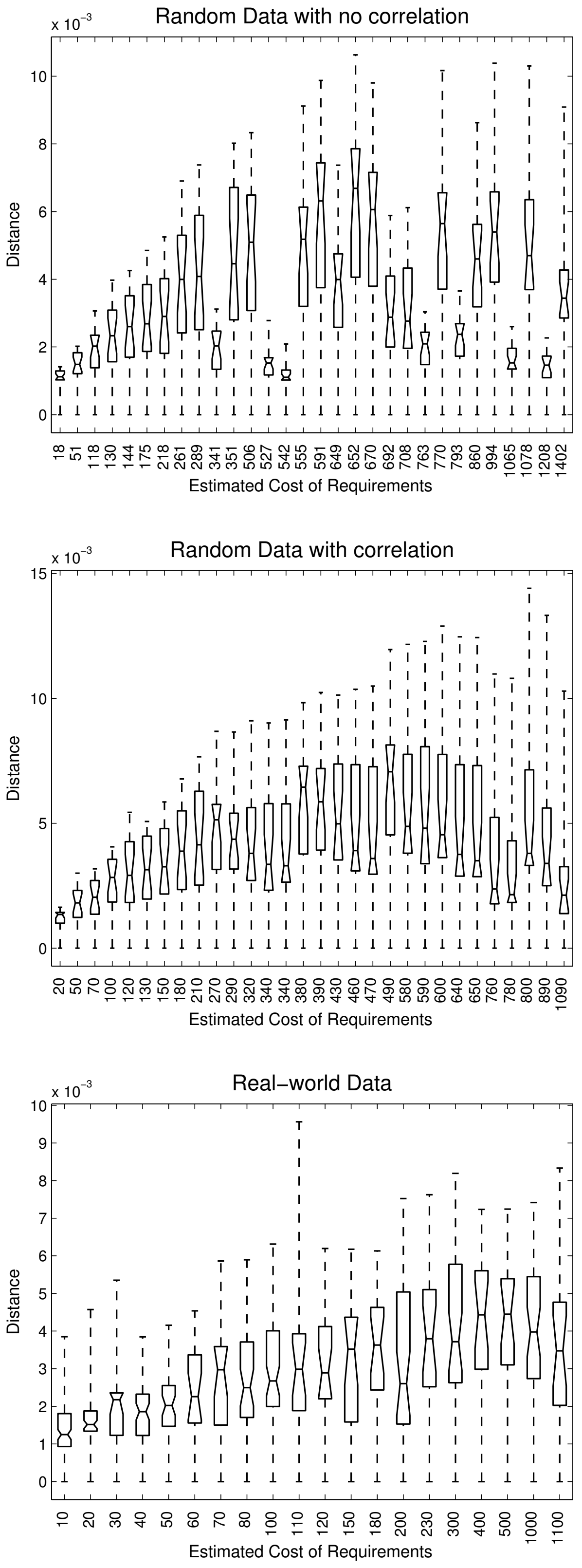


Figure 4.5: Boxplots of Euclidean distances between Pareto-fronts for different costs of requirements

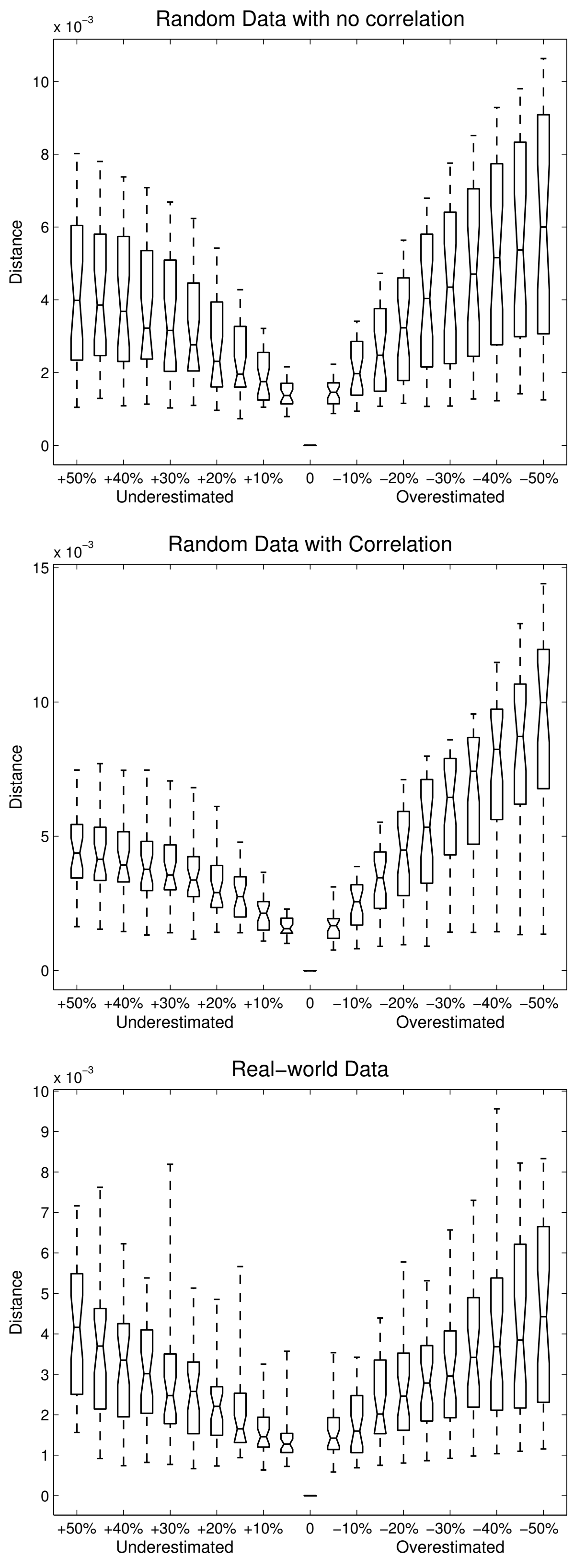


Figure 4.6: Boxplots of Euclidean distances between Pareto-fronts for different PIAC values

4.5.3 Statistical Analysis

Figure 4.5 and Figure 4.6 show the boxplots of Euclidean distances measured with different sets of data. Each boxplot in Figure 4.5 represents the Euclidean distances measured from all requirements that share the same value of development cost. Each boxplot in Figure 4.6 represents the Euclidean distances measured from all requirements in the data set for a specific PICA value. In both figures, the general trend is a positive correlation between Euclidean distance and PICA or cost, meaning that larger PICA values and larger development cost will have a greater impact on the result.

The random data set with no correlation between cost and revenue shows several unique data points that do not follow the overall trend. The position and number of these exceptions correspond to the exceptions observed in the corresponding heat-map in Figure 4.4. This implies that if the data set contains requirements that do not fit the Positive Correlation Assumption, there are likely to exist exceptional requirements. With the random data set with positive correlation between cost and revenue, the PCA trend is more consistent and smooth.

To test PCA statistically, Figure 4.5 and Figure 4.6 are statistically analysed using Spearman's rank correlation analysis. Spearman's rank correlation coefficient is used to quantitatively describe the relationship between two pairs of separate variables, without assuming any linear relation between them. First, correlation coefficient ρ is calculated using the Spearman's formula. Secondly, we compare the calculated ρ value with the critical value of ρ at the 0.05 significant level. If the calculated value exceeds the critical value, we can conclude that there is a strong correlation between the pair of variables, in which case it indicates that 95 times in 100, the monotonic relationship between two sets of variables occurred because a correlation exists, and not because of pure chance. Furthermore, the calculated p-value can also reflect the significance of the correlation. It represents the probability of there is no correlation between the two variables.

Table 4.1 shows the Spearman's rank correlation coefficient values between cost of requirements and Euclidean distance for the real-world data set for each PICA value. The observed ρ values show strong monotonic correlation between cost and

Table 4.1: Spearman's rank correlation coefficient between PICA value and Euclidean distance. For all requirements, the observed ρ values are statistically significant at the confidence level of 95%.

Req.	ρ_{PIAC}	p	*Req.*	ρ_{PIAC}	p
1	0.9474	0.0000	19	0.7318	0.0002
2	0.8591	0.0000	20	0.8929	0.0000
3	0.8825	0.0000	21	0.7188	0.0002
4	0.8591	0.0000	22	0.8591	0.0000
5	0.9604	0.0000	23	0.7786	0.0000
6	0.9630	0.0000	24	0.7890	0.0000
7	0.7942	0.0000	25	0.8721	0.0000
8	0.7890	0.0000	26	0.9136	0.0000
9	0.9032	0.0000	27	0.6929	0.0005
10	0.4487	0.0413	28	0.8617	0.0000
11	0.8643	0.0000	29	0.2071	0.3676
12	0.8773	0.0000	30	0.8877	0.0000
13	0.9266	0.0000	31	0.9578	0.0000
14	0.3188	0.1589	32	0.8123	0.0000
15	0.6305	0.0022	33	0.9162	0.0000
16	0.8981	0.0000	34	0.8331	0.0000
17	0.2461	0.2822	35	0.7838	0.0000
18	0.6929	0.0005			

Table 4.2: Spearman's rank correlation coefficient between cost and Euclidean distance. For all PICA values, the observed ρ values are statistically significant at the confidence level of 95%.

PICA	ρ_{cost}	p_{cost}	*PICA*	ρ_{cost}	p_{cost}
+50%	0.6648	0.0000	−50%	0.9246	0.0000
+45%	0.8204	0.0000	−45%	0.8918	0.0000
+40%	0.7386	0.0000	−40%	0.8912	0.0000
+35%	0.8093	0.0000	−35%	0.8940	0.0000
+30%	0.7104	0.0000	−30%	0.8646	0.0000
+25%	0.6194	0.0001	−25%	0.8520	0.0000
+20%	0.5704	0.0003	−20%	0.7560	0.0000
+15%	0.8155	0.0000	−15%	0.6370	0.0000
+10%	0.6256	0.0001	−10%	0.4645	0.0049
+5%	0.4239	0.0112	−5%	0.3600	0.0336
0%	0.5026	0.0021			

Euclidean distance, exceeding the critical value 0.008 at the significance level of 0.05 for sample size of 500. Again, this confirms the general trend predicted by PCA-1.

Similarly, Table 4.2 shows the correlation coefficient calculated for the relation between PICA values and Euclidean distance between two Pareto fronts for the real-world data set. The average coefficient, ρ, is 0.5871, which comfortably exceeds the critical value of 0.334 at the significance level of 0.05 for sample size of 35. This confirms the general trend predicted by PCA-2.

4.5.4 Answers to the Research Questions

RQ1 and **RQ2** are answered by the statistical analysis shown in Table 4.2 and Table 4.1. The Spearman's rank correlation coefficient confirms that there exists a positive correlation between the cost of each requirement and the impact, and between the level of inaccuracy and the impact. The correlation is statistically significant with confidence level of 95%.

However, it is the overall trends and the exceptions observed in Figure 4.4, Figure 4.5 and Figure 4.6 that would be of particular interest to the decision maker. First, while the PCA is statistically confirmed in general, there are exceptions to the trends. In Figure 4.4, the heat-map for the random data set with no correlation shows that the requirements that have relatively high and low impact factor form distinct horizontal bands. This phenomenon is weakened in the second heat-map for the random data set with correlation. Finally, the real world data shows much more complex patterns with very few distinct horizontal bands.

Comparing the first and the second heat-map, it can be said that the correlation between the cost and the expected revenue of requirements is an important factor in sensitivity analysis. More specifically, if it is likely that some requirements have high cost and low revenue, or vice versa, these requirements are more likely to contribute to create the sensitive region in NRP solution.

Figure 4.5 and Figure 4.6 also visually confirm PCA-1 and PCA-2 respectively. In Figure 4.5, we can observe unique boxplots with very small variance which correspond to the low-impact horizontal bands observed in the first heat-map in Figure 4.4. Another interesting observation found in Figure 4.6 is that overestimation tends to have a bigger impact on the solutions of NRP than underestimation; boxplots on the right side of Figure 4.6 shows steeper increase in mean values than those on the left side. This trend has a very interesting implication to practitioners, because under uncertainties, a human decision maker is more likely to overestimate than underestimate. This qualitative assessment of the statistical analysis forms the answer to **RQ3**.

4.6 Related work

In the Next Release Problem (NRP), the goal is to select an optimal subset of requirements for the next release of a product. Bagnall et al. first suggested the term NRP and applied various modern heuristics including greedy, hill climbers and simulated annealing algorithm [Bagnall et al., 2001]. Baker et al. [Baker et al., 2006] applied Search-Based Software Engineering approach to NRP by using single objective optimisation algorithms: the greedy algorithm and the simulated annealing algorithm. A variation of the problem using integer linear programming is studied in Van den Akker's work [van den Akker et al., 2008], to find exact solutions within budgetary constraints.

Zhang et al. [Zhang et al., 2007] introduced new formulations of Multi-objective Next Release Problem (MONRP). In Zhang's MO-NRP formulations, at least two parameters (possibly conflicting) are considered as two optimisation objectives simultaneously.

Sensitivity analysis has been widely applied in various areas including complex engineering system, environmental studies, economics, health care, etc. [Baniotopoulos, 1991,Christopher Frey and Patil, 2002,Gunawan et al., 2005,Levine and Renelt, 1992] It has been used as one of the principal quantitative techniques in risk management [Boehm et al., 2000]. It can be used to provide an insight into the reliability and robustness of a problem model result when making decisions [Saltelli et al., 2000]. However, the present work is the first to introduce Sensitivity Analysis in multi-objective optimisation problems in the area of software engineering.

The proof-of-principle study by Deb et al. [Deb and Gupta, 2005] introduced robust optimisation procedures to multi-objective optimisation problems for the purpose of searching for robust Pareto-optimal solutions in multi-objective optimisation problems.

4.7 Summary

The work introduces an SBSE approach to identify requirements that are anomaly sensitive to inaccurate cost estimation. Sensitive requirements are those that have significant impact on the final solution of NRP when their cost estimates are inaccurate. The work presents an automated sensitivity analysis approach based on SBSE for both single- and multi-objective NRP formulations. The results of the sensitivity analysis is summarised in an intuitive heat-map style visualisation to aid the decision maker in identifying sensitive regions in the data.

Through the empirical studies of both synthetic and real-world requirement data, the work presents a statistical analysis that confirms the Positive Correlation Assumption, that more expensive requirements and higher level of inaccuracies tend to have greater impact on NRP. However, the heat-map visualisation also reveals that there exist exceptions to this assumption. Identifying these exceptions can guide the decision maker towards more accurate estimation and safer decision making.

Chapter 5

Cooperative Co-evolutionary Job Sequencing and Team Sizing

5.1 Introduction

Software project management has been the subject of much recent work in the SBSE literature. Previous work has investigated the project staffing and planning problem either as a single-objective problem, or as a multi-objective problem in which the multiple objectives are, to some degree, conflicting objectives [Alba and Chicano, 2005, Alba and Chicano, 2007, Di Penta et al., 2011]. In this chapter we introduce an alternative approach based on the use of a Cooperative Co-Evolutionary Algorithm (CCEA). We believe that a Cooperative Co-Evolutionary approach to project management is attractive because it allows us to model a problem in terms of sub problems (e.g., in project scheduling and staffing, the allocation of work packages to teams and allocation of staff to teams). These subproblems can be inter-related, but separate problems, for which the overall solution depends on the identification of suitable sympathetic sub-solutions to each of the subproblems.

We show how the two primary features of a project plan—the allocation of staff to teams and the allocation of teams to work packages—can be formulated as two populations in a Cooperative Co-evolutionary search. Co-evolution has been previously used in SBSE work [Adamopoulos et al., 2004, Arcuri and Yao, 2008, Arcuri and Yao, 2010], but all previous approaches have used *competing* subpopulations; the so-called predator–prey model of Co-evolution. In this chapter, we adopt the alternative approach to co-evolution, Cooperative Co-evolution, in which the subpopulations work symbiotically rather than in conflict with one another. We believe that this form of co-evolution may also find many other applications in SBSE work, since many Software Engineering problems are characterized by a need to find cooperating subsystems that are evolved specifically to work together symbiotically.

We implemented our approach and evaluated it on data from four real world software projects from four different companies, ranging in sizes form 60 to 253 individual work packages. We reported the results on the efficiency and effectiveness of our approach, compared to a random search and to a single population approach. Our results indicate that the co-evolutionary approach has great promise; over 30 runs for each approach, co-evolution significantly outperforms both random and single population approaches for the effectiveness of the project plans found, while it also appears to be at least as efficient as a single population approach.

The work makes two primary contributions: (1) The work introduces a novel formulation of the Software Project Planning Problem using Cooperative Co-evolution and, to the best of our knowledge, this is the first work in the SBSE literature to use cooperative co-evolution. (2) The work reports the results of an empirical study with an implementation of our co-evolutionary approach, compared to random and single population evolution. The obtained results provide evidence to support the claim that cooperative co-evolution is more efficient and effective than single population evolution and random search.

5.2 Problem Statement and Definitions

This section describes the problem model for the work packages scheduling and staff assignment problem in detail and addresses the use of the CCEA.

Finding an optimal work package scheduling for a large project is difficult due to the large search space and many different considerations that need to be balanced. Also, finding an optimal way to construct its project teams is crucial as well. In this chapter, we focus on team construction with regards to team size.

In order to formulate this problem into a model, we make the following assumptions to simplify the problem: (1) staff members are identical in terms of skills and expertise, and staff only work on one team during the whole project, (2) WPs are sequentially distributed to teams, but they may still be processed at the same time, and (3) only one kind of dependency is considered: Finish-to-Start (FS). All three assumptions were found to be applicable to the four projects studied, all of which are real world software projects and therefore, though limiting, our assumptions do not preclude real world application.

5.2.1 Ordering/Sequence of Work Packages

To model the work needed to complete a project, we decompose the project according to its Work Breakdown Structure (WBS). WBS is widely used as a method of project decomposition. In a given WBS, the whole project is divided into a number of l small Work Packages (WPs): $\mathbf{WP} = \{wp_1, wp_2, \cdots, wp_l\}$. Two attributes of a WP, wp_i, are considered: (1) the estimated effort, e_i, required to complete wp_i, and (2) the WP predecessor(s), dep_i, which need to be completed before wp_i can start to be processed. The estimated efforts for all WPs are represented as a vector: $\mathbf{E} = \{e_1, e_2, \cdots, e_l\}$, e.g.: wp_i requires e_i person-days to complete; and dependence information is represented as a two-dimensional vector as: $\mathbf{Dep} = \{dep_1, dep_2, \cdots, dep_l\}$ where $dep_i = \{wp_j, \cdots, wp_k\}$ if the predecessors of wp_i are wp_j, $\cdots$, and wp_k.

The order in which the WPs are considered is represented as a string, shown in Figure 6.2, where the WP ordering in the string indicates a specific sequence for distributing the WPs to project teams. Constraints of precedence relationships are satisfied as each is processed, with the effect that a project cannot start until its dependent WPs have been completed.

Work Package Distributing Order:	$1st$	$2nd$	$3rd$	$\cdots$	$(l-1)th$	l-th
Work Package ID:	3	2	6	$\cdots$	l	$l-4$

Figure 5.1: WPO Chromosome: The gray area is the representation of the solutions for the ordering for distributing a set of l work packages. A solution is represented by a string of length l, each gene corresponding to the distributing order of the WPs and the alleles, drawn from $\{1, ..., l\}$, representing an individual WP.

5.2.2 Staff Assignments to Teams

A total of n staff are assigned to m teams to execute the WPs. The size of each team (their ‘capacity’) is denoted by a sequence $\mathbf{C} = \{c_1, c_2, \cdots, c_m\}$.

Staff:	S_1	S_2	S_3	$\cdots$	S_{n-1}	S_n
Assigned To Team No.:	2	4	3	$\cdots$	m	3

Figure 5.2: TC Chromosome: The gray area is the representation of the solutions for Team Construction or the assignments of a set of n staff to a set of m teams. A solution is represented by a string of length n, with each gene corresponding to a staff and the alleles, drawn from $\{1, ..., m\}$, representing assignment of the staff.

5.2.3 Scheduling Simulation

We use a single objective fitness evaluation for both populations in our co-evolutionary approach, i.e., the project completion time. The processing of the WPs by the teams is simulated by a simple queuing simulation as described in previous work [Di Penta et al., 2011, Di Penta et al., 2007]. In this approach, the WP dependence constraints are satisfied by arranging the order in which WPs are assigned to teams. However, we want to avoid the order in which the successor wp_i is right after its predecessor wp_j. In such a case, wp_i has to wait until wp_j is finished before it can be distributed to an available team. There are two ways for the managers to minimize the team's unused available time: 1) interlacing: managers can choose to insert one or more WPs between wp_i and wp_j so when wp_i is waiting for wp_j those inserted WPs can keep all the teams functioning, or 2) using mitigation: distribute the predecessor wp_j to a team with the highest possible capacity, so that the completion time of the predecessor is the shortest, and therefore, the waiting time of wp_i is mitigated to be the shortest one. In our case, we simply rely on the search based algorithm that, by producing different WP orderings, can enact both interlacing or mitigation. Further details about the simulation of WP scheduling can be found in a previous work [Di Penta et al., 2011].

5.3 Optimisation Method: Cooperative Co-evolutionary Algorithm

The Cooperative Co-Evolution Algorithm (CCEA) [Potter and Jong, 1994] was proposed to solve large and complex problems by implementing a divide-and-conquer strategy. CCEA was originally designed to decompose a high-dimensional problem into smaller sub-problems that could be handled by conventional evolutionary algorithms [Yang et al., 2008b]. Using CCEA, the individuals from each population represent a sub-solution to the given problem. To search for a solution, the members of each population are evolved independently, and interactions between populations only occur to obtain fitness.

5.3.1 Solution Representations and Genetic Operators

There are two species of solutions in this evolutionary process: One (WPO) contains solutions representing the ordering in which WPs are distributed to teams, and the other (TC) represents the Team Constructions, i.e., the number of staffing persons for each team.

For solutions representing staff assignments or TC, as shown in Figure 5.2, we encoded the solutions in the following format. The assignment of a set of n staff to a set of m teams is represented as a string of length n. Each gene of the chromosome corresponds to a staff member. The alleles ranging from 1 to m represent the team that the staff is assigned to. A single-point crossover is performed for the TC species. Basically, the offspring takes half of the chromosome from each of both parents as shown in Figure 5.3. The mutation operator is thus set to assign each staff randomly to another team.

To achieve a fair comparison between projects, we chose the number of staff $n = 50$ and the number of working teams $m = 5$. Also, we verify every new generated solution of TCs before evaluating it with the fitness function, to ensure in each solutions every team has at least one staff member. This “no empty team” check is required because a team can be empty during the evolutionary process if all staff members are assigned to other teams.

The representation of WP ordering is shown in Figure 6.2. The crossover op-

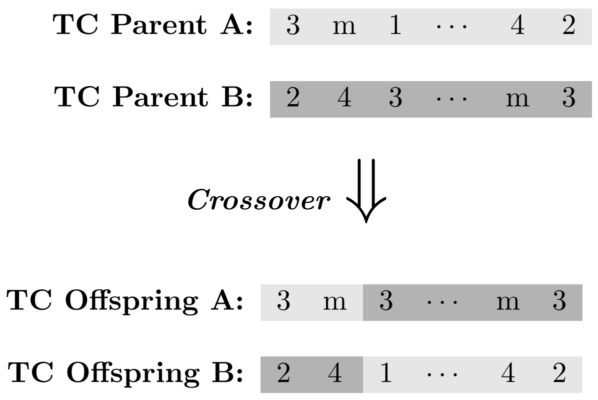

TC Parent A: 3 m 1 ··· 4 2
TC Parent B: 2 4 3 ··· m 3
Crossover
TC Offspring A: 3 m 3 ··· m 3
TC Offspring B: 2 4 1 ··· 4 2

Figure 5.3: Crossover for Team Construction solutions: Single Point Crossover

erator for such a representation is explained in [Di Penta et al., 2011], while the mutation operator randomly swaps a WP to another position in the queue, as shown in Figure 5.4. The mutation rate of is 20% per gene, calibrated after trying other (higher and lower) rates.

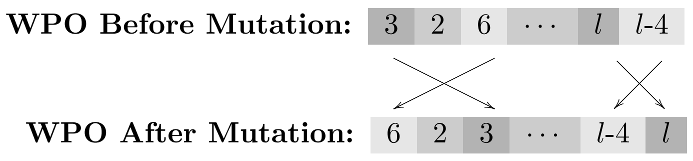

WPO Before Mutation: 3 2 6 ··· l l-4
WPO After Mutation: 6 2 3 ··· l-4 l

Figure 5.4: Mutation on Work Package Ordering solution: Randomly Swap WPs' Positions

To satisfy the dependency constraints among WPs, a dependency check is required. Solutions that violate the dependency constraints are "repaired" as explained in Section 5.2.3.

5.3.2 Initial Populations

The initial populations are randomly generated and subject to satisfy the "no empty team" rule and dependency constraints among WPs. The population size is set to 50 for both species.

5.3.3 Termination Condition

We experimented with 3 sets of configurations formed of the Internal (I) and External (E) number of generations for CCEA. The number of internal generations relates to the evolution of each sub-population, while the number of external generations represents how many times each population provides an updated reference to help the other population co-evolve. As shown in Table 5.1, these 3 configurations are all allowed the same budget of fitness evaluations. Although all these three configurations evolve solutions in both TC and WPO populations for the same number of generations, the level of communication between the two populations varies.

Table 5.1: Three sets of configurations for CCEA each of which requires the same total number of evaluations before it is terminated. F represents the number of evaluation required in one generation, and it is fixed for all configurations in this empirical study.

Config.	*# of Internal Generations*	*# of External Generations*	*Total # of Fitness Evaluations*
I	1	100	100*F
II	10	10	100*F
III	100	1	100*F

For instance, with Config. I, CCEA evolves solutions in both populations for a single generation only (internal generation) and then provides the updated individuals for fitness evaluations of the other population. Finally, before the evolution process terminates, the TC population provides an updated reference for the WPO population for a total of 100 iterations (external generation), and vice versa. Config. III is not a CCEA by definition because during the whole period of the cooperative co-evolutionary process, the communication happens only once. Therefore Config. III is a 'non-co-evolutionary' approach, against which we compare the other (co-evolutionary) approaches. By implementing the non co-evolutionary approach as a 'special case' (by suitable choice of parameters), we remove one source of possible bias that would otherwise result from experimenting with two totally different implementations: one co-evolutionary and the other not.

5.4 Empirical Study

The goal of this empirical study is to compare our new CCEA approach with a non-co-evolutionary genetic algorithm and a random search. We study the effectiveness and efficiency of our approach, alternatives and (for purely 'sanity check' purposes) random search on four industrial projects, named *Projects A, B, C and D*, described below and for which quantitative data are summarized in Table 5.2.

Table 5.2: Characteristics of the four industrial projects

Projects	*#WPs*	*#Dependencies*	*Total Effort (person-days)*
A	84	0	4287
B	120	102	594
C	253	226	833
D	60	57	68

More detailed descriptions of these four projects can be found in Section 3.2 along with the details of all the other industrial data sets used in this thesis.

The empirical study addresses the following research questions:

RQ1: (Sanity Check) Do the CCEA approach and the single population alternative significantly outperform random search?

RQ1.1: Does the single population GA outperform random search?

RQ1.2: Do the CCEAs outperform random search?

RQ2: (Effectiveness) How effective is the CCEA approach compared to the alternatives in terms of finding an earlier completion time?

RQ3: (Efficiency) Given the same number of evaluations, which algorithm finds the best-so-far solution the quickest?

The population size in our implementation was set to 100 and the number of generations is listed in Table 5.1 for the three different configurations. For the CCEA, the fitness of an individual in either species depends on the results of its simulations with 6 individuals from the other species. Offspring compete with their parents, and, to select parents for reproduction, the algorithm randomly picks a number of parents and performs a tournament selection to identify parents for breeding. The pool of offspring is half the population size, i.e., 25. That is, the best 25 offspring had the chance to compete with their parents.

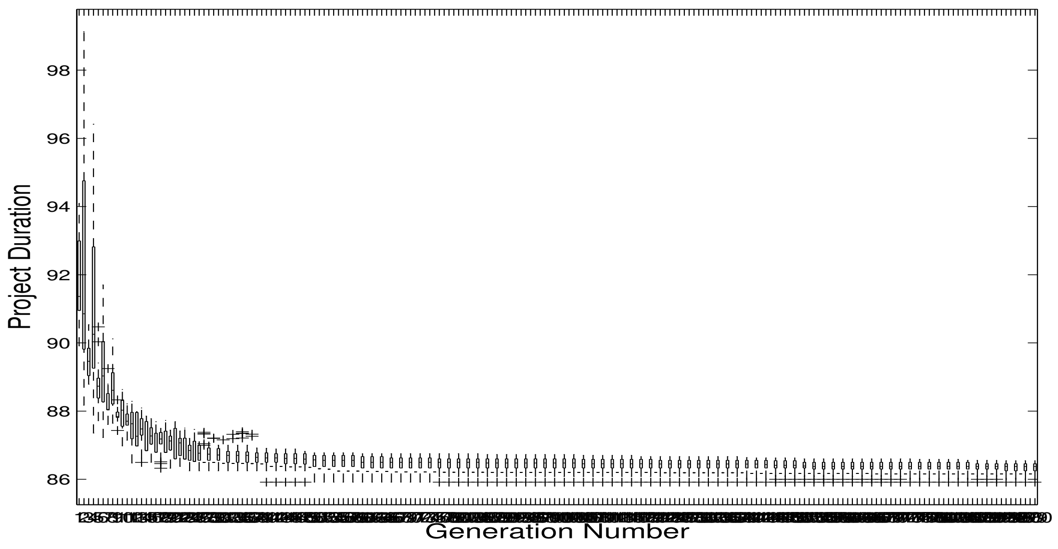
5.5 Empirical Study Results

In this section, we report results of the study described in Section 5.4.

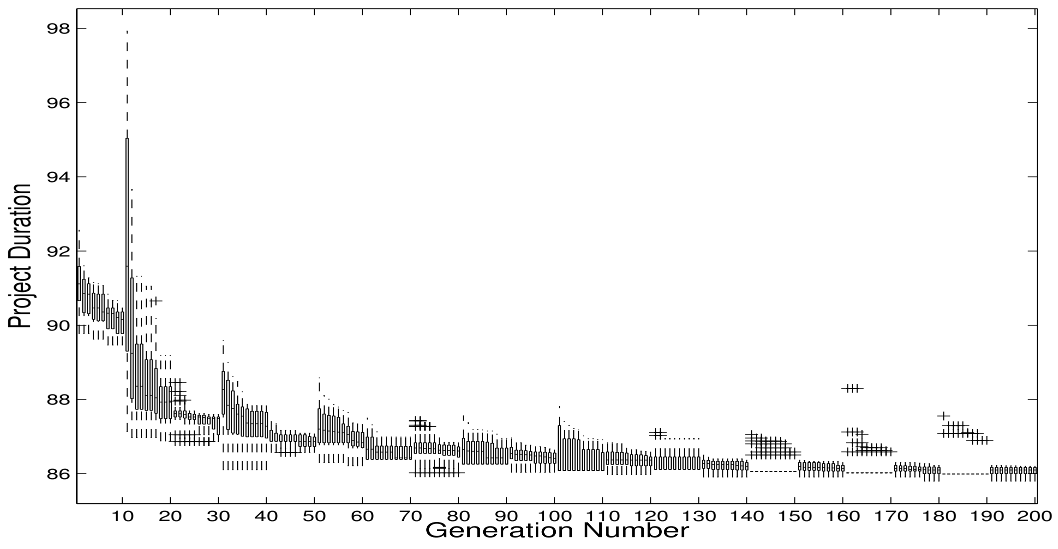
5.5.1 Analysis of the Cooperative Co-Evolutionary Progress

Results for all four projects and for the 3 CCEA configurations are plotted on Figures 5.5 to 5.8. In each sub-figure, the tick labels on the horizontal axis indicate the total number of internal generations that have been carried out, and also indicates the point at which the algorithm updated the population used for fitness computation. At each point on the horizontal axis, the entire population is depicted using a boxplot to give both a sense of the values obtained for completion time as the evolution progresses and the distribution of the fitness values in the population.

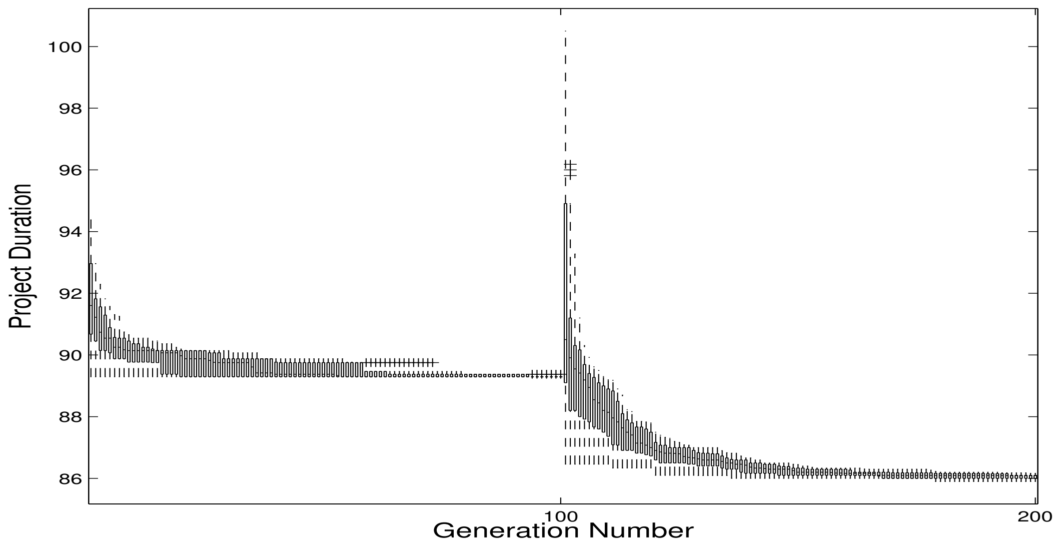
The fitness values of the entire population during the whole CCEA process are represented as boxplots. We can observe that the number of internal generations is the same for both populations within a specific sub-figure, as they fill equally spread vertical bands on the sub-figures. As can be seen from the sub-figures in the top rows in Figures 5.5 to 5.8, Config. I tends to find better solutions sooner than the other two configurations. On the second rows, we can observe an noticeable interlacing of the co-evolutions between the two populations. For instance, as in Figure 5.5(b), the evolutionary process on each population takes 10 internal generations. The first 10 internal generations—plotted within the interval [1, 10] on the horizontal axis—record the evolutionary progress of TC. After the first round of evolution on TC, the solutions on WPO start evolving during the interval [11, 20]. On the third row of the sub-figures, it can be seen that the optimisation of WPO produced more benefit—in terms of project completion time—than what was done for TC. This can be noticed in Figure 5.5(c), where both species evolved for only one round (i.e., one external generation) and, in each round, they evolved for 100 generations (i.e., 100 internal generations). As indicated by the generation number on the horizontal axis, results from the TC species are plotted on the interval [1, 100], while results from the WPO species are plotted on the interval [101, 200]. As we can see from the completion time (vertical axis) the optimisation of WPO leads to a noticeable improvement in the fitness function values obtained.



(a) Project A, Config. I

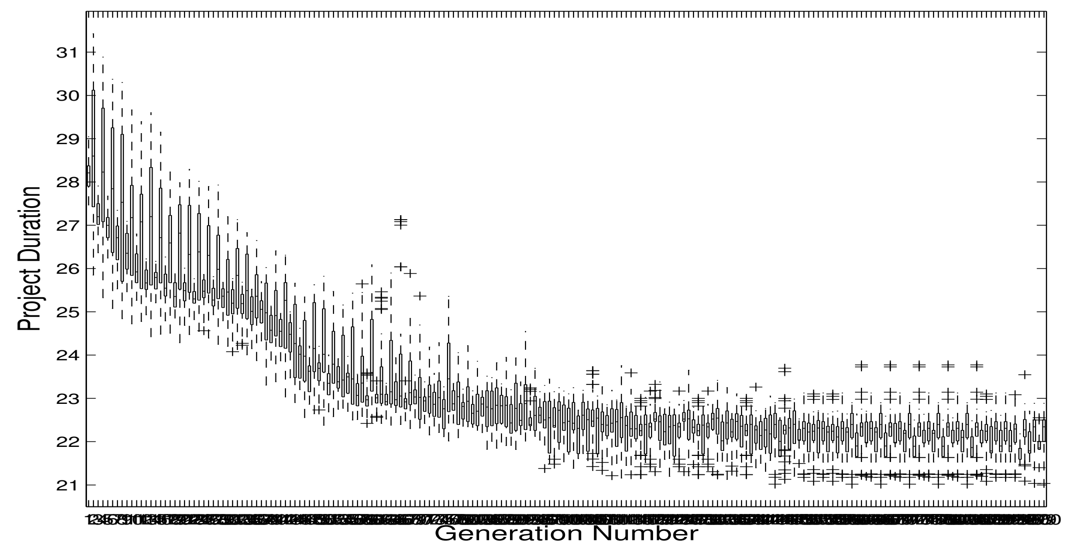


(b) Project A, Config. II

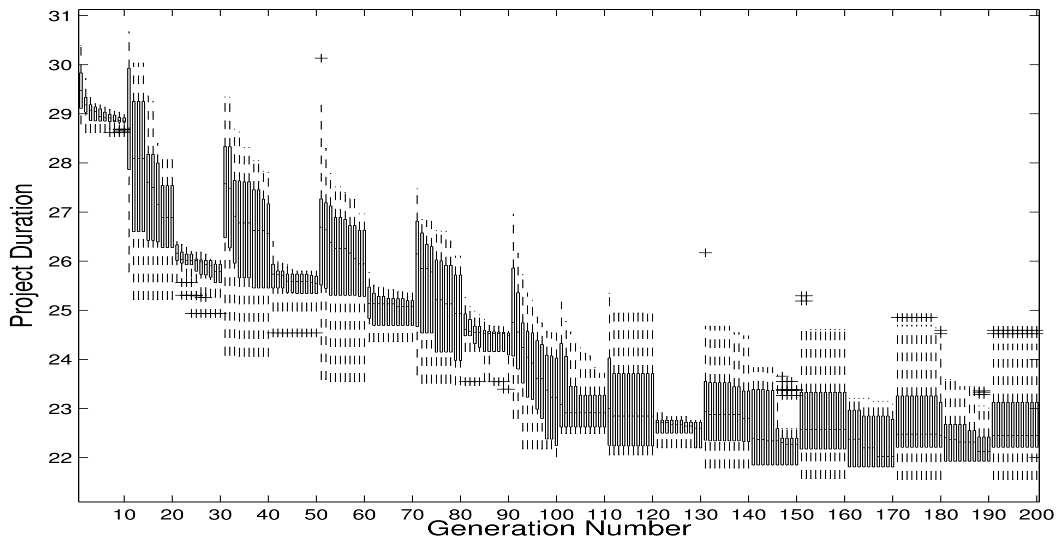


(c) Project A, Config. III

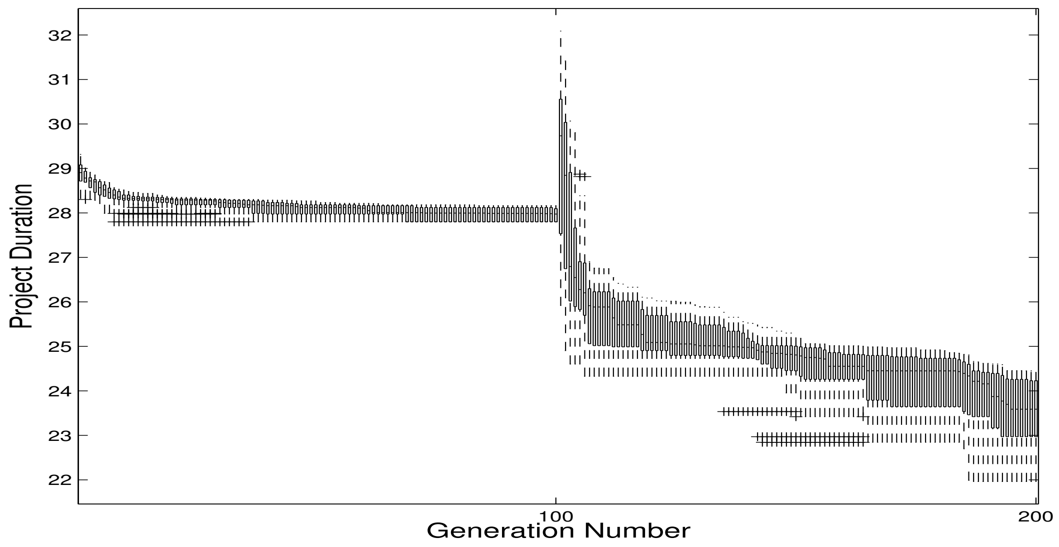
Figure 5.5: Projects A: Boxplots of completion times for all solutions found by different CCEAs configurations



(a) Project B, Config. I

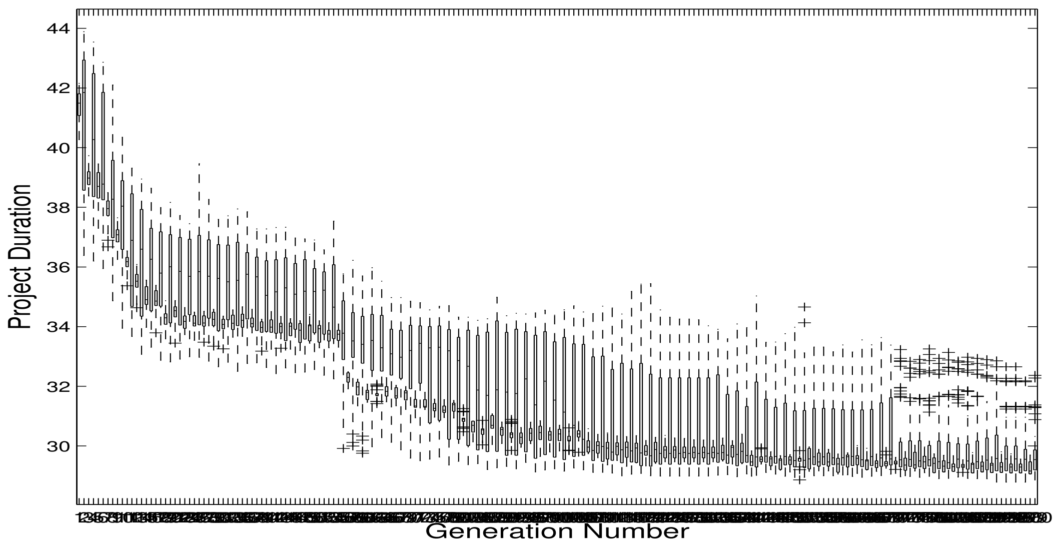


(b) Project B, Config. II

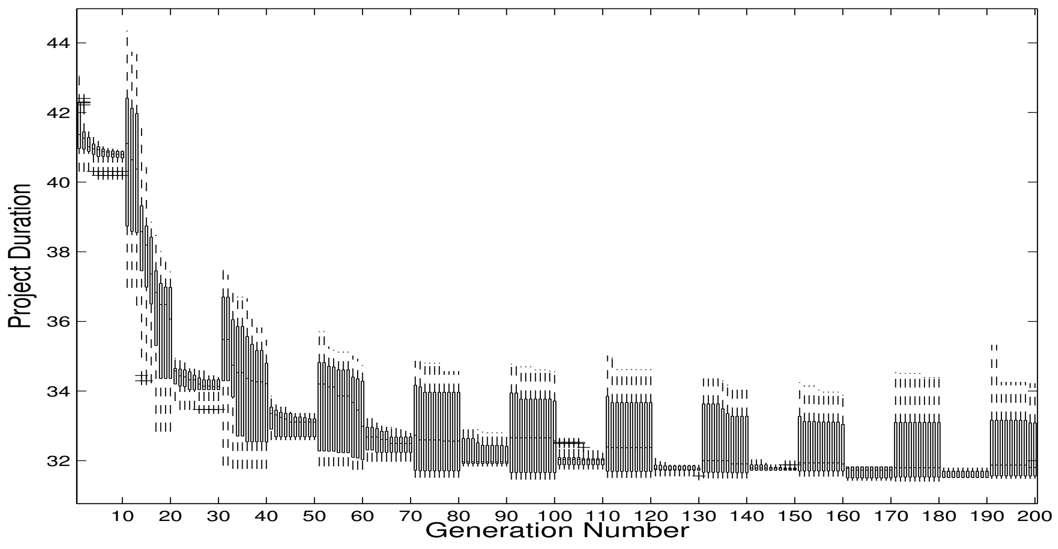


(c) Project B, Config. III

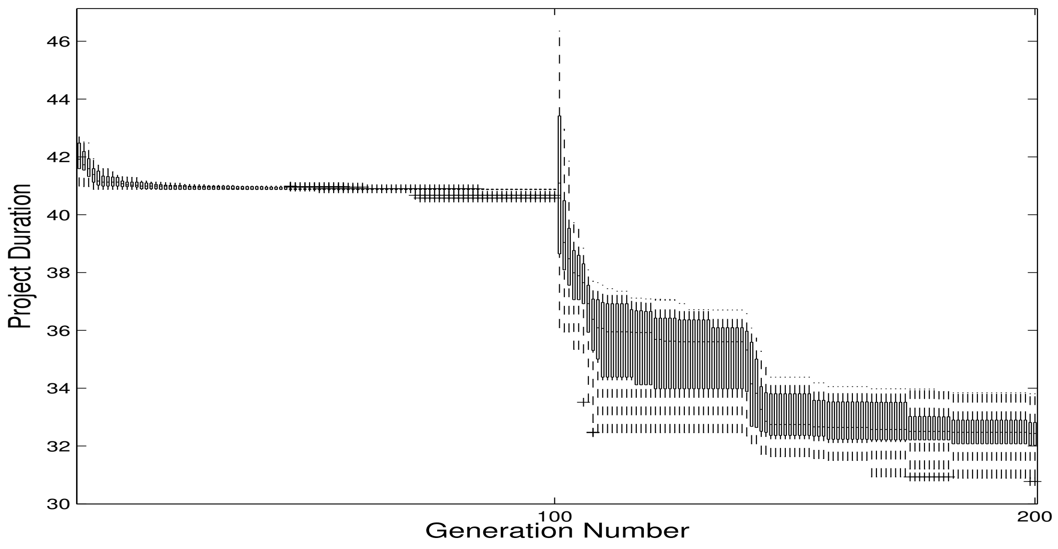
Figure 5.6: Projects B: Boxplots of completion times for all solutions found by different CCEAs configurations



(a) Project C, Config. I

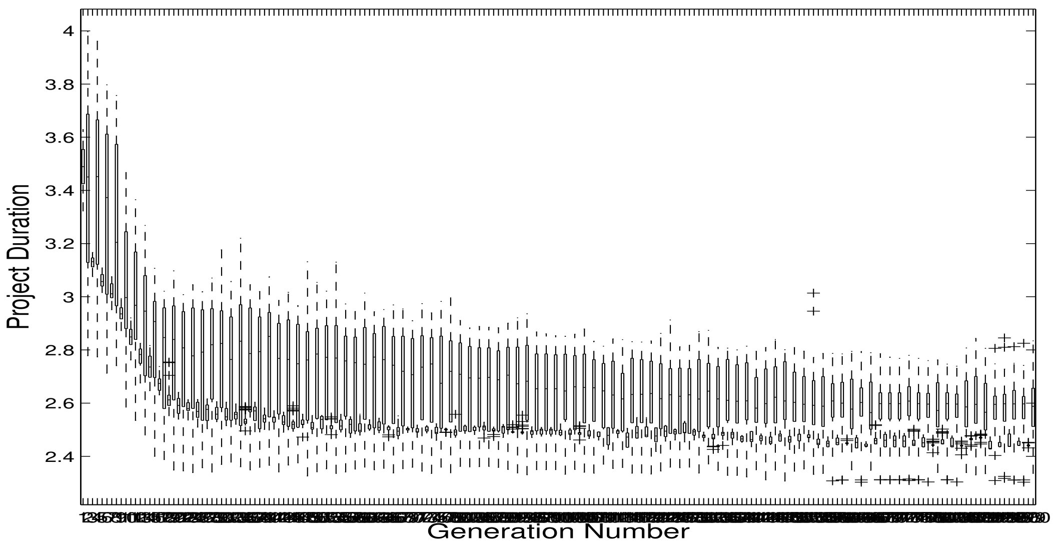


(b) Project C, Config. II

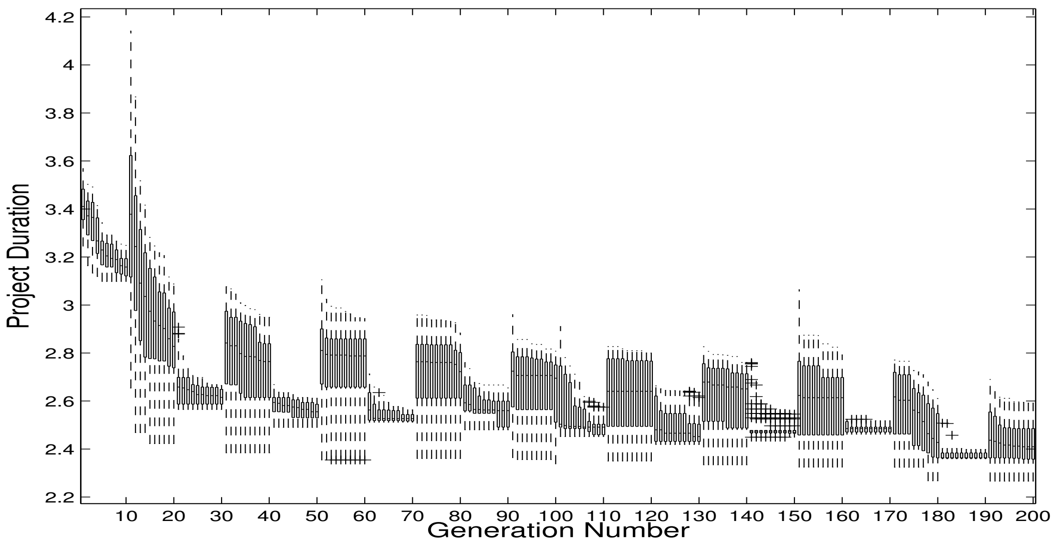


(c) Project C, Config. III

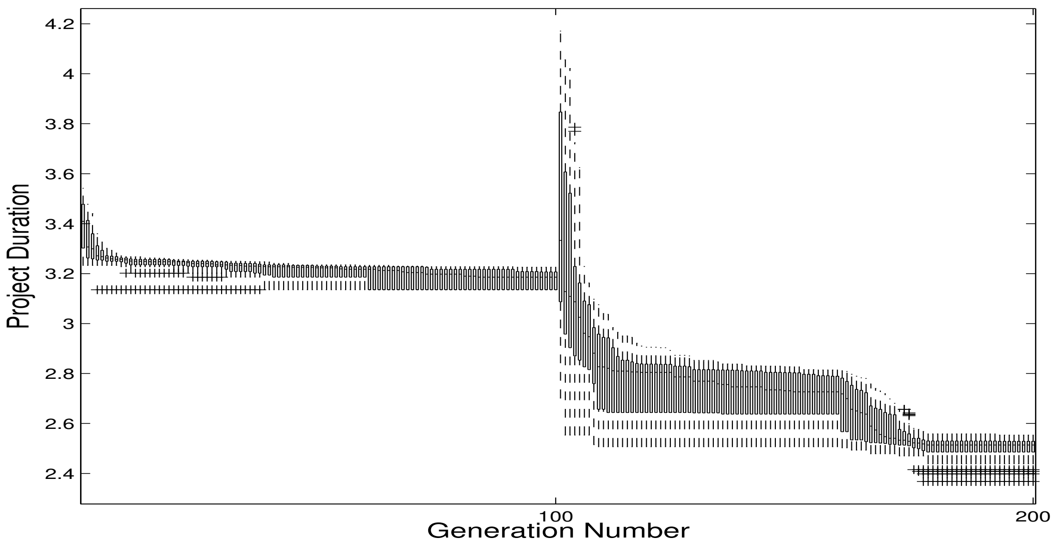
Figure 5.7: Projects C: Boxplots of completion times for all solutions found by different CCEAs configurations



(a) Project D, Config. I



(b) Project D, Config. II



(c) Project D, Config. III

Figure 5.8: Projects D: Boxplots of completion times for all solutions found by different CCEAs configurations

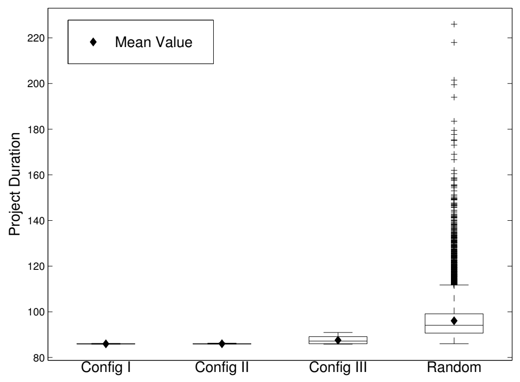
5.5.2 Results on Effectiveness

In this section we report the comparison of the effectiveness of three sets of CCEA configuration and the random search. Each algorithm was run 30 times on each of the 4 sets of project data to allow for statistical evaluation and comparison of the results.

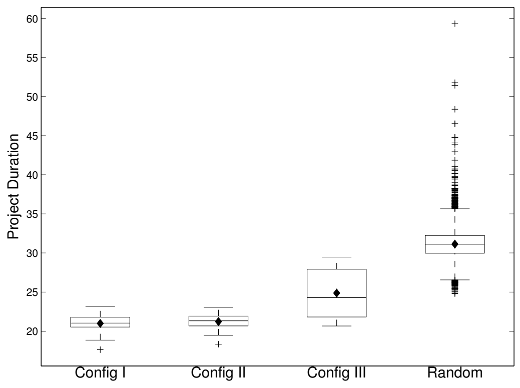
Figure 5.9 reports—for the various configurations—fitness values for the best individual solutions found by CCEA in the 30 runs.

As shown in the figures for Projects B, C, and D, in terms of the ability to effectively find the best solutions, CCEA performs better with Config. I and worse with Config. III. As explained in Section 5.3.3, Config. III is the single population evolutionary algorithm, while Config. I and II are bona fidè CCEAs. Therefore, our results provide evidence to support the claim that CCEAs outperform the single population evolutionary algorithm.

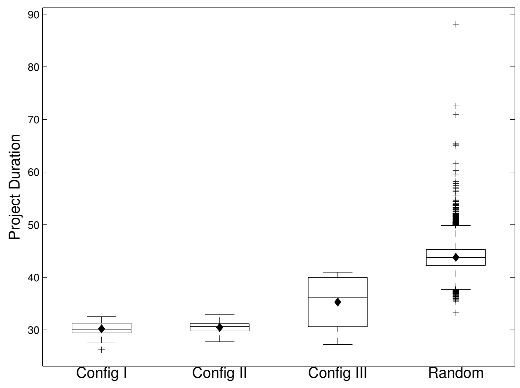
The random search generates twice the number of solutions of the CCEAs during the evolutionary process, and despite that, is clearly outperformed by the CCEAs in terms of fitness function quality. This observation is supported by a Wilcoxon Rank Sum Test (WRST) performed to calculate the statistical significance of the difference between the solutions produced by the different CCEAs configurations and Random. Since we are performing multiple comparisons on the same data set, *p-values* are corrected using the Holm's correction. The Wilcoxon Rank Sum Test (WRST) *p-values* reported in Table 5.3, as well as boxplots shown in Figure 5.9, indicate that all evolutionary algorithms perform significantly better than a random search, and that the best solutions found by the CCEAs (Config. I and II) perform significantly better than the single population evolutionary algorithm (Config. III).



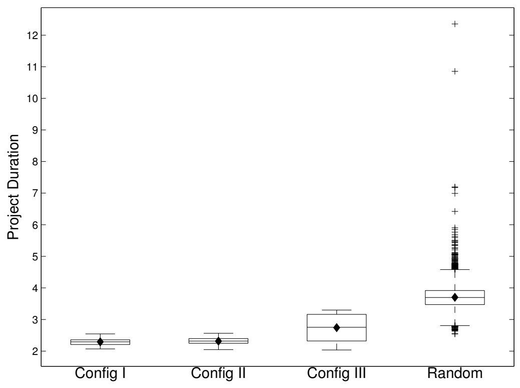
(a) Project A



(b) Project B



(c) Project C



(d) Project D

Figure 5.9: Boxplots of all the best solutions found in 30 runs of the three CCEA configurations, and in random search runs

Table 5.3: Wilcoxon Rank Sum Test (unpaired) test adjusted p-values for the pairwise comparison of the three configurations

p-values for WRST	Projects			
	A	B	C	D
Config. I vs II	0.7229	0.1885	0.4481	0.2449
Config. I vs III	5.04E-08	3.00E-11	2.78E-07	2.19E-07
Config. II vs III	1.47E-07	8.86E-10	2.08E-06	1.28E-06
Config. I vs Random	3.97E-40	3.82E-40	3.83E-40	3.83E-40
Config. II vs Random	3.97E-40	3.82E-40	3.83E-40	3.83E-40
Config. III vs Random	2.70E-30	6.04E-37	3.13E-36	3.66E-36

For Project A, while all CCEAs perform significantly better than random search, the practical benefit in terms of lower project completion time achieved is not as evident as for the other projects. This is because in Project A there are no dependencies between WPs; the project is a conceptually simple, multiple application of a massive maintenance task (fixing Y2K problem repeatedly using a windowing approach). Since there are no dependencies, there is no delay introduced by the need for waiting on dependent WPs. For this reason, the WP scheduling and team construction have little impact on the overall completion time.

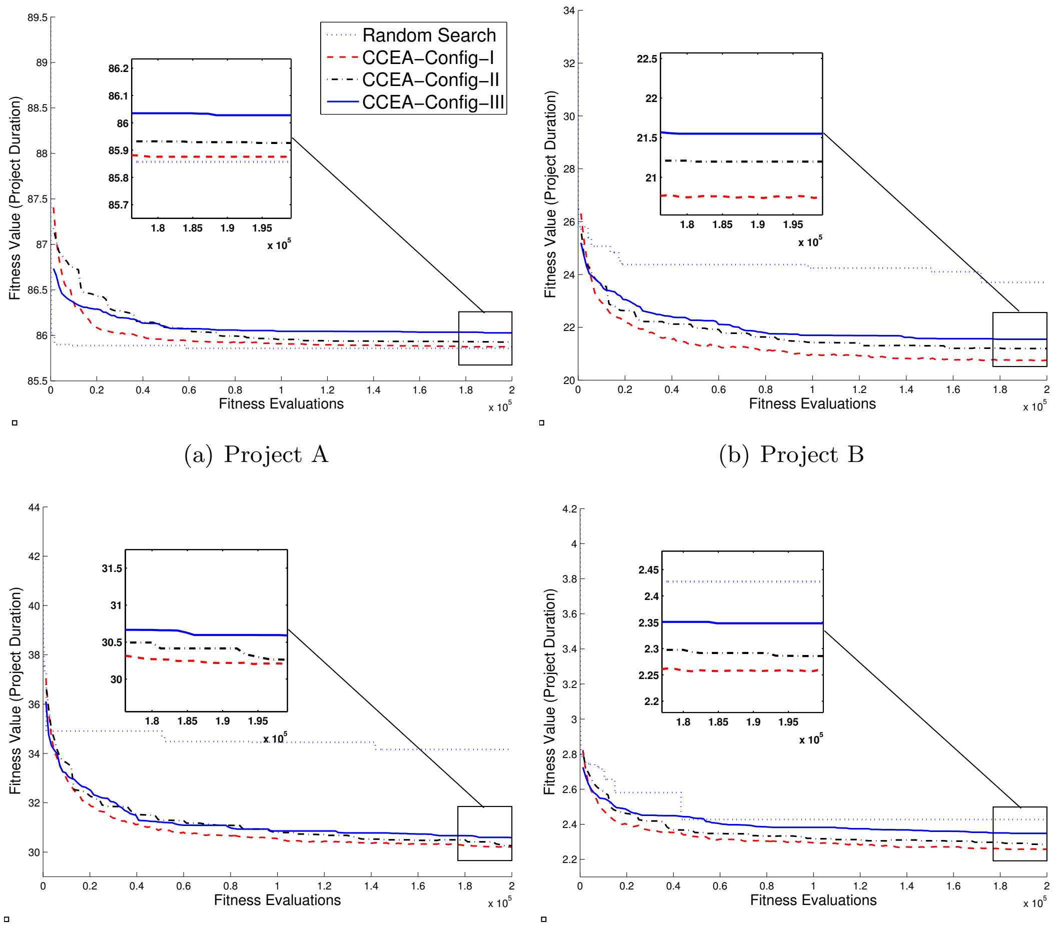
In conclusion, the obtained results support the following two claims: (1) all three CCEAs were found to perform better than the random search, which means the CCEAs passed the 'sanity check' set by **RQ1**, and (2) **RQ2** is answered with the result of the WRST test that indicated the best solutions found by Config. I and II are significantly better than those found by Config. III. We conclude that there is evidence to suggest that co-evolution is effective to deal with software project staffing and scheduling.

5.5.3 Results on Efficiency

To answer **RQ3**, we extended the experiments with 30 runs of 3 CCEAs configurations until the solutions produced by all algorithms became stable, and, to allow a fair comparison, the random search was set to have the same number of fitness evaluations. The progress of the CCEAs and the random search in finding better solutions are plotted in Figure 5.10. The fitness values are averaged over 30 runs for CCEAs, while for the random search, the figure shows the best solutions found for the number of evaluations indicated on the horizontal axis.

As shown in Figures 5.10(b), 5.10(c), and 5.10(d), respectively for Projects B, C and D, in most cases, the CCEAs find better solutions than the non-cooperative algorithm. However, there is an exception found for Project A as shown in Figure 5.10(a), for which the CCEA does not outperform its rivals. We believe that this is due to the dependence-free nature of Project A (as mentioned before, it has no dependencies).

In conclusion, with regard to the efficiency of finding better solutions (**RQ3**), we find evidence that CCEAs outperform random search in general, and that the CCEA



(a) Project A (b) Project B

(c) Project C (d) Project D

Figure 5.10: Efficiency Comparison of the Random Search and CCEAs

with more frequent communication between two populations (Config. I) performs better than the others (Config. II, III, and Random).

5.5.4 Threats to Validity

Construct validity threats may be due to the simplifications made when modelling the development/maintenance process through a simulation. In particular, (i) we assumed communication overhead negligible and (ii) we did not consider developers' expertise. However, accounting for these variables was out of scope for this work, as here the intent was to compare CCEA with non-co-evolutionary genetic algorithms.

Threats to *internal validity* can be due, in this study, to the bias introduced in our results by the intrinsic randomness of GA and, of course, of the random approach. We mitigate such a threat by performing statistical tests on the results.

Threats to *conclusion validity* concern the relationship between treatment and outcome. Wherever possible, we use appropriate statistics—Wilcoxon test with Holm's correction in particular—to robustly test our conclusions.

Threats to *external validity* concern the generalization of our findings. We performed experiments on data from four industrial projects having different characteristics in terms of size, domain, and relationships among WPs. However, further studies are desirable to corroborate the obtained results.

5.6 Related Work

Chao *et al.* were the first to publish on search-based project planning [Chao et al., 1993], with their introduction of the Software Project Management Net (SPMNet) approach for project scheduling and resource allocation, which was evaluated on simulated project data. Aguilar-Ruiz *et al.* [Aguilar-Ruiz et al., 2002] also presented early results on evolutionary optimisation for search-based project planning, once again evaluated on synthetically created software project data. Chicano and Alba [Alba and Chicano, 2005, Alba and Chicano, 2007] applied search algorithms to software projects to seek to find allocations that achieve earliest completion time. Alvarez-Valdes *et al.* [Alvarez-Valdes et al., 2006] applied a scatter search approach to the problem of minimizing project completion duration.

Project management has recently [de Souza et al., 2010] been the subject of a study of the human-competitiveness of SBSE, which found that optimisation techniques are able to produce effective results in a shorter time than human decision makers. This work demonstrates that SBSE is a suitable approach to consider for project planning activities since it can find solutions that the human decision maker may otherwise miss. While the ultimate decision is likely to rest with the human decision maker, it is therefore important to find suitable SBSE techniques that can support this decision making activity.

Other authors have also worked on SBSE as a means of decision support for software engineering managers and decision makers in the planning stages of software engineering projects focusing on early lifecycle planning [Barreto et al., 2008, Cortellessa et al., 2008, Kapur et al., 2008, Kremmel et al., 2011, Xiao et al., 2010] as we do in the present chapter, but also reaching forward to subsequent aspects of the software engineering lifecycle that also require planning, such as scheduling of bug fixing tasks [Fernando Netto and Alvim, 2009, Xiao and Afzal, 2010]. Like our present work, some of this work has considered multiple objectives [Alba and Chicano, 2007, Gueorguiev et al., 2009]. This is very natural in software project planning which is typified by many different concerns, each of which must be balanced against the others; and issue that is reported to be inherently as part of much work on SBSE [Harman, 2007]. However, no previous work has used co-evolution for project planning to simultaneously pursue these different objectives.

SBSE can also be used as a way to analyse and understand Software Project Planning, yielding insight into planning issues, rather than seeking to necessarily provide a specific 'best' project plan [Harman, 2010a]. For example, SBSE has been used to study the effect of Brooks' law [Brooks, Jr., 1975] on project planning [Di Penta et al., 2007]. It has also been used to balance the competing concerns of risk and completion time [Gueorguiev et al., 2009]. Our work may be used in this way, since we can study the way the two populations evolve with respect to one another and the ways in which they are symbiotic. A thorough exploration of this possibility remains a topic for future work.

Di Penta *et al.* [Di Penta et al., 2011] compared the performance of different search-based optimisation techniques, namely Genetic Algorithms, Simulated Annealing, and Hill Climbing to perform project planning on data from two industrial projects (Projects A and B also used in this study). The present work can be thought of as an extension of the previous work of Di Penta *et al.*, because it uses the same representation and fitness function, while proposing and evaluating the use of a completely unexplored optimisation approach: co-evolutionary optimisation.

5.7 Summary

This chapter proposes the use of Cooperative Co-Evolutionary Algorithms (CCEA) to solve software project planning and staffing problems. The co-evolutionary algorithm evolves two populations, one representing WP ordering in a queue (which determines their assignment to teams), and the other representing developers distribution among teams.

We conducted an empirical study using data from four industrial software projects, aimed at comparing CCEA project planning and staffing with (i) random search and (ii) single population optimisation using genetic algorithms. Results of the empirical study show that CCEA is able to outperform random search and single population GA, in terms of effectiveness (i.e., best solutions proposed in terms of project completion time) and efficiency (i.e., a smaller number of evaluations required).

Chapter 6

Co-evolutionary Project Planning Optimisation under Staff Absence

6.1 Introduction

Most computer-based project management optimisation techniques either simply assume full attendance from all employees [Alba and Chicano, 2007, Chang et al., 2001] or implicitly inject a certain ratio to simulate the staff absence [Berman and Larson, 1993, Hur et al., 2004, Zhu et al., 2005]. However, in real world cases, this assumption can hardly be true due to the uncertainties caused by staff turnover and absenteeism.

The uncertainties of staff attendance at the project planning stage lead to expensive operational costs to recover the scheduled project plan, and the wellness of recovery purely relies on the project manager's experiences and available resources at the time when the decision was made, which is lacking support from technical insights and subject to the constraints of available resources. However, the available resources at any given point during the course of a project were designed or planned under the assumption of full attendance of staff. Therefore, to utilise any of the available resources to mitigate the unplanned absence will cause inevitable disruptions that prevent further progress on the project to be made as scheduled/planned.

As a matter of fact, employee absenteeism has been identified as one of the most costly disruptions in project management [CIPD, 2012, Taylor et al., 2010]. Significant efforts have been made to reduce the employee absence from the perspectives of policy-making or in psychology, such as "Return to work interviews" and "Attendance bonus" [CBI, 2011]. Previous techniques can only be utilised during the course of the project, although they are often referred as "proactive" tools. On the other hand, we believe it is vitally important that project managers are provided with an automatic tool for simulating and analysing the uncertainty of employees' absence in passive manners. The automatically generated results can help project managers understand certain interesting "what-if" scenarios on staff absence and work package scheduling well in advance. Thus, project managers can decide whether or not to take certain precautions that are going to be effective. For example, in some projects, it will be very useful to the project manager if she/he could understand in advance that even though there are only a small amount of staff absences, its impact on project duration cannot be mitigated by rescheduling the work packages.

The harmful effects of absence on productivity are well documented [Hausknecht et al., 2008]. According to a recent survey by the Confederation of British Industry (CBI) [CBI, 2011] nearly 190 million days were lost to staff absence. The direct costs of absence alone amounted to over £17 billion across the UK economy in 2010, and the median total cost for each absent employee in 2010 was £760 a year. Over two thirds (68%) of all working time lost to employee absence is attributable to short-term conditions. More than a third of employers have set an explicit target for reducing absences over the coming year. A review of the health of the working age population by the Department of Work and Pensions [Black, 2008] reports that the economic costs of sickness absence and worklessness due to ill health amounted to over £100 billion a year, which is greater than the annual budget of the National Health Service (NHS) [Higgins et al., 2012]. There is also analysis on sickness absence rates by age, gender and other socio-economic characteristics of workers. These relationships prove to be similar across countries with widely differing mean rates of absence [Barmby et al., 2002].

The different absence rates in Figure 6.1 from [Black, 2008] shows different industrial absence rate reported by CBI, the Chartered Institute of Personnel and Development (CIPD), and the Office for National Statistics (ONS) through its Labour Force Survey (LFS). This is due to the difference in the population sampled, response rate, and, crucially, data collected. The complexity of differences between surveys

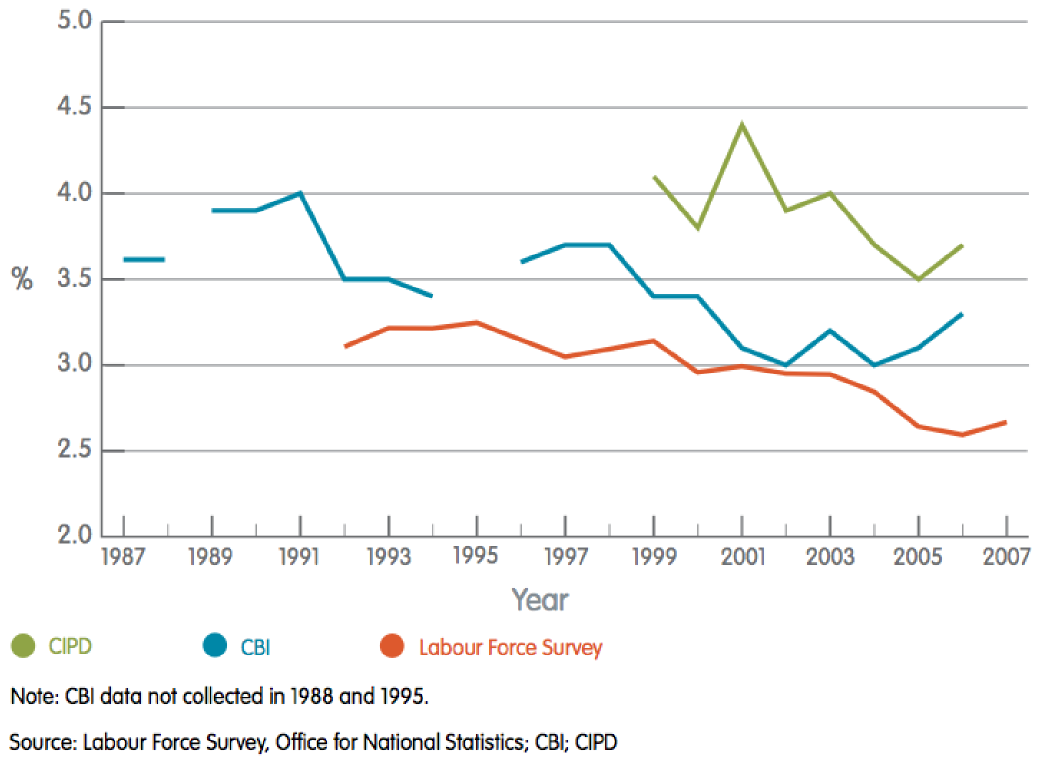


Figure 6.1: Sickness absence as a proportion of working time. Figure adapted from [Black, 2008]

made it difficult to draw any general conclusion from the direct comparison of the absence rate between these surveys [Holmes, 2008], Among these figures, because it provides more comprehensive breakdowns of the raw data according to various categories, we decided to adopt the figures from CIPD [CIPD, 2011] as the base line. CIPD surveys suggested that the absent rate ranges from approximately 3.5% to 4.5%.

In addition, this absence rate needs to be adjusted to take into consideration of paid holidays, because the paid holidays of employees can also effect the project progress even though it is not considered as part of the absence in general. The number of paid holidays varies from 20 to 35 days per year in UK [Danzer and Dolton, 2012]. Therefore, the average absence rate of employee is roughly adjusted to the range of 10.9% to 17.4%. In the implementation of our simulations, we assume that the range of absence rate from 0% to 25% is sufficient to cover the real case.

To clarify, the term *absence rate* used in this thesis refers to the average rate of a staff being absent from work when other staff are in attendance. It consists of paid holidays, sickness absence, away because of family responsibilities, stress or training, etc.. It excludes weekends (104 days per year) and public holidays (8-11 days per year in UK) by definition.

In our work reported in this chapter, instead of attempting to reduce the staff absences, we quantitatively study the analysis of various possible scenarios of staff absence and work package scheduling, and their impacts on project finish time. Previous scheduling and staff rostering techniques are unable to customise the staff availabilities calendar to simulate unavailabilities of staff in an intuitive way. We are the first to propose Co-evolutionary Optimisation techniques to deal with this kind of problem. The improvement of our previous work on using the Cooperative Co-evolutionary Algorithm on software project management is reported in this chapter. Employee skills are considered in the process of simulating the project process.

A set of four industrial project data has been used to validate our techniques, results and analysis for each specific cases is provided, and suggestions on customisations to suit different real world cases have also been put forward.

6.1.1 Research Questions

The following three research questions will be answered through the studies of work:

RQ1: How do the co-evolutionary optimisation techniques find desirable extreme solutions for the best and worst case scenarios?

RQ2: How do the co-evolutionary optimisation techniques reveal the dynamic correlation between work package scheduling and staff absence?

RQ3: How does the staff absence rate interfere with the project completion time?

6.2 Problem Statement

The results reported in this chapter come from the application of our approach to *scheduling* and *staffing* problems for four real–world software projects. Each of these project plans has a set of Work Packages (WPs) that ought to be executed by a number of staff.

On the one hand, information on work packages consists of the estimations of the effort of executing each work package, and the constraint of dependency among WPs need to be satisfied. Furthermore, each work package consists of a set of sub work packages that can only be executed by staff with the required skills. The problem of finding optimal solutions of scheduling problem is formulated as finding *Work Package Ordering*.

On the other hand, each member of staff possesses a particular skill required by one or more WPs in the project, and multiple members of staff that possess the same skill are considered identical in terms of their performance of executing corresponding WPs. In addition, the absence of staff from the project is considered in this model. The problem of finding optimal solutions of staffing problem is formulated as finding *Staff Availability Calendar*.

6.2.1 *Work Package Ordering* (WPO)

Work Package Ordering (WPO) essentially defines the order of considering the execution of the work packages in a project. An abstract representation is illustrated in Figure 6.2.

On the project level, WPO is partially affected by the dependency constraints among work packages. More importantly, a large enough number of automatically generated WPOs can properly explore the potential "parallelism" of the executions on those WPs which are without dependency constraint and thus "good" solutions are said to be found in the case that the resource (staff) is utilised in a more extreme way (either more efficient or more inefficient). Therefore, the fitness of a WPO must be evaluated against the staff availabilities, because the "goodness" of a WPO is subject to the constraint of the availabilities of required staff.

6.2.2 *Staff Availability Calendar* (STCAL)

Staff Availability Calendar (STCAL) reflects all sorts of possible combinations of absence of all members of staff on the day that a regular full time employee is supposed to be available. An abstract representation is shown in Figure 6.3.

During the course of a project, staff with different skills become unavailable to work for many possible reasons, most of which cannot be properly planned in advance. The delay caused by such staff absence may vary depending on: 1) whether there is a corresponding work package planned to be executed during the absence and 2) whether and when another member of staff with the corresponding skill becomes available. Thus, similarly to the evaluation of WPO, the fitness of STCAL also must be evaluated against the order of considering to execute the work packages.

6.3 Co-evolution

As discussed in the previous section, we aim to optimise the solutions of two problems. The Co-evolutionary Optimisation technique is adapted to evolving two species of these solutions. In this section, we first introduce the *representation* of these two kinds of solutions, and the *genetic operators* which illustrate the methods of reproducing new solutions based on existing solutions. Secondly, the mechanism of deciding which are the solutions to be kept along the evolutionary process is stated in the *fitness evaluation and selection* subsection. Finally, an abstract level of the algorithm that describes the implementation of the co-evolutionary procedure among two species is presented.

6.3.1 Genetic Representations

6.3.1.1 Array of WP's IDs representing *Work Package Ordering*

As the representation of the solution and the genetic operator for WPO is the same as we proposed in SSBSE CCEA–PM paper [Ren et al., 2011], they are briefly repeated below for the sake of convenience.

Work Package Order:	$1st$	$2nd$	$3rd$	$\cdots$	$(l-1)th$	l-th
Work Package ID:	3	2	6	$\cdots$	l	$l-4$

Figure 6.2: WPO Chromosome: The gray area is the representation of one specific ordering for distributing a set of l work packages. As shown the solution is represented by a string of length l, each gene corresponding to the distributing order of the WPs and the alleles, drawn from $\{1, ..., l\}$, representing one WP's ID.

6.3.1.2 Boolean matrix representing *Staff Availability Calendar*

The staff availability calendar is represented as an N_S by N_D table in which N_S is the number of staff, and N_D is the number of days of the project duration. An illustrative example is given in Figure 6.3. The number “1” indicates the absence of the corresponding member of staff on specified days, e.g. sickness leave, planned holiday.

Calendar (n-th Day)

		1	2	3	$\cdots$	N_D
	Alice	0	**1**	0		**1**
	Bob	0	0	**1**		0
Staff	Carol	**1**	0	0		**1**
	$\vdots$				$\ddots$	
	Steve	**1**	0	0		**1**

Figure 6.3: The representation of *Staff Availability Calendar* with "1" indicating the day a member of staff is not available

6.3.2 Genetic Operators

6.3.2.1 Order Crossover on WPO

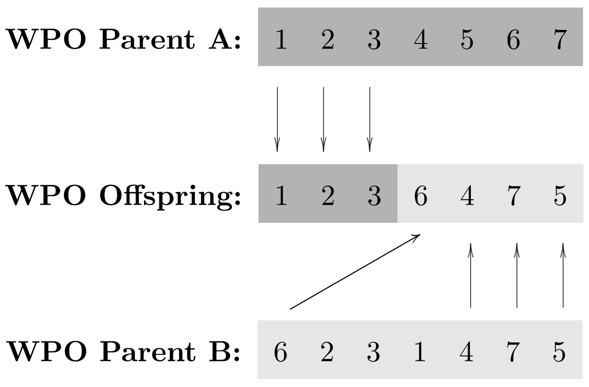


Figure 6.4: WPO Crossover: Order Crossover

6.3.2.2 Mutation on WPO

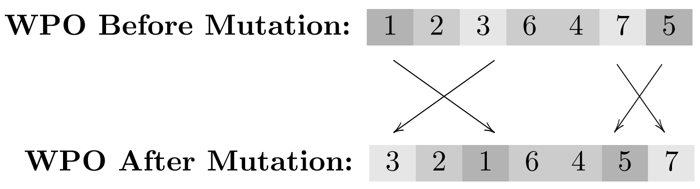


Figure 6.5: WPO Mutation: Randomly Swap WPs' Positions

6.3.2.3 Dependency and Duplication Verification on “newborn” WPO

After the crossover and mutation on ordering the WPs, a “newborn” solution of WPO is produced with potential violations of WPs’ dependency constraints. Because the scheduling simulator is designed to process only those WPOs without any violations of dependency constraint, all “newborn” WPOs need to be verified by a dependency checker, and any dependency violations should be removed.

The dependency checker goes through every single WP in an offspring WPO, one by one, to verify whether a WP’s predecessors are all placed before it in the ordering. The verification is considered as passed only if the predecessor is before the successors for every pair of WPs with dependency constraints. Otherwise, the violation will be repaired by placing the predecessor to the position just in front of its successor.

After the verification process, the offspring will be further examined for possible duplications. If the newly-generated WPO has the same ordering with the existing individuals, it will be mutated and verified again until a new and different solution is found.

6.3.2.4 Uniform Crossover on *Staff Availability Calendar*

Uniform crossover will be performed along the vertical dimension on the *Staff Availability Calendar*. This means that the offspring inherits the availability of one specific member of staff, the same with one of its parents, while it has equal chance to inherit it from either of the parents. The crossover process is illustrated in Figure 6.6.

6.3.2.5 Mutation on *Staff Availability Calendar*

As shown in each row of Figure 6.3, a member of staff’s availability is represented where “1” indicates the absence of one member of staff or “0” indicates otherwise. The mutation is performed as swapping the bits of “1” randomly with a bit of “0” with the probability of 20%. The mutation process is shown in Figure 6.7. It is worth mentioning that the total amount of sick–days is fixed for each member of staff and for the whole project.

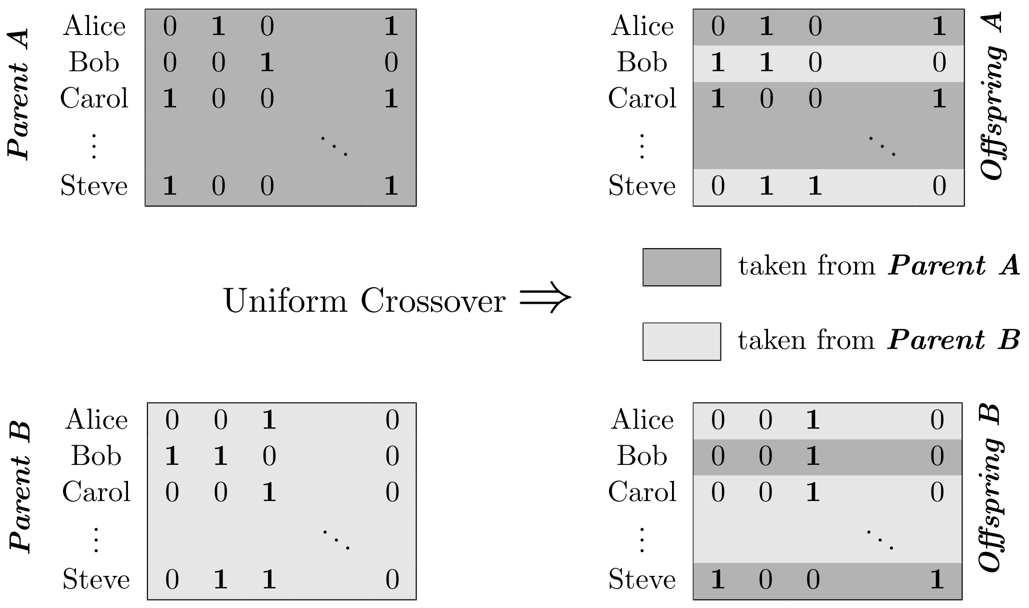


Figure 6.6: Uniform Crossover on *Staff Availability Calendar*: offspring inherit availabilities of one specific member of staff as for the whole period of the project (a whole row on the chromosome) from either of the parents with equal probability.

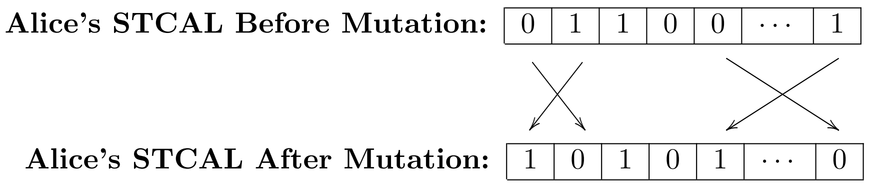


Figure 6.7: Mutation on *Staff Availability Calendar*: for each member of staff, the positions of a '1' and a random '0' are swapped with a defined probability.

6.3.3 Fitness Evaluation and Selection of Candidate Solutions

6.3.3.1 Scheduling Simulation

As shown in Figure 6.8, the *Scheduler* (the scheduling simulator) takes one WPO and one STCAL solution as its input. In the course of simulating a project, provided there is a prioritised list of all the WPs (WPO), a *Scheduler* understands the order of WPs to be considered assigning a member of staff to. At the same time, provided there is a STCAL, the *Scheduler* understands the availability of staff at any given time of the project.

In addition to the inputs (WPO and STCAL), some other essential information of WPs and staff is also needed for the simulation: 1) required skills and efforts of

	1*st*	2*nd*	3*rd*	$\cdots$	$(l-1)th$	l-*th*
WP ID	3	2	6	$\cdots$	l	$l-4$

Work Package Ordering

Calendar

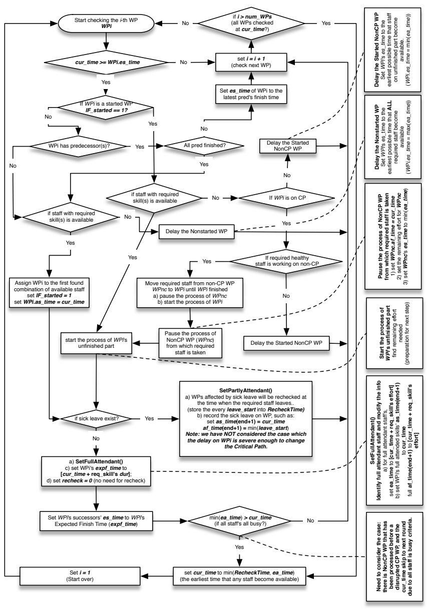
$$\begin{array}{c|cccc|}
 & & & & \\
\textit{Staff} & 0 & \mathbf{1} & 0 & \mathbf{1} \\
 & 0 & 0 & \mathbf{1} & 0 \\
 & \mathbf{1} & 0 & 0 & \mathbf{1} \\
 & & & \ddots & \\
 & \mathbf{1} & 0 & 0 & \mathbf{1}
\end{array}$$

Staff Availability Calendar

$\searrow \quad \swarrow$

Scheduling Simulator

For a given project, the information of *work package dependency, required skills and efforts, and the number of staff and the skills they have* is not varied by different input solutions and therefore stored on the side.


Start checking the i-th WP WPi
if i > num_WPs (all WPs checked at cur_time?)
cur_time >= WPi.es_time
set i = i + 1 (check next WP)
Set es_time of WPi to the latest pred's finish time
If WPi is a started WP IF_started == 1?
WPi has predecessor(s)?
All pred finished?
Delay the Started NonCP WP
if staff with required skill(s) is available
If WPi is on CP
Delay the Nonstarted WP
If required healthy staff is working on non-CP
Assign WPi to the first found combination of available staff set IF_started = 1 set WPi.as_time = cur_time
Move required staff from non-CP WP WPnc to WPi until WPi finished a) pause the process of WPnc b) start the process of WPi
start the process of WPi's unfinished part
Pause the process of NonCP WP (WPnc) from which required staff is taken
Delay the Started NonCP WP
SetPartlyAttendant() a) WPs affected by sick leave will be rechecked at the time when the required staff leaves.. (store the every leave_start into RecheckTime) b) record the sick leave on WP, such as: set as_time(end+1) = cur_time af_time(end+1) = min(leave_start) Note: we have NOT considered the case which the delay on WPi is severe enough to change the Critical Path.
if sick leave exist?
a) SetFullAttendant() c) set WPi's expf_time to [cur_time + req_skill's dur]; d) set recheck = 0 (no need for recheck)
Set WPi's successors' es_time to WPi's Expected Finish Time (expf_time)
min(ea_time) > cur_time (if all staff's all busy?)
Set i = 1 (Start over)
set cur_time to min(RecheckTime, ea_time) (the earliest time that any staff become available)
Yes
No

$\Downarrow$

Simulation Result

The fitness of either of the input solutions will be evaluated mainly according to the *project completion time* in the simulation result.

Figure 6.8: Scheduling simulation with *Work Package Ordering* and *Staff Availability Calendar*. The scheduling simulator takes one WPO solution and one STCAL solution as its inputs. The simulation result illustrates the process of a corresponding project being executed, such as: *project completion time*, *staff-to-WP allocation.*

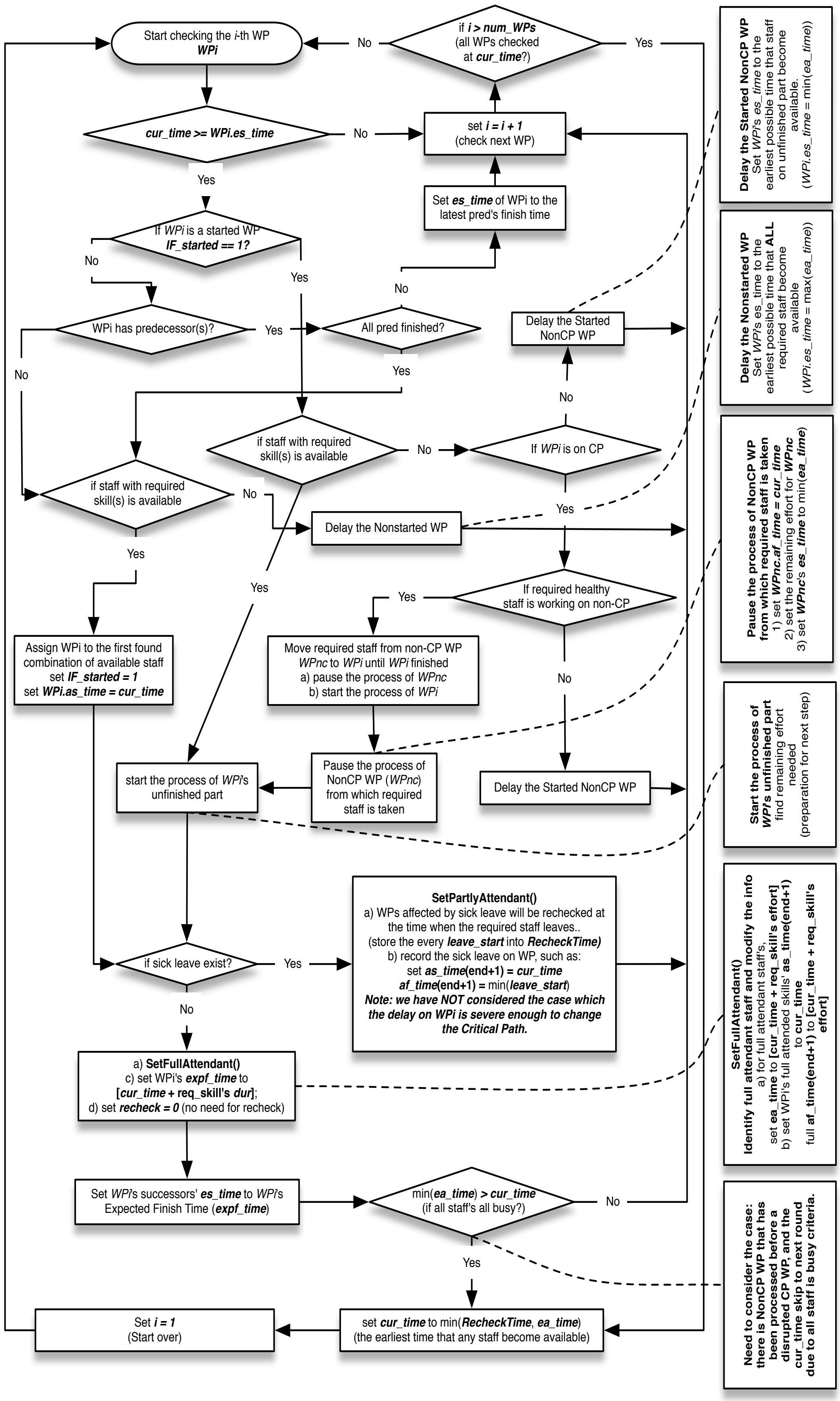

Start checking the i-th WP WPi
if i > num_WPs (all WPs checked at cur_time?)
No
Yes
cur_time >= WPi.es_time
No
set i = i + 1 (check next WP)
Yes
Set es_time of WPi to the latest pred's finish time
If WPi is a started WP IF_started == 1?
No
Yes
No
WPi has predecessor(s)?
Yes
All pred finished?
Delay the Started NonCP WP
No
Yes
No
if staff with required skill(s) is available
No
If WPi is on CP
if staff with required skill(s) is available
No
Yes
Delay the Nonstarted WP
Yes
Yes
If required healthy staff is working on non-CP
Yes
Assign WPi to the first found combination of available staff set IF_started = 1 set WPi.as_time = cur_time
Move required staff from non-CP WP WPnc to WPi until WPi finished a) pause the process of WPnc b) start the process of WPi
No
start the process of WPi's unfinished part
Pause the process of NonCP WP (WPnc) from which required staff is taken
Delay the Started NonCP WP
SetPartlyAttendant() a) WPs affected by sick leave will be rechecked at the time when the required staff leaves.. (store the every leave_start into RecheckTime) b) record the sick leave on WP, such as: set as_time(end+1) = cur_time af_time(end+1) = min(leave_start) Note: we have NOT considered the case which the delay on WPi is severe enough to change the Critical Path.
if sick leave exist?
Yes
No
a) SetFullAttendant() c) set WPi's expf_time to [cur_time + req_skill's dur]; d) set recheck = 0 (no need for recheck)
Set WPi's successors' es_time to WPi's Expected Finish Time (expf_time)
min(ea_time) > cur_time (if all staff's all busy?)
No
Yes
Set i = 1 (Start over)
set cur_time to min(RecheckTime, ea_time) (the earliest time that any staff become available)
Delay the Started NonCP WP Set WPi's es_time to the earliest possible time that staff on unfinished part become available. (WPi.es_time = min(ea_time))
Delay the Nonstarted WP Set WPi's es_time to the earliest possible time that ALL required staff become available (WPi.es_time = max(ea_time))
Pause the process of NonCP WP from which required staff is taken 1) set WPnc.af_time = cur_time 2) set the remaining effort for WPnc 3) set WPnc's es_time to min(ea_time)
Start the process of WPi's unfinished part find remaining effort needed (preparation for next step)
SetFullAttendant() Identify full attendant staff and modify the info a) for full attendant staff's, set ea_time to [cur_time + req_skill's effort] b) set WPi's full attended skills' as_time(end+1) to cur_time full af_time(end+1) to [cur_time + req_skill's effort]
Need to consider the case: there is NonCP WP that has been processed before a disrupted CP WP, and the cur_time skip to next round due to all staff is busy criteria.

Figure 6.9: Program flowchart of the scheduling simulation.

each WP, and dependency among all WPs; and 2) the corresponding skills processed by each individual member of staff. However, because they are assumed to not be changed for the same project, they are stored as static values that are always accessible to the *Scheduler*.

A simplified but comprehensive decision making process of the *Scheduler* is represented as a program flowchart in Figure 6.9. In summary, when the *Scheduler* decides whether or not there is an available member of staff that can be assigned to execute a WP at a given moment, the following main criteria needs to be verified: 1) whether or not all this WP's predecessors are finished before the current moment; 2) whether or not there is an appropriate member of staff available; and 3) if this WP is on the critical path, staff working on non-critical WP should be reallocated to this WP subject to the constraints of skill requirement.

6.3.4 Overall Co-evolution Procedure

As previously stated, the software project management problem is divided into two subproblems: finding optimal WPO and STCAL. As finding optimal solutions for WPO and STCAL are two separately but closely related tasks and the evaluation criteria of these two subproblems are the same, a 2–population based cooperative co-evolution optimisation technique is adapted.

On the project level, the solutions for managing a project are composed of two pieces: WPO and STCAL. The solution is evaluated by simulating the project execution of these two pieces together in the scheduling simulator. The result of the simulation of such management plan is the *Project Duration*, which is shared by these two pieces of subsolution to form their fitness values.

The co-evolutionary optimisation procedure is described in Algorithm 3. Initially, two random population of parents, P_{wpo} and P_{stcal}, are created, evaluated, and ranked. Each individual solution is assigned a fitness value according to the scheduling simulation result with solutions from the other population. A chronicle records the initial population. In the main loop of the algorithm, tournament selection, crossover and mutation operators are used to create the offspring population. The parents are joined by the offspring population to form the intermediate population, and then each solution in *intermediate* population is evaluated by the simulation with the *parent* solutions in the other population. Selections on better solutions are proceeded based on the evaluation. Only the surviving solutions in the intermediate population are recorded in C_{wpo} and C_{stcal}, and are passed on to breed the next generation. This main loop of reproducing and selection process repeats until the stopping criteria is met. Finally, after the co-evolutionary procedure is terminated, all the surviving individuals recorded in C_{wpo} and C_{stcal} are assessed with two same sets of individuals H_{wpo} and H_{stcal} that are selected as elites from the entire history in the other population.

6.4 Empirical Study

This section report the results of an empirical study of our absenteeism management approach on four real world software projects.

6.4.1 Parameter Setting

In the implementation, the following details are taken from the industrial data sets, and they are different for each project: 1) total number of WPs, 2) dependence among WPs, 3) required skills and effort to execute each WP, 4) total number of Staff, and 5) the skills each staff member possesses. For the purpose of setting a meaningful value for the N_D in *Staff Availability Calendar*, the length of the critical path is calculated in advance and set as the number of days of the project duration.

A set of three values, [0.1%, 10%, 25%], for the *Staff Absence Rate* have been tested for all the projects. These three values of absence rate are used as in the following three scenarios respectively: 1) all staff attend the project without any absence, 2) staff attend the project with an absence rate that is close to industrial average rate as reviewed in Section 6.6, and 3) staff attend the project with an abnormally high absence rate.

Although the situations of absence for each employee over a project period are simulated independently to the other employees, the absence rate is the same for all staff in a single run. The motivation of applying a flat absence rate is attempting to provide equal opportunities to every single staff member, because we should not hold the bias of assuming different rates for certain staff members without the knowledge of previous record of individual staff. It is certain that the assumption of flat absence

```
begin Initialisation
    N ← the total number of generations of the co-evolution process
    P - main population:
      $P_{wpo}$ ← randomly generated solutions of WPO
      $P_{stcal}$ ← randomly generated solutions of STCAL
      initial fitness evaluation on both population
    O - offspring population:   $O_{wpo} \leftarrow \emptyset$;   $O_{stcal} \leftarrow \emptyset$
    I - intermediate population:   $I_{wpo} \leftarrow \emptyset$;   $I_{stcal} \leftarrow \emptyset$
    C - chronicle (record the entire survived population in runtime):
      $C_{wpo}(1) \leftarrow P_{wpo}$
      $C_{stcal}(1) \leftarrow P_{stcal}$
    H - hall of fame (select only a small number of elites): $H_{wpo} \leftarrow \emptyset$; $H_{stcal} \leftarrow \emptyset$
end
for $n$ from 1 to $N$ do
    //evolve the WPO population
    for each individual $P_i \in P_{wpo}$ do
        $P'_i$ ← TournamentSelect($P_{wpo}$)        /* select a second individual */
        $O_{wpo}$ ← Join($O_{wpo}$, CrossoverAndMutation($P_i$, $P'_i$))           /* breed */
    end
    $I_{wpo}$ ← Join($P_{wpo}$, $O_{wpo}$)          /* form the intermediate population */
    EvaluateInternalFitness($I_{wpo}$, $C_{stcal}(n)$)
    $P_{wpo}$ ← SelectFirstHalf($I_{wpo}$)                   /* preserve the better half */

    //evolve the STCAL population
    for each individual $P_i \in P_{stcal}$ do
        $P'_i$ ← TournamentSelect($P_{stcal}$)
        $O_{stcal}$ ← Join($O_{stcal}$, CrossoverAndMutation($P_i$, $P'_i$))
    end
    $I_{stcal}$ ← Join($P_{stcal}$, $O_{stcal}$)
    EvaluateInternalFitness($I_{stcal}$, $C_{wpo}(n)$)
    $P_{stcal}$ ← SelectFirstHalf($I_{stcal}$)

    //record the survived individuals
    $C_{wpo}(n+1) \leftarrow P_{wpo}$
    $C_{stcal}(n+1) \leftarrow P_{stcal}$
end
begin Assessment of External Fitness
    $H_{wpo}$ ← SelectHallofFame($C_{wpo}$)
    $H_{stcal}$ ← SelectHallofFame($C_{stcal}$)

    for $n$ from 1 to $N$ do             /* assess all survived individuals */
        for each $P_i \in C_{wpo}$ do
            AssessExternalFitness($P_i$, $H_{stcal}$)
        end
        for each $P_i \in C_{stcal}$ do
            AssessExternalFitness($P_i$, $H_{wpo}$)
        end
    end
end
```

Algorithm 3: 2-Population Cooperative Co-evolutionary Algorithm

rates is not true in the real world, although it reduces the complexity of the model dramatically.

After trials of simulation on four industrial projects with different difficulties of finding "good-enough" solutions, the following settings for the co-evolutionary algorithm are found to be able to demonstrate the optimisation progress for all of the project data. The *Number of External Generations* is set to 25 and the *Size of Population* is set to 80. The *Number of Internal Generations* is set to 1 to achieve the most frequent possible communications between two populations. The *Mutation Rate* is set to 30%.

6.4.2 Four Configurations of Co-evolutionary Optimisation

As explained in detail in the simulation of project execution in Section 6.3.3.1, it takes a WPO and a STCAL cooparatively to simulate a particular real–world case of project execution. The absolute simulating result is essentially the same for these two inputs because the total project duration of such project execution is considered as the only evaluation criteria for both parts. Even though the absolute fitness value is the same for both parts from two populations, the selection function (based on internal fitness values) that guides the evolutionary process can be different. To be more specific, the selection process can choose to keep those solutions with either the highest fitness value, or the lowest fitness values. This mechanism allows the co-evolutionary process to be able to perform four kinds of configurations as introduced as followings.

For either population, the evolutionary process can be competitive or cooperative with the other population on the fitness value. For the sake of clarity, the solutions associated with *longer* durations are considered as *worse* solutions as the project that adapts this plan has a longer execution time, and a *better* solution has relatively a *shorter* duration. There are four possible situations listed below:

***Configuration BWWS*: Competitively Searching for Better WPOs and Worse STCALs**: in this case, the optimisation process aims to find good WPOs and bad STCALs. It helps mangers explore the worst possible impact of staff absence assuming good managements on work package ordering.

***Configuration WWBS*: Competitively Searching for Worse WPOs and Better STCALs**: the optimisation process aims to find bad WPOs and good STCALs. It helps to investigate the possibility of saving a project by arranging staff absence to compensate the impact of bad scheduling on work packages.

***Configuration WWWS*: Cooperatively Searching for Worse WPOs and Worse STCALs**: in both populations, solutions evolve with increasingly longer durations for project execution. The optimisation process aims to find solutions for both WPO and STCAL that can cause longer delays on the project. These solutions can help project managers learn about the worse case scenarios to be avoided in practice.

***Configuration BWBS*: Cooperatively Searching for Better WPOs and Better STCALs**: in both populations, solutions evolve with decreasingly shorter durations for project execution. These solutions can help to review the best case scenarios that in favour.

In general, we can assume that the solutions found by these four configurations should follow the following patterns. Given a project and a fixed staff absence rate: *Configuration WWWS* should eventually find solutions with the longest project completion time, and similarly *Configuration BWBS* should find solutions with the shortest durations, while *Configuration BWWS* and *WWBS* should find solutions in between.

6.5 Results Analysis

While all the raw results are presented in the figures attached in the Appendix A, two tables provide a general summary of the results about the quality of the outcome (Table 6.1) and the abilities of the algorithms in finding them (Table 6.2).

The statistics shown in these two tables are based on the External Fitness Assessments conducted by evaluating all the surviving individuals in every generation against a fixed set of elites in the history of the evolutionary process (a.k.a.: "hall of fame"). Whilst the Internal Fitness helps the optimisation system to determine selection during co-evolutionary process, the External Fitness helps to gauge the progress of the algorithm. The External Fitness is chosen for statistical comparison, because it is an absolute measurement that makes the general comparison between individual solutions across different generations become meaningful.

By analysing the statistics, observations have been recorded and discussed in detail. An example of specific case is also presented. The research questions are answered in the end of this section, and suggestions for using this tool as a source of decision support to the project manager are also proposed.

6.5.1 Running Time

The volume of the raw results is large. There are 4 configurations of Co-evolutionary GAs {BWWS, WWBS, WWWS, BWBS}, 3 levels of staff absence rates {0.1%, 10%, 25% }, 2 populations evolving simultaneously, and all the combinations above are to be run on 4 real–world project data. That makes 4*3*2*4 = 96 rounds of co-evolutionary processes in total to produce the raw results, and then another 96 rounds of external fitness assessments to assess the entire surviving population during the entire co-evolutionary process.

On the one hand, the running time of these co-evolutionary processes vary from just over ten minutes to over four hours, depending on the features of the project as well as the configurations. When the Matlab's parallel computing feature is enabled, it takes nearly two hours on average for each of these 96 simulations to run on a PC with Intel i5 3.2GHZ, 3.4GB RAM, Windows 8 32bit.

On the other hand, another 96 rounds of external fitness assessments are done to assess whether or not the optimisation is making progress during the process of producing the raw results. The external fitness assessment takes about half of the running time as that of the optimisation process.

Table 6.1: The table shows the average fitness of the solutions found by algorithms in terms of their externally assessed project finish time (Days). This reveals the quality of the solutions found in the last generation of the co-evolutionary process. The maximum and minimum values in each row are highlighted in bold font. Extrema tend to be found in cooperative search, especially in the solutions of WPO in cooperative search. In competitive search, optimisation on WPO dominates the competition on complex projects (C and E), whilst optimisation on STCAL dominates the competition on simpler projects (D and F).

		Co-evolutionary Algorithm Configurations							
		Competitive Search				Cooperative Search			
Project Name		BWWS		WWBS		WWWS		BWBS	
Absence Rate		WPO	STCAL	WPO	STCAL	WPO	STCAL	WPO	STCAL
	0.1%	126.5	126.2	137.9	142.0	147.1	**147.9**	**116.6**	117.1
C	*10%*	134.7	134.2	141.1	142.1	**162.0**	160.3	**127.2**	127.5
	25%	149.6	150.0	153.7	154.0	**184.9**	181.4	**142.4**	142.7
	0.1%	**25.0**	24.1	**21.0**	**21.0**	**25.0**	**25.0**	**21.0**	**21.0**
D	*10%*	29.0	29.0	**21.0**	**21.0**	**30.0**	**30.0**	**21.0**	**21.0**
	25%	35.3	35.1	22.4	22.5	**36.0**	**36.0**	**22.0**	22.1
	0.1%	336.6	338.0	361.8	360.4	**408.7**	408.1	**299.7**	300.3
E	*10%*	403.3	397.8	393.3	392.0	**463.9**	446.8	**335.1**	344.4
	25%	431.1	432.4	443.7	441.6	**497.0**	490.5	**379.1**	385.1
	0.1%	75.1	77.2	70.3	70.3	**83.1**	79.5	**69.2**	**69.2**
F	*10%*	85.8	84.5	70.0	70.4	**87.1**	85.1	**69.2**	**69.2**
	25%	87.9	87.4	76.8	76.7	**90.0**	87.7	**74.8**	75.4

6.5.2 Average External Assessment of the Solutions Found

Table 6.1 shows the average external fitness value of solutions in the last generation for all the configurations and projects. In other words, for each of the four real–world project data, the table reveals the quality of the solutions of WPO and STCAL, in terms of their finish time, found by four co-evolutionary configurations under three levels of absence rate.

Observation I: extrema tend to be found by cooperative searching. In the Table 6.1, for the purpose of comparing the results among different algorithm configurations under the same given project and absence rate, the shortest and longest average durations in each row are highlighted in bold font. The highlighted values tend to concentrate on the right hand side of the table (where the results of *cooperative* co-evolutionary optimisation are shown) because, with the cooperative co-evolutionary configurations, WWWS and BWBS, both populations aim to evolve towards the same directions concurrently. For example, with *Configuration WWWS*, the population of solutions for both WPO and STCAL are evolved in finding the worst case – longer project completion time. As a result, given a project data and the same staff absence rate, the maximum durations tend to be found by the *Configuration WWWS*. Similarly, the minimum durations tend to be found by the *Configuration BWBS*.

Essentially, this observation also confirms the effectiveness of the optimisation on both populations. To achieve solutions to fulfil either longer or shorter project durations, the optimisations on both WPO and STCAL are equally important.

Observation I(a): extrema tend to be found in the solutions of WPO. Observation I confirms that the optimisation on both WPO and STCAL are equally important. Nevertheless, when comparing only the results of WWWS and BWBS, the extrema tend to be found in the solutions of WPO rather than distributed evenly among the two population.

The reason for this observation is unclear by the time of writing up this thesis. This might have something to do with the method of External Fitness Assessment. The best solutions found on the [1/2, 3/4, 1] (half way, third quarter way, and the finale) of the entire evolutionary process were selected as the "hall of fames", which are later on used to externally assess the progress of the solutions found by algorithms. The terms *External Assessment* refers to the assessment that has no effect on the algorithms' ability to find desirable solutions, unlike *Internal Evaluation.*

Observation II: in competitive search, the degree of complexity of the project tend to decide the species that dominates the competition. When comparing each row of the results under "BWWS" and "WWBS", average durations of the solutions under "BWWS" tend to be shorter than those under "WWBS" on project C and E. This magnitude relation coincides with the objective of the optimisation on WPO, meaning that, when it is searching for a "Better" WPO, the average durations are shorter than the case when it is in the search of a "Worse" WPO, in spite of the fact the other species (STCAL) is optimised towards the opposite direction. In contrast, for project D and F, projects are much more simpler and the entries under "BWWS" are all larger than those under "WWBS", which coincide with the objective of the optimisation on STCAL.

As previously mentioned, Project C and E are considered more complex than Project D and F on the grounds that Project C and E have larger numbers of work packages, as well as larger numbers of dependencies among them.

Given our knowledge on the projects and the observation that dominating species in competitive search is not the same for all projects, it can be concluded that in the competitive search, the optimisation on WPO dominates the competition in complex projects (C and E), whilst the optimisation on STCAL dominates the competition in relatively simpler projects (D and F).

Table 6.2: The Spearman's rank correlation coefficient table indicates the trend in the improvements of average external fitness values of the entire population as the co-evolutionary progress proceeds over generations. The values highlighted in bold font indicate the cases in which there is no statistically significant overall trend of improvement.

		Co-evolutionary Algorithm Configurations							
		Competitive Search				Cooperative Search			
Project Name		BWWS		WWBS		WWWS		BWBS	
Absence Rate		WPO	STCAL	WPO	STCAL	WPO	STCAL	WPO	STCAL
	0.1%	-0.726	0.933	0.782	-0.535	1.000	0.969	-0.992	-0.906
C	*10%*	-0.965	0.963	0.808	-0.962	0.997	0.993	-0.923	-0.939
	25%	-0.920	0.996	0.827	-0.952	0.999	0.994	-0.985	-0.876
	0.1%	**0.000**	0.880	**0.000**	**-0.471**	**0.000**	0.993	**0.000**	**-0.471**
D	*10%*	**-0.471**	0.957	**0.000**	-0.956	**0.340**	0.943	**0.000**	-0.886
	25%	**0.000**	1.000	**0.000**	-1.000	0.549	1.000	-0.763	-1.000
	0.1%	-0.843	0.962	0.922	-0.833	0.943	0.983	-0.987	-0.635
E	*10%*	-0.584	0.984	0.939	-0.818	0.980	0.993	-0.973	-0.999
	25%	-0.822	0.970	0.935	-0.980	0.958	0.996	-0.708	-0.993
	0.1%	**-0.274**	0.928	0.788	-0.791	0.743	0.955	**0.000**	**-0.340**
F	*10%*	**-0.146**	0.988	**0.272**	-0.997	0.799	0.988	-0.887	-1.000
	25%	**0.221**	0.988	0.788	-0.984	0.624	0.993	-0.878	-0.998

6.5.3 Trend of Improvement on Solutions in the Process of Searching

In order to observe the trend of the improvement on the quality of the solutions, the Spearman's rank correlation coefficient (SRCC) is conducted to indicate the correlation between the average external fitness values, and the generation numbers. SRCC is used to assess the probability of the existence of a monotonic function that can be used to describe the correlation of two variables. The ρ value always lies in the range ($-1 \leq \rho \leq 1$). In practice, no strong correlation can be concluded for $-0.5 \leq \rho \leq 0.5$. The closer the value to 1, the stronger the positive correlation; the closer the value to -1, the stronger the negative correlation.

The comparison of external fitness values among generations becomes meaningful, because the external fitness measurement is an absolute measurement by assessing all the solutions across generations against the same set of elites.

As shown in Table 6.2, the observed ρ values indicate the probability of the average external fitness values (i.e., Finish Time) of the entire population is increasing or decreasing as the optimisation progress proceeds over generations (i.e., Generation Number).

In our experiments, a high value of ρ indicates a desirable improvement in finish time is observed. More specifically, a high positive value of ρ indicates that an upward trend in finish time improvement, and a high negative value of ρ indicates a downward trend in improvement.

Observation III (a): for simple projects, the cases of no improvements (indicating by small value of ρ) are most likely to be observed on the solutions of WPO rather than STCAL. By definition, a small ρ value ($|\rho| < 0.5$) indicates that there is no overall significant trend of improvements. Those ρ values, which have been are in bold font in Table 6.2, could indicate either a static fitness value or a trembling line. All these highlighted cases are to be found in the result for Project D and F. Because these two projects are relatively simpler, the optimisation on WPO found good solutions in an early stage of the process, and thereafter no further improvement on the fitness values, such as the left plot in Figure A.4 (b).

Exceptionally, an interesting trembling line was found for Project F in one specific configuration. The WPO's average fitness value in the Configuration (BWWS, 0.1%) is trembling during the optimisation process, as shown on the top-left plot in Figure A.8 (a). Because there is insufficient empirical results to investigate this exception further, it is left for future work.

6.5.4 Detailed Case Analysis on Configuration BWBS

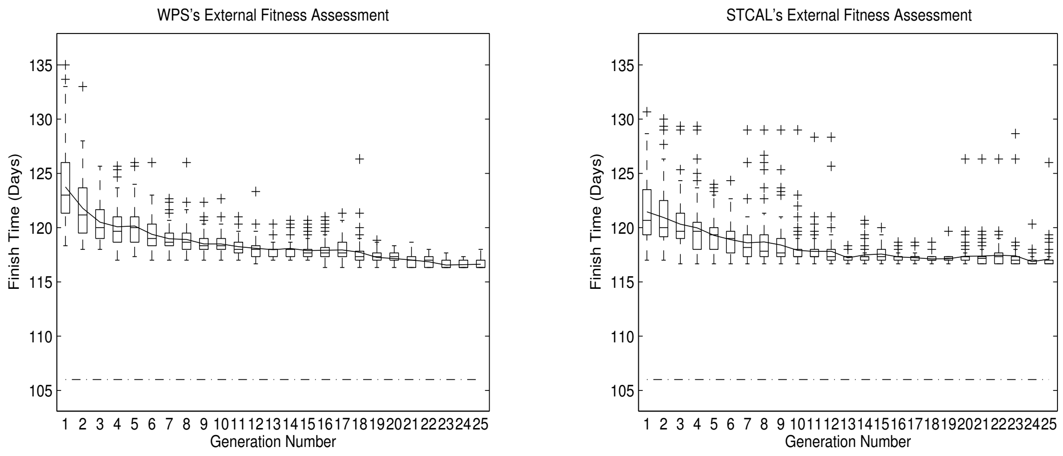
In addition to the general analysis provided above, the result of Configuration BWBS is picked out for detailed analysis, for the purpose of demonstrating that our co-evolutionary optimisation technique can be helpful to the decision maker to obtain an insight into the project.

With Configuration BWBS, because the optimisation objectives for both WPO and STCAL are to find solutions with shorter project finish times, those solutions for scheduling and staffing are considered as optimal solutions if the external fitness value (i.e., Finish Time) is equal to the shortest possible duration. In general, the shortest possible duration of completing a project is equal to the length of the critical path provided that all required resources are sufficient. To provide such a reference, the duration of executing the critical path for each project is plotted accordingly as a dashed line near the bottom of every sub-figure in Figure 6.10, 6.11, 6.12, and 6.13.

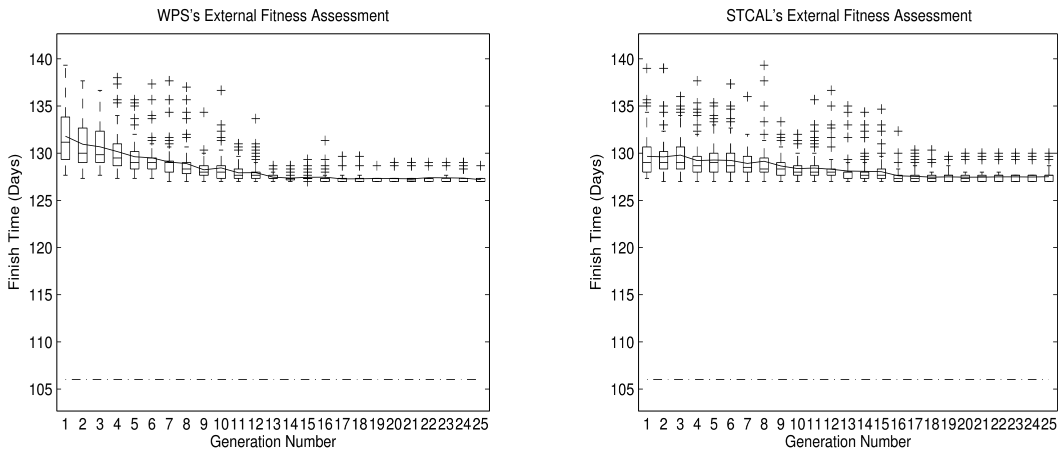
In each of these sub-figures, all surviving solutions in the entire optimisation process are box-plotted according to their external fitness values (i.e., Finish Time) indicated on a vertical axis, while the numbers on horizontal axis indicate the generation when they were survived.

Observation III (b): in cooperative search for shorter project finish time, when the project is less complex, small rate staff absence can be compensated by the optimisation. In the comparisons on the results of Configuration BWBS on all four projects over three levels of staff absence rate, the results of Project D and F stand out, because optimal scheduling and staffing solutions were only found for these two projects and only on the small and medium levels of staff absence.

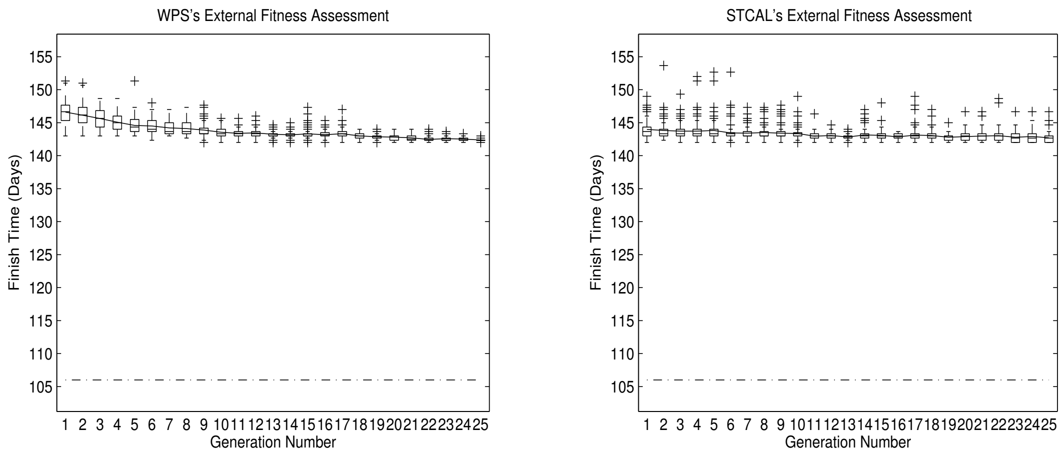
For these two relatively simpler projects, the original staffing level is sufficient to guarantee the project will not be delayed because of resource conflict. Therefore,



(a) Staff Absence Rate=0.001

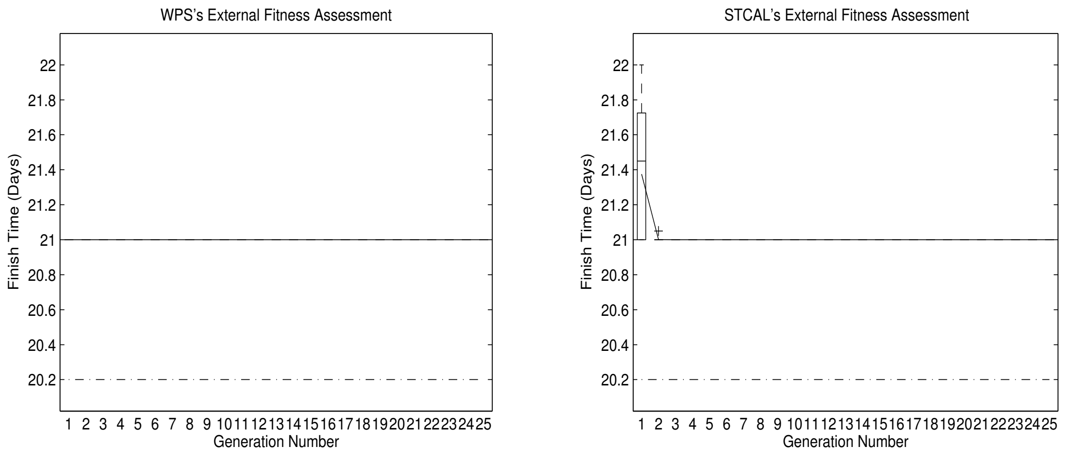


(b) Staff Absence Rate=0.1

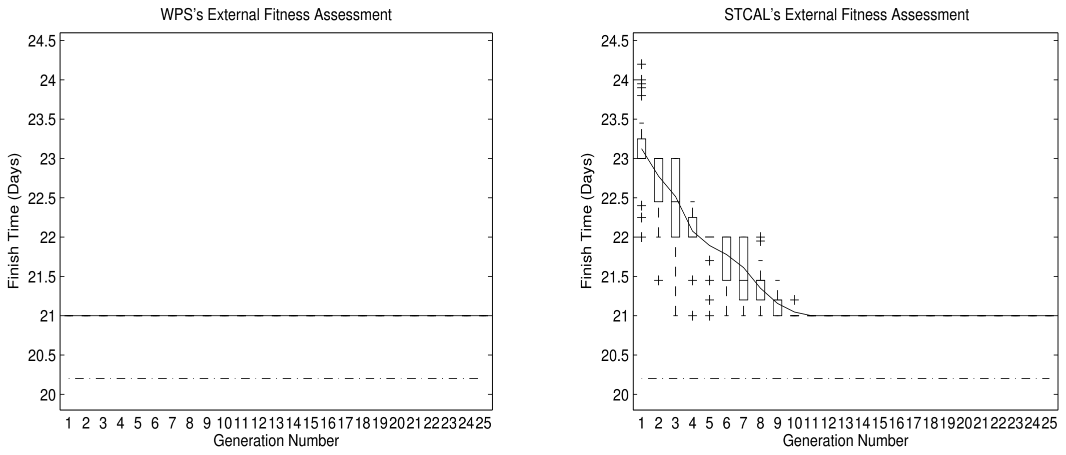


(c) Staff Absence Rate=0.25

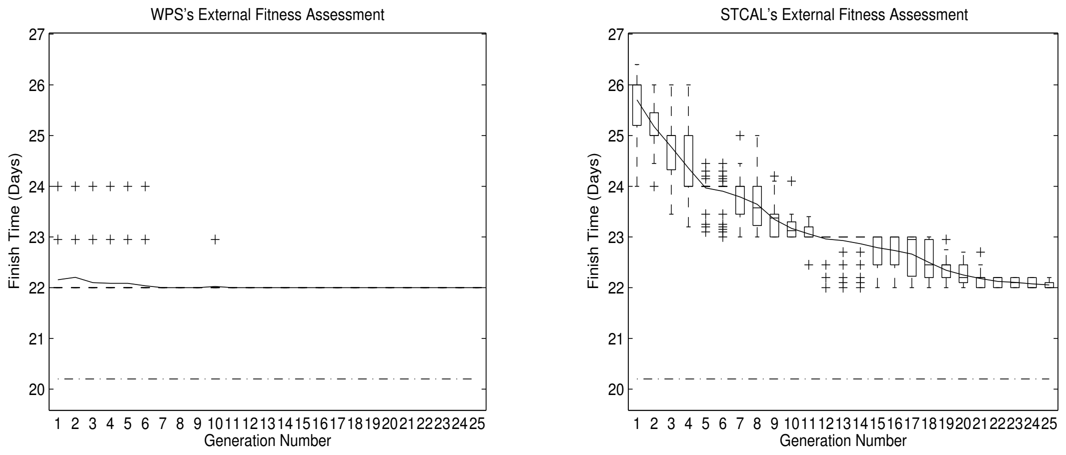
Figure 6.10: External fitness of solutions by Configuration BWBS on Project C. The solutions for two species are plotted separately in two side-by-side sub-figures, with the WPOs on the left and STCALs on the right. The sub-figures are arranged in three rows according to their levels of the absence rate equal to 0.001, 0.1, and 0.25 respectively. The finish time of the entire population is plotted for each generation along the co-evolutionary process. A solid line on the boxes is the mean value of each generation. The duration of executing the critical path for each project is plotted accordingly as a dashed line near the bottom of each sub-figure.



(a) Staff Absence Rate=0.001

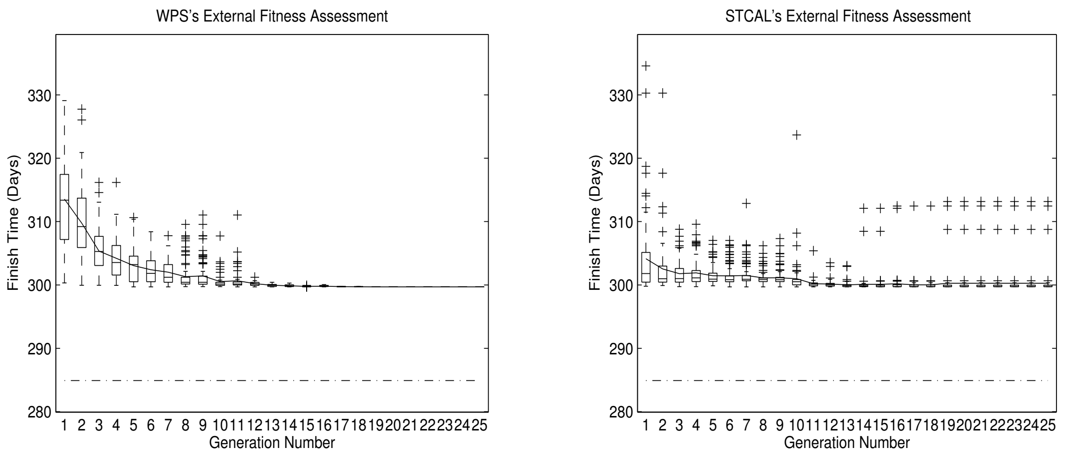


(b) Staff Absence Rate=0.1

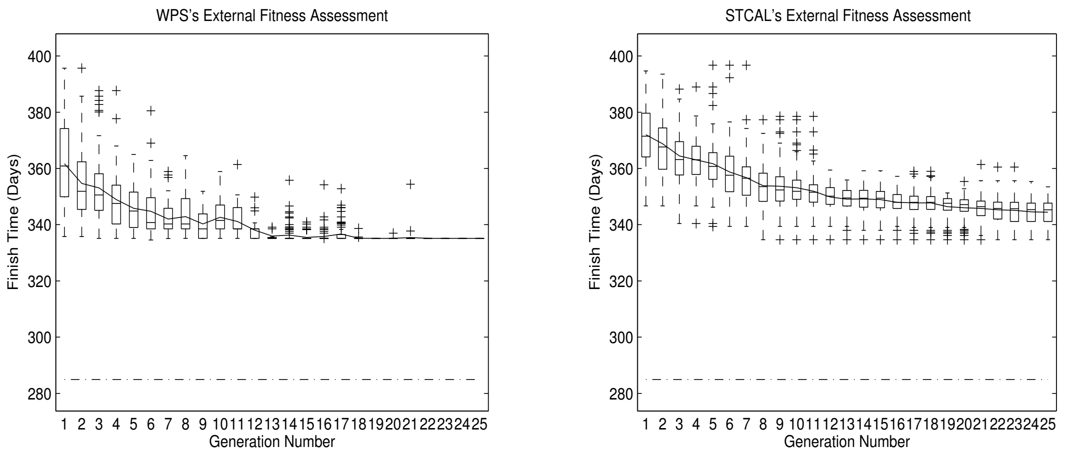


(c) Staff Absence Rate=0.25

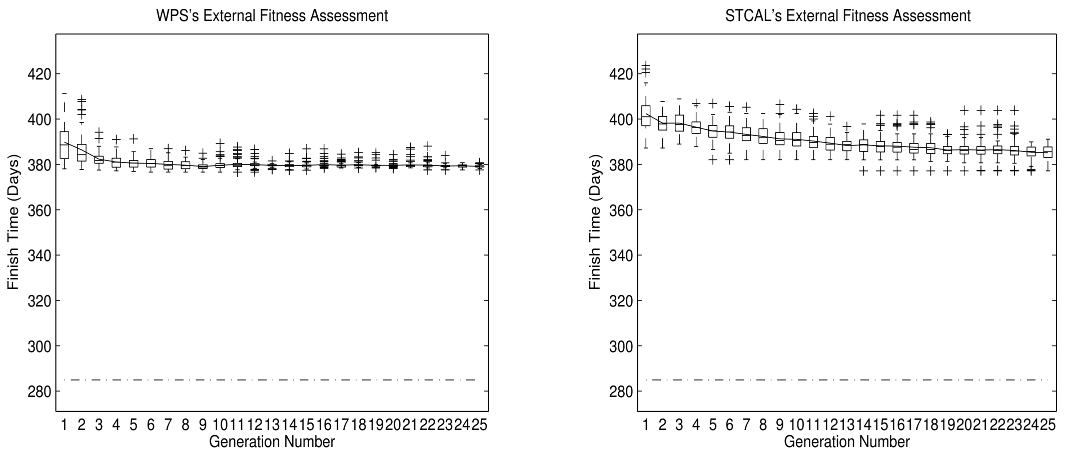
Figure 6.11: External fitness of solutions by Configuration BWBS on Project D



(a) Staff Absence Rate=0.001

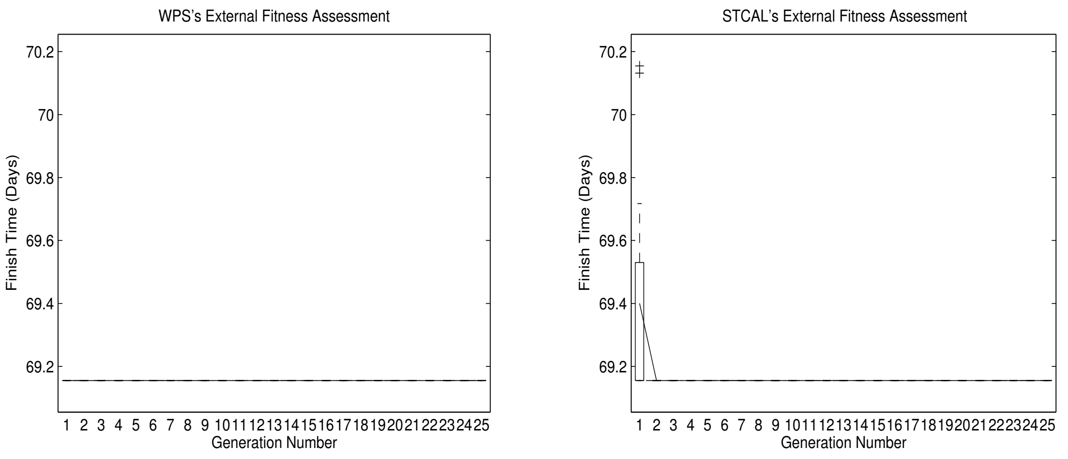


(b) Staff Absence Rate=0.1

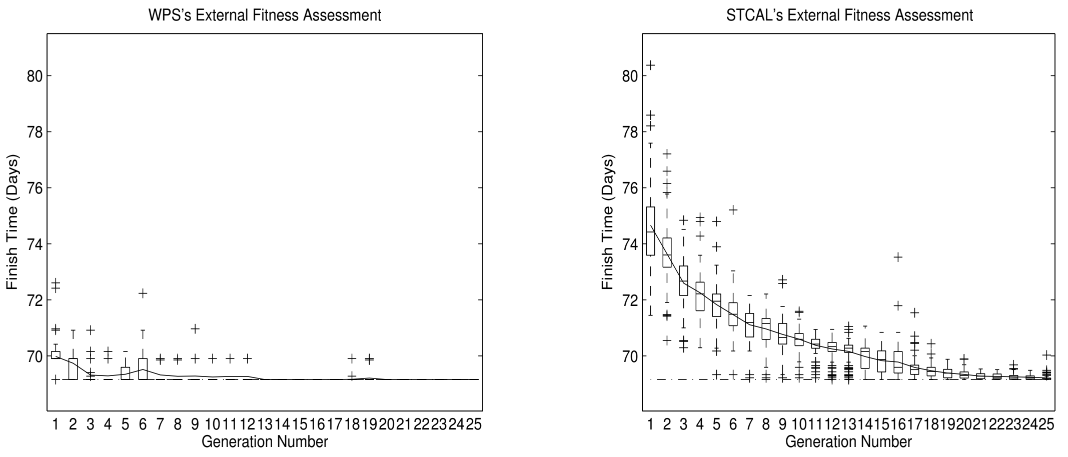


(c) Staff Absence Rate=0.25

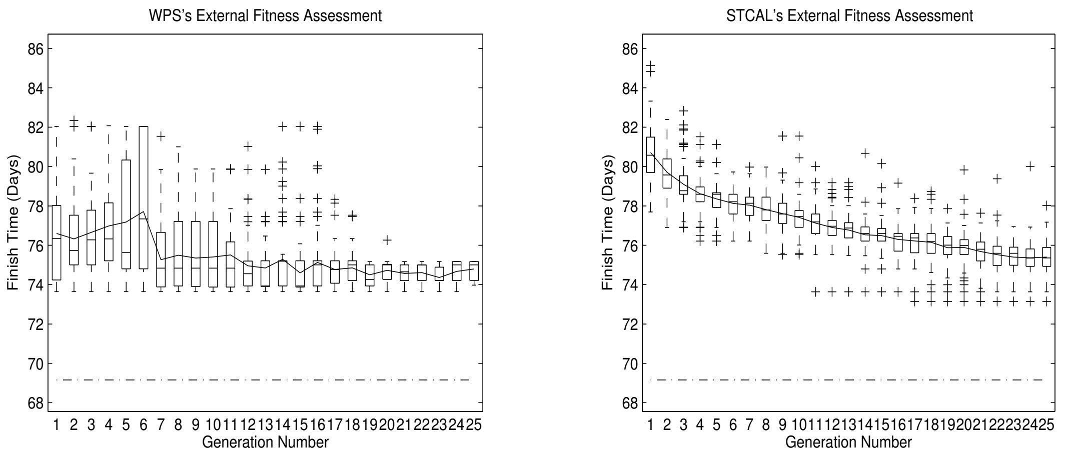
Figure 6.12: External fitness of solutions by Configuration BWBS on Project E



(a) Staff Absence Rate=0.001



(b) Staff Absence Rate=0.1



(c) Staff Absence Rate=0.25

Figure 6.13: External fitness of solutions by Configuration BWBS on Project F

as shown in Figure 6.11 (a) and 6.13 (a), the best solutions found in the early stage of the evolutionary process are already close to or have already overlapped with the dashed line.

For Project D, the lower bound of the actual measured fitness values are considered to be the optimal value, although they cannot possibly overlap the dashed line, because of the limitation of the implementation in rounding small numbers.

An example of detailed analysis is demonstrated to further understand the reasons why Project D is simpler. As illustrated in Figure 3.3, *Project D* has a long critical path (31 out of 60 WPs are on the critical path), and more importantly only a couple of the non-critical path WPs require the same resources as required by the WPs on critical path. This means there are only a few very small time windows in which a resource conflict might occur. The project delay is increased only when a WP on the critical path cannot be processed due to resource conflict or absence. Therefore, a low chance of resource conflict and minimised resource absence will essentially minimise the possibility of a delay. From another perspective, no resource conflict means the execution of WPs can be highly parallel, in which case the chance of delay on the project is eventually minimised.

This unique feature (i.e., having a long critical path) of *Project D* makes the optimisation on WPO have significantly less impact on the final project duration. This is because two very different scheduling orders of WPs make much less difference in the project completion time if the executions of WPs can be highly parallel.

6.5.5 Answers to the Research Questions and Proposals to the PM

Given the discussions on the observations above, the research questions can be answered as follows, and proposals to the software project manager are made accordingly for the purpose of taking full advantage of the co-evolutionary optimisation in practice.

RQ1: How do co-evolutionary optimisation techniques find desirable extreme solutions for the best and worst case scenarios?

Answer 1: As demonstrated in **Observation I**, comparing the results from all the configurations of the project simulations, the cooperative searching Configurations BWBS and WWWS are able to find the best and worst solutions.

Proposal 1: Because it is assumed that no compensations are made to cancel the optimisation efforts devoted by the other species in the cooperative co-evolution process, the best and worst solutions found by cooperative search (BWBS and WWWS) can be proposed to be used as the lower bound and upper bound of the execution time of project. This enables the project manager to know the lower bound and upper bound of a project's duration based on a given degree of staff absence rate, and essentially provides a great deal of advantage in the negotiation with shareholders on the delivery deadline.

RQ2: How do the co-evolutionary optimisation techniques reveal the dynamic correlation between workpackage scheduling and staff absence rate?

Answer 2: The competitive search process is useful to find the dynamic correlation between work package scheduling and staff absence for each individual project. **Observation II** shows that it can identify the key problem to be solved by revealing the dominating party in the optimisation of both WPO and STCAL.

Proposal 2: The Configuration BWWS are proposed to reveal whether or not it is possible to shorten the project completion time by optimising the work package scheduling under a certain degree of staff absence. This aid is particularly helpful in the case of predictable staff absence. Because it gives the project manager a good reference to guide the decision as to whether he/she should devote more effort in managing the work package scheduling to avoid additional cost spent on acquiring extra staff.

RQ3: How does the staff absence rate interfere with the project completion time?

Answer 3: The results not only confirm the common scenario that a higher degree of staff absence rate leads to longer project completion time, but they also also reveal that the complexity of the project, as discussed in **Observations III(a)** and **III(b)**, is also found to be a very important factor that needs to be taken into consideration.

Proposal 3: Although it is always safe for the project manager to assume that a higher degree of staff absence will consequently lead to more delay on the project, this automatic tool makes further attempts to quantify the degree of consequent delay to provide project manager with additional insightful knowledge about the project's complexity.

6.6 Measuring the Absence Rate

The following three techniques are the most commonly used methods for quantitatively measuring the degree of absence [Seccombe, 1995,CIPD, 2011]. They focus on different aspects of measuring the lost of workforce.

Absence Rate shows the total duration of all employees' absence in proportion to the total duration of all available contracted working time in the same period.

$$Absence\ Rate = \frac{Total\ Duration\ of\ All\ Absence}{Total\ Contracted\ Working\ Time}$$

It gives the average length of all absence within a group or organisation, but no information on the frequency of absence is given.

Frequency Rate measure provides the average number of absence events per employee.

$$Frequency\ Rate = \frac{Number\ of\ Spells\ of\ Absence}{Number\ of\ Employees}$$

It indicates the average frequency of absence, but it does not give any information about the length of each absence.

By measuring both the number of spells and lengths of absence for each individual employee, the ***Bradford Factor*** provides an indication of the degree of absence for each employee.

$$Bradford\ Score = S^2 \times D$$

where:

$$S = Number\ of\ Spells\ of\ Absence$$
$$D = Total\ Duration\ of\ Absence$$

Considering two employees who have the same total duration of absences, this indicator is able to highlight the one with more frequent absences which are relatively shorter.

6.7 Summary

This work introduces a co-evolutionary approach to software project scheduling and staffing problems with respect to analysing the uncertainties due to staff absences. While the solutions to these two problems are automatically co-evolved, three levels of staff absence are injected to simulate the scenarios of almost zero, normal, and high staff absence rate. In order to analyse the impacts of staff absence and the algorithm's ability to find good solutions, four co-evolutionary configurations have been applied to real-world software project planning data. In addition to our previous work on applying co-evolutionary techniques on software project management problem, the problem model is redefined in a way that the skill requirement associated between staff and work package is taken into consideration. The results of the simulations are analysed in detail with comprehensive discussions on the observations.

Through the empirical studies of four sets of real-world software projects, it has been revealed that co-evolutionary techniques can be helpful in providing the project manager with insightful information on the project: 1) by cooperatively searching for either shorter or longer finish time on both scheduling and staffing problem, it can provide the project manager with a good reference of the lower bound and upper bound of a project finish time. 2) in competitively searching, by observing the dynamic correlation between the solutions of the two problems, it helps the project managers identify the dominating problem. 3) by analysing the correlation between staff absence rate and the extreme solutions obtained, it can help the project manager with an insightful understanding on the non-linear correlation.

Chapter 7

Conclusions and Future Works

"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."
Abraham H. Maslow (1966), The Psychology of Science

This thesis aims to advance the state of the art in the application of SBSE techniques to three software project management problems at the stages of project initiation, planning and enactment. As though it were the only tool we have, the application of SBSE technique has been exploited in this thesis. However, it is useful to exploit the metaphor of hammers and nails to understand which 'nails' have been hammered with which 'hammers' as the results of the findings of this thesis.

Table 7.1: Thesis Summary

	Three different stages of a software project		
	Initiation	*Planning*	*Enactment*
Challenges	Information is not yet fully available.	What is the right team size?	Execution deviates from the plan.
Nails	Mis-estimations on requirements' cost	Project Staffing and Scheduling	Unplanned Staff Absence
Hammers	Sensitivity Analysis and NSGA-II on NRP	Cooperative Co-evolutionary Optimisation	Co-evolutionary Optimisation
Improvements	Focus on analysing error-sensitive requirements & budgets	"Co-optimise" two inter-related problems	Study various "what-if" scenarios
Advantages	Gain more accurate estimations	Achieve the global optimum	Anticipate and ameliorate the impact of staff absence
In Chapters	Chapter 4	Chapter 5	Chapter 6

As summarised in Table 7.1, the thesis first proposes an SBSE sensitive analysis to achieve more accurate estimations at the stage of requirement selection. Secondly, the cooperative co-evolutionary algorithm is applied to effectively co-optimise project staffing and scheduling problems at the stage of project planning. Finally, at the stage of project enactment, the co-evolutionary model is extended to study the impact of staff absence on project completion time.

7.1 Summary of Contributions

The major contributions of this thesis are summarised as follows:

Search based data sensitive analysis applied to requirement engineering

Through the empirical studies of both synthetic and real-world requirement data, the work presents a statistical analysis that confirms the Positive Correlation Assumption, i.e. more expensive requirements and higher level of inaccuracies tend to have greater impact on NRP. However, the heat-map visualisation also reveals that there exist exceptions to this assumption. Identifying these exceptions can guide the decision maker towards more accurate estimation and safer decision making.

Co-evolving software project staffing and scheduling

It is the first time that the Cooperative Co-evolutionary Algorithm was introduced to the field of SBSE for solving software project staffing and scheduling problems. We conducted an empirical study using data from four industrial software projects, aimed at comparing CCEA project planning and staffing with (i) random search and (ii) single population optimisation using genetic algorithms. Results of the empirical study show that CCEA is able to outperform random search and single population GA, in terms of effectiveness (i.e., best solutions proposed in terms of project completion time) and efficiency (i.e., a smaller number of evaluations required).

Co-evolutionary project planning optimisation under staff absence

This work introduces a co-evolutionary approach to software project scheduling and staffing problems with respect to analysing the uncertainties of the staff absences. In addition to our previous work on applying co-evolutionary techniques on software project management problems, the problem model is redefined in a way that the skill requirement associated between staff and work package is taken into consideration. Through the empirical studies of four sets of real-world software projects, it has been revealed that co-evolutionary techniques can be very helpful in providing the project manager with insightful information on the project: 1) by cooperatively searching for either shorter or longer finish time on both scheduling and staffing problems, it can provide the project manager with a good reference of the lower bound and upper bound of a project finish time. 2) in competitively searching, by observing the dynamic correlation between the solutions of the two problems, it helps the project managers identify the dominating problem. 3) by analysing the correlation between staff absence rate and the extreme solutions obtained, it can help the project manager with an insightful understanding of the non-linear correlation.

Establishment of an SBSE project management tool - Amphisbaena

During the course of the research, several functions of our automated SBSE project management tool were developed. Amphisbaena provides not only proactive analysis on staffing and scheduling, but also reveal great insights on the project itself.

7.2 Summary of Future Work

In this thesis, a number of new SBSE techniques are introduced for software project management, and new tools are developed to evaluate the advancements. This promises further investigation and more empirical studies to thoroughly explore their advantages and find the best scenario to apply them. Promising future work includes:

To consider more complex aspects of the Next Release Problem. It is expected that more detailed studies in the future will be conducted, and it is encouraging to apply sensitivity analysis procedures to more real-world multi-objective optimisation problems, such as to consider the complex dependency relation between requirements, and to further conquer the scalability issue of performing the global sensitivity analysis method.

To extend the usage of Co-evolutionary Algorithms to more sophisticated software project management models. Through thorough empirical studies on a total of six sets of real-world software project data, this thesis demonstrates the suitability of co-evolution as a methodology for solving software project staffing and scheduling problems. Future work will aim at extending this with further data sets and, above all, at considering a more sophisticated project model, which accounts for further factors not considered in this study, such as issues of developers' non-uniform absence rates and performance inconsistency, training overhead, communication overhead, requirement changes, and schedule robustness.

To enrich and make available the SBSE management tool - Amphisbaena. As discussed, Amphisbaena can provide proactive project planning analysis. Most importantly, our tool establishes the connections between research studies and practical executions of software projects, because it applies SBSE approaches directly and automatically on Microsoft® Project Plan (`*.mpp`) file, which is one of the dominant project management software in practice. It is expected to attract more plan data sets on the tool once it becomes available to a wider range of industry partners.

7.3 Closing Remark

Software project managers are faced with the challenging task of managing the project process filled with uncertainties. In practice, they have to make decisions based on incomplete knowledge. Therefore, extra contingency plans or budgets are usually made, based on experiences, in the hope that all the deviations from plans can be absorbed. This thesis presents several SBSE techniques that can automatically provide various insights on the requirements, project team staffing and project scheduling. Provided with this computer-aided assistance, more control over specific factors becomes available to software project managers who will subsequently be able to confront problems with the abilities of identifying uncertainties, minimising risks, and preallocate contingency resource accordingly with more scientific guidance.

Appendix A

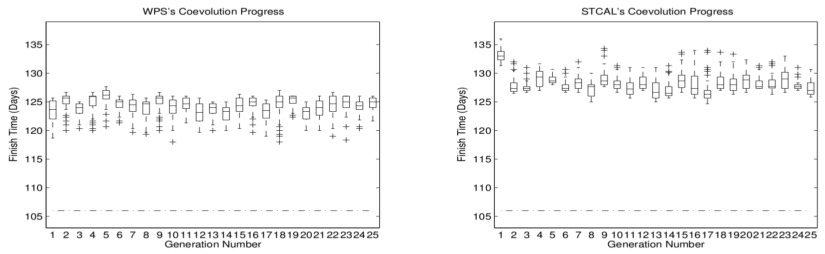
Results of Co-evolutionary Project Management Optimisation on Four Real–world Projects

This appendix contains the all the results of simulations produced in Chapter 6. All the results are firstly arranged into four sections (A.1, A.2, A.3, and A.4,) according to four different configurations (BWWS, WWBS, WWWS and BWBS). The results then are further grouped according to the four sets of industry data used.

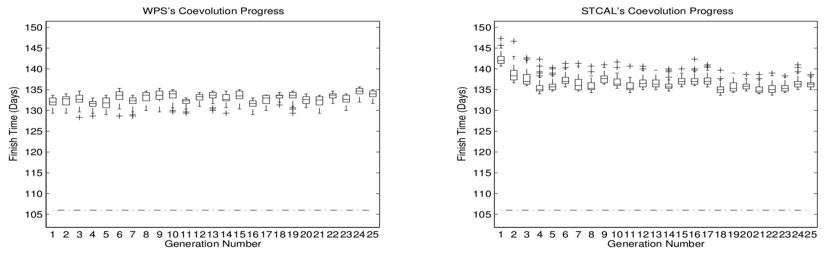
In each page, there are the twelve figures which represent the results for one particular real–world industrial project tested on one configuration of the algorithm running with three different staff absence rates. The six figures on the left hand side display the result of the Internal Fitness Evaluation, and the other six figures of the results of External Fitness Evaluation are plotted on the right. In the figures for either internal or external fitness evaluation, the results for both populations are plotted side by side with the WPO on the left and STCAL on the right.

A.1 Competitive Searching for Better WPOs and Worse STCALs

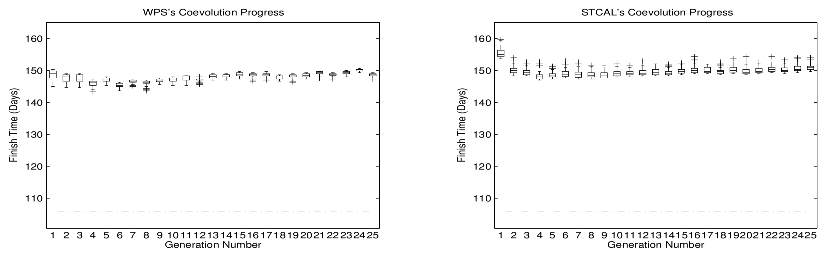
A.1.1 Project C (SoftChoice) – Config. BWWS (Competitive Searching for Better WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

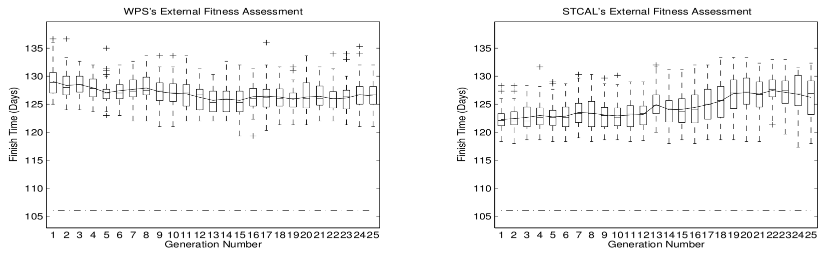


(b) Staff Absence Rate = 0.1

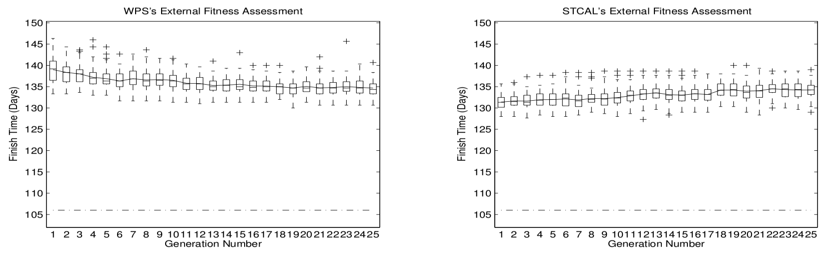


(c) Staff Absence Rate = 0.25

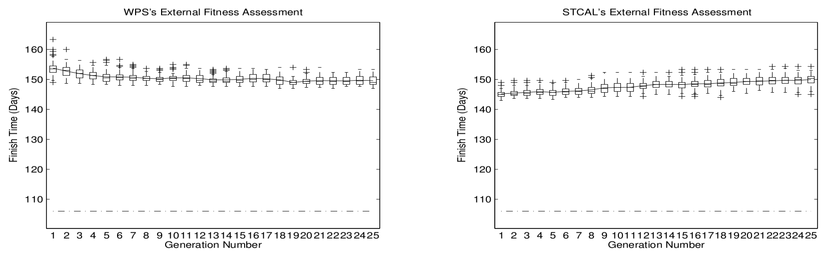
Figure A.1: C – BWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



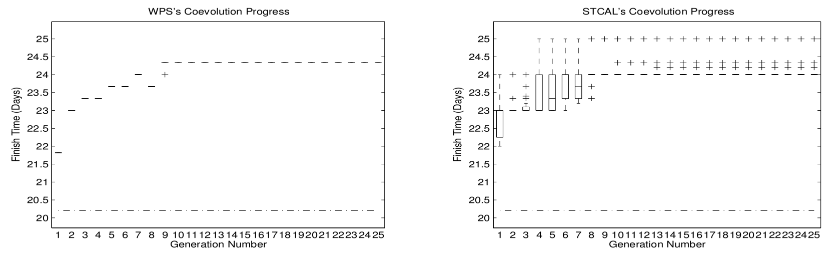
(b) Staff Absence Rate = 0.1



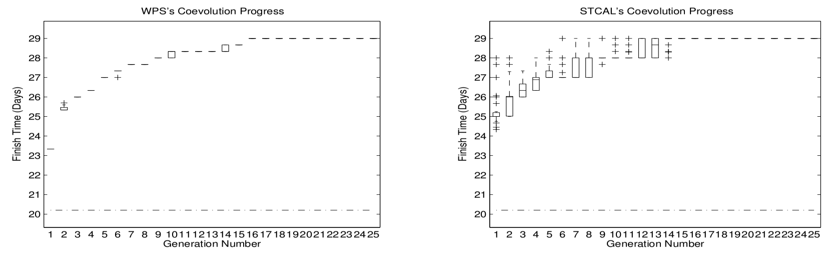
(c) Staff Absence Rate = 0.25

Figure A.2: C – BWWS – External Fitness

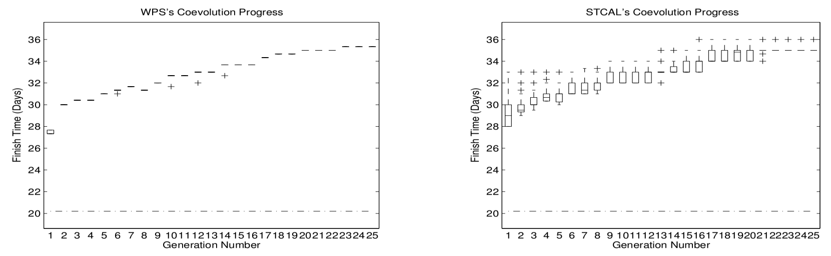
A.1.2 Project D (QuoteToOrder) – Config. BWWS (Competitive Searching for Better WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

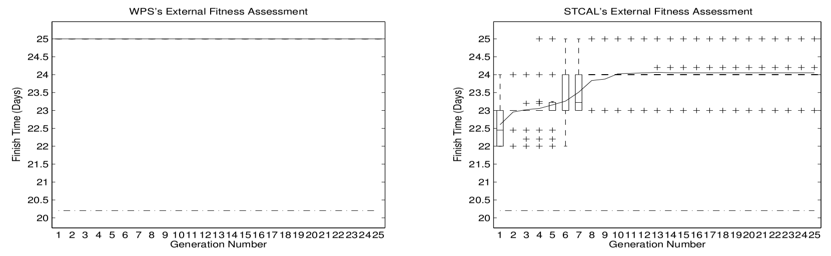


(b) Staff Absence Rate = 0.1

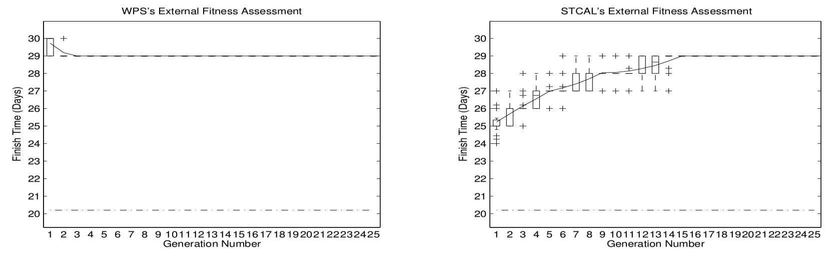


(c) Staff Absence Rate = 0.25

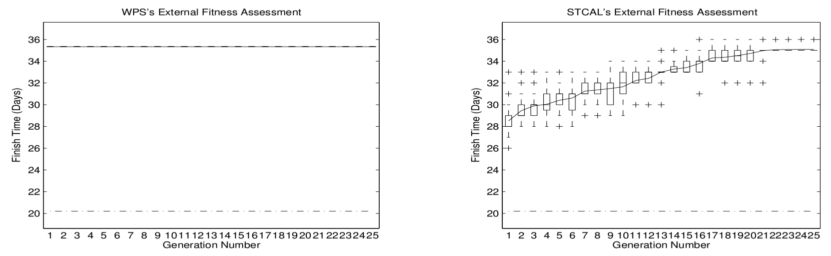
Figure A.3: D – BWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



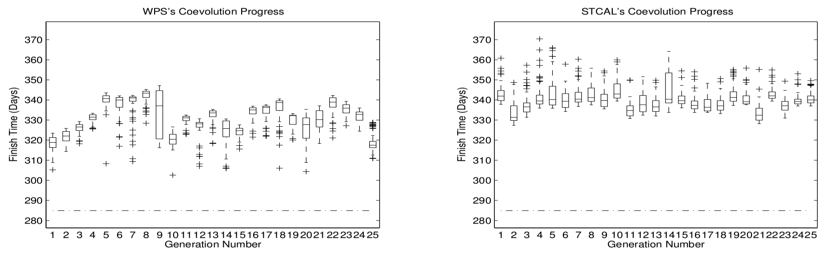
(b) Staff Absence Rate = 0.1



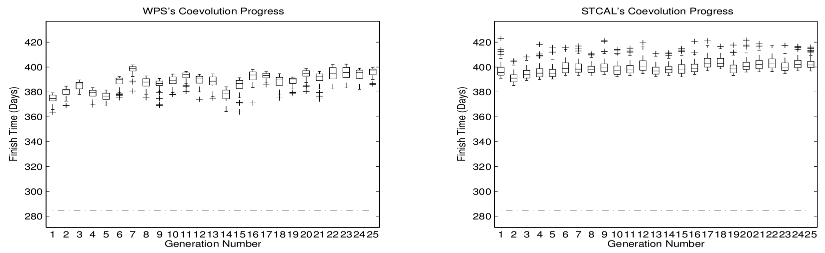
(c) Staff Absence Rate = 0.25

Figure A.4: D – BWWS – External Fitness

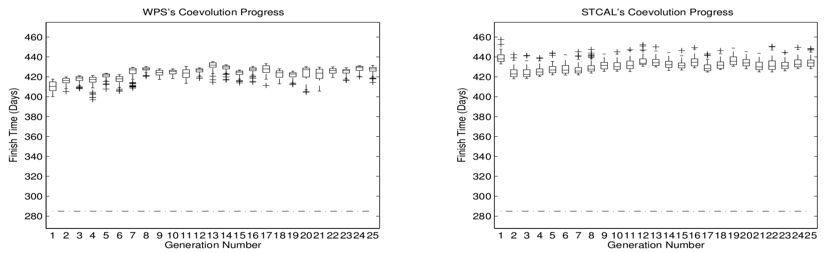
A.1.3 Project E (Database) – Config. BWWS (Competitive Searching for Better WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

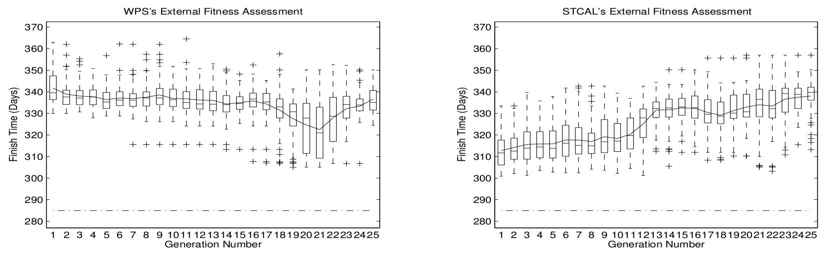


(b) Staff Absence Rate = 0.1

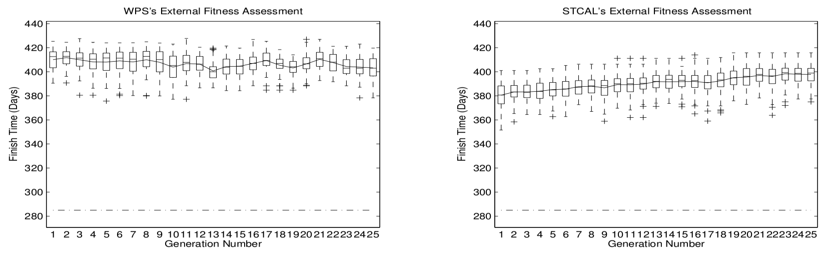


(c) Staff Absence Rate = 0.25

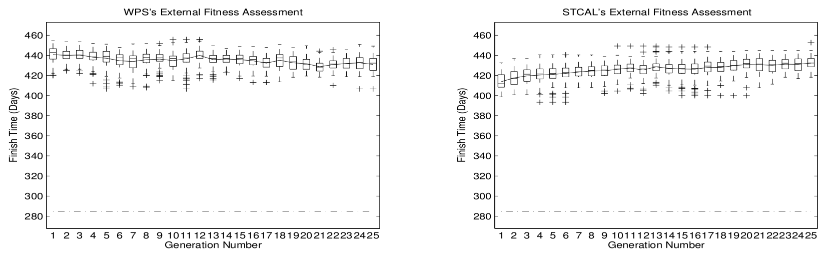
Figure A.5: E – BWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



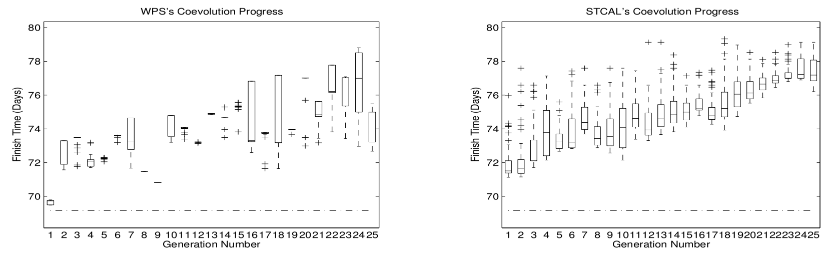
(b) Staff Absence Rate = 0.1



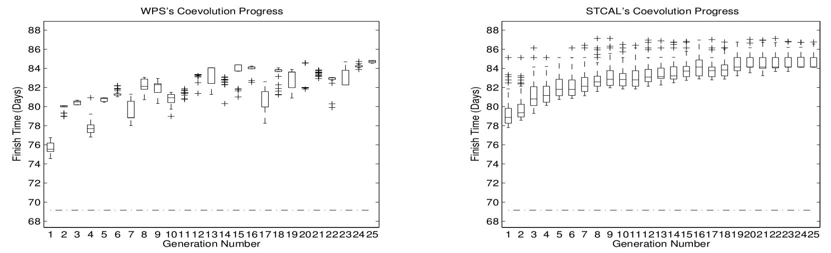
(c) Staff Absence Rate = 0.25

Figure A.6: E – BWWS – External Fitness

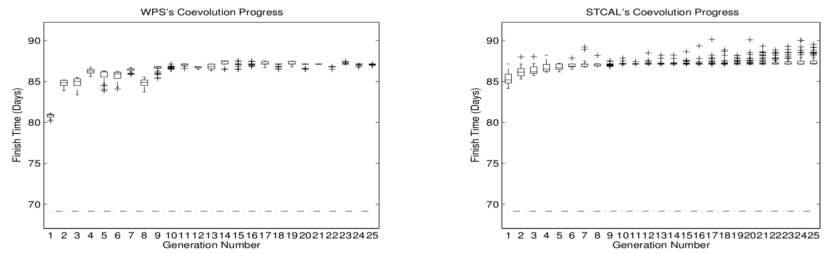
A.1.4 Project F (SmartPrice) – Config. BWWS (Competitive Searching for Better WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

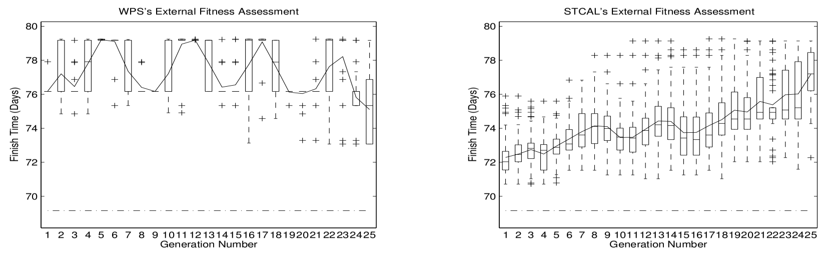


(b) Staff Absence Rate = 0.1

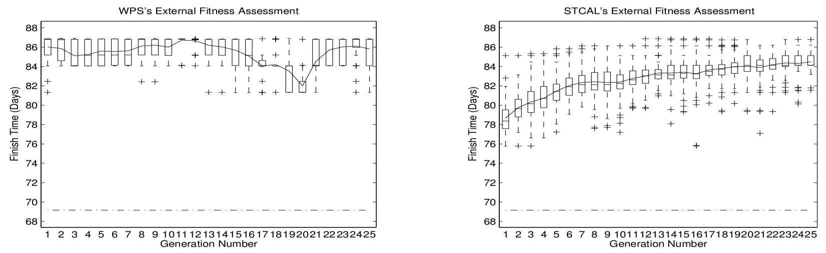


(c) Staff Absence Rate = 0.25

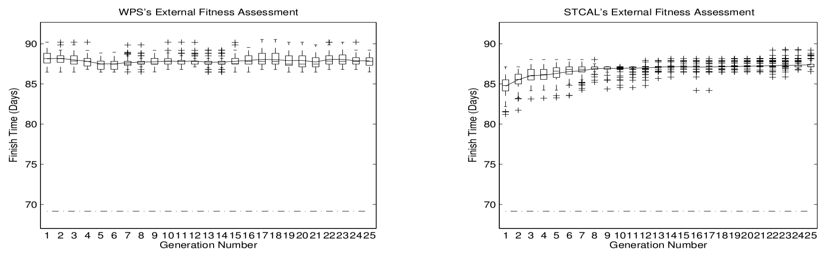
Figure A.7: F – BWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



(b) Staff Absence Rate = 0.1

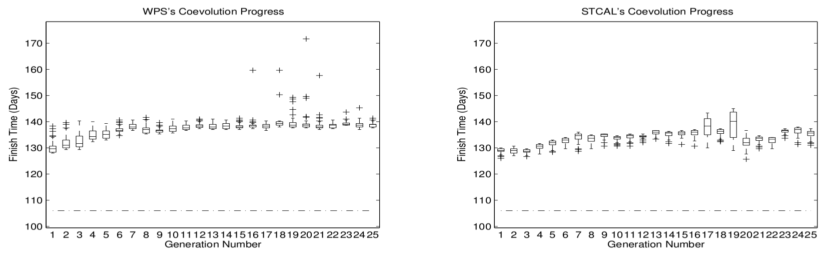


(c) Staff Absence Rate = 0.25

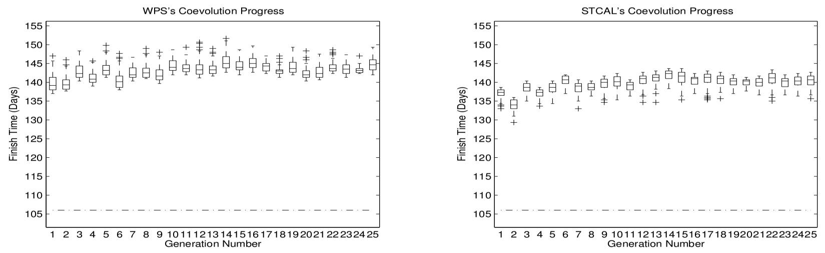
Figure A.8: F – BWWS – External Fitness

A.2 Competitive Searching for Worse WPOs and Better STCALs

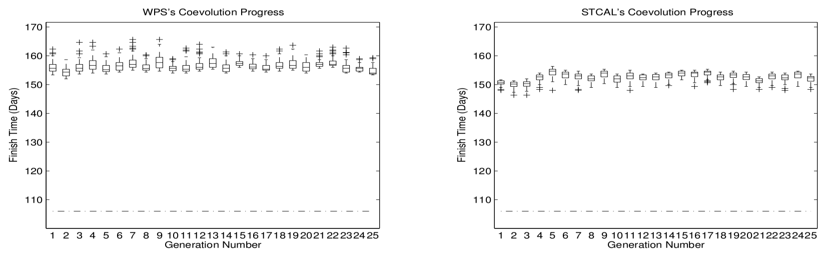
A.2.1 Project C (SoftChoice) – Config. WWBS (Competitive Searching for Worse WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

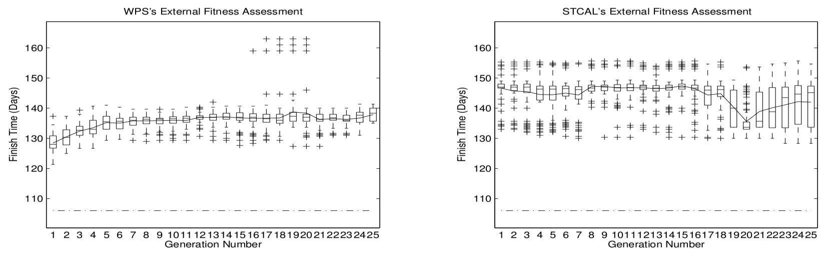


(b) Staff Absence Rate = 0.1

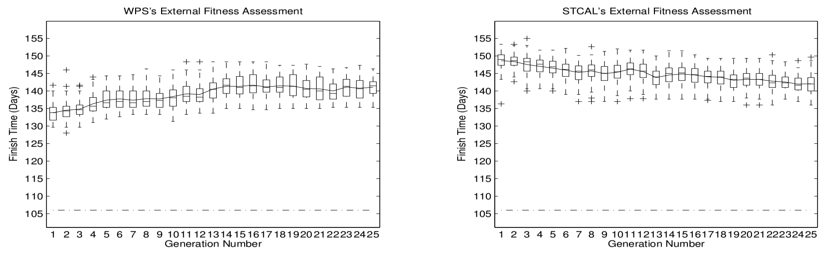


(c) Staff Absence Rate = 0.25

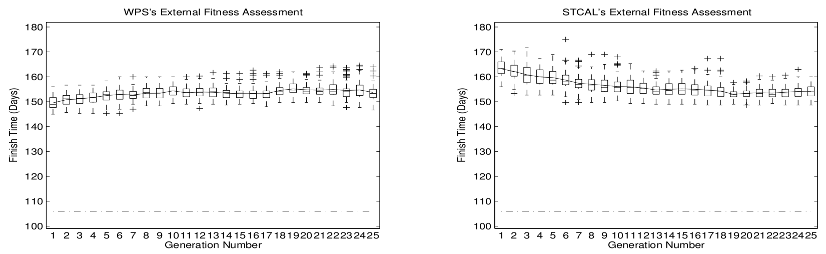
Figure A.9: C – WWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



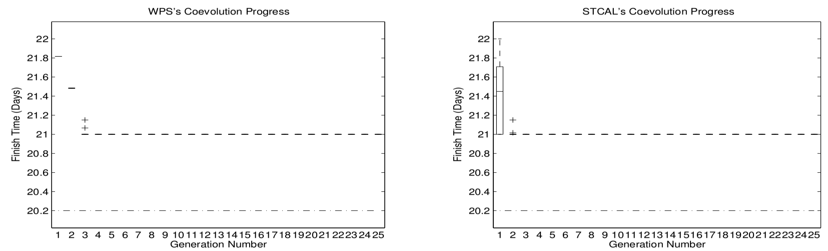
(b) Staff Absence Rate = 0.1



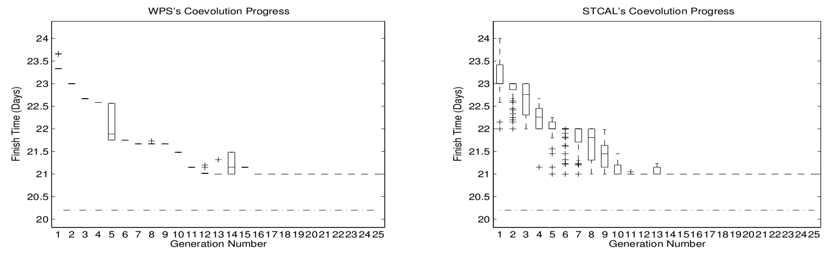
(c) Staff Absence Rate = 0.25

Figure A.10: C – WWBS – External Fitness

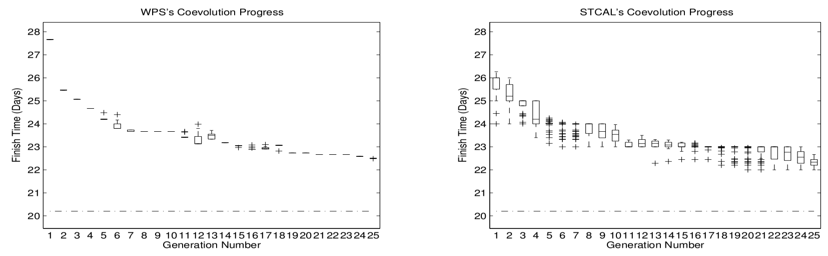
A.2.2 Project D (QuoteToOrder) – Config. WWBS (Competitive Searching for Worse WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

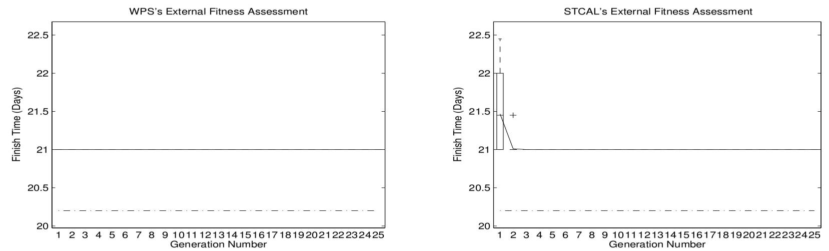


(b) Staff Absence Rate = 0.1

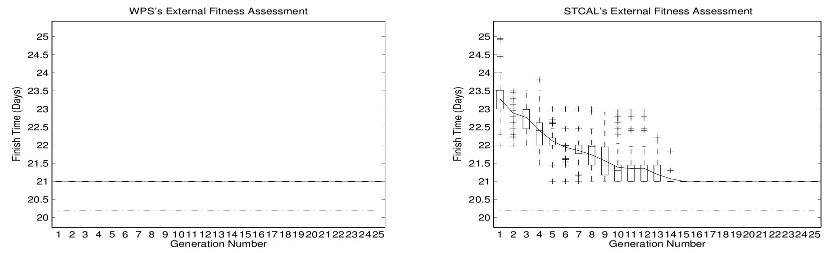


(c) Staff Absence Rate = 0.25

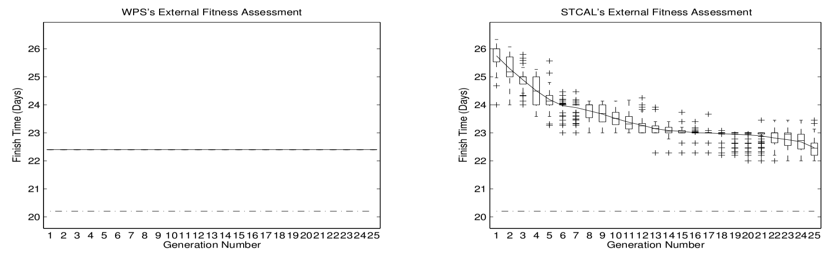
Figure A.11: D – WWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



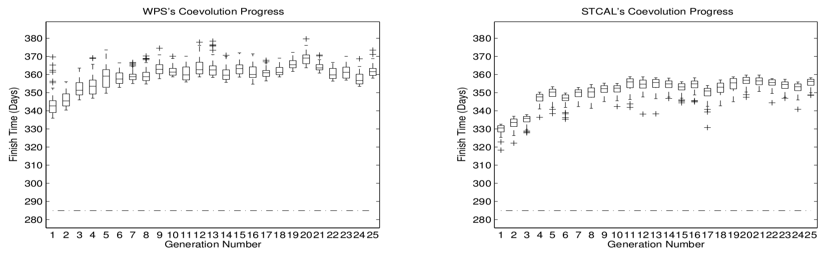
(b) Staff Absence Rate = 0.1



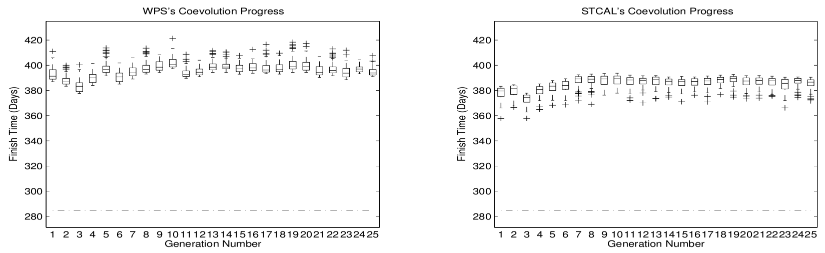
(c) Staff Absence Rate = 0.25

Figure A.12: D – WWBS – External Fitness

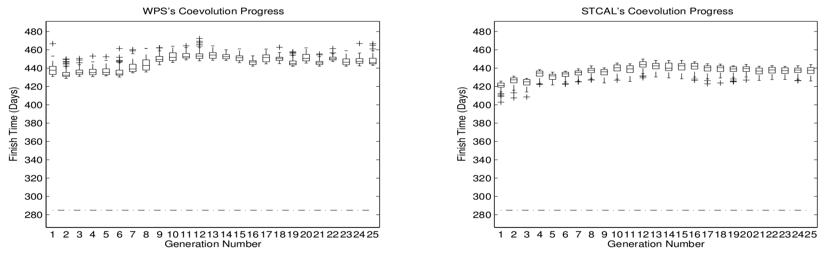
A.2.3 Project E (Database) – Config. WWBS (Competitive Searching for Worse WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

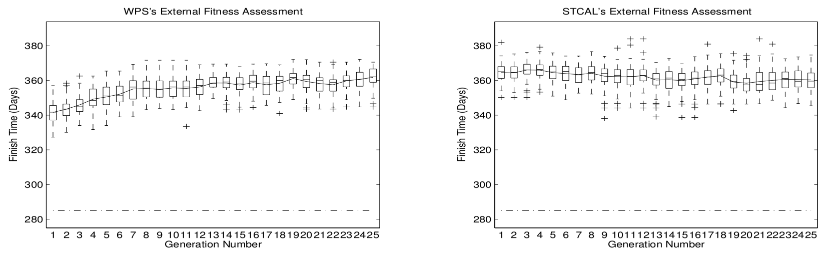


(b) Staff Absence Rate = 0.1

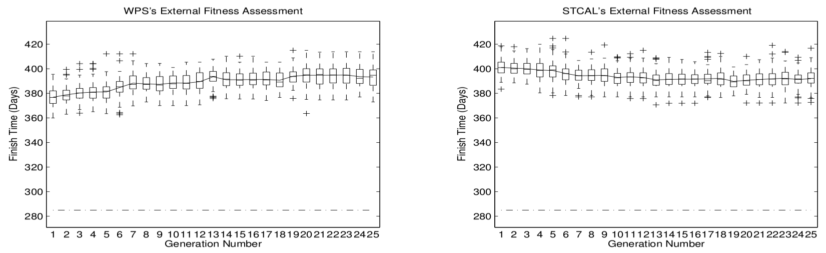


(c) Staff Absence Rate = 0.25

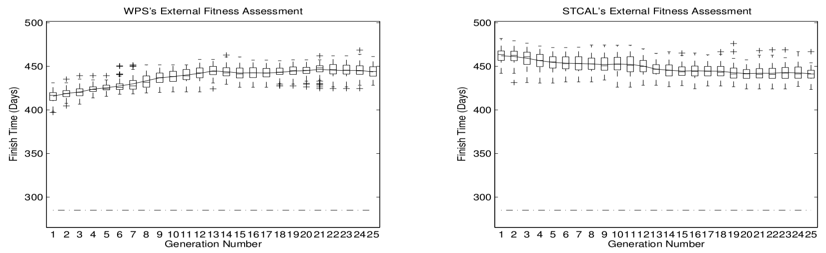
Figure A.13: E – WWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



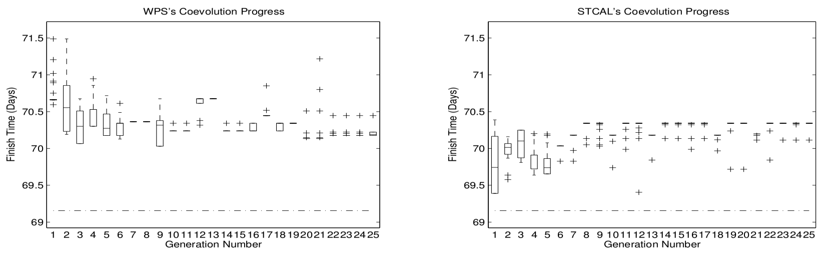
(b) Staff Absence Rate = 0.1



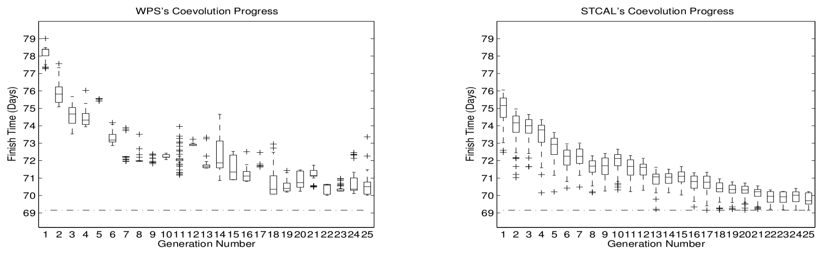
(c) Staff Absence Rate = 0.25

Figure A.14: E – WWBS – External Fitness

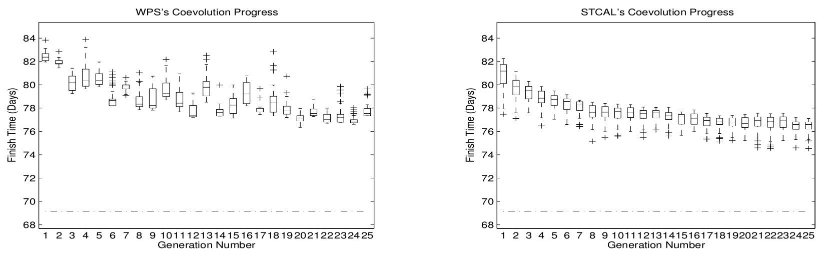
A.2.4 Project F (SmartPrice) – Config. WWBS (Competitive Searching for Worse WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

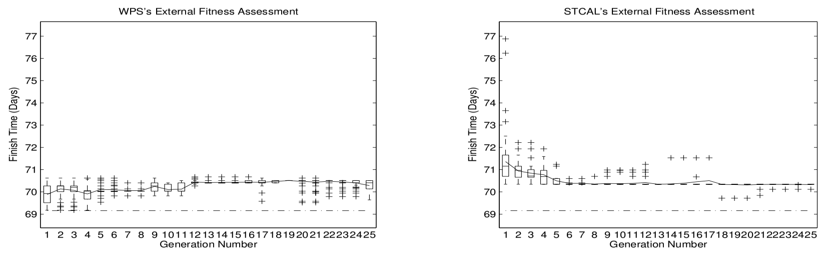


(b) Staff Absence Rate = 0.1

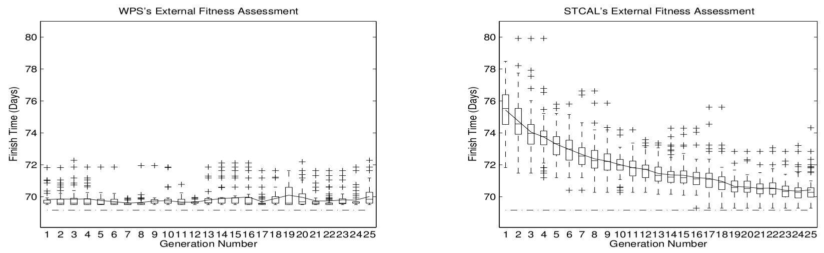


(c) Staff Absence Rate = 0.25

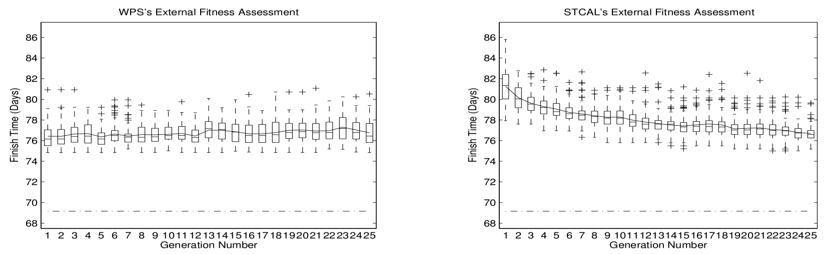
Figure A.15: F – WWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



(b) Staff Absence Rate = 0.1

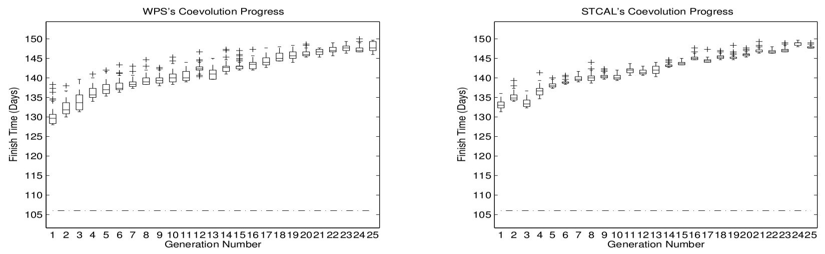


(c) Staff Absence Rate = 0.25

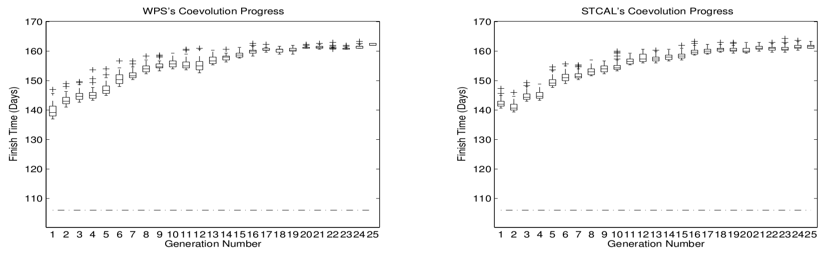
Figure A.16: F – WWBS – Internal Fitness

A.3 Cooperative Searching for Worse WPOs and Worse STCALs

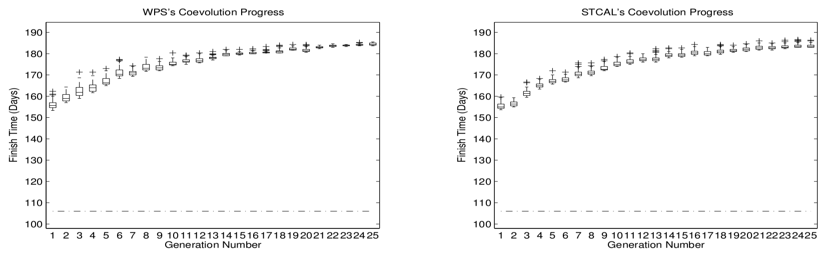
A.3.1 Project C (SoftChoice) – Config. WWWS (Cooperative Searching for Worse WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

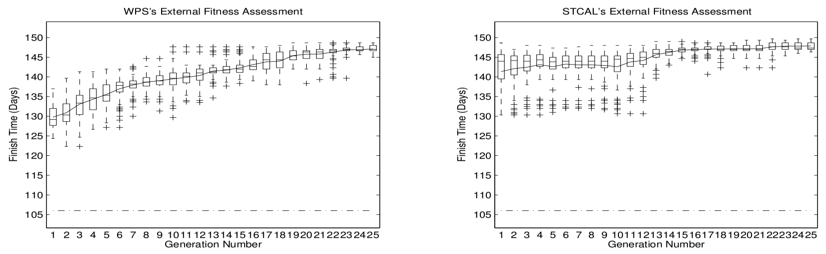


(b) Staff Absence Rate = 0.1

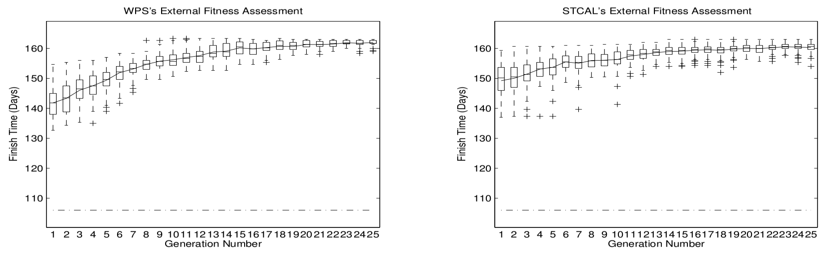


(c) Staff Absence Rate = 0.25

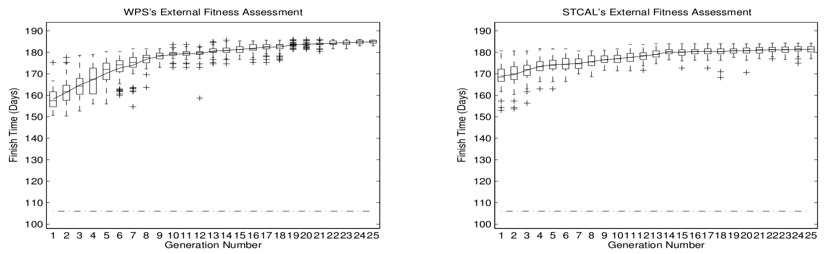
Figure A.17: C – WWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



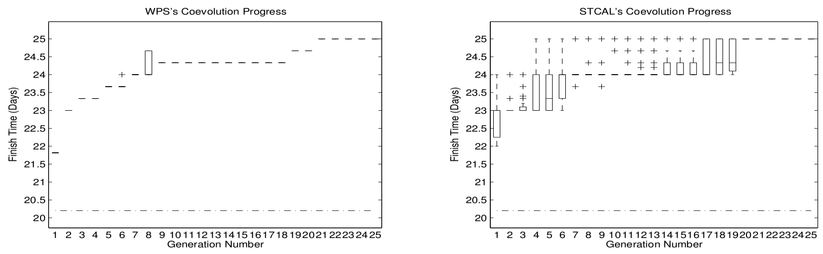
(b) Staff Absence Rate = 0.1



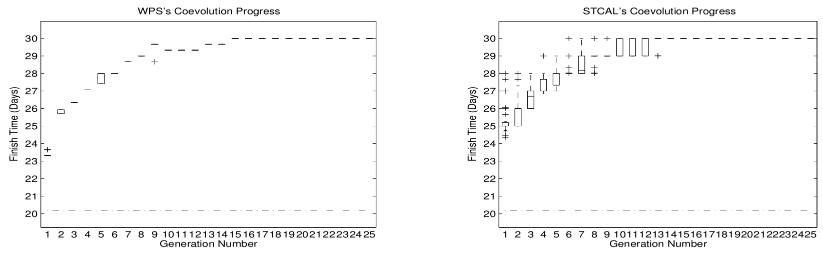
(c) Staff Absence Rate = 0.25

Figure A.18: C – WWWS – External Fitness

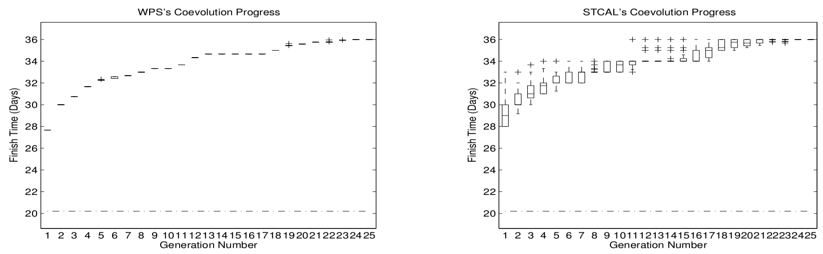
A.3.2 Project D (QuoteToOrder) – Config. WWWS (Cooperative Searching for Worse WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

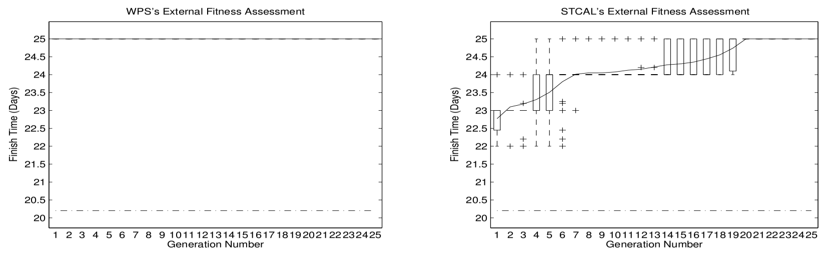


(b) Staff Absence Rate = 0.1

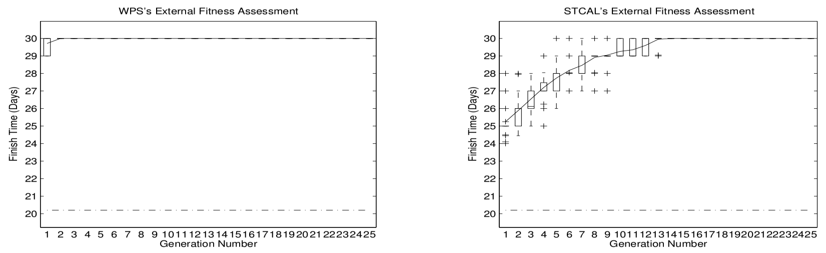


(c) Staff Absence Rate = 0.25

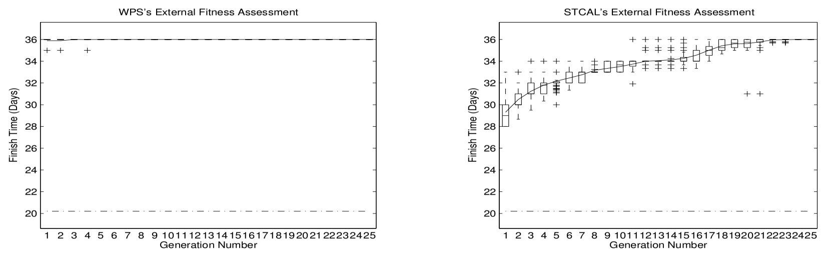
Figure A.19: D – WWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



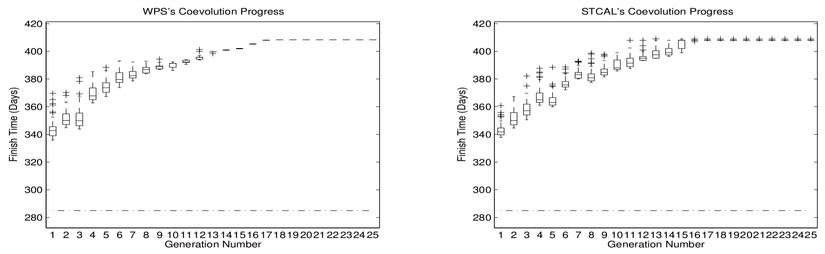
(b) Staff Absence Rate = 0.1



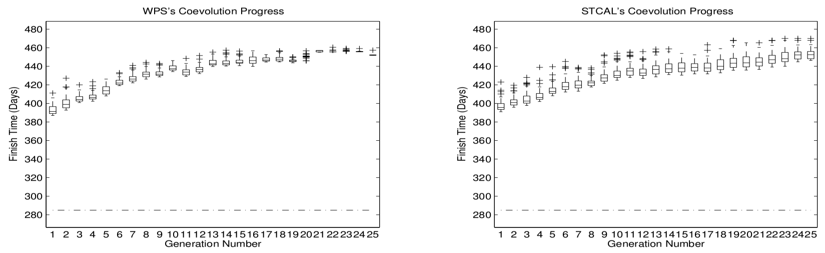
(c) Staff Absence Rate = 0.25

Figure A.20: D – WWWS – External Fitness

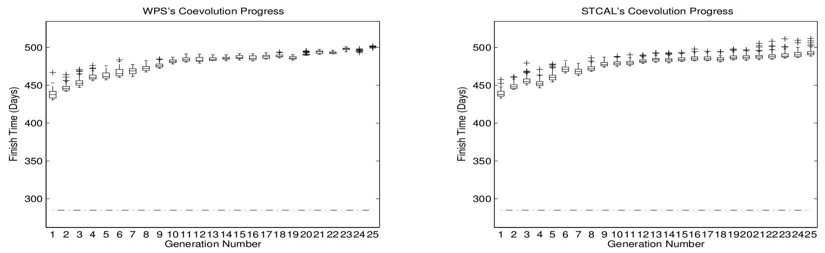
A.3.3 Project E (Database) – Config. WWWS (Cooperative Searching for Worse WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

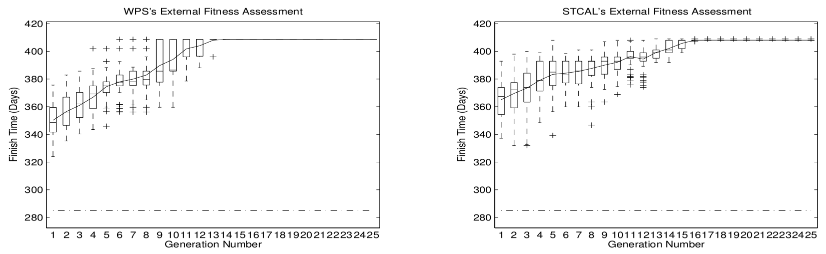


(b) Staff Absence Rate = 0.1

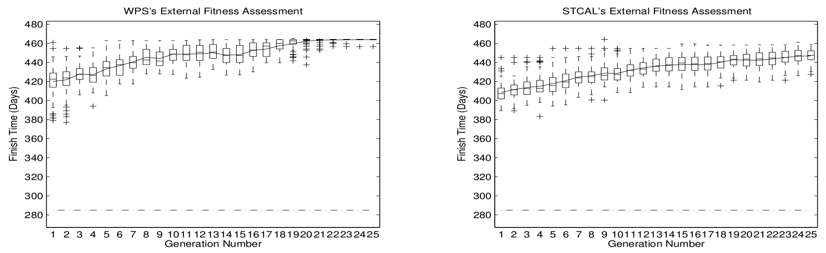


(c) Staff Absence Rate = 0.25

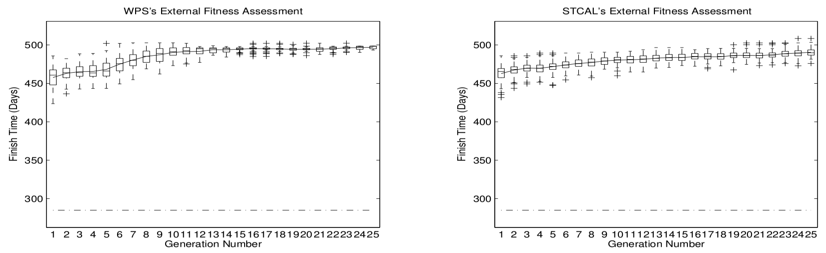
Figure A.21: E – WWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



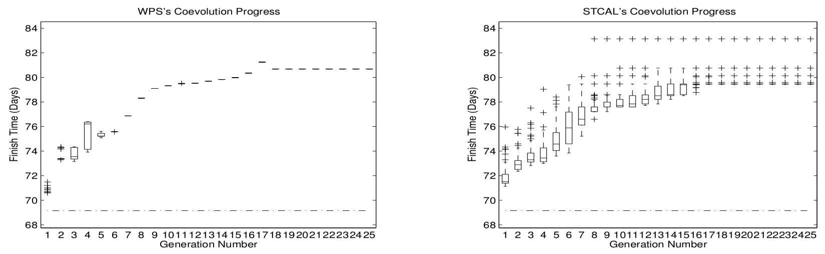
(b) Staff Absence Rate = 0.1



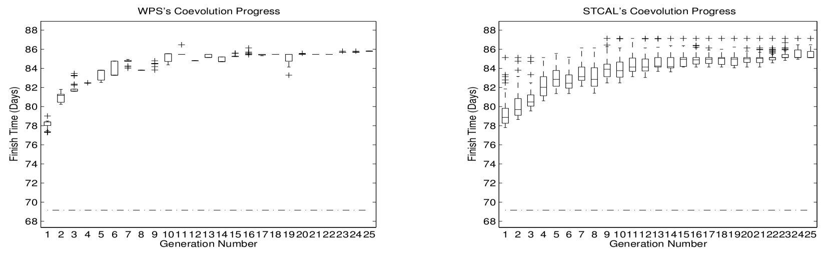
(c) Staff Absence Rate = 0.25

Figure A.22: E – WWWS – External Fitness

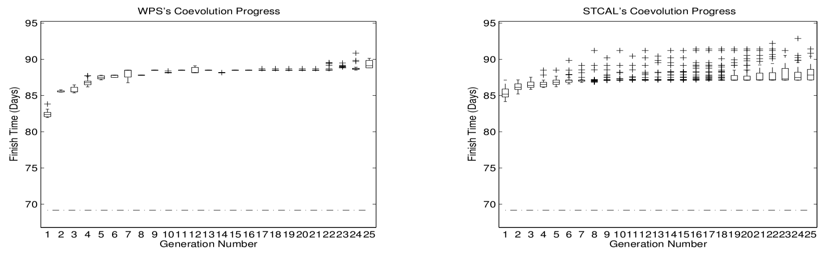
A.3.4 Project F (SmartPrice) – Config. WWWS (Cooperative Searching for Worse WPOs and Worse STCALs)



(a) Staff Absence Rate = 0.001

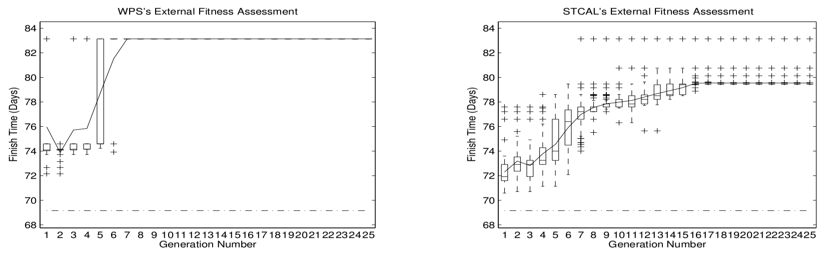


(b) Staff Absence Rate = 0.1

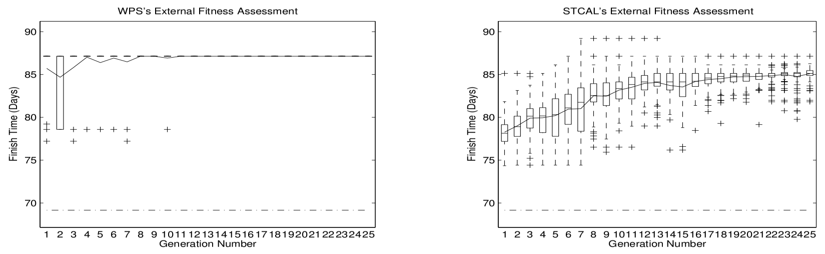


(c) Staff Absence Rate = 0.25

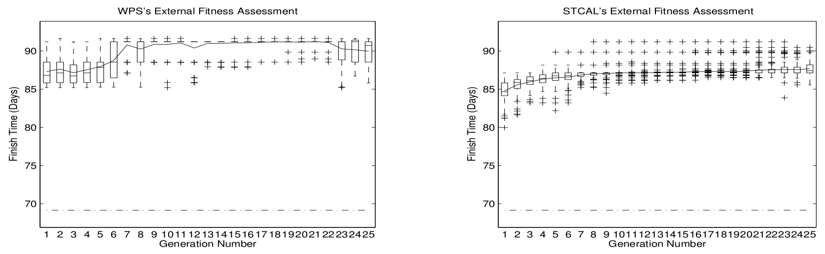
Figure A.23: F – WWWS – Internal Fitness



(a) Staff Absence Rate = 0.001



(b) Staff Absence Rate = 0.1

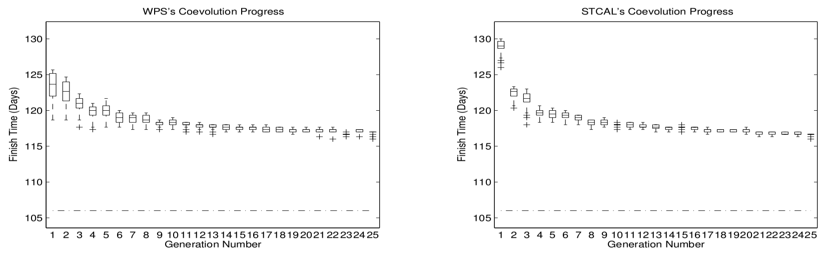


(c) Staff Absence Rate = 0.25

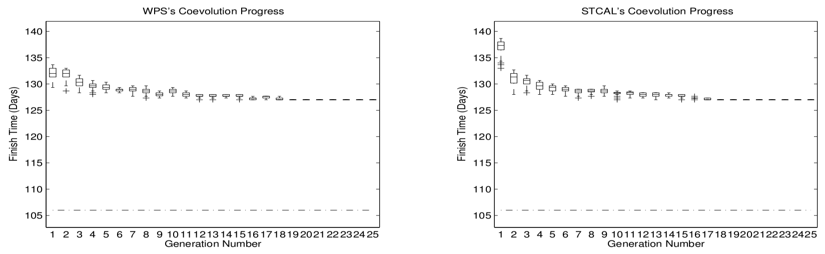
Figure A.24: F – WWWS – External Fitness

A.4 Cooperative Searching for Better WPOs and Better STCALs

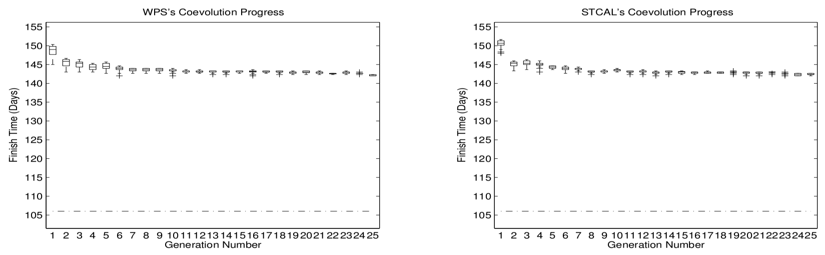
A.4.1 Project C (SoftChoice) – Config. BWBS (Cooperative Searching for Better WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

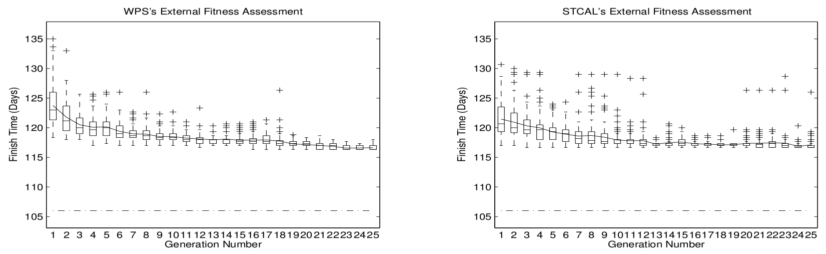


(b) Staff Absence Rate = 0.1

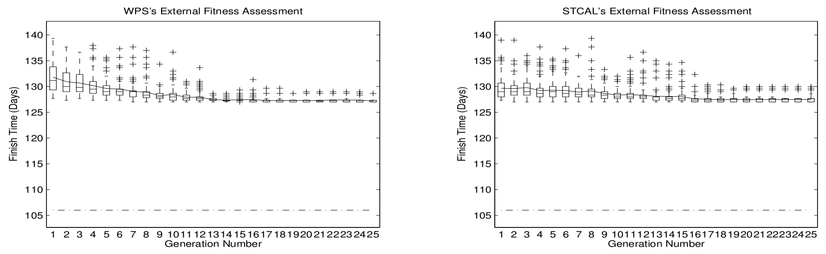


(c) Staff Absence Rate = 0.25

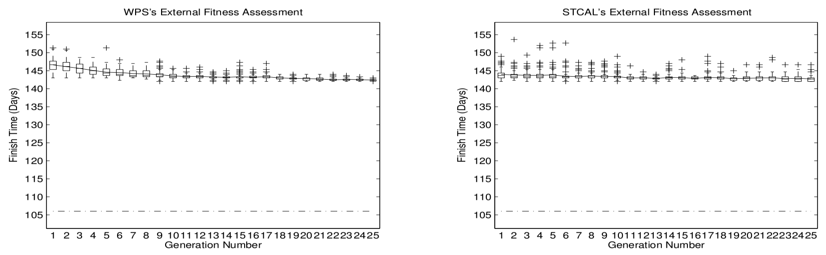
Figure A.25: C – BWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



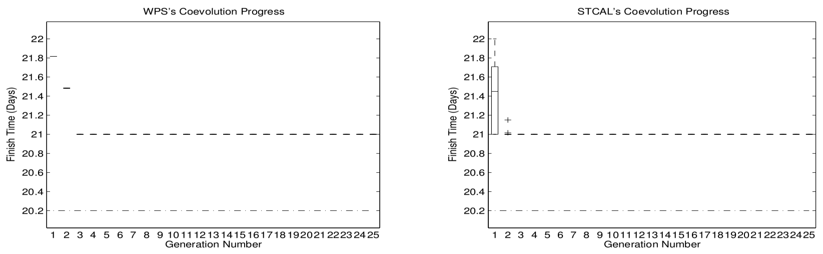
(b) Staff Absence Rate = 0.1



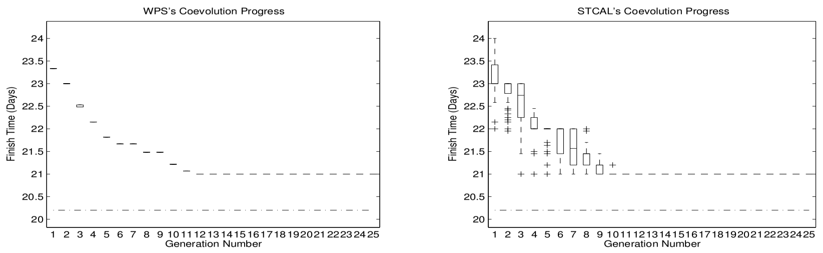
(c) Staff Absence Rate = 0.25

Figure A.26: C – BWBS – External Fitness

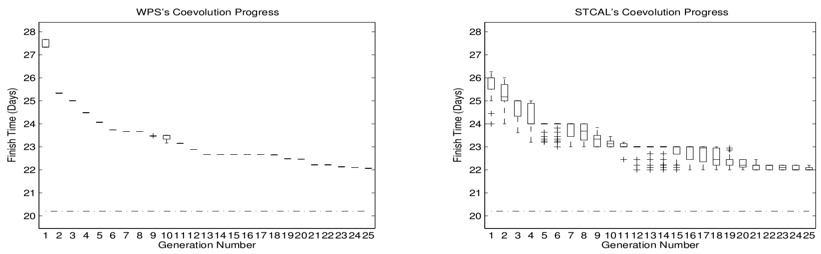
A.4.2 Project D (QuoteToOrder) – Config. BWBS (Cooperative Searching for Better WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

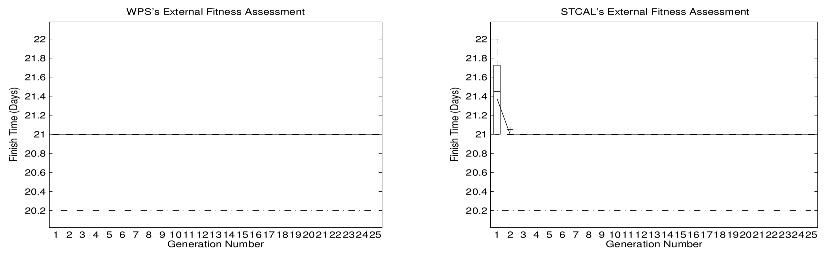


(b) Staff Absence Rate = 0.1

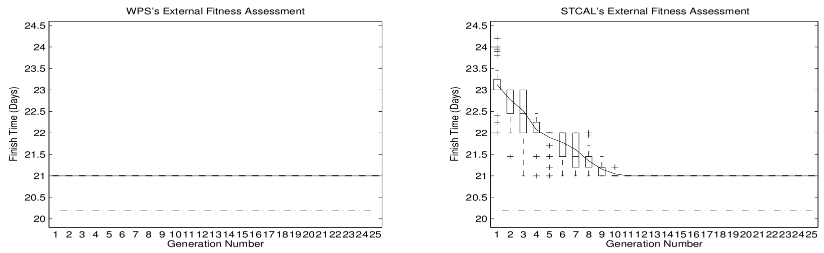


(c) Staff Absence Rate = 0.25

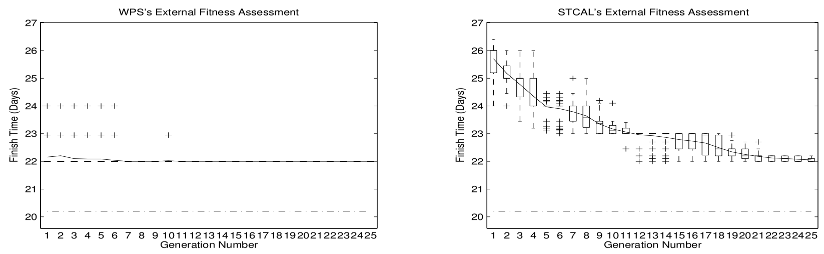
Figure A.27: D – BWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



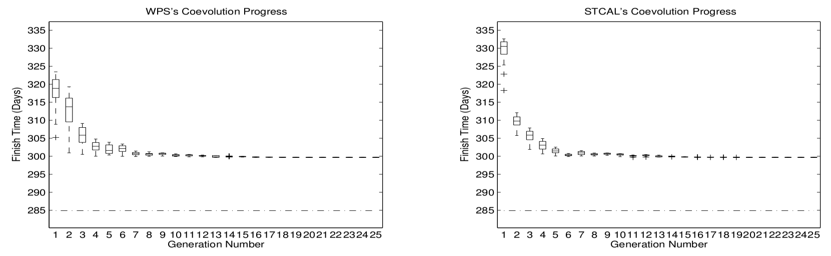
(b) Staff Absence Rate = 0.1



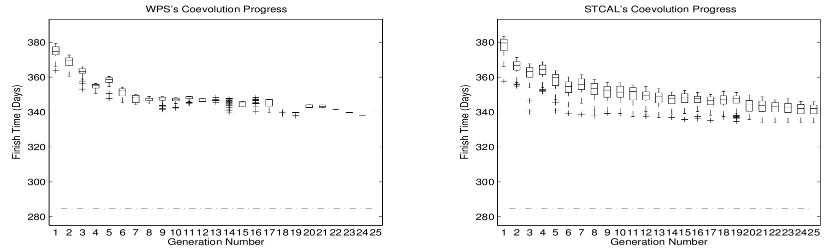
(c) Staff Absence Rate = 0.25

Figure A.28: D – BWBS – External Fitness

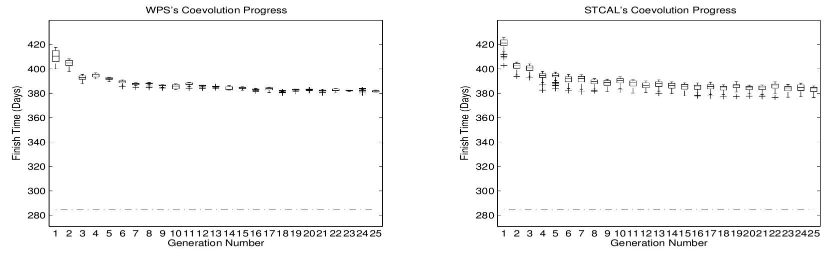
A.4.3 Project E (Database) – Config. BWBS (Cooperative Searching for Better WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

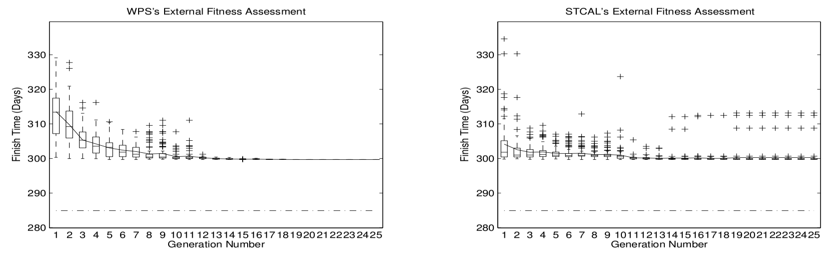


(b) Staff Absence Rate = 0.1

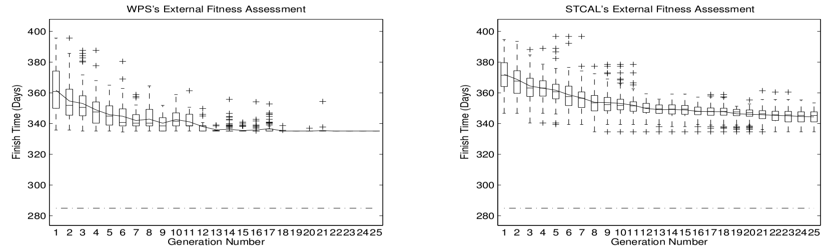


(c) Staff Absence Rate = 0.25

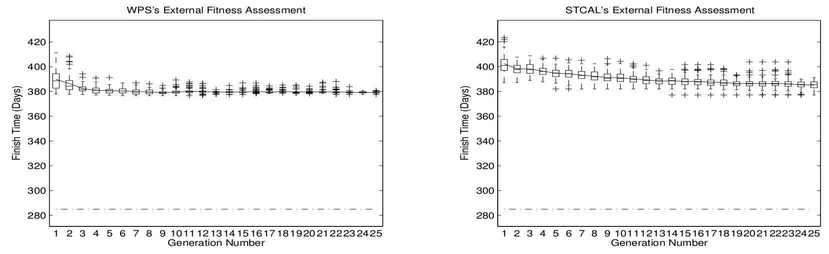
Figure A.29: E – BWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



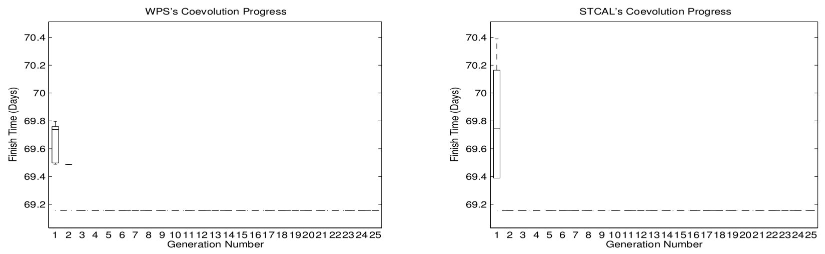
(b) Staff Absence Rate = 0.1



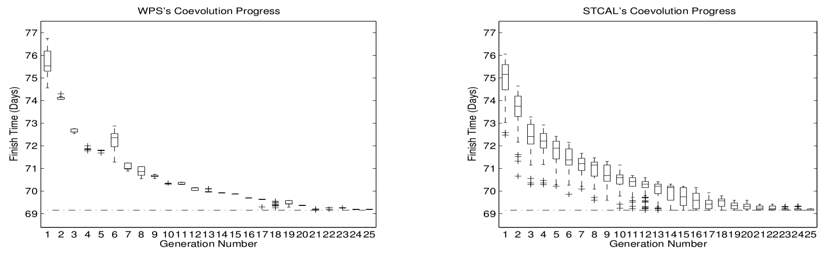
(c) Staff Absence Rate = 0.25

Figure A.30: E – BWBS – External Fitness

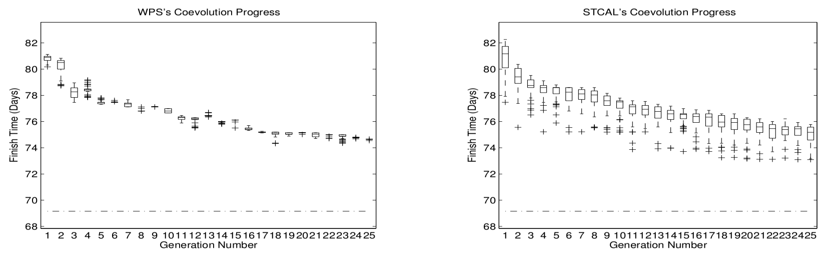
A.4.4 Project F (SmartPrice) – Config. BWBS (Cooperative Searching for Better WPOs and Better STCALs)



(a) Staff Absence Rate = 0.001

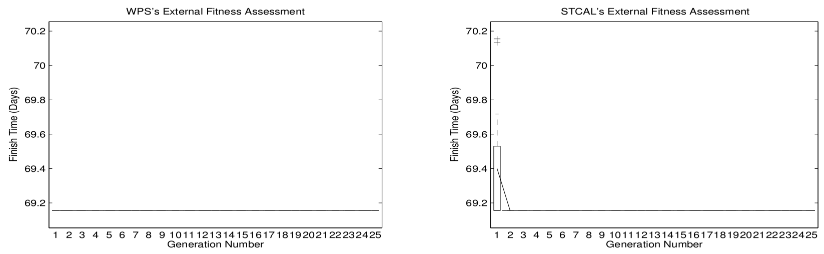


(b) Staff Absence Rate = 0.1

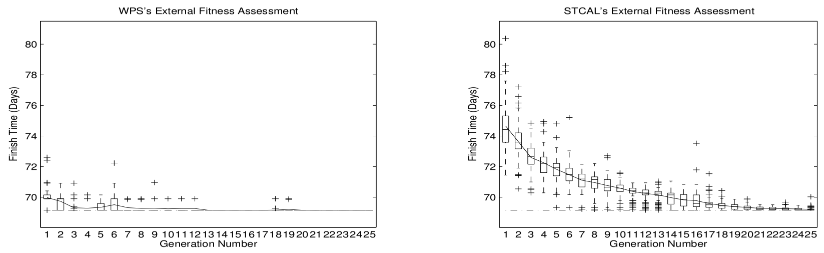


(c) Staff Absence Rate = 0.25

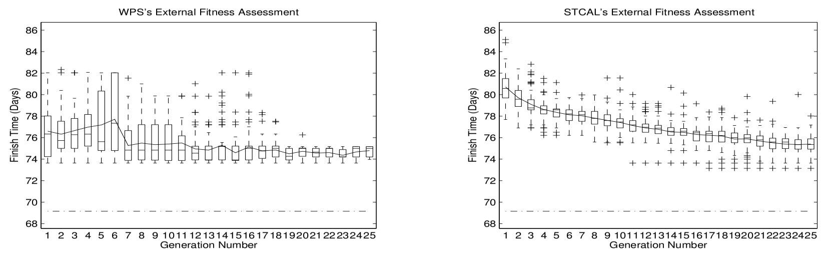
Figure A.31: F – BWBS – Internal Fitness



(a) Staff Absence Rate = 0.001



(b) Staff Absence Rate = 0.1



(c) Staff Absence Rate = 0.25

Figure A.32: F – BWBS – External Fitness

Bibliography

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications, IOS Press*, 7(1):39–59. 36

[Abran et al., 2004] Abran, A., Moore, J., Bourque, P., Dupuis, R., and Tripp, L. (2004). *Guide to the software engineering body of knowledge: 2004 version*. IEEE Computer Society. 10, 26, 27, 29

[Adamopoulos et al., 2004] Adamopoulos, K., Harman, M., and Hierons, R. M. (2004). Mutation Testing Using Genetic Algorithms: A Co-evolution Approach. In *Genetic and Evolutionary Computation Conference (GECCO 2004), LNCS 3103*, pages 1338–1349, Seattle, Washington, USA. Springer. 84

[Afzal et al., 2009] Afzal, W., Torkar, R., and Feldt, R. (2009). A systematic review of search-based testing for non-functional system properties. *Information and Software Technology*, 51(6):957–976. 45

[Aguilar-Ruiz et al., 2002] Aguilar-Ruiz, J. S., Santos, J. C. R., and Ramos, I. (2002). Natural Evolutionary Coding: An Application to Estimating Software Development Projects. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1–8, New York, USA. 101

[Aho et al., 1972] Aho, a. V., Garey, M. R., and Ullman, J. D. (1972). The Transitive Reduction of a Directed Graph. *SIAM Journal on Computing*, 1(2):131. 53

[Aickelin and Dowsland, 2004] Aickelin, U. and Dowsland, K. A. (2004). An indirect Genetic Algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761–778. 49

[Alba and Chicano, 2005] Alba, E. and Chicano, F. (2005). Management of Software Projects with GAs. In *Proceedings of the 6th Metaheuristics International Conference (MIC '05)*, pages 13–18, Vienna, Austria. Elsevier Science Inc. 20, 47, 84, 101

[Alba and Chicano, 2007] Alba, E. and Chicano, F. (2007). Software project management with GAs. *Information Sciences*, 177(11):2380–2401. 20, 45, 47, 84, 101, 102, 104

[Ali et al., 2010] Ali, S., Briand, L. C., Hemmati, H., and Panesar-Walawege, R. K. (2010). A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. *Software Engineering, IEEE Transactions on*, 36(6):742–762. 45

[Alvarez-Valdes et al., 2006] Alvarez-Valdes, R., Crespo, E., Tamarit, J. M., and Villa, F. (2006). A Scatter Search Algorithm for Project Scheduling under Partially Renewable Resources. *Journal of Heuristics*, 12(1-2):95–113. 20, 47, 101

[Antoniol et al., 2004] Antoniol, G., Cimitile, A., Lucca, G. D., and Di, M. (2004). Assessing staffing needs for a software maintenance project through queuing simulation. *Software, IEEE Transactions on*, 30(1):43–58. 46, 53

[Antoniol et al., 2005] Antoniol, G., Di Penta, M., and Harman, M. (2005). Search-Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pages 240–249. 20, 47, 53

[Arcuri and Yao, 2008] Arcuri, A. and Yao, X. (2008). A Novel Co-evolutionary Approach to Automatic Software Bug Fixing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pages 162–168, Hongkong, China. IEEE Computer Society. 84

[Arcuri and Yao, 2010] Arcuri, A. and Yao, X. (2010). Co-evolutionary automatic programming for software development. *Information Sciences (Available online 4 January 2010)*. 84

[Armour, 2002] Armour, P. (2002). Ten unmyths of project estimation. *Commun. ACM*, 45(11):15–18. 61

[Axelrod and Dion, 1988] Axelrod, R. and Dion, D. (1988). The further evolution of cooperation. *Science (New York, N.Y.)*, 242(4884):1385–90. 43

[Axelrod and Hamilton, 1981] Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *Science (New York, N.Y.)*, 211(4489):1390–6. 43

[Bäck and Schwefel, 1993] Bäck, T. and Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation, MIT Press*, 1(1):1–23. 43

[Bagnall et al., 2001] Bagnall, A. J., Rayward-Smith, V. J., and Whittley, I. M. (2001). The next release problem. *Information and Software Technology*, 43(14):883–890. 62, 63, 64, 82

[Baker et al., 2006] Baker, P., Harman, M., Steinhofel, K., and Skaliotis, A. (2006). Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In *22nd IEEE International Conference on Software Maintenance*, pages 176–185, Philadelphia, USA. IEEE Computer Society. 15, 50, 51, 82

[Baniotopoulos, 1991] Baniotopoulos, C. C. (1991). A contribution to the sensitivity analysis of the sea-bed-structure interaction problem for underwater pipelines. *Computers &amp; Structures*, 40(6):1421–1427. 82

[Barki et al., 1993] Barki, H., Rivard, S., and Talbot, J. (1993). Toward an Assessment of Software Development Risk. *Journal of Management Information Systems*, 10:203–225. 34

[Barmby et al., 2002] Barmby, T. A., Ercolani, M. G., and Treble, J. G. (2002). Sickness Absence: An International Comparison. *The Economic Journal*, 112(480):F315—-F331. 105

[Barreto et al., 2008] Barreto, A., de O. Barros, M., and Werner, C. M. L. (2008). Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers & Operations Research*, 35(10):3073–3089. 101

[Beasley and Chu, 1996] Beasley, J. E. and Chu, P. C. (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404. 48

[Beck and Andres, 2004] Beck, K. and Andres, C. (2004). *Extreme programming explained: embrace change*. Addison-Wesley Professional. 29

[Beck et al., 2001] Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for Agile Software Development. 29

[Berman and Larson, 1993] Berman, O. and Larson, R. C. (1993). Optimal workforce configuration incorporating absenteeism and daily workload variability. *Socio-Economic Planning Sciences*, 27(2):91–96. 104

[Black, 2008] Black, D. C. (2008). Working for a healthier tomorrow: review of the health of Britain's working age population. Technical report, Department of Work and Pensions, London. 12, 105, 106

[Blazewicz et al., 1983] Blazewicz, J., Lenstra, J. K., and Kan, A. H. G. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24. 46

[Boehm, 1991] Boehm, B. (1991). Software risk management: principles and practices. *IEEE Software*, 8(1):32–41. 32

[Boehm, 2006] Boehm, B. (2006). A view of 20th and 21st century software engineering. *Proceeding of the 28th international conference on Software engineering - ICSE '06*, page 12. 29

[Boehm et al., 2000] Boehm, B., Abts, C., and Chulani, S. (2000). Software development cost estimation approachesA survey. *Annals of Software Engineering*, 10:177–205. 35, 82

[Boehm et al., 1995] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1):57–94. 36

[Boehm and Ross, 1989] Boehm, B. and Ross, R. (1989). Theory-W software project management principles and examples. *Software Engineering, IEEE Transactions on*, 15(7):902–916. 32, 34

[Boehm, 1984] Boehm, B. W. (1984). Software Engineering Economics. *Software Engineering, IEEE Transactions on*, SE-10(1):4–21. 36, 61

[Boehm, 1988] Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72. 28

[Box et al., 1978] Box, G., Hunter, J., and Hunter, W. (1978). *Statistics for experimenters: an introduction to design, data analysis, and model building.* John Wiley & Sons. New York, USA. 38

[Box and Wilson, 1951] Box, G. and Wilson, K. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (*, 13(1):1–45. 39

[Brooks, Jr., 1975] Brooks, Jr., F. P. (1975). *The Mythical Man Month: Essays on Software Engineering.* Addison-Wesley, Reading, MA, USA. 102

[Bryde, 2003] Bryde, D. (2003). Project management concepts, methods and application. *International Journal of Operations and Production Management*, 23(7/8):775–793. 31

[Bucher, 1990] Bucher, C. (1990). A fast and efficient response surface approach for structural reliability problems. *Structural Safety*, 7(1):57–66. 39

[Campolongo and Braddock, 1999] Campolongo, F. and Braddock, R. (1999). Sensitivity analysis of the IMAGE Greenhouse model. *Environmental Modelling and Software*, 14(4):275–282. 39

[Campolongo et al., 2007] Campolongo, F., Cariboni, J., and Saltelli, a. (2007). An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, 22(10):1509–1518. 39

[Carley et al., 2004] Carley, K. M., Kamneva, N. Y., and Reminga, J. (2004). Response Surface Methodology. Technical report, CASOS Technical Report, Carnegie Mellon University. 39

[Carr et al., 1993] Carr, M., Konda, S., Monarch, I., and Ulrich, F. (1993). Taxonomy-based risk identification. Technical Report June, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. 34

[Cattrysse and Wassenhove, 1992] Cattrysse, D. G. and Wassenhove, L. N. V. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272. 48

[CBI, 2011] CBI (2011). Healthy returns? Absence and workplace health survey 2011. Technical report, Confederation of British Industry, London. 105

[Chan et al., 1996] Chan, W.-T., Chua, D. K. H., and Kannan, G. (1996). Construction Resource Scheduling with Genetic Algorithms. *Journal of Construction Engineering and Management*, 122(2):125–132. 20, 47

[Chang et al., 1998] Chang, C., Chao, C., Nguyen, T., and Christensen, M. (1998). Software project management net: a new methodology on software management. *Proceedings of the 22nd International Computer Software and Applications Conference*, pages 534—-539. 20, 47

[Chang et al., 2001] Chang, C., Christensen, M., and Zhang, T. (2001). Genetic algorithms for project management. *Annals of Software Engineering*, 11(1):107–139. 20, 47, 104

[Chang et al., 2008] Chang, C. K., Jiang, H., Di, Y., Zhu, D., and Ge, Y. (2008). Time-line based model for software project scheduling with genetic algorithms. *Information and Software Technology*, 50(11):1142–1154. 20, 47

[Chao et al., 1993] Chao, C., Komada, J., Liu, Q., Muteja, M., Alsalqan, Y., and Chang, C. (1993). An Application of Genetic Algorithms to Software Project Management. In *Proceedings of the 9th International Advanced Science and Technology*, pages 247–252, Chicago, Illinois, USA. 101

[Christopher Frey and Patil, 2002] Christopher Frey, H. and Patil, S. R. (2002). Identification and Review of Sensitivity Analysis Methods. *Risk analysis : an official publication of the Society for Risk Analysis*, 22(3):553–578. 38, 82

[Chu and Beasley, 1997] Chu, P. C. and Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23. 48

[CIPD, 2011] CIPD (2011). Absence Management: Annual Survey Report 2011. Technical report, CIPD (Chartered Institute of Personnel and Development), London. 106, 133

[CIPD, 2012] CIPD (2012). Absence Management: Annual Survey Report 2012. Technical report, CIPD (Chartered Institute of Personnel and Development), London. 104

[Coello, 2000] Coello, C. A. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Comput. Surv.*, 32(2):109–143. 65

[Cortellessa et al., 2008] Cortellessa, V., Marinelli, F., and Potena, P. (2008). An Optimization Framework for "Build-or-Buy" Decisions in Software Architecture. *Computers & Operations Research*, 35(10):3090–3106. 101

[Danzer and Dolton, 2012] Danzer, A. M. and Dolton, P. J. (2012). Total Reward and pensions in the UK in the public and private sectors. *Labour Economics*, 19(4):584–594. 106

[Darwin, 1859] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London: J. Murray. 40

[de Souza et al., 2010] de Souza, J. T., Maia, C. L., de Freitas, F. G., and Coutinho, D. P. (2010). The Human Competitiveness of Search Based Software Engineering. In *Second International Symposium on Search Based Software Engineering (SSBSE), 2010*, pages 143–152. 101

[Deb and Gupta, 2005] Deb, K. and Gupta, H. (2005). Searching for Robust Pareto-Optimal Solutions in Multi-objective Optimization. In Coello Coello, C., Hernández Aguirre, A., and Zitzler, E., editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 150–164. Springer Berlin / Heidelberg. 82

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions*, 6(2):182–197. 70

[Di Penta et al., 2011] Di Penta, M., Harman, M., and Antoniol, G. (2011). The Use of Search-Based Optimization Techniques to Schedule and Staff Software Projects: An Approach and An Empirical Study. *Softw., Pract. Exper.*, 41(5):495–519. 45, 46, 50, 53, 84, 87, 89, 102

[Di Penta et al., 2007] Di Penta, M., Harman, M., Antoniol, G., and Qureshi, F. (2007). The Effect of Communication Overhead on Software Maintenance Project Staffing: a Search-Based Approach. In *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, pages 315–324. 45, 46, 53, 87, 102

[Fairley, 1994] Fairley, R. (1994). Risk management for software projects. *IEEE software*. 33

[Fernando Netto and Alvim, 2009] Fernando Netto, M. B. and Alvim, A. (2009). A Hybrid Heuristic Approach for Scheduling Bug Fix Tasks to Software. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*, Cumberland Lodge, Windsor, UK. IEEE. 102

[Finkelstein et al., 2009] Finkelstein, A., Harman, M., Mansouri, S., Ren, J., and Zhang, Y. (2009). A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Special Issue on RE'08: Requirements Engineering for a Sustainable World; Guest Editor: T. Tamai*, 14(4):231–245. 44

[Finkelstein et al., 2008] Finkelstein, A., Harman, M., Mansouri, S. A., Ren, J., and Zhang, Y. (2008). Fairness Analysis in Requirements Assignments. In *16th IEEE International Requirements Engineering Conference*, pages 115–124. IEEE. 44

[Fisher et al., 1986] Fisher, M. L., Jaikumar, R., and Wassenhove, L. N. V. (1986). A Multiplier Adjustment Method for the Generalized Assignment Problem. *Management Science*, 32(9):pp. 1095–1103. 48

[Flyvbjerg et al., 2002] Flyvbjerg, B., Holm, M. S., and Buhl, S. (2002). Underestimating Costs in Public Works Projects: Error or Lie? *Journal of the American Planning Association*, 68(3):279–295. 61

[Flyvbjerg et al., 2005] Flyvbjerg, B., Skamris Holm, M. K., and Buhl, S. r. L. (2005). How (In)accurate Are Demand Forecasts in Public Works Projects?: The Case of Transportation. *Journal of the American Planning Association*, 71(2):131–146. 61

[Fogel, 1991] Fogel, D. B. (1991). *System identification through simulated evolution: a machine learning approach to modeling.* Ginn Press, USA. 43

[Fogel et al., 1966] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial intelligence through simulated evolution.* John Wiley and Sons. 43

[Forrester and Wright, 1961] Forrester, J. W. and Wright, J. (1961). *Industrial dynamics.* MIT press Cambridge, MA. 36

[Garrett and Dasgupta, 2008] Garrett, D. and Dasgupta, D. (2008). Multiobjective Landscape Analysis and the Generalized Assignment Problem. In Maniezzo, V., Battiti, R., and Watson, J.-P., editors, *Learning and Intelligent Optimization*, volume 5313 of *Lecture Notes in Computer Science*, pages 110–124. Springer Berlin / Heidelberg. 49

[Ge and Chang, 2006] Ge, Y. and Chang, C. (2006). Capability-based Project Scheduling with Genetic Algorithms. *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*, pages 161–161. 20, 47

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Reading, MA, USA. 36, 42

[Griffiths et al., 1993] Griffiths, W. E., Hill, R. C., and Judge, G. G. (1993). *Learning and practicing econometrics.* John Wiley & Sons Inc. 36

[Gueorguiev, 2008] Gueorguiev, S. (2008). *Using SBSE for Project Management Optimisation: Finding Robust Project Plans.* Msc thesis, King's College London. 54

[Gueorguiev et al., 2009] Gueorguiev, S., Harman, M., and Antoniol, G. (2009). Software Project Planning for Robustness and Completion Time in the Presence of Uncertainty using Multi Objective Search Based Software Engineering. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 1673–1680, Montral, Canada. ACM. 20, 35, 37, 47, 50, 102

[Guignard and Rosenwein, 1989] Guignard, M. and Rosenwein, M. B. (1989). An Improved Dual Based Algorithm for the Generalized Assignment Problem. *Operations Research*, 37(4):pp. 658–663. 48

[Gunawan et al., 2005] Gunawan, R., Cao, Y., Petzold, L., and III, F. J. D. (2005). Sensitivity Analysis of Discrete Stochastic Systems. *Biophysical Journal*, 88(4):2530–2540. 82

[Hajri-Gabouj, 2003] Hajri-Gabouj, S. (2003). A fuzzy genetic multiobjective optimization algorithm for a multilevel generalized assignment problem. *Systems,* *Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 33(2):214–224. 49

[Hamby, 1994] Hamby, D. M. (1994). A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, 32(2):135–154. 37

[Harman, 2007] Harman, M. (2007). The Current State and Future of Search Based Software Engineering. In Briand, L. and Wolf, A., editors, *Future of Software Engineering 2007*, pages 342–357, Los Alamitos, California, USA. 18, 102

[Harman, 2010a] Harman, M. (2010a). The Relationship between Search Based Software Engineering and Predictive Modeling. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering, ACM*, pages 1–13, Timisoara, Romania. ACM. 102

[Harman, 2010b] Harman, M. (2010b). Why the Virtual Nature of Software Makes It Ideal for Search Based Optimization. In Rosenblum, D. and Taentzer, G., editors, *Fundamental Approaches to Software Engineering*, volume 6013 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg. 44

[Harman et al., 2012] Harman, M., Burke, E., Clark, J., and Yao, X. (2012). Dynamic adaptive search based software engineering. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, pages 1–8, New York, NY, USA. ACM. 18

[Harman and Jones, 2001] Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833–839. 18, 44

[Harman et al., 2009a] Harman, M., Krinke, J., Ren, J., and Yoo, S. (2009a). Search based data sensitivity analysis applied to requirement engineering. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1681–1688, New York, NY, USA. ACM. 37, 44, 45

[Harman et al., 2009b] Harman, M., Mansouri, S., and Zhang, Y. (2009b). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical report, Technical Report TR-09-03, Kings College London. 29, 44, 45

[Harman and McMinn, 2010] Harman, M. and McMinn, P. (2010). A Theoretical and Empirical Study of Search-Based Testing: Local, Global, and Hybrid Search. *Software Engineering, IEEE Transactions on*, 36(2):226–247. 45

[Harper et al., 2005] Harper, P. R., de Senna, V., Vieira, I. T., and Shahani, A. K. (2005). A genetic algorithm for the project assignment problem. *Computers & Operations Research*, 32(5):1255–1265. 49

[Hart et al., 2005] Hart, E., Ross, P., and Corne, D. (2005). Evolutionary Scheduling: A Review. *Genetic Programming and Evolvable Machines*, 6(2):191–220. 46

[Hausknecht et al., 2008] Hausknecht, J., Hiller, N., and Vance, R. (2008). Work-Unit Absenteeism: Effects of Satisfaction, Commitment, Labor Market Conditions, and Time. *The Academy of Management Journal ARCHIVE*, 51(6):1223–1245. 105

[Helmer et al., 1966] Helmer, O., Brown, B., and Gordon, T. (1966). *Social technology*. Basic Books New York. 36

[Helton et al., 2006] Helton, J., Johnson, J., Sallaberry, C., and Storlie, C. (2006). Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety*, 91(10-11):1175–1209. 37, 38, 40

[Herroelen, 2005] Herroelen, W. (2005). Project SchedulingTheory and Practice. *Production and Operations Management*, 14(4):413–432. 30, 45

[Higgins et al., 2012] Higgins, A., OHalloran, P., and Porter, S. (2012). Management of Long Term Sickness Absence: A Systematic Realist Review. *Journal of Occupational Rehabilitation*, 22(3):322–332. 105

[Hill and Hunter, 1966] Hill, W. and Hunter, W. (1966). A review of response surface methodology: a literature survey. *Technometrics*, 8(4):571–590. 39

[Hillis, 1990] Hillis, W. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42:228–234. 20, 43

[Hindi et al., 2002] Hindi, K. S., Yang, H., and Fleszar, K. (2002). An evolutionary algorithm for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, 6(5):512–518. 20, 47

[Holland, 1992] Holland, J. H. (1992). *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA. 18, 42

[Holmes, 2008] Holmes, E. (2008). The feasibility of comparing sickness absence surveys and the Labour Force Survey. Technical report, HSE Contract Research Report RR673. 106

[Hughes et al., 2004] Hughes, B., Ireland, R., West, B., Smith, N., and Shepherd, D. (2004). *Project management for IT-related projects: Textbook for the Iseb Foundation Certificate in Is Project Management.* Swindon: The British Computer Society. 26

[Hur et al., 2004] Hur, D., Mabert, V. A., and Bretthauer, K. M. (2004). Real-Time Work Schedule Adjustment Decisions: An Investigation and Evaluation. *Production and Operations Management*, 13(4):322–339. 104

[Husbands and Mill, 1991] Husbands, P. and Mill, F. (1991). Simulated co-evolution as the mechanism for emergent planning and scheduling. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264—-270. 43

[IEEE610.12-1990, 1990] IEEE610.12-1990 (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE StMcMinn2005andard*, 610.12-90. 26

[Isukapalli et al., 2000] Isukapalli, S. S., Roy, a., and Georgopoulos, P. G. (2000). Efficient sensitivity/uncertainty analysis using the combined stochastic response surface method and automated differentiation: application to environmental and biological systems. *Risk analysis : an official publication of the Society for Risk Analysis*, 20(5):591–602. 39

[Jø rgensen, 2004a] Jø rgensen, M. (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60. 37

[Jø rgensen, 2004b] Jø rgensen, M. (2004b). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, 46(1):3–16. 31, 36

[Jø rgensen et al., 2009] Jø rgensen, M., Boehm, B., and Rifkin, S. (2009). Software Development Effort Estimation: Formal Models or Expert Judgment? *IEEE Software*, 26(1):14–19. 35, 36

[Jø rgensen and Shepperd, 2007] Jø rgensen, M. and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *Software Engineering, IEEE Transactions on*, 33(1):33–53. 30

[Jones, 1997] Jones, C. (1997). *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill. 36

[Jose, 2008] Jose, F. D. (2008). *Sensitivity Analysis for Search-Based Software Project Management*. PhD thesis, King's College London, UK. 54

[Kapur et al., 2008] Kapur, P., Ngo-the, A., Ruhe, G., and Smith, A. (2008). Optimized staffing for product releases and its application at Chartwell Technology. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(5):365–386. 101

[Keil et al., 1998] Keil, M., Cule, P., Lyytinen, K., and Schmidt, R. (1998). A framework for identifying software project risks. *Communications of the ACM*, 41(11):76–83. 34

[Keil et al., 2003] Keil, M., Rai, A., Cheney Mann, J., and Zhang, G. (2003). Why software projects escalate: The importance of project management constructs. *Engineering Management, IEEE Transactions on*, 50(3):251–261. 31

[Kelley Jr and Walker, 1959] Kelley Jr, J. E. and Walker, M. R. (1959). Critical-path planning and scheduling. In *IRE-AIEE-ACM '59 (Eastern): Papers pre-*

sented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference, IRE-AIEE-ACM '59 (Eastern), pages 160–173, New York, NY, USA. ACM. 31, 46

[Khuri and Mukhopadhyay, 2010] Khuri, A. I. and Mukhopadhyay, S. (2010). Response surface methodology. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):128—-149. 39

[Kicinger et al., 2005] Kicinger, R., Arciszewski, T., and De Jong, K. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23-24):1943–1978. 42

[Kolisch and Hartmann, 2006] Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37. 46

[Koza and Poli, 2005] Koza, J. and Poli, R. (2005). Genetic programming. *Search Methodologies, Springer*, pages 127—-164. 42

[Koza, 1992] Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection.* Cambridge, Mass.: The MIT press. 42

[Kremmel et al., 2011] Kremmel, T., Kubalík, J., and Biffl, S. (2011). Software Project Portfolio Optimization with Advanced Multiobjective Evolutionary Algorithms. *Applied Soft Computing*, 11(1):1416–1426. 101

[Langdon and Poli, 2002] Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming.* Springer. 42

[Levine, 1996] Levine, D. (1996). Application of a hybrid genetic algorithm to airline crew scheduling. *Computers & Operations Research*, 23(6):547–558. 49

[Levine and Renelt, 1992] Levine, R. and Renelt, D. (1992). A sensitivity analysis of cross-country growth regressions. *The American Economic Review*, 82(4):942–963. 38, 82

[Majumdar and Bhunia, 2007] Majumdar, J. and Bhunia, A. K. (2007). Elitist genetic algorithm for assignment problem with imprecise goal. *European Journal of Operational Research*, 177(2):684–692. 49

[Malcolm et al., 1959] Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. (1959). Application of a Technique for Research and Development Program Evaluation. *Operations Research*, 7(5):pp. 646–669. 31, 46

[McMinn, 2004] McMinn, P. (2004). Search-based software test data generation: a survey. *Software Testing, Verification and Reliability*, 14(2):105–156. 45

[McMinn, 2005] McMinn, P. (2005). *Evolutionary search for test data in the presence of state behaviour*. PhD thesis, University of Sheffield. 45

[Miller and Spooner, 1976] Miller, W. and Spooner, D. (1976). Automatic Generation of Floating-Point Test Data. *Software Engineering, IEEE Transactions on*, SE-2(3):223–226. 44

[Miyazaki et al., 1994] Miyazaki, Y., Terakado, M., Ozaki, K., and Nozaki, H. (1994). Robust regression for developing software estimation models. *Journal of Systems and Software*, 27:3–16. 36

[Mokotoff, 2001] Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research*, 18(2):193–242. 47

[Morris, 1991] Morris, M. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174. 38

[Moynihan, 1997] Moynihan, T. (1997). How experienced project managers assess risk. *Software, IEEE*, 14(3):35–41. 33

[Newman et al., 1999] Newman, J., Taylor, A., Barnwell, R., and Newman, P. (1999). Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *Journal of Aircraft*, 36(1):87–96. 37

[Odzaly and Des Greer, 2009] Odzaly, E. E. and Des Greer, P. S. (2009). Software risk management barriers: An empirical study. In *Proceedings of the ACM-IEEE*

International Symposium on Empirical Software Engineering and Measurement, pages 418–421. Ieee. 35

[Paredis, 1994] Paredis, J. (1994). Co-evolutionary constraint satisfaction. *Parallel Problem Solving from Nature (PPSN III)*, 37(2):121–133. 43

[PMI, 2004] PMI (2004). *A Guide to the Project Management Body of Knowledge: PMBOK{\textregistered} Guide.* Project Management Institute, Pennsylvania, USA. 31

[Potter, 1997] Potter, M. (1997). *The design and analysis of a computational model of cooperative coevolution.* PhD thesis, George Mason University. 44

[Potter and Jong, 1994] Potter, M. A. and Jong, K. A. D. (1994). A Cooperative Coevolutionary Approach to Function Optimization. In *The Third Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer-Verlag. 20, 44, 88

[Putnam and Myers, 1992] Putnam, L. H. and Myers, W. (1992). *Measures for excellence: reliable software on time, within budget.* Yourdon Press. 36

[Racu et al., 2005] Racu, R., Jersak, M., and Ernst, R. (2005). Applying sensitivity analysis in real-time distributed systems. *11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 160–169. 37

[Rechenberg, 1964] Rechenberg, I. (1964). Cybernetic solution path of an experimental problem. *Library Translation 1122. Farnborough, UK: Royal Aircraft Establishment.* 42

[Ren et al., 2011] Ren, J., Harman, M., and Di Penta, M. (2011). Cooperative Coevolutionary Optimization of Software Project Staff Assignments and Job Scheduling. In *Proceedings of the Third International Symposium of Search Based Software Engineering, SSBSE 2011*, volume 6956 of *Lecture Notes in Computer Science*, pages 127–141. Springer Berlin / Heidelberg. 109

[Ross and Soland, 1975] Ross, G. T. and Soland, R. M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8(1):91–103. 48

[Royce, 1970] Royce, W. (1970). Managing the development of large software systems. *Proceedings of Western Electronic Show and Convention (IEEE WESCON)*, 26(August):1–9. 27

[Saltelli, 2004] Saltelli, A. (2004). *Sensitivity analysis in practice: a guide to assessing scientific models*. John Wiley & Sons. 40

[Saltelli, 2005] Saltelli, A. (2005). Global sensitivity analysis: an introduction. *. 4th International Conference on Sensitivity Analysis of*, pages 27–43. 40

[Saltelli and Annoni, 2010] Saltelli, A. and Annoni, P. (2010). How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, pages 1–10. 38, 40

[Saltelli et al., 2004] Saltelli, A., Chan, K., Scott, E. M., and Others (2004). *Sensitivity analysis*. John Wiley & Sons. 40

[Saltelli et al., 2008] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons. 38, 40

[Saltelli et al., 2000] Saltelli, A., Tarantola, S., and Campolongo, F. (2000). Sensitivity Anaysis as an Ingredient of Modeling. *Statistical Science*, 15(4):377–395. 38, 82

[Sandu et al., 2003] Sandu, a., Daescu, D., and Carmichael, G. (2003). Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I–theory and software tools. *Atmospheric Environment*, 37(36):5083–5096. 37

[Schwaber and Beedle, 2001] Schwaber, K. and Beedle, M. (2001). *Agile software development with Scrum*. Prentice Hall, Upper Saddle River, NJ, USA. 29, 31

[Schwefel, 1965] Schwefel, H. P. (1965). Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. *Master's thesis, Technical University of Berlin*. 42

[Seccombe, 1995] Seccombe, I. J. (1995). *Measuring and monitoring absence from work*. Institute for Employment Studies, IES Report 288. 133

[Shepperd, 2007] Shepperd, M. (2007). Software project economics: a roadmap. In *Future of Software Engineering, 2007. FOSE '07*, pages 304–315. 37

[Sommerville, 1992] Sommerville, I. (1992). *Software Engineering*. Addison-Wesley, UK. 28

[Tausworthe, 1980] Tausworthe, R. (1980). The Work Breakdown Structure in Software Project Management. *Journal of Systems and Software*, 1:181—-186. 31, 36

[Taylor et al., 2010] Taylor, P., Cunningham, I., Newsome, K., and Scholarios, D. (2010). Too scared to go sick reformulating the research agenda on sickness absence. *Industrial Relations Journal*, 41(4):270–288. 104

[Thogmartin, 2010] Thogmartin, W. E. (2010). Sensitivity analysis of North American bird population estimates. *Ecological Modelling*, 221(2):173–177. 38

[van den Akker et al., 2008] van den Akker, M., Brinkkemper, S., Diepen, G., and Versendaal, J. (2008). Software product release planning through optimization and what-if analysis. *Information and Software Technology*, 50(1-2):101–111. 82

[Van Veldhuizen, 1999] Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Air Force Institute of Technology, Wright Patterson, OH, USA. 67

[Wallace and Keil, 2004] Wallace, L. and Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4):68–73. 34

[Weise, 2009] Weise, T. (2009). Global Optimization Algorithms Theory and Application. *URL: http://www.it-weise.de, Abrufdatum*, E-book. 43

[Wiegand, 2003] Wiegand, R. (2003). *An analysis of cooperative coevolutionary algorithms*. PhD thesis, George Mason University. 44

[Wilson, 1997] Wilson, J. M. (1997). A Genetic Algorithm for the Generalised Assignment Problem. *The Journal of the Operational Research Society*, 48(8):pp. 804–809. 48

[Xiao and Afzal, 2010] Xiao, J. and Afzal, W. (2010). Search-based Resource Scheduling for Bug Fixing Tasks. In *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*, pages 133–142, Benevento, Italy. IEEE. 102

[Xiao et al., 2010] Xiao, J., Osterwell, L. J., Wang, Q., and Li, M. (2010). Dynamic Resource Scheduling in Disruption-Prone Software Development Environments. In *Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering (FASE '10) Held as Part of the Joint European Conferences on Theory and Practice of Software (ETAPS '10)*, volume 6013, pages 107–122, Paphos, Cyprus. Springer. 101

[Yang et al., 2008a] Yang, D., Wang, Q., Li, M., Yang, Y., Ye, K., and Du, J. (2008a). A survey on software cost estimation in the chinese software industry. *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 253—-262. 37

[Yang et al., 2008b] Yang, Z., Tang, K., and Yao, X. (2008b). Large Scale Evolutionary Optimization Using Cooperative Coevolution. *Information Sciences*, 178(15):2985–2999. 44, 88

[Yao, 1996] Yao, X. (1996). An Overview of Evolutionary Computation. *Chinese Journal Of Advanced Software Research*, 3(1):12—-29. 42

[Zhang et al., 2007] Zhang, Y., Harman, M., and Mansouri, S. A. (2007). The multi-objective next release problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1129–1137, New York, NY, USA. ACM. 44, 65, 70, 82

[Zhu et al., 2005] Zhu, G., Bard, J. F., and Yu, G. (2005). Disruption Management for Resource-Constrained Project Scheduling. *The Journal of the Operational Research Society*, 56(4):365–381. 104