

AMERICAN UNIVERSITY OF BEIRUT

SCHEDULING PRODUCT DEVELOPMENT PROJECTS
USING GENETIC ALGORITHMS

by
OMAR MAJED MOSTAFA

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Engineering Management Program
of the Faculty of Engineering
at the American University of Beirut

Beirut, Lebanon
June 2014

AMERICAN UNIVERSITY OF BEIRUT

SCHEDULING PRODUCT DEVELOPMENT PROJECTS
USING GENETIC ALGORITHMS

by
OMAR MAJED MOSTAFA

Approved by:

Dr. Ali Yassine, Professor
Engineering Management Department



Advisor

Dr. Bacel Maddah, Associate Professor
Engineering Management Department



Member of Committee

Dr. Walid Nasr, Assistant Professor
Engineering Management Department



Member of Committee

Date of thesis/dissertation defense: [June 30, 2014]

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Mostafa Omar Majid

Master's Thesis
Dissertation

Master's Project

Doctoral

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.



Signature

18.09.2014

Date

ACKNOWLEDGMENTS

Special thanks are due for Prof Ali Yassine for his continual efforts and guidance. Thanks are also extended to Mr. Saeed Khodor for his assistance in the programming phase of this thesis.

Most importantly, none of this would have been possible without the love, support and patience of my family. I would also like to thank my one and only love for her bright presence in each and every step of mine. She was always there cheering me up and standing by my side through the good times and the bad.

Last but not the least, the one above all of us, the omnipresent God, for answering my prayers for giving me the strength to plod on despite my constitution wanting to give up and throw in the towel, thank you so much Dear Lord.

AN ABSTRACT OF THE THESIS OF

Omar Majid Mostafa for Master of Engineering
Major: Engineering Management

Title: Scheduling Product Development Projects Using Genetic Algorithms

Resources for development projects are often scarce in the real world. Generally, many projects are to be completed that rely on a common pool of resources. Besides resource constraints, there exist precedence constraints among tasks within each project. Beyond the feed-forward dependencies between tasks, it is common in development projects the existence of feedback dependencies that constitute a new level of scheduling complexity for these projects.

In this thesis, two genetic algorithm (GA) approaches (Variable Sample GA and Variable Length GA) are proposed for scheduling project activities in order to minimize the overall duration or makespan of development projects in a resource constrained, multi project environment without violating inter-project resource constraints or intra-project precedence constraints. Additionally, the proposed GAs allow for the existence of stochastic feedback between activities or rework of activities. These proposed GAs, with several variants of GA parameters, are tested on sample scheduling problems with and without stochastic feedback. The algorithms provide quick convergence to a global optimal solution and detect the most likely schedules, makespan range, as well as the minimum makespan and its schedule.

Two objectives functions were used in this study: project lateness and portfolio lateness. Using several measures for project and portfolio scheduling problems (with feedback) characteristics, we conducted a comparative analysis between 31 published priority rules and the proposed GAs. Test problems were generated to the specifications of project, activity, and resource-related characteristics including network complexity, resource distribution and contention and rework probability. The GAs performed better than the PRs as the level of iteration increases as well as the three other factors increased, including project complexity, resource utilization and resource loading. I close the thesis by providing managers with a decision matrix showing when (i.e. under what project/portfolio conditions) it is best to use the published PRs and when it is best to use the GAs.

Keywords: Design Structure Matrix (DSM), Makespan, Resource Constrained Multi Project Scheduling Problem (RCMPSP), Genetic Algorithm (GA)

CONTENT

ACKNOWLEDGEMENTS.....	v
ABSTRACT.....	vi
LIST OF ILLUSTRATIONS.....	ix
LIST OF TABLES.....	xi

Chapter

1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	6
2.1. Design Structure Matrix (DSM).....	7
2.2. Genetic Algorithms	12
2.3. Resource Constrained Project Scheduling Problem (RCPSP) and Multi-16 Project Scheduling Problem (RCMPSP).....	16
2.4. Network Generators	20
2.5. Project Characteristics	23
3. PROPOSED MODELS.....	26
3.1. Problem Description.....	26
3.2. Solution Methodology.....	29
3.2.1. Variable Sample (Sampling) GA.....	35
3.2.2. Variable Length GA	41
3.3. Operators for GA.....	50
3.4. Guideline to Scheduling Decisions	50

4. EXCEL IMPLEMENTATION.....	55
5. COMPUTATIONAL RESULTS AND INSIGHTS.....	56
5.1. Data Used in Analyses.....	56
5.2. Model Calibration by Performing Sensitivity Analyses on GA Factors and Network Characteristics.....	57
5.3. Model Validation.....	65
5.3.1. Hartmann’s Benchmark	65
5.3.2. Browning &Eppinger’s Case Study.....	65
5.4. Setup and Computational Results.....	70
5.4.1. Results of O1	71
5.4.2. Results of O2.....	73
5.4.3. Summary of Results	75
6. SUMMARY AND CONCLUSION.....	79
Appendix	
1. APPENDIX A.....	86
2. APPENDIX B.....	88
3. APPENDIX C.....	89
4. APPENDIX D.....	90
5. APPENDIX E.....	97
5. APPENDIX F.....	101
REFERENCES.....	81

ILLUSTRATIONS

Figure	Page
1. Tasks relationships representation.....	8
2. DSM model.....	9
3. DSM project example.....	11
4. GA flowchart.....	14
5. Multi-Project Scheduling Problem DSM.....	29
6. Chromosomes of different length.....	31
7. Example problem composed of three projects.....	34
8. Permutation process.....	38
9. Tasks predecessors.....	42
10. Example for steps 4 to 8 of Variable Length GA.....	45
11. Variable Length GA Flowchart.....	46
12. Instantaneous Feasibility Check.....	48
13a. Duration/Makespan Distribution for B&E example.....	51
13b. New duration distribution.....	52
14. Histograms of Duration of GA vs Simulation.....	53
15. Range of number of tasks based on Sample Size	58
16. Mutation and Crossover factors sensitivity analyses.....	60
17. Two-way interaction plots for number of generations (G), population size (P) and complexity (C).....	61
18. Two-way interaction plots for number of generations (G), population size (P) and resources constraints (NARLF, MAUF).....	62
19. Two-way interaction plots for number of generations (G), population size (P) and iteration level.....	63

20.	DSM for Browning and Eppinger (2002) problem.....	66
21.	Histograms showing distribution of makespan.....	68
22.	Makespan data distributions for operators test.....	69
23.	One-way analysis of means (ANOM) for O1 (no iteration).....	71
24.	One-way analysis of means (ANOM) for O1 (with iteration).....	72
25.	One-way analysis of means (ANOM) for O2 (no iteration).....	73
26.	One-way analysis of means (ANOM) for O2 (iteration only).....	74

TABLES

Table		Page
1.	Differences between S-GA/V-GA and regular GA.....	49
2.	Statistical results with ranges of significant and related variables for the individual aggregate models.....	61
3.	Impact of Operators (Percentage Change).....	62
4.	Cases based on Browning and Eppinger (2002) problem.....	65
5.	Examples Comparison	67
6.	Results for O1.....	75
7.	Results for O2.....	76

CHAPTER I

INTRODUCTION

Many organizations rely on their research and development projects for their survival. The delivery of new products and services is critical for the success of the organization. This is true for all types of organizations whether project-based, such as large aerospace companies, or companies that compete in the market based on a flow of new products and services, such as in the automotive, electronics, and software industries. Due to the increase in market competition as well as customers' needs and impatience, it has become very important to improve the efficiency with which projects are completed and new products are brought to market. Moreover, many organizations are faced with the challenge of managing the simultaneous execution and management of a portfolio of projects under tight time and resource constraints. In such an environment, project management and scheduling skills become very critical to the organization.

A survey by Anderson and Tucker (1994) reveals that about one third of development projects miss cost and schedule targets. Several development problems may be due to design defects and can be traced back to the design process (Bramble & Cipollini 1995). A review by Morris and Hough (1987) of some 3500 projects revealed that 'overruns are the norm, being typically between 40 and 200 %'. Another survey by Roberts (1992) of corporate R&D projects found that less than 50 % met their time to market and budget objectives. A World Bank survey of its projects (World Bank, 1992) found that only 70 % of recent projects had been rated 'satisfactory', with only one-third substantially achieving institution development goals and with delays in completion averaging 50 %.

Problems of cost and schedule overrun on projects have persisted for decades, in spite of numerous advances in the field of project management. Starting with Project Evaluation and Review Technique (PERT) and then the critical path method (CPM) in the 1950s, network techniques have continued to develop, and integrate with resource loading and probabilistic assessments; alternative approaches to software development, such as the waterfall and spiral methods have been advanced; teaming, concurrent engineering, and the recognition and emphasis on ‘soft’ and people factors have emerged as methods of enhancing project performance.

There is abundant literature (which will be described later) on resource constrained project scheduling problem (RCPSP), which present an extension to CPM and PERT by the inclusion of the availability of resources during scheduling. However, a company may often have several concurrent development projects. While projects may be unrelated, they are dependent on a common pool of resources to execute. In other words, no precedence relationships exist between projects, but the same set of people is assigned tasks in multiple projects. The problem described above is known as the resource constrained multi project scheduling problem (RCMPSP) (Kolisch and Padman, 1997). RCMPSPs can be very complex as the number of tasks increases. Furthermore, the addition of projects may also complicate the scheduling as it may significantly increase the feasible solution space. For instance, if m projects, totaling n tasks are to be performed, then there can be $\frac{n!}{m!}$ potential feasible schedules when considering intra-project precedence constraints. It is obvious that even a small increase in the total number of tasks may lead to a remarkably large solution space.

Being strongly NP-hard, there are no known algorithms for finding optimal solutions in polynomial time (Lenstra&Kan,1978). Hence, most research has sought efficient heuristics and meta-heuristics. Priority rule (PR) heuristics are the most common heuristics considered for very large problems, and are known for their speed, simplicity and ability to construct initial solutions (Browning &Yassine, 2010).

To complicate things further, in product development projects, scheduling activity iteration or repeated activities has been always a challenge. Even though particular iterations are not necessarily known with certainty prior to project execution, a skilled manager can identify many potentially iterative development activities and plan accordingly. Understanding the web of information flow in a project can help identify potential iterations. Unfortunately, many PD managers fail to plan for iterations in advance.

Several meta-heuristics have been used for project scheduling, and even though developed before genetic algorithms, meta-heuristics such as simulated annealing, tabu search or ant colony optimization has gradually been superseded by GA. This thesis introduces two genetic algorithm based approaches to solve a resource constrained multi-project scheduling problem (RCMPSP) with additional consideration given to task iteration. The procedure is based on modified genetic encoding of standard GA and GA operators to suit this problem. The classic GA is described in section 2 while the proposed model and the GA modifications are discussed in section 3.

The main objectives of the thesis are: 1) to test the performance of PRs regarding resource constrained project scheduling problems with feedback, 2) to find an optimal or near optimal duration (makespan) distribution for a project, 3) to find a

most probable schedule for iterative projects as well as keep dynamic tracking for the scheduling process of such uncertain problems.

The model was tested on well-known project scheduling problems. Examples developed by Kolisch et al. (1996) and used by Hartmann (1998) were used as benchmark. Those examples are resource constrained projects without feedback. Another example considering iteration or potential rework is the one developed by Browning and Eppinger (2002) and is used also as a benchmark to study the validity of our model. The model results are consistent with those of the Hartmann (1998) and Browning and Eppinger (2002).

We address the RCMPSP with feedback (RCMPSPWF) with two lateness objectives. Then, using a full factorial experiment with randomly generated problem instances, we demonstrate the superiority of the proposed GA-based approach by comparing our results to published priority rules (PRs).

We found significant differences in the performance of the GAs relative to PRs, in cases of high levels of iteration, complexity and resource constraints; moreover, the GAs showed better convergence towards optimal solutions (shortest makespans) especially in high complexity projects, as well as projects with feedback. Finally, we organize these results for managers, distinguishing between the project and portfolio management perspectives.

The rest of the thesis is organized as follows. In chapter 2 we describe relevant literature in traditional project management, DSM, and genetic algorithms. Chapter 3 describes the two proposed GA approaches (Sampling and variable length GA) discussing all parameters and methodologies and gives managers a guideline to scheduling decisions. Chapter 4 discusses the Excel implementation of the problem,

while Chapter 5 introduces the data used in the analyses, as well as model calibration by performing Sensitivity Analyses on GA parameters and network characteristics. It also discusses model validation according to Hartmann's benchmark and Browning & Eppinger's case study. Furthermore, Chapter 5 describes the setup and computational results of the problem measuring the accuracy of priority rules and determining where the GA outperforms PRs. Chapter 6 summarizes and draws a conclusion on the thesis.

CHAPTER II

LITERATURE REVIEW

Early effort in project scheduling focuses on minimizing the overall project duration. Well-known techniques include Critical Path Method (CPM) and the Project Evaluation and Review Technique (PERT) (Spinner, 1989). These methods assume scheduling under unlimited availability of resources for each activity (Agarwal and Erenguc, 2006). However, in practice, resources are available only in limited quantities and the resource demands of concurrent activities may not be satisfied. Scheduling large-scale projects with resource constraints is extremely hard due to its sheer complexity.

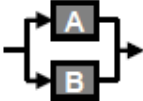


Iteration or activity repetition is another complicating factor. In product development some of the activities that have already completed may have to be reworked due to discovering an error downstream or due to assumptions that were made earlier. In any case, traditional project scheduling techniques (e.g., CPM and PERT) will not be useful in iterative project situations. Recently, a tool called the design structure matrix (DSM) has been proposed to help in scheduling iterative projects.

The literature review covers in detail both RCPSP and the DSM method. However, within this vast literature, we focus on meta-heuristics approaches (Genetic Algorithms, in particular) in the RCPSP and simulation approaches in the DSM literature.

2.1. Design Structure Matrix (DSM)

A design structure matrix (DSM) is an efficient and commonly used method of showing the relationships between tasks within a project (Yassine & Braha, 2003). Given a set of n tasks in a project, the corresponding DSM is an $n \times n$ matrix where the project tasks are the row and column headings listed in the same order. The row of the activity represents its inputs and its column shows its outputs. The precedence relationship between tasks corresponds to the off-diagonal elements of the matrix. In order to complete the task, a certain resource needs to be allocated to the task. The resource could be a machine, raw material, person, and the amount of the resource needed to complete the task could be all, or a fraction of the resource. All cells along the diagonal of the matrix represent a deterministic (i.e. nominal) completion time for the corresponding task.

This method represents information flows rather than work flows, showing the relationships and sequence between activities. There are three basic building blocks for describing the relationship amongst system elements: parallel (or concurrent), sequential (or dependent) and coupled (or interdependent) (Yassine, 2004). Figure 1 shows those three blocks (Yassine, 2004).

Three Configurations that Characterize a System			
Relationship	Parallel	Sequential	Coupled
Graph Representation			

Three Configurations that Characterize a System																														
Relationship	Parallel	Sequential	Coupled																											
DSM Representation	<table border="1" data-bbox="552 568 740 728"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>■</td></tr> <tr><td>B</td><td>■</td><td>■</td></tr> </table>		A	B	A	■	■	B	■	■	<table border="1" data-bbox="871 568 1059 728"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>■</td></tr> <tr><td>B</td><td>X</td><td>■</td></tr> </table>		A	B	A	■	■	B	X	■	<table border="1" data-bbox="1190 568 1378 728"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>X</td></tr> <tr><td>B</td><td>X</td><td>■</td></tr> </table>		A	B	A	■	X	B	X	■
	A	B																												
A	■	■																												
B	■	■																												
	A	B																												
A	■	■																												
B	X	■																												
	A	B																												
A	■	X																												
B	X	■																												

Figure 1: Tasks relationships representation

If there exists an edge from node i to node j , then the value of element ij (column i , row j) is unity (or flagged with a mark such as “X”). Otherwise, the value of the element is zero (or left empty) (Yassine, 2004). The marks below the diagonal represent forward information from task i to task j , while those above the diagonal represent backward or feedback information from task j to task i . Those are respectively called downstream and upstream. Parallel tasks are tasks that have to interference between one another, this means that the two may be in action at the same time. They are said to be independent. An example of a DSM model is shown in Figure 2 and may be linked with the example stated earlier.

Task	1	2	3	4	5
1	D1		X		
2	X	D2			
3	X		D3		
4		X		D4	
5		X		X	D5

Figure 2: DSM model

Feed forward dependencies are easy to deal with, in the contrary to feedback dependencies which prove to be more challenging. The latter dependencies exist actually due to the uncertainty in performing some activities due to lack of information. When you information appears along through the process, this uncertainty declines and may either validate how activities were done or may reveal some mistakes during performing any of the tasks done. Reconsideration of those errors or mistakes is important and some activities should be redone. This referral to previous done activities is notated as rework. Iteration takes place to converge on a solution, or rework may take place due to errors or undesired/expected test outcomes.

Eppinger et al. (1994) used DSMs in order to improve task order, Yassine et al. (2001) used DSMs to simulate rework in the automotive industry, and analyze its sensitivity to rework probabilities and Cho and Eppinger (2005) even though believed simplifying assumptions and poor scaling limit the real simulation, illustrated in sequence, in parallel, or overlapping iterated tasks based on their dependencies.

The model used in this thesis is according to (Browning & Eppinger 2002) where the project is resembled as a network of activities that exchange deliverables. If an activity does work and produces an output based on inputs or assumptions, then a change in either may imply rework for the activity. The accomplishment of that rework then changes the activity's outputs, thereby potentially affecting other activities in the same way (second-order rework).

Rework in an upstream activity, caused by downstream activity, can also cause *second-order rework*. Thus, in, super-diagonal numbers represent the probability of iteration (returning to previous activities), while the sub-diagonal numbers note the probability of second-order rework (following first-order iteration).

In our basic model, these probabilities are held constant through successive iterations.

Figure 3 shows a sample DSM where the resource needed to complete the task is a predetermined person and there is feedback from task 8 to task 1, for example. That is, the DSM shows a 10% probability that task 1 will be repeated after task 8 finishes performing its job. Task 1 is a predecessor of task 7 and the DSM shows a second order rework probability of 0.5 for task 7 after redoing task 1. This means there exists 50% possibility that task 7 will be redone if task one is too. The DSM containing these probabilities is called a probability DSM. A similar matrix that reflects the proportion of the original task to be reworked is called the impact DSM. Both the probability and impact DSMs are necessary to carry DSM Simulation.

Activity name	1	2	3	4	5	6	7	8	9	10
Task 1	■							0.1		
Task 2		■				0.1				
Task 3			■							0.1
Task 4				■			0.1			
Task 5					■		0.1			
Task 6						■				
Task 7	0.5	0.5	0.1	0.4		0.2	■			
Task 8		0.5	0.8	1	0.2			■		
Task 9			0.6			0.7			■	
Task 10			0.3	1		0.1				■

Figure 3: DSM project example

Rework can also have a variable *impact* on an activity. While every likely, some changes in inputs can be absorbed by a robust activity with little impact. The consequences of changing other inputs may be more severe. The model uses an impact measure—the percentage of the activity that must be reworked—for each input to each activity.

Cho and Eppinger (2005) presented a DSM-based process model using advanced simulation. The model accounts for important characteristics of engineering design processes, including information transfer patterns, uncertain task durations, resource conflicts, overlapping and sequential iterations, and task concurrency.

The model addresses several limitations of previous analytical and simulation-based approaches. It can be applied to a wide range of processes, where iteration takes place among sequential, parallel, and overlapped tasks in a resource-constrained project. Increased understanding of realistic behavior of engineering design processes can be achieved through modeling information flows and predicting distributions of project lead time. The model is also useful for evaluating different project plans and

for identifying strategies for process improvements. Proactive risk management can be achieved by assessing the status of the project as it progresses.

2.2. Genetic Algorithms

Genetic Algorithms are robust stochastic search algorithms, inspired by the process of biological evolution (Goldberg, 1989). As per Wall (1996), genetic algorithms are well suited to solving production scheduling problems, because unlike heuristic methods genetic algorithms operate on a population of solutions rather than a single solution. In production scheduling this population of solutions consists of many answers that may have different sometimes conflicting objectives. For example, in one solution we may be optimizing a production process to be completed in a minimal amount of time. In another solution we may be optimizing for a minimal amount of defects. By cranking up the speed at which we produce we may run into an increase in defects in our final product.

As we increase the number of objectives we are trying to achieve we also increase the number of constraints on the problem and similarly increase the complexity. Genetic algorithms are ideal for these types of problems where the search space is large and the number of feasible solutions is small.

GAs consider a set of population of solutions as opposed to only one solution throughout local search. Following an initial population through a random generation, new solutions are produced by mating two existing ones (crossover) and/or by altering an existing one (mutation). Selection scheme determines which solutions survive via evaluating fitness or objective function. The fittest solutions take over the next generation while the others are deleted. This optimization process finishes with a

convergence to an optimal or near-optimal solution. The terms used in genetic algorithms are (Chakraborty, 2010):

- Chromosome (Individual): a set of genes; a chromosome contains the solution in terms of genes.
- Gene: a part of chromosome; a gene contains a part of solution. It determines the solution. e.g. 16743 is a chromosome and 1, 6, 7, 4 and 3 are its genes.
- Population: number of individuals present with same length of chromosome.
- Fitness: the value assigned to an individual based on how far or close an individual is from the solution; greater the fitness value better the solution it contains.
- Fitness function: a function that assigns fitness value to the individual. It is problem specific.
- Breeding (Crossover): taking two fit individuals and then intermingling their chromosome to create new two individuals.
- Mutation: changing a random gene in an individual.
- Selection: selecting individuals for creating the next generation.

The structure of the Genetic Algorithm is shown graphically in the flowchart of Figure 4. The details of each step in the algorithm are discussed in Section 3 where the model is described.

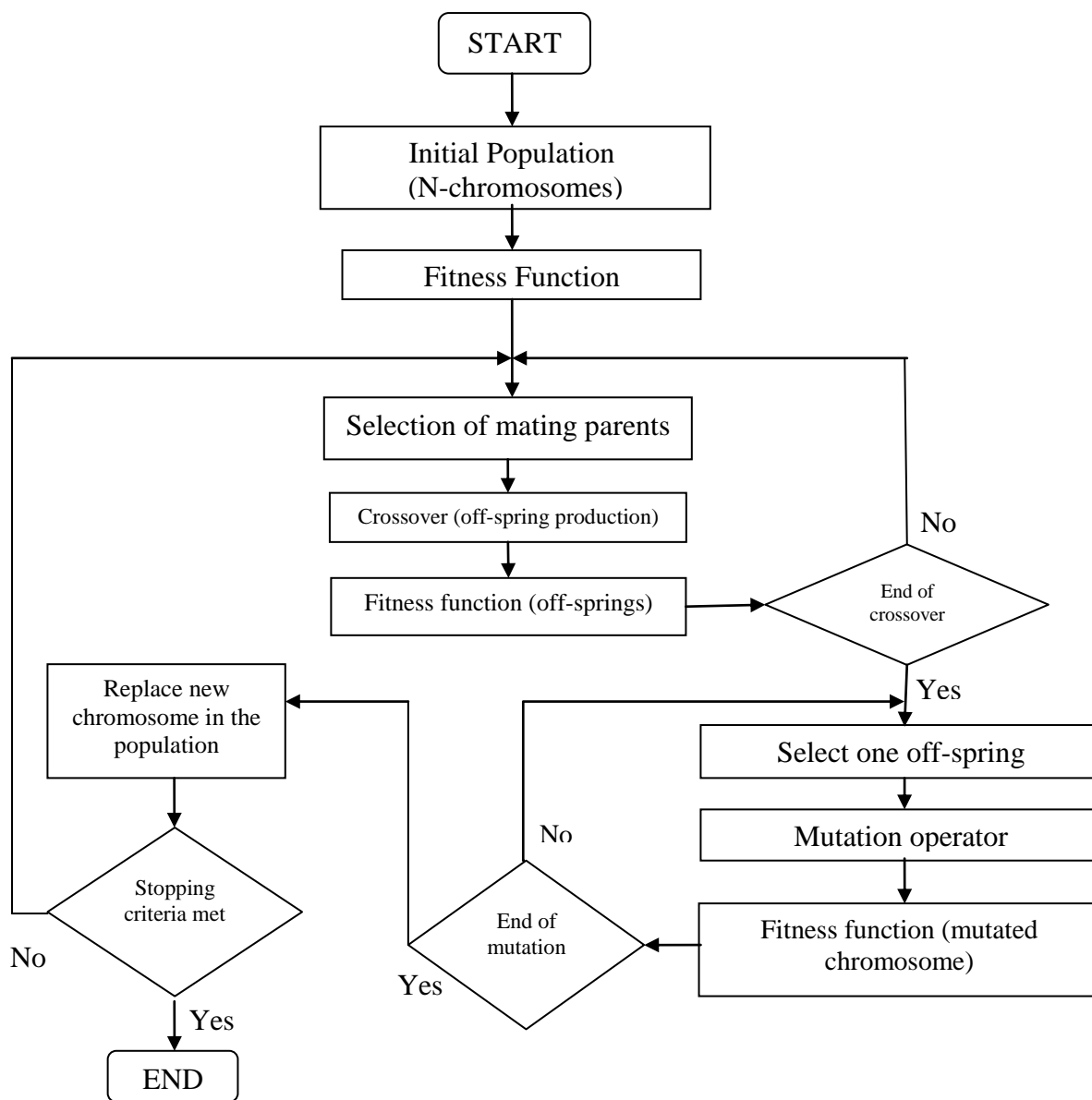


Figure 4: GA flowchart

The use of GAs in scheduling has been a topic of interest in numerous studies. Rogers (1994, 1996) for instance, developed a tool called DEMAID (Design Manager’s Aid for Intelligent Decomposition), which is GA-based, and made it more comprehensive to support design managers’ process structuring decisions using a DSM. McCulley and Bloebaum created another tool called GENDES (Genetic Design

Sequence) that re-orders a DSM, employing a GA in accordance to user-defined goals. The authors completed several tests associated with the GA parameters and the sequencing objectives, and compared the performance of GENDES with DEMAID. A third instrument called AGENDA (A GENetic algorithm for Decomposition of Analyses) augments the decomposition of multidisciplinary design optimization (MDO) quandaries by utilizing GAs as the optimization technique and DSMs as the system model. When compared to DEMAID/GA and GENDES, there is no paramount difference in the description of the underlying GA and its operators. On the contrary, Whitfield et al. (2003) mainly examined and associated GA crossover and mutation operators for sequencing DSMs. They found that earlier publications acknowledged the efficacy of GAs in combinatorial problems but did not exploit its whole repertoire. Accurate information about the setting of GA parameters and information about the incentive behind the use of certain GA operators was absent or only based on incomplete tests when present. Critical improvements in GAs such as hybridization or niching techniques were not present. In addition, since “competent GAs” had not been yet tapped into for DSM sequencing, an advanced investigation was critical, and has indeed been explored in later research papers.

Meier et al. (2007) also utilized Genetic Algorithms (GAs) within a DSM model in order to define an optimal sequence for a set of design activities. The optimality of a solution is contingent upon the goal of re-sequencing. In an activity ordering context, goals comprise one or more of the following: minimizing iteration/feedback, maximizing concurrency and diminishing development lead-time and cost. Meier et al. (2007) adopted a matrix-based illustration structure, the design structure matrix (DSM), for the information flow models. Assessments designated that certain DSM characteristics (e.g., size, sparseness, and sequencing objective) are the basis of grave problems for simple Genetic Algorithm (SGA) designs. To manage

this deficiency, Meier et al. explored the use of a competent GA: the ordering messy GA (OmeGA). Their investigations also confirmed the supremacy of the OmeGA over an SGA for hard DSM problems.

2.3. Resource Constrained Project Scheduling Problem (RCPSP) and Multi-Project Scheduling Problem (RCMPSP)

In Resource Constrained Project Scheduling Problem (RCPSP) a group of activities are to be executed, with this execution limited by two types of constraints; Precedence relationships that force to some activities to begin after the finalization of others and resource constraints which state that every activity requires a predefined type and amount of resources, which are available in limited quantities in every time unit for the whole project. In a multi-project environment, many projects are to be completed that rely on a common pool of scarce resources. In addition to resource constraints, there exist precedence relationships among activities of individual projects. A RCMPSP is portfolio of simultaneous projects with identical start times. Each project consists of precedence-constrained activities that draw from common pools of resources, which are usually not large enough for all of the activities to work concurrently. The goal is to prioritize projects and tasks in order to optimize an objective function, such as minimizing the delay of each project or of the whole portfolio. Such is the basic *resource-constrained multi-project scheduling problem* (RCMPSP).

It was earlier shown that the scheduling problem subject to precedence and resource constraints is NP-Hard (Garey & Johnson, 1979), leaving exact methods time consuming and inefficient at solving large problems and real-world applications.

Besides the impact of problem size, problem complexity also relies on how highly constrained the problem is (Browning & Yassine, 2010). There exist benchmark instances with as few as 60 tasks which have failed to be solved to optimality (Hartmann, 1998). Kolisch and Padman (1997) surveyed a number of techniques developed for resource constrained project scheduling. They include dynamic programming, zero-one programming, and implicit enumeration with branch and bound. Some examples of exact solution methods can be found in Demeulemeester and Herroelen (1992), Demeulemeester and Herroelen (1996), and Sprecher (1996).

Among them, branch and bound approach is the most widely applied. It, however, is rather a depth-first or breadth-first search in nature and cannot survive an exhaustive search in a large-scale project scheduling problem. Simulation modeling provides a new perspective to view RCPSP. A simulation model is proposed for multi-project resource allocation, interpreted as a multi-channel queuing Ghomi and Ashjari (2002). The innate drawback of simulation in time and cost, as well as deploying a particular simulation language will hinder its dissemination. Activity list based GA outperforms all other approaches in terms of average derivations from the optimal solution.

Genetic Algorithms were first used by (Davis, 1985) for the RCPSP as well as many other NP-Hard scheduling and sequencing problems. The application of GAs in production scheduling has expanded in recent years. To minimize the penalties caused by early and tardy delivery of components, assemblies and products is an emerging direction for GAs. Ip et al. (2000) applied GA to a multi-product scheduling in a multi-process production environment. Pongcharoen et al. (2004) enriched the problem used by Ip et al. (2000) by incorporating assemblies and components and constructing a tree structure for multi-product manufacturing process. Product

structure identifier, product instance identifier and operation identifier are integrated in chromosome representation. A repair process is performed to account for any mismatch caused by genetic operation. Lam et al. (Lam et al., 1999) developed a GA to minimize the overall completion time for designing products. The authors adopt a more general problem, which considers precedence relations among activities in an unrelated parallel machine environment with m machines and jobs can be processed on any machine. Meanwhile, resource constraint is relaxed by allowing an individual task to be performed by more than one resource (engineer), and the skill level of resources is weighed for each job. Generally, a large-scale task scheduling problem (TSP) for parallel projects deals with exponentially growing computational complexity. Most of the effort has been made to attack scheduling problems by relaxing resource constraint. Therefore, a task can be handled by different resources at different time and even unlimited resources. In this case, a multi-objective GA can be developed to minimize the makespan while employing the least number of resources.

A variety of operators, as well as implementation methods, had been developed for, or is applicable to scheduling problems. With a special focus on a single-project problem, Hartman developed a GA with permutation-based encoding (Hartmann, 1998) and introduced a self-adapting representation scheme which automatically determines the best schedule decoding procedure (Hartmann, 2002). Gonçalves *et al.* (2004) used a SGA approach for the RCMPSP based on a random key chromosome encoding and a schedule generation procedure which creates so-called parameterized active schedules. Recently, Valls et al. (2007) proposed a hybrid GA tailored to the RCPSP with a specific crossover and local search operator.

In the RCPSP, each listed task needs to be performed only once. Therefore, real coded GA is ideal for solving order-based problems, e.g. project scheduling

problem. Crossover operators such as cycle crossover (CX) (Oliver et al., 1987) and partial-mapping crossover (PMX) (Goldberg, 1989) can fix character duplication. Union crossover (UX2), another efficient order-based crossover operator, was proposed in (Poon and Carter, 1995). However, these operators do not guarantee the preservation of precedence constraints and would result in the need to repair illegal strings or give the constraint violation a penalty. Union crossover 3 (UX3) was then developed (Leu and Yang, 1999) based on UX2, which consider precedence constraints as well. When applied to precedence feasible parent chromosomes, UX3 ensures precedence feasible offspring. Their paper presents a useful multi-criterion model to solve the RCPSP, but does not consider parallel projects using resources that cannot be divided and they do not fully discuss the initialization of a precedence feasible population.

Zhaung and Yassine (2004) also employed a Genetic Algorithm to optimize the RCPSP problem by means of the Dependency Structure Matrix. Their suggested GA integrates stochastic feedback or rework of tasks. It has the ability to capture the local optimum for every generation and hence guarantee a global best solution. The Genetic Algorithm proposed, along with numerous variants of GA parameters, is tested on sample scheduling problems with and without stochastic feedback. This algorithm demonstrates to provide a quick convergence to a global optimal solution and detect the most likely makespan range for parallel projects of tasks with stochastic feedback.

Later, in a paper published in 2007, Lancaster et al. extended their research to encompass the use of Genetic Algorithms in the Project Scheduling Problem (PSP). Their research extends prior studies and integrates advances to include PSP, utilizing the Design Structure Matrix (DSM). In the industrial field, there is a need for the

presence of optimization algorithms which can support the determination of optimal schedules when offered a network with several possible choices. The optimization requirement may be understated, performing only slight resource leveling or more insightful, selecting an ideal means of implementation for numerous activities or evaluating a set of substitute approaches.

In another study in 2007, Yassine et al. suggested a Competent Genetic Algorithm (CGA), crossbred with a local search approach in order to diminish the overall period or makespan of the resource constrained multi-project scheduling problem (RCMPSP) without infringing on inter-project resource constraints or intra-project precedence constraints. The suggested GA with numerous changed parameters was evaluated on sample scheduling problems generated based upon Average Utilization Factor (AUF) and Average Resource Load Factor (ARLF). Subsequently, Yassine et al. established the dominance of the commended CGA over simple GAs and other known heuristics.

2.4. Network Generators

Efficiency tests of solution methods in the field of activity networks necessitate the employment of several randomly generated networks. Three elements distinguish every network: its size (N,A), random network structure and random values for the parameters of activities.

In a study published in 1993, Demeulemeester et al. developed a program to generate strongly random activity-on-the-arc project networks where all of the above elements can be randomized. The authors stated that when the size and structure of an activity are known, and randomization is restricted to the parameter of the activity, the AN is labeled as weakly random. Conversely, when size, structure and parameters are

randomized, the AN is labeled as strongly random. Many circumstances necessitate the generation and testing of a set of strongly random ANs, and this in turn entails the generation of a set of (N,A) pairs, where a number of network structures are generated using DM or AM for each pair. Subsequent activity parameters are also generated.

The number of nodes in an activity (N) is either specified or chosen randomly from the range, according to the solution method(s) to be established. Once N is specified, the number of arcs (A) is either specified or generated randomly from the range $[N-1, N(N-1)/2]$. Usually, a number of A values for each N will be utilized. Taking into account the number of structures related to a network size (N,A), then treating all the values of A in the aforementioned range in an equivalently probable way is not considered a correct randomization of A. Demeulemeester et al. demonstrated that a more precise option would be to link a weight with each value of A, the weight being proportional to the number of structures $G(N,A)$ and in agreement with the size (N,A). In other words, the distribution of A is established given N. Nonetheless, this option entails the listing of the structures $G(N,A)$ for all values of A in the abovementioned range, a task that is complicated for significant values of N. Accordingly, Demeulemeester et al. opted for a heuristic procedure to estimate the distribution of A.

Another random activity generator is ProGen, described by Kolisch et al. in 1995. ProGen is a generator of activity-on-the-node project networks that permits users to specify topology parameters. Schwindt (1995) later expanded it to ProGen/Max, and Agrawal et al. (1996) further developed the activity-on-arc project network generator DAGEN, where the user can specify the Complexity Index earlier described by Bein et al. (1992).

In another study published in 2000, Vanhoucke et al. describe another random network generator for activity-on-the-node (AoN) networks for diverse categories of

project scheduling problems. Their purpose was to build random networks with pre-specified parameters so the association between the hardness of a problem instance and the logic of the underlying network is controlled. Vanhoucke et al.'s generator generates problem instances that cover the entire range of problem complexity. It also ensures networks with predetermined complexity index values (CI) and order strengths (OS). Vanhoucke et al. maintain this as an advantage, as it is unfeasible to generate project networks with a low OS using ProGen/Max.

Similarly, Demeulemeester et al. (2003) depict the employment of RanGen, a random generator of AoN networks. RanGen also exploits the use of OS and CI in network generating, as these have been demonstrated in earlier research to serve as steady indicators of the hardness of different types of project scheduling problems. Single and multi-mode measures have been employed to illustrate the association between the presence of different resource types and their effect on problem hardness. RanGen was equipped with various resource measures, and in generating these measures, several decisions have to be taken. The density of the resource demand matrix shall be calculated in order to specify whether or not an activity uses a specific resource. This is established by calculating RF (Resource Factor) and RU (Resource Use). Further, when resource demand and availability are generated, RS and RC are used to determine resource strength and resource constrainedness, respectively.

Later, Gutierrez et al. (2004) developed HierGen, a generator that automates the generation of hierarchical networks to assist the simulation of real-life environments and the testing of planning and scheduling techniques. HierGen is a computer tool whose functionalities and internal algorithms can generate sets of project networks of flat, horizontally aggregated or hierarchical structure, of the desired size and complexity. These project networks can be drawn in a visually structured way. Once generated, they are stored in a database in a defined format,

where they can be accessed by the simulation or planning and scheduling tools that use them.

2.5. Project Characteristics

In an attempt to determine whether the majority of total resource requirements are in the front or back half of a problem's critical duration path, and in order to establish the relative size of the discrepancy, Kurtulus and Davis (1982) identified summary measures in the multi-project context, namely *ARLF*, or the *average resource loading factor*, and *AUF*, or the *average (resource) utilization factor*. Nonetheless, the defined measures had several limitations, and this encouraged our devising of enhanced measures, where we quantify resource distribution or loading, we use a *normalized ARLF (NARLF)* defined by Browning and Yassine (2010) as:

$$NARLF = \frac{1}{L \cdot CP_{max}} \sum_{l=1}^L \sum_{t=1}^{CP_l} \sum_{k=1}^{K_{il}} \sum_{i=1}^{N_l} Z_{ilt} X_{ilt} \left(\frac{r_{ilk}}{K_{il}} \right)$$

Where:

$$Z_{ilt} = \begin{cases} -1 & \text{for } t \leq \frac{CP_l}{2} \\ 1 & \text{for } t > \frac{CP_l}{2} \end{cases}$$

$$X_{ilt} = \begin{cases} 1 & \text{if activity } i \text{ of project } l \text{ is active at time } t, \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{ilt} X_{ilt} \in \{-1, 0, 1\}, CP_{max} = \text{Max}(CP_1, \dots, CPL),$$

K_{il} is the number of types of resources required by activity i in project l , and r_{ilk} is the amount of resource type k required by task I in project l .

This formulation allows for the normalization of the ARLF over the entire problem's critical path, rather than over that of every individual project.

In this study, we also use the *modified average utilization factor* or *MAUF*, proposed by Browning and Yassine (2010), in order to measure resource contention. *MAUF* is calculated for each resource type as an averaged ratio of the total amount required to the amount available in each time interval over the problem's critical path duration:

$$MAUF_k = \frac{1}{S} \sum_{s=1}^S \frac{W_{sk}}{R_k D_s}$$

Where R_k the renewable amount of resource type k is available at each interval, D is the size of the interval, S is the number of intervals (indexes in s), and the total amount of resource k required over any interval is given by:

$$W_{sk} = \sum_{t=1}^D \sum_{l=1}^L \sum_{i=1}^{N_l} r_{ilk} X_{ilt}$$

The MAUF for a problem involving K types of resources is given by Browning and Yassine (2010) as:

$$MAUF = \text{Max}(MAUF_1, MAUF_2, \dots, MAUF_k)$$

Finally, RCMPS literature brings into light the causal relationship between network complexity and the performance of scheduling rules and algorithms. Low-density networks are less precedence-constrained, and this allows for more degrees of freedom in determining a solution, hence making the activities harder to resolve (Browning and Yassine, 2010). On the other hand, high-density networks comprise additional precedence relationships among the activities, and therefore less possibility

for starting activities sooner or later without impeding the project. In this paper, we adopt Browning and Yassine's (2010) project network density measure:

$$C = \frac{4A' - 4N + 4}{(N - 2)^2}$$

Where A' is the number of non-redundant, feed-forward arcs, N is the number of nodes (activities), and C is normalized over $[0,1]$.

CHAPTER III

PROPOSED MODELS

In this section, we introduce the RCPSP with feedback (RCPSPwF). Then we describe the general mechanics of the GA-based approaches proposed in this thesis. Based on this, two GA-based algorithms are introduced to solve the RCPSPwF: Variable sample GA and variable length GA.

3.1. Problem Description

We consider resource constrained projects each represented by both its probability and impact DSMs. Each task in a project is defined with the following features:

- Name: each task has its own unique name or symbol
- Duration: each task has a Triangularly distributed duration (minimum, most likely, maximum)
- Number of reworks: each task has a pre-specified maximum number of reworks possible.
- Resources: each task requires certain resources defined by name and quantity.
- Learning factor: as tasks are reworked multiple times, the duration needed by this task to be performed the next time is reduced.

A project is considered to consist of J activities labeled $j = \{1 \dots J\}$. Precedence relations between these activities are given by sets of immediate predecessors P_j , indicating that an activity j may not be started before all of its predecessors are completed. Analogously, S_j is the set of the immediate successors of activity j . The set of resources is referred to as K . The duration of an activity j is denoted as d_j , its

request for resource k (*amount needed of resource k*) is given by r_{jk} . Once started, an activity may not be interrupted (i.e. pre-emption is not allowed). For every activity j , the duration takes three values; WCV_j , MLV_j , and BCV_j which are pessimistic (worst case value), most likely value and optimistic (best case value) respectively. The learning factor for an activity is IC_j ; where IC_j is a predefined improvement curve vector factor associated with each task and represents the percentage of learning each time a task is reworked such that the duration of a reworked task will be equal to: $(1 - IC_j) \times \text{original (nominal) duration of task}$. Finally, the objective is to determine a schedule with minimal duration or delay such that both the precedence and resource constraints are respected. Measurement of project portfolio delays will be discussed in the next section.

The estimation of task duration is one of the most challenging aspects in project scheduling and it is vital for cost estimation later on. Accurate task durations make scheduling easier, but these durations are rarely overestimated and frequently underestimated. This is why task durations are usually defined according to a triangular distribution that takes into consideration what is most likely to occur, as well as the optimistic and pessimistic expectations. However, estimation of duration is a complex procedure since task durations depend on different variables including labor skill, work time efficiency, mistakes, misunderstandings, staff and resource availability.

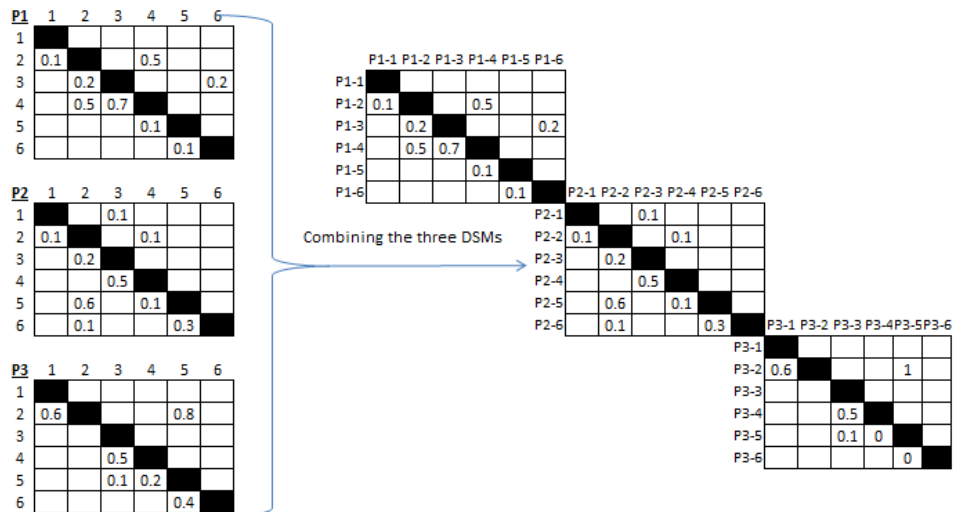
A task may have several predecessors. A task is only allowed to start when all its predecessors are completed. At the same time each task requires resources to be performed. These resources, in quantity and type, are usually shared by several tasks. This causes a task to wait for those resources to be free in order to start.

Feedback is not a definite event and it depends on the rework probabilities. The initiation of a feedback cycle occurs according to the DSM probabilities, and the magnitude of a rework for a specific task is defined in the rework impact DSM.

In a RCMPSP environment, a company has m concurrent projects $P1 \dots Pm$, each comprised of a set of activities, and each with its characteristics as defined earlier in this section. Although projects may be unrelated by precedence constraints, they depend on a common pool of resources and are therefore related by resource constraints. We consider a set of renewable resources where the per-period usage of activity j of resource k in project h is written as r_{jhk} .

When dealing with multiple projects, two approaches have been used: (1) a single-project approach, using dummy activities and precedence arcs to combine the projects into a single mega project, thereby reducing the RCMPSP to a RCPSP with a single critical path or (2) a multi-project (MP) approach, maintaining the RCMPSP and a separate critical path per project (Kurtulus and Davis, 1982). The second approach is used in our study for several reasons, since the latter is more realistic, newly studied, presents a greater improvement opportunity (Herroelen, 2005), and constitutes a decision guidance for managers.

Accordingly, using DSM, these multi-projects can be treated as a single project considering one large DSM combining all the projects' DSMs. As a result the portfolio will be treated as a whole project associated with the combined DSM (Figure 5).



Portfolio

	P1-1	P1-2	P1-3	P1-4	P1-5	P1-6	P2-1	P2-2	P2-3	P2-4	P2-5	P2-6	P3-1	P3-2	P3-3	P3-4	P3-5	P3-6	
P1-1																			
P1-2	0.1			0.5															
P1-3		0.2				0.2													
P1-4		0.5	0.7																
P1-5				0.1															
P1-6					0.1														
P2-1									0.1										
P2-2							0.1			0.1									
P2-3								0.2											
P2-4									0.5										
P2-5									0.6		0.1								
P2-6									0.1			0.3							
P3-1																			
P3-2													0.6					0.8	
P3-3																			
P3-4															0.5				
P3-5															0.1	0.2			
P3-6																	0.4		

Figure 5: Multi-Project Scheduling Problem DSM

3.2. Solution Methodology

The new GA-based solution methodology we introduce is based on Hartman's (1998) approach; permutation-based GA. This GA showed to be the most promising among other heuristics and metaheuristics such as the simulated annealing (SA) procedures, the local search approach, the disjunctive arc based two-phase procedure,

and the local constraint based analysis (LCBA) method. Also, recent numerical comparisons of meta-heuristic methods like Tabu search, simulated annealing and genetic algorithms indicate that the three most efficient procedures are (in order) the genetic algorithms of Alcaraz and Maroto (2001), the simulated annealing methods of Bouleimen and Lecocq (2003) and the genetic algorithm of Hartmann (1998).

A job sequence is assumed to be a precedence feasible permutation of the set of activities when each genotype (chromosome) is related to a uniquely determined schedule (phenotype). This schedule of activities is defined in the order that is prescribed by the sequence so that each activity is assigned to a set of predecessors and a feasible start time. This introduces a redundancy in the search space as distinct elements of the search space (i.e., genotypes) may be related to the same schedule. Exchanging some activities in a job sequence, we obtain a different precedence feasible genotype. However, both genotypes are related to the same schedule. Complex relationships between activities in a RCPSP with feedback make it nearly impossible to have a feasible schedule or a fit chromosome by random generation. Dealing with rework in a RCPSP introduces probabilistic tasks to a sequence which makes precedence feasible permutation probabilistic too. Having new precedence constraints according to rework makes it preferable to fit the individuals and sequences based on precedence only so that all possible schedules will be considered. Afterwards the makespan and fitness for the fit chromosomes or defined sequence are calculated taking into consideration resource constraints and possibility of parallel tasks. This gives a broader range of possible feasible sequences than assigning each activity to its earliest feasible start time.

String Representation: Chromosome length is variable due to the probabilities of rework. Each chromosome, represented by a sequence of tasks, must be completed to

determine the overall makespan. Figure 6 shows how different length chromosomes can result due to rework; *for chromosome tasks' annotation, please refer to section 4.*



Figure 6: Chromosomes of different length

Constraints: Optimization of the completion time is subject to the following constraints:

- **Precedence Constraint:** within each project, each task needs to be checked if its immediate predecessor(s) have been completed before being performed. Recall that precedence relationships are represented in the DSM. It can be formulated as: $t_j + d_j \leq t_i$, $c_{ij} = 1$ where c_{ij} is the cell value in the i^{th} row and j^{th} column of DSM.

- **Resource Constraint:** Two resource conflicting tasks cannot have overlapping time while performed. $t_j + d_j \leq t_i$ or $t_j \leq t_i + d_i$, if $r_i = r_j$ where r_i is the resource value present in the task definition.
- **Feedback:** Probability of rework.

Initialization: Highly constrained resource allocation problems have a small feasible search space. Therefore random generation of strings and incorporating a penalty into the objective function could result in the generation of a large number of infeasible solutions. So a permutation based simulation (Hartmann, 1998) is used to produce an initial population of precedence feasible individuals. This procedure proceeds as follows:

- Randomly select a task from all unselected task pool, and check if its immediate predecessor(s) are already selected. If not yet selected, continue this random selection until a satisfying task is found.
- Repeat step 1 until the set of unselected task is empty, which generate a chromosome that consists of all tasks.
- Repeat 1 and 2 until all chromosomes of population size are generated. This chromosome initialization procedure does not however give the makespan, task starting times or consider resource constraints. The initialization procedure simply provides precedence feasible solutions.

Makespan: The starting time is set for each task to obtain the makespan of the completion of all parallel projects for each individual string or chromosome of the population. The makespan is then used for the fitness measurement for evaluating members of the population. Based on the ordered lists of tasks given as the

chromosome representation, it allocates the limited resources to tasks in turn and keeps track of when resources become available. The steps to this process are as follows:

1. Initialize the available time of all resources to 0
2. Start with the first task index listed in the chromosome, compute:

$$\text{Max} \{ t_l + d_l, \max \{ s_i + d_i \} \}$$

where l is the index of the task last performed by the resource to use

i is the index immediate preceding task of this task

3. Repeat the step 1 and 2 until all the tasks have been scanned and computed and the makespan of string is then obtained.

It is important to note that based on the chromosome representation of task indexes, different chromosomes may potentially have the same fitness value and essentially represent the same schedule. For example, interchanging two tasks in the sequence that have the same starting time would result in a different chromosome, but the same schedule. Expressed in chromosome representation, such tasks can have as large as several tasks in between as long as these possess the same starting time. The swap of them poses no impact on the optimal solution. A question is therefore raised concerning how to capture the underlying schema. Similar issues were discussed in (Fang et al., 1993) concerning a Job shop Scheduling Problem, although the authors believe this number is small compared to the number of distinct schedules as problems become large.

Objective functions: Seeking to minimize project or portfolio delay (tardiness). We do this by defining a due date for each project, based on the length of its resource-unconstrained critical path (CP), and then measuring the delay beyond that point. Project and problem delays can be best measured in two ways, as defined in the following equations based on the three-project example problem shown in Figure 7. The measures O1 and O2 are the objective functions used in our proposed approach.

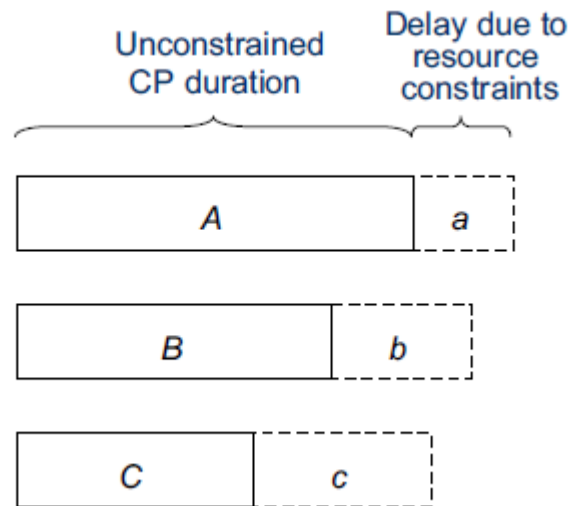


Figure 7: Example problem composed of three projects

$$O1: \text{Average percent delay} = \frac{\left(\frac{a}{A}\right) + \left(\frac{b}{B}\right) + \left(\frac{c}{C}\right)}{3} \quad (1)$$

$$O2: \text{Average percent delay} = \frac{\text{Max}(A+a, B+b, C+c) - \text{Max}(A, B, C)}{\text{Max}(A, B, C)} \quad (2)$$

The first measure (O1) is a project measures while the second (O2) is a problem (portfolio) measure. Focusing on O1 and O2, taking delay as a percentage of duration, allows comparison of projects and problems with different durations. For example, O1 recognizes that a 10-day delay on a 1-day project is probably worse than a 10-day delay on a 100-day project. Furthermore, O1 and O2 represent the individual

project manager's and the portfolio manager's respective points of view. That is, while a project manager will care about the effects of delays on his or her individual project, a portfolio manager might choose to focus on delays to the entire portfolio of projects. O2 is a less sensitive measure than O1, because in most cases it is affected only by delays to the longest project in a problem.

One more measure or objective function is defined and used throughout the coming sections, which is the project makespan itself defined as follows:

$$T = \max\{t_i + d_i \mid i=1,2,\dots,l_{chrom}\} \quad (3)$$

where t_i is the starting time of task I and d_i is the duration of task i

3.2.1. Variable Sample (Sampling) GA

This approach breaks down an iterative RCPSP to multiple RCPSP without feedbacks. Each new problem will represent a possible outcome and a sequence of activities for the initial problem. Applying GA to each problem separately results in a set of minimal makespans. This makes the function that calculates the project makespan, for each GA implementation, as follows:

$$\text{Minimize } T = \max\{t_i + d_i \mid i=1,2,\dots,l_{chrom}\}$$

Where t_i is the starting time of task I and d_i is the duration of task i

The procedure starts as follows and its steps are explained in details later:

1. *Generate a large initial population:* this population contains different project sizes and each project is represented by a chromosome. This variability in length is due to the uncertainty of rework. The size of this population is

defined by a “Sample Size” that is predefined and different than the population size defined for the GA. The sample size defines the number of projects to be generated. This sample size should be large enough to properly represent the whole project, but not to increase the time of execution of the GA process. (Check Section 5.2 for sensitivity analyses).

2. *Each and every chromosome of the initial population is then considered as a separate problem or project; each separate project consists of a set of tasks, some of which are initially real and well defined and others that may appear due to feedback and iteration. The latter become real and defined tasks for the separate project they appear in and so all its properties are defined (durations, predecessors and resources).*
3. *Each project or problem is treated separately and undergoes the GA process solely as if no feedback exists.*
4. *The optimal or near-optimal schedule of each project is selected and a distribution is made up of all the optimal schedules found for the predefined sample size.*

The inputs of this model are:

- The project DSM's for rework probabilities and impacts
- The tasks' definition (Name, Resources needed, Durations, Maximum number of reworks and Learning Factor)
- The sample size
- The number of generations
- The population size per generation
- Crossover factor
- Mutation factor

The sample size defines the number of projects to be generated. This is what is called the preparation for genetics. When this preparation occurs and the set of independent problems is defined, each problem will undergo the following GA process (See GA flowchart in Figure 1):

1. *Define an initial population*; this population is a set of possible combinations of tasks. The number of combinations or chromosomes is called the population size.
2. *Fitting the population*; the presence of precedence constraints may lead to rarely having a fit chromosome in the population in case this population is generated randomly and no fitness test goes on. The fitting of an unfit chromosome occurs through permutation as follows and demonstrated in the example of Figure 8:
 - a. *Move through the chromosome from left to right and for every task check if its predecessors are present before it.*

- b. If yes, the task is kept in place, else it is shifted to a position after all its necessary predecessors.
- c. Throughout the process we will obtain a set of fit chromosomes.

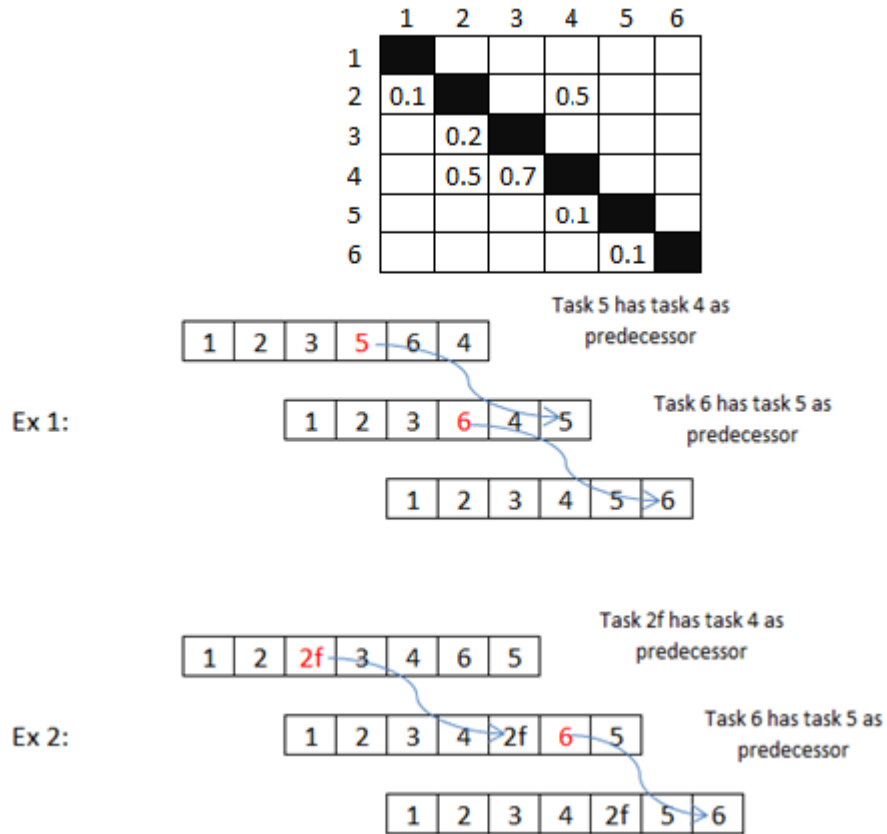


Figure 8: Permutation process

3. For each chromosome the duration is computed, and it is defined by the position of tasks, the use of resources and the parallelism in execution.
4. The fitness is calculated for each chromosome: $fitness = \frac{1}{duration}$; this gives that the higher the duration is the lowest the fitness of the chromosome.
5. Select chromosomes for crossover; roulette selection takes place. This selection functions according to proportional fitness. The roulette wheel shows how much an individual has potential for surviving. And the probability of individual 'i' to be selected is then given by: $p_i = \frac{F_i}{\sum_{i=1}^N F_i}$ where F_i is the

fitness of a chromosome and N is the population size. In cases where the durations are near, fitness is usually so close and so will be the probabilities, which may lead to distraction in the process of selecting the fittest. This is where we may use triple based tournament selection, where randomly three chromosomes are selected from the population and the one with the highest fitness is selected.

6. *Crossover between selected chromosomes happens and the children survive for the next generation;* after choosing a mating parent chromosome a random number between 0 and 1 is generated. If this random number is less than the crossover factor the crossover is performed else not and the chromosome survives to the next generation as it is. In case the crossover proceeds, and exist two mating parent chromosomes, another random number is generated and tested along the chromosome genes to determine a crossover point. In case of using one-point crossover operator, once this point has been determined, the parts of the chromosomes to the right of the point are exchanged producing new chromosomes. On the other hand, using two-point crossover operator, two crossover points are determined, the parts of the chromosomes in between those points are exchanged producing new chromosomes. The chromosome selected for crossover is already a competent chromosome for survival, this is why we should maintain a high probability that this chromosome will survive as it is. However to maintain the evolution of the population from generation to another crossover should still happen. This means the crossover factor should be high, but not high enough not to give chance for immediate survival of fit chromosomes and well as not to increase the execution time of the problem. This is why it is commonly set to be between 0.6 and 0.8. (i.e.

Hartmann (1998) and Lancaster (2007)) and this was validated in this thesis in section 5.2 by sensitivity analyses.

7. *Each of the resulting chromosomes is subjected to a mutation process; a set of random numbers between 0 and 1 is generated through the chromosome genes and mutation alters one or more genes specifically ones with a random number less than the mutation factor. The main use of mutation factor is to avoid the case of getting stuck with the same population. However it should not alter already fit and competent or optimal chromosomes. This is why this factor is chosen to be small enough ([0.01, 0.1]), (i.e. Hartmann (1998) and Lancaster (2004)) and this was validated in this thesis in section 5.2 by sensitivity analyses.*
8. *The chromosomes resulting from the crossover and mutation are tested for feasibility and fitted before they survive to the next generation*
9. *This crossover and mutation process proceeds until the new generation is produced and the defined population size is reached.*
10. *The new generation will undergo the same procedure as the previous one starting with the selection process.*
11. *The whole GA stops when we reach the N^{th} generation where N is the number of generation defined.*
12. *This N^{th} generation for each problem will have the optimal or near optimal duration and project schedule of this problem.*

A large project or set of projects with many potential reworks may result in a huge number of possible projects or scenarios (i.e. sample size). Usually in GA the number of generations is $m \sim 2m$ and the population size is $2m \sim 4m$, where m is the number of tasks of a project, according to Balaji (2011). In this case and in the presence of

fitting $m/2 \sim m$ generations and $m \sim 2m$ population size would be sufficient. (Check Section 5.2) Finally, we note that there is no need to take the feedback probabilities into account when regrouping the data for the whole project. These data are treated equi-probably as the sample is assumed to show all possible scenarios. Each chromosome already includes its probability of occurrence, which means when we get to analyze the data we do not need to consider any probability of existence of any chromosome present in the final generation.

Each scenario from the sample will result in an optimal or near optimal solution. The sample size being known (i.e. S), we will have S projects. After applying GA to those projects the fittest chromosome from the last generation of each project is picked as the optimal solution. As a result, we will have S solutions which will be grouped and considered as the distribution of the whole iterative project/problem.

3.2.2. Variable Length GA

This approach tends to change the stochastic rework into deterministic tasks, taking into consideration the rework probabilities and rework impact. Assuming that the maximum number of potential reworks for a given task is known and defined, the DSM is changed into a lower triangular matrix. However each task will hold a new trait noted as the probability of execution. This probability determines whether the task will be chosen or not while forming a chromosome in the GA. One GA is then performed with variable length chromosomes. The procedure is as follows:

1. *According to the DSM and the number of reworks of each activity, all possible tasks to be performed are defined, both original and reworked tasks: For*

example if task 1 in project A (A1) has a potential number of reworks equals to 2 then new tasks are defined and denoted as A1f and A1ff.

2. *Original tasks maintain their properties, while reworked tasks are defined by their durations, resources, predecessors, and probability of occurrence: the reworked task needs resources same as its associated original task. If this task is in its first rework level, its predecessors are the original tasks that initiate the feedback cycle, while if it is in its nth rework level, its predecessors are the nth rework level tasks of the initial predecessors of its original associated task. (see example in Figure 9) For exact tasks, probability of occurrence is 1, while for others it depends on their feedback and rework probabilities. The durations of reworked tasks are calculated according to the following rule:*

$$\text{Duration} = \text{initial duration} \times \text{rework impact} \times (1 - \text{learning factor})$$

In this way the DSM will be changed into a lower triangular matrix only, but will expand in size. The reworked tasks are treated as real independent tasks but with an associated probability of occurrence. The probability of occurrence for a given task is the probability that this task is going to show up in the schedule.

Task	Task Detail	Predecessors
1	Original work	-
2	Original work	1
3	Original work	2
4	Original work	2 3
5	Original work	4 all rework cycles initiated by task 4
6	Original work	5
2f	1st order rework	4
3f	2nd order rework	2f 3
2ff	1st order rework second time	4f 2f

	1	2	3	4	5	6
1	■					
2	0.1	■		0.5		
3		0.2	■			
4		0.5	0.7	■		
5				0.1	■	
6					0.1	■

Figure 9: Tasks predecessors

3. *Preparation of genetic algorithm here occurs by defining all the tasks forming a large messy pool of activities. From this pool the initial population is formed and it will constitute the initial population for the GA or the 1st generation.*

The GA in this method is as follows and is shown in Figure 11:

1. *Form an initial population: this population is a set of possible combinations of tasks. The number of combinations or chromosomes is defined by the population size which will be relatively large to demonstrate all possible combinations.*
2. *The initial population chromosomes are tested for feasibility and fitted through permutation.*
3. *Duration and fitness of each chromosome are calculated.*
4. *Chromosomes are grouped into smaller pools according to their lengths. Each pool will then have a number of chromosomes called length counter. According to this number each pool will have a probability of selection as follows:*

$$\text{probability of group selection} = \frac{\text{number of chromosomes in group}}{\text{number of total chromosomes in the population}}$$

5. *To select mating parent: A pool of those length pools is selected and then randomly a chromosome from this pool is selected. This is done randomly to maintain the lengths and the possible projects throughout the generations; else selection according to fitness will end up having the shortest schedules only in the final generation which does not demonstrate the reality of the problem behavior.*

6. *Crossover between selected chromosomes takes place. In this case crossover may occur between two chromosomes of different lengths. This crossover again may be one or two point crossover.*
7. *Each of the child chromosomes is subjected to a mutation process.*
8. *Again to maintain the presence of all lengths and the survival of the fittest in each length, the resulted children are fitted and both the child and the parent of the same length are compared with the one with highest fitness surviving. In this case sometimes the parent is surviving and not the child.*
9. *This crossover and mutation process proceeds until the new generation is produced and the defined population size is reached.*
10. *The new generation will undergo the same procedure as the previous one starting with the selection process.*
11. *The whole GA stops when we reach the N^{th} generation where N is the number of generation defined.*
12. *This N^{th} generation will have a set of optimal or near optimal durations and project schedules of this problem, with its chromosomes being of different lengths and resembling different possible scenarios of the whole problem.*

Figure 10 illustrates steps 4 to 8 for the shown DSM and population example.

	1	2	3	4	5	6
1	█					
2	0.1	█		0.5		
3		0.2	█			
4		0.5	0.7	█		
5				0.1	█	0.2
6					0.1	█

A population of 12 chromosomes resulting from the DSM:

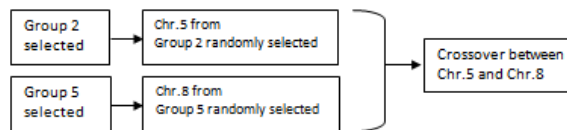
- Chr1: [1 | 2 | 3 | 4 | 5 | 6]
- Chr2: [1 | 2 | 3 | 4 | 2f | 5 | 6]
- Chr3: [1 | 2 | 3 | 4 | 2f | 3f | 5 | 6]
- Chr4: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 2ff | 5 | 6]
- Chr5: [1 | 2 | 3 | 4 | 5 | 6 | 5f]
- Chr6: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 2ff | 5 | 6]
- Chr7: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 5f | 5 | 6]
- Chr8: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 5 | 6 | 5f]
- Chr9: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 2ff | 3ff | 4ff | 5 | 6]
- Chr10: [1 | 2 | 3 | 4 | 2f | 3f | 5 | 6 | 5f | 6f]
- Chr11: [1 | 2 | 3 | 4 | 2f | 3f | 5 | 6 | 5f | 6f | 5ff]
- Chr12: [1 | 2 | 3 | 4 | 2f | 3f | 4f | 5 | 6]

Step 4:

	Group1	Group2	Group3	Group4	Group5	Group6	Group7
Chromosome Length	6	7	8	9	10	11	12
Chromosomes in length groups	Chr.1	Chr.2 Chr.5	Chr.3	Chr.12	Chr.4 Chr.6 Chr.7 Chr.8 Chr.10	Chr.11	Chr.9

Group	Probability of selection
Group 1	= 1/12 = 0.0833
Group 2	= 2/12 = 0.1667
Group 3	= 1/12 = 0.0833
Group 4	= 1/12 = 0.0833
Group 5	= 5/12 = 0.4167
Group 6	= 1/12 = 0.0833
Group 7	= 1/12 = 0.0833

Step 5:



Steps 6, 7 and 8:

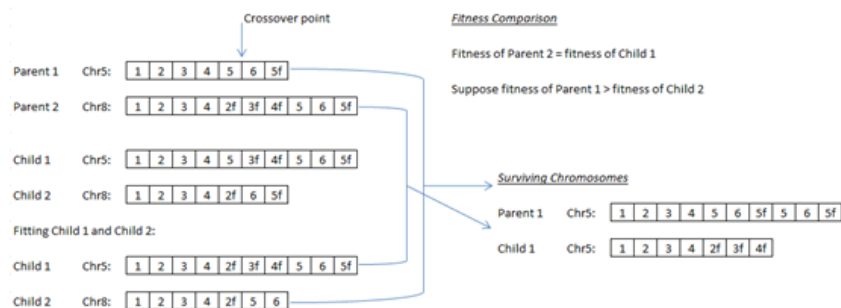


Figure 10: Example for steps 4 to 8 of Variable Length GA

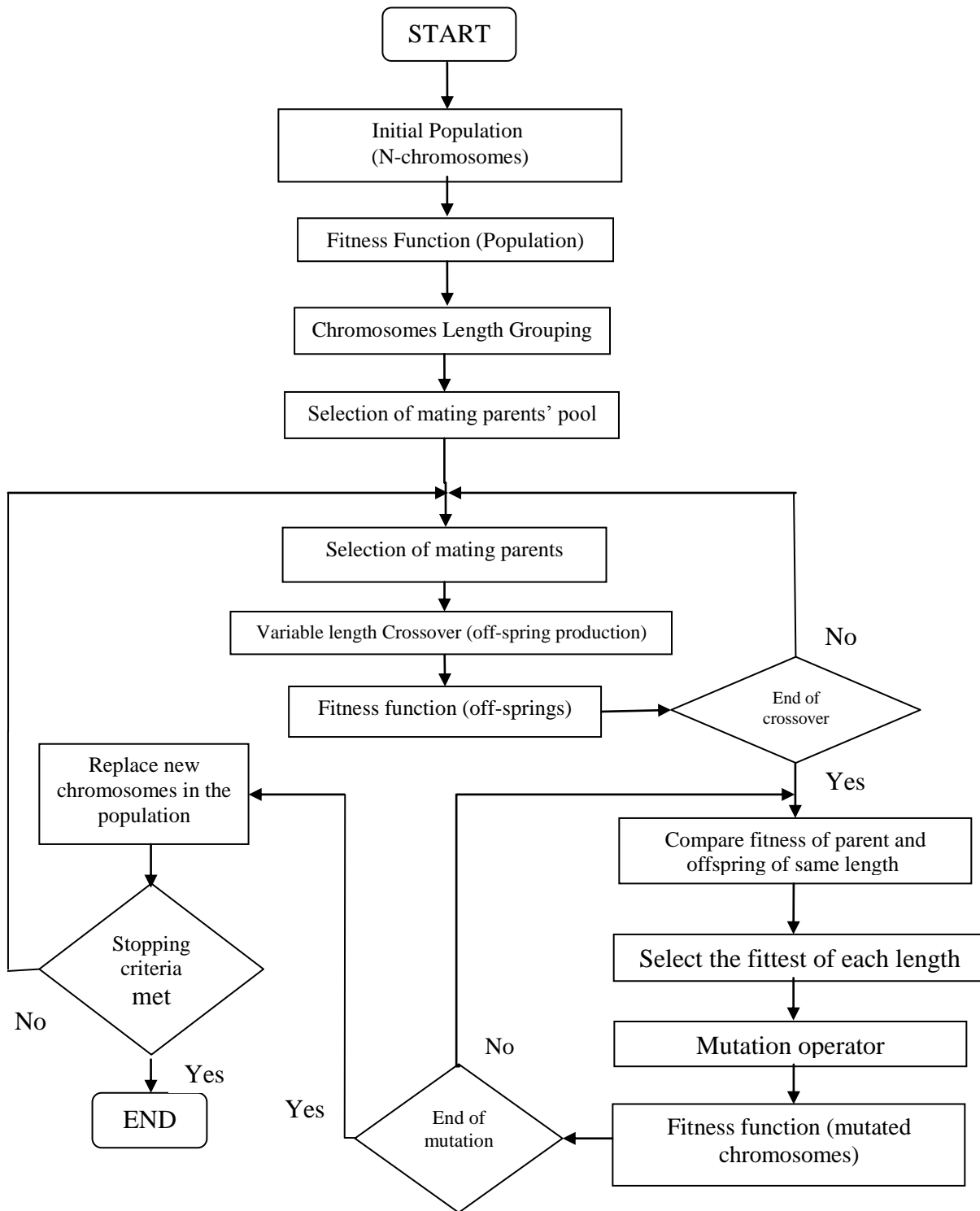


Figure 11: Variable Length GA Flowchart

A large pool containing all of the possible tasks is defined. From all of the tasks defined in this pool, the initial population is generated of size $4m$ where m is the number of possible tasks (exact and iterative) or $6m \sim 10m$ where m is the number of

exact tasks. This population will contain chromosomes of different sizes. GA is then applied to this population and crossover occurs between different size chromosomes.

This way all sizes and scenarios are maintained throughout the generations and the final generation will contain optimal or close to optimal solutions of different sizes. This approach is compact in number of generations and GA processes, but huge in size of population. It may miss some chromosome lengths and possibilities also. It takes time to generate initial population but decreases the time of the GA processes present in the previous approach.

The variable size GA ends by considering the final generation as a set of optimal or near optimal solutions for the possible scenarios or schedules of the project/problem or set of projects. The chromosomes of the final generation and their durations are used to form a distribution of the project.

In both approaches feasibility check occurs through an instantaneous feasibility check indicator (IFC). Any set of tasks represented in a chromosome, may be feasible or not according to the following constraints: Precedence constraints consideration, all first time (original) tasks exist, and no duplicates exist. As a result the feasibility check function instantaneously checks for those and applies changes to the sequence of tasks for the resulting chromosome to be feasible and applicable which eventually increases its fitness from a null. At each generation all the generated chromosomes are checked for feasibility which alters non-fit ones and introduces a fit whole generation. This check occurs in the following manner as demonstrated in the example of Figure 12:

1. Check if any duplicate exists.

2. Check if all the first time tasks presented in the project DSM exist in the chromosome or chain of tasks.
3. If not then insert a missing exact task instead of any duplicate, if exists; else instead of the last level reworked (non-exact) task in the chromosome.
4. Exchange positions of tasks by dragging a task to a position where its precedence constraints are fulfilled and located before it.

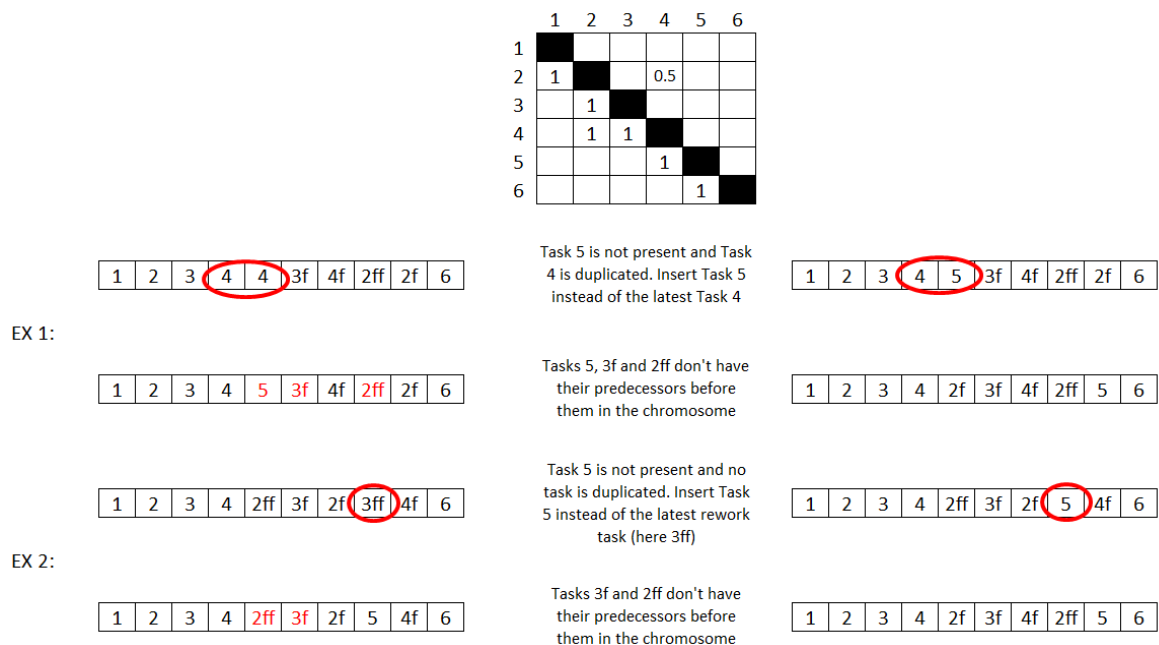


Figure 12: Instantaneous Feasibility Check

This instantaneous check may have both its advantages and drawbacks. It surely produces better fit chromosomes which may fasten reaching a final close to optimal or optimal solutions and fitness distribution. It also takes away the possibility of survival of non-fit chromosomes to the coming generation.

On the other hand, it may lock the GA process to a solution which is not optimal since some solutions may not show anymore. Another disadvantage is that the generation will no longer be random and contain different chains. Moreover it may be

time consuming as the check for feasibility for a large population through all generations takes much time to execute.

However, the randomness in such problems and applying GA may result in non-fit chromosomes throughout all the generations because of the large permutation existing in the initial generation and because of the possibility duplicates upcoming and exact task leaving a chromosome after several generations.

Differences between regular GA and our two GA approaches are presented in

Table1:

Differences from regular GA

Variable Sample GA	Variable Length GA
<i>GA applied to a sample separate problems (An iterative RCPCP problem is broken down to RCPCP without feedback problems)</i>	<i>New tasks (iterative) are defined and DSM extended with new precedence relationships</i>
<i>Generation of Large Initial Population (sampling)</i>	<i>Tasks are associated with a probability of occurrence each.</i>
<i>Population individuals fitting (not random)</i>	<i>Selection for crossover happens in two stages: Selection of chromosome length and selection of chromosome itself</i>
<i>Fitting chromosomes after crossover and mutation</i>	<i>Population constitutes of variable length chromosomes</i>
<i>Most fit chromosome from the last generation of each sample is considered only, and those of all samples sum up to present the optimal generation of the initial iterative problem.</i>	<i>Crossover exists between different lengths chromosomes</i>
	<i>Fittest parent and fittest child survive to next generation</i>

Table 1: Differences between S-GA/V-GA and regular GA

3.3. Operators for GA

Applying the GA approaches introduced, note that both one-point or two-point crossovers may be chosen to be used in our model. Also both roulette and tournament selections may be chosen to be used in our model. Furthermore a by-pass operator may be used to by-pass the fittest chromosome from a generation to another new generation driving the algorithm towards optimality. Using the by-pass, before generating any new generation the fittest chromosome from the previous generation is selected and moved to the new one immediately.

3.4. Guideline to Scheduling Decisions

One of the main objectives of this thesis is to give guidelines for managers on which they can base their scheduling decisions for projects and portfolios. Using sampling GA, the fittest (optimal) chromosome in the final generation for each of the projects in the sample is chosen and a distribution is formed from all those fit chromosomes. This would be the duration or makespan distribution of the project. On the other hand, the chromosomes of the end generation in case of variable length GA constitute the distribution of the whole project. Considering this project distribution from either approaches the manager may decide to go with the makespan having the highest probability/frequency. This makespan may correspond to several schedules or chromosomes and the manager chooses one of them as the project's schedule. However, with the progress of the project in the case of uncertain (iterative) problems, new tasks (reworked) may occur which forces the manager to change the chosen schedule. Accordingly, the manager will choose again a new schedule that has the

same sequence that already happened till the time of the new decision and the highest probable makespan too.

To show how managers can benefit from GAs in having a project schedule, Browning & Eppinger (2002) example is studied using sampling GA with the following parameters; sample size = 42, number of generations = 7, population size = 14, crossover factor = 0.7, mutation factor = 0.02.

The resulted duration distribution is shown in Figure 13:

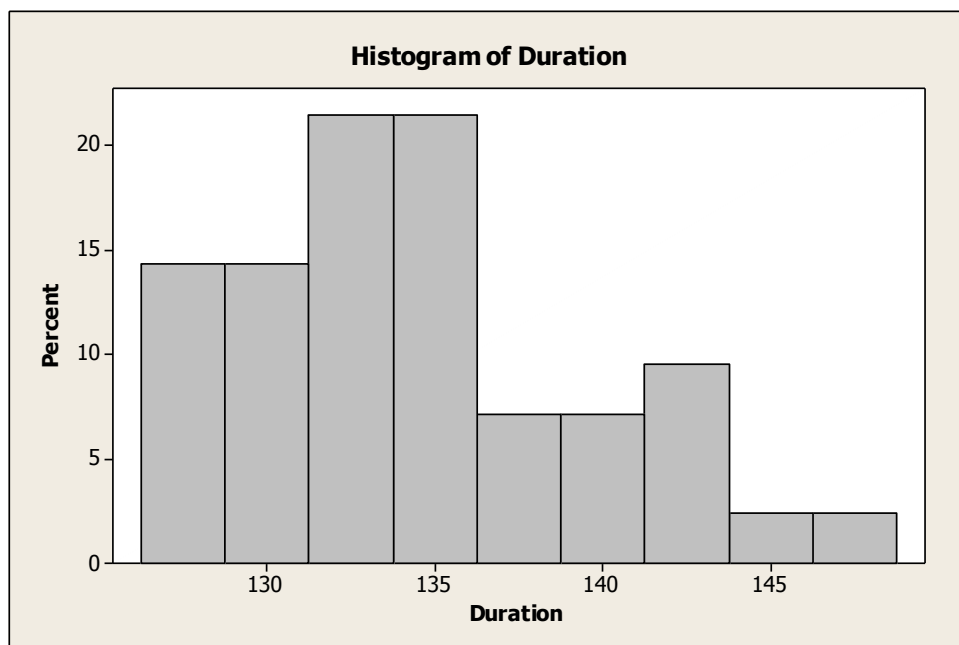


Figure 13a: Duration/Makespan Distribution for B&E example

A manager may choose a schedule that has duration of 135, knowing it is the one of the most probable schedules (20 %). We assume that the chromosome (schedule) chosen is:

B1-B2-B3-B4-B5-B6-B7-B8-B9-B10-B11-B12-B13-B14.

After task 4, if task 3 was reworked, the chosen schedule is no longer applicable and the manager needs to go with a new schedule. This schedule is also chosen so that it has a most probable duration and it is most likely to happen. Looking

at Figure 13 or even looking at a new distribution shown in Figure 14 for all schedules that start B1-B2-B3-B4 and task 3 is reworked (B3f) after task 4 is done, The new schedule will be B1-B2-B3-B4-B3f-B5-B6-B8-B9-B7-B10-B11-B12-B13-B14 of total duration (makespan) of 132.25.

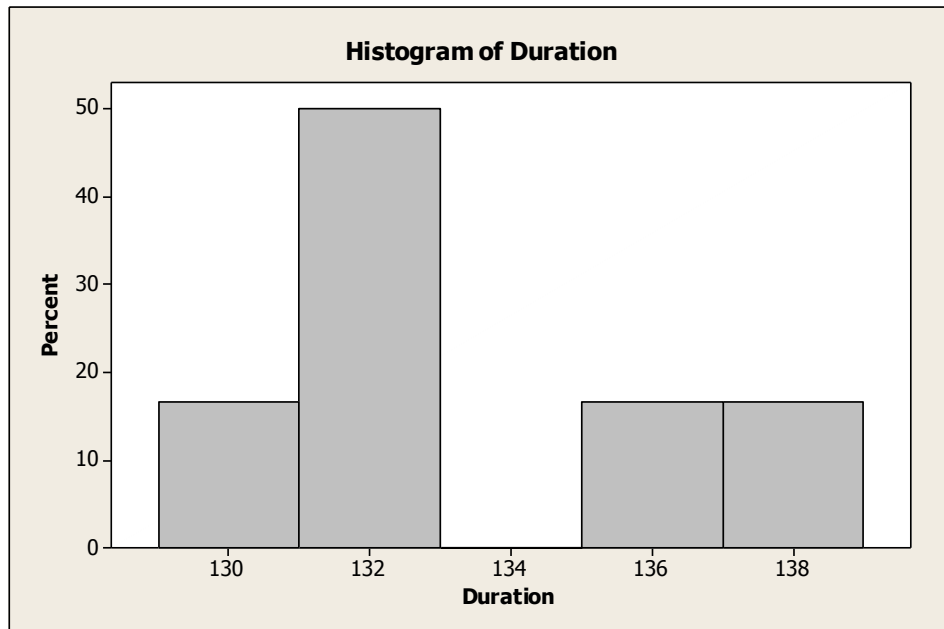


Figure 13b: New duration distribution

Browning & Eppinger (2002) introduced a simulation process to handle project scheduling problems with feedback. The GA differs from the simulation by making sure that whenever a set of activities is present the resulting schedule has an optimal makespan. This will result in shorter makespans and will shift the whole project distribution to the left. Our concern is not only demonstrating the possible project schedules but also minimize the project total duration. According to Browning & Eppinger if a schedule contains a number of defined tasks, it won't have more than one possibility since the shortest duration task is always chosen to start whenever its precedence constraints are met. GA considers different possible combinations or sequences of those tasks searching for the most optimal one.

Considering the results of the GA, three simulations are done to those results to check the impact of following the guidance for scheduling decisions mentioned above on the optimal problem solution given by GA.

The histogram in the following figure compares the distribution of the durations resulting from the GA with that resulting from the simulations.

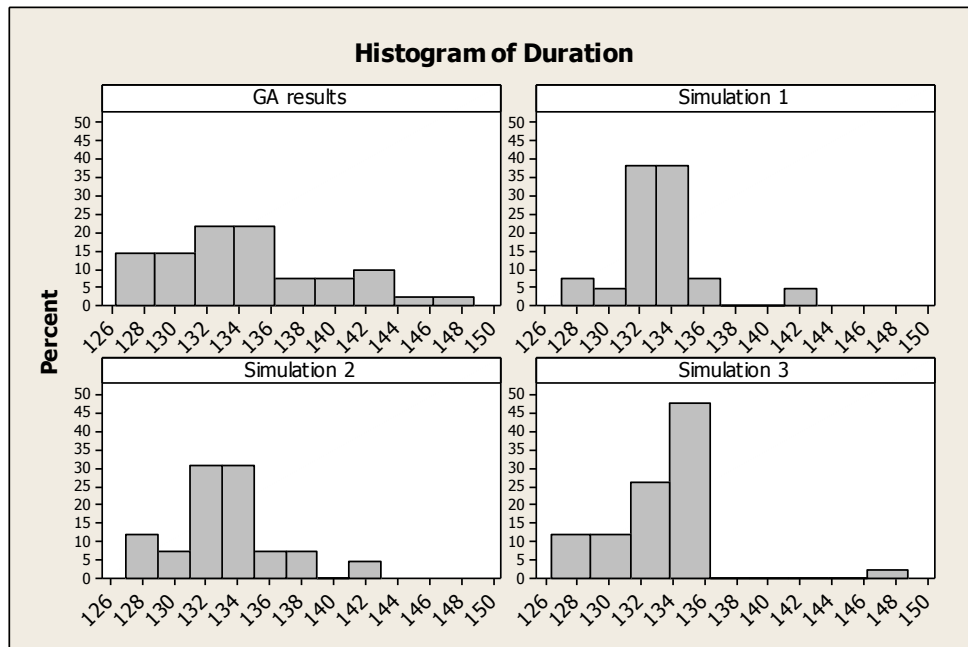


Figure 14: Histograms of Duration of GA vs Simulation

It is noticeable that the choice of the manager for the schedule having the most frequent duration as a first decision will make the resulting duration distribution tighter from the edges towards the mode as the frequency of the durations near to the chosen duration will be the highest. For example (Figure 14) according to GA results the schedule duration is 40% between 131 and 136 days, while due to the 3 simulations this probability increases to above 80%, 60% and 70% consecutively.

A summary of the descriptive statistics of the four cases shows in table 2:

	Mean	StDev	Variance	Minimum	Median	Maximum
GA results	134.36	5.04	25.40	126.54	133.64	146.41
Simulation 1	133.26	2.77	7.67	128.07	132.96	142.31
Simulation 2	133.40	3.29	10.81	128.07	132.96	142.31
Simulation 3	133.05	3.18	10.09	127.55	133.64	146.41

Table 2 : Statistical Comparison of GA with Simulation

The mean is almost the same for both GA results and simulation. This is because the simulation is based on a first decision by a manager to choose the schedule having the most probable duration and the project duration nearly follows a normal distribution. The standard deviation and variance decrease significantly after simulation and this is expected also due to the decision guidelines discussed early in this section.

CHAPTER IV

EXCEL IMPLEMENTATION

The two proposed GAs are implemented in Java with an Excel interface for input and output presentation. A project is represented in an Excel file in three sheets. The first sheet shows the project's DSM and rework matrix. The second sheet contains all the available resources defined by name and quantity. The last sheet is a set of all activities present in the project set in a table defined by their properties as mentioned before.

A task is notated by the project's name, the task's number and the level of execution. For example "A1" means the first task in project A, while "A1ff" means the first task of project A in its second rework level, where each "f" resembles a rework level. The absence of "f" means that the task is a real one and is performed for the first time.

The results are also generated in excel sheets showing the time of execution, the critical paths, the detailed generations and chromosomes, as well as the final generation population with durations, delays and the values of O1 and O2 for each chromosome or project network. An example of the input and output excel sheets is presented in Appendix A and B respectively. The interface of the models' software is shown in Appendix C.

CHAPTER V

COMPUTATIONAL RESULTS AND INSIGHTS

The previous sections introduced two GA approaches used in resource constrained single and multi-project scheduling with feedback. Those GAs are designed to cope with network characteristics as precedence constraints, precedence feasibility, resource constraints, parallel networks as well as iterative activities. The GA models as well as their characteristics are tested for validation based on different benchmarks, and then are used in the RCMPSP test case where the results are compared to those of different priority rules heuristics.

5.1. Data used in Analyses:

A set of examples (3 projects per problem, 20 activities per project, 4 types of resources per activity) were considered varying several factors. Four main factors are varied; NARLF (-2, 0, 2), MAUF (0.7, 1.1, 1.5), iteration (0, 0.1, 0.25) as well as the complexity level C (HHH, HLL, LLL), where HHH stands for high complexity problems and LLL resembles low complexity problems, with HLL is a complexity level in between. Three replications were considered for each combination of those factors. This resulted in a total of 243 problems that were solved using the proposed two GA-based approaches.

5.2. Model Calibration by Performing Sensitivity Analyses on GA Factors and Network Characteristics

Sensitivity analyses in this section shows how GA parameters are calibrated based on the network characteristics. Model calibration is based on performing on sensitivity analyses on the data (i.e. set of projects) described in Section 5.1.

The data set considered in this section is a part of the set of examples described in Section 5.1. The extreme values of each of the main factors are considered: NARLF (-2, 2), MAUF (0.7, 1.5), iteration (0, 0.25) and C (HHH, LLL). O1 and O2 are considered as the objective functions in sensitivity analyses. This data set is used to study the GA parameters and their effects, in addition to the interaction between project settings and the GA parameters/solution. It is interesting to determine if the GA works best or worst with particular combinations of GA parameters and also to result in a calibrated GA model to be used for our case studies. To study the effect of the network generators on the choice of the GA parameters, both project/problem characteristic and the GA parameter are varied with one another and the results (O1 and O2) are presented in main effect plots. For example, to study the interaction between number of generations and MAUF, O1 and O2 are calculated for each combination, i.e. (MAUF=0.7, number of generations = m).

The values of GA parameters and their variation are also defined for the analyses; Crossover Factor (0.2, 0.4, 0.7, 0.9), Mutation Factor (0.02, 0.3, 0.7), Number of generations (m/4, m/2, m, 2m), Population size (m/2, m, 2m, 4m). In non-iterative projects the number of tasks is defined, which is not the case for iterative projects. The existence of different scenarios with different number of tasks in iterative projects makes it important to define the sample size (S) in the Variable

Sample GA such that it properly represents the projects. For example, considering (Iteration = 0.25, C = HHH, NARLF = 2, MAUF = 1.5) a sample size of m showed chromosomes lengths (projects) of minimum 75 and maximum 106 tasks (range = 31); however a sample size of $2m$ showed projects having 71 tasks and others having 122 tasks (range = 51). In this way increasing the sample size increased the range of chromosome lengths and gave a wider presentation of the project or portfolio.

The number of tasks range between the shortest and longest possible scenarios (chromosomes), is used as a choice criterion for the sample size as shown in Figure 13 below. This plot results from determining the shortest and longest chromosome or project generated according to the sample size.

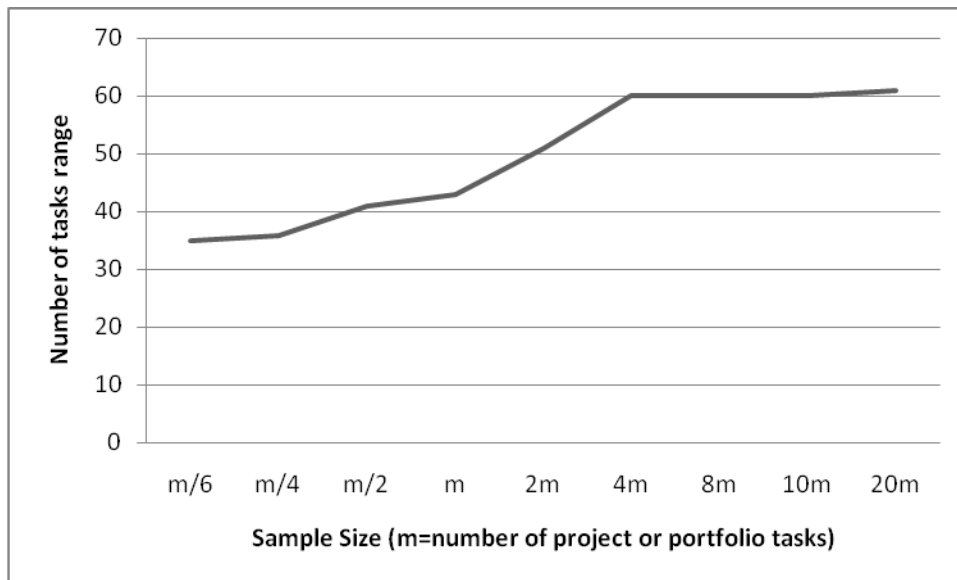


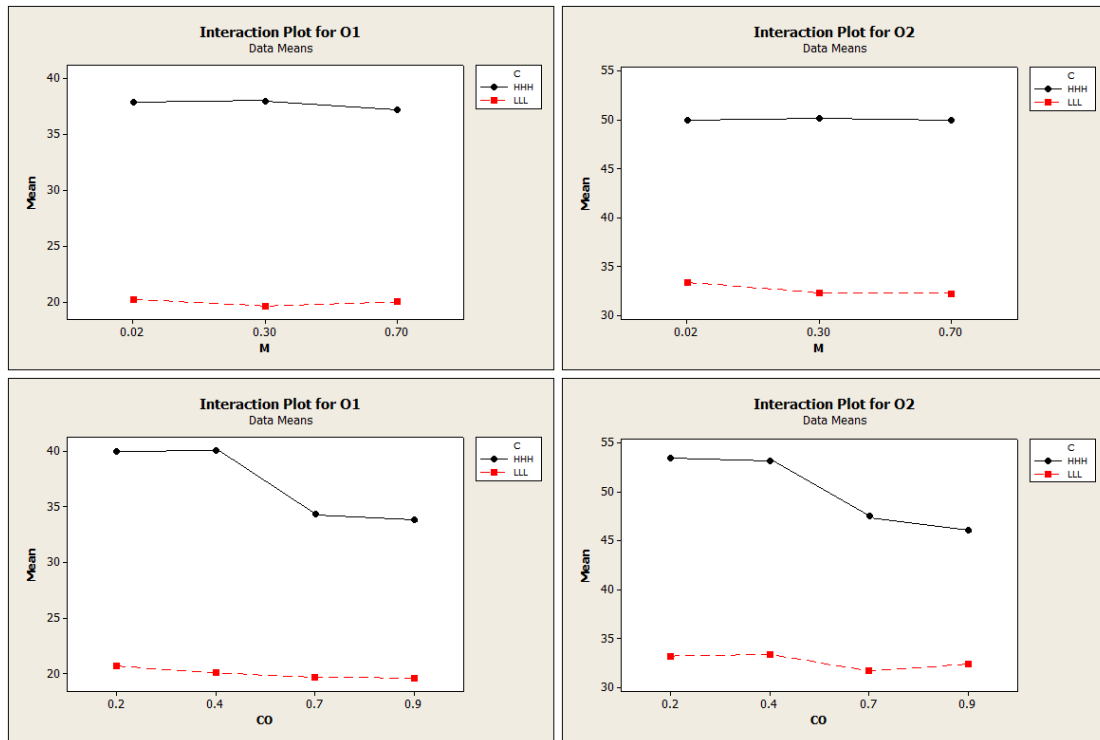
Figure 15: Range of number of tasks based on Sample Size

Figure 15 shows that it is best to choose a sample size between $2m$ and $4m$ as below $2m$ it will not represent the project well and beyond $4m$, no significant change in the number of tasks range exists which lead to unneeded increase of GA execution time.

Figure 16 shows that higher mutation factor doesn't lead to more optimal solutions. Increasing the mutation factor (M) increases the execution time of the GA since chromosomes will not directly go to next generation after crossover and will be subjected to fitting due to possible network constraints violence due to mutation. Besides mutation may alter already fit chromosomes and worsen the fitness function. The main reason behind the mutation operator here is to avoid that the GA gets stuck in a non-optimal solution as well as to sample the solution space widely and to broaden the search.

The interaction of complexity with crossover and mutation factor respectively is studied by applying the GA to the data set for three levels of mutation (0.02,0.3,0.7) as well as four levels of crossover (0.2,0.4,0.7,0.9).

However, as the crossover factor (CO) increases the GA most likely reaches optimal or near optimal solutions. The impact of crossover factor is more significant in complex problems. The lateness of the individual projects (O1) as well as portfolio (O2) decreases with the increase of the crossover factor. Beyond the value of 0.7, this decrease becomes less significant, and the probability of fit chromosomes to survive immediately from a generation to another without undergoing crossover decreases.

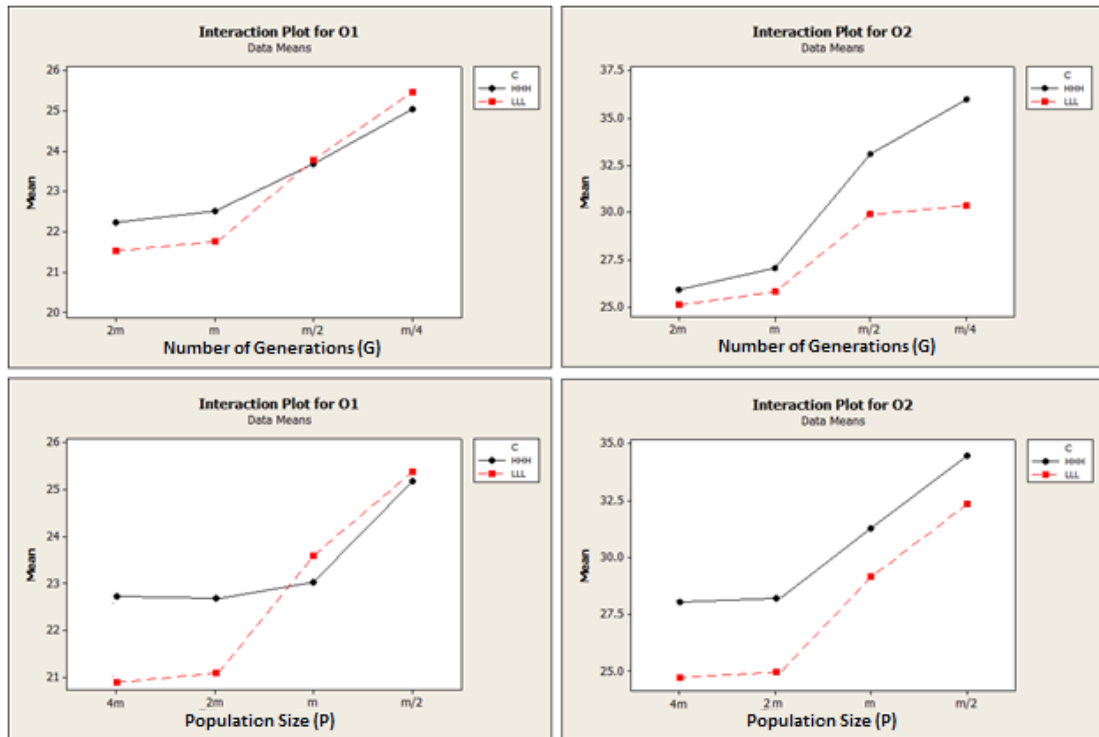


*CO: Crossover Factor, M: Mutation Factor

*GA parameters used: Number of Generations = m , Population Size = $2m$ (m =number of tasks)

Figure 16: Mutation and Crossover factors sensitivity analyses

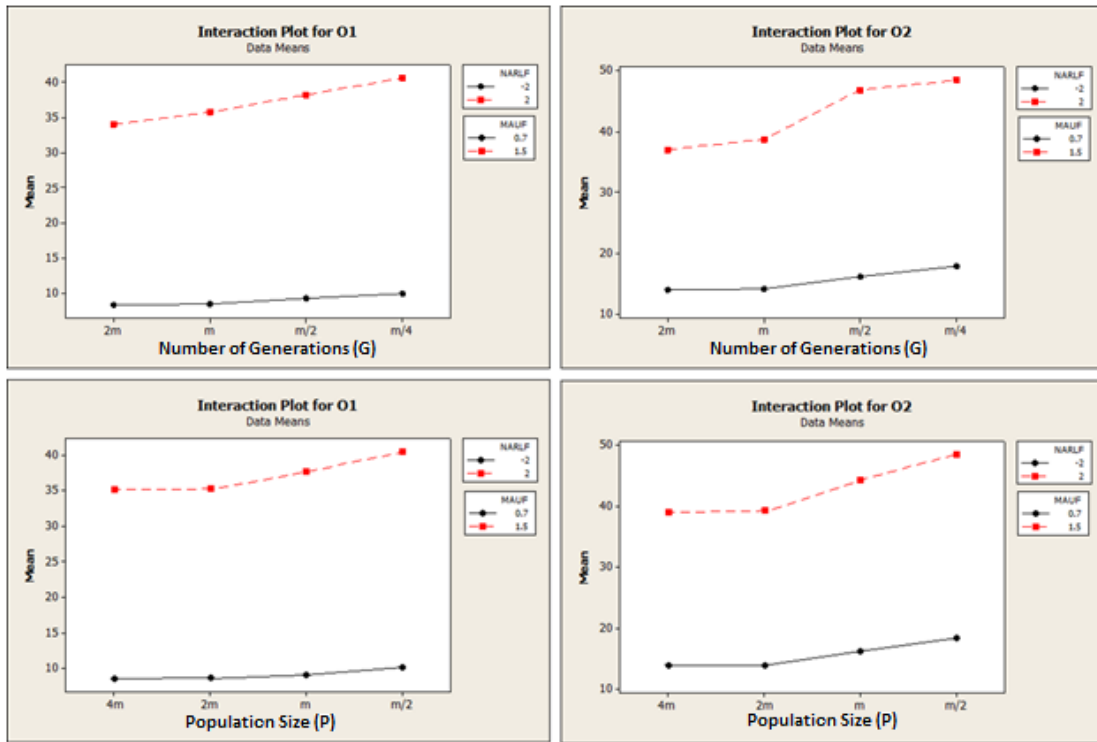
With m being the number of tasks in a project or portfolio of projects, the number of generations and population size are defined in terms of m . Figure 17 shows the interaction between both number of generations and populations size and with O1 and O2 based on the complexity of the projects. GA is applied to the data set considering different combinations of population size ($m/2, m, 2m, 4m$) and number of generations ($m/4, m/2, m, 2m$). As the number of generations increases O1 and O2 decrease, however the slope of this decrease becomes insignificant beyond a number of generations m . Similarly for the population size where the significant decrease is between m and $2m$. An interesting observation is that the impact of both GA factors, is larger in less complex projects and this is because these projects have less precedence constraints which make the possible feasible solutions larger in number.



*GA parameters used: Crossover Factor = 0.7, Mutation Factor = 0.02

Figure 17: Two-way interaction plots for number of generations (G), population size (P) and complexity (C)

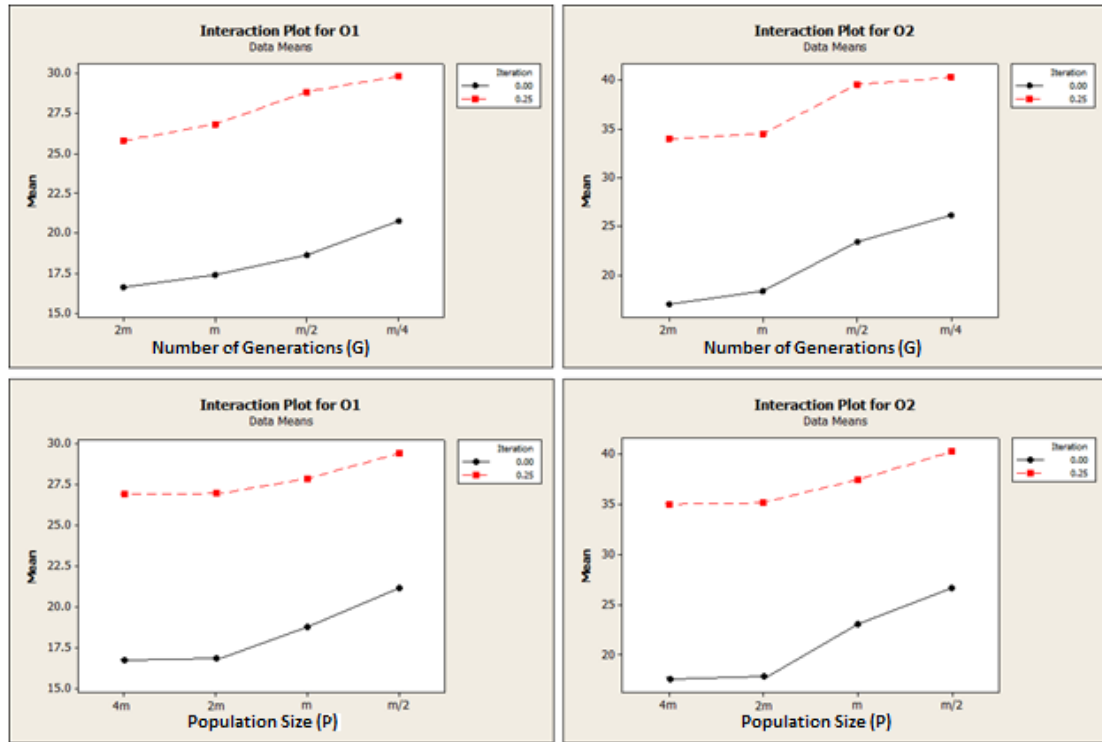
Figure 18 shows that the higher the level of resource constraints is (NARLF=2, MAUF=1.5), the more significant is the impact of both number of generations and population size on projects lateness and the larger both factors should be. For O2 this impact is somehow higher than O1 and for NRLF=-2 and MAUF=0.7 no sound enhancement is achieved by increasing those factors. The presence of resource constraints increases the number of possible networks as well as possible starting times for a task.



*GA parameters used: Crossover Factor = 0.7, Mutation Factor = 0.02

Figure 18: Two-way interaction plots for number of generations (G), population size (P) and resources constraints (NARLF, MAUF)

Figure 19 shows the interaction between both number of generations and populations size with iteration level and O1 and O2. As the level of iteration in projects increases, the delay of these projects will also increase and new iterative tasks are executed. As a result the project network becomes larger; however those iterative tasks have defined predecessors. This means that the increase of the number of tasks due to iteration will not increase the possibilities of a certain sequence of task. This is why there exist no significant or interesting impact for both number of generations and population size based on iterations. The effect of iteration is tackled by the choice of the sample size in sampling GA and in the initial population in variable length GA.



*GA parameters used: Crossover Factor = 0.7, Mutation Factor = 0.02

Figure 19: Two-way interaction plots for number of generations (G), population size (P) and iteration level

As a result, Table 3 shows the calibrated GA models used to run the test examples which are mentioned in the following sections.

Factor	GA Model	
	Sampling GA	Variable Length GA
Sample size (S)	[m - 4m]	N/A
No. of Generations (G)	[m/2 - m]	[m/2 - m]
Population size (P)	[m - 2m]	4m <i>if m is the no. of all possible tasks (exact and iterative)</i> [6m-10m] <i>(if m is the no. of exact tasks)</i>
Crossover Factor(CO)	0.7	0.7
Mutation factor (M)	0.02	0.02

Table 3: GA factors for Calibrated Models

Further analysis for the impact of using operators as mentioned in Section 3.3 is shown in Table 4 below, using the set of projects mentioned in Section 5.1.

C	Operator	Effect (Percentage change) (%)	
		O1	O2
LLL	By pass	-3.47	-5.16
	Tournament Selection (instead of Roulette selection)	-1.13	-1.05
	Two-points crossover (instead of one-point crossover)	-0.66	0.13
	Tournament Selection & Two-points crossover	-5.96	-3.29
HHH	By pass	-1.01	-3.90
	Tournament Selection (instead of Roulette selection)	-1.02	-3.23
	Two-points crossover (instead of one-point crossover)	-0.36	-7.28
	Tournament Selection & Two-points crossover	-1.51	-4.52

* GA parameters used: Crossover Factor = 0.7, Mutation Factor = 0.02,
Number of Generations = m, Population Size = 2m (m=number of tasks)

Table 4: Impact of Operators (Percentage Change)

Table 3 gives the percentage change of O1 and O2 due to the use of each operator. For example in high complexity problems (HHH), the use of two-point crossover instead of one-point decreases O2 by 7.28%.

Based on the table any of the two selections may be used in the GA models (Roulette and Tournament), as well as any of the one-point or two-point crossovers. The use of the by-pass operator somehow enhances the optimality of the results, mainly for less complex projects.

5.3. Model Validation

5.3.1. Hartmann's Benchmarks

The first test for the model was using Hartmann's benchmarks. Test problems constructed by the project generator ProGen developed by Kolisch et al. (1995) were used. These instances are available in the project scheduling problem library PSPLIB from the University of Kiel (<http://www.mpsplib.com>). In our study, we have used the ProGen problem instances with 30, 60, 90 and 120 non-dummy activities. Running our model the results were consistent with the literature showing: 0.00% deviation from optimality, 100% optimality and short time of execution which showed 19s for 30 task project, 46s for 60 tasks project, 1m 28s for 90 tasks project and 2m 14s for 120 tasks project. The mentioned results showed that our model is valid for single-project instances without rework. These examples and all others throughout the section were handled using processor Intel® Core™ i7-3770S CPU @ 3.10GHz (8 CPUs), ~3.1GHz. (Check Appendix E)

5.3.2. Browning & Eppinger's Case Study

To study the behavior and consistency of the model, we applied it to Browning and Eppinger's (2002) example, demonstrated in the DSMs in Figure 20. Different methods and factor variations were applied to this problem and the results are shown in Table 5. Cases 1 and 2 are 300 and 1000 simulations respectively. Case 3 is for the variable lengths GA, while cases 4, 5 and 6 are for the variable sample GA. Note that simulation can be obtained by using the variable sample method setting the sample size to the number of simulations we need and setting both number of generations and population size to 1.

	Activity name	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Prepare UCAV Preliminary DR&O														
2	Create UCAV Preliminary Design Architecture	0.4								0.2					
3	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings		0.5		0.4										
4	Perform Aerodynamics Analyses & Evaluation	0.3		0.5											
5	Create Initial Structural Geometry	0.4		0.5			0.1		0.1				0.3	0.1	
6	Prepare Structural Geometry & Notes for FEM	0.1				0.4									
7	Develop Structural Design Conditions	0.4					0.4								
8	Perform Weight & Inertias Analyses						0.5						0.5		
9	Perform S&C Analyses& Evaluation	0.4		0.5	0.5				0.5						
10	Develop Balanced Freebody Diagrams& External Applied Loads				0.1		0.5	0.2	0.1			0.4			
11	Establish Internal Load Distributions						0.5	0.5	0.5		0.5				
12	Evaluate Structural Strength, Stiffness, & Life	0.4					0.4	0.5			0.5	0.4			
13	Preliminary Manufacturing Planning & Analyses	0.5				0.5							0.4		
14	Prepare UCAV Proposal	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	

a. Problem DSM

	Activity name	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Prepare UCAV Preliminary DR&O														
2	Create UCAV Preliminary Design Architecture	0.5								0.1					
3	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings		0.3		0.5										
4	Perform Aerodynamics Analyses & Evaluation	0.4		0.8											
5	Create Initial Structural Geometry	0.1		0.1			0.1						0.3	0.1	
6	Prepare Structural Geometry & Notes for FEM	0.1				0.3									
7	Develop Structural Design Conditions	0.5					0.8								
8	Perform Weight & Inertias Analyses						0.5						0.5		
9	Perform S&C Analyses & Evaluation	0.3		0.3	0.3				0.3						
10	Develop Balanced Freebody Diagrams & External Applied Loads				0.1		0.5	0.4	0.3			0.3			
11	Establish Internal Load Distributions						0.5	0.5	0.3		0.3				
12	Evaluate Structural Strength, Stiffness, & Life	0.5					0.3	0.5			0.5	0.5			
13	Preliminary Manufacturing Planning & Analyses	0.9				0.9							0.3		
14	Prepare UCAV Proposal	0.5	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	

b. Rework impact DSM

Figure 20: DSM for Browning and Eppinger (2002) problem

Case number	Method	Factors	Time of execution	Makespan Mean (days)	Std Dev
<i>Case 1</i>	Simulation	N=300	1 min	134.7	5.111
<i>Case 2</i>	Simulation	N=1000	2 min	134.2	4.663
<i>Case 3</i>	Variable Lengths GA	Pop = 300 Gen = 14	5 min	134.3	4.976
<i>Case 4</i>	Variable Sample GA	Sample size=56 Pop=28 Gen=14	24 min	134.1	5.26
<i>Case 5</i>	Variable Sample GA	Sample size =140 Pop=14 Gen=7	15 min	133.5	5.108
<i>Case 6</i>	Variable Sample GA	Sample size =140 Pop=14 Gen=7	15 min	133.2	5.102

Table 5: Cases based on Browning and Eppinger (2002) problem

Figure 21 shows that the results remain almost similar if the project is ran multiple times with the same inputs, which shows stability and reliability of the resultant makespan distribution (cases 5 and 6). Comparing cases 4 and 5 shows that an increase of the sample size provides better illustration of the makespan distribution and causes more chromosomes to show up and thus new makespans. The significant increase of the number of simulations gives a better view of the distribution which tends to be nearly normal.

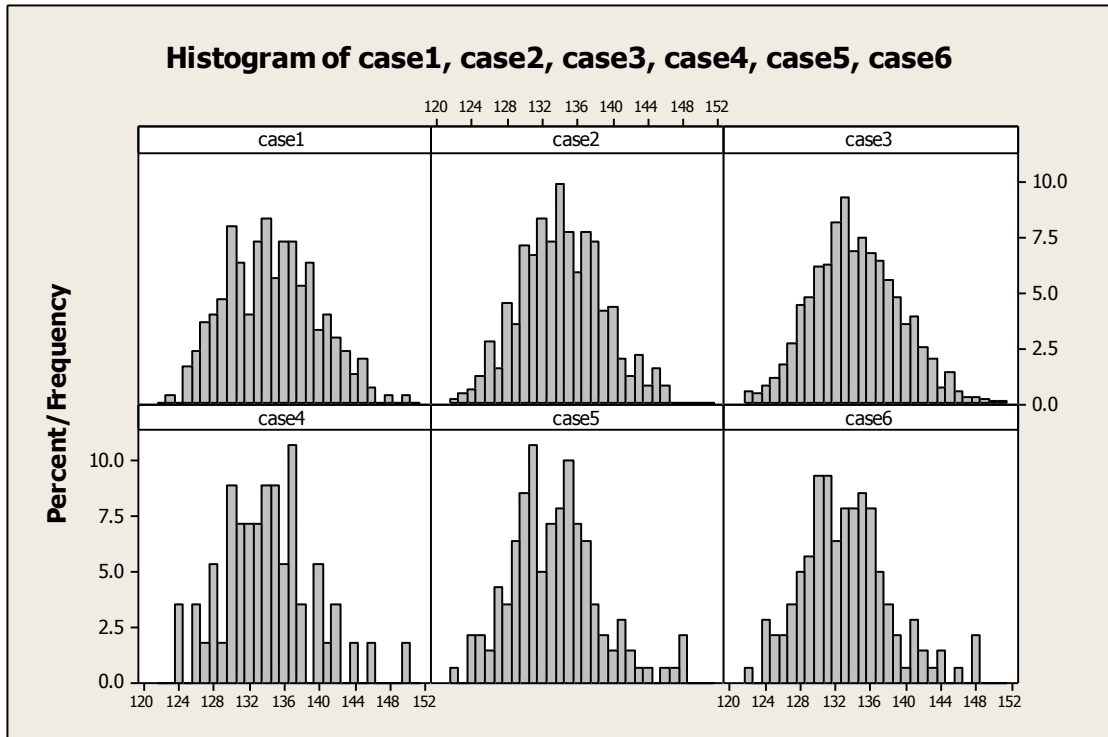


Figure 21: Histograms showing distribution of makespan

Another example, introduced in this paper and shown in Appendix D, was used to test the validation of the model’s behavior.

In order to test the different suggested operators mentioned in section 3.3, Browning & Eppinger (2002) example is considered. We then obtain four different test cases based on those operator variations, as shown in Table 6 and Figure 22. The first test is for using roulette selection and one-point crossover in the GA process and it is the main benchmark. The second case tests the variation if the tournament selection is used instead of roulette. The third case tests the impact of using two-point crossover and the fourth tests the impact of introducing the by-pass operator on the GA.

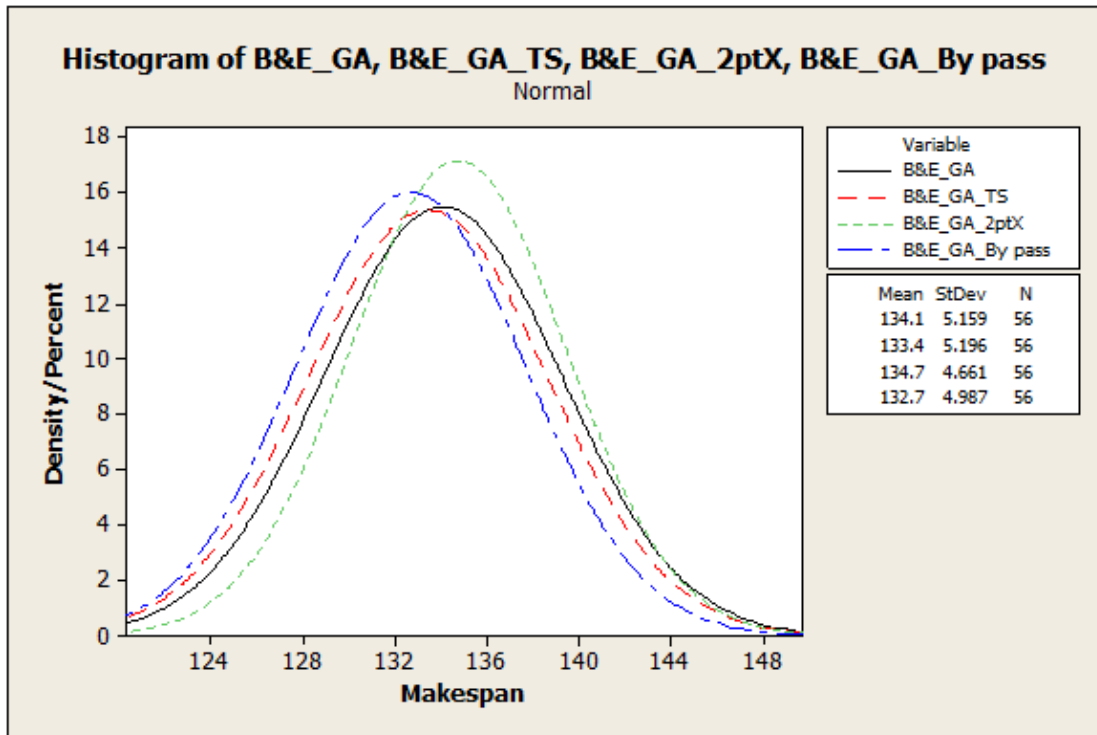


Figure 22: Makespan data distributions for operators test

Problem	Method Description	Sample Size	Pop	Gen	Time of execution	Mean	St Dev
<i>B&E_GA</i> (Roulette selection, 1pt Crossover)	Sampling GA (CO=0.7, M=0.02)	56	14	7	3m 45s	134.1	5.159
<i>B&E_GA_TS</i> (tournament selection, 1pt Crossover)	Sampling GA (CO=0.7, M=0.02)	56	14	7	4m	133.4	5.196
<i>B&E_GA_2ptX</i> (Roulette selection, 2pt Crossover)	Sampling GA (CO=0.7, M=0.02)	56	14	7	9m	134.7	4.661
<i>B&E_GA_by pass</i> (Roulette selection, 1pt Crossover, by pass)	Sampling GA (CO=0.7, M=0.02)	56	14	7	5m	132.7	4.987

Table 6: Examples Comparison

Using the tournament selection may help gain lower makespans shifting the distribution to the left, since the selection is direct for the fittest chromosome holding

the least makespan. On the other hand, using this type of selection may alter the normality of the project makespan distribution since the selection may stick to specific chromosomes only. Using the two point crossover instead of the one point crossover did not have significant effects on the results, so anyone of the two may be used.

Using the by-pass operator which means that before crossover happens the fittest chromosome is bypassed to the next generation caused the shift of the makespan distribution to the left and made sure we always obtain near optimal solutions. However, the GA will no longer be spontaneous when this by-pass operator is used.

5.4. Set up and computational results

Browning and Yassine (2013) compiled a set of 31 priority rules (PRs), some of which were from existing literature and developed specifically for the RCMPSP and others which have been successful in a single- project environment. To test the performance of GAs with respect to those PRs the set of examples, mentioned in Section 5.1, are used. Note that the solution of these problems using the PRs was taken from Browning and Yassine (2013). One of the objectives of the thesis is to measure the accuracy of these PRs by comparing them to the GA solution. Wherever the GA wins over the PR, this means it is not significant and accurate to use the PRs chosen to schedule a project with defined network characteristics.

5.4.1. Results for O1

Starting with O1, Figures 23 and 24 show a one-way analysis of means (ANOM) for all 31 PRs and the 2 GA approaches (averaging overall other factors), assuming a 95% confidence level and corresponding to the case where no iteration exist and when iterations are present respectively. These figures show that the GA's perform better than PRs when iteration exists. However, when no iteration exists, the GA's resulted in smaller overall means of delay, but there is no statistical significance that they outperformed the PRs. In fact these results may be biased by some factors when all the data is combined together. Also, we note that both proposed GA approaches performed almost identical.

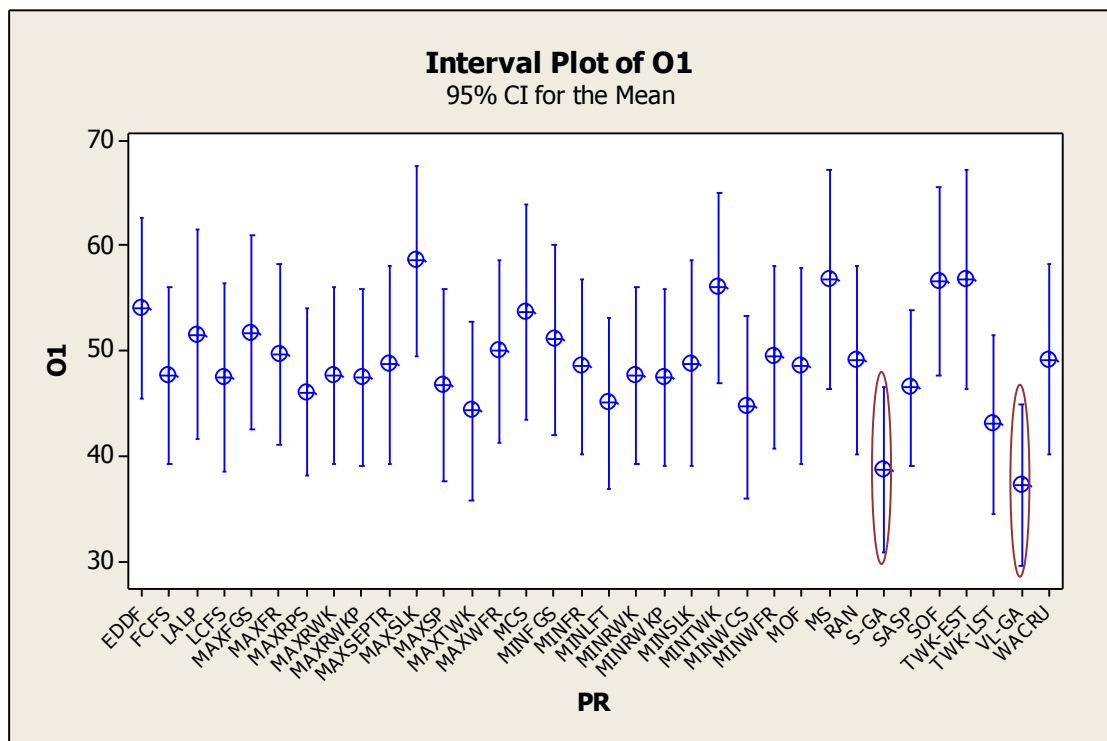


Figure 23: One-way analysis of means (ANOM) for O1 (no iteration)

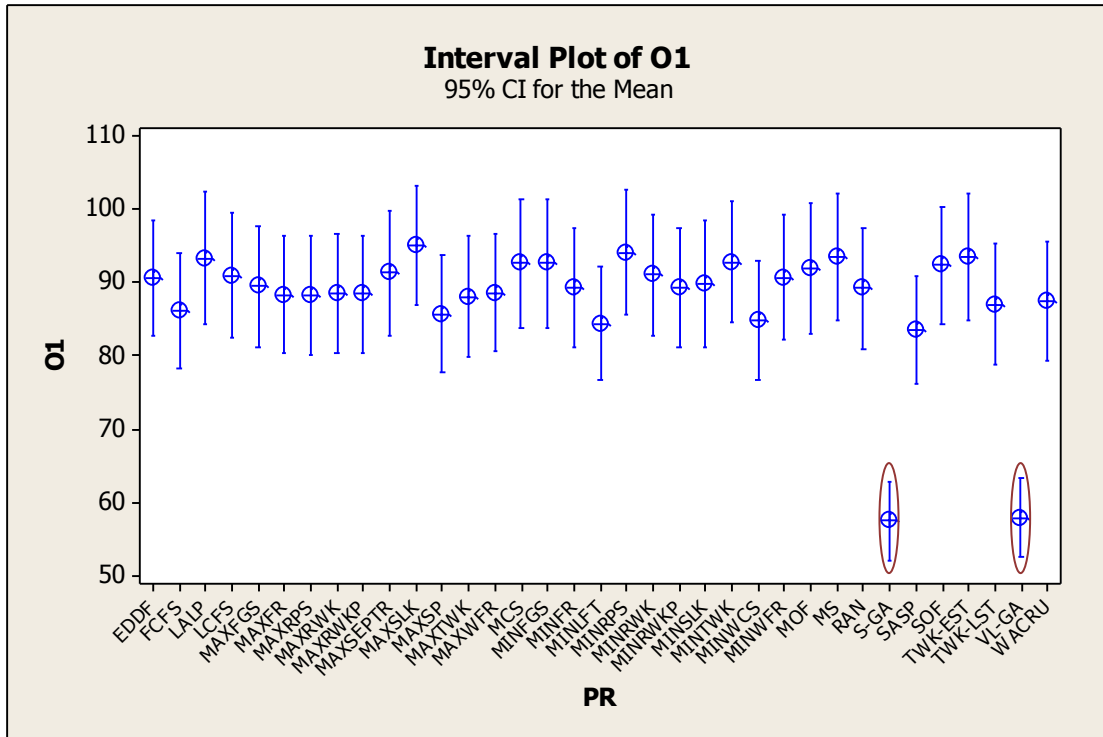


Figure 24: One-way analysis of means (ANOM) for O1 (with iteration)

To investigate the underlying reasons for the success of the GAs in minimizing O1, it is necessary to look into how the GA works. The GAs consider parallel tasks as well as all projects simultaneously. Also GAs allocate resources effectively giving the priority for tasks that need higher quantities of resources not to keep many resources available, which favors the execution of parallel tasks that share same resources. In examples where iteration exists, the iterative cycle does not stop the overall project or projects and may occur in parallel with other tasks. Moreover, our GAs take into consideration successors of an iteration cycle in a way to avoid unnecessary loops, which is applicable in the product development projects, where a task will wait for any changes before it's done. Each time a task is reworked; its rework state is incremented and considered whenever it is revisited by any other coming rework cycle. This ensures a global flow of information in the project or portfolio so that unnecessary reworks will not exist.

5.4.2. Results for O2

While O1 attends to the effects of delay on the projects individually, O2 only accounts for delays that lengthen the overall portfolio of projects. While individual project managers would care more about O1, portfolio managers would have reason to focus on O2. However, since O2 is driven by the single longest project in a problem, it is a less sensitive measure than O1.

Figures 25 and 26 show a one-way analysis of means (ANOM) for all 31PRs and the 2 GA approaches (averaging overall other factors), assuming a 95% confidence level for iteration equal to 0 and greater than 0 respectively.

The similarity between the results of O1 and O2 regarding GA performance, is due to the fact that through the GA approaches the portfolio of projects is treated as one whole project. This makes delays in the single projects highly correlated with that of the whole portfolio. The generation of sequences in GA based on both resource and precedence constraints without considering starting or finishes times of tasks and projects, also helps in the presence of those similar results.

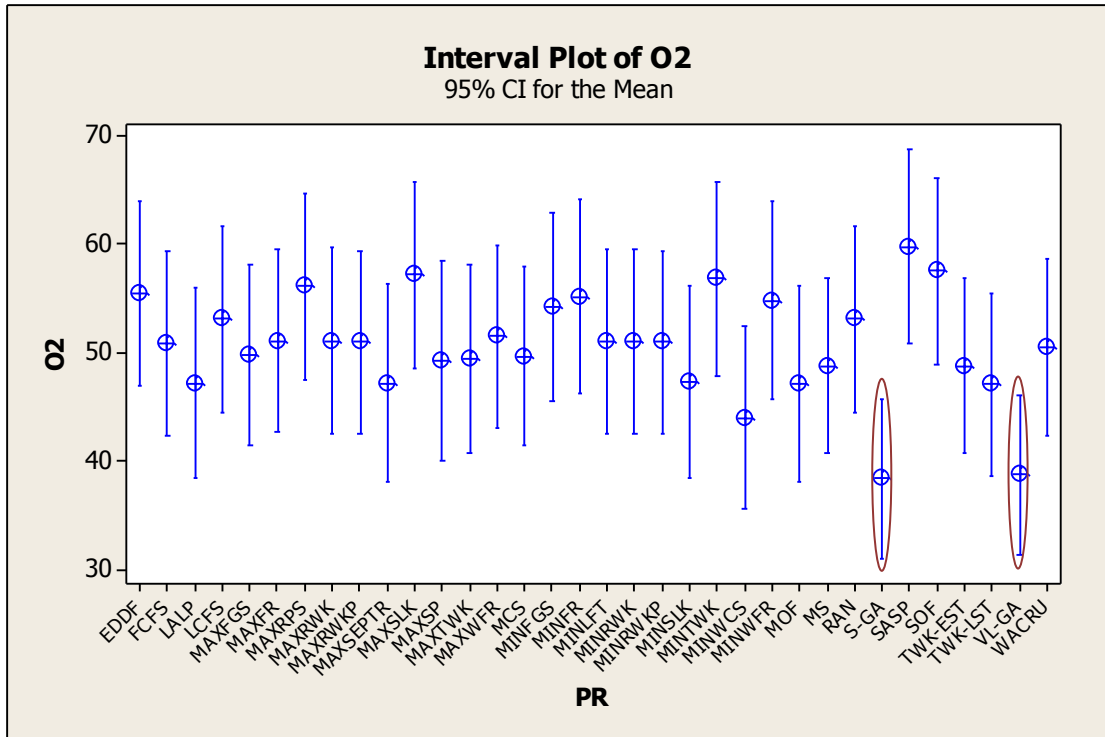


Figure 25: One-way analysis of means (ANOM) for O2 (no iteration)

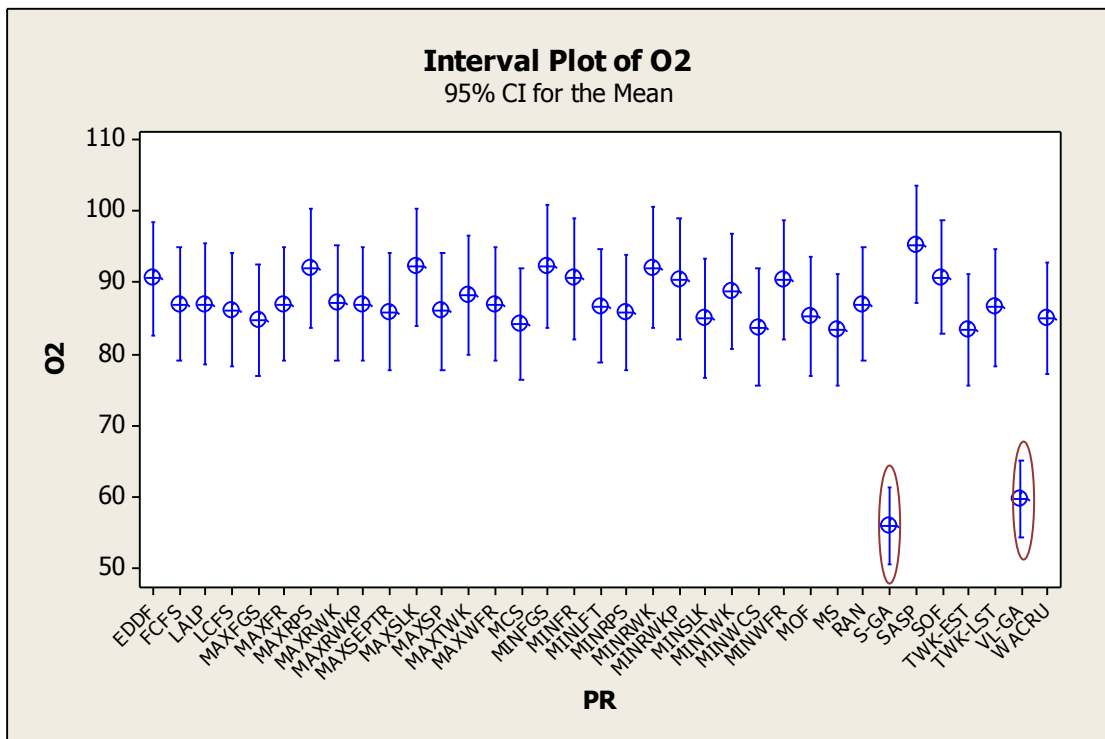


Figure 26: One-way analysis of means (ANOM) for O2 (iteration only)

For both O1 and O2, the GA showed to win over PR's in high complexity problems (HHH) for high iteration level (0.25) as well as for moderate iteration level (0.1) whenever MAUF is large enough (1.1 and above). As the iteration level increases, the GA will start winning PR's for high values of NARLF (0 and above) and MAUF (above 1.1). Also for high values of NARLF (2), MAUF(1.5), iteration (0.25) and complexity (HHH), GA shows to be a better approach than PR's.

5.4.3. Summary of results

In the previous section, we identified, confirmed, and discussed several important factors that contribute to project and portfolio delay. To distill these results for managers, we developed two decision tables (Tables 7 and 8) to aid in the selection of GAs instead of PRs for a particular situation. Our concern is to find cases when the GAs perform better than PRs. Here, we clearly see the different results for O1 and O2. Table 7 and Table 8 also show the percentages by which the GAs win over PRs.

From an individual project manager's point of view, O1 is a more appropriate objective, whereas O2 aligns more with an executive's or portfolio manager's point of view. The different results obtained by these two objectives may relate to the friction that occurs between managers at different organizational levels. We observe several patterns in these tables. First, the GA is less favored when there is no iteration, the chances of GAs winning (statistically) over PRs increases with greater iteration level, MAUF, NARLF and C in both tables.

To benefit from our results and recommendations as summarized in Tables 7 and 8, managers must be able to characterize their projects in terms of complexity (C), amount of resource contention (MAUF), and resource distribution (NARLF). First, regarding C, managers can qualitatively estimate whether they are dealing with a high-C situation or a low-C one without having to precisely obtain a numerical estimate. The level of complexity may be determined based on the dominance of sequential or parallel activities, as well as the number of dependencies in a project (precedence constraints). In addition, the distribution of resources and the qualitative level of resource contention can be ascertained without too much effort. However, managers need to define the actual network activities, in terms of precedence relationships and tasks characteristics (DSM) as well as to well assess the type and quantity of available resources for their portfolios. Having all these data in hand, the GA will not only give managers optimal or near optimal job sequence, but also provide them with a statistical distribution and overview for the portfolio possible sequences to help them account for possible lateness or any other inconvenience due to rework, resources constraints and complex precedence feasibilities or any other inconvenience due to rework, resources constraints and complex networks. Hence, these results are readily applicable to practical issues facing project and portfolio managers.

ITERATION=0	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	MINWCS MAXSP MINSLK MOF	MAXSEPTR MAXSP MOF MINWCS MINSLK	MINWCS MAXSP TWK-LST MINSLK	MAXSP MINWCS MINSLK MOF	LALP MOF MAXSEPTR TWK-LST	MINWCS MAXSP MINSLK TWK-LST	MINWCS MINSLK MAXSP MOF	MINWCS	MINWCS MAXSP MINSLK TWK-LST
MAUF=1.1	TWK-LST MAXTWK	TWK-LST MAXTWK MAXRPS	SASP TWK-LST MINWCS EDDF	TWK-LST MAXTWK	RAN MINLFT	TWK-LST SASP EDDF MINLFT	TWK-LST	MAXSP	TWK-LST EDDF MINLFT SASP
MAUF=1.5	TWK-LST MAXTWK FCFS TWK-EST	TWKLST	SASP MINLFT	TWK-LST	SASP LCFS	SASP TWK-LST	S-GA: 41.27% VL-GA: 34.92%	MAXRPS SASP	SASP LCFS

ITERATION=0.1	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	MINWCS MAXSP	MAXSP MINSLK MINWCS MAXSEPTR	SASP MINRWKP MINFR	MINWCS MAXSP	MINSLK MAXSP LALP MAXSEPTR	MINWCS MINSLK MAXSP MOF	S-GA: 52% VL-GA: 40%	MINSLK MAXSP MINWCS MAXSEPTR	MINWCS MAXSP MOF
MAUF=1.1	TWK-LST SASP	STWK-LST	SASP MINWCS	TWK-LST	MINLFT RAN SASP	SASP MINLFT MINFGS MINWCS	S-GA: 70% VL-GA: 70%	MAXSP MINWCS SASP MINLFT	SASP TWK-LST MINWCS
MAUF=1.5	SASP	MINLFT TWK-LST	SASP MINRWKP MINFR MAXPRS	VL-GA: 45.55% S-GA: 38.89%	SASP WACRU	SASP TWK-LST MAXTWK MINWCS	S-GA: 42.55 % VL-GA: 39.15 %	S-GA: 48.23% VL-GA: 41.18 %	SASP MAXRPS

ITERATION=0.25	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	S-GA: 70% VL-GA: 65%	MAXSP MINWCS	MINWCS MINLSK MINLFT	S-GA: 63.63% VL-GA: 63.63%	MAXSP TWK-EST TWK-LST MAXSEPTR	MINWCS MAXSP TW-LST	VL-GA: 63.48% S-GA: 60.86%	MINWCS MAXSP MINLFT TWK-EST	MINWCS MAXSP MOF
MAUF=1.1	S-GA: 61.53% VL-GA: 54%	MINWCS SASP	SASP MAXRPS MINWCS	VL-GA: 65.33% S-GA: 63.33%	SASP MINLFT	SASP MINLFT MAXWFR MAXSP	S-GA: 49.64% VL-GA: 47.48%	S-GA: 43.53% VL-GA: 43.53%	S-GA: 53.22% VL-GA: 50%
MAUF=1.5	S-GA: 37.14% VL-GA: 37.14%	SASP MINLFT	SASP	VL-GA: 66.11% S-GA: 62.22%	S-GA: 50% VL-GA: 45%	SASP	VL-GA: 70.86% S-GA: 66.28%	S-GA: 61.54% VL-GA: 57.69%	S-GA: 49.56% VL-GA: 46.02%

Table 7: Results for O1

Multiple entries in each cell are listed in order of increasing means (i.e., the best PR is listed first)

ITERATION=0	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	LALP MINSLK MINWCS MAXSP	MAXSEPTR MOF	LALP MS MCS MINSLK	LALP MINSLK MINWCS MOF	MOF LALP MAXSEPTR MAXSP	LALP MINWCS MINSLK MAXSP	LALP MOF MINSLK MAXSP	MOF MATWK	LALP MINWCS MINSLK MAXSP
MAUF=1.1	MINWCS LALP MOF MINSLK	MINWCS MINSLK	MS MINWCS MCS	MINWCS LALP MINSLK MOF	MINWCS TWK-LST TWK-EST	MINWCS MS MCS	MINWCS MOF LALP	MOF TWK-LST	MINWCS MS MCS
MAUF=1.5	MINWCS LALP MOF MINSLK	MAXTWK MINWCS	MS MCS MINWCS	MINWCS	MOF	MS MCS MINWCS	S-GA: 40% VL-GA: 33.33%	S-GA: 35.06% VL-GA: 32.47%	MINWCS MS MCS EDDF

ITERATION=0.1	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	MINSLK MAXTWK	MOF MINSLK	TWK-EST MS	MINWCS MINSLK	MINSLK MAXTWK LALP MOF	LALP MCS MINRPS MINWCS	MINSLK MINWCS	LALP MOF	MINWCS MAXSP LALP MINSLK
MAUF=1.1	S-GA: 33.33% VL-GA: 33.33%	MINWCS	TWK-EST MS MOF MAXSEPTR	S-GA: 36% VL-GA: 30%	MINWCS MINFGS MAXWFR	MCS TWK- EST MS MINWCS	S-GA: 20% VL-GA: 18%	TWK-LST MAXTWK LALP MOF	TWK- EST MINWCS MS MINRPS
MAUF=1.5	VL-GA: 46.32% S-GA: 42.1%	MOF MINSLK MAXFGS	TWK-EST MS	S-GA: 39% VL-GA: 38%	MOF	TWK- EST MCS MS WACRU	S-GA: 25.26% VL-GA: 23.16%	S-GA: 35.06% VL-GA: 32.47%	MS TWK- EST

ITERATION=0.25	NARLF=-2			NARLF=0			NARLF=2		
C:	HHH	HLL	LLL	HHH	HLL	LLL	HHH	HLL	LLL
MAUF=0.7	VL-GA: 65% S-GA: 60%	MAXSP MAXRWK	LALP MCS	S-GA: 67.29% VL-GA: 62.62%	WACRU MAXRWKP MAXRW FCFS	LALP LCFS MINTWK TWK- EST	S-GA: 68% VL-GA: 60%	MAXSP RAN MOF MINSLK	MINSLK MINWCS
MAUF=1.1	S-GA: 64% VL-GA: 56%	WACRU TWK-EST	TWK-EST MS MCS MOF	VL-GA: 43.08% S-GA: 41.54%	WACRU MAXRWKP MAXFR FCFS	MAXRPS WACRU LCFS	S-GA: 57.14% VL-GA: 50%	MCS	S-GA: 72.31% VL-GA: 67.69%
MAUF=1.5	S-GA: 55.88% VL-GA: 41.18%	WACRU	MCS	VL-GA: 45.71% S-GA: 44.28%	S-GA: 70.37% VL-GA: 66.67%	WACRU	S-GA: 77.78% VL-GA: 69.44%	S-GA: 59.26% VL-GA: 55.56%	S-GA: 52% VL-GA: 44%

Table 8: Results for O2

Multiple entries in each cell are listed in order of increasing means (i.e., the best PR is listed first)

CHAPTER VI

SUMMARY AND CONCLUSION

Project as well as portfolio (multi-project) management is becoming ever-more crucial for the survival of organizations. Relying on their research and development projects, organizations are trying to stay competent and deliver new products and services to assure their success. This competency is based on managing the problems of cost and schedule overrun on projects and finding optimal schedule that minimizes project cost, variation of resource profiles, and project duration. Decisions about which activities to do when (based on resource allocations, precedence network details and iterative activities integration) have a tremendous effect on project completion times. Yet, many project managers, who often do not have an activity network model to which they might apply more advanced techniques, make resource allocation and project makespan estimation decisions based on their own experience or some “rules of thumb” such as some priority rules.

Considering iterative RCPSP (or RCMPSP) with, this thesis uses relatively new measures and network generators along with two genetic algorithm approaches (Sampling GA and Variable Length GA), making a number of observations and comparing those to the most popular PRs. GA settings’ influence on its efficacy and performance is explored in relation to iterative RCPSP and RCMPSP.

For the study, we generated project portfolios (each consisting of three projects) according to a full factorial experiment that included four factors at various levels. Our analysis provides much-needed guidance on the use of certain GAs in varied project situations and objectives.

Accordingly, a decision table is introduced to guide managers in choosing between best PRs and GA based on MAUF, NARLF, C and iteration level. These results show how different objectives for individual project managers and portfolio managers can lead to the preference for different decision rules and thus organizational tensions. GA is less favored when there is no iteration, while the chances of GAs winning (statistically) over PRs increases with higher iteration level, MAUF, NARLF and C considering both O1 and O2.

While several studies regarding project scheduling, PRs, GAs and related topics exist, it is in some ways astounding that no firmer guidance has appeared for decision makers in an iterative project or multi-projects with limited resources, the most realistic situation in contemporary practice. Thus, explaining the conditions under which using GAs performs well (or poorly) compared to certain PRs is an important contribution that allows managers to sift through the conflicting results in the literature. However, these results could be immediately applicable in practical situations as long as managers are able to characterize their projects and network details.

Future research could expand our study to combine both GAs and PRs mainly in determining a project or portfolio makespan, explore other RCMPSP formulations (such as with preemption, stochastic activity durations, or dynamic project arrivals) in a similarly comprehensive study, or introduce new GA operators that may increase the performance of the GA approach. The results reported can also be used for the development of improved GAs that may perform better than PRs in cases where they failed to in this study.

REFERENCES

- Agrawal, M.K., Elmaghraby, S.E. & Herroelen, W.S. (1996). DAGEN: A Generator of Test Sets for Project Activity Nets. *European Journal of Operational Research*, 90:376-382
- Alarcon, L. F. & Ashley, D. B. (1992). *Project Performance Modeling: A Methodology for Evaluating Project Execution Strategies*. Source Document 80, The Construction Industry Institute, University of Texas at Austin.
- Alcaraz, J & Maroto, C. (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operation Research*, 102: 83-109
- Anderson, S. D. & Tucker, R. L. (1994). Improving Project Management of Design. *J. Manage. Eng.*, 10(4), 35-44.
- Balaji, C. (2011). Lecture 38: *Design and Optimization of Energy Systems* [Youtube video]. Department of Mechanical Engineering, IIT Madras. Retrieved from http://www.youtube.com/watch?v=Z_8MpZeMdD4
- Bien, W.W., Kamburowski J. & Stallmann, M.F.M. (1992). Optimal Reduction of Two-Terminal Directed Acyclic Graphs. *SIAM J. Comput.*, 21:1112-1129
- Bouleimen, K & Lecocq, H. (2003). A New Efficient Simulated Annealing Algorithm for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, 149: 268-281
- Bramble, B. B. & Cipollini, M. D. (1995). *Synthesis of Highway Practice 214: Resolution of Disputes to Avoid Construction Claims*. Radnor, Pennsylvania: Day and Zimmermann International, Inc.
- Browning, T., Yassine, A., (2013). Managing a Portfolio of Product Development Projects under Resource Constraints," *Submitted*. Dec. 2013.
- Browning, T.R. & Yassine, A.A. (2010a). A Random Generator of Resource-Constrained Multi-Project network Problems. *Journal of Scheduling*: 13(2): 143-161.
- Browning, T.R. & Yassine, A.A. (2010b). Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited. *International Journal of Production Economics*, 126(2): 212-228
- Browning, T.R. & Eppinger, S.D. (2002). Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development. *Engineering Management*, 49(4)
- Chakraborty, R.C. (2010) *Fundamentals of Genetic Algorithms* [PowerPoint slides]: AI Course Lecture notes, slides 39 – 40. Retrieved from www.myreaders.info/09_Genetic Algorithms

- Chakraborty, A., Kumar Mitra, S.K. & Naskar, M.K. (2011). A Genetic Algorithm inspired Routing Protocol for Wireless Sensor Network. *International Journal of Computational Intelligence Theory and Practice*, 6(1)
- Chalabi, A. F., Salazar, G. F., & Beaudin, B. J. (1986). *Defining and Evaluating Input Variables Impacting Design Effectiveness. Research Phase I. Report.* The Construction Industry Institute, University of Texas
- Cho, S. & Eppinger, S.D. (2005). A Simulation-Based Process Model for Managing Complex Design Projects. *IEEE Transactions on Engineering Management*: 52(3)
- Colak, S., Agarwal, A. & Erenguc, S. (2013). Multi-Mode Resource-Constrained Project-Scheduling Problem with Renewable Resources: New Solution Approaches. *Journal of Business & Economics Research*, 11(11)
- Cooper, K.G., Lyneis, J.M. & Bryant, B.J. (2002). Learning To Learn, From Past To Future. *International Journal of Project Management*, 20(3)
- Davis, L. (1985). *Job-Shop Scheduling With Genetic Algorithms.* Hillsdale: L. Erlbaum Associates Inc.
- Demeulemeester, E. & Herroelen, W. (1992). A Branch-and-Bound Procedure for The Resource Constrained Project Scheduling Problem. *European Journal of Operational Research*: 111(1)
- Deemeulmeester, E., Doden, B. & Herroelen, W. (1993). A Random Activity Network Generator. *Operations Research*, 41(5), pp.972-980
- Demeulemeester, E. & Herroelen, W. (1996). An Efficient Optimal Solution Procedure for Resource Constrained Project Scheduling Problem. *European Journal of Operational Research*: 90 (2)
- Deemeulmeester, E., Vanhoucke, M. & Herroelen, W. (2003). RanGen: A Random Network Generator for Activity-On-The-Node Networks. *Journal of Scheduling*, 6: 17-38
- Eppinger, S.D., Whitney, D.E., Smith, R.P. & Gebala, D.A. (1994). A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design*, 6(1)
- Fang, H.L., Ross, P. & Corne, D. (1993). A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems. *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest (Ed.), San Mateo, Morgan Kaufmann (pp. 375-382)
- Fatemi-Ghomi, S.M.T & Ashjari, B. (2002). A Simulation Model for Multi-Project Resource Allocation. *International Journal of Project Management*: 20(2)
- Franco, E.G., Zurita, F.L.T. & Delgadillo, G.M. (2007). A Genetic Algorithm for The Resource Constrained Project Scheduling Problem (RCPSP). *The 19th International Conference on Production Research, ICPR*, Bogota, Colombia

- Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman & Co.
- Garey, M.R. & Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0-7167-1045-5
- Goldberg, D. E. & Deb, K. (1991). A Comparative Analysis of Selection Schemes Used In Genetic Algorithms. In G. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 69-93), The University of Alabama, The Clearinghouse for Genetic Algorithms
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley
- Gonçalves, J.F., De Magalhaes Mendes, J.J. & Resende, M.G.C. (2004). A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem, AT&T Labs Technical Report TD-668LM4
- Gutiérrez, M., Durán, A., Alegre, D. & Sastrón, F. (2004). HierGen: A Computer Tool for the Generation of Activity-on-the-Node Hierarchical Project Networks. *Computer Science*, 3045: 857-866
- Hartmann, S. (1998). A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics*: 45(7)
- Hartmann, S. (2002). A Self-Adapting Genetic Algorithm for Project Scheduling Under Resource Constraints. *Naval Research Logistics* 49(5), 433-448
- Herroelen, W.S. (2005). Project Scheduling—Theory and Practice. *Production and Operations Management*, 14(4), 413–432
- Ip, W. H., Li, Y., Man, K. F. & Tang, K. S. (2000). Multi Product Planning and Scheduling Using Genetic Algorithm Approach. *Computer & Industrial Engineering*: 38(2)
- Kolisch R. & Hartmann S. (1998). Heuristic Algorithms for Solving The Resource Constrained Project Scheduling Problem: Classification and Computational Analysis. In J. Węglarz (Ed.), *Project Scheduling* (pp. 147-178), Springer US
- Kolisch, R., Sprecher, A. & Drexel, A., 1995. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Management Science*, 41 (10), 1693–1703.
- Kolisch, R., Sprecher, A., 1996. PSPLIB – A project scheduling problem library. *European Journal of Operational Research* 96, 205–216.
- Kolisch, R. & Padman, R. (1997). An Integrated Survey of Project Scheduling. *Omega*: 29(3)
- Kolisch R. & S. Hartmann (1998). Heuristic Algorithms for Solving the Resource Constrained Project Scheduling Problem: Classification and Computational

- Analysis. *International Series in Operations Research and Management Science*, 14: 147-178
- Kurtulus, I. & Davis, E.W. (1982). Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *Management Science*: 28(2)
- Lam, F.S.C., Lin, B.C., Sriskandarajah, C. & Yan, H. (1999). Scheduling To Minimize Product Design Time Using a Genetic Algorithm. *International Journal of Production Research*: 37(6)
- Lancaster, J. (2007). Evolutionary Algorithms Applied to Project Scheduling Problems—A Survey of the State-Of-The-Art. *International Journal of Production Research*, 45(2)
- Lenstra, J.K. & Kan, A.H.G.R. (1978). Complexity of Scheduling Under Precedence Constraints. *Operations Research*, 26 (1), 22–35.
- Leu, S. & Yang, C. (1999). A GA-Based Multicriteria Optimal Model for Construction Scheduling. *Journal of Construction Engineering and Management*: 125(6)
- Morris, P. & Hough, G. (1987). *The Anatomy of Major Projects*. New York, Wiley
- Oliver, I. M., Smith, D. J. & Holland, J. R. C. (1987). A Study of Permutation Crossover Operators on the Traveling Salesman Problem. *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and Their Application* (pp. 224-230), Mahwah, NJ, USA: L. E. Associates, Inc.. ISBN: 0-8058-0158-8
- Payne, J.H. (1995). Management of Multiple Simultaneous Projects: A State-of-The-Art Review. *International Journal of Project Management*, 13(3), 163–168.
- Pongcharoen, P., Hicks, C. & Braiden, P. M. (2004). The Development of Genetic Algorithms for the Capacity Scheduling of Complex Products, with Multiple Levels of Product Structure. *European Journal of Operational Research*: 152(1)
- Poon, P. W. & Carter, J. N. (1995). Genetic Algorithms Crossover Operators for Ordering Applications. *Computers and Operation Research*: 22(1)
- Repenning, N.P., Goncalves, P. & Black, L.J. (2001). Past the Tipping Point: The Persistence of Firefighting in Product Development, *California Management Review*, 43(4): 44-63
- Roberts, E.B. (1992). Strategic Management of Technology: Global Benchmarking. (*Results of a survey sponsored by the Massachusetts Institute of Technology, Cambridge, MA and PA Consulting Group, London*).
- Massachusetts Institute of Technology. (1992). Strategic Management of Technology: Global Benchmarking. London, MA and PA Consulting Group
- Sanvido, V., Grobler, F., Parfitt, K., and Guvenis, M. (1992). Critical Success Factors for Construction Projects. *J. Constr. Eng. Manage*, 118(1)
- Schwindt, C. (1995). A New Problem Generator for Different Resource-Constrained

Project Scheduling Problems with Minimal and Maximal Time Lags. *WIOR-Report-449*, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Spinner, M. (1989). *Improving Project Management Skills and Techniques*. NJ: Prentice Hall

Sprecher, A. (1996). *Solving the RCPSP Efficiently at Modest Memory Requirements*. Germany: Inst. für Betriebswirtschaftslehre

The World Bank OED (1992) 1992 Evaluation Results.

Tucker, R. L. & Scarlett, B. R. (1986). *Evaluation of Design Effectiveness*. The Construction Industry Institute, University of Texas, Austin, Tex

Valls, V., Ballestín, F. & Quintanilla, S. (2007). A Hybrid Genetic Algorithm for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, 185, pp 50 – 62

Vanhoucke, M., Deemeulmeester, E. & Herroelen, W. (2000). A New Random Network Generator for Activity-On-The-Node Networks. *Proceedings of the Seventh International Workshop on Project Management and Scheduling*, Osnabrück, Germany

Wall, M.B (1996). A Genetic Algorithm for Resource-Constrained Scheduling. *Massachusetts Institute of Technology*.

Yassine, A. (2004). An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix Method, Working Paper

Yassine, A. & Braha, D. (2003). Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method, *Concurrent Engineering Research & Applications*, 11(3): 165-176

Yassine, A., Whitney, D.E. & Zambito, T. (2001). Assessment of Rework Probabilities for Simulating Product Development Processes Using the Design Structure Matrix (DSM). *Proceedings of the DETC 01: ASME 2001 International Design Technical Engineering Conferences*, Pittsburgh PA

Yassine, A.A., Chidiac, R.H. & Osman, I.H. (2013). Simultaneous optimization of products, processes, and people in development projects. *Journal of Engineering Design*: 24(4)

APPENDIX A: MODEL EXCEL INPUTS

Project “A” DSM’s:

A	Activity Name	1	2	3	4	5	6	7	8	9	10
1	Design PS Test	4			0.5						
2	Perform PS Test	1	10								
3	Evaluate PS Test		1	4							
4	Determine Success of Test			0.7	2						
5	Design Seasoning Test					4			0.6		
6	Perform Seasoning Test					1	10				
7	Evaluate Seasoning Test						1	4			
8	Determine Success of Seas. Test							0.5	2		
9	Verify paper replenishment rate meets project goals							1	0.9	2	
10	Report project plan to Business Manager to support launch				1				1	1	1

Rework Impact

A	Activity Name	1	2	3	4	5	6	7	8	9	10
1	Design PS Test	4			0.7						
2	Perform PS Test	0.2	10								
3	Evaluate PS Test		1	4							
4	Determine Success of Test			1	2						
5	Design Seasoning Test					4			0.7		
6	Perform Seasoning Test					0.5	10				
7	Evaluate Seasoning Test						0.2	4			
8	Determine Success of Seas. Test							1	2		
9	Verify paper replenishment rate meets project goals							1	1	2	
10	Report project plan to Business Manager to support launch				1				1	1	1

Resources:

Resources Available	Quantity
Al	1
John	1
Jean	1
Joy	1
Sheridan	1
Mickey	1
Miroff	1
Lanny	1
Fritz	1
Tester	2
Computer	3
Machine	1

Tasks Definition:

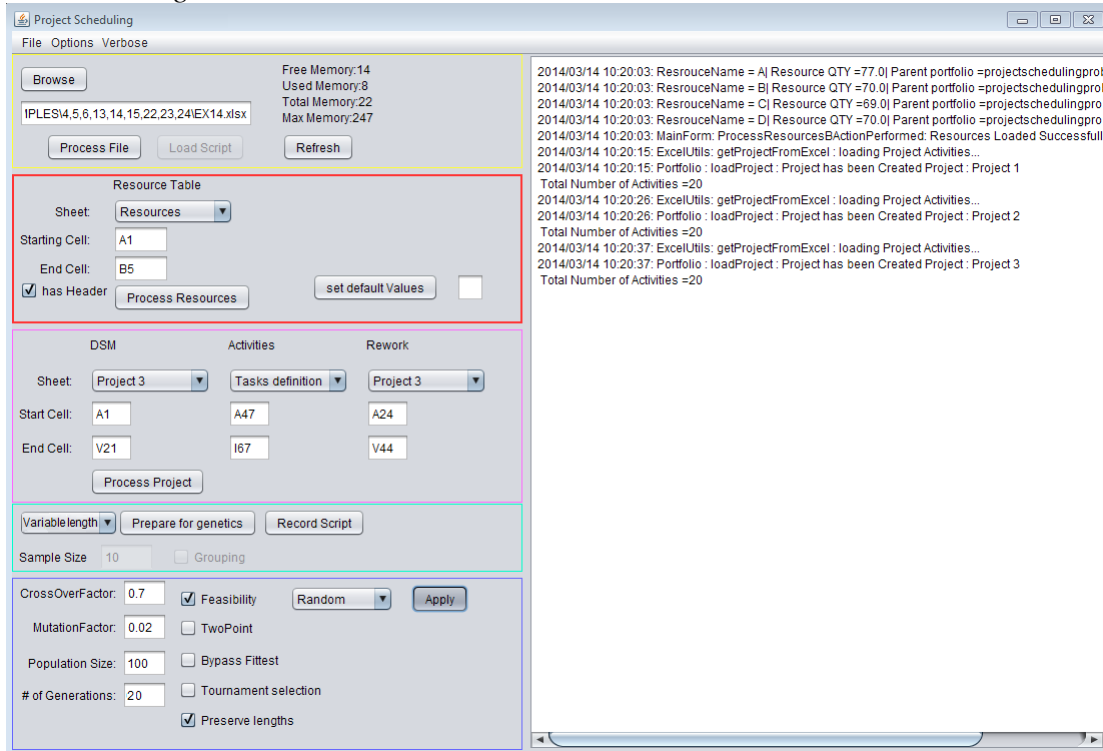
A	Activity Name	Time (minimum)	Time (most Likely)	Time (maximum)	Resources	Resource Quantity	Maximum Number of Reworks	Learning Factor
1	Design PS Test	2	4	6	John,Computer	1,1	2	0.2
2	Perform PS Test	6	10	12	AI,Computer	1,3	3	0.8
3	Evaluate PS Test	3	4	5	John	1	2	0.9
4	Determine Success of Test	1	2	3	John	1	1	0.6
5	Design Seasoning Test	2	4	5	John,Computer	1,2	5	0.45
6	Perform Seasoning Test	7	10	13	AI,Computer	1,3	2	0.015
7	Evaluate Seasoning Test	2	4	6	John	1	3	0.78
8	Determine Success of Seas. Test	1	2	4	John	1	4	0.5
9	Verify paper replenishment rate meets project goals	1	2	3	John	1	1	0.4
10	Report project plan to Business Manager to support launch	1	1	2	John	1	0	0.3

APPENDIX B: MODEL EXCEL OUTPUTS

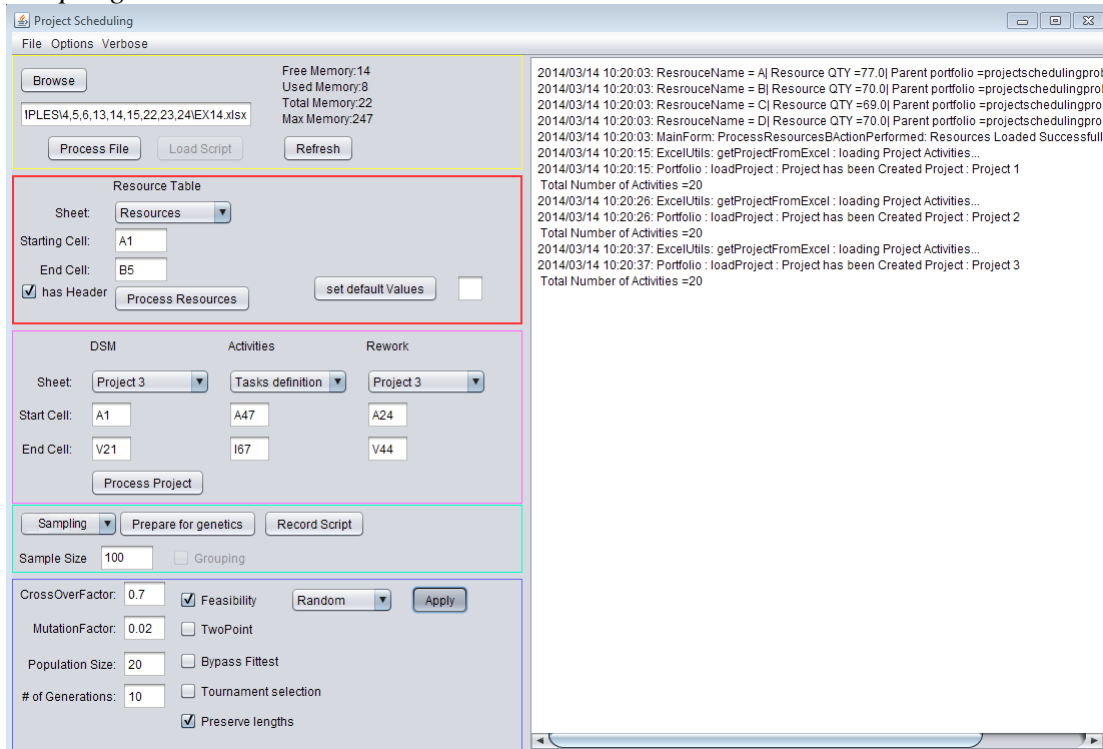
Project ID	Gen ID	Chromosome	Duration	Fitness	Project Running Stats	A	O1	O2
GPID-1	5	A1.0;A5.0;A2.0;A6.0;A3.0;A7.0;A4.0;A8.0;A1.0-f;A9.0;A2.0-f;A3.0-f;A4.0-f;A1.0-ff;A2.0-ff;A3.0-ff;A10.0;	41.947	0.024	Chromosome ExcutionStatsPname A ST 0.0 ET 41.94729 Duration 41.94729 CriticalpathDur=23.7087	18.239	0.435	0.435
GPID-2	5	A5.0;A1.0;A2.0;A6.0;A3.0;A7.0;A4.0;A8.0;A5.0-f;A6.0-f;A7.0-f;A8.0-f;A5.0-ff;A6.0-ff;A7.0-ff;A9.0;A10.0;	49.352	0.020	Chromosome ExcutionStatsPname A ST 0.0 ET 49.35191 Duration 49.35191 CriticalpathDur=24.20651	25.145	0.510	0.510
GPID-3	5	A1.0;A5.0;A6.0;A7.0;A2.0;A8.0;A3.0;A5.0-f;A4.0;A6.0-f;A7.0-f;A9.0;A10.0;	39.066	0.026	Chromosome ExcutionStatsPname A ST 0.0 ET 39.06616 Duration 39.06616 CriticalpathDur=22.9063	16.160	0.414	0.414
GPID-4	5	A1.0;A5.0;A6.0;A2.0;A3.0;A7.0;A8.0;A4.0;A5.0-f;A6.0-f;A7.0-f;A9.0;A10.0;	41.144	0.024	Chromosome ExcutionStatsPname A ST 0.0 ET 41.14374 Duration 41.14374 CriticalpathDur=20.83988	20.304	0.493	0.493
GPID-5	5	A5.0;A1.0;A6.0;A2.0;A7.0;A3.0;A4.0;A1.0-f;A8.0;A2.0-f;A5.0-f;A3.0-f;A6.0-f;A4.0-f;A7.0-f;A1.0-ff;A2.0-ff;A8.0-f;A3.0-ff;A5.0-ff;A6.0-ff;A7.0-ff;A8.0-ff;A5.0-fff;A9.0;A10.0;	53.563	0.019	Chromosome ExcutionStatsPname A ST 0.0 ET 53.56283 Duration 53.56283 CriticalpathDur=24.82876	28.734	0.536	0.536
GPID-6	5	A1.0;A5.0;A6.0;A2.0;A3.0;A4.0;A7.0;A8.0;A9.0;A10.0;	34.721	0.029	Chromosome ExcutionStatsPname A ST 0.0 ET 34.7207 Duration 34.7207 CriticalpathDur=22.82323	11.897	0.343	0.343
GPID-7	5	A1.0;A5.0;A6.0;A2.0;A7.0;A8.0;A3.0;A5.0-f;A4.0;A6.0-f;A7.0-f;A9.0;A10.0;	37.114	0.027	Chromosome ExcutionStatsPname A ST 0.0 ET 37.11417 Duration 37.11417 CriticalpathDur=22.06956	15.045	0.405	0.405
GPID-8	5	A1.0;A5.0;A2.0;A3.0;A4.0;A1.0-f;A2.0-f;A3.0-f;A6.0;A7.0;A8.0;A5.0-f;A6.0-f;A7.0-f;A9.0;A10.0;	48.909	0.020	Chromosome ExcutionStatsPname A ST 0.0 ET 48.90937 Duration 48.90937 CriticalpathDur=25.41928	23.490	0.480	0.480
GPID-9	5	A5.0;A1.0;A6.0;A2.0;A7.0;A3.0;A8.0;A4.0;A5.0-f;A6.0-f;A7.0-f;A8.0-f;A5.0-ff;A6.0-ff;A7.0-ff;A8.0-ff;A9.0;A10.0;	47.852	0.021	Chromosome ExcutionStatsPname A ST 0.0 ET 47.85245 Duration 47.85245 CriticalpathDur=23.20343	24.649	0.515	0.515
GPID-10	5	A1.0;A5.0;A6.0;A7.0;A8.0;A5.0-f;A6.0-f;A7.0-f;A9.0;A2.0;A3.0;A4.0;A1.0-f;A2.0-f;A3.0-f;A4.0-f;A1.0-ff;A2.0-ff;A3.0-ff;A10.0;	45.589	0.022	Chromosome ExcutionStatsPname A ST 0.0 ET 45.58851 Duration 45.58851 CriticalpathDur=24.92749	20.661	0.453	0.453

APPENDIX C: MODEL SOFTWARE INTERFACE

Variable Length GA:



Sampling GA:



APPENDIX D: THREE-PROJECT PORTFOLIO EXAMPLE

A portfolio of three projects: A (10 activities), B (17 activities) and C (33 activities) is considered in this example.

Project A:

A	Activity Name	1	2	3	4	5	6	7	8	9	10
1	Design PS Test	4			0.5						
2	Perform PS Test	1	10								
3	Evaluate PS Test		1	4							
4	Determine Success of Test			0.7	2						
5	Design Seasoning Test					4			0.6		
6	Perform Seasoning Test					1	10				
7	Evaluate Seasoning Test						1	4			
8	Determine Success of Seas. Test							0.5	2		
9	Verify paper replenishment rate meets project goals							1	0.9	2	
10	Report project plan to Business Manager to support				1				1	1	1

Rework Impact

A	Activity Name	1	2	3	4	5	6	7	8	9	10
1	Design PS Test	4			0.7						
2	Perform PS Test	0.2	10								
3	Evaluate PS Test		1	4							
4	Determine Success of Test			1	2						
5	Design Seasoning Test					4			0.7		
6	Perform Seasoning Test					0.5	10				
7	Evaluate Seasoning Test						0.2	4			
8	Determine Success of Seas. Test							1	2		
9	Verify paper replenishment rate meets project goals							1	1	2	
10	Report project plan to Business Manager to support				1				1	1	1

A	Time (minimum)	Time (most Likely)	Time (maximum)	Resources	Resource Quantity	Maximum Number of Reworks	Learning Factor
1	2	4	6	John, Computer	1,1	2	0.2
2	6	10	12	AI, Computer	1,3	3	0.8
3	3	4	5	John	1	2	0.9
4	1	2	3	John	1	1	0.6
5	2	4	5	John, Computer	1,2	5	0.45
6	7	10	13	AI, Computer	1,3	2	0.015
7	2	4	6	John	1	3	0.78
8	1	2	4	John	1	4	0.5
9	1	2	3	John	1	1	0.4
10	1	1	2	John	1	0	0.3

Project B:

B	Activity name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	Deliver Heplenisher Formula to Photochemicals Manufacturing	44		0.4				0.6						0.3				
2	Develop Concentrate Formula	1	5	0.6								0.4						
3	Crystallization Test		1	11														
4	Corrosion Testing		0.6		5													
5	Design Seasoning Test	1				2												
6	Run Seasoning Test					1	5											
7	Evaluate Seasoning Test						0.9	2										
8	Determine practical aim concentrations for chemical components in developer	1						1	10									
9	Verify that cost UMC are met.		1							5								
10	Produce Pilot Batch		0.7	1							20	0.8						
11	Verify by trade test that project goals are met in practical use by customers				1						1	50						
12	Furnish publications editor details for any changes in customer recommendations	1							1			0.8	10					
13	Verify compatibility of new developer formula with new paper A	0.8										1		20				
14	Verify compatibility of new developer formula with new paper B	1										1			20			
15	Verify formula meets Heq's and restrictions for Health, Environment, and	1														20		
16	Verify there is freedom to use the formula per any patents of interest	0.9	1															30
17	Report project plan to business manager to support launch	1								0.4	1	1	0.7	1	1	0.7	1	1

Rework Impact

B	Activity name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	Deliver Heplenisher Formula to Photochemicals Manufacturing	44		0.5				1						0.7				
2	Develop Concentrate Formula	1	5	0.5								0.2						
3	Crystallization Test		1	11														
4	Corrosion Testing		1		5													
5	Design Seasoning Test	0.7				2												
6	Run Seasoning Test					1	5											
7	Evaluate Seasoning Test						1	2										
8	Determine practical aim concentrations for chemical components in developer	1						1	10									
9	Verify that cost UMC are met.		0.6							5								
10	Produce Pilot Batch		1	1							20	1						
11	Verify by trade test that project goals are met in practical use by customers				0.8						1	50						
12	Furnish publications editor details for any changes in customer recommendations	1							0.7			1	10					
13	Verify compatibility of new developer formula with new paper A	1										1		20				
14	Verify compatibility of new developer formula with new paper B	0.4										0.7			20			
15	Verify formula meets Heq's and restrictions for Health, Environment, and	1														20		
16	Verify there is freedom to use the formula per any patents of interest	0.8	1															30
17	Report project plan to business manager to support launch	1								0.2	1	1	1	0.5	1	1	1	1

B	Time (minimum)	Time(most Likely)	Time (maximum)	Resources	Resource Quantity	Maximum Number of Reworks	Learning Factor
1	40	44	46	Sheridan	1	1	0.3
2	3	5	6	Jean	1	1	0.3
3	8	11	13	Jean,Tester	1,1	2	0.0
4	4	5	7	Jean,Tester	1,1	2	0.0
5	1	2	3	Sheridan,Computer	1,1	1	0.3
6	5	5	6	Mickey,Computer	1,2	2	0.8
7	1	2	4	Sheridan	1	3	0.5
8	9	10	12	Sheridan	1	1	0.7
9	4	5	7	Miroff	1	2	0.1
10	18	20	24	Miroff	1	2	0.1
11	45	50	55	Mickey	1	3	0.8
12	9	10	15	Sheridan	1	3	0.1
13	14	20	22	John	1	1	0.5
14	18	20	21	Joy	1	3	0.7
15	19	20	20	Fritz	1	2	0.2
16	27	30	32	Lanny	1	1	0.7
17	1	1	2	Sheridan	1	0	0.2

Project C:

C	Activity Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33			
1	Market need identification	30																																			
2	Quantify customer need	1	10																																		
3	Determine & Prioritize tech. req'ts		1	10		0.3					0.4																										
4	Review available tech. & current products	1	1	1	5																																
5	Patent / liter search	1	1	1	1	10																															
6	Review long term goals		1	1		1	2																														
7	Brainstorm potential solutions	1	1	1	1	1	1	10																													
8	Down select technical options (level 1)		1	1		1		1	2																												
9	Screening test to evaluate level 2 options		1	1		1				1	10																										
10	Define which S.G.'s must be compatible	1									2																										
11	Define Tank Aim constraints					1					1	1																									
12	Review raw material used in prod'n							1	1				2																								
13	Determine solubility											1	15											0.2	0.3								0.3		0.5		
14	Design DOE experiment	1				1				1	1	1	1	1	10					0.3											0.4	0.3	0.3		0.5	0.3	
15	Obtain Raw material for testing									1	1				1	15																					
16	Execute DOE														1	1	10																				
17	Submit DOE output for analytical testing														1	1																					
18	Analyze data		1	1		1	1				1	1							1	5																	
19	Choose TCA technology					1					1	1								1	15																
20	File invention summary	1				1			1	1										1	1	1															
21	Determine raw material specs		1	1										1						1	1		5		0.3									0.5			
22	Choose raw material vendor if new												1	1									1	20													
23	Develop R.O.M. tank formula from DOE		1	1		1					1	1										1	1														
24	Determine carry over rates	1																																			
25	Determine S.G. usage rates	1									1																										
26	Determine oxidation rates	1	1	1							1																										
27	Define desired replenishment rate		1	1		1					1																										
28	Develop R.O.M. replenisher formula		1			1																															
29	Develop R.O.M. concentrate formula		1																																		
30	perform mfg assess't																																				
31	Determine if intermediaries are formed																																				
32	Crystallization test	1																																			
33	Perform machine testing	1									1																										

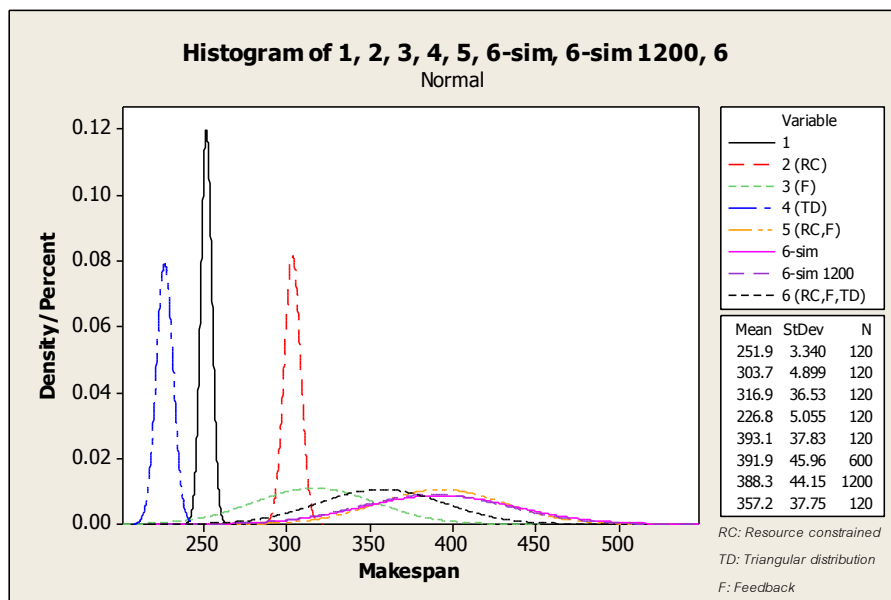
C	Activity Name	Rework Impact																																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33				
1	Market need identification	30																																				
2	Quantify customer need	1	10																																			
3	Determine & Prioritize tech. req'ts		1	10	0.3					0.4																												
4	Review available tech. & current products	1	1	1	5																																	
5	Patent / liter search	1	1	1	1	10																																
6	Review long term goals		1	1		1	2																															
7	Brainstorm potential solutions	1	1	1	1	1	1	10																														
8	Down select technical options (level 1)		1	1	1	1		1	2																													
9	Screening test to evaluate level 2 options		1	1		1				1	10																											
10	Define which S.G.'s must be compatible	1									2																											
11	Define Tank Aim constraints						1				1	1																										
12	Review raw material used in prod'n							1	1				2																									
13	Determine solubility										1	15										0.2	0.3									0.3		0.5				
14	Design DOE experiment	1				1				1	1	1	1	1	10					0.3			0.2							0.4	0.3	0.3		0.5	0.3			
15	Obtain Raw material for testing								1	1				1	15																							
16	Execute DOE													1	1	10																						
17	Submit DOE output for analytical testing														1	1					1	1																
18	Analyze data		1	1		1	1				1	1			1						1	5																
19	Choose TCA technology					1					1		1								1	15																
20	File invention summary	1				1			1	1											1	1	1															
21	Determine raw material specs		1	1									1								1	1		5										0.5				
22	Choose raw material vendor if new											1	1									1	20															
23	Develop R.O.M. tank formula from DOE		1	1		1					1	1									1	1			5													
24	Determine carry over rates	1																							5													
25	Determine S.G. usage rates	1									1															5												
26	Determine oxidation rates	1	1	1							1															5												
27	Define desired replenishment rate		1	1		1					1																							20				
28	Develop R.O.M. replenisher formula		1			1																												15	0.7	0.5	0.3	
29	Develop R.O.M. concentrate formula		1																															1	10	0.7	0.5	
30	perform mfg assess't																																					
31	Determine if intermediaries are formed																																		1	1	5	
32	Crystallization test	1																																		1		10
33	Perform machine testing	1																																				20

C	Time (minimum)	Time(most Likely)	Time (maximum)	Resources	Resource Quantity	Maximum Number of Reworks	Learning Factor
1	28	30	32	Al	1	2	0.1
2	8	10	12	Al	1	2	0.2
3	7	10	11	Miroff	1	1	0.4
4	4	5	6	Miroff	1	1	0.5
5	9	10	13	Al	1	1	0.9
6	2	2	2	Joy	1	2	0.1
7	9	10	11	Joy	1	1	0.5
8	2	2	4	Miroff	1	1	0.7
9	5	10	15	John	1	2	0.4
10	1	2	2	John	1	1	0.7
11	1	1	3	Sheridan	1	2	0.4
12	1	2	4	Fritz	1	1	0.1
13	10	15	17	Mickey,Machine	1,1	1	0.7
14	8	10	16	Mickey	1	1	0.9
15	13	15	16	Miroff	1	2	0.5
16	6	10	11	Lanny	1	1	0.1
17	1	1	2	Sheridan	1	1	0.4
18	4	5	7	John	1	2	0.6
19	13	15	16	Sheridan	1	1	0.8
20	0.5	1	2	Fritz	1	1	0.7
21	3	5	8	Jean	1	2	0.8
22	14	20	24	Joy	1	1	0.1
23	2.5	5	7.5	Jean	1	2	0.3
24	4	5	6	Sheridan	1	1	0.9
25	3	5	9	Sheridan	1	2	0.4
26	5	5	10	Sheridan	1	2	0.2
27	10	20	30	Sheridan	1	1	0.3
28	12	15	17	Jean	1	1	0.8
29	7	10	14	Jean	1	1	0.1
30	18	20	22	Joy	1	1	0.8
31	4	5	6	Jean	1	1	0.9
32	9	10	12	Jean	1	1	0.7
33	19	20	21	Mickey,Machine	1,1	1	0.8

Portfolio Resources:

Resources Available	Quantity
Al	1
John	1
Jean	1
Joy	1
Sheridan	1
Mickey	1
Miroff	1
Lanny	1
Fritz	1
Tester	2
Computer Machine	3

The resultant distributions due to different variations in the inputs are shown in the figure and table below.



Problem	Problem Description		Method Description	Sample Size	Pop	Gen	Time of execution	Mean	St Dev
1	PSP	Project scheduling problem	Sampling GA (CO=0.7, M=0.02)	120	30	15	70m 28s	251.9	3.34
2	RCPSP (constant durations)	Resource Constrained Project scheduling problem with each task having a constant duration	Sampling GA (CO=0.7, M=0.02)	120	30	15	69m 24s	303.7	4.899
3	PSP (with feedback)	Iterative Project scheduling problem	Sampling GA (CO=0.7, M=0.02)	120	30	15	70m	316.9	36.53
4	RCPSP (triangular distributed durations)	Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Sampling GA (CO=0.7, M=0.02)	120	30	15	85m	226.8	5.055
5	RCPSP (with feedback)	Iterative Resource Constrained Project scheduling problem	Sampling GA (CO=0.7, M=0.02)	120	30	15	84m 24s	393.1	37.83
6	RCPSP (with feedback & triangular distribution)	Iterative Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Sampling GA (CO=0.7, M=0.02)	120	30	15	98m	357.2	37.75
6-sim 600	RCPSP (with feedback & triangular distribution)	Iterative Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Simulation	600	1	1	2m	388.3	44.15
6-sim 1200	RCPSP (with feedback & triangular distribution)	Iterative Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Simulation	1200	1	1	12m	391.9	45.96
6	RCPSP (with feedback & triangular distribution)	Iterative Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Variable lengths GA	N/A	240	60	21m	361.3	37.3
6	RCPSP (with feedback & triangular distribution)	Iterative Resource Constrained Project scheduling problem with each a task's duration following a triangular distribution	Variable lengths GA (CO=0.7, M=0.02)	N/A	600	60	52m	357.3	36.88

It is obvious how the presence of resource constraints shifted the distribution to the right given resulting in higher makespans. Assuming triangular distribution of the tasks duration gives a more realistic and normal distribution of the project, while it interestingly shifts the distribution to the left. The presence of feedback definitely shifts the distribution to the right, since rework increases the makespan of the project. Besides, the GA showed to be consistent with the simulation process showing that GA can be used to find the distribution of the project. However, the distribution resulting from GA, is more left-deviated than that resulting from simulation as the GA tends to produce optimal or near optimal makespans.

APPENDIX E: HARTMANN'S BENCHMARK

Example j301_1:

The project form PSLIB is defined as follows:

```

*****
projects                : 1
jobs (incl. supersource/sink ) : 32
horizon                 : 158
RESOURCES
- renewable             : 4   R
- nonrenewable          : 0   N
- doubly constrained    : 0   D
*****
PROJECT INFORMATION:
pronr.  #jobs rel.dateduedatetardcostMPM-Time
      1   30   0   38   26   38
*****
PRECEDENCE RELATIONS:
jobnr.  #modes #successors  successors
      1     1     3         2  3  4
      2     1     3         6 11 15
      3     1     3         7  8 13
      4     1     3         5  9 10
      5     1     1         20
      6     1     1         30
      7     1     1         27
      8     1     3         12 19 27
      9     1     1         14
     10     1     2         16 25
     11     1     2         20 26
     12     1     1         14
     13     1     2         17 18
     14     1     1         17
     15     1     1         25
     16     1     2         21 22
     17     1     1         22
     18     1     2         20 22
     19     1     2         24 29
     20     1     2         23 25
     21     1     1         28
     22     1     1         23
     23     1     1         24
     24     1     1         30
     25     1     1         30
     26     1     1         31
     27     1     1         28
     28     1     1         31
     29     1     1         32
     30     1     1         32
     31     1     1         32
     32     1     0

```


REQUESTS/DURATIONS:

jobnr. mode duration R 1 R 2 R 3 R 4

```
-----
-----
  1      1      0      0      0      0      0
  2      1      8      4      0      0      0
  3      1      4     10      0      0      0
  4      1      6      0      0      0      3
  5      1      3      3      0      0      0
  6      1      8      0      0      0      8
  7      1      5      4      0      0      0
  8      1      9      0      1      0      0
  9      1      2      6      0      0      0
 10     1      7      0      0      0      1
 11     1      9      0      5      0      0
 12     1      2      0      7      0      0
 13     1      6      4      0      0      0
 14     1      3      0      8      0      0
 15     1      9      3      0      0      0
 16     1     10      0      0      0      5
 17     1      6      0      0      0      8
 18     1      5      0      0      0      7
 19     1      3      0      1      0      0
 20     1      7      0     10      0      0
 21     1      2      0      0      0      6
 22     1      7      2      0      0      0
 23     1      2      3      0      0      0
 24     1      3      0      9      0      0
 25     1      3      4      0      0      0
 26     1      7      0      0      4      0
 27     1      8      0      0      0      7
 28     1      3      0      8      0      0
 29     1      7      0      7      0      0
 30     1      2      0      7      0      0
 31     1      2      0      0      2      0
 32     1      0      0      0      0      0
```


RESOURCEAVAILABILITIES:

R 1 R 2 R 3 R 4
 12 13 4 12

Excel replication:

j	30	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32			
1																																					
2		1																																			
3		1																																			
4		1																																			
5			1		1																																
6						1																															
7				1			1																														
8				1				1																													
9					1				1																												
10					1					1																											
11			1								1																										
12									1																												
13				1																																	
14										1			1																								
15			1																																		
16												1																									
17																1	1																				
18																																					
19										1																											
20					1							1											1														
21																							1														
22																							1	1	1												
23																																					
24																																					
25																																					
26																																					
27																																					
28																																					
29																																					
30																																					
31																																					
32																																					

Resources Available	Quantity
R1	12
R2	13
R3	4
R4	12

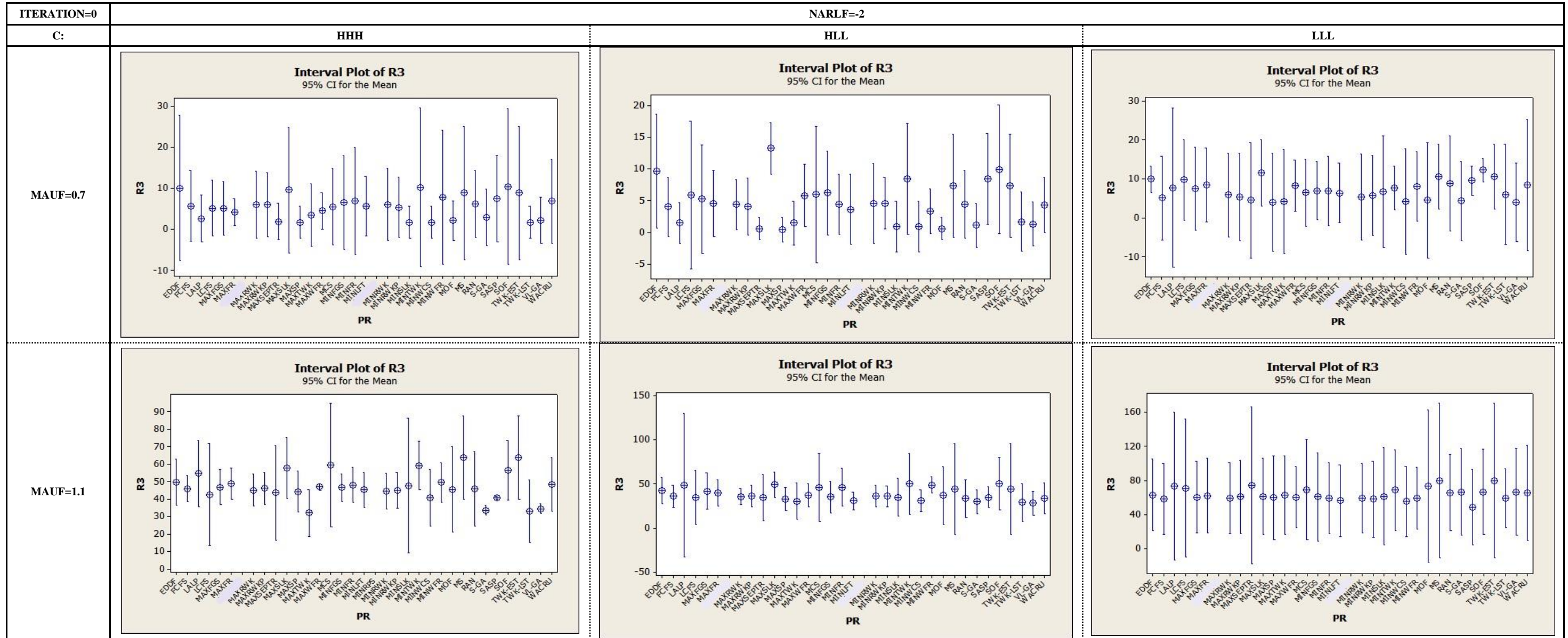
j301_1	Time (minimum)	Time(most Likely)	Time (maximum)	Resources	Resource Quantiti	Maximum Number of Rework	Learning Factor
1	0	0	0	R1,R2,R3,R4	0,0,0,0	0	0.0
2	8	8	8	R1,R2,R3,R4	4,0,0,0	0	0.0
3	4	4	4	R1,R2,R3,R4	10,0,0,0	0	0.0
4	6	6	6	R1,R2,R3,R4	0,0,0,3	0	0.0
5	3	3	3	R1,R2,R3,R4	3,0,0,0	0	0.0
6	8	8	8	R1,R2,R3,R4	0,0,0,8	0	0.0
7	5	5	5	R1,R2,R3,R4	4,0,0,0	0	0.0
8	9	9	9	R1,R2,R3,R4	0,1,0,0	0	0.0
9	2	2	2	R1,R2,R3,R4	6,0,0,0	0	0.0
10	7	7	7	R1,R2,R3,R4	0,0,0,1	0	0.0
11	9	9	9	R1,R2,R3,R4	0,5,0,0	0	0.0
12	2	2	2	R1,R2,R3,R4	0,7,0,0	0	0.0
13	6	6	6	R1,R2,R3,R4	4,0,0,0	0	0.0
14	3	3	3	R1,R2,R3,R4	0,8,0,0	0	0.0
15	9	9	9	R1,R2,R3,R4	3,0,0,0	0	0.0
16	10	10	10	R1,R2,R3,R4	0,0,0,5	0	0.0
17	6	6	6	R1,R2,R3,R4	0,0,0,8	0	0.0
18	5	5	5	R1,R2,R3,R4	0,0,0,7	0	0.0
19	3	3	3	R1,R2,R3,R4	0,1,0,0	0	0.0
20	7	7	7	R1,R2,R3,R4	0,10,0,0	0	0.0
21	2	2	2	R1,R2,R3,R4	0,0,0,6	0	0.0
22	7	7	7	R1,R2,R3,R4	2,0,0,0	0	0.0
23	2	2	2	R1,R2,R3,R4	3,0,0,0	0	0.0
24	3	3	3	R1,R2,R3,R4	0,9,0,0	0	0.0
25	3	3	3	R1,R2,R3,R4	4,0,0,0	0	0.0
26	7	7	7	R1,R2,R3,R4	0,0,4,0	0	0.0
27	8	8	8	R1,R2,R3,R4	0,0,0,7	0	0.0
28	3	3	3	R1,R2,R3,R4	0,8,0,0	0	0.0
29	7	7	7	R1,R2,R3,R4	0,7,0,0	0	0.0
30	2	2	2	R1,R2,R3,R4	0,7,0,0	0	0.0
31	2	2	2	R1,R2,R3,R4	0,0,2,0	0	0.0
32	0	0	0	R1,R2,R3,R4	0,0,0,0	0	0.0

In our study, we have used the ProGen problem instances with 30, 60, 90 and 120 non-dummy activities. The results for running the projects (j301_1, j302_5, j3048_10), (j601_1, j602_5, j6048_10), (j901_1, j902_5, j9048_10), (j1201_1, j1202_5, j12048_10) are shown in the following table.

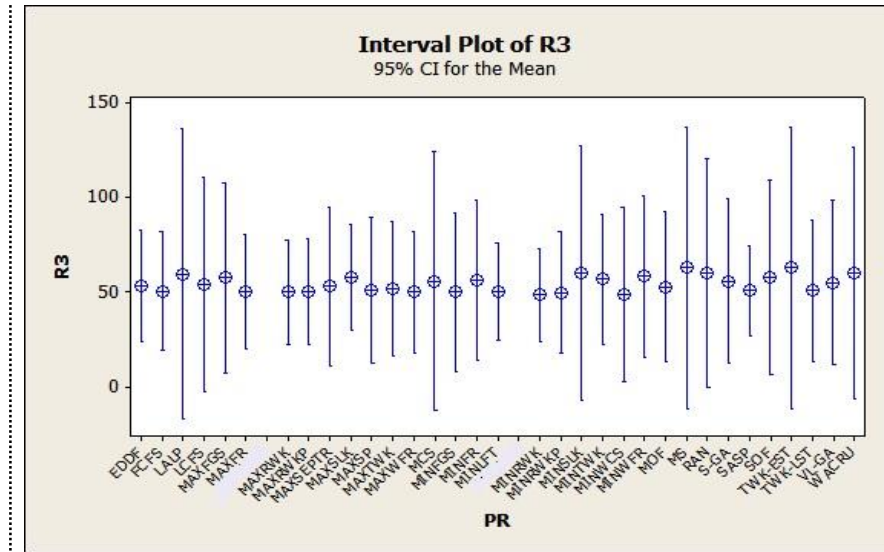
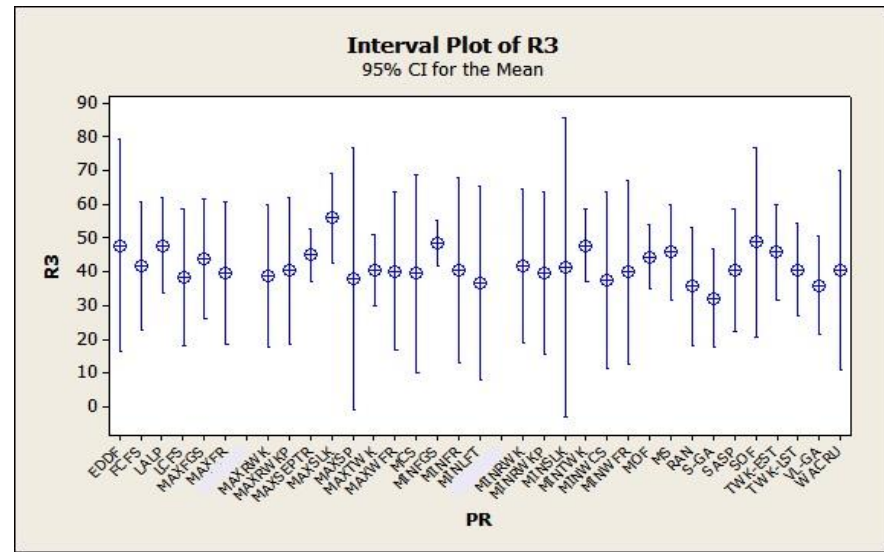
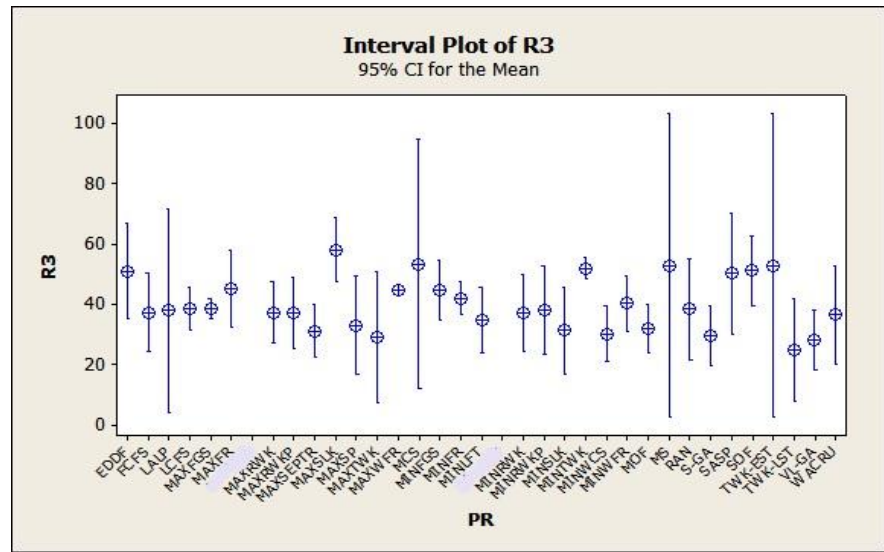
Project	Algorithm	Av. Deviation	Optimal	Av. Time of execution
j301_1	S-GA / VL-GA	0.00%	100%	19 s
j302_5				
j3048_10				
j601_1	S-GA / VL-GA	0.00%	100%	46 s
j602_5				
j6048_10				
j901_1	S-GA / VL-GA	0.00%	100%	28 s
j902_5				
j9048_10				
j1201_1	S-GA / VL-GA	0.00%	100%	2m 40s
j1202_5				
j12048_10				

APPENDIX F: INTERVAL PLOT GRAPHS OF O1 AND O2

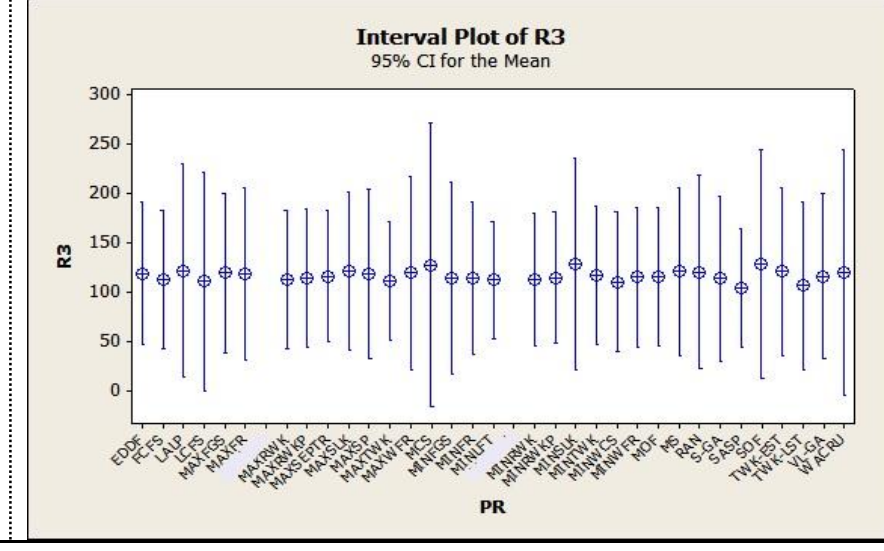
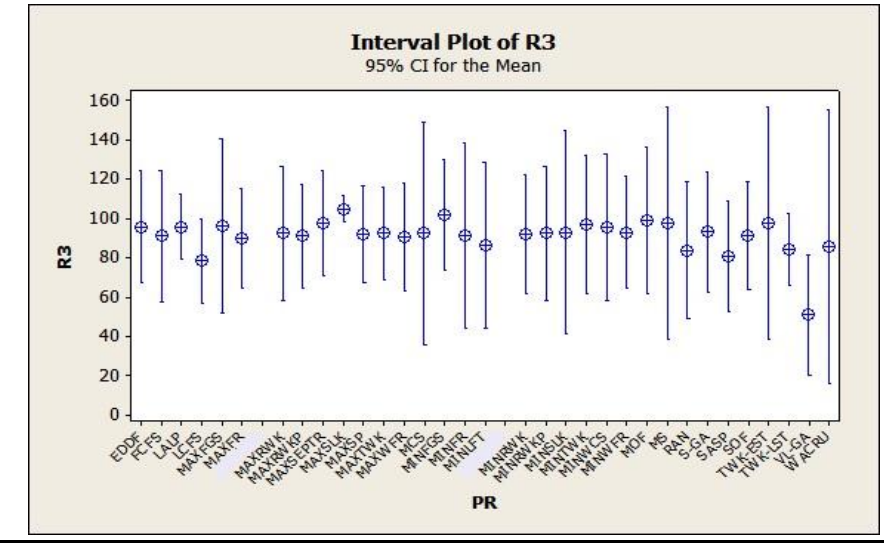
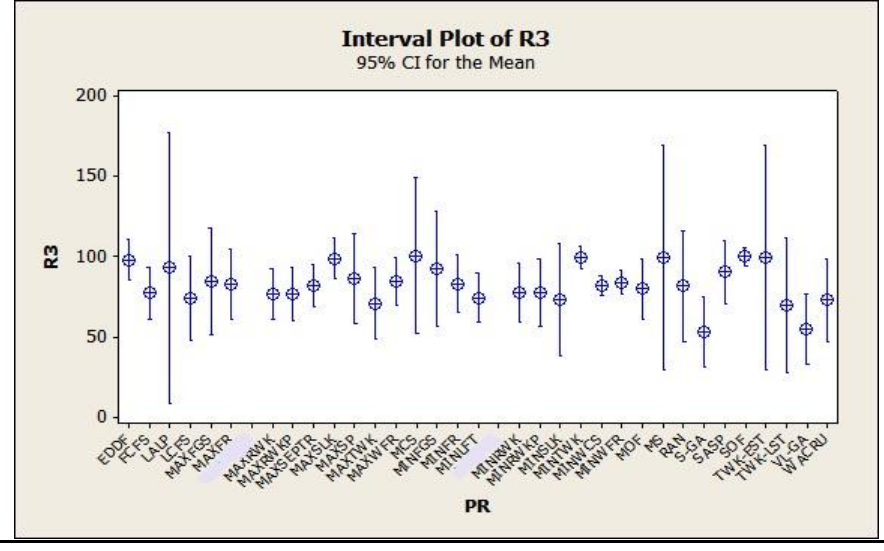
Note that R3 and R5 mentioned in the graphs are the notations for our objective functions O1 and O2 as mentioned by Browning & Yassine (2010).

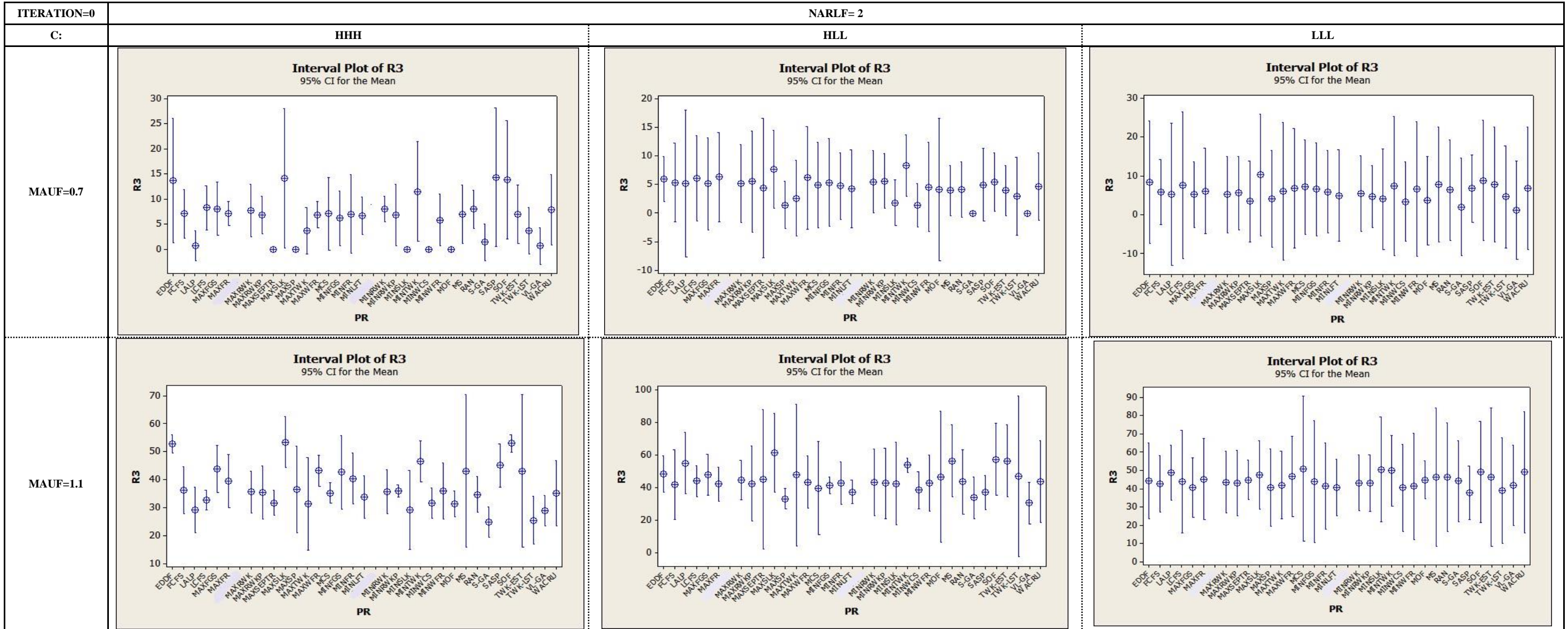


MAUF=1.1

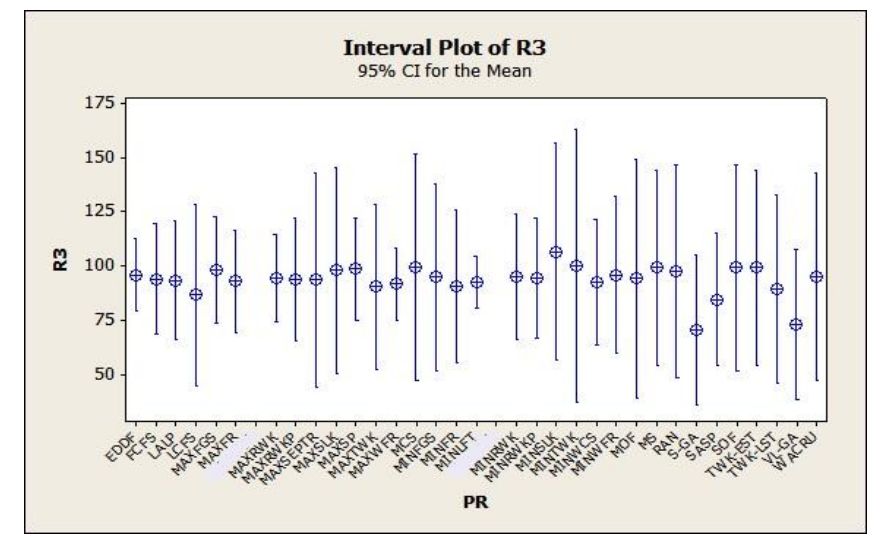
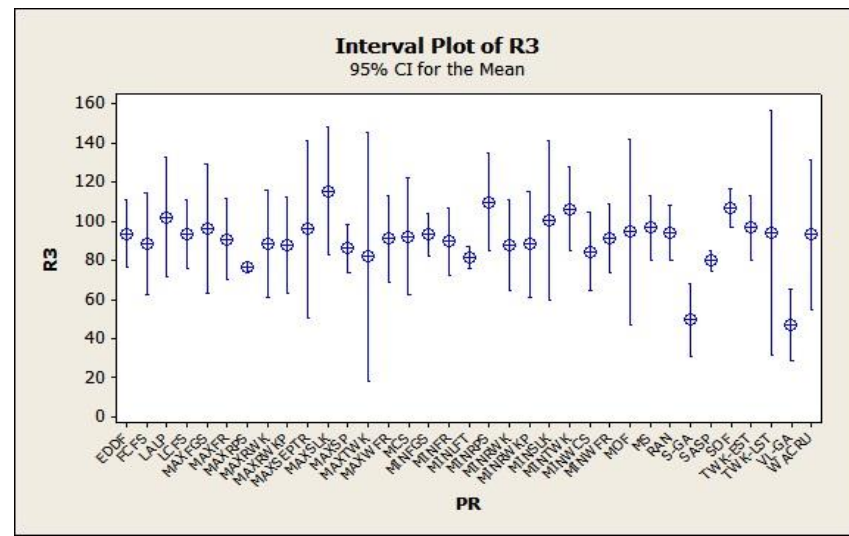
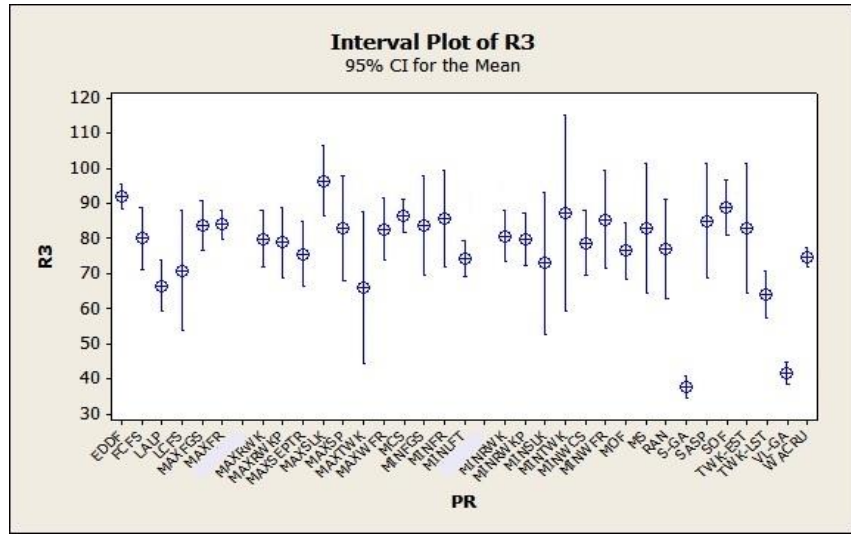


MAUF=1.5





MAUF=1.5



ITERATION=0.1

NARLF=-2

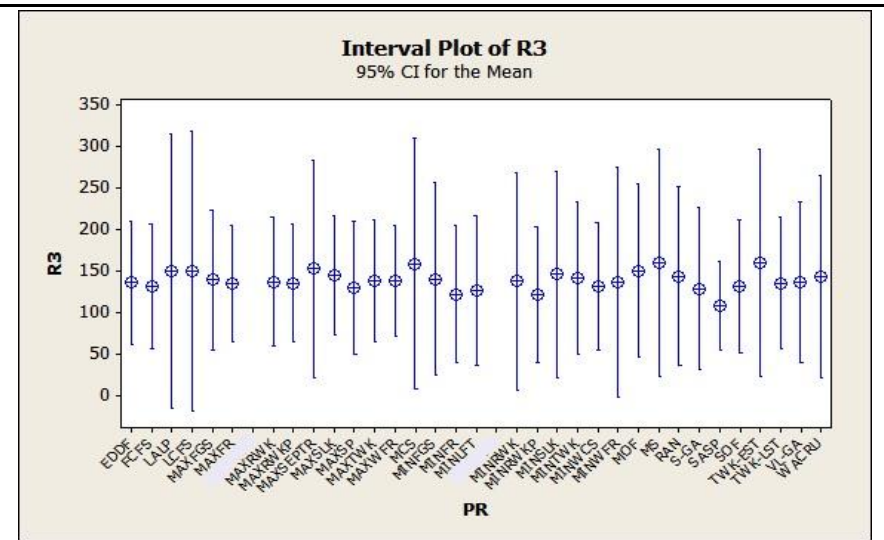
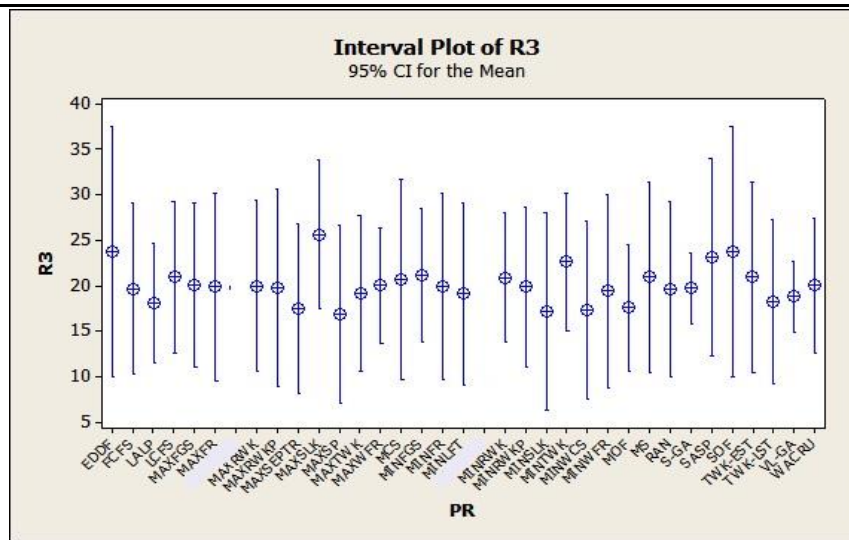
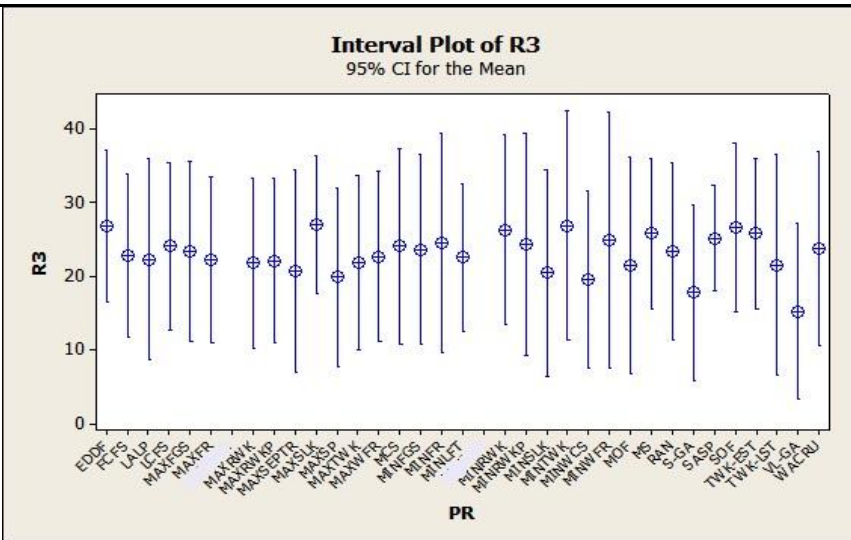
C:

HHH

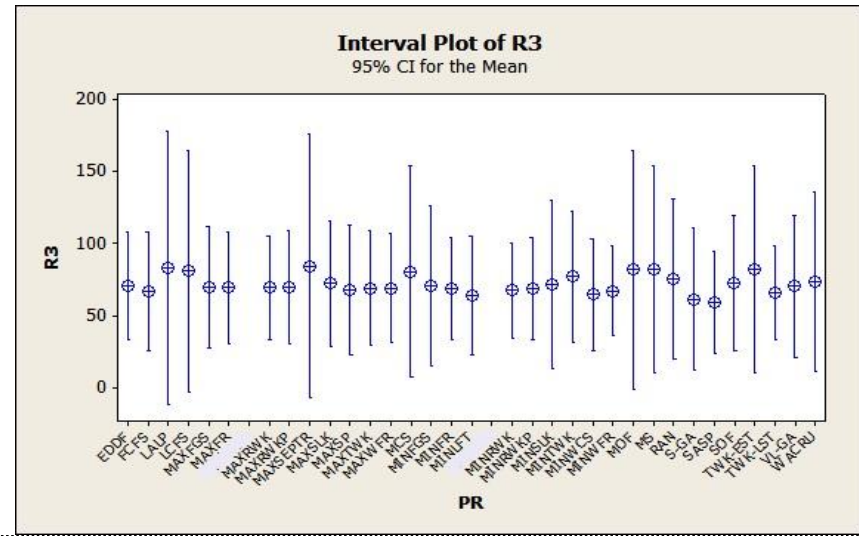
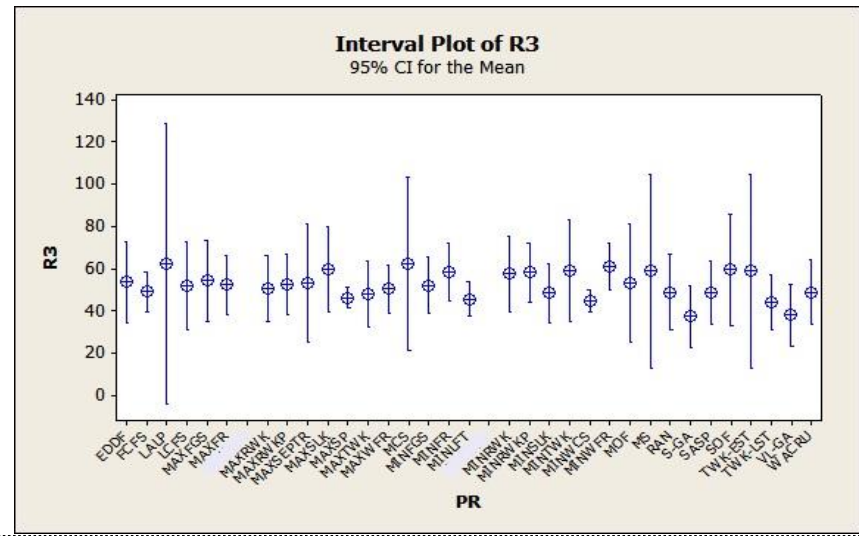
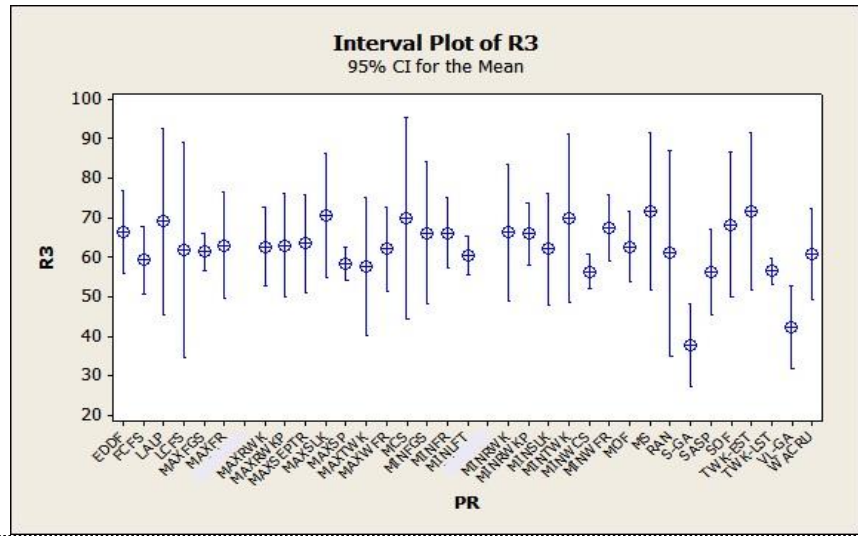
HLL

LLL

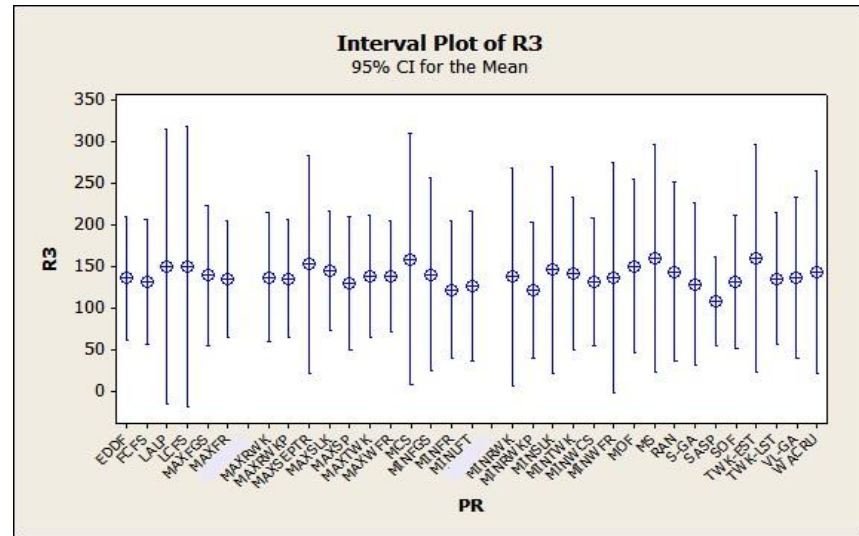
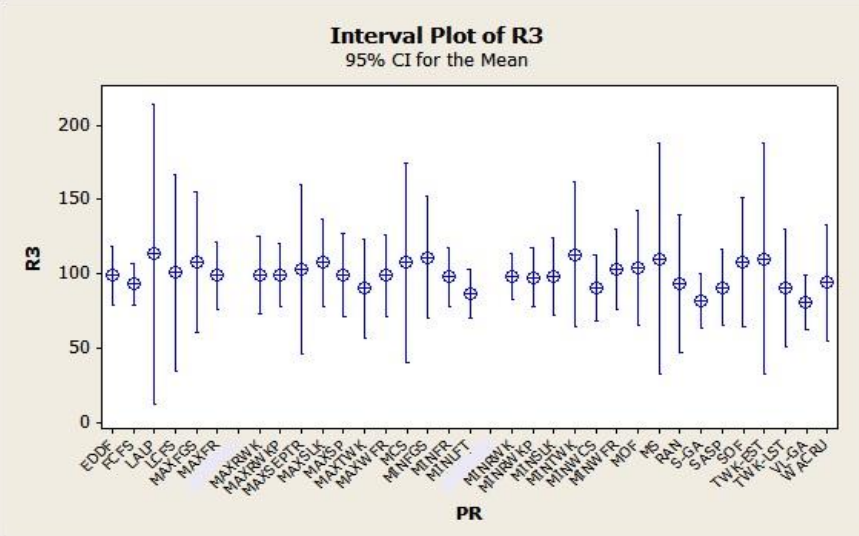
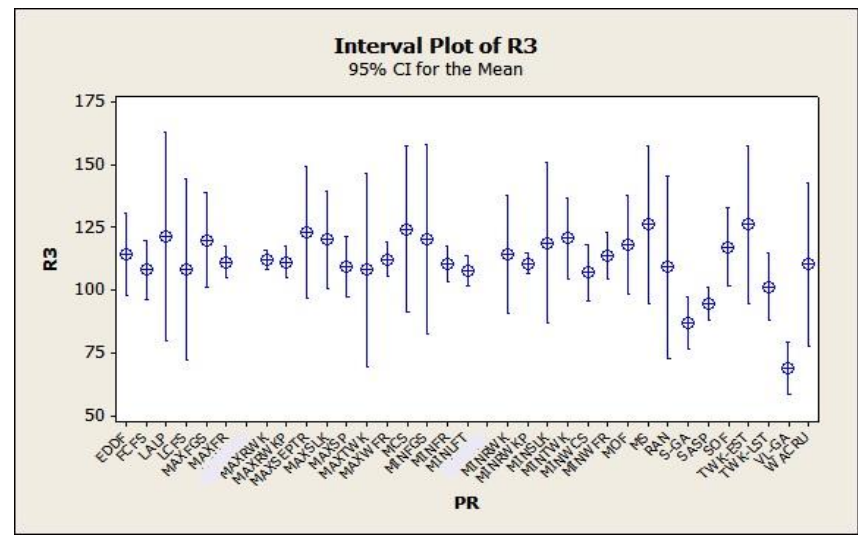
MAUF=0.7

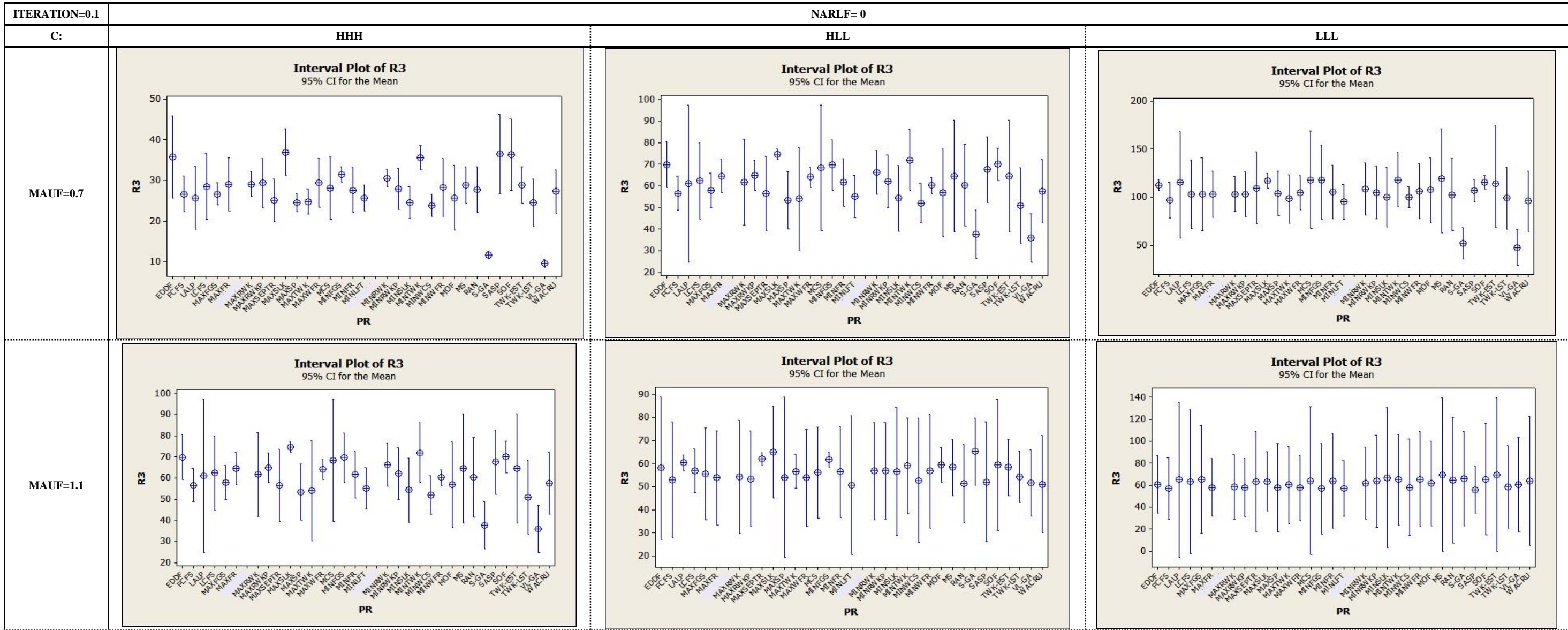


MAUF=1.1

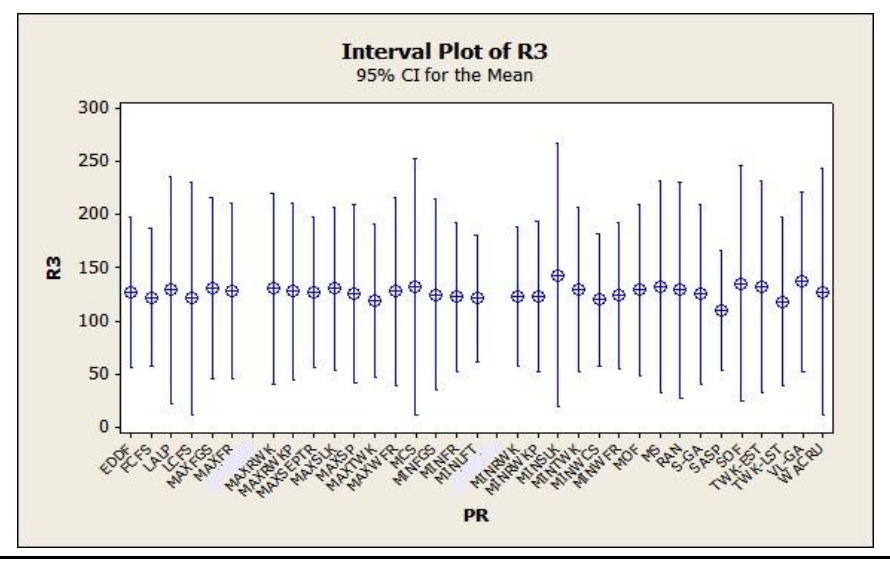
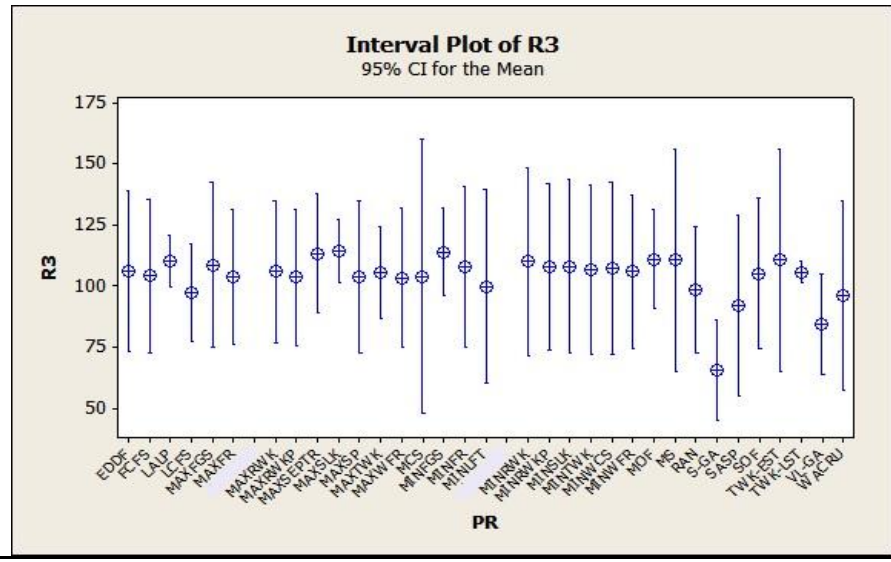
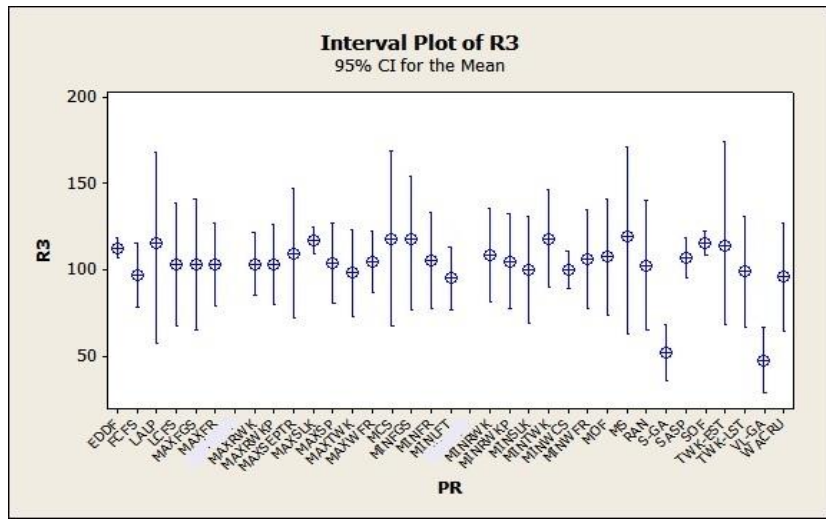


MAUF=1.5





MAUF=1.5



ITERATION=0.1

NARLF= 2

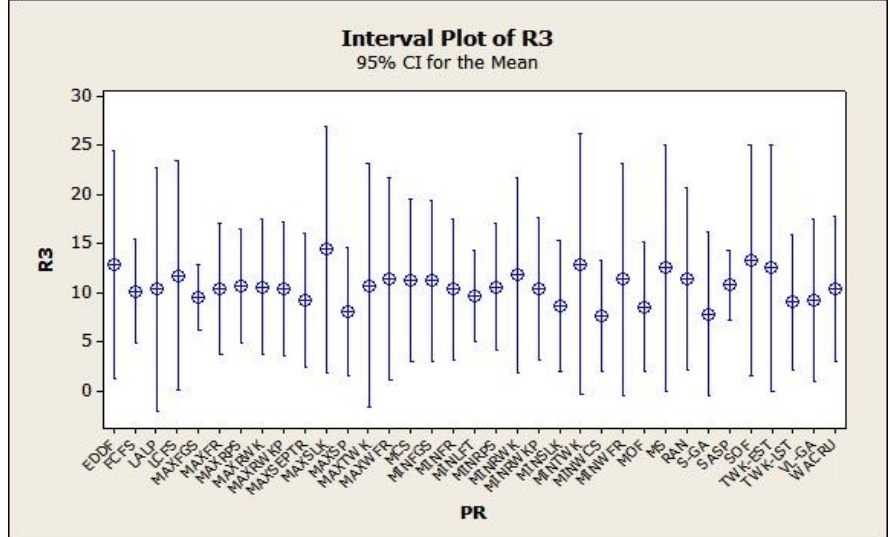
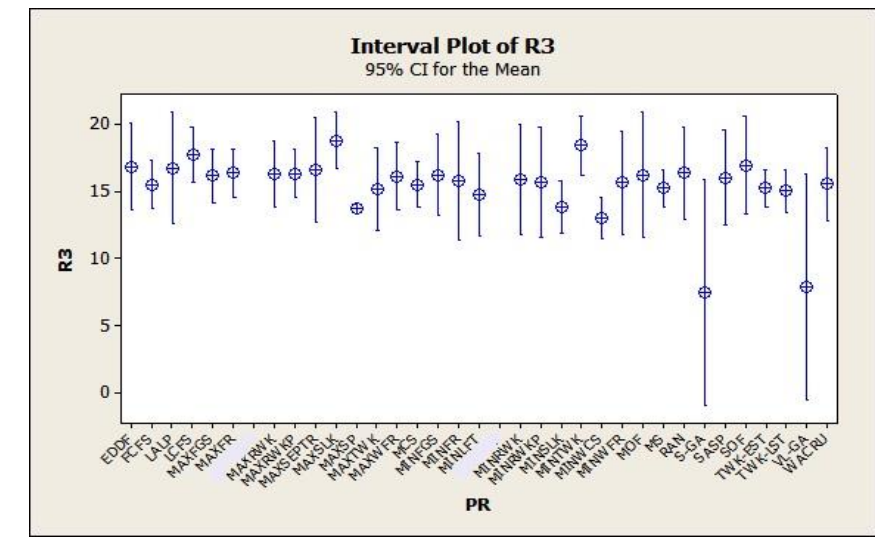
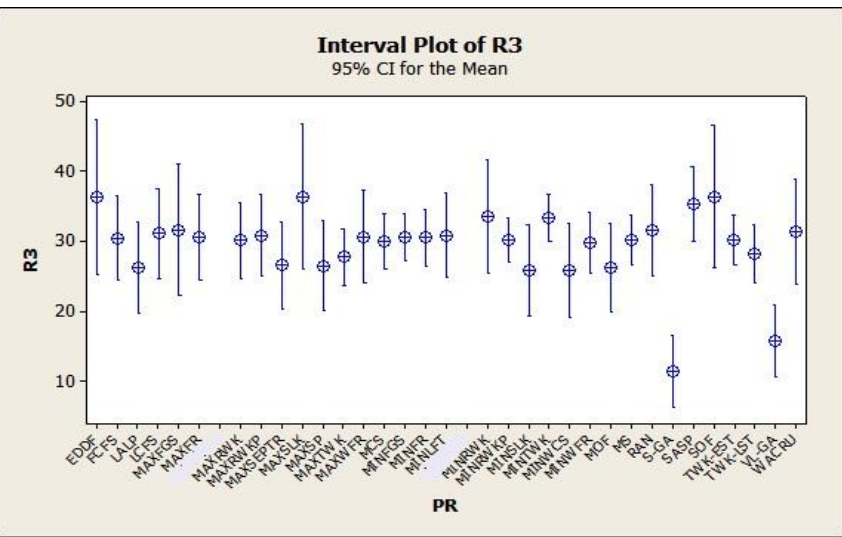
C:

HHH

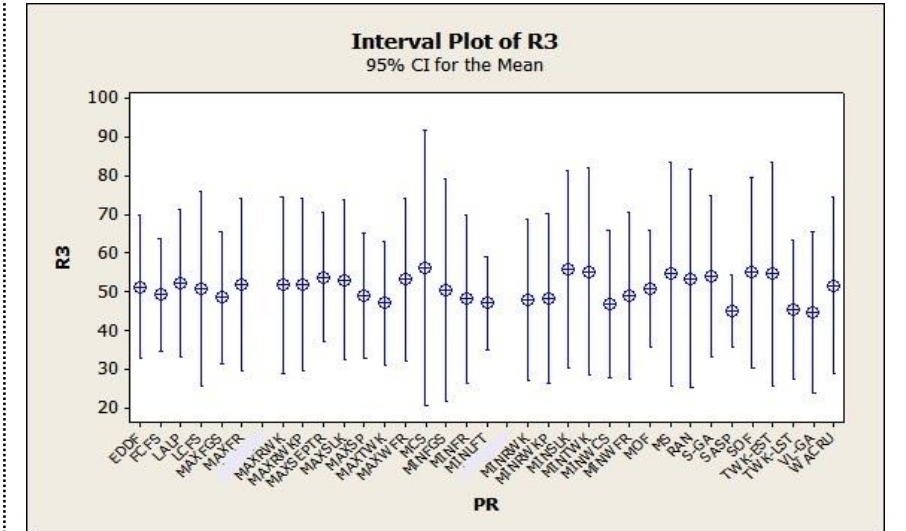
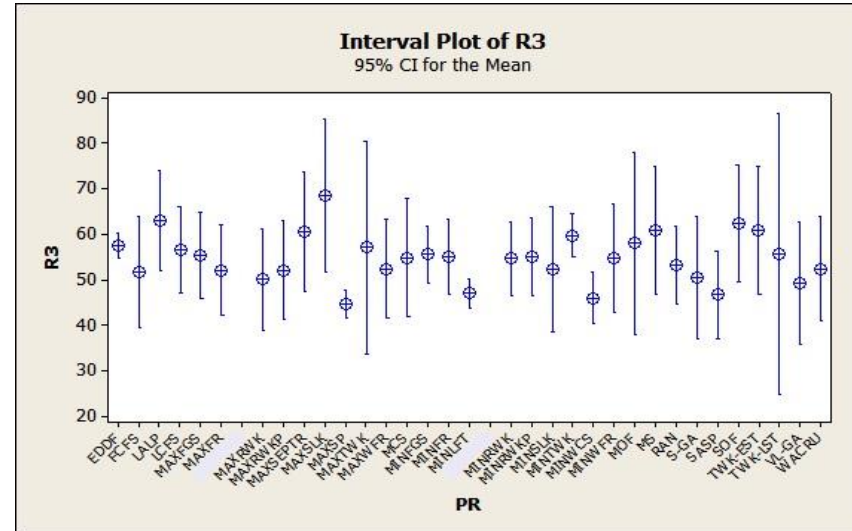
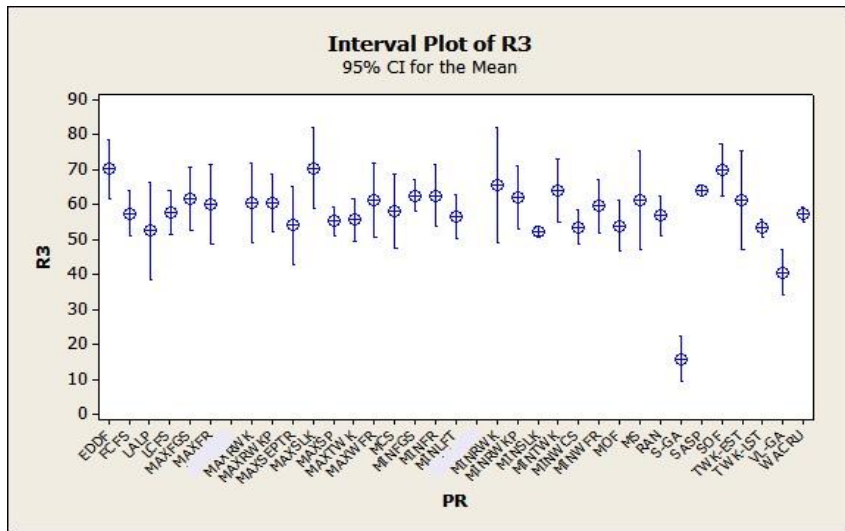
HLL

LLL

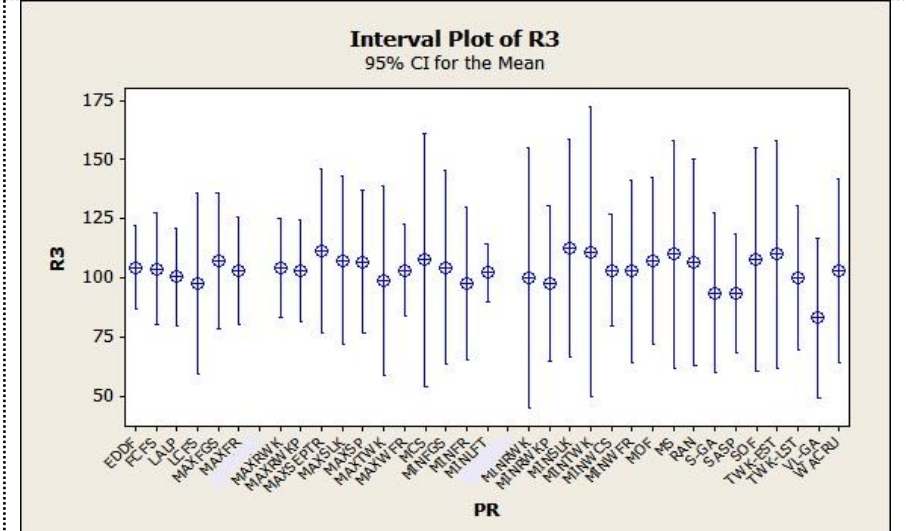
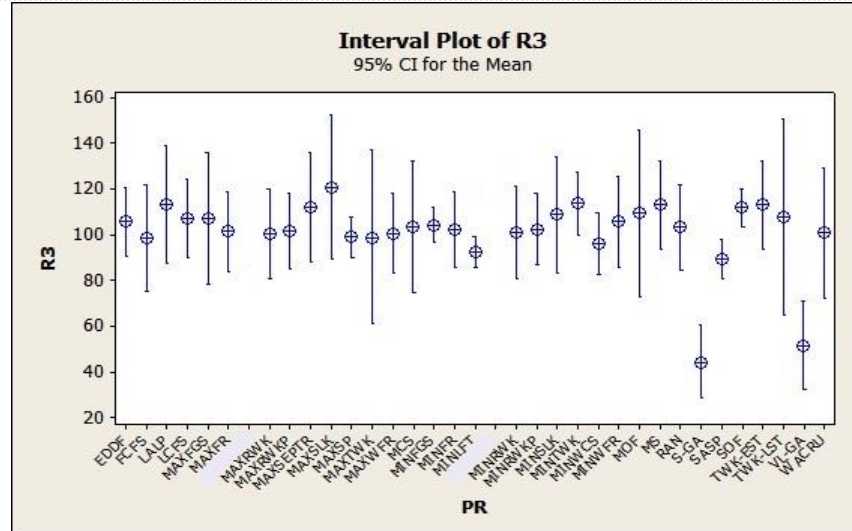
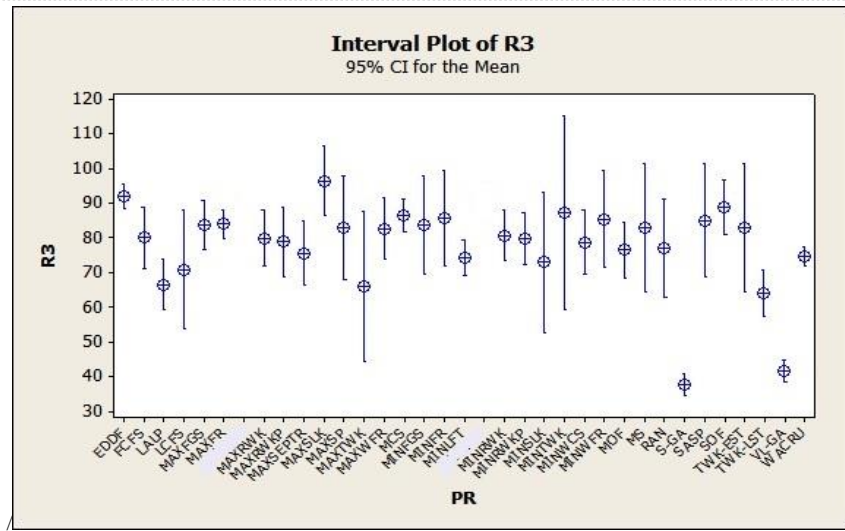
MAUF=0.7



MAUF=1.1

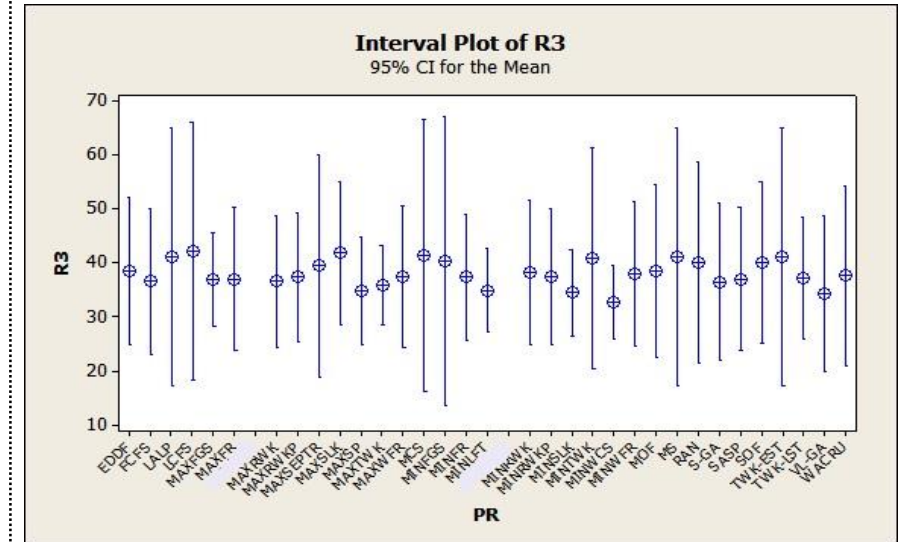
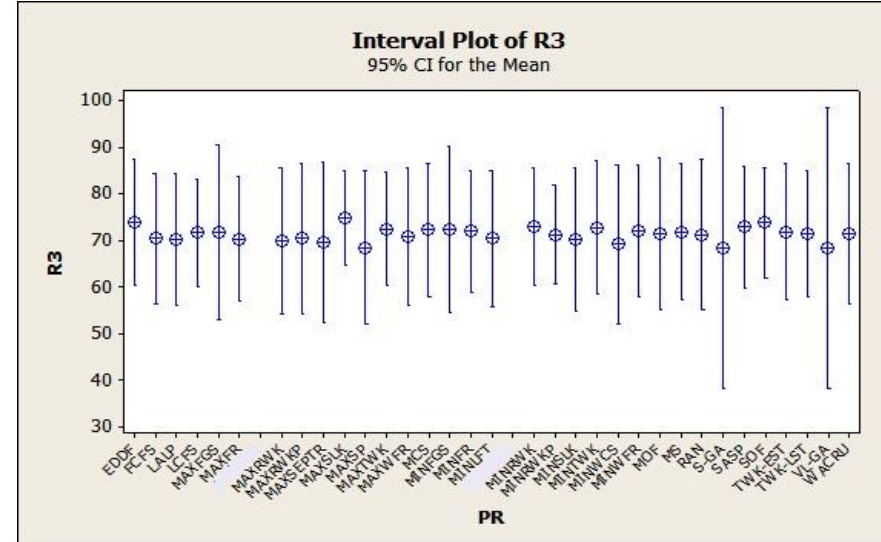
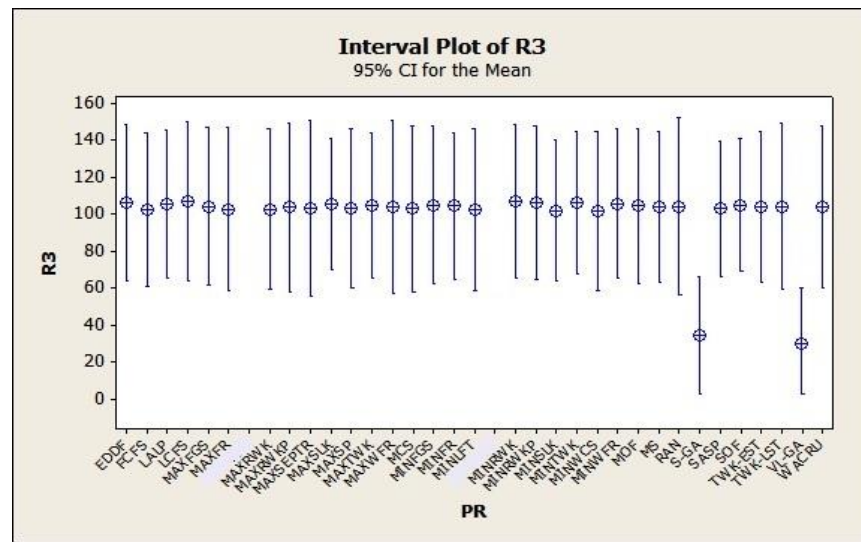


MAUF=1.5

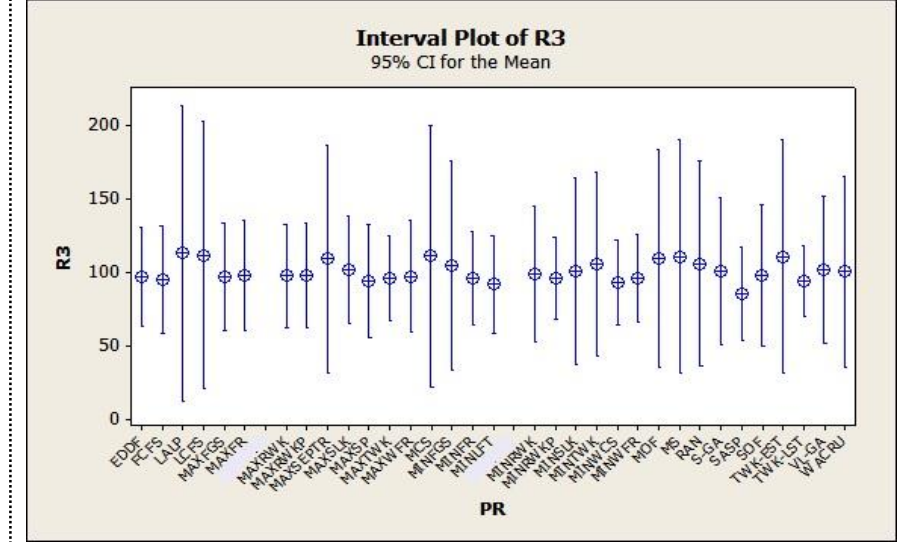
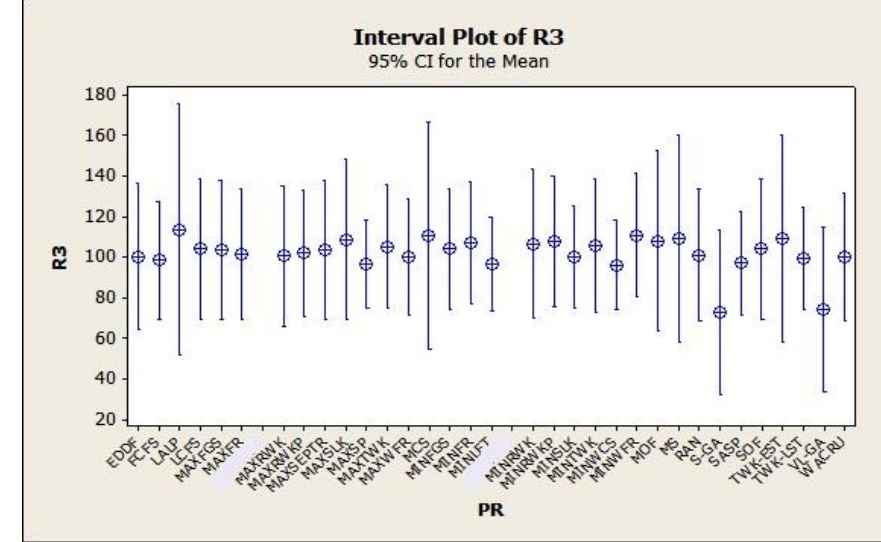
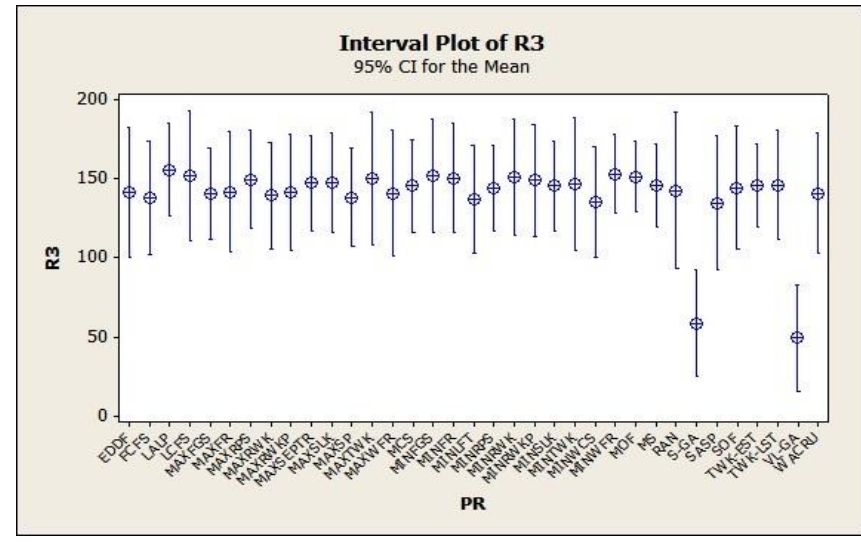


ITERATION=0.25	NARLF=-2		
C:	HHH	HLL	LLL

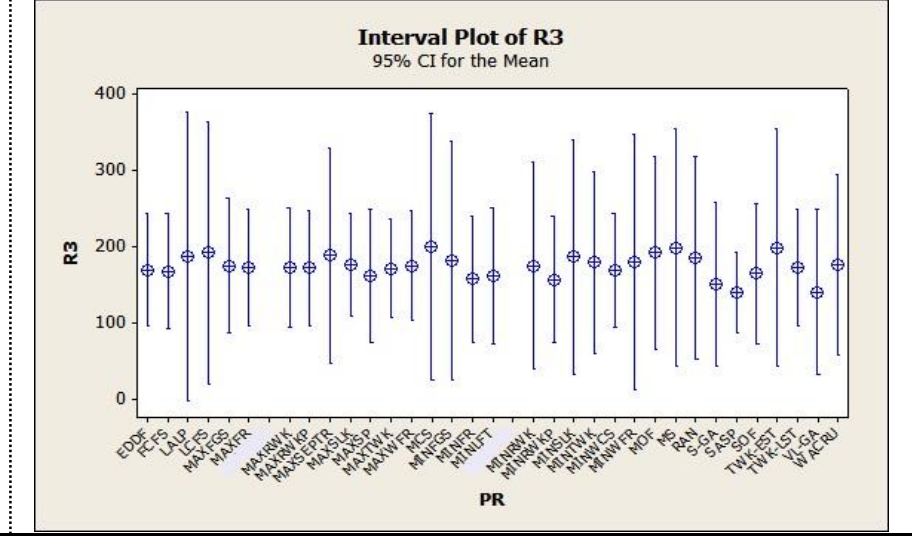
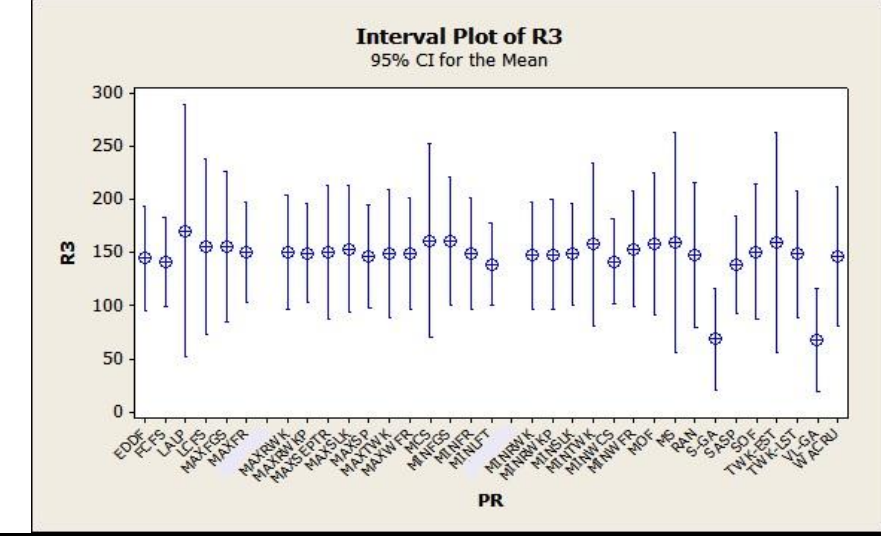
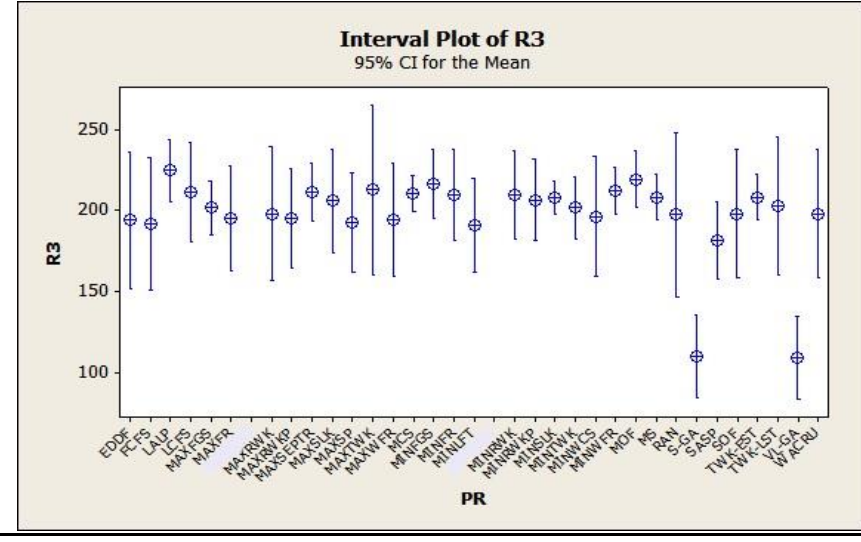
MAUF=0.7

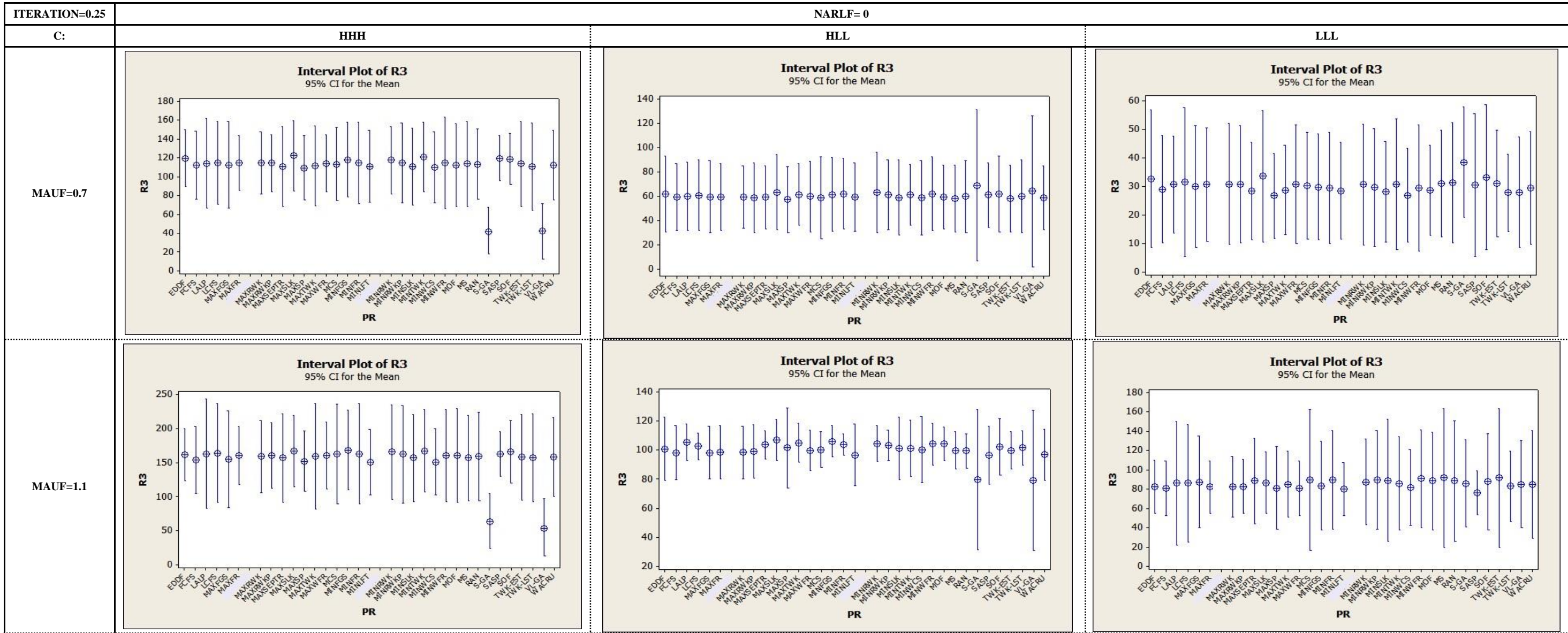


MAUF=1.1

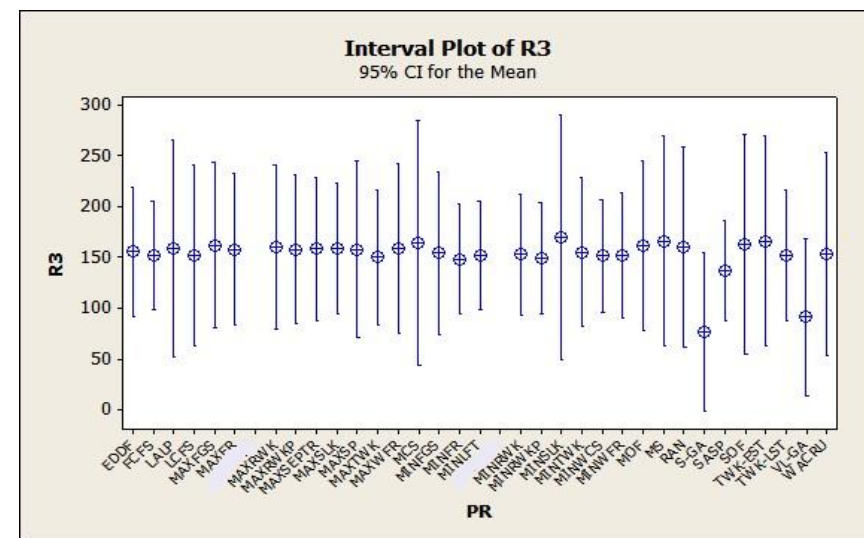
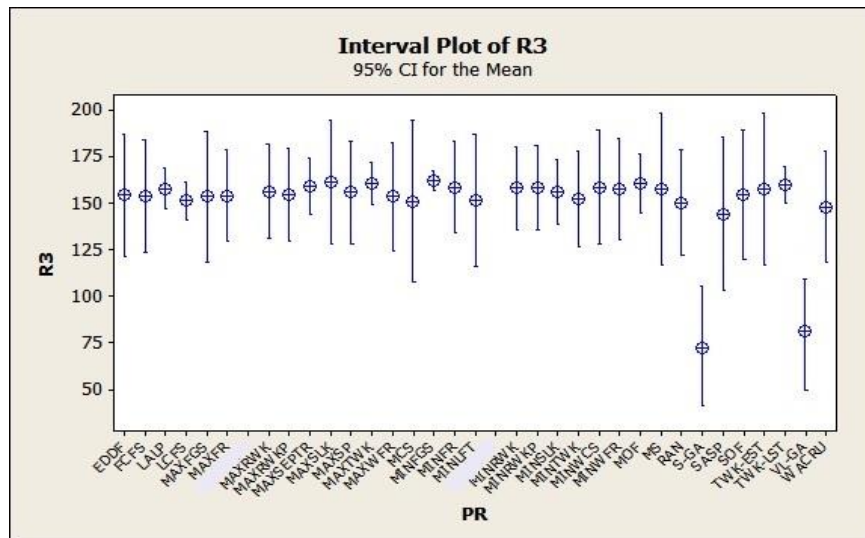
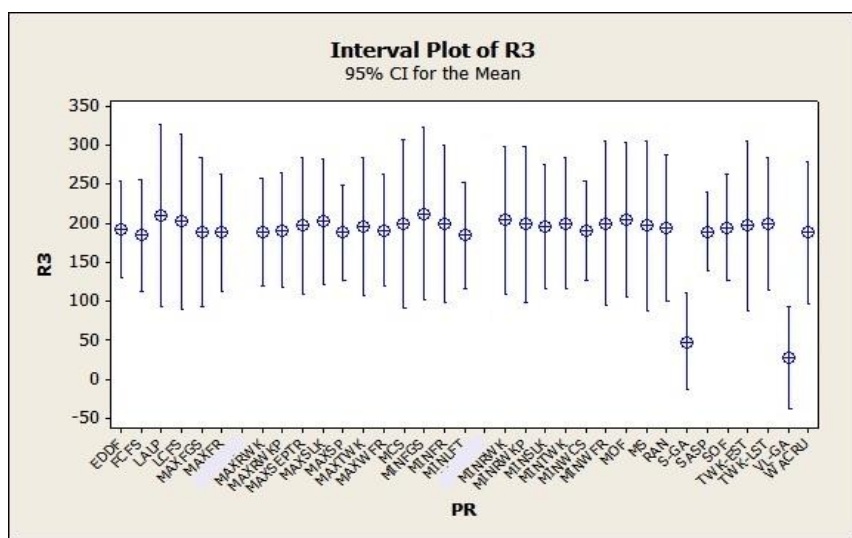


MAUF=1.5





MAUF=1.5



ITERATION=0.25

NARLF= 2

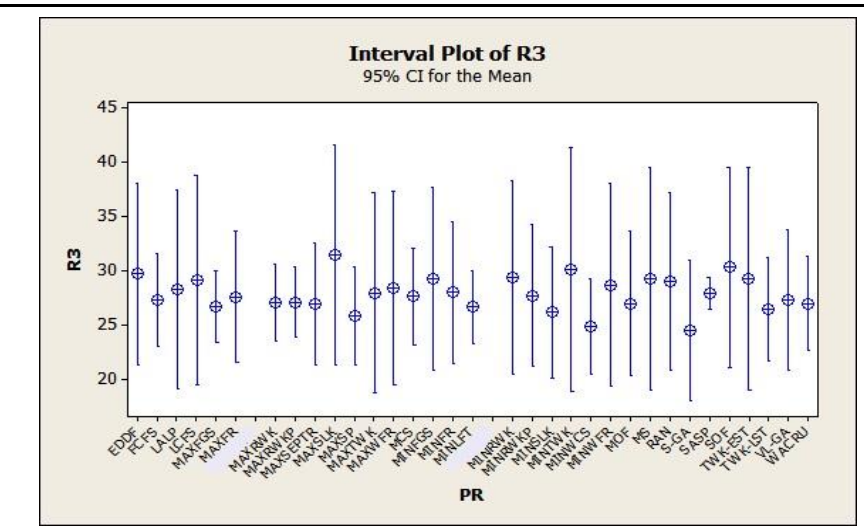
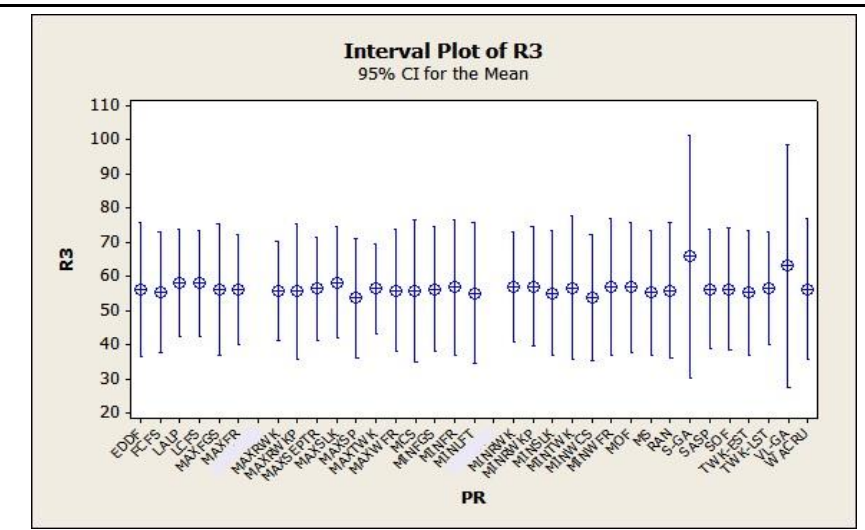
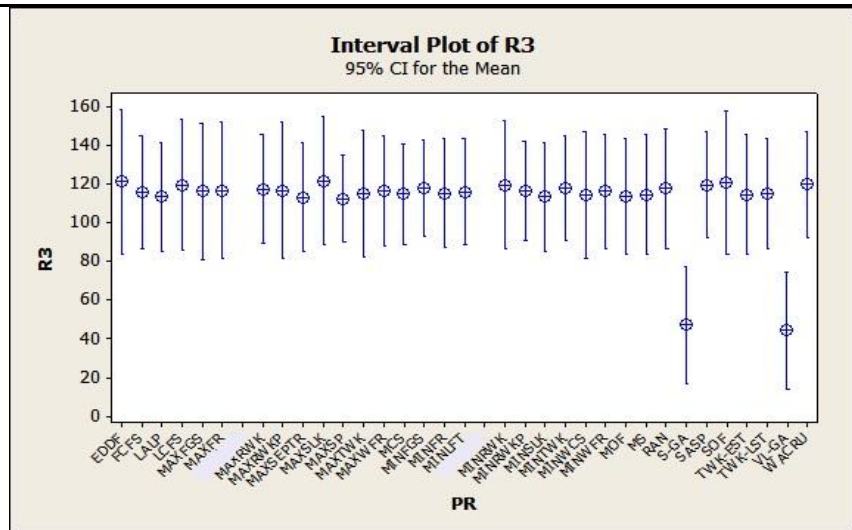
C:

HHH

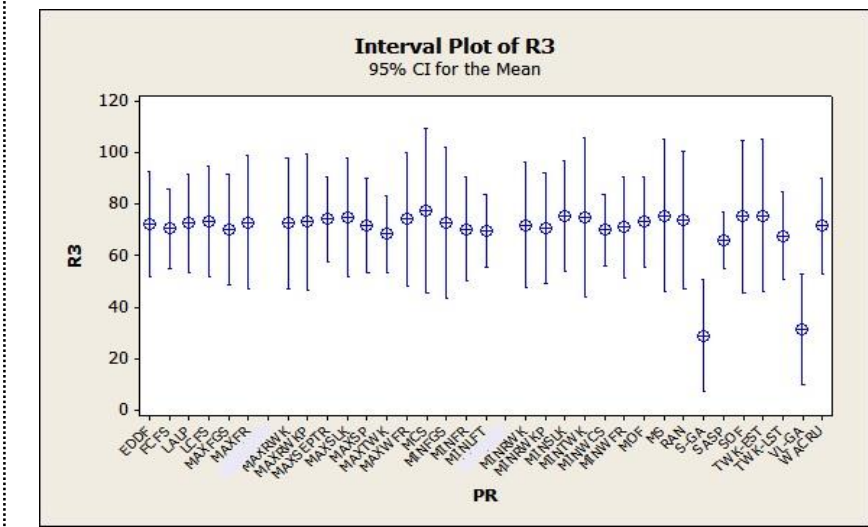
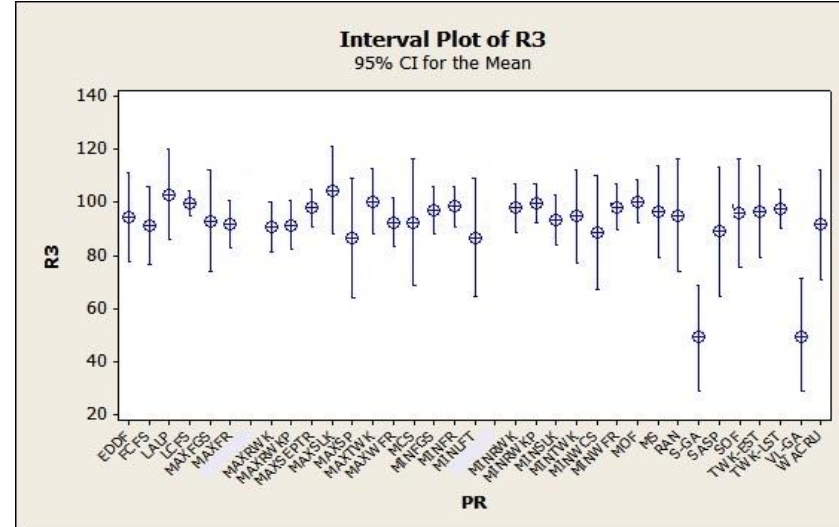
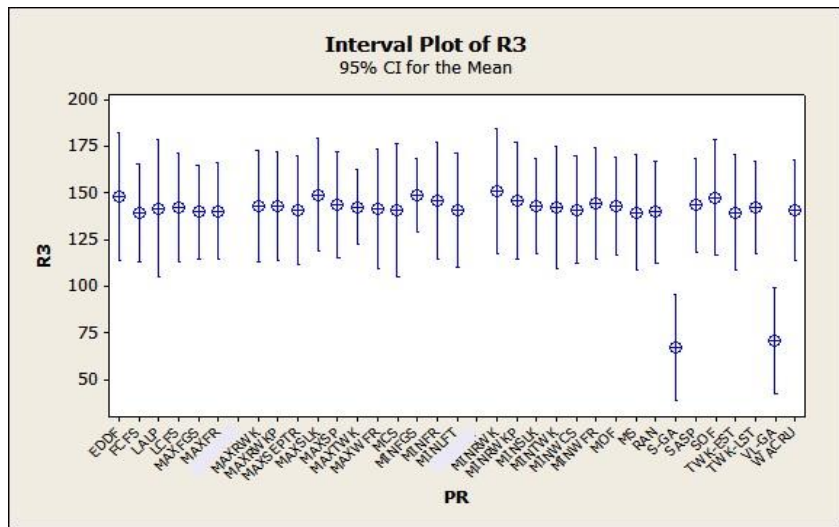
HLL

LLL

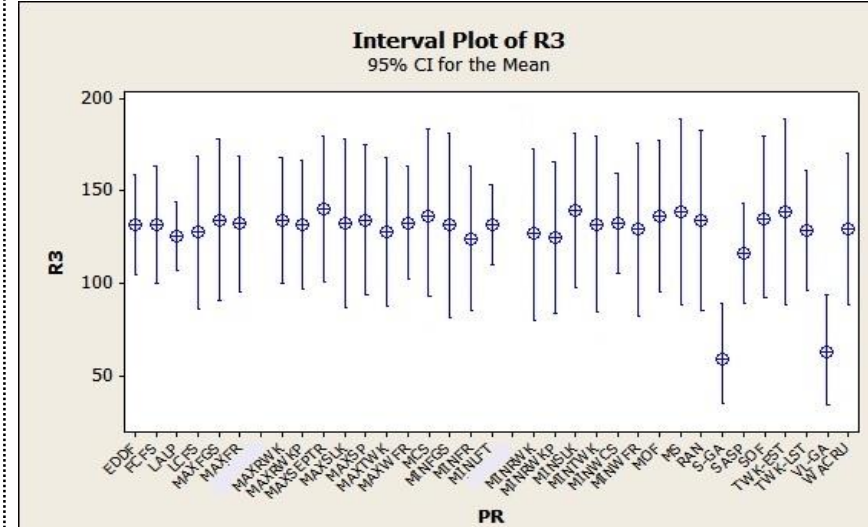
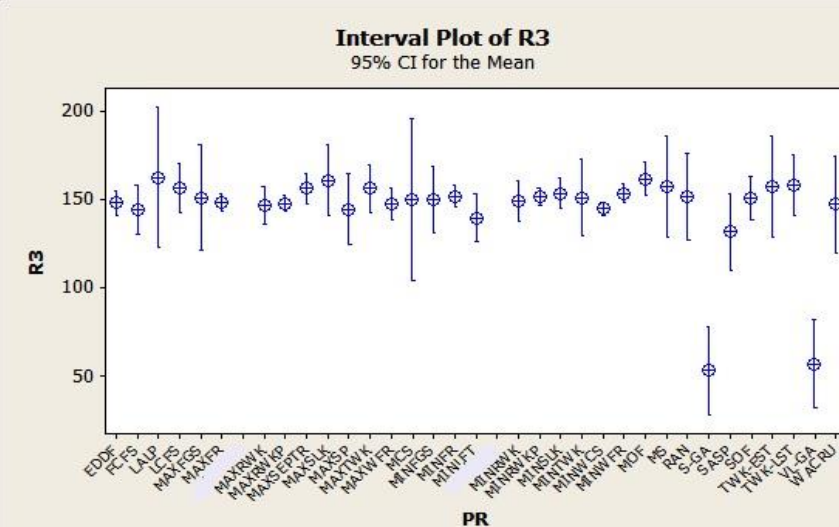
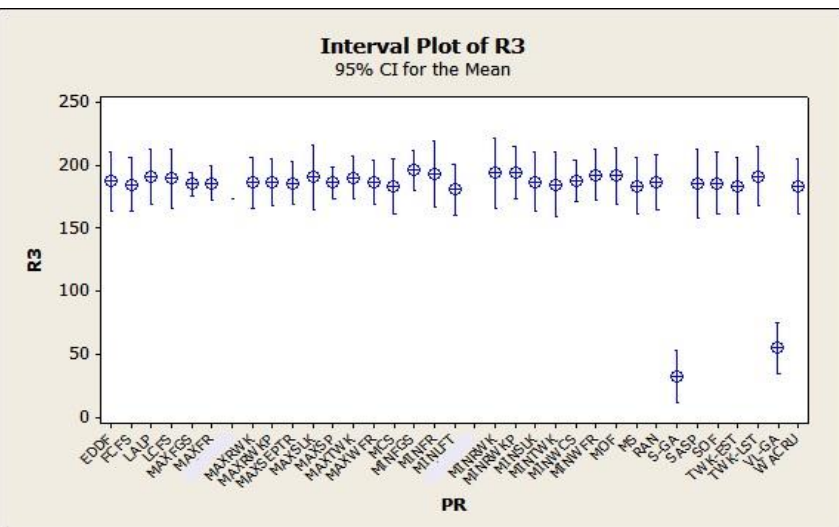
MAUF=0.7

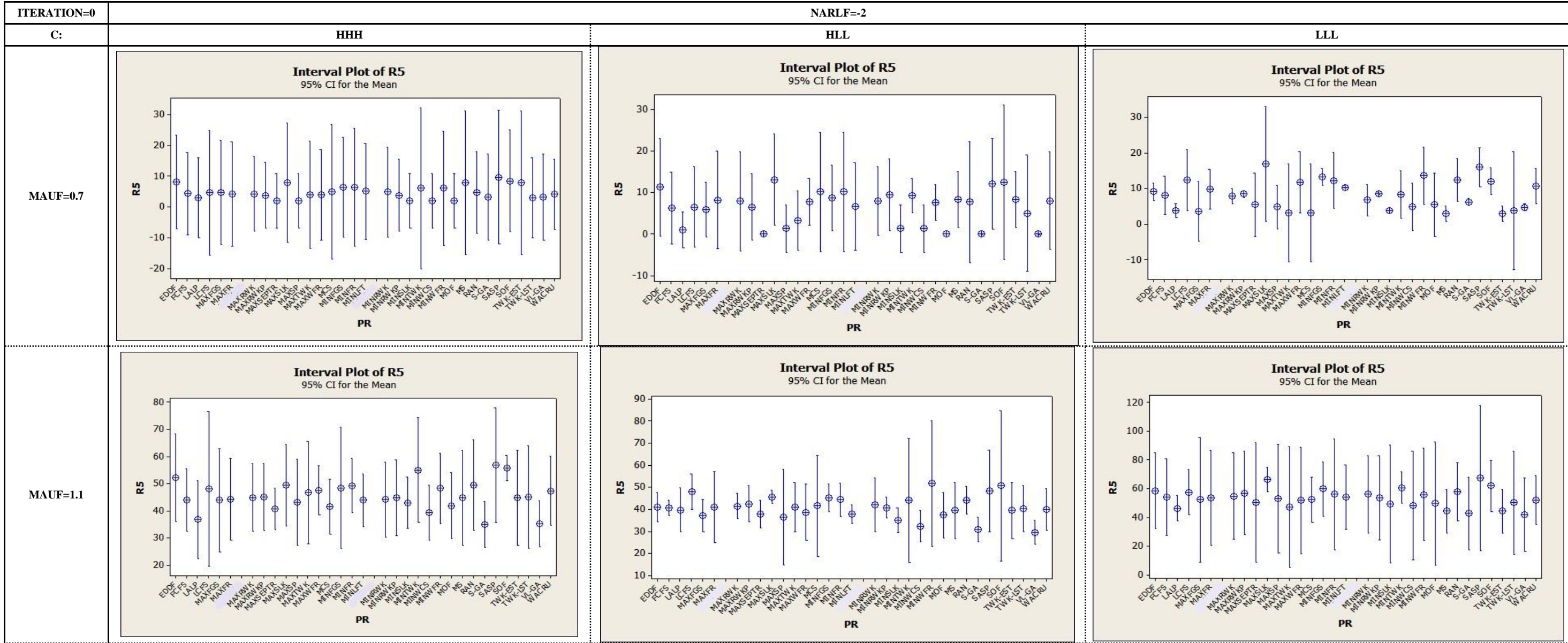


MAUF=1.1

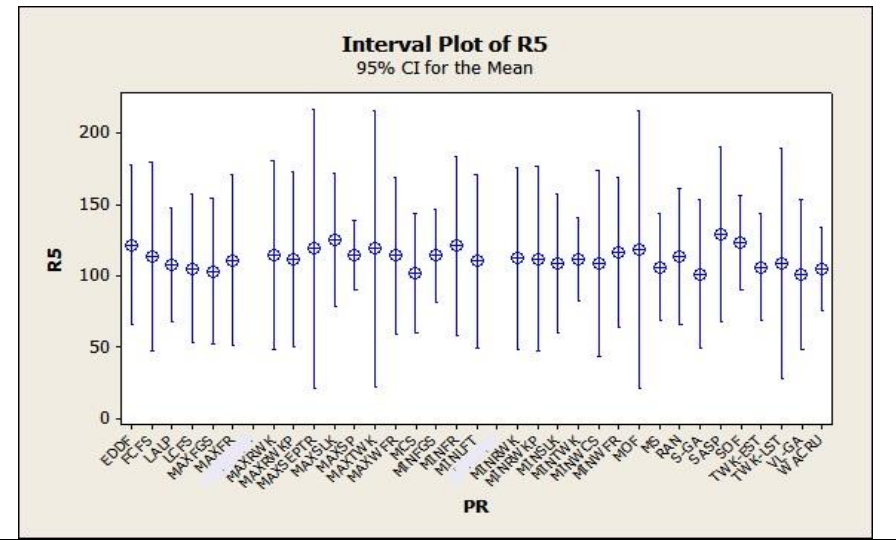
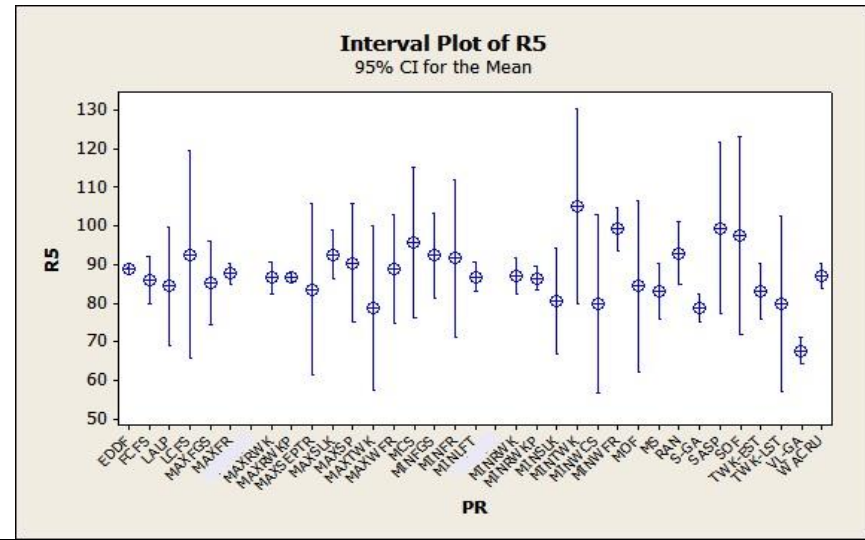
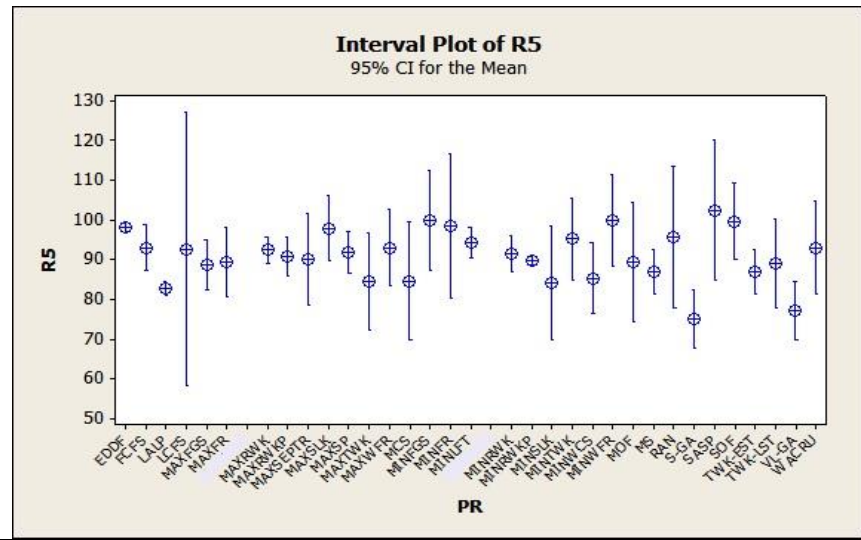


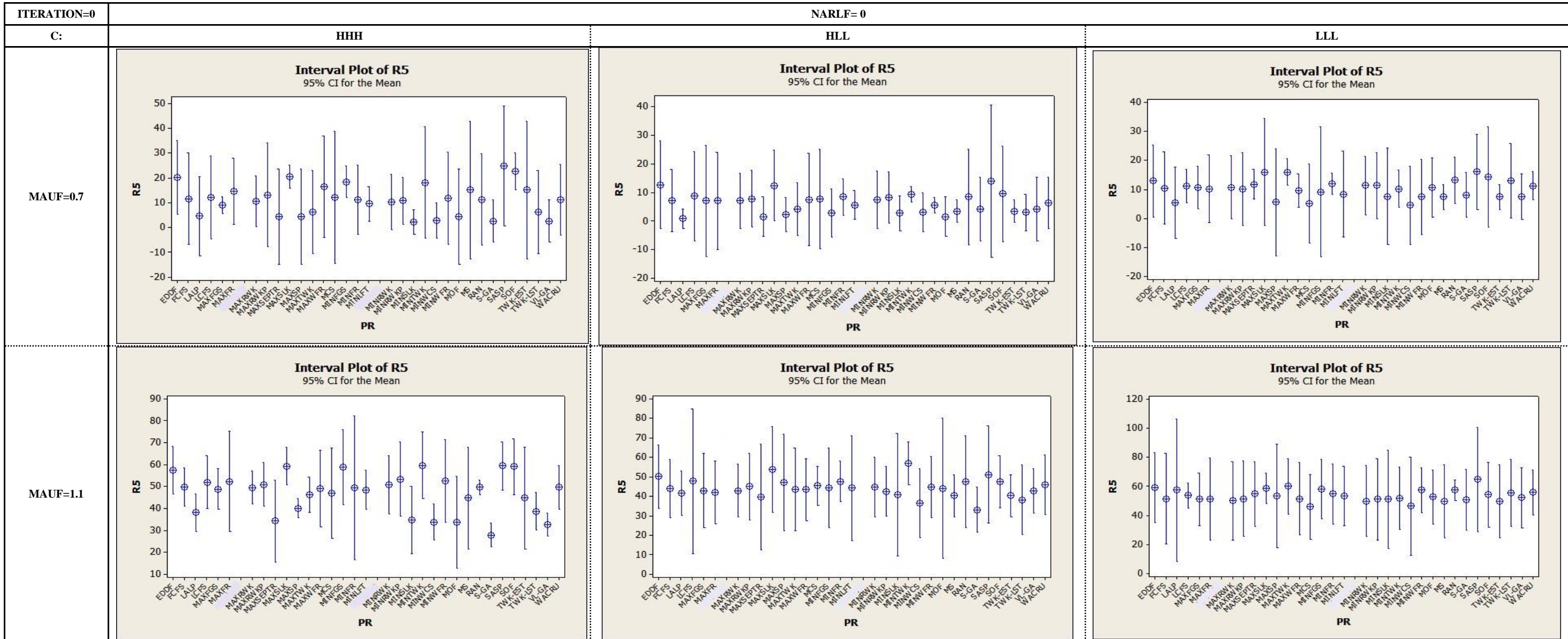
MAUF=1.5



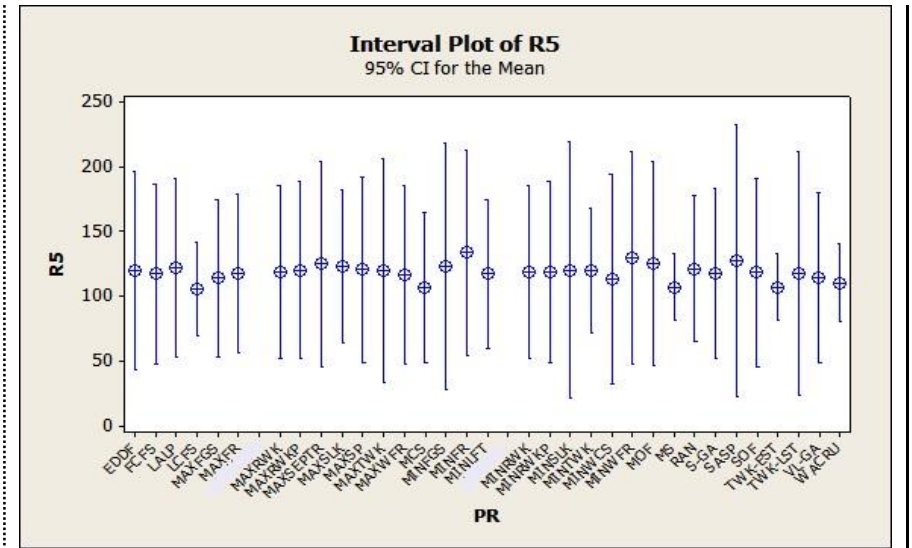
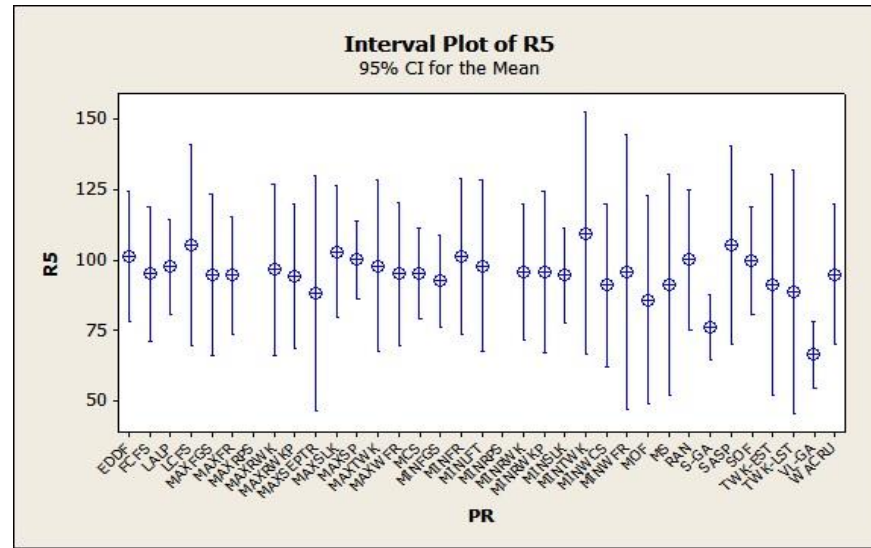
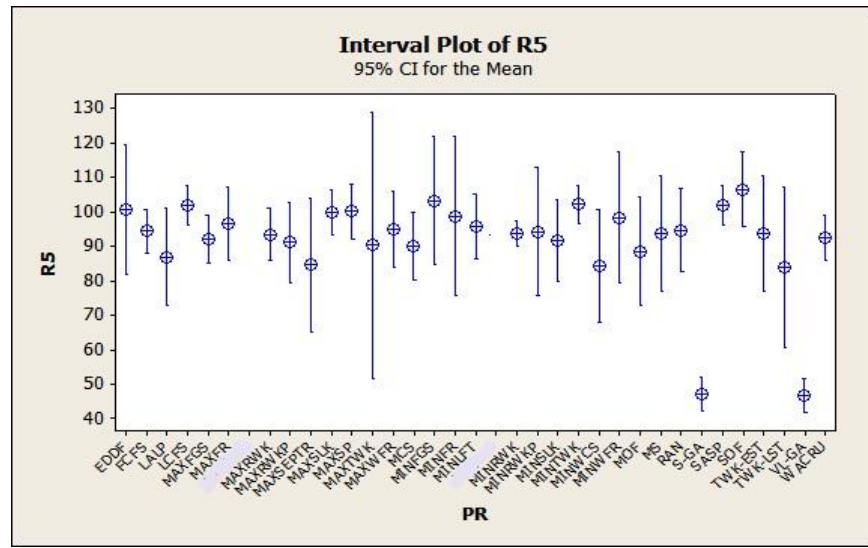


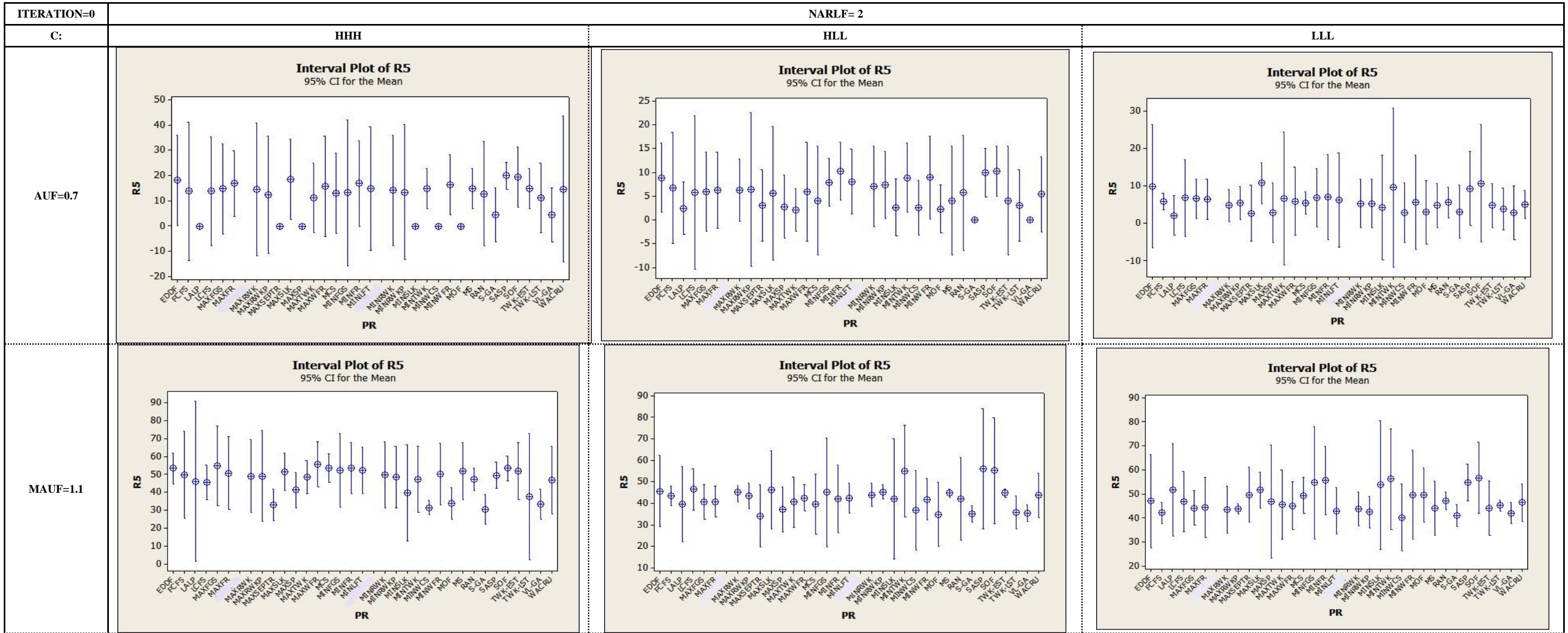
MAUF=1.5



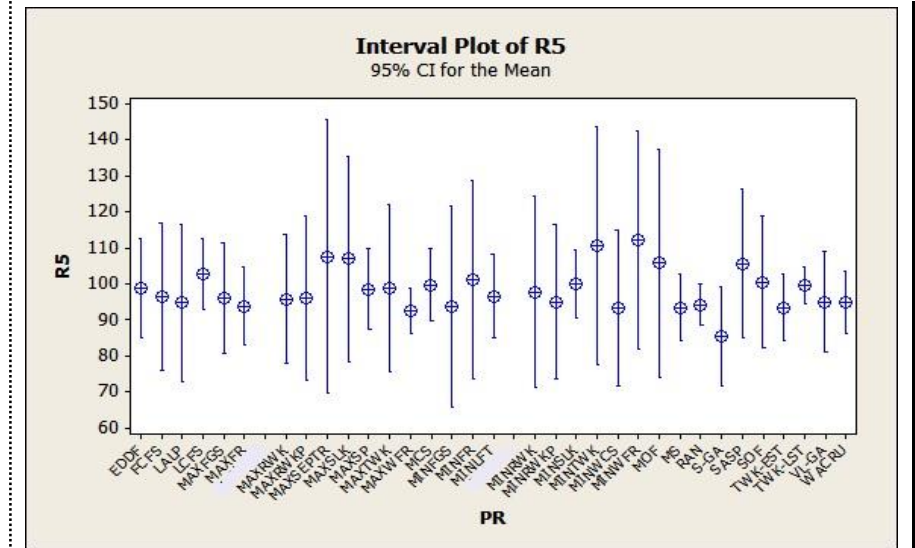
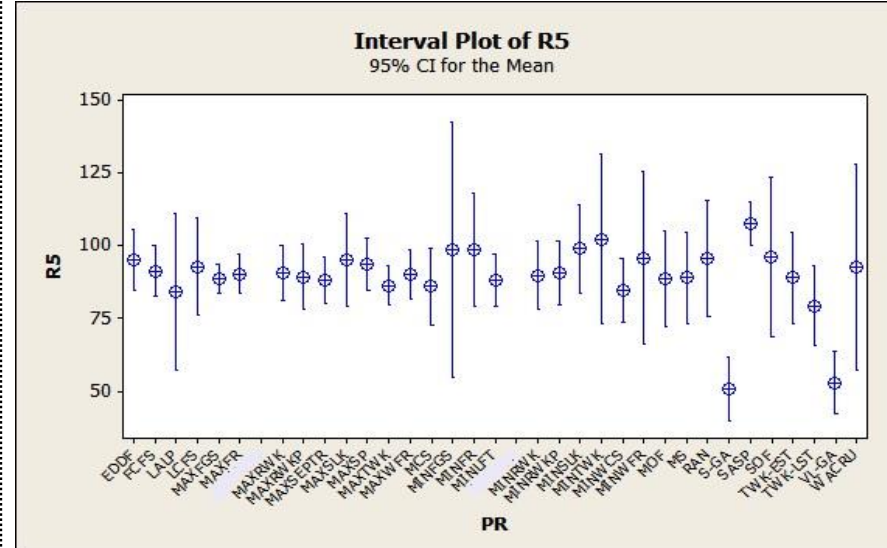
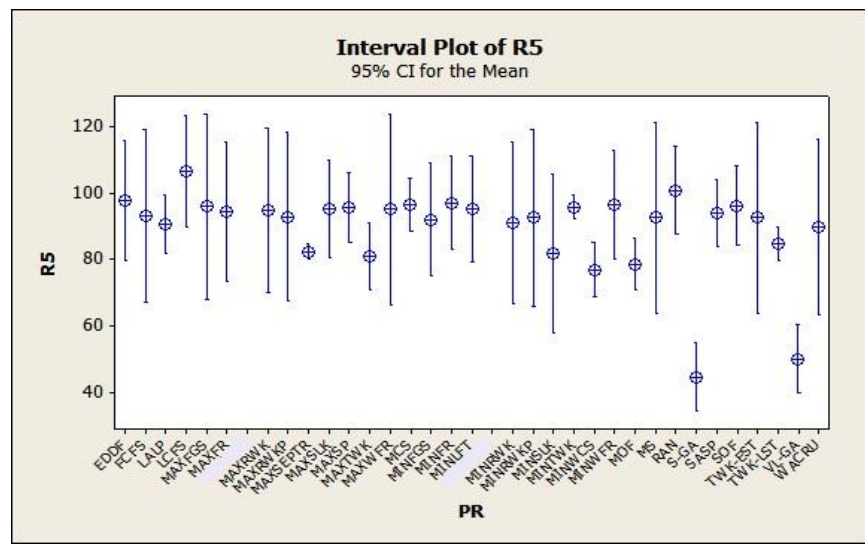


MAUF=1.5





MAUF=1.5



ITERATION=0.1

NARLF=-2

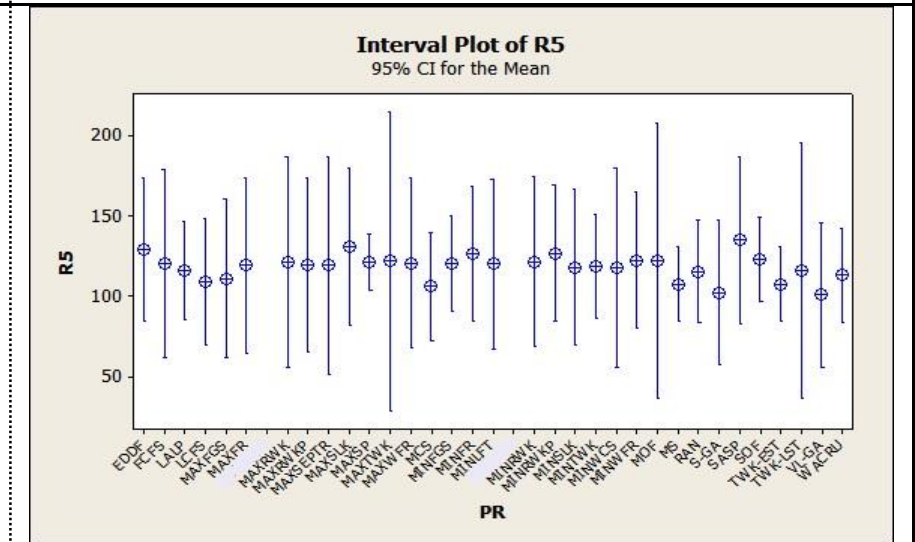
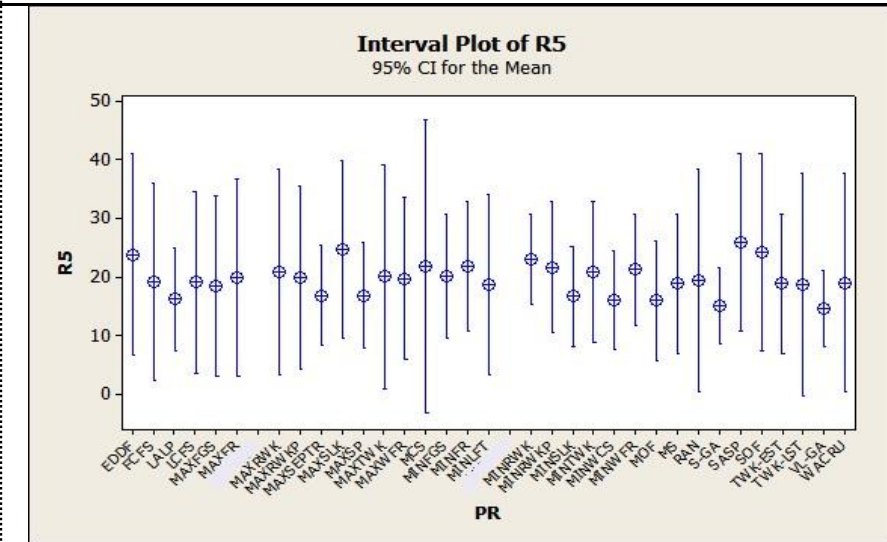
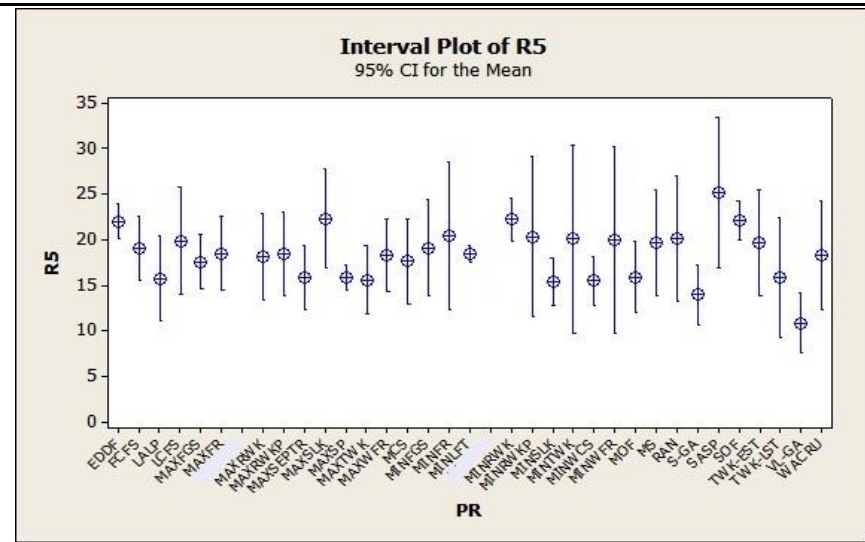
C:

HHH

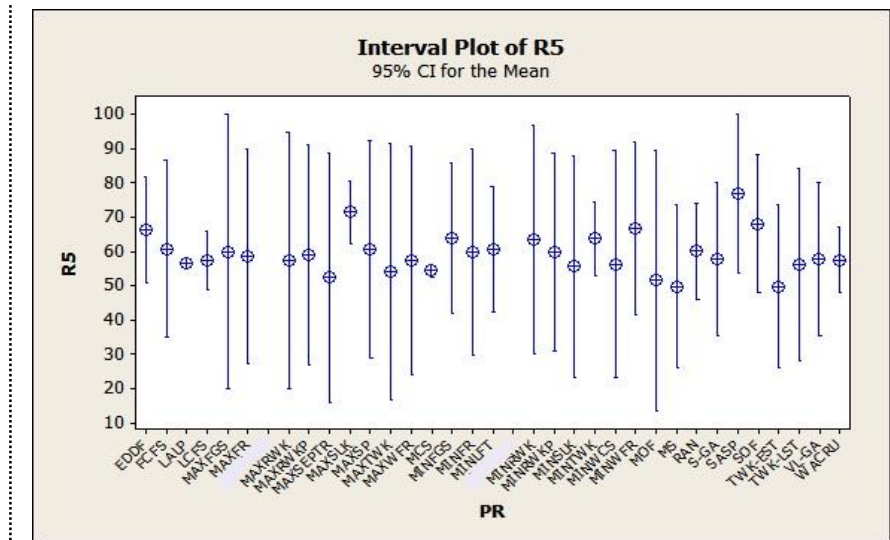
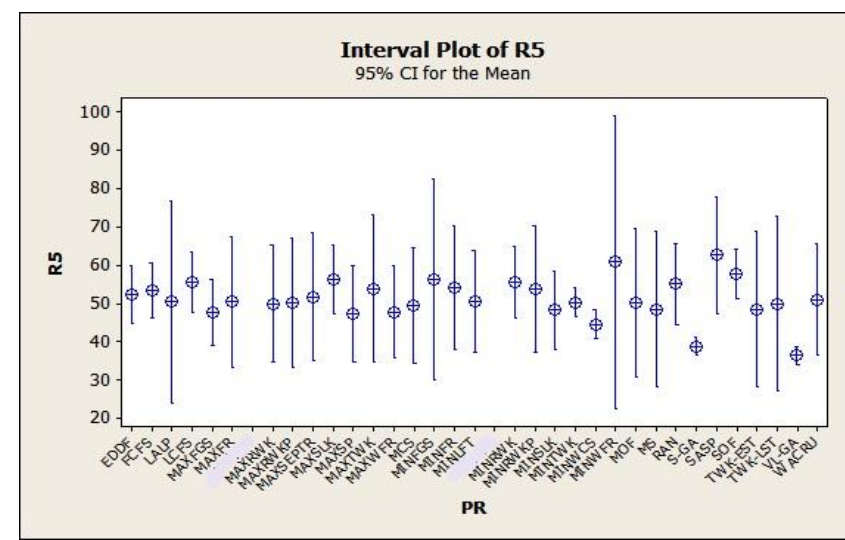
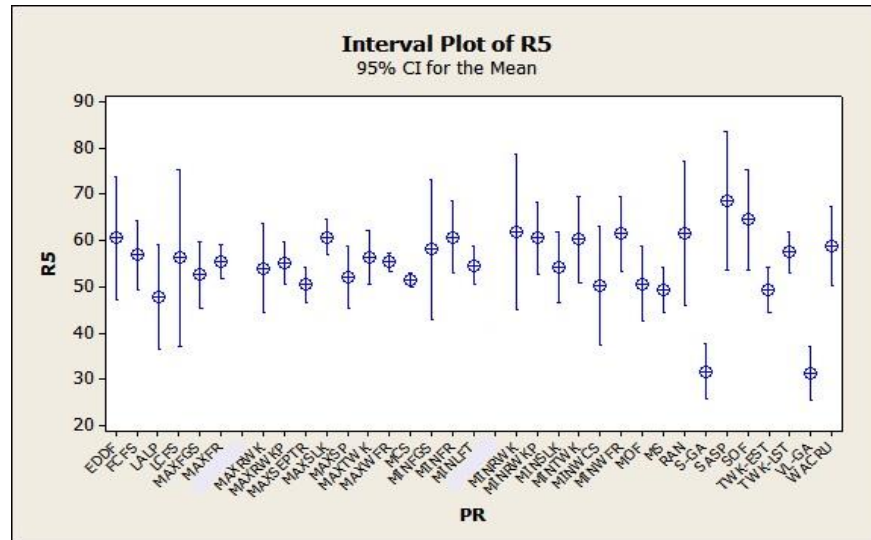
HLL

LLL

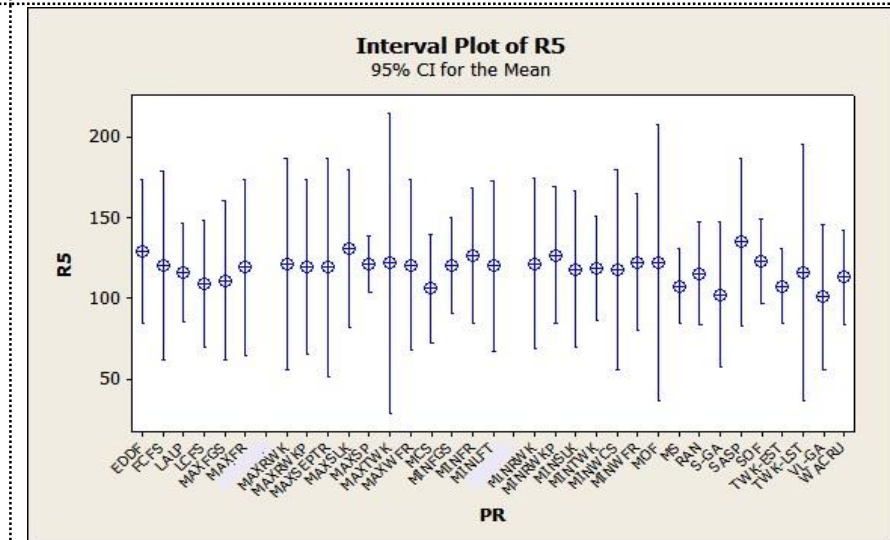
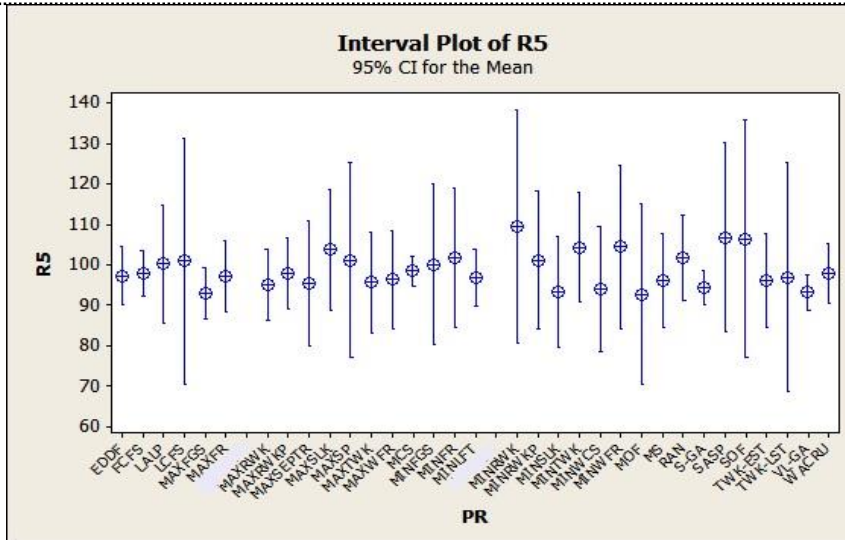
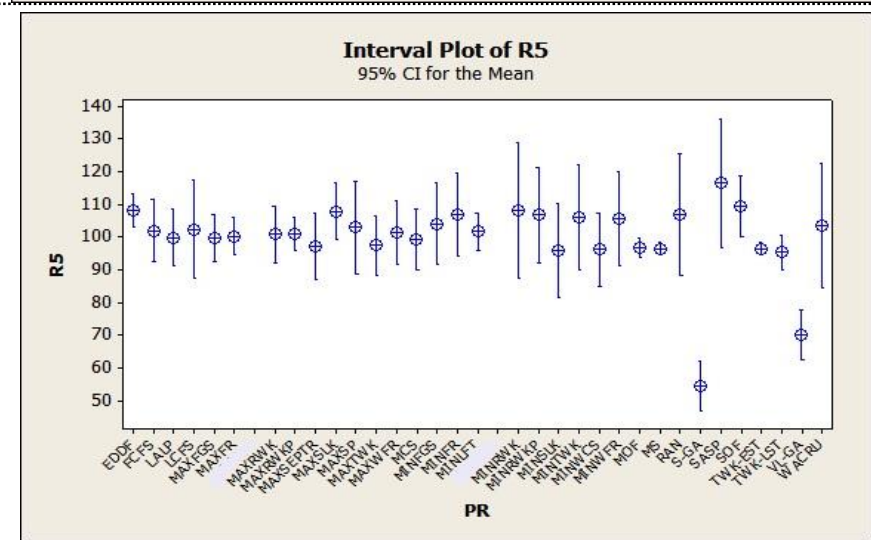
MAUF=0.7

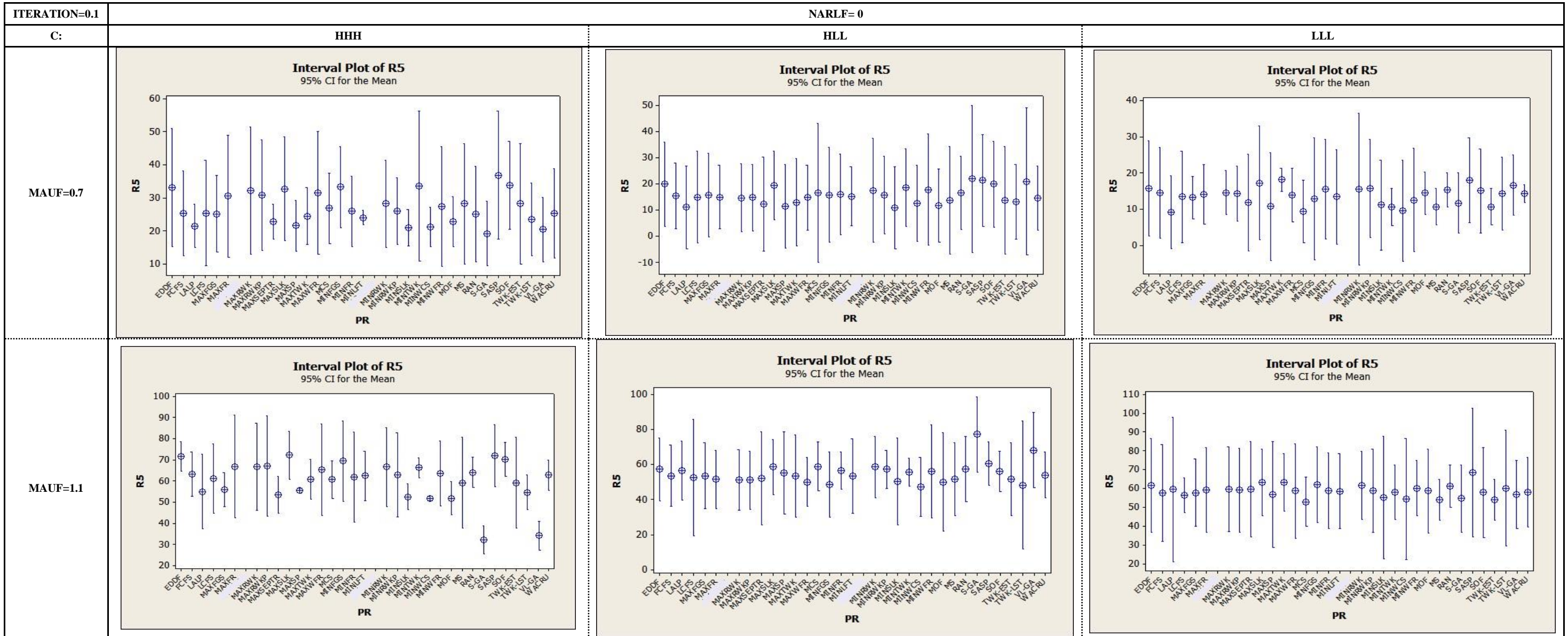


MAUF=1.1

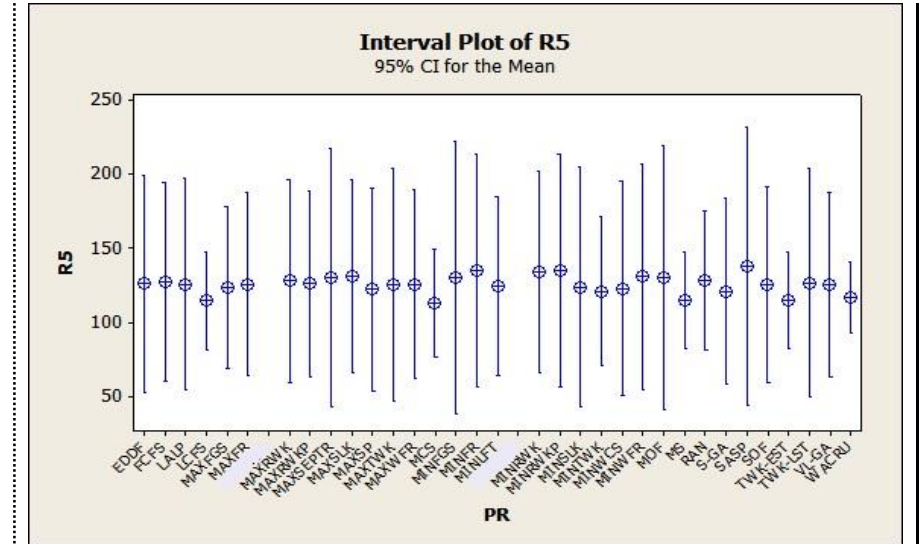
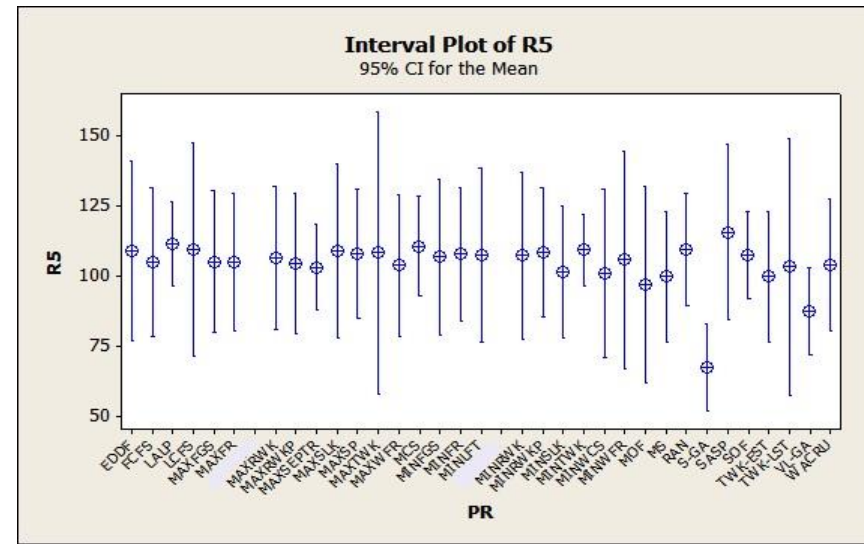
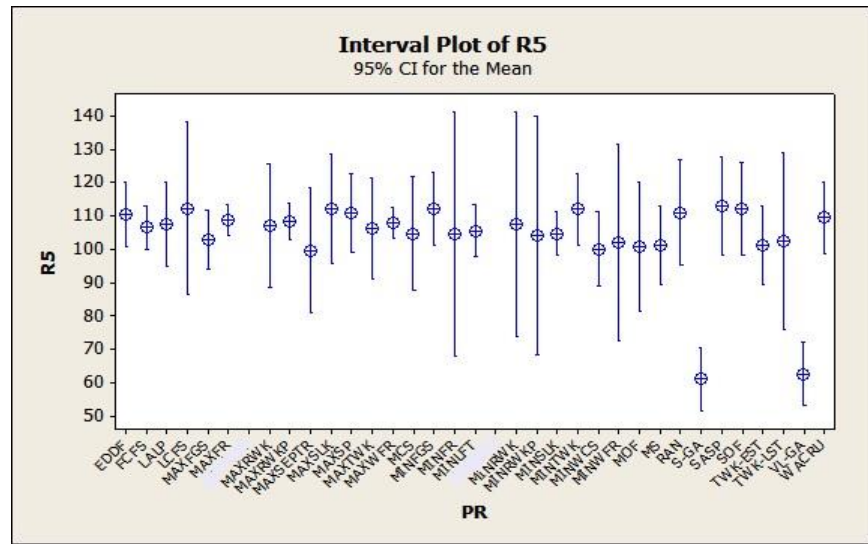


MAUF=1.5





MAUF=1.5



ITERATION=0.1

NARLF= 2

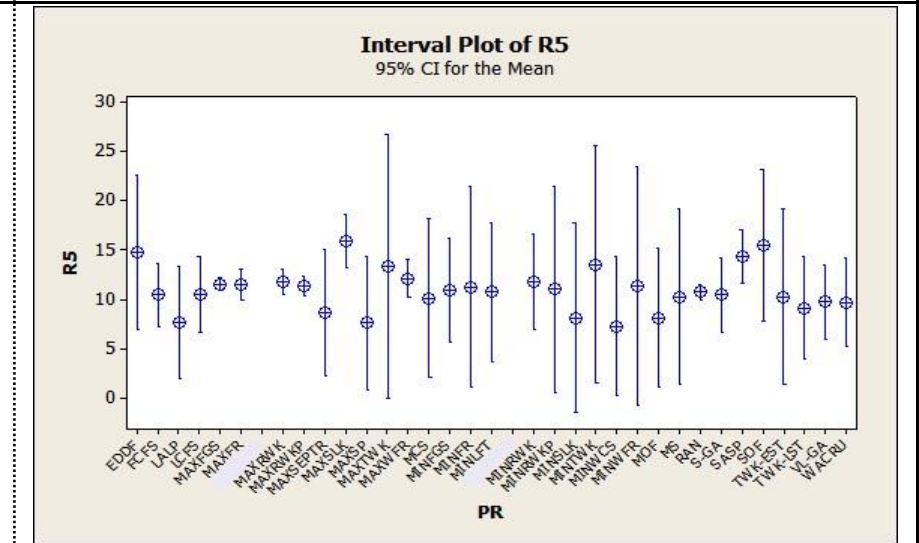
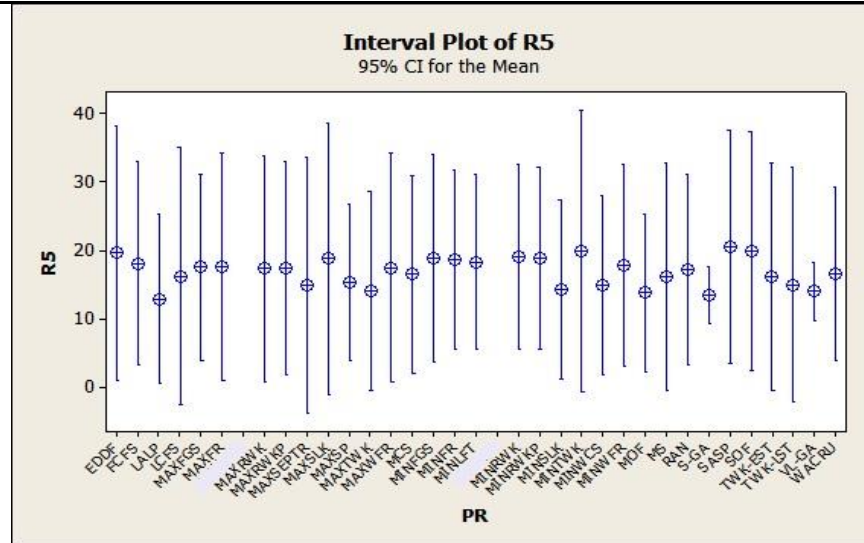
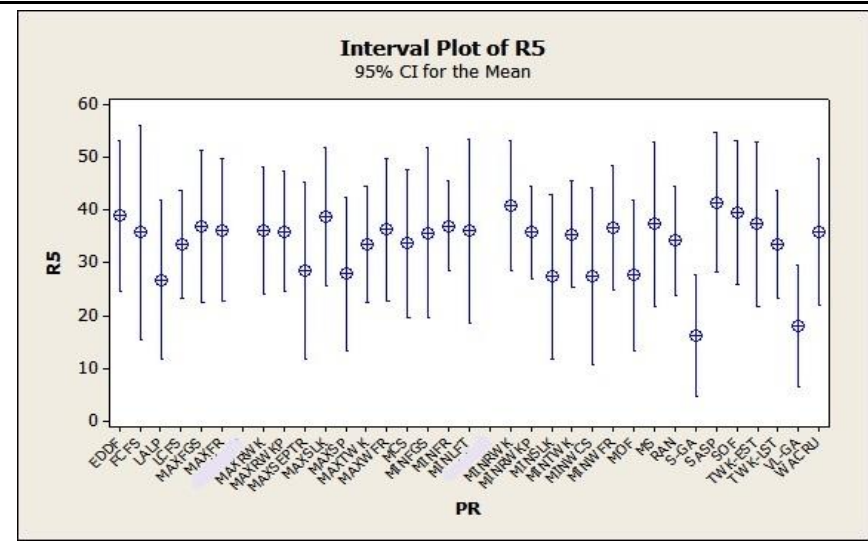
C:

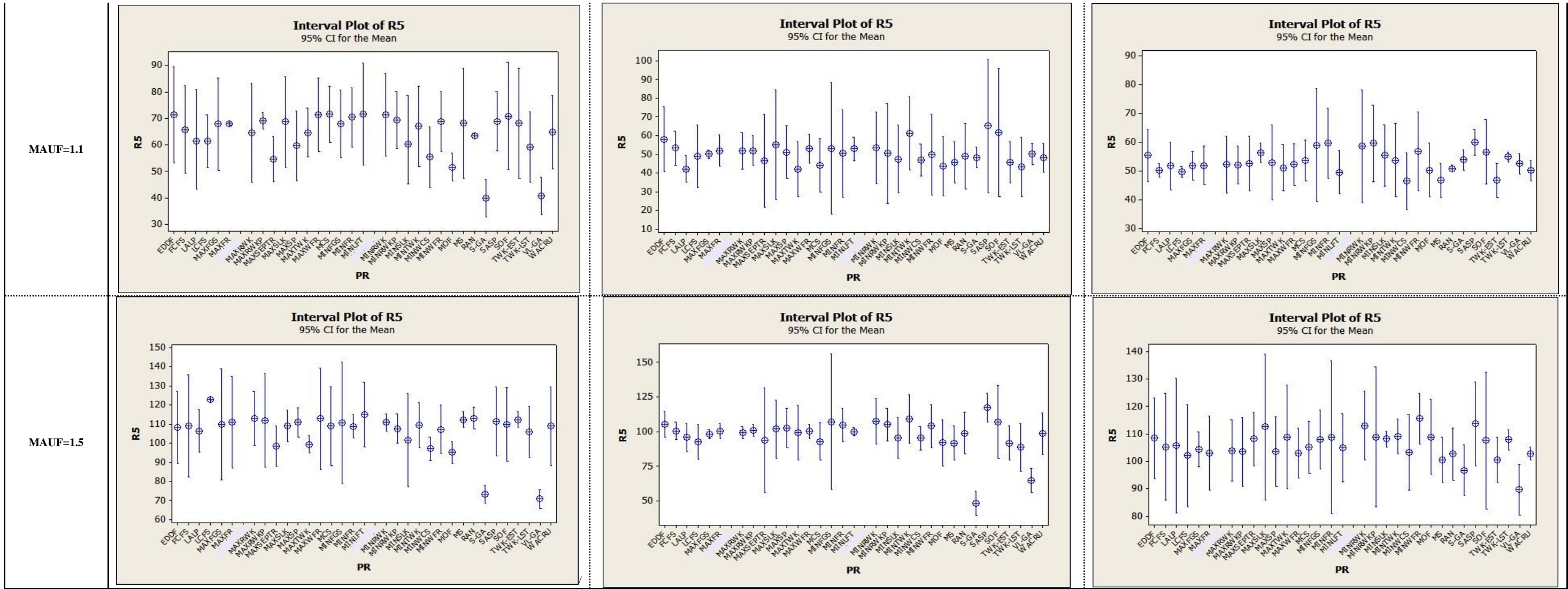
HHH

HLL

LLL

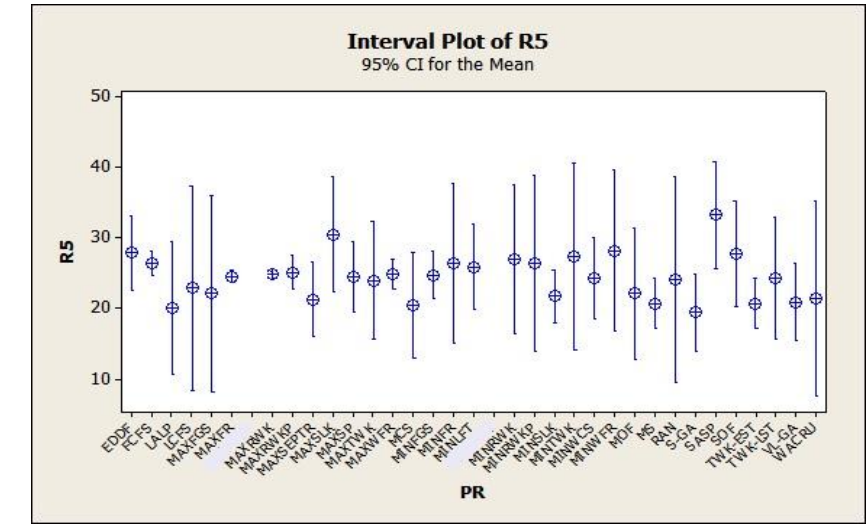
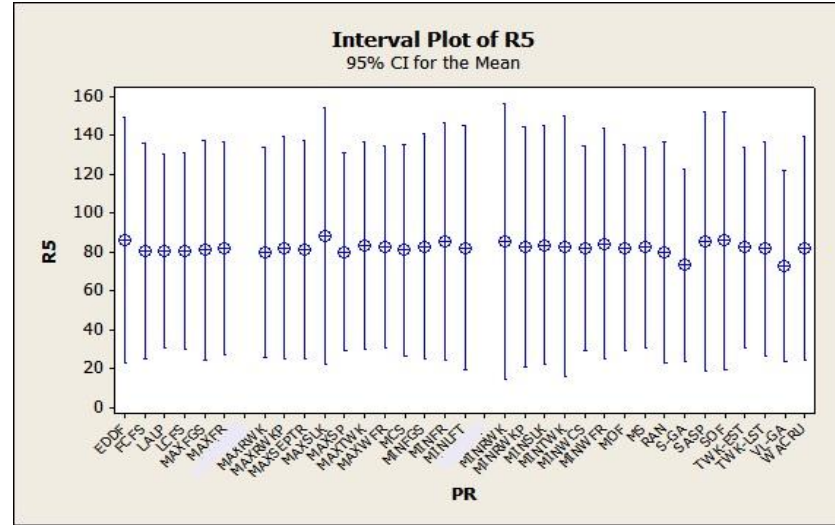
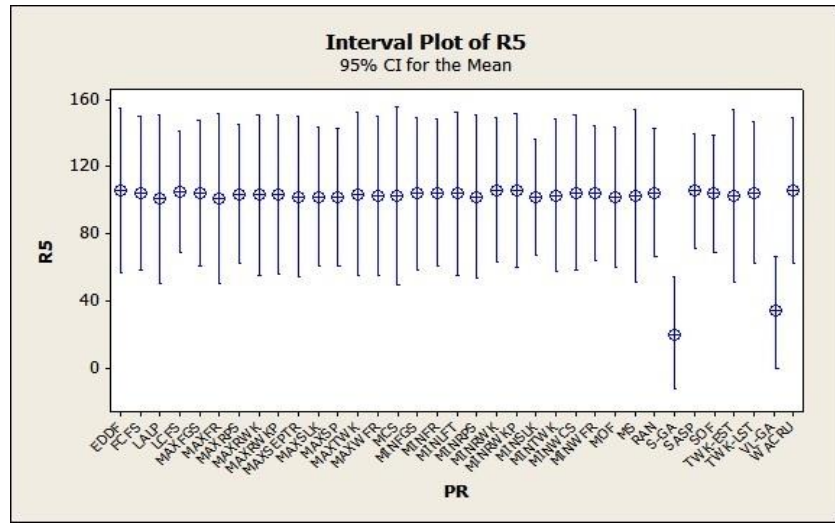
MAUF=0.7



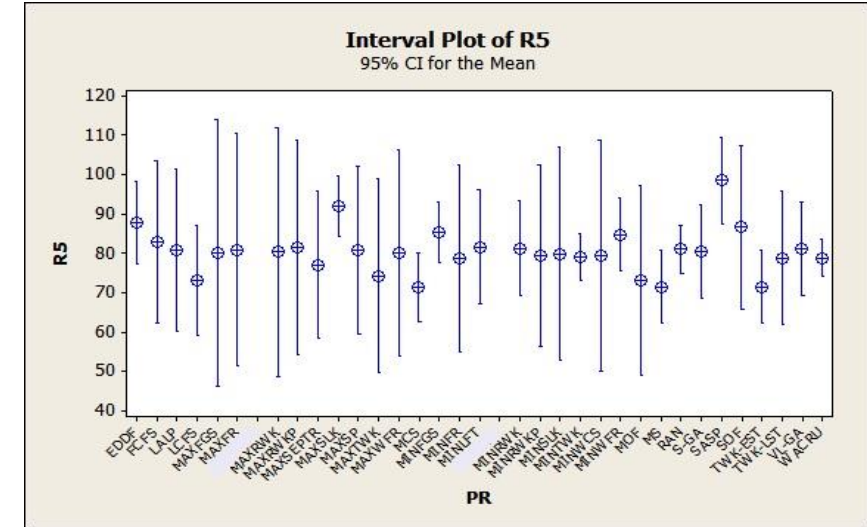
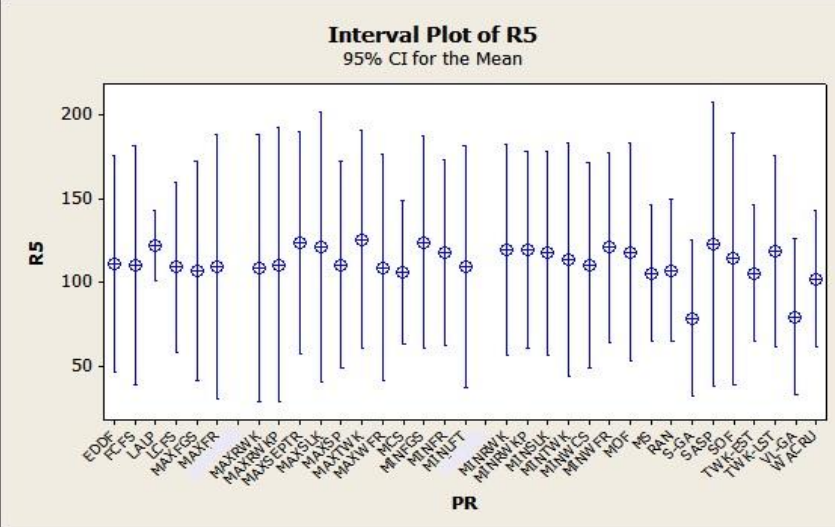
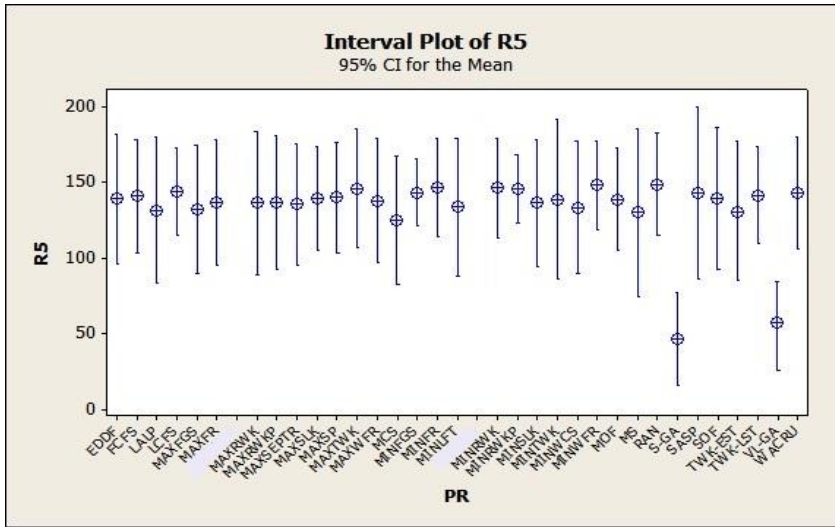


ITERATION=0.25	NARLF=-2		
C:	HHH	HLL	LLL

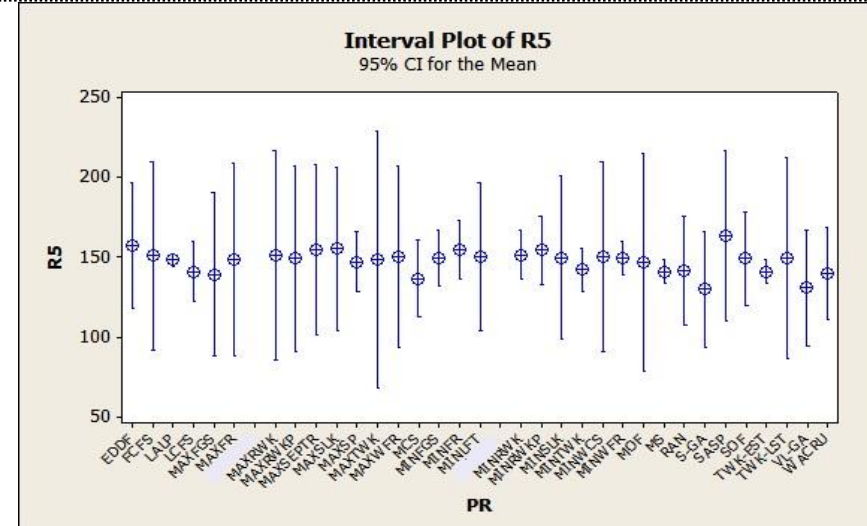
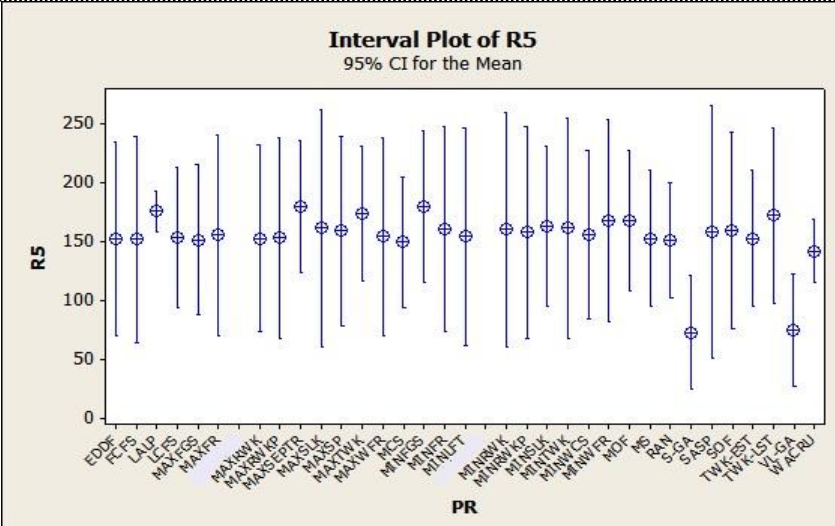
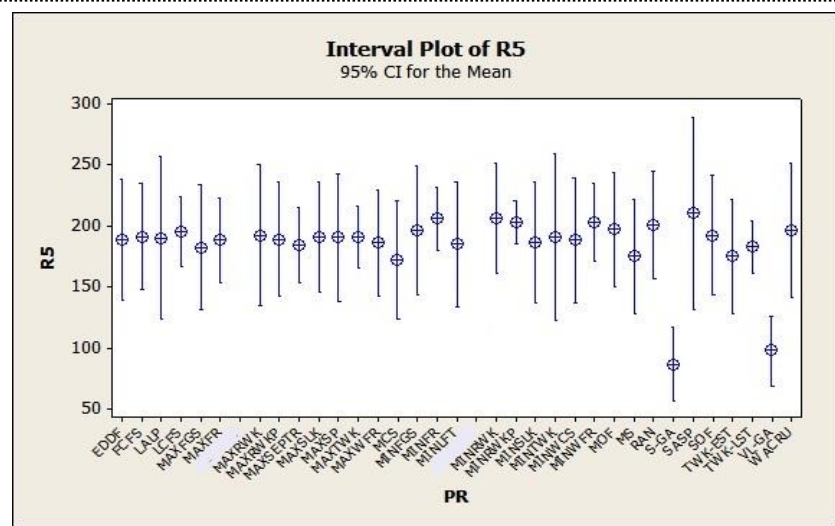
MAUF=0.7

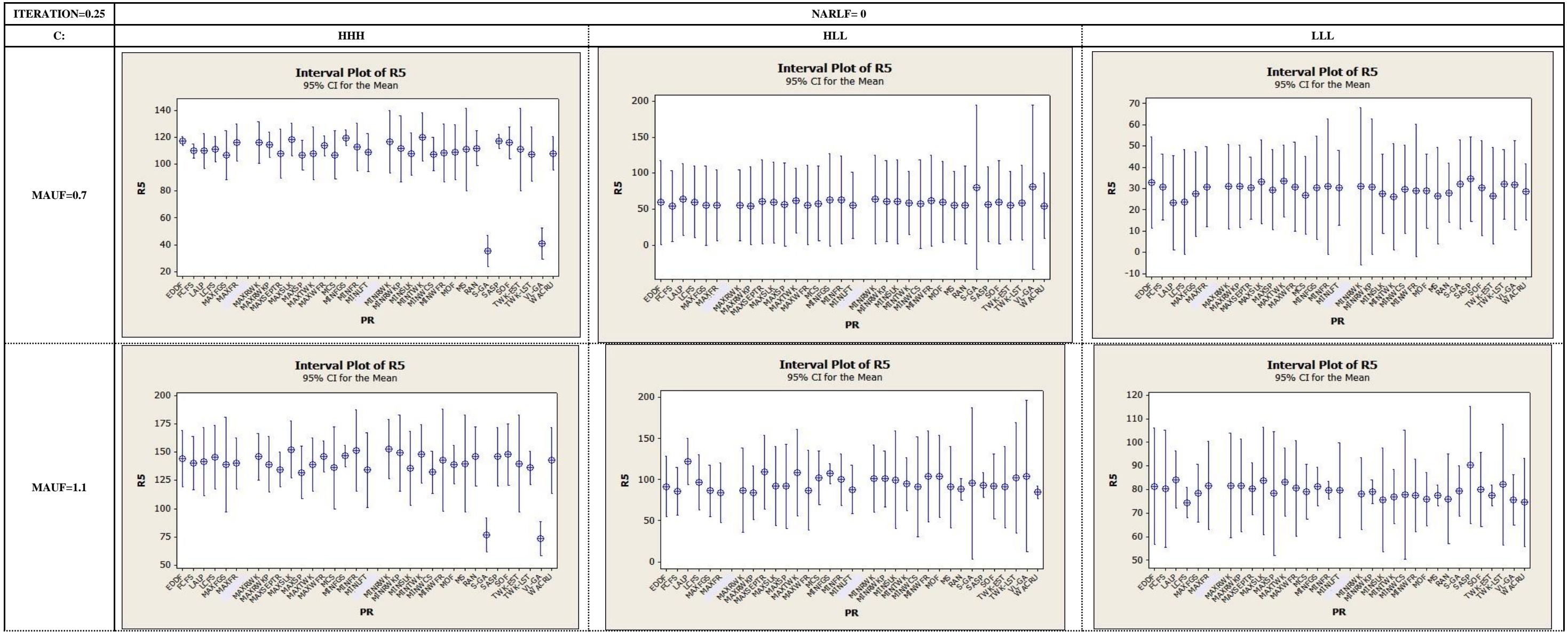


MAUF=1.1

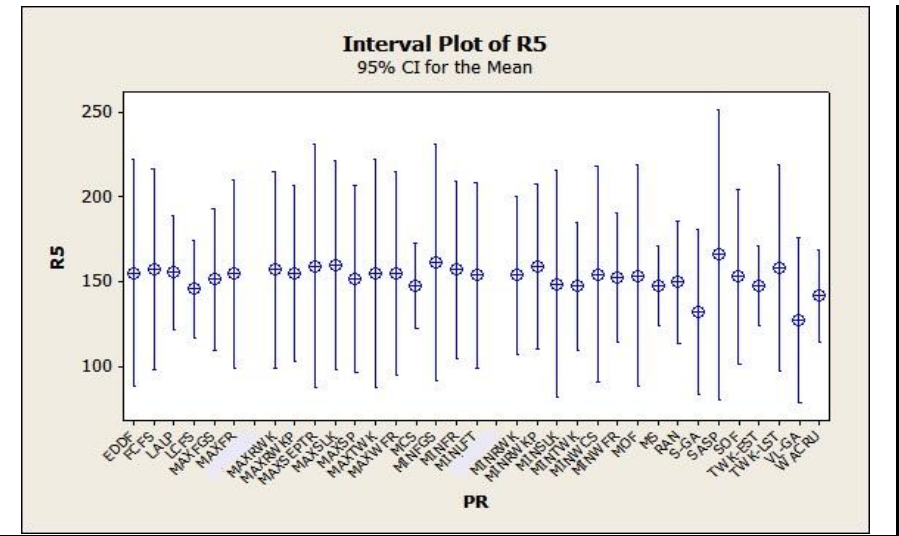
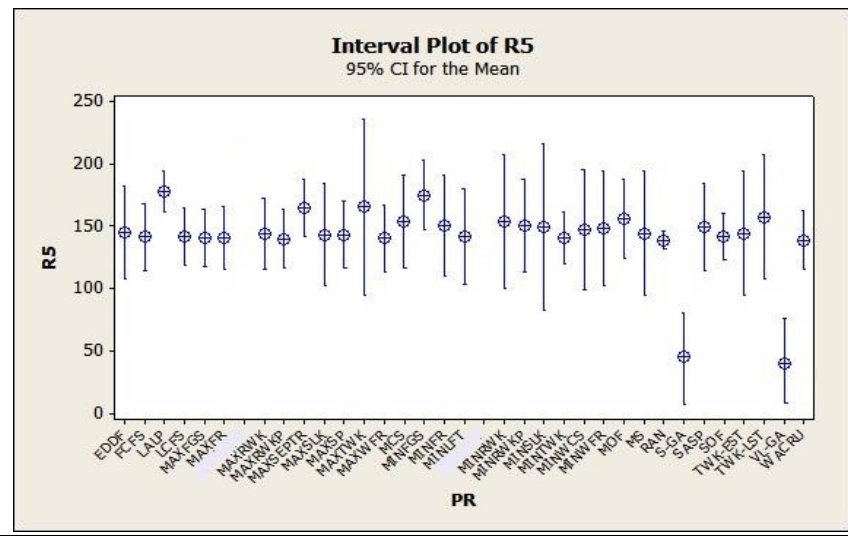
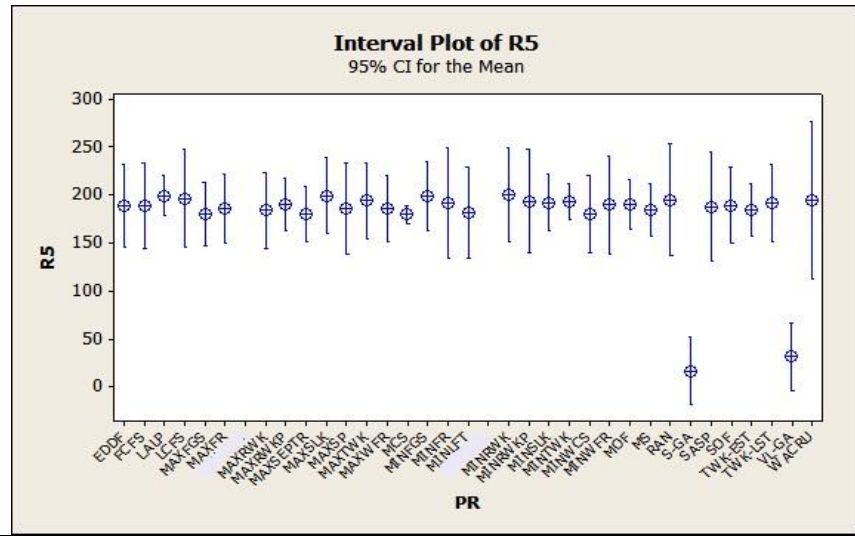


MAUF=1.5





MAUF=1.5



ITERATION=0.25

NARLF= 2

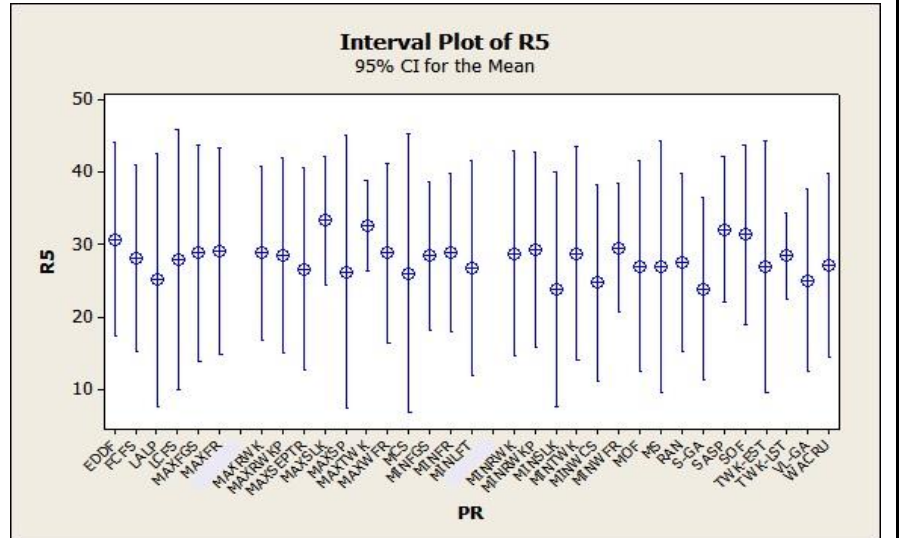
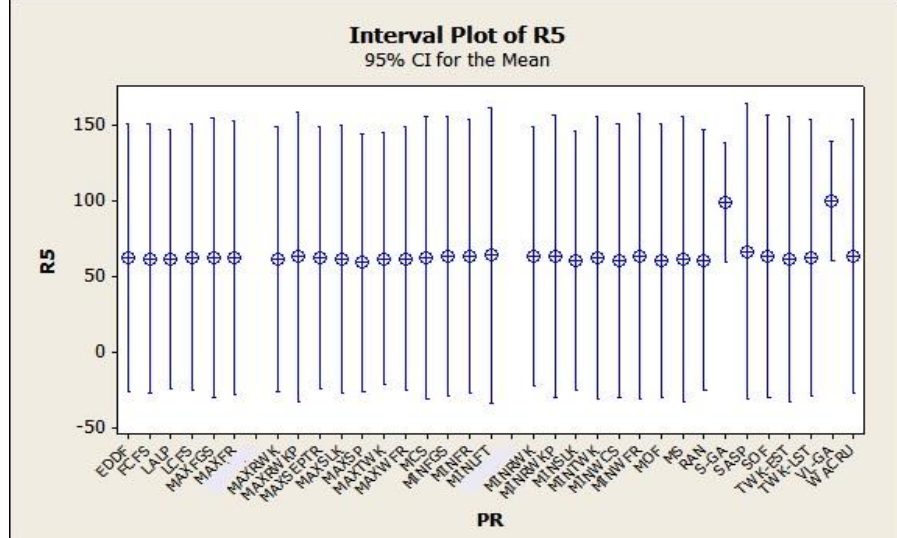
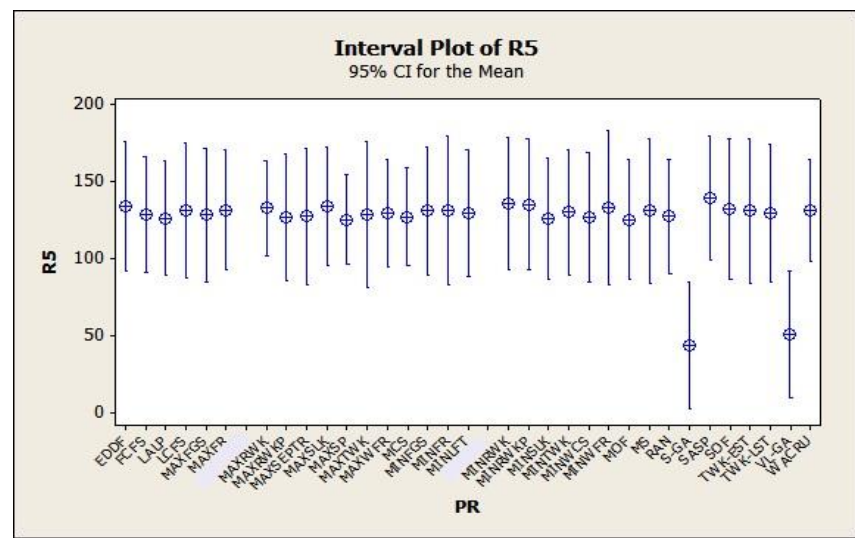
C:

HHH

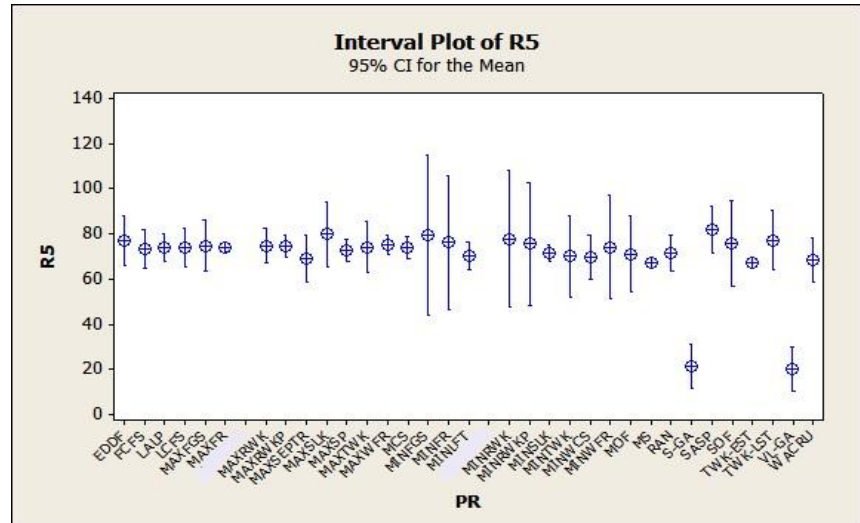
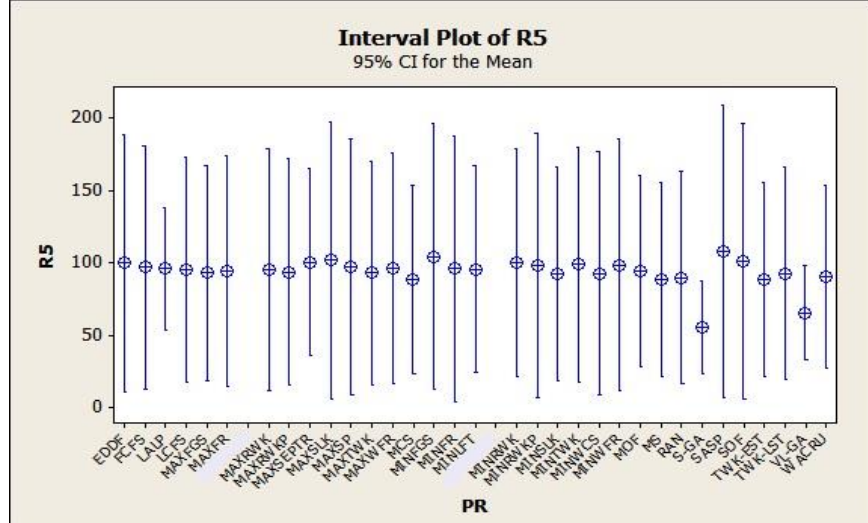
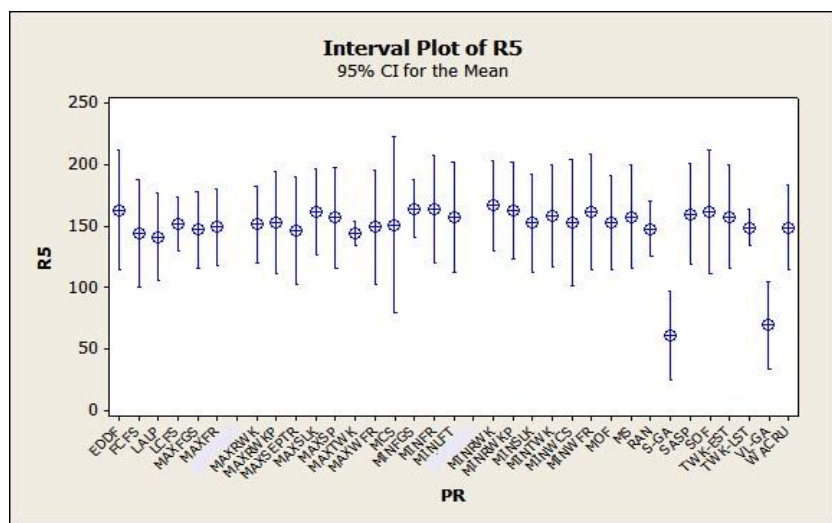
HLL

LLL

MAUF=0.7



MAUF=1.1



MAUF=1.5

