12-2011

Prognostics-Based Two-Operator Competition for Maintenance and Service Part Logistics

Faranak Fathi Aghdam
faranak.fathi@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Faranak Fathi Aghdam entitled "Prognostics-Based Two-Operator Competition for Maintenance and Service Part Logistics." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Industrial Engineering.

Haitao Liao, Major Professor

We have read this thesis and recommend its acceptance:

Xueping Li, Joseph Wilck

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Prognostics-Based Two-Operator Competition for Maintenance and Service Part Logistics

A thesis

Presented for the

Master of Science

Degree

University of Tennessee, Knoxville

Faranak Fathi Aghdam

December 2011

DEDICATION

I dedicate my thesis to my God and Lord, to my parents and my fiancé for their understanding, encouragement and support. I love you.

ACKNOWLEDGEMENTS

It is with great pleasure that I thank all of the people that helped me get to this point.

I would like to express my sincere gratitude to my advisor, **Dr. Haitao Liao**, for his guidance into the world of academic research and the exciting world of game theory. I sincerely thank him for giving me the opportunity to be part of his research group and for his persistent support and understanding.

It is with an overwhelming sense of gratitude that I thank my reading committee: Dr. Xueping Li and Dr. Joseph Wilck.

Words fail me to express my appreciation to the love of my life: my fiancé: Hamed. Thank you for all your guidance, dedication, patience, love and endless support during this time. Although we were far apart during this time, I've felt your pure love in my heart every moment.

Last, but certainly not the least, I would like to acknowledge the commitment, sacrifice and support of my parents, who have always motivated me.

ABSTRACT

Prognostics and timely maintenance of components are critical to the continuing operation of a system. By implementing prognostics, it is possible for the operator to maintain the system in the right place at the right time. However, the complexity in the real world makes near-zero downtime difficult to achieve partly because of a possible shortage of required service parts. This is realistic and quite important in maintenance practice. To coordinate with a prognostics-based maintenance schedule, the operator must decide when to order service parts and how to compete with other operators who also need the same parts. This research addresses a joint decision-making approach that assists two operators in making proactive maintenance decisions and strategically competing for a service part that both operators rely on for their individual operations. To this end, a maintenance policy involving competition in service part procurement is developed based on the Stackelberg game-theoretic model. Variations of the policy are formulated for three different scenarios and solved via either backward induction or genetic algorithm methods. Unlike the first two scenarios, the possibility for either of the operators being the leader in such competitions is considered in the third scenario. A numerical study on wind turbine operation is provided to demonstrate the use of the joint decision-making approach in maintenance and service part logistics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

1. Introduction and General Information

1.1 Preventive maintenance and replacement scheduling

It is well documented that managing maintenance activities in a proactive rather than a reactive manner results in lower operation and maintenance costs and superior asset performance. This is easy to say but difficult to do in actual practice. As components in a system are aging with time, preventive maintenance (PM) that prevents failures may be economically justified. Unlike corrective maintenance (CM) involving repair or replacement of failed components, the intention of performing PM is to restore system reliability by maintaining the aged components or replacing them before they actually fail. Among many PM strategies, preventive replacement can be implemented for non-repairable components, which can be classified into two categories: time-based replacement and condition-based replacement. There are two types of time-based replacement schemes, i.e., age replacement and block replacement [1]. In age replacement, a scheduled replacement occurs whenever an operating unit reaches a certain age *T*, while for block replacement all operating units are replaced at regular time intervals regardless of the actual age of individual units. For condition-based replacement, an action to be taken on a single unit after each inspection (or upon condition monitoring via an in-situ sensor) is determined based on the unit's current state.

Possible actions are (1) replacing the unit right away, (2) determining the next service time to replace the unit or (3) no action.

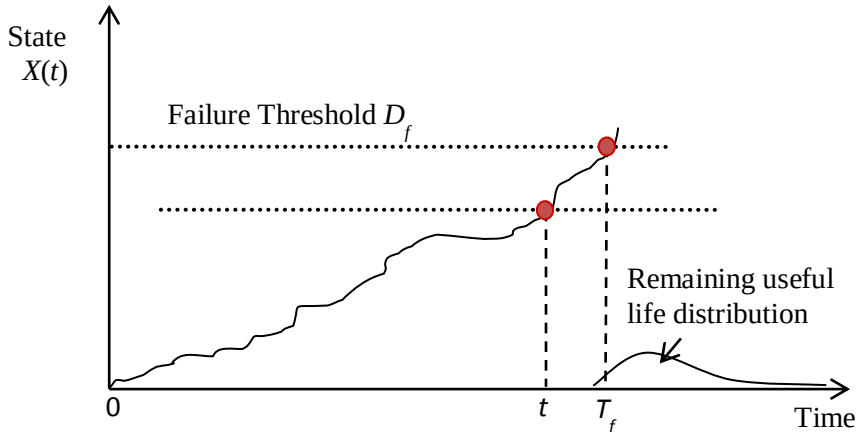

Figure 1. Degradation process and remaining useful life distribution

Fig. 1 illustrates the methodology of condition-based replacement. As the unit's degradation process $\{X(t), t \geq 0\}$ evolves, the remaining useful life (RUL) of the unit can be predicted based on a stochastic model for $\{X(t), t \geq 0\}$ with unit-specific parameters. Let D_f be the failure threshold and $T_f = inf\{t|: X(t) \geq D_f, t > 0\}$ be the actual failure time of the unit. The distribution of the unit's $RUL = T_f - t > 0$ at time t, can be expressed as $P_r(RUL < u)$ where $u > 0$. One of the possible actions will be determined based on the predicted RUL at present. When the unit is decided to be replaced either right away or for the next service time, it would be straightforward to do so if a service part is currently in hand or will be available prior to the next service time. However, a shortage of a required service part often makes timely replacement difficult to achieve. In addition to inherent delays, such as replenishment lead time, operators sometimes are

forced to compete with others for a service part that all of them rely on for their individual operations. This makes the availability of service parts questionable to each operator.

This thesis studies a joint decision-making mechanism for proactive replacement and competition in service part procurement between two operators. A specific type of component is considered, and the joint decision is made based on the predicted values of RUL of the units being used by the operators. Considering the limited availability of service part and their affordable prices and losses, the two operators must determine the best times for replacing their units and for ordering the service part with a possible competition with each other. Three different scenarios based on the Stackelberg game-theoretic model are formulated in this thesis. The backward induction method for solving Stackelberg games is used in finding the optimal preventive replacement and ordering times. In addition, a genetic algorithm (GA) is utilized for a multi objective case.

1.2 Research Contributions

In this thesis the following contributions are made:

1. In this work the competition between two operators on optimizing their maintenance policies based on minimizing the costs is modeled by a new optimization model.
2. The relationship between the operators is modeled based on a leader-follower game theoretic model.

3. A multi objective model is developed based on a set of assumptions. This model is optimized via either backward induction or genetic algorithms.
4. Finally, a numerical example based on real numbers from wind turbine gearbox reliability databases will be considered as the application of the developed model.

1.3 Outline of the study

The remainder of this thesis is organized as follows:

In chapter 2, a comprehensive literature review of various models and algorithms in spare and service part inventory control and PM optimization problems is presented.

Chapter 3 clearly provides the problem description and mathematical formulations of the joint decision-making models and also addresses the proposed optimization methods for solving three scenarios.

In chapter 4, a numerical example on wind turbine operation is provided to demonstrate the use of the proposed models and their solution methods. Effect of competition is assessed in this section.

Finally, chapter 5 gives concluding remarks and recommendation for future work.

CHAPTER II

2. Literature Review

In this chapter, the studies in the literature that are related with this study are summarized. The subjects of the papers and proposed models are explained for each of them.

Under sections 2.1, 2.2, 2.3 and 2.4 literature related to reliability and maintainability optimization models, service part inventory control, multi objective optimization models and game theory are discussed. In part 2.5 by discussing the reliability issues of wind turbines, the reason of using a wind turbine problem as the numerical example in chapter 4 will be cleared.

2.1 Reliability and maintainability

IEEE defines reliability as:

"The ability of a system or component to perform its required functions under stated conditions for a specified period of time".

The study of maintenance policies is one of the most important areas of interest in reliability field. The two different criteria that are known in the optimization of replacement intervals are PM and CM. PM is defined as the activity undertaken regularly at pre-selected intervals while the device is satisfactorily operating, to reduce or eliminate the accumulated deterioration [18].

A performance criterion for maintenance systems is minimizing the total cost of maintenance, which includes PM cost, CM cost or cost of failure. (fig2)

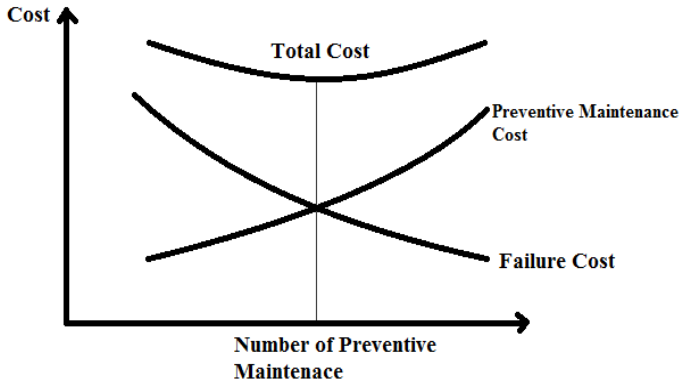

Figure 2. Total Maintenance Cost

PM has been extensively investigated in the reliability field. In terms of mathematical modeling, most PM models are focused on the minimum cost, economic system lifetime, and highest system availability. Chen and Feldman [19] presented a repair/replacement problem based on age-replacement policy. Panagiotidou and Tagaras [20] presented an economic model for the optimization of PM in a production process with two quality states (in-control and out-of-control).They found the optimum time to perform PM based on the actual (observable) state of the process. Yeh et al. [21] analyzed the effects of a free-repair warranty on the optimal periodic replacement policy for both a warranted and non-warranted repairable products by optimizing the long-run cost rate. Dehayem Nodem et al. [22] presented a hierarchical decision-making approach in production and repair/replacement planning with imperfect repairs under uncertainties to minimize the total costs over an infinite planning horizon. A semi-Markov decision model was used to

determine the optimal repair and replacement policy, and the production rate was determined based on the obtained repair and replacement policy. Berg [23] extended existing maintenance policies that are based only on the present repair cost by considering the future costs. Essentially, the repair and replacement policy is analyzed and optimized using the marginal cost analysis.

The integration of preventive replacement and service part logistics has also been studied in many papers. Zohrul Kabir and Al-Olayan [24] presented a simulation model that minimizes the total cost of replacement and inventory by incorporating both age-replacement policy and continuous review of stocking inventory policy. Vaughan [25] developed a stochastic dynamic programming model to characterize the ordering policy due to regularly scheduled PM and random failure of units in service. Wang et al. [26] optimized the presented simulation model for deteriorating systems which combines the condition-based replacement policy with periodic inspections and the base stock inventory policy. Wang [27] presented a joint optimization model for both the inventory control of the spare parts and the PM inspection interval to find the optimum value for the order interval, PM interval and order quantity via dynamic programming. Liao et al. [28] introduces a condition-based availability limit policy which achieves the maximum availability of a system by optimally scheduling maintenance actions.

In order to optimize the maintenance policy for a component with deterioration and random failure rate, a linear programming model was proposed by Jayakumar and Asgarpoor [29]. In their model, they determined optimal mean times of minor and major PM actions based on maximizing the availability of the component.

Duarte et al. [30] considered a system with series component that have linearly increasing failure rate and constant improvement factor for imperfect maintenance. They presented a model and algorithm to optimize the interval of time between maintenance actions by considering the total cost and total downtime as the objective functions.

In another study, Tam et al. [31] presented three models to determine the optimal maintenance intervals for a multi component system under maintenance actions without considering the replacement actions. He considered three different models to minimize total cost subject to satisfying a required reliability, one that maximizes reliability at a given budget, and one that minimizes the expected total cost including expected breakdown outages cost and maintenance cost.

Another paper is by Shirmohammadi et al. [32] which developed an age based nonlinear optimization model to determine the optimal PM schedule for a single component system. They considered the cost per unit time as the objective function to find the optimal time between preventive replacements and the cut-off age. They utilized MAPLE to solve the optimization model.

2.1.1 Multi Objective Algorithm

Multi objective optimization involves trying to simultaneously optimize two or more objectives. In addition to single objective problems, multi objective PM optimization models have also been investigated. The problem usually has a number of constraints, which must be satisfied by any feasible solution.

Berrichi et al. [33] considered an algorithm based on bi-objective Ant Colony Optimization in handling both production and maintenance scheduling problem to simultaneously determine the best assignment of production tasks to machines as well as PM (PM) periods of the production system. Moradi et al. [34] investigated integrated flexible job shop problem with PM activities under the multi objective optimization approaches. Two decisions are made at the same time: finding the appropriate assignment of *n* jobs on *m* machines in order to minimize the makespan and the best time to execute PM to minimize the system unavailability. Quan et al. [35] presented a novel evolutionary algorithm to solve a PM scheduling problem, which is formulated as a multi objective problem.

In a paper by Herabat [36], they developed a multi objective optimization model to support the multi year decision making process of the highway maintenance management in Thailand. PM is focused in this research since it helps prolong the life of the infrastructures. This study selects the flexible pavements in the Pathumthani province to be the study area. Both single- and multi objective optimization models are developed for a multi year maintenance planning by incorporating the constraint-based genetic algorithms to deal with the combined characteristics of the network-level maintenance planning. Two constraints of budget limitation and the network system preservation are employed in the developed models.

Certa [37] recently presented a paper which aims to propose a resolution approach for a multi objective maintenance problem with relation to a system that needs to operate without interruption between two consecutive fixed stops. The proposed algorithm has

several advantages compared with both the classical methods and the most recent genetic approaches. In particular, it goes over the limits of the other approaches due to the incapability in individuating all Pareto solutions and in exploring a not convex Pareto frontier.

A comprehensive study on multi objective genetic algorithms and their applications in reliability optimization problems is presented in a paper by Konak et al. [38]. They reviewed 55 research papers and discussed the recent techniques and methodologies.

2.2 Service part inventory control optimization

Spare parts inventories differ from work-in-process (WIP) inventories and finished product inventories from many aspects and are kept in stock to support maintenance operations and to protect against equipment failures. Managing spare parts is an important component of an overall maintenance policy, which can be a major determinant of operational efficiency in a manufacturing system.

Spare part definition in Wikipedia is: “A spare part, service part, or spare, is an item of inventory that is used for the repair or replacement of failed parts” “accessed on 11/06/2011.” Service Parts Management is one of the main components of strategic service logistics, requiring a complex decision making process that companies use to ensure that right spare parts and resources are at the right place at the right time. From a producer point of view spare parts are considered uneconomical since they involve logistical and economical requirements. However, without spare parts on hand,

customers' satisfaction level may drop, since customers have to wait for a long time before their products can be fixed [2].

Spare parts can be generally classified into non-repairable and repairable. Repairable parts are parts that are deemed worthy of repair, usually by virtue of economic consideration of their repair cost. Parts that are not repairable are considered consumable parts. Consumable parts are usually scrapped, or condemned, when they are found to have failed.

In the literature, the most commonly used approaches to develop a possible spare provisioning decision model are simulation and mathematical programming. Mathematical programming is based on linear programming, dynamic programming, goal programming, etc. [3].

The question of how many spare parts to stock and when is the best time to order the spare part have been addressed by numerous researchers and has originated a wide variety of models. A survey of the literature by Kennedy [4] is an update of the discussion of maintenance inventories and a discussion of the future research needed.

Maintenance, including tests, measurements, adjustments, and replacement, performed specifically to prevent faults from occurring. The goal of maintenance is to avoid or mitigate the consequences of failure of equipment. Maintenance has mainly been defined as two parts by its nature: PM and CM.

When discussing improvement opportunities in the plant, the PM discussion must occur. PM is preventing the failure before it actually occurs. It is designed to preserve and restore equipment reliability by replacing worn components before they actually fail.

CM involves the repair or replacement of components which have failed or broken down. For failure modes which lend themselves to condition monitoring, CM should be the result of a regular inspection which identifies the failure in time for CM to be planned and scheduled, then performed during a routine plant outage.

Today modern production systems are more complicated and mechanized. This causes unplanned failures to have a severe impact on the systems. Unplanned failures can decrease productivity and increase variance of production quality.

In general, the maintenance and spare parts inventory policies are treated either separately or sequentially in industry. However, since the stock level of spare parts is often dependent on the maintenance policies, it is better to deal with these problems simultaneously [5].

There are a limited number of published research papers that mentions the importance of integrating the maintenance strategy with spares and repair capacity (e.g. [6]; [7]). These articles do not present quantitative models. Spare provisioning policy has been taken into account simultaneously with the maintenance policy by Kabir and Farrash [8] and Park [9]. They deal with an age-based maintenance strategy and non-repairable components.

Brezavscek and Hudoklin [10] considered the problem of joint optimization of "PM" and "spare-provisioning policy" for system components subject to wear-out failures. This

model can be readily applied to optimize maintenance procedures for variety of industrial systems and to upgrade maintenance policy in situations where block replacement PM is already in use.

Huang [11] published a paper that considered a generalized joint optimization policy of block replacement & periodic review spare inventory with random lead time. In another paper the block replacement interval, the optimal stock level as well as the replenishment cycle is optimized simultaneously. Again the components are not repairable, which is encountered in most models that are concerned with joint optimization of a maintenance policy and a spares provisioning policy [12].

In another recent paper, Kolahan and Sharifinya [13] proposed a multi objective optimization problem in a single machine for simultaneous part sequencing and tool replacement schedule, with respect to tool reliability and sequence–dependent set up times has been addressed. The main objectives include determining optimal part sequence, tool selection for operations, tool replacement schedule, and number of spares for each tool type, in such a way that total expected production cost is minimized. Considering the defective cost by using tool reliability instead of tool life, processing operations with tool alternatives and tool loading by considering the limited tool magazine capacity, are the main originalities of this research. Since the problem under consideration is NP-hard, they propose a Simulated Annealing and Tabu Search heuristic algorithms to, simultaneously, provide part sequencing, tool replacement intervals and number of spare tools required. The proposed algorithms are examined and the results are compared by solving a real-sized example problem. The computational results

demonstrate the effectiveness of these methods towards solving large-sized, multi objective planning problems.

As mentioned before, spare parts can be generally classified into non-repairable and repairable. After an initial applied study with the Canadian oil producer Syncrude (see [14]), researchers at the Condition-Based Maintenance Laboratory at the University of Toronto have investigated and Developed 3 models to calculate the optimal stock size in the cases of non-repairable and repairable components. A repairable part is one that upon removal from operation (due to a preventive replacement or failure), is sent to a repair or reconditioning facility, where it is returned to an operational (ready-to-operate) state. Non-repairable parts, on the other hand, have to be discarded once they have been removed from operation (as it is uneconomical or physically impossible to repair them).Inventory control models used in each case are different, thus they will be treated separately. They have presented a number of basic spares inventory models used to determine the optimal stock size for the cases of non-repairable and repairable critical components, according to different optimization criteria, namely: (i) reliability of the stock (instantaneous or interval, depending on the application), (ii) availability (in the case of repairable components), and (iii) cost. In addition, procedures to find the interval of supportability given a stock level and desired reliability are introduced. Three brief case studies were reviewed, illustrating industrial spares stockholding problems. Most of the models discussed have been incorporated into a prototype software called SMS (Spares Management Software), developed by the Condition-Based Maintenance Laboratory at the University of Toronto [15].

Another paper in combination of spare parts and PM is by Tunali [16]. In this study, a simulation optimization approach using genetic algorithms (GAs) has been proposed for the joint optimization of PM and spare provisioning policies of a manufacturing system operating in the automotive sector. A factorial experiment was carried out to identify the best values for the GA parameters, including the probabilities of crossover and mutation, the population size, and the number of generations. The unavailability of spare parts at the time they are needed by the maintenance department is a major problem for many industrial organizations. The common approach to solve this problem is overstocking the spare parts at a substantial inventory-carrying cost. However, a cost effective solution to this problem requires a trade-off between overstocking and shortages of spare parts. In order to deal with this trade-off, the problem should be solved by joint, rather than separate or sequential optimization of PM and spare parts inventory policies. A simulation model of the manufacturing system was developed and a GA was integrated with this model to optimize the parameters of the simulation model. Moreover, a set of designed experiments was carried out to determine the best combination of GA parameters. The best solution proposed by the GA was compared to the current combination of control variables in terms of total annual cost and average monthly production. It was found that the total annual cost could be reduced by about 53% while achieving a larger amount of throughput.

Nosoohi and Hejazi [17] presented a novel multi objective model that considers age replacement policy and provision of spare parts both together. Despite most of previous studies where the cost objective has been the main concern in maintenance planning, this

paper presents a novel multi objective model (Cost objective, Corrective failure objective, Residual lifetime objective and Investment objective) for preventive replacement of a part over a planning horizon. The proposed model considers different objectives and practical issues, such as corrective replacement and its consequences, residual lifetime objective, and kind of productivity index. Also, the model determines number of spare parts, required for replacement with the defected part, to be provided at the beginning of the planning horizon. In this paper, unlike the previous researches and regarding practical issues, a new multi objective model was proposed. The classical cost objective was developed based on Bernoulli distribution. Along these lines, a function in the form of exponential distribution was used to show the effects of working situations and number of surplus spare parts on the probability of having spare part at the replacement times. Also they have shown, how non-dominated and the preferred solutions can be generated based on the ε-constraint and min max methods, for the proposed model.

2.3 Optimization Models

2.3.1 Dynamic programming

One of the most common techniques to solve the maintenance and replacement actions optimization models is dynamic programming. One of first studies in this field is a study by Canfield [39]. He mentions that PM actions do not change or affect deterioration behavior of failure rate, so the developed failure function is constant with maintenance actions. He proposed a model to minimize the cost of maintenance for a system that has

Weibull distribution failure rate. This model was solved by applying dynamic programming. Ben-Akiva [40] developed a dynamic programming method for finding an optimal maintenance and inspection policy, in the presence of inspection error.

2.3.2 Heuristics and Meta-Heuristics Algorithms

Genetic algorithm:

Recently, artificial intelligent technologies have better results in solving the optimization of nonlinear models; one of them is genetic algorithm. Genetic algorithms are inspired by Darwin's theory about evolution. Genetic algorithms in general are searching procedures based on the principle of natural selection and genetic recombination. They imitate nature by using the mechanics of evolution and natural selection to improve a set of initial solutions called a population using recombination and mutation of the genetic material.

Like any other optimization algorithm it begins by defining the optimization variables, the fitness function, and ends by testing for convergence. In between, however, this algorithm is quite different, i.e. it uses specific GA operators.

Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly and then improve it through repetitive application of mutation, crossover, inversion and selection operators.

It consists of the following procedures:

Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several

hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are

more "biology inspired", some research suggest more than two "parents" are better to be used to reproduce a good quality chromosome.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

Although Crossover and Mutation are known as the main genetic operators, it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

Simple generational genetic algorithm procedure:

- Choose the initial population of individuals
- Evaluate the fitness of each individual in that population
- Repeat on this generation until termination (time limit, sufficient fitness achieved, etc.):
- Select the best-fit individuals for reproduction
- Breed new individuals through crossover and mutation operations to give birth to offspring
- Evaluate the individual fitness of new individuals
- Replace least-fit population with new individuals

The most popular example in Genetic Algorithm is the eight queens puzzle. In chess, a queen can move as far as she pleases, horizontally, vertically, or diagonally. A chess board has 8 rows and 8 columns. The standard 8 by 8 queen's problem asks how to place 8 queens on an ordinary chess board so that none of them can hit any other in one move. Thus, a solution requires that no two queens share the same row, column, or diagonal. Solving this problem with a genetic algorithm is a basic example in tutorials.

GAs, initially introduced by Holland [41], constitute meta-heuristic population based, derivative-free optimization techniques, which exploit the mechanics of natural evolution in order to gradually approach optimality conditions [42].

The main advantage of GA over the classical optimization methods is that GA uses a 'fitness' function of various candidate solutions as the only information to guide the search. In addition, GA can easily deal with non-linear constraints and a large number of variables, and no derivatives or auxiliary information is needed [43]. GAs have been demonstrated to be particularly effective in determining solutions to multi objective optimization problems. Techniques such as MOGA (Multi Objective GA) and VEGA (Vector Evaluated GA) have been developed, and these algorithms have been refined so that they find the Pareto front in many problem instances. Munõz et al. [44] planned the component maintenance model by using the genetic algorithm to find a maintenance policy, which reached the minimum risk and cost. Tsai et al. [45] applied GA to provide PM and preventive replacement policies for a system from the viewpoint of unit lifetime cost.

Chen et al. [46] applied GA to determine an optimal PM policy of an *n*-component series system with deteriorated components in a mechanical system, and the effect of PM activities with reliability and failure rates of components under an age reduction model was studied. Marseguerra et al. [47] used Monte Carlo simulation and genetic algorithms to determine the optimal degradation level beyond which a PM intervention should be taken by optimizing profit and availability.

Moreover, Usher et al. [48] proposed an optimization maintenance and replacement model for a single-component system. They presented a new model to optimize the PM schedule for a system with increasing failure rate and compared the results from genetic algorithm method and branch and bound algorithm with each other.

Levetin and Lisnianski [49] proposed an optimization model for a multi state system to determine PM actions that affect the effective age of components. They used GA in order to minimize the total cost with a required level of reliability as the constraint. They had another paper in which they proposed a model to determine the optimal time for replacement in a multi state series-parallel system with an increasing failure rate. They utilized GA to solve the total cost objective function.[50]. In another paper by Wang [51], he presents a more efficient GA for unit maintenance scheduling based on the specific characteristic of PM scheduling problem for power systems. This new GA improves GA computation performance by adopting a code-specific and constraint-transparent integral coding method. To form a more promising convergence sequence and to refrain from the occurrence of unfeasible solutions, in this new GA, GA operators are redesigned according to the specific characteristics of the problem to be solved. Comparisons of this new GA with a traditional binary GA are also discussed in this paper.

In a paper by Cavory [52], a model to optimize the schedule of maintenance tasks of all the machines in a single product manufacturing production line was proposed. They considered the total throughput of the line as the objective function and tried to maximize it by applying genetic algorithm to find the best combination of PM tasks. They set the GA parameters by constructing an experimental design and validated the results by utilizing Taguchi method and statistical analysis. There are similar papers that used GA to optimize the cost function, another one is a paper by Leou [53] that considered maintenance crew and duration of maintenance as the additional constraints to this method. He applied the optimization model in a case study with six electric generators.

Han et.al. [54] illustrated the dynamic relationship between failure rate and PM activity. The proposed nonlinear optimal PM policy model satisfies the reliability constraints in finite time horizon following Weibull distribution. They applied GA as the optimization method.

In another work by Limbourg [55], they used heuristics and meta-heuristics optimization algorithms for PM scheduling models and presented several nonstandard input representations and compares them to the standard binary representation by a heuristic algorithm. An evolutionary algorithm with extensions to handle variable length genomes is used for the comparison. The results demonstrate that two new representations perform better than the binary representation scheme. A second analysis shows that the performance may be even more increased using modified genetic operators.

There are some other meta-heuristics methods that have been used broadly for solving the maintenance optimization problems. In a paper by Wang [56], they tried to minimize the periodic PM cost for a series-parallel system using the particle swarm optimization (IPSO). The importance measure of components is utilized to evaluate the effects of components on system reliability when maintaining a component. I can mention another paper by Samrout et al. [57]. This article is based on a previous study made by Bris [58]. They use genetic algorithm to minimize PM cost problem for the series–parallel systems. Samrout proposed to improve their results developing a new method based on another technique, the Ant Colony Optimization (ACO). The resolution consists in determining the solution vector of system component inspection periods, T_P. Those calculations were

applied within the programming tool Matlab. They obtained highly interesting results and improvements of previous studies.

2.4 Game Theory

Game theory is the formal study of decision making where several players must make choices that potentially affect the interests of the other players.

Game theory is the formal study of conflict and cooperation. Game theoretic concepts apply whenever the actions of several agents are interdependent. These agents may be individuals, groups, firms, or any combination of these. The concepts of game theory provide a language to formulate structure, analyze, and understand strategic scenarios.

The first theorem of game theory is from Zermelo who showed that chess is strictly deterministic. But, the idea of general theory of games back to 1944 when Von Neumann and Morgenstern published the book "Theory of Games and Economics Behavior." They proposed that most economic questions should be analyzed as games and introduced the method of finding mutually consistent solutions for two-person zero-sum games. During the late 1940s, cooperative game theory had been studied to analyze how groups of individuals should cooperate with each other to improve their positions in a game.

A game consists of a set of players, a set of moves (or strategies) available to those players, and a specification of payoffs for each combination of strategies.

2.4.1 Basic Elements and Assumptions of Game Theory

Player

An economic agent is, by definition, an entity with preferences. A player is an agent who makes decisions in a game. Further, we also assume that each member acts rationally, i.e. each member will not raise its own cost for the purpose of raising cost of the other members.

Strategy

A player's strategy will determine the action the player will take at any stage of the game, for every possible history of play up to that stage.

Payoff

The payoffs represent the welfare of the players at the end of the game. They are the basis on which each player chooses his strategy.

2.4.2 Representation of games

Strategic Form Games

A game in strategic form, also called normal form, is a compact representation of a game in which players simultaneously choose their strategies. The resulting payoffs are presented in a table with a cell for each strategy combination.

To define a game in strategic form we need only specify the set of players in the game, the set of options available to each player, and the way that players' payoffs depend on the options they choose (payoff functions) [59]. Classical example of a two-player finite strategic form game is the famous prisoners' dilemma game.

Extensive Form Games

An extensive game (or extensive form game) describes with a tree how a game is played. It depicts the order in which players make moves, and the information each player has at each decision point.

2.4.3 Types of games

Perfect information and imperfect information:

Games are often classified by the amount of information available to the players. If a player has access to all the information they require about the game during play, then the game can be classified as having perfect information. However, if some of that information is hidden from the player the game is known as having imperfect information. Take for example the game of chess. Chess is a game of perfect information because each player can look down upon the board and obtain all the information necessary to make their playing decisions. On the other hand, the game of poker is a game of imperfect information. In poker, players are given cards which only they can see; therefore players now have to make decisions based on hidden information because they cannot see their opponents' cards. Games with incomplete information can be modeled as Bayesian games, where the uncertainty is handled by using probability distributions.

Deterministic or stochastic:

Games can be further classified as either deterministic or stochastic. If a game contains chance elements, such as the roll of a dice, this introduces randomness into the game. These types of games are known as stochastic games and examples include bridge,

backgammon and poker. The absence of these chance elements ensures the game is deterministic. Games such as chess, checkers and go are examples of deterministic games.

Cooperative or non-cooperative:

The word non-cooperative means that the players' choices are based only on their perceived self-interest. The most important models used for representing non-cooperative games are the strategic form and the extensive form. The first is conceptually simpler and is generally viewed as being derived from the extensive form, which is more richly structured way to describe game situations.

Zero-sum and non-zero-sum:

Zero-sum games are a special case of constant-sum games, in which choices by players can neither increase nor decrease the available resources. In zero-sum games the total benefit to all players in the game, for every combination of strategies, always adds to zero.

Simultaneous and sequential:

Simultaneous games are games where both players move simultaneously, or if they do not move simultaneously, the later players are unaware of the earlier players' actions (making them effectively simultaneous). Sequential games (or dynamic games) are games where later players have some knowledge about earlier actions. This need not be perfect information about every action of earlier players; it might be very little knowledge.

2.4.4 Equilibrium Solutions

A solution concept for a game is any rule for specifying predictions as to how players might be expected to behave in any given game. The most important solution concept in game theory is Nash's concept of equilibrium [60].

Nash equilibrium

A Nash equilibrium, also called strategic equilibrium, is a list of strategies, one for each player, which has the property that no player can unilaterally change his strategy and get a better payoff.

Nash equilibrium is widely considered as the solution of non-cooperative games.

$$\pi_i\left(s_i^{Nash}, s_{-i}^{Nash}\right) \geq \pi_i\left(s_i, s_{-i}^{Nash}\right)$$

For

$i = 1,2,\ldots,G$ and $\forall\ s_i \in S_i$

Where $\pi_i\left(s_i, s_{-i}^{Nash}\right)$ implies the payoff of player i when player i selects S_i as his strategy and at the same time all the other players except player i select s_{-i}^{Nash} as their strategies.

That is, $s_{-i}^{Nash} = [\, s_1^{Nash}, s_2^{Nash}, \ldots, s_{i-1}^{Nash}, s_{i+1}^{Nash}, \ldots, s_G^{Nash}]$

S_i is the feasible strategy set of player i.

Backward induction

Backward induction is a technique to solve a game of perfect information. It first considers the moves that are the last in the game, and determines the best move for the player in each case. Then, taking these as given future actions, it proceeds backwards in time, again determining the best move for the respective player, until the beginning of the game is reached.

There are several papers that use a game theoretic approach in maintenance scheduling.

A novel approach to a generating unit maintenance scheduling problem in competitive electricity markets is presented in a paper by Kim [61]. The objective is to develop a game-theoretic framework for analyzing strategic behaviors of generating companies (Gencos) from the standpoint of the generating unit maintenance scheduling (GMS) game and for obtaining the equilibrium solution for the GMS game. The GMS problem is formulated as a dynamic non-cooperative game with complete information. The players correspond to profit maximizing individual Gencos, and the payoff of each player is defined as the profits from the energy market. The optimal schedule is defined by Nash equilibrium (equilibriums) of the game. Numerical results for two-Genco system are used to demonstrate that the proposed framework can be successfully applied to analyzing the strategic behaviors of each Genco and to obtaining the corresponding Nash equilibrium. The result indicates that generating unit maintenance schedule is one of the major strategic behaviors whereby Gencos maximize their profits in a competitive market environment.

A tutorial on the subject is provided by Cachon and Netessine [62], where both non-cooperative and cooperative game theories in static and dynamic settings are discussed.

For more extensive concepts of game theory, the readers are referred to [63].

2.4.5 Leader-Follower Game (Stackelberg Games)

As one of the most important types of game, Stackelberg games originate from H. von Stackelberg who studied a duopoly model where the other company had a dominant position being able to make its decision first. In general, Stackelberg games are leader-follower games where the players act sequentially. Stackelberg solution is an important hierarchical solution concept for both static and dynamic game models. From an optimization point of view, two-player Stackelberg games are two level hierarchical optimization problems where the leader optimizes his utility subject to follower's optimization problem. When the players mutually benefit from the leadership of one of them, the solution is called concurrent. If each player prefers to be the leader himself, then the Stackelberg solution is called non-concurrent and the Stackelberg game where neither of the players wants to be the leader is called stalemate [64].

Genetic algorithms have been applied in the distributed computation of both Stackelberg and incentive Stackelberg solutions. Vallee and Basar[65][66] study off-line computation of the Stackelberg solution (single-leader–single-follower) in a repeated game framework, utilizing the Genetic Algorithm. In this paper they consider natural leader and natural follower as fixed roles.

Nedim and Sirakaya [67] develop a method to compute the Stackelberg equilibrium in sequential games. They construct a normal form game which is interactively played by an artificially intelligent leader, GA_L, and a follower, GA_F. The leader is a genetic algorithm breeding a population of potential actions to better anticipate the follower's reaction. The follower is also a genetic algorithm training on-line a suitable neural network to evolve a population of rules to respond to any move in the leader's action space. When GAs repeatedly plays this game updating each other synchronously, populations converge to the Stackelberg equilibrium of the sequential game.

D'Amato et al. [68] developed a computational methodology to obtain a Stackelberg - Nash solution for a hierarchical game via genetic algorithm (GA). There is one (or more) players acting as leader(s) in a two level leader-follower model, the rest of players play a non-cooperative game and react to the optimal decision taken by the leader(s). The leader(s) takes into account the followers' best reply and solve an optimization problem (a Nash equilibrium problem). In this model the uniqueness of the Nash equilibrium of the follower players has been supposed.

2.5 Wind Turbine Reliability

Wind turbine industry has gained a remarkable stand in the US industry since the turn of the century. Reliability of wind turbines has attracted much attention especially in recent years. Wind power is a fast growing renewable energy resource. Reliability evaluation and enhancement are an important factor in modern power system planning and

operation. Consequently, reliability assessment of wind turbines is of great importance and will receive more attention in the future due to the increase in wind power utilization.

The reliability of wind turbines as a part of a large power system is assessed in many references [69][70].

In another paper by Arabian [71], they propose a reliability model for the electrical subassemblies of geared wind turbine systems with induction generators.

The wind turbine system consists of different subassemblies such as blades, tower, bearings and shaft, gearbox for indirect drive, generator, converter for variable-speed and the necessary control units.

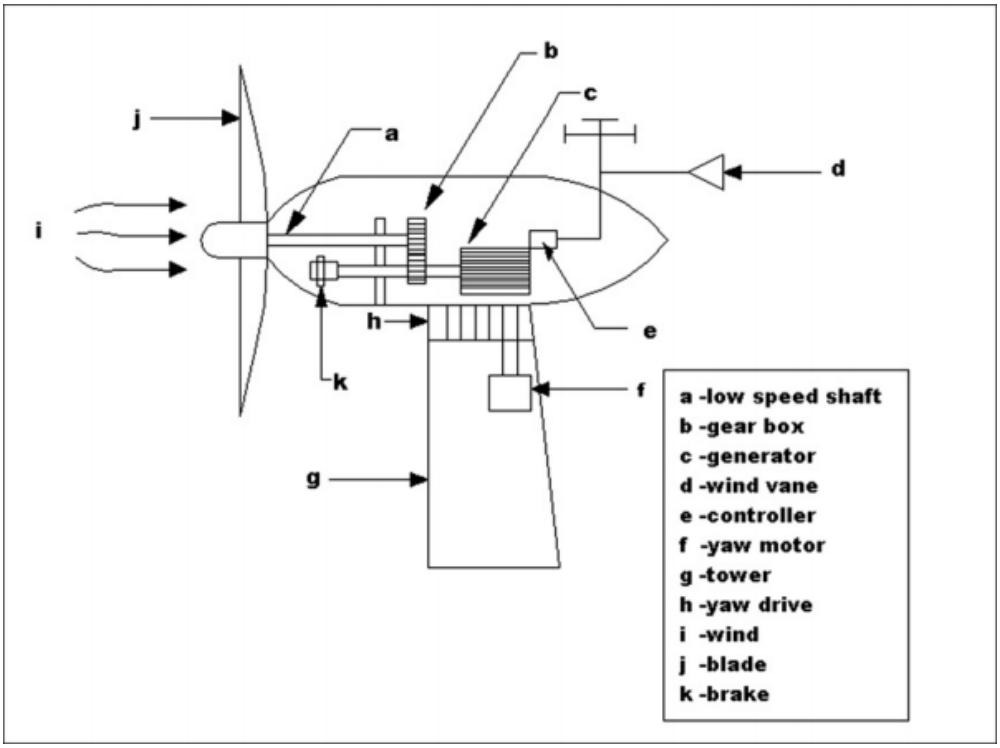

Figure 3. Diagram of Wind Turbine Generator [71]

Some of the power available in the wind is converted by the rotor blades to mechanical power acting on the wind turbine rotor shaft.

The wind energy industry typically uses reactive maintenance approach or run-to-failure maintenance. This form of maintenance has been shown to be the most costly Operations and Maintenance (O&M) practice available to operators. There are several papers that worked on the reliability wind turbines. For example, in a paper by Cohen [72], they proposed a model by considering both scheduled and unscheduled maintenance.

2.5.1 Scheduled (Preventive) Maintenance of wind turbines

The objective of PM is to replace components and refurbish systems that have defined useful lives, usually much shorter than the projected life of the turbine. Tasks associated with scheduled maintenance fall into this category. These tasks include periodic inspections of the equipment, oil and filter changes, calibration and adjustment of sensors and actuators, and replacement of consumables such as brake pads and seals. Housekeeping and blade cleaning generally fall into this category. The specific tasks and their frequency are usually explicitly defined in the maintenance manuals supplied by the turbine manufacturer. Costs associated with planned maintenance can be estimated with reasonable accuracy, but can vary with local labor costs and the location and accessibility of the site. Scheduled maintenance costs are also dependent on the type and cost of consumables used.

2.5.2 Unscheduled (Failure Related) Maintenance of wind turbines

A certain amount of unscheduled maintenance must be anticipated with any project. Commercial wind turbines contain a variety of complex systems that must all function correctly for the turbine to perform; rarely are redundant components or systems incorporated. Failure or malfunction of a minor component will frequently shut down the turbine and require the attention of maintenance personnel.

Unscheduled costs can be separated into direct and indirect costs. The direct costs are associated with the labor and equipment required to repair or replace, with the component costs themselves, and with any consumables used in the process. The indirect costs result from lost revenue due to turbine downtime.

Labor costs are driven by the difficulty of accessing and working on the components. With the exception of some switchgear and power conversion equipment, most the turbine equipment is accessed by climbing the tower. For safety reasons, a two-person crew is generally required for any up-tower activity. In remote locations, access to the turbine itself may be difficult and limited by weather. Working conditions can be in extreme temperature conditions and may be curtailed by high winds. Some turbines are equipped with hoists and rigging equipment, but in general, all tools and equipment, in addition to spares, must be lifted into the nacelle. Space is limited inside the nacelle and working positions may be awkward. Work outside of the nacelle, including transitions into the hub on some turbines, requires working with a safety harness and lanyards [73].

A good source for reliability information is the renowned reliability expert Paul Barringer, who has developed a Weibull reliability database for failure data for various

components, available on his websites (http://www.barringer1.com/) as a service to reliability engineers. This database lists components that are also found in wind turbines including roller bearings, gears, lubrications pumps, couplings, gaskets, circuit breakers, AC motors, and synthetic lubrications oils that all have typical Weibull characteristic life in the 50,000 to 100,000 hours [74].

2.5.3 Operation and Maintenance Costs of Wind Generated Power

The industry-wide accepted turbine lifetime is 20 years (Due to the relative infancy of the wind energy industry, there are only a few turbines that have reached their life expectancy of 20 years). Thus, the reliability of a turbine is the percentage of time (probability) that turbine will be functioning at full capacity (intended function) during appropriate wind conditions at a site with specified wind resource characterization (stated conditions) for a 20-year life (time).

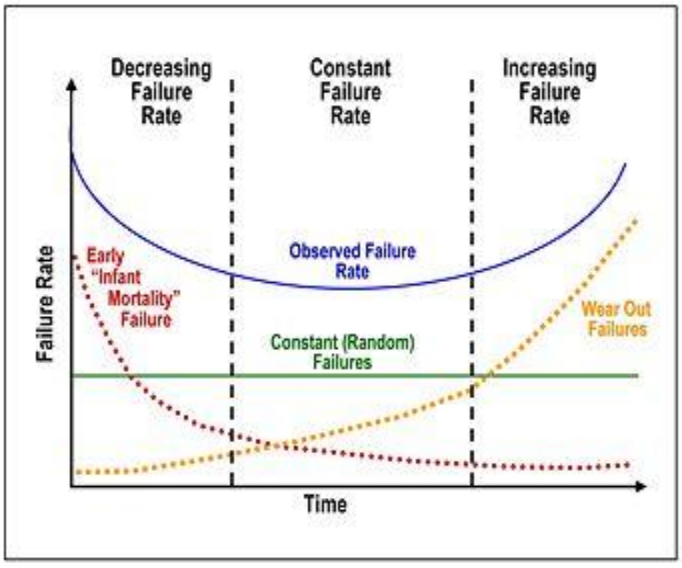

Figure 4. Wind Turbine Bathtub Curve [75]

In the wind turbine reliability, understanding and minimizing wind turbine operation and maintenance costs have been made through a number of studies. The annual O&M cost is indicated in $/kWh as the plant age ranges from the first year of operation through year 20 as shown in Figure 4.

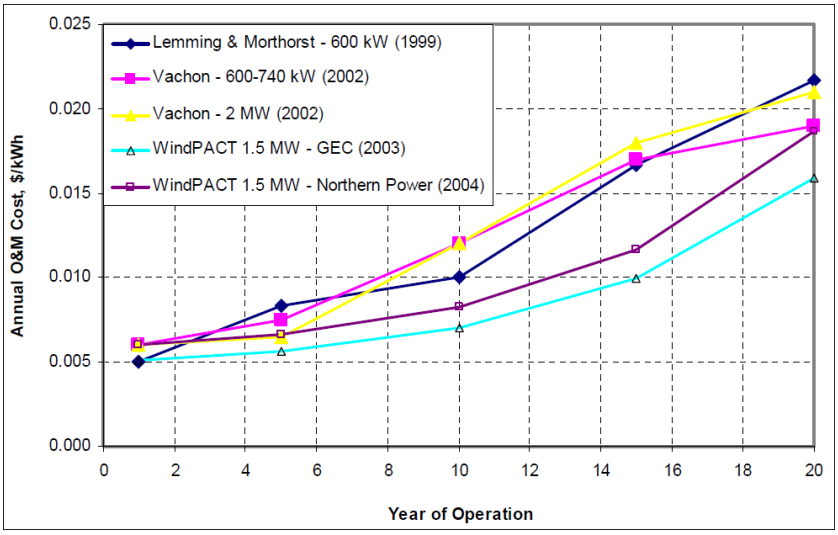

Figure 5. Total Operations and Maintenance Costs Increase with Age Due to Wear-Out Related Failures [75]

Operation and maintenance (O&M) costs constitute a sizeable share of the total annual costs of a wind turbine. For a new turbine, O&M costs may easily make up 20-25 per cent of the total levelised cost per kWh produced over the lifetime of the turbine. If the turbine is fairly new, the share may only be 10-15 per cent, but this may increase to at least 20-35 per cent by the end of the turbine's lifetime. As a result, O&M costs are attracting greater attention, as manufacturers attempt to lower these costs significantly by developing new turbine designs that require fewer regular service visits and less turbine downtime.

O&M costs are related to a limited number of cost components, including:

- Insurance;
- Regular maintenance;
- Repair;
- Spare parts, and
- Administration.

Some of these cost components can be estimated relatively easily. For insurance and regular maintenance, it is possible to obtain standard contracts covering a considerable share of the wind turbine's total lifetime. Conversely, costs for repair and related spare parts are much more difficult to predict. And although all cost components tend to increase as the turbine gets older, costs for repair and spare parts are particularly influenced by turbine age; starting low and increasing over time [75].

More simply, the Electric Power Research Institute (EPRI) has detailed case studies in the electric power industry and has shown that reactive maintenance (running the machine until it fails) is the least effective and the most costly approach to power generation equipment maintenance. EPRI's comparative maintenance costs are listed below:

- Reactive maintenance (run to failure) costs $17.00 USD per horsepower per year (This is the baseline.)

- PM (scheduled maintenance according to the manufacturer's recommendations) costs $24.00 USD per horsepower per year (a savings of 24 percent compared to reactive maintenance)
- Predictive maintenance (using condition monitoring to predict maintenance needs) costs $9.00 USD per horsepower per year (a savings of 47 percent compared to reactive maintenance)

If turbine components are allowed to run to failure, the overall energy production is significantly decreased due to unscheduled downtime. At the same time, the cost of rushed parts and crane operations, as well as collateral damage caused by the failing component leading to additional damage, further increases maintenance costs. Reactive maintenance costs are then significant cost increases far above the cost of predictive maintenance using an online condition monitoring system. The condition monitoring system’s function is to continuously monitor components and predict mechanical problems, enabling operators to schedule maintenance and avoid catastrophic failures.

2.5.4 Gearbox

According to the gearbox's reputation for a high failure rate, one of the biggest concerns remaining in the wind industry is the reliability of the gearbox. Gearboxes in WTs are used to increase the speed from the main shaft to the generator shaft, which turns at 1500 rpm (with mains frequency 50 Hz) for conventional generators. The gearbox is one of the heaviest and most expensive components in a WT. In this context, it is unfortunate that under dimensioned gearboxes have had a large part in WT failures. The reason for under-

dimensioned gearboxes can be that the gearbox manufacturers do not fully understand the operating conditions.

Indeed, gearbox failures are regarded as one of the most serious breakdown causes in a wind turbine for two reasons. Firstly, because of the high cost of repairing or replacing the gearbox and, secondly, because of the resulting downtime. Replacing a wind turbine gearbox involves primarily the gearbox cost itself, which typically represents around 10% of the total wind turbine cost. On top of this expense, must be added its transportation to site, crane rental and mobilization cost, and the man-hours spent on the replacement. It means that the value can quickly reach about €200,000 – €500,000, depending on the turbine size and the wind farm's location.

A gearbox failure typically causes two to three times more downtime than any other component failure. In general, a gearbox replacement takes about a week, assuming that the required spare gearbox is available. Customers may have invested in a few spare gearboxes to handle isolated failure cases, but mobilizing the cash to keep spares in inventories for a complete fleet of wind turbines approaching the critical '7 – 11 year' milestone will be a challenge of a different magnitude for wind farm owners. This uncertainty therefore adds to the gearbox replacement cost a significant unavailability risk that is difficult to assess and include in wind farm business plans.

Gearboxes are built up of shafts, gears, bearings and seals, mounted in a metal cover. The weight of the gearbox increases dramatically in relation to the rated power of the WT. The main load a gearbox has to handle is torque of the rotor. This load is sometimes constant and sometimes fluctuating. It also suffers loads from the generator when starting

up. These loads mainly affect bearings, gear teeth and seals, causing them to fail. To minimize fatigue of gearbox parts, a functional and efficient lubrication system is highly relevant [76]. A problem with the gearbox is that even if it is only a small cog breaking. The whole system needs to be cleaned out and thoroughly tested. Faults with gearboxes are primarily discovered within the first two years of operation. If a gearbox last the first two years it is likely that it will last for many years.

2.6 Positioning of the study in cited literature

Although the modeling concept of this work is new, four of the studies in the literature discussed up to this point are closer to this study than the others in terms of its modeling aspects ([65][66][67][68]). They developed a computational methodology to obtain a Stackelberg - Nash solution for a hierarchical game via genetic algorithm. It should be mentioned that these studies considered fixed roles for players (leader and follower). This thesis tries to look upon the probabilities of being leader for each of the operators rather than having fixed roles, so two cases based on cooperative game concept and joint optimization will be proposed. The proposed model can be used for various problems in industry, wherever there is a competition on resource allocation.

CHAPTER III

3 Optimization Model

3.5 Introduction

This chapter describes the goals that this research seeks to accomplish. We will present a novel model for maintenance policy evaluation based upon a game theoretic model and optimize the proposed model by backward induction and Genetic Algorithm methods and compare the results.

The goal of this thesis is to introduce a joint maintenance decision-making mechanism for the two operators that minimize the average of the expected operational costs. In fact, these two operators should compete with each other on ordering the gearbox in the best time that minimizes the total cost.

As the optimization methodology, at first, the common method for solving Stackelberg game problems, backward induction, has been conducted to find the optimal PM and ordering times. Then, genetic algorithm for a multi objective model is utilized and the results are compared with each other. The effectiveness of the approach is presented through the use of numerical examples.

3.6 Problem Statement

3.2.1 Assumptions and overview

The following assumptions are considered.

Assumptions:

Two repairable systems are considered, which are operated by two individual operators. Each system requires one unit of a specific type of component for being operable. When the two systems are working, both units are operational and subject to failures.

Without loss of generality, the Weibull distribution is assumed for the *RUL* of each aging unit (with an increasing failure rate). For the Weibull distribution, the associate reliability function at time t can be expressed as

$$R\,(t) = e^{-\left(\frac{t}{\eta}\right)^{\beta}}, \tag{1}$$

where β is the shape parameter and η is the scale parameter. This can be justified by considering a degradation process $Z(t)$, which after an appropriate transformation $\varphi(\cdot)$ can be expressed as $X(t) = \varphi(Z(t)) = x_0 + \alpha t, t > 0$, where x_0 is a constant, and α follows the reciprocal Weibull distribution with probability density function:

$$g(\alpha) = \beta_0 \alpha_0^{\beta_0}\, \alpha^{-\beta_0 - 1} \exp\left[-\left(\frac{\alpha}{\alpha_0}\right)^{-\beta_0}\right], \text{ for } \alpha > 0,$$

with parameters α_0 and β_0. It can be shown that the corresponding failure time of $X(t)$ for a predetermined failure threshold $D_f > x_0$ follows the Weibull distribution with probability density function $f_{T_f}(t_f) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}\ e^{-\left(\frac{t}{\eta}\right)^{\beta}}$, where $\beta = \beta_0$ and $\eta = \frac{D_f - x_0}{\alpha_0}$.

There is only one supplier who makes the service part for the two operators, and a make-to-order (*MTO*) strategy is implemented by the supplier.

There are two causes of downtime: failure replacement and preventative replacement. Each unit will be maintained after time T_i, and the system is good-as-new after corrective replacement or preventive replacement. When a failure occurs, the failed unit is replaced by a ready-for-use part from the temporary storage. If the part required for replacement is not available at the storage, an order must be placed and the replenishment takes τ_1 days. We suppose when a failure occurs, the operator will replace the failed part with a new one. It will take τ_3 days to diagnose and replace the part. The time needed to perform a preventive replacement at a fix interval of T_i is τ_2 days.

We consider a leader-follower relationship between these two operators. The assumption which we make for this problem is that for a new part, its failure-free time period >> τ_1 (i.e., the operators will not compete again before the operator, who lost in the previous competition, eventually receives the needed part). First, like regular cases, we consider fixed roles, which means that one of the operators is always the leader and the other one is always the follower. We will optimize the Stackelberg game model by backward induction. Second, we will consider the possible probabilities for each of these operators to be leader and decide first and solve this new model by computing the Stackelberg solution with the GA. Our algorithm uses a weighted sum and an expected sum (based on the *RUL*s of units) of multiple objectives as fitness functions. The fitness function is utilized when a pair of parent solutions is selected for generating a new solution by crossover and mutation operations.

3.2.2 Acronyms and notation

RUL	remaining useful life
MTO	make-to-order
ETRC	expected total replacement cost
τ_1	replenishment lead time under *MTO*
τ_2	time needed to perform preventive replacement
τ_3	time needed to perform corrective replacement ($\tau_3 > \tau_2$)
C_o	regular ordering cost
M	extra charge for bidding on a part
$D_{cost(k)}$	failure downtime cost for Operator k
$\delta_k(D_{cost(k)} > M)$	indicator function $\begin{cases} 1 & \text{Operator } k \text{ will pay more to get the part (when } D_{cost(k)} > M) \\ 0 & \text{Otherwise} \end{cases}$
$f_j(t)$	probability density function of *RUL* of the unit owned by Operator j
$F_j(t)$	cumulative distribution function of *RUL* of the unit owned by Operator j
$\Pi_j(t_j)$	objective function of Operator j
w_j	weight assigned to the objective function of Operator j
p_j	probability for Operator j to be the leader
$c_{p(j)}$	unit downtime cost due to preventive replacement for Operator j
$c_{f(j)}$	unit downtime cost due to a failure for Operator j

$c_{h(j)}$ unit holding cost for Operator *j*

T_j time to perform preventive replacement for Operator *j*

$T_{o(j)}$ time to order a service part for Operator *j*

$T_{f(j)}$ random failure time of the unit being used by Operator *j*

$\rho(T_{o(k)} - T_{o(j)} < \tau_1)$ indicator function

$$\begin{cases} 1 & \text{Operator } k \text{ decides to order the part before Operator } j \text{ gets it} \\ 0 & \text{Otherwise} \end{cases}$$

3.2.3 Decision: Pay more or wait

Without loss of generality, let Operator *j* be the leader and Operator *k* be the follower. We define the following indicator functions first:

$\rho(T_{o(k)} - T_{o(j)} < \tau_1)$

$$= \begin{cases} 1 & \text{Operator k decides to order the part before Operator } j \text{ gets it} \\ 0 & \text{Otherwise} \end{cases}$$

and $\delta_k(D_{cost(k)} > M)$ =

$$\begin{cases} 1 & \text{Operator } k \text{ will pay more to get the part (when } D_{cost(k)} > M) \\ 0 & \text{Otherwise} \end{cases}$$

To explain possible cases, let *j*=2 and *k*=1 for example. In Fig. 2, the two diagrams at the top show the two possible situations for the operators. The left diagram depicts the case in which $T_{o(1)}$-$T_{o(2)}$is less than replenishment lead time. In this case, there is a competition between these two operators to achieve the service part. If the downtime cost $D_{cost(1)}$ is greater than the extra charge (*M*), Operator 1 will pay C_o+*M* to get the part (i.e., $\delta_1(D_{cost(1)} > M) = 1$). Otherwise, it is not beneficial for Operator 1 to pay the

extra for bidding on the part and he/she would prefer to wait and pay the downtime cost (i.e., $\delta_1(\ D_{cost(1)}\ >\ M) = 0$). The right diagram depicts the case where $T_{o(1)}$-$T_{o(2)}$ is not less than replenishment lead time. In this case, there is no interference between their ordering times, so the lead operator will first order the service part and the follower operator can order another one after that without any competition.

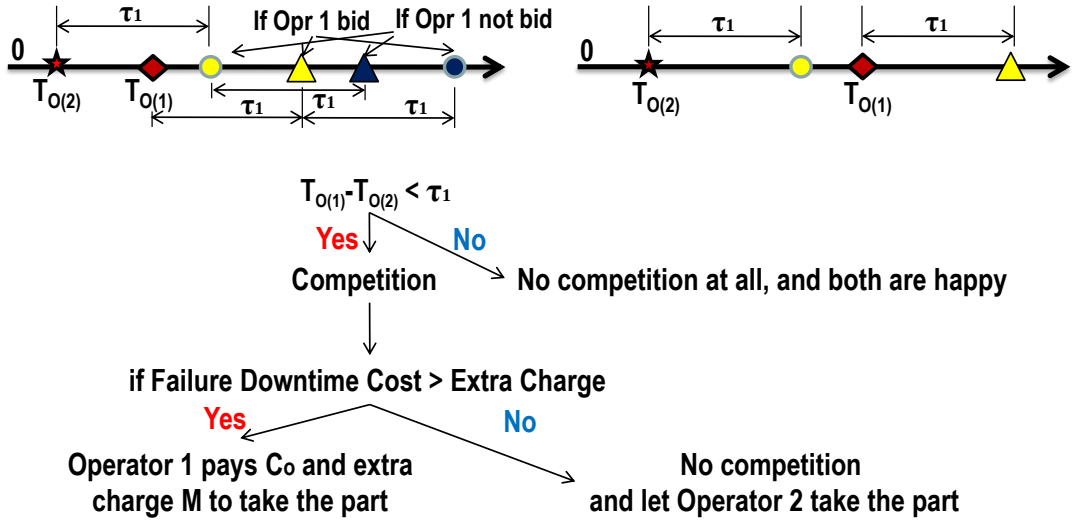

Figure 6. Decision: Pay more or wait

3.2.4 Waiting times for getting a part

Considering an *MTO* strategy, whenever the leader places an order and if the order is not interrupted by the follower, the leader has to wait for τ_1 days to get the part. In Fig. 3, the upper diagram shows that if Operator 1 decides to bid on the service part (δ_1=1), he/she should wait for τ_1 days until the blue triangle, and Operator 2 should wait for $T_{o(1)}$-$T_{o(2)}$+2τ_1 days to receive the part (yellow circle). In case Operator 1 doesn't tend to pay more on the service part, he/she should wait for $T_{o(2)}$-$T_{o(1)}$+ 2τ_1 days (green triangle), and

Operator 2 will receive the part after τ_1 days (red circle). The other diagram depicts the case in which there is no competition between the two operators and each of them should wait for τ_1 days to receive the part (Fig. 3).

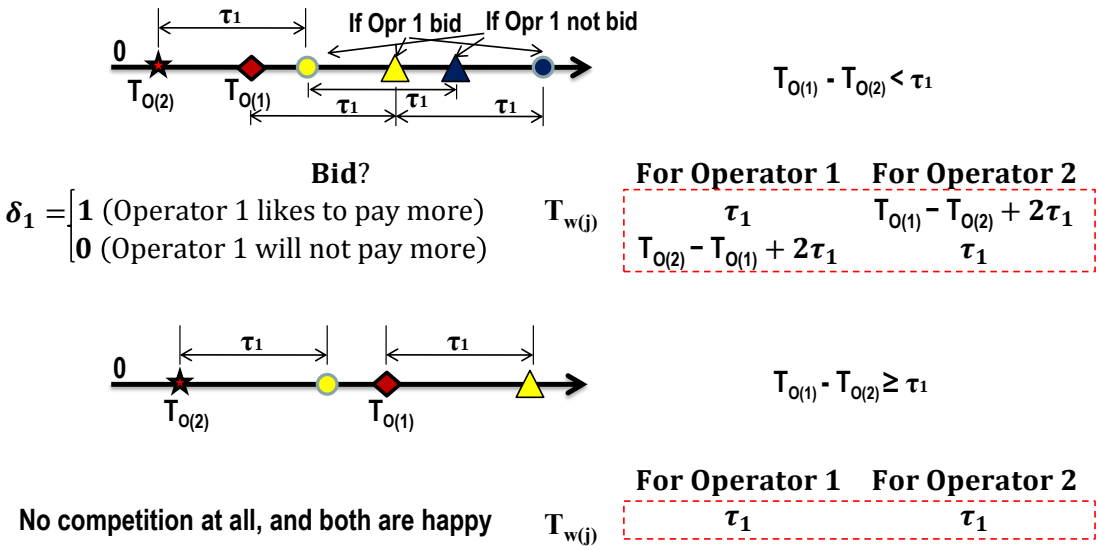

Figure 7. Waiting times before receiving a part

3.2.5 Cost functions for Operator j (leader)

The actual waiting time for Operator *j* can be expressed as:

$$T_{w(j)} = \rho(T_{o(k)} - T_{o(j)} < \tau_1)[\delta_k\,(D_{cost(k)} > M) \cdot max\,(\tau_1 + T_{o(k)} - T_{o(j)}\,, 0) + \tau_1] + (1 - \rho(T_{o(k)} - T_{o(j)} < \tau_1)\,)\,\tau_1 \qquad (2)$$

By taking into account the preventive replacement downtime cost, holding cost, and regular ordering cost for the service part, the total preventive replacement cost for Operator *j* is given by:

$$C_{p(j)} = c_{p(j)}(max\,(T_{w(j)} + T_{o(j)} - T_j,\, 0) + \tau_2) + c_{h(j)}(max\,(T_j - T_{w(j)} - T_{o(j)},\, 0)) + C_o \qquad (3)$$

Likewise, considering the failure replacement downtime cost along with others, the total corrective replacement cost is:

$$C_{u(j)}= c_{f(j)}(max\ (T_{w(j)}+T_{o(j)}-T_{f(j)}, 0) + \tau_3) + c_{h(j)}(max\ (T_{f(j)}-T_{w(j)}-T_{o(j)}, 0)) + C_o \quad (4)$$

3.2.6 Cost functions for Operator k (follower)

The actual waiting time for Operator *k* is given by:

$$T_{w(k)}= \rho(T_{o(k)} - T_{o(j)} < \tau_1)[(1-\delta_k(D_{cost(k)} > M)) \cdot max\ (\tau_1+ T_{o(j)} - T_{o(k)}, 0) + \tau_1]$$

$$+(1 - \rho(T_{o(k)} - T_{o(j)} < \tau_1)\ )\ \tau_1 \quad (5)$$

In this problem we have:

$$D_{cost(k)} = c_{f(k)}(max\ (T_{w(k)}+T_{o(k)}-T_{f(k)}, 0) + \tau_3) \quad (6)$$

Therefore, the total preventive replacement cost can be expressed as:

$$C_{p(k)}= c_{p(k)}(max\ (T_{w(k)}+T_{o(k)}- T_k, 0) + \tau_2) +c_{h(k)}(max\ (T_k- T_{w(k)}-T_{o(k)}, 0))$$

$$+C_o+ \delta_k(D_{cost(k)}>M)\ M \quad (7)$$

and the total corrective replacement cost is:

$$C_{u(k)}= D_{cost(k)} + c_{h(k)}(max\ (T_{f(k)}-T_{w(k)}-T_{o(k)}, 0)) +C_o+ \delta_k(D_{cost(k)}>M)\ M \quad (8)$$

3.2.7 Decision-making criterion

Considering both preventive replacement and corrective replacement, the expected total replacement cost (*ETRC*) for each operator can be expressed as:

$$ETRC_i(T_i\ , T_{o(i)})= \mathrm{E}[\text{Total replacement cost } i] = C_{p(i)}R_i(T_i) + \int_0^{T_i} C_{u(i)}\ f(t_{f(i)})dT_{f(i)},$$

$$= C_{p(i)}R_i(T_i) + \int_0^{T_i} C_{u(i)}\ \frac{\beta}{\eta}\left(\frac{T_{f(i)}}{\eta}\right)^{\beta-1} \mathrm{e}^{-\left(\frac{T_{f(i)}}{\eta}\right)^{\beta}} dT_{f(i)} \qquad i =1, 2 \quad (9)$$

where $R_i(T_i)$ is the reliability function of the Weibull distribution given in Eq. (1), and

$f(t_{f(i)}) = -\frac{dR_i(t_{f(i)})}{dt_{f(i)}}$ is the corresponding probability density function.

3.3 Three scenarios of competition

3.3.1 Hierarchical game: Stackelberg equilibrium (with fix-leader and fix-follower roles)

Originally Stackelberg game is a model for a leader-follower game in which two players act sequentially such that the first player (the leader) chooses her strategy and the other player (the follower) reacts rationally to that strategy. Any player is assumed to minimize her own payoff function corresponding to a cost function.

We consider the following basic assumptions for the game:

1. Perfect information: Each player has perfect information about the other's actions and strategy.
2. Rationality: Both players act optimally.
3. Determinism: Each player chooses deterministically among alternative optima.

For each strategy proposed by the leader, the follower has in fact to determine a strategy that minimizes his/her objective function until equilibrium is found when the leader has also minimized his/her objective function. In mathematical terms:

$(x^*,y^*) \in A \times B$ is Stackelberg equilibrium if and only if:

$$f_A(x^*,y^*) = \inf \Pi_A(x,Y) \qquad x \in A \tag{10}$$

$$Y = \min \Pi_B(x,y) \qquad y \in B \tag{11}$$

3.3.1.a Backward induction

For solving this game theory problem we can find *Y* from Backward Induction method, which works as follows: since the leader will make the first move she knows that a rational follower will react by minimizing her payoff. The leader takes that into account before making the first move.

For solving the problem by using backward induction we should start from the follower's problem. First the fixed leader (say Operator 1) bid on the component and propose a price for that, then Operator 2 which is the follower with regard to the probability of failure time and preventive time and holding cost, will decide whether she wants to pay more and add extra *M* $ to the proposed price by the leader and attain the part or she prefers to not compete at that time. So first we suppose the follower knows the bidding cost and ordering time of the leader and optimize the follower cost function. Then we will substitute the optimum solutions for the follower in the leader's objective function to find the Nash Equilibrium.

3.3.2 Hierarchical Stackelberg-Nash using GA

Because of the computational complexity of dynamic programming to solve real large-scale problems and its weakness to solve such problems in a reasonable time, we apply a heuristic method to tackle the problem. This part presents a multi objective optimization model to find the optimal PM and replacement schedule. The obtained results from this approach will then compare to the results obtained from backward induction method.

Stackelberg solution is a hierarchical solution concept, so finding a Stackelberg solution

requires solving a hierarchical optimization problem. One possible heuristic for solving the problem is the GA that is suitable for solving complex optimization problems and, particularly, bi-objective programming problems. The GA is implemented in the two-player incentive Stackelberg game problem through the following procedure:

1. Calculation of $(T_i^*, T_{o(i)}^*)$, (the initial population for the players is provided with a random seeding in the leader's strategy space).
2. Choice of a population of K incentives $(\lambda_1, \ldots, \lambda_K)$ (note: λ is the player's strategy)
3. For $i = 1$ to K
    - The leader announces an incentive strategy λ_i.
    - The follower reacts to minimize his/her own cost function.
    - The leader performs his/her strategy.
    - The fitness of the incentive strategy λ_i is evaluated.
4. A new generation of solutions is created using the genetic operators.
5. Go to Step 2 and start of a new round if the termination condition is not met.

In this paper, we consider the probability for each of the operators to be the leader. This is accomplished by introducing a weight or distance into the fitness function. There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function. Determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the correct selection of the weights or utility functions to characterize the decision-makers preferences. In practice, it can be difficult to precisely and accurately select these weights. The second general approach is to determine an entire Pareto

optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other. Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems, since the final solution of the decision maker is always a trade-off between crucial parameters. Pareto optimal sets can be of varied sizes, and the size of Pareto set increases with the increase in the number of objectives.

3.2.2.a Joint decision-making considering priority

Weighted sum approach

The classical approach to solve a multi objective optimization problem is to assign a weight w_i to each normalized objective function so that the problem is converted to a single objective problem with a scalar objective function as follows:

$$\text{Fitness} = w_1 \Pi_1(X) + w_2 \Pi_2(X) + \ldots + w_m \Pi_m(X) \quad (12)$$

where $w_1, w_2, \ldots, w_m$ are nonnegative weights such that $w_1+w_2+\ldots+w_m=1$, $W = (w_1, w_2, \ldots, w_m)$ is a weight vector, and X is the vector of decision variables.

This approach is called a priori approach since the user is expected to provide the weights. If multiple solutions are desired, the problem should be solved multiple times with different weight combinations. The main difficulty with this approach is selecting a weight vector for each run.

In our codes, in order to show the probability of being leader we consider different

combinations of weights for the cost function. As the fitness function is equal to a weighted sum of respective objective functions, it will be dominated by the objective function assigned with a larger weight (i.e., if a fitness function is equal to a weighted sum of objective functions, it may be dominated by the objective functions with larger weights).

The fitness function (overall objective function) can be expressed as:

$$J_P\,(T_i, T_{o(i)}) = w_1 \Pi_1\,(T_1, T_{o(1)}) + w_2 \Pi_2\,(T_2, T_{o(2)}) \tag{13}$$

3.2.2.b Game with random leader-follower relationship

The overall objective function can be formulated as:

$$J_R = p_1\,(\Pi_1\,(T_1, T_{o(1)}) + \Pi_2\,(T_2, T_{o(2)})) + p_2\,(\Pi_1\,(T_1, T_{o(1)}) + \Pi_2\,(T_2, T_{o(2)})) \tag{14}$$

where p_1 is the probability for Operator 1 to be the leader, for which p_1+p_2 = 1. We assume that:

$$p_j = \frac{\frac{1}{\mathrm{E}[RUL_j]}}{\frac{1}{\mathrm{E}[RUL_j]}+\frac{1}{\mathrm{E}[RUL_k]}} = \frac{\mathrm{E}[RUL_k]}{\mathrm{E}[RUL_j]+\mathrm{E}[RUL_k]} \tag{15}$$

where E[RUL_j] is the expected *RUL* of the unit owned by Operator *j*:

$$\mathrm{E}[RUL_j] = \eta_j\,\Gamma\,(\frac{1}{\beta_j} + 1) \tag{16}$$

in which $\Gamma(n) = \int_0^\infty e^{-x}\,x^{n-1}\,dx$ is the gamma function.

3.4 Genetic Algorithm options

There are two ways to specify options for the genetic algorithm. We can write the codes in command line or use the optimization tool (optimtool). In this thesis, we directly use the optimization tool. In problem setup and results section, we should define our fitness function, number of variables and constraints. For fitness function simply we call our fitness weighted sum fitness function in the main code (@fitness).

In the right section of the page we can specify the options for the genetic algorithm code.

3.4.1 Population Options

In this section we can specify the data type of the input to the fitness function. You can set Population type to be one of the following:

- Double Vector: Use this option if the individuals in the population have type double. This is the default.
- Bit string: Use this option if the individuals in the population are bit strings.
- Custom: Use this option to create a population whose data type is neither of the preceding. If you use a custom population type, you must write your own creation, mutation, and crossover functions that accept inputs of that population type.

Population size

It specifies how many individuals there are in each generation. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing

the chance that the algorithm will return a local minimum that is not a global minimum. However, a large population size also causes the algorithm to run more slowly.

Creation function

In this part we can specify the function that creates the initial population for ga. You can choose from the following functions:

- Uniform: creates a random initial population with a uniform distribution. This is the default if there are no constraints or bound constraints.
- Feasible population: creates a random initial population that satisfies all bounds and linear constraints. It is biased to create individuals that are on the boundaries of the constraints, and to create well-dispersed populations. This is the default if there are linear constraints.
- Custom: enables you to write your own creation function, which must generate data of the type that you specify in Population type.

Initial population

This part specifies an initial population for the genetic algorithm. The default value is [], in which case ga uses the default Creation function to create an initial population. If you enter a nonempty array in the Initial population field, the array must have no more than Population size rows, and exactly Number of variables columns. In this case, the genetic algorithm calls a Creation function to generate the remaining individuals, if required.

Initial scores

We can specify initial scores for the initial population. The initial scores can also be partial.

Initial range

It specifies the range of the vectors in the initial population that is generated by a creation function. You can set Initial range to be a matrix with two rows and Number of variables columns, each column of which has the form [lb; ub], where lb is the lower bound and ub is the upper bound for the entries in that coordinate. If you specify Initial range to be a 2-by-1 vector, each entry is expanded to a constant row of length Number of variables.

3.4.2 Fitness Scaling Options

Fitness scaling converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for the selection function. You can specify options for fitness scaling in the Fitness scaling pane.

Scaling function: specifies the function that performs the scaling. The options are:

- Rank: The default fitness scaling function, Rank, scales the raw scores based on the rank of each individual instead of its score. The rank of an individual is its position in the sorted scores. An individual with rank r has scaled score proportional to. So the scaled score of the most fit individual is proportional to 1, the scaled score of the next most fit is proportional to, and so on. Rank fitness scaling removes the effect of the spread of the raw scores. The square root makes poorly ranked individuals more nearly equal in score, compared to rank scoring.

- Proportional: Proportional scaling makes the scaled value of an individual proportional to its raw fitness score.
- Top: Top scaling scales the top individuals equally. Selecting Top displays an additional field, Quantity, which specifies the number of individuals that are assigned positive scaled values. Quantity can be an integer between 1 and the population size or a fraction between 0 and 1 specifying a fraction of the population size. The default value is 0.4. Each of the individuals that produce offspring is assigned an equal scaled value, while the rest are assigned the value 0. The scaled values have the form [01/n 1/n 0 0 1/n 0 0 1/n ...].

3.4.3 Selection Options

Selection options specify how the genetic algorithm chooses parents for the next generation. You can specify the function the algorithm uses in the Selection function field in the Selection options pane. The options are:

- **Stochastic uniform:** The default selection function, stochastic uniform, lays out a line in which each parent corresponds to a section of the line of length proportional to its scaled value. The algorithm moves along the line in steps of equal size. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.
- **Remainder**: Remainder selection assigns parents deterministically from the integer part of each individual's scaled value and then uses roulette selection on the remaining fractional part. For example, if the scaled value of an individual is 2.3, that individual is listed twice as a parent because the integer part is 2. After

parents have been assigned according to the integer parts of the scaled values, the rest of the parents are chosen stochastically. The probability that a parent is chosen in this step is proportional to the fractional part of its scaled value.

- **Uniform**: Uniform selection chooses parents using the expectations and number of parents. Uniform selection is useful for debugging and testing, but is not a very effective search strategy.
- **Roulette**: Roulette selection chooses parents by simulating a roulette wheel, in which the area of the section of the wheel corresponding to an individual is proportional to the individual's expectation. The algorithm uses a random number to select one of the sections with a probability equal to its area.
- Tournament: Tournament selection chooses each parent by choosing Tournament size players at random and then choosing the best individual out of that set to be a parent. Tournament size must be at least 2. The default value of Tournament size is 4.

3.4.4 Reproduction Options

Reproduction options specify how the genetic algorithm creates children for the next generation:

- Elite count: specifies the number of individuals that are guaranteed to survive to the next generation. We should set Elite count to be a positive integer less than or equal to the population size. The default value is 2.
- Crossover fraction: specifies the fraction of the next generation, other than elite children, that are produced by crossover. Set Crossover fraction to be a fraction

between 0 and 1, either by entering the fraction in the text box or moving the slider. The default value is 0.8.

3.4.5 Mutation Options

Mutation options specify how the genetic algorithm makes small random changes in the individuals in the population to create mutation children. Mutation provides genetic diversity and enables the genetic algorithm to search a broader space. You can specify the mutation function in the Mutation function field in the Mutation options pane. You can choose from the following functions:

- Gaussian: The default mutation function, Gaussian, adds a random number taken from a Gaussian distribution with mean 0 to each entry of the parent vector. The standard deviation of this distribution is determined by the parameters Scale and Shrink, which are displayed when you select Gaussian, and by the Initial range setting in the Population options.
  - The Scale parameter determines the standard deviation at the first generation.
  - The Shrink parameter controls how the standard deviation shrinks as generations go by.
- Uniform: Uniform mutation is a two-step process. First, the algorithm selects a fraction of the vector entries of an individual for mutation, where each entry has a probability Rate of being mutated. The default value of Rate is 0.01. In the second step, the algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.

- Adaptive Feasible: randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation. The feasible region is bounded by the constraints and inequality constraints. A step length is chosen along each direction so that linear constraints and bounds are satisfied.
- Custom enables you to write your own mutation function.

3.4.6 Crossover Options

Crossover options specify how the genetic algorithm combines two individuals, or parents, to form a crossover child for the next generation.

The following functions are provided in the toolbox:

- Scattered: the default crossover function, creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child.
- Single point: chooses a random integer n between 1 and Number of variables and then
  - Selects vector entries numbered less than or equal to n from the first parent.
  - Selects vector entries numbered greater than n from the second parent.
  - Concatenates these entries to form a child vector.
- Two points: selects two random integers m and n between 1 and Number of variables. The function selects
  - Vector entries numbered less than or equal to m from the first parent
  - Vector entries numbered from m+1 to n, inclusive, from the second parent
  - Vector entries numbered greater than n from the first parent.

- Intermediate: creates children by taking a weighted average of the parents. You can specify the weights by a single parameter, Ratio, which can be a scalar or a row vector of length Number of variables. The default is a vector of all 1's. The function creates the child from parent1 and parent2 using the following formula.
- Heuristic: returns a child that lies on the line containing the two parents, a small distance away from the parent with the better fitness value in the direction away from the parent with the worse fitness value. You can specify how far the child is from the better parent by the parameter Ratio, which appears when you select Heuristic. The default value of Ratio is 1.2. If parent1 and parent2 are the parents, and parent1 has the better fitness value, the function returns the child
- Arithmetic: creates children that are the weighted arithmetic mean of two parents. Children are always feasible with respect to linear constraints and bounds.
- Custom enables you to write your own crossover function.

Mutation and Crossover

The genetic algorithm uses the individuals in the current generation to create the children that make up the next generation. Besides elite children, which correspond to the individuals in the current generation with the best fitness values, the algorithm creates:

- Crossover children by selecting vector entries, or genes, from a pair of individuals in the current generation and combines them to form a child
- Mutation children by applying random changes to a single individual in the current generation to create a child

Both processes are essential to the genetic algorithm. Crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. Mutation adds to the diversity of a population and thereby increases the likelihood that the algorithm will generate individuals with better fitness values.

3.4.7 Stopping Criteria Options

Stopping criteria determine what causes the algorithm to terminate. You can specify the following options:

- Generations: Specifies the maximum number of iterations for the genetic algorithm to perform. The default is 100.
- Time limit: Specifies the maximum time in seconds the genetic algorithm runs before stopping.
- Fitness limit: The algorithm stops if the best fitness value is less than or equal to the value of Fitness limit.
- Stall generations: The algorithm stops if the weighted average change in the fitness function value over Stall generations is less than Function tolerance.
- Stall time limit: The algorithm stops if there is no improvement in the best fitness value for an interval of time in seconds specified by Stall time.
- Function tolerance: The algorithm runs until the cumulative change in the fitness function value over Stall generations is less than or equal to Function Tolerance.
- Nonlinear constraint tolerance: The Nonlinear constraint tolerance is not used as stopping criterion. It is used to determine the feasibility with respect to nonlinear constraints.

3.4.8 Plot function

Plot options enable us to plot data from the genetic algorithm while it is running. When you select plot functions and run the genetic algorithm, a plot window displays the plots on separate axes. You can select any of the following plot functions in the Plot functions pane:

- Best fitness: plots the best function value versus generation.
- Expectation: plots the expected number of children versus the raw scores at each generation.
- Score diversity: plots a histogram of the scores at each generation.
- Stopping: plots stopping criteria levels.
- Best individual: plots the vector entries of the individual with the best fitness function value in each generation.
- Genealogy: plots the genealogy of individuals. Lines from one generation to the next are color-coded as follows:
  - Red lines indicate mutation children.
  - Blue lines indicate crossover children.
  - Black lines indicate elite individuals.
- Scores: plots the scores of the individuals at each generation.
- Max constraint: plots the maximum nonlinear constraint violation at each generation.

- Distance: plots the average distance between individuals at each generation.
- Range: plots the minimum, maximum, and mean fitness function values in each generation.
- Selection: plots a histogram of the parents

Differences between gamultiobj and ga:

The syntax and options for gamultiobj are similar to those for ga, with the following differences:

- gamultiobj does not have nonlinear constraints, so its syntax has fewer inputs.
- gamultiobj takes an option DistanceMeasureFcn, a function that assigns a distance measure to each individual with respect to its neighbors.
- gamultiobj takes an option ParetoFraction, a number between 0 and 1 that specifies the fraction of the population on the best Pareto frontier to be kept during the optimization. If there is only one Pareto frontier, this option is ignored.
- gamultiobj uses only the Tournament selection function.
- gamultiobj uses elite individuals differently than ga. It sorts noninferior individuals above inferior ones, so it uses elite individuals automatically.
- gamultiobj has only one hybrid function, fgoalattain.
- gamultiobj does not have a stall time limit.
- gamultiobj has different plot functions available.
- gamultiobj does not have a choice of scaling function.

CHAPTER IV

4. Numerical Example

Wind turbine industry has gained a remarkable stand in the US industry since the turn of the century. Studying the reliability of wind turbines is a critical factor in the success of related projects. The wind energy industry typically uses reactive maintenance approach or run-to-failure maintenance. This form of maintenance has been shown to be the most costly Operations and Maintenance (O&M) practice available to operators.

According to the gearbox's reputation for a high failure rate, one of the biggest concerns remaining in the wind industry is the reliability of the gearbox. In this work, The Weibull distribution was chosen due to the good representation provided for components under aging effects (increasing failure rate).

Prognostics and timely maintenance of components are critical to the continuing operation of a wind farm. To maximize the power generation of the wind farm, limited maintenance resources with uncertainty must be appropriately dealt with based on the current health status of wind turbines. This numerical example will show the application of the proposed models for two gearboxes in two wind turbines.

In order to illustrate the model numerically and the proposed solution procedure, we used data set presented in Table 1. The mathematics formulations fully coded in MATLAB 7.12 was run on a Sony VAIO computer, with an Intel Pentium processor operating at 2.30 GHz and 6 GB of RAM. In addition, we set the GA parameters as presented in Table

2. MATLAB Genetic Algorithm and direct search toolbox is used to obtain Pareto-optimal solutions to this problem as well as to define the fitness functions.

Table 1. Parameters for the the case 1

	η (year)	B	M ($)	τ_1 (year)	τ_2 (year)	τ_3 (year)	$c_{p(i)}$ ($/year)	$c_{f(i)}$ ($/year)	$c_{h(i)}$ ($/year)	c_0 ($)
Gearbox 1	**3**	**3**	**10000 , 2000**	**2.08**	**0.41**	**0.83**	**12500**	**25000**	**5000**	**20000**
Gearbox 2	**2.5**	**2.5**	**10000 , 2000**	**2.08**	**0.41**	**0.83**	**12500**	**25000**	**5000**	**20000**

4.1 Computational results

4.1.1 Case 1: Stackelberg game

In this example, Because of the complexity of this decision-making problem, it is difficult, if not impossible, to foresee the behavior of objective functions. One viable tool for overcoming this challenge is the space-filling experimental design method that aids in getting information about the entire strategy space [77][78]. In particular, a Constrained Maximin Design [78] is used to select typical strategies that cover the leader's strategy space under the constraints on the decision variables. Such a sampling scheme maximizes the minimum distance between two design points. Let $x_2^{(q)} = \left[x_{21}^{(q)}, x_{22}^{(q)}\right] = \left[T_2^{(q)}, T_{o(2)}^{(q)}\right], q \in \{1,2,\dots,K\}$, be the *n* design points (i.e., the leader's strategies) within the leader's feasible strategy space *A*. These strategies can be determined by solving the following optimization problem:

$$\max \min_{\substack{x_l, x_m \in X \\ s,t \in \{1,2,\ldots,K\}, s<t}} \left\| x_2^{(s)} - x_2^{(t)} \right\|_2 \tag{17}$$

$$\text{Subject to} \quad x_{21}^{(q)} > x_{22}^{(q)}, \text{ for all } q \in \{1,2,\ldots,K\},$$

where $\left\| x_2^{(s)} - x_2^{(t)} \right\|_2 = \sqrt{\left(x_{21}^{(s)} - x_{21}^{(t)}\right)^2 + \left(x_{22}^{(s)} - x_{22}^{(t)}\right)^2)}$ is the Euclidean distance of two design points $x_2^{(s)}$ and $x_2^{(t)}$. It is equivalent to solving:

$$\max d_{min} \tag{18}$$

$$\text{Subject to} \quad d_{min} \leq \left\| x_2^{(s)} - x_2^{(t)} \right\|_2, \quad \text{for all } 1 \leq s < t \leq K$$

$$x_{21}^{(q)} > x_{22}^{(q)}, \text{ for all } q \in \{1,2,\ldots,K\}.$$

In this example, twenty levels for each of T_2 and $T_{o(2)}$ are considered. Fig. 5 shows the resulting design with K = 20 design points (d_{min}^*= 3.60555).

Leader's Strategy Space (discrete)

Maximin design to discretize the leader's strategy space

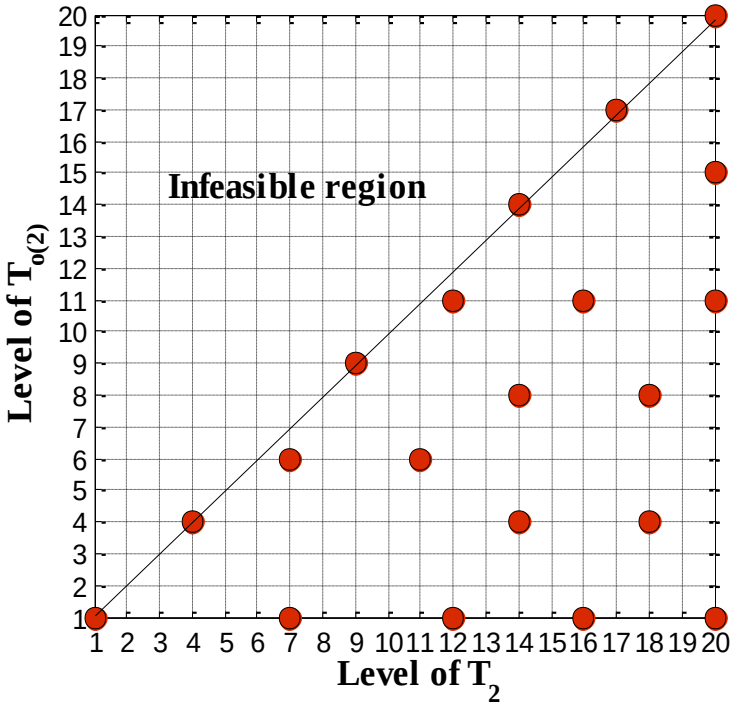

Figure 8. 20 levels for each decision variable

Tables 1 and 2 show the results for two cases. In the first case (Table1) we assumed M=10000 and based on the calculations, in this case operator 1 (The follower) prefers to wait more and not to pay the extra M charge to take the part. This shows the downtime cost is not greater than M. In the second case (Table 2), We changed the value of M (M=2000) and optimized the players' objective functions again. In this case the downtime cost will be greater than M and the follower prefers to pay M extra dollors to take the part first. The red rectangular show the Nash Equilibrium solution of the game and no player can benefit by changing his or her strategy while the other player keep his/her unchanged

Table 2. Case1 Results (M=10000)

Design point in the leader's strategy space	1	2	3	4	5	6	7	8	9	10
$x_{21}^{(q)} = T_2^{(q)}$	16	1	4	12	7	7	9	11	12	14
$x_{22}^{(q)} = T_{o(2)}^{(q)}$	1	1	4	1	1	6	9	6	11	14
$x_{11}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_1^{(q)*}$	18	14	20	18	18	20	20	20	20	20
$x_{12}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_{o(1)}^{(q)*}$	4	4	11	4	4	11	11	11	15	15
The follower's objective* ($)	8920	6727	1740	5576	5382	2528	2761	3094	3155	4431
The leader's corresponding objective ($)	8432	89352	123509	47414	90652	123946	105633	70948	67924	35217
Design point in the leader's strategy space	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
$x_{21}^{(q)} = T_2^{(q)}$	20	14	17	14	20	18	18	16	20	20
$x_{22}^{(q)} = T_{o(2)}^{(q)}$	1	8	17	4	11	4	8	11	15	20
$x_{11}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_1^{(q)*}$	18	20	20	20	20	20	20	20	20	-
$x_{12}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_{o(1)}^{(q)*}$	4	15	20	11	15	11	11	15	20	-
The follower's objective* ($)	8727	2209	5578	3081	3342	2470	3944	4058	5669	-
The leader's corresponding objective ($)	1096	40858	14347	37503	1879	4815	5857	15384	2476	-

Table 3. Case1 Results (M=2000)

Design point in the leader's strategy space	1	2	3	4	5	6	7	8	9	10
$x_{21}^{(q)} = T_2^{(q)}$	16	1	4	12	7	7	9	11	12	14
$x_{22}^{(q)} = T_{o(2)}^{(q)}$	1	1	4	1	1	6	9	6	11	14
$x_{11}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_1^{(q)*}$	18	18	20	18	18	20	20	20	20	20
$x_{12}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_{o(1)}^{(q)*}$	4	4	11	4	4	11	11	11	15	15
The follower's objective* ($)	5839	5661	2906	5822	5911	2908	2875	2862	3668	3645
The leader's corresponding objective ($)	10720	89840	125570	41980	89380	121090	101380	73400	67900	37640

Design point in the leader's strategy space	11	12	13	14	15	16	17	18	19	20
$x_{21}^{(q)} = T_2^{(q)}$	20	14	17	14	20	18	18	16	20	20
$x_{22}^{(q)} = T_{o(2)}^{(q)}$	1	8	17	4	11	4	8	11	15	20
$x_{11}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_1^{(q)*}$	18	20	20	20	20	20	20	20	20	-
$x_{12}^{(q)*}(x_{21}^{(q)}, x_{22}^{(q)}) = T_{o(1)}^{(q)*}$	4	11	20	11	15	11	11	15	20	-
The follower's objective* ($)	5775	2883	4664	2909	3588	2878	2941	3682	4699	-
The leader's corresponding objective ($)	12300	32540	13000	32540	2020	5650	5650	17460	2400	-

4.1.2 Selected options in Genetic Algorithm toolbox

Population size

The Population size field in population options determines the size of the population at each generation. Increasing the population size enables the genetic algorithm to search more points and thereby obtain a better result. However, the larger the population size, the longer the genetic algorithm takes to compute each generation. As the proposed model is complicated with lots of variables and it takes a long time to compute the integral parts, we determine this section by 10 individuals in each generation.

Fitness Scaling

Top scaling is used in this part because it restricts parents to the fittest individuals and creates less diverse populations than rank scaling which is the default option.

Selection

A more deterministic selection option is **Remainder**, which performs two steps:

- In the first step, the function selects parents deterministically according to the integer part of the scaled value for each individual. For example, if an individual's scaled value is 2.3, the function selects that individual twice as a parent.
- In the second step, the selection function selects additional parents using the fractional parts of the scaled values, as in stochastic uniform selection. The function lays out a line in sections, whose lengths are proportional to the

fractional part of the scaled value of the individuals, and moves along the line in equal steps to select the parents.

Note that if the fractional parts of the scaled values all equal 0, as can occur using top scaling, the selection is entirely deterministic.

The other parts are kept as the default.

Table 4. Parameters for the the case 2 and 3

	η (year)	B	M ($)	τ_1 (year)	τ_2 (year)	τ_3 (year)	$c_{p(i)}$ ($/year)	$c_{f(i)}$ ($/year)	$c_{h(i)}$ ($/year)	c_0 ($)
Gearbox 1	**8**	**3**	**10000**	**0.013**	**0.0027**	**0.01**	**360000**	**720000**	**144000**	**20000**
Gearbox 2	**7.7**	**2.5**	**10000**	**0.013**	**0.0027**	**0.01**	**360000**	**720000**	**144000**	**20000**

Table 5. Parameters of the Genetic Algorithm

Parameter	Value
Number of generations	50
Population size	10
Scaling function	Top
Selection function	Remainder
Crossover function	Intermediate

4.1.3 Case 2: Joint decision-making considering priority

As stated before weighted sum method is the most common approach to multi-objective optimization models. Minimizing of the joint objective function is sufficient for Pareto optimality [79]. We consider different combination of weights and run GA for weighted sum fitness function for the set of weights for both objective functions and achieved non-

inferior solutions for the optimum PM time and ordering time are presented in Table 3. Up to the decision maker each of these combinations can be used.

Table 6. Non-inferior solutions for different combinations of weights

Weights		Decision variables				Objective function
w_1	w_2	T_1^* (days)	T_2^* (days)	$T_{o(1)}^*$ (days)	$T_{o(2)}^*$ (days)	J_P^* (\$)
0.1	0.9	307	283	281	255	31,585.51
0.2	0.8	303	281	270	266	34,772.62
0.3	0.7	284	230	240	193	24,285.87
0.4	0.6	248	230	241	195	24,103.73
0.6	0.4	303	317	193	225	26,261.99
0.7	0.3	313	315	200	222	25,375.76
0.8	0.2	314	328	259	262	27,098.98
0.9	0.1	310	306	226	229	25,944.22

4.1.4 Case 3: Game with random leader-follower relationship

For case 3 we defined the formula for the remaining useful life time and the proportion of them as the P_i in our Matlab codes. By the use of P_i we converted multi objective optimization model to single-objective model. In this case each generation took about 2 hours to be completed. The computed PM replacement time for operator 1 is after 344.92 days and for Operator 2 is after 342.83 days. The best time to order the part for Operator 1 is after 235.42 days and for Operator 2 is after 336.16 days. So this scenario recommends it's better for Operator1 to pay more and get the part before Operator 2. Although the preventive time for Operator 1 is after Operator 2 but based on the provided numbers Operator 1 should get the part first in order to prevent the downtime cost due to unexpected failures. Figs. 9 shows the Genetic Algorithm optimization process and the optimum solutions. Best fitness diagram plots the best function value versus 50

generations. Best individual diagram plots the vector entries of the individual with the best fitness function value in each generation. Genealogy plots the genealogy of individuals. Lines from one generation to the next are color-coded as follows:

- Red lines indicate mutation children.
- Blue lines indicate crossover children.
- Black lines indicate elite individuals (As the default for the number of elite children in each generation is 2 you see two black dots in each generation.)

And Distance diagram plots the average distance between individuals at each generation. High values of distance show high diversity in population and vice versa.

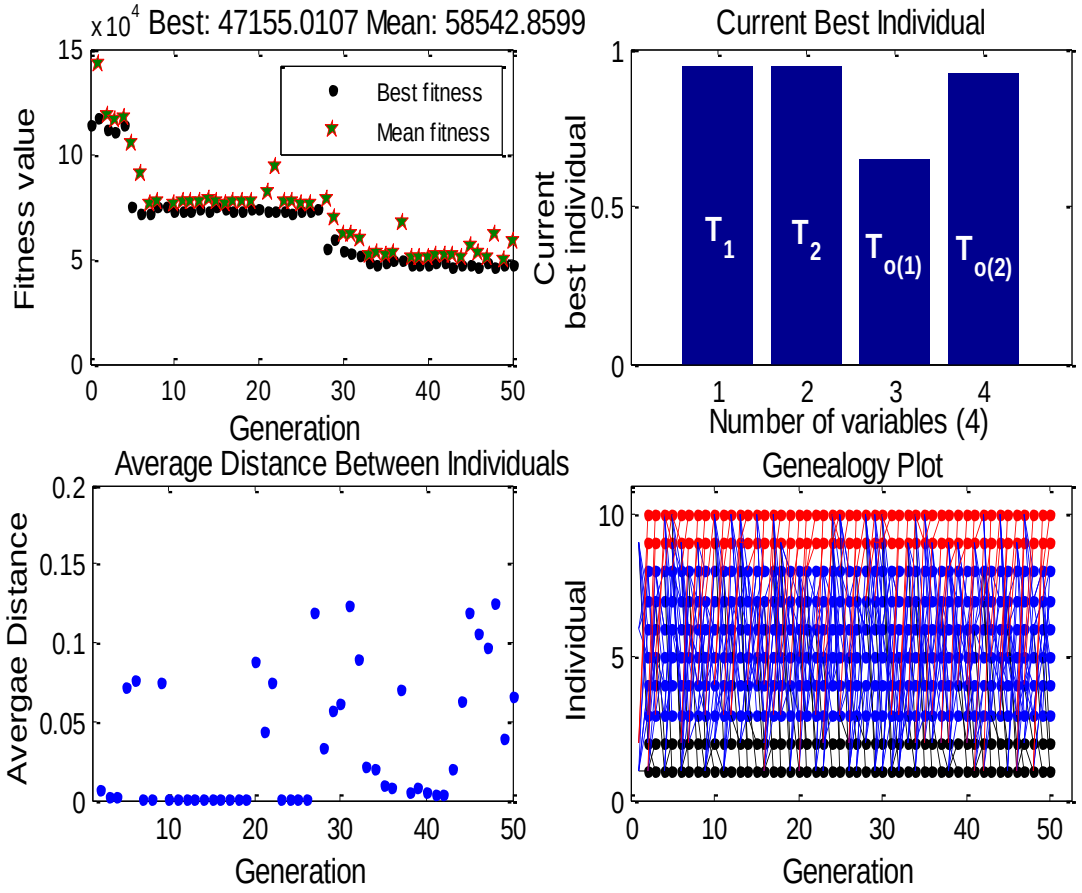

Figure 9. GA diagram for scenario 3

CHAPTER V

5. Conclusions and Recommendations

In this thesis, we presented a new approach based upon game theoretic models for PM and replacement scheduling of two competitive systems. These models seek to find the best time to ordering the service part and performing PM subject to minimizing the total cost. Three different cases were defined and utilized as the solution procedures to achieve the best non-inferior solutions (Pareto optimal solutions) and GA was utilized to solve the models. By analyzing the computational results of each algorithm with each fitness function, we could show the efficiency and effectiveness of algorithms and fitness functions. The developed models in this thesis can be applied in a wide variety of industries. In this work, the provided numerical example which is based on real numbers from wind turbine gearbox reliability databases will be considered as the application of developed model. Future work in this area is needed to investigate the models for n-player games, single-leader multiple-follower Stackelberg game as well as other techniques to solve the optimization problem and estimating key model parameters.

LIST OF REFERENCES

1. Elsayed, E.A., 1996. Reliability Engineering, Addison-Wesley, New York, NY.
2. Wang, H., 2002. A survey of maintenance policies of deteriorating systems. European Journal of Operational Research 139 (3), 469-489.
3. Ilgin, M.A., Tunali, S., Joint optimization of spare parts inventory and maintenance policies using genetic algorithms, International Journal of Advanced Manufacturing Technology 34(5-6) (2007) 594–604.
4. Vaurio, J.K., On time-dependent availability and maintenance optimization of standby units under various maintenance policies, Reliability Engineering & System Safety 56 (1) (1997) 79–89.
5. Lonardo, P., Anghinolfi, D., Paolucci, M., Tonelli, F., A stochastic linear programming approach for service parts optimization, CIRP Annals - Manufacturing Technology 57(1) (2008) 441–444.
6. Gross, R., Miller, D., Soland, R., On common interests among reliability,inventory and queuing, IEEE Transactionson Reliability 34(3) (1985) 204–208.
7. Kumar, U.D., Crocker, J., Knezevic, J., El-Haram, M., Reliability, Maintenance and Logistic Support: A Life Cycle Approach, Kluwer Academic; 2000.
8. Kabir, A.B.M., Farrash, S.H.A., Simulation of an integrated age replacement and spare provisioning policy using SLAM, Reliability Engineering and System Safety 52(2) (1996) 129–138.
9. Park, Y.T., Park,K.S., Generalized spare ordering policies with random lead time, European Journal of Operational Research 23 (1986) 320–30.
10. Brezavscek, A., Hudoklin, A., Joint optimization of block-replacement and periodic-review spare-provisioning policy. IEEE Transactions on Reliability 52(1) (2003) 112-117.
11. Huang, R., Meng, L., Xi, L., Liu, C.R., Modeling and Analyzing a Joint Optimization Policy of Block-Replacement and Spare Inventory With Random-Leadtime, IEEE Transactions on Reliability 57(1) (2008) 113 - 124

12. Chelbi, A., At-Kadi, D., Spare provisioning strategy for preventively replaced systems subjected to random failure, International Journal of Production Economics 74(1) (2001) 183–189.
13. Kolahan, F., Sharifinya, A., Combinatorial Part sequencing and tool replacement based reliability by heuristic algorithms, Journal of Amirkabir, 18(66-B) (2007) 35-44.
14. Wong, J.Y.F., Chung, D.W.C., Ngai, B.M.T., Banjevic, D., Evaluation of spares requirements using statistical and probability analysis techniques. Transactions of Mechanical Engineering, I.E. Aust (22) (1997) 77-84.
15. Louit, D., Pascual, R., Banjevic, D., Optimization models for critical spare parts inventories-a reliability approach, Journal of the Operational Research Society 62 (2011) 992–1004
16. Kennedy, W.J., Patterson, J.W., Fredendall, L.D., An overview of recent literature on spare parts inventories, International Journal of Production Economics 76(2) (2002) 201–215.
17. Nosoohi, I., Hejazi, S.R., A multi-objective approach to simultaneous determination of spare part numbers and preventive replacement times, Applied Mathematical Modelling, 35(3) (2011) 1157-1166.
18. Sim, S.H., Endrenyi, J., Optimal preventive maintenance with repair, IEEE Transactions on Reliability 37 (1) (1988) 92–96.
19. Chen, M., Feldman, R.M., 1997. Optimal replacement policies with minimal repair and age-dependent costs. European Journal of Operational Research 98 (1), 75-84.
20. Panagiotidou, S., Tagaras, G., 2007. Optimal preventive maintenance for equipment with two quality states and general failure time distributions. European Journal of Operational Research 180 (1), 329-353.
21. Yeh, R.H., Chen, M., Lin, C., 2007. Optimal periodic replacement policy for repairable products under free-repair warranty. European Journal of Operational Research 176 (3), 1678-1686.
22. Dehayem Nodem, F.I., Kenne, J.P., Gharbi, A., 2009. Hierarchical decision

making in production and repair/replacement planning with imperfect repairs under uncertainties. European Journal of Operational Research 198 (1), 173-189.

23. Berg, M.P., 1995. The marginal cost analysis and its application to repair and replacement policies. European Journal of Operational Research 82 (2), 214-224.
24. Zohrul Kabir, A.B.M., Al-Olayan, A.S., 1996. A stocking policy for spare part provisioning under age based preventive replacement. European Journal of Operational Research 90 (1), 171-181.
25. Vaughan, T.S., 2005. Failure replacement and preventive maintenance spare parts ordering policy. European Journal of Operational Research 161 (1), 183-190.
26. Wang, L., Chu, J., Mao, W., 2009. A condition-based replacement and spare provisioning policy for deteriorating systems with uncertain deterioration to failure replacement. European Journal of Operational Research 194 (1), 184-205.
27. Wang,W., 2012. A stochastic model for joint spare parts inventory and planned maintenance optimization. European Journal of Operational Research 216 (1), 127-139.
28. Berrichi, A., Yalaoui, F., Amodeo, L., Mezghiche, M., Bi-Objective ant colony optimization approach to optimize production and maintenance scheduling, Computers & Operations Research 37 (9) (2010) 1584–1596.
29. Moradi, E., Fatemi Ghomi, S.M.T., Zandieh, M., Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem, Expert Systems with Applications 38 (6) (2011) 7169–7178.
30. Quan, G., Greenwood, G.W., Liu, D., Hu, S., 2007. Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms. European Journal of Operational Research 177 (3), 1969-1984.
31. Herabat, P., Tangphaisankun, A., Multi-Objective Optimization Model using Constraint-Based Genetic Algorithms for Thailand Pavement Management, Journal of the Eastern Asia Society for Transportation Studies 6 (2005) 1137 - 1152.

32. Certa, A., Galante, G., Lupo, T., Passannanti, G., Determination of Pareto frontier in multi-objective maintenance optimization, Reliability Engineering and System Safety 96(7) (2011) 861–867.
33. Liao, H., Elsayed, E. A. and Chan, L.-Y, Maintenance of continuously monitored degrading systems, European Journal of Operational Research, 175 (2006) (821-835).
34. Jayakumar, A., Asagarpoor, S., Maintenance optimization of equipment by linear programming, Probability in the Engineering and Information Science 20(1) (2006) 183-193.
35. Duarte, J., Soares, C., Optimization of the preventive maintenance plan of a series components system, Reliability: Theory & Applications (Special Issue) 2(3-4) (2007) 33-39.
36. Tam, A.S.B., Chan, W.M., Price, J.W.H., Optimal maintenance intervals for multi-component system, Production Planning and Control 17(8) (2006) 769-779.
37. Shirmohammadi, A.H., Zhang, Z.G., Love, E., A computational model for determining the optimal preventive maintenance policy with random breakdowns and imperfect repairs, IEEE Transactions on Reliability 52(2) (2007) 332-339.
38. Konak, A., Coit, D.W., Smith, .A.E., Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engineering and System Safety 91 (9) (2006) 992-1007.
39. Canfield, R.V., Cost optimization of periodic preventive maintenance, IEEE Transactions on Reliability 35 (1) (1986) 78-81.
40. Madanat, S., Ben-Akiva, M., Optimal inspection and repair policies for infrastructure facilities, Transportation Science 28(1) (1994) 55–62.
41. Holland, J.H., Adaptation in Natural and Artificial Systems, 2nd edition, Cambridge, MA: MIT Press; 1992.
42. Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA: Addison-Wesley; 1989.

43. Kančev, D., Gjorgiev, B., Čepin, M., Optimization of test interval for ageing equipment: A multi-objective genetic algorithm approach, Journal of Loss Prevention in the Process Industries 24 (4) (2011) 397-404.
44. Munõz, A., Martorell, S., Serradell, V., Genetic algorithms in optimizing surveillance and maintenance of components, Reliability Engineering & System Safety 57 (2) (1997) 107-120.
45. Tsai, Y.T., Wang, K.S., Teng, H.Y., Optimizing preventive maintenance for mechanical components using genetic algorithms, Reliability Engineering & System Safety 74 (1) (2001) 89-97.
46. Chen, Y.S., Yang, W.N., Weng. C.J., A Study of Preventive Maintenance Policy in Age Reduction Model, in: Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference, 2004.
47. Marseguerra, M., Zio, E. and Podofillini, L., Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation, Reliability Engineering & System Safety 77(2) (2002) 151-165.
48. Usher, J.S., Kamal, A.H., Syed, W.H., Cost optimal preventive maintenance and replacement scheduling, IIE Transactions 30 (12) (1998) 1121-1128.
49. Levitin, G., Lisnianski, A., Optimization of imperfect preventive maintenance for multi-state systems, Reliability Engineering and System Safety 67(2) (2000) 193-203.
50. Levitin, G., Lisnianski, A., Optimal replacement scheduling in multi-state series-parallel systems, Quality and Reliability Engineering 16(2) (2000) 157-62.
51. Wang, Y., Handschin, E., A new genetic algorithm for preventive unit maintenance scheduling of power systems. International Journal of Electrical Power and Energy Systems, 22(5) (2000) 343-348.
52. Cavory, G., Dupas, R., Goncalves, G., A genetic approach to the scheduling of preventive maintenance tasks on a single product manufacturing production line. International Journal of Production Economics 74(1-3) (2001) 135-46.
53. Leou, R.C., A new method for unit maintenance scheduling based on genetic algorithm. IEEE Power Engineering Society General Meeting (1) (2003) 246-251.

54. Han, B.J., Pan, J., Fan, X.M., Ma, D.Z., Optimization of preventive maintenance scheduling for production machine of production system in finite time horizon, Journal of Donghua University (English Edition) 21(1) (2004) 112-116.
55. Limbourg, P., Kochs, H.D., Preventive maintenance scheduling by variable dimension evolutionary algorithms, International Journal of Pressure Vessels and Piping 83(4) (2006) 262-269.
56. Wang, C-H., Lin, T-W., Optimizing minimize periodic preventive maintenance model for series-parallel systems based on particle swarm optimization, in Computers and Industrial Engineering (CIE), 2010 40th International Conference.
57. Samrout, M., Yalaoui, F., Chatelet, E., Chebbo, N., New methods to minimize the preventive maintenance cost of series-parallel systems using ant colony optimization, Reliability Engineering and System Safety 89 (3) (2005) 346-354.
58. Bris, R., Chatelet, E., Yalaoui, F., New method to minimise the preventive maintenance cost of series–parallel systems, Reliab Eng Syst Saf 82 (2003) 247–55.
59. Myerson, R.B., Game Theory: Analysis of Conflict, Harvard University Press, Cambridge Massachusetts (1991).
60. Nash, J.F., Equilibrium points in N-person games, Proceedings of the National Academy of Sciences of the United States of America 36(1) (1950) 48-49.
61. Kim, J.H., Park, J.B., Park, J.K., Chun, Y.H., Generating unit maintenance scheduling under competitive market environments, Electrical Power and Energy Systems 27 (3) (2005) 189–194.
62. Cachon, G., Netessine, S., Game theoretic applications in supply chain analysis, in Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era (International Series in Operations Research & Management Science), Simchi-Levi, D., Wu, S.D., Shen, Z. (Eds.), 2004.
63. Chen, Y., Multimedia Social Networks: Game Theoretic Modeling and Equilibrium Analysis. Ph.D. dissertation, University of Maryland, 2011.

64. Basar, T., Olsder, G. J., Dynamic Non cooperative Game Theory, Academic Press, Second Edition, 1995 (SIAM Classics in Applied Mathematics, number 23, 1998).
65. Vallée, T., Basar, T., Incentive Stackelberg Solutions and the Genetic Algorithm, in: The International Symposium of Dynamic Games and Applications, 1998, pp. 633-639.
66. Vallée, T., Basar, T., Off-line computation of Stackelberg solutions with the genetic algorithm, Computational Economics 13 (3) (1999) 201-209.
67. Nedim, M.A., Sirakaya, S., On-line computation of Stackelberg equilibria with synchronous parallel genetic algorithms, Economic Dynamics and Control 27 (8) (2003) 1503-1515.
68. D'Amato, E., Daniele, E., Mallozzi, L., Petrone, G., Hierarchical Multi-follower Decision Making Models with Genetic Algorithms and Applications, 3 rd International Conference on the Dynamics of Information Systems, University of Florida (2011)
69. Stensgaard, H., Aqrensen, J., Reliability-based design of wind turbine blades, Structural Safety, 2011.
70. Guo, H.T., Watson, S.J., Tavner, P.J., Jiangping Xiang, Reliability analysis for wind turbines with incomplete failure data collected from after the date of initial installation, Reliability Engineering and System Safety 94(6) (2009) 1057–1063.
71. Arabian-Hoseynabadi, H., Oraee, H., Tavner, P.J., Wind turbine productivity considering electrical subassembly reliability, Renewable Energy 35(1) (2010) 190–197.
72. Cohen, J.M., A Methodology for Computing Wind Turbine Cost of Electricity Using Utility Economic Assumptions, in Windpower 89 Proceedings (1989) San Francisco, California.
73. Walford, C.A., Wind Turbine Reliability: Understanding and Minimizing Wind Turbine Operation and Maintenance Costs (2006) Sandia National Laboratories.

74. Hill, R., Stinebaugh, J., Briand, D., Sandia National Laboratories Dr. Benjamin. A., Linsday, J., ARES Corporation, Wind Turbine Reliability: A Database and Analysis Approach (2008) Sandia National Laboratories.
75. Iuga, D., European Wind Energy Association. Available from: URL: http://www.wind-energy-the-facts.org.
76. Manwell J.F., McGowan, J., Rogers A.L., Wind Energy Explained: Theory, Design and Application. 2nd Edition. John Wiley & Sons, Inc; 2002.
77. Petelet, M., Iooss, B., Asserin, O., Loredo, A., 2010. Latin hypercube sampling with inequality constraints. Advances in Statistical Analysis 54 (4), 325-339.
78. Stinstra, E., Den Hertog, D.D., Stehouwer, P., Vestjens, A., 2003. Constrained maximin designs for comupter experiments. Technometrics 45 (4), 340-346.
79. Zadeh, L.A., Optimality and non-scalar-valued performance criteria, IEEE Trans. Autom. Control AC, 8 (1963) 59-60.

APPENDIX

Matlab Codes

Case 1:

```
% Failure Function Parameters (Weibull Distribution)
etta1=3;       %Scale parameter
betta1=3;      %Shape parameter
etta2=2.5;
betta2=2.5;
%-------------------------------------------------------------------------
M=2000;          %Extra charge for bidding on a part

tao1=2.08;       %Time for replenishment
tao2=0.41;       %Time to perform preventive maintenance
tao3=0.83;       %Time to perform corrective replacement (tao3 > tao2)

P1=12500;        %Unit downtime cost due to preventive maintenance ($ per day)
P2=12500;
C1=25000;        %Unit downtime cost due to a failure
C2=25000;
Ch1=5000;        %Unit holding cost
Ch2=5000;
co=20000;        %Regular ordering cost
%-------------------------------------------------------------------------
% to1=1;
% t1=4.5;
% t2=0.25;
% to2=0.25;
load exam.txt
t1=exam(:,1);
t2=exam(:,1);
```

```
to1=exam(:,2);
to2=exam(:,2);
for ii=1:20
   for jj=1:20

%-------------------------------------------------------------------------
for i=1:500,
tf1(i)=wblrnd(etta1,betta1,1,1);
tf2(i)=wblrnd(etta2,betta2,1,1);
%-------------------------------------------------------------------------
% RUL1=etta1*gamma(1/betta1 + 1);   %Remaining useful life
% RUL2=etta2*gamma(1/betta2 + 1);
%-------------------------------------------------------------------------

%if RUL2<RUL1
if (to2(ii)<to1(jj)) && (to1(jj)-to2(ii)<tao1)
   ro=1;
   tw1=max(tao1+to2-to1(jj),0)+tao1;
   dcost1=C1*(max(tw1+to1(jj)-tf1(i),0)+tao3);
   if dcost1>M,
      delta=1;
   else
      delta=0;
   end

   tw1=ro*((1-delta)*max(tao1+to2(ii)-to1(jj),0)+tao1)+(1-ro)*tao1;
   tw2=ro*(delta*max(tao1+to1(jj)-to2(ii),0)+tao1)+(1-ro)*tao1;
%%%%
   dcostp1=P1*(max(tw1+to1(jj)-t1(jj),0)+tao2);
   hcostp1=Ch1*(max(t1(jj)-tw1-to1(jj),0));
```

```
   dcostu1=C1*(max(tw1+to1(jj)-tf1(i),0)+tao3);
   hcostu1=Ch1*(max(tf1(i)-tw1-to1(jj),0));

   dcostp2=P2*(max(tw2+to2(ii)-t2(ii),0)+tao2);
   hcostp2=Ch2*(max(t2(ii)-tw2-to2(ii),0));
   dcostu2=C2*(max(tw2+to2(ii)-tf2(i),0)+tao3);
   hcostu2=Ch2*(max(tf2(i)-tw2-to2(ii),0));

   cp1=dcostp1+hcostp1+co+delta*M;
   cu1=dcostu1+hcostu1+co+delta*M;
   cp2=dcostp2+hcostp2+co;
   cu2=dcostu2+hcostu2+co;


 %---------------------------------------------------------------------


 X1(i)=  cp2*(exp(-(t2(ii)/etta2)^betta2))+  cu2*((betta2/etta2)*(t2(ii)/etta2)^(betta2-
1)*exp(-(t2(ii)/etta2)^betta2));

 Y1(i)=  cp1*(exp(-(t1(jj)/etta1)^betta1))+cu1*((betta1/etta1)*(t1(jj)/etta1)^(betta1-
1)*exp(-(t1(jj)/etta1)^betta1));
 else
    X1(i)=inf;
    Y1(i)=inf;
 end


end
X(ii,jj)=mean(X1);
```

```
Y(ii,jj)=mean(Y1);
   end
end
X
Y
```

Case 2:

```
%Fitness Function for both Objective Functions

function fcombined = fitness2(x)
% x = [t1 t2 to1 to2];

t1 = x(1);      %Preventive time of gearbox 1
t2 = x(2);      %Preventive time of gearbox 2
to1= x(3);      %Ordering time of gearbox 1
to2= x(4);      %Ordering time of gearbox 2
%-------------------------------------------------------------------------
% Failure Function Parameters (Weibull Distribution)
etta1=8;        %Scale parameter
betta1=3;       %Shape parameter
etta2=7.7;
betta2=2.5;
%-------------------------------------------------------------------------
M=10000;        %Extra charge for bidding on a part
tao1=0.013;     %Time for replenishment
tao2=0.0027;    %Time to perform preventive maintenance
tao3=0.01;      %Time to perform corrective replacement (tao3 > tao2)
P1=360000;      %Unit downtime cost due to preventive maintenance ($ per day)
```

```
P2=360000;
C1=720000;        %Unit downtime cost due to a failure
C2=720000;
Ch1=144000;       %Unit holding cost
Ch2=144000;
co=20000;         %Regular ordering cost
%-------------------------------------------------------------------------
for i=1:200,
tf1=wblrnd(etta1,betta1,1,1);
tf2=wblrnd(etta2,betta2,1,1);
%-------------------------------------------------------------------------
if to2<to1,
  if to1-to2<tao1
    ro=1;
    tw1=max(tao1+to2-to1,0)+tao1;
    dcost1=C1*(max(tw1+to1-tf1,0)+tao3);
    if dcost1>M,
       delta=1;
    else
       delta=0;
    end
  end
    tw1=ro*((1-delta)*max(tao1+to2-to1,0)+tao1)+(1-ro)*tao1;
    tw2=ro*(delta*max(tao1+to1-to2,0)+tao1)+(1-ro)*tao1;
%%%%
dcostp1=P1*(max(tw1+to1-t1,0)+tao2);
hcostp1=Ch1*(max(t1-tw1-to1,0));
dcostu1=C1*(max(tw1+to1-tf1,0)+tao3);
hcostu1=Ch1*(max(tf1-tw1-to1,0));
```

```
dcostp2=P2*(max(tw2+to2-t2,0)+tao2);
hcostp2=Ch2*(max(t2-tw2-to2,0));
dcostu2=C2*(max(tw2+to2-tf2,0)+tao3);
hcostu2=Ch2*(max(tf2-tw2-to2,0));

cp1=dcostp1+hcostp1+co+delta*M;
cu1=dcostu1+hcostu1+co+delta*M;
cp2=dcostp2+hcostp2+co;
cu2=dcostu2+hcostu2+co;

else
  if to2-to1<tao1
    ro=1;
    tw2=max(tao1+to1-to2,0)+tao1;
    dcost2=C2*(max(tw2+to2-tf2,0)+tao3);
    if dcost2>M,
        delta=1;
    else
        delta=0;
    end
  end
    tw2=ro*((1-delta)*max(tao1+to1-to2,0)+tao1)+(1-ro)*tao1;
    tw1=ro*(delta*max(tao1+to2-to1,0)+tao1)+(1-ro)*tao1;
%%%%
dcostp1=P1*(max(tw1+to1-t1,0)+tao2);
hcostp1=Ch1*(max(t1-tw1-to1,0));
dcostu1=C1*(max(tw1+to1-tf1,0)+tao3);
hcostu1=Ch1*(max(tf1-tw1-to1,0));

dcostp2=P2*(max(tw2+to2-t2,0)+tao2);
```

```
hcostp2=Ch2*(max(t2-tw2-to2,0));
dcostu2=C2*(max(tw2+to2-tf2,0)+tao3);
hcostu2=Ch2*(max(tf2-tw2-to2,0));

cp1=dcostp1+hcostp1+co;
cu1=dcostu1+hcostu1+co;
cp2=dcostp2+hcostp2+co+delta*M;
cu2=dcostu2+hcostu2+co+delta*M;
end
  %------------------------------------------------------------------------
  syms tf1
  f(1)=  (cp1*(exp(-(t1/etta1)^betta1))+int(cu1*((betta1/etta1)*(tf1/etta1)^(betta1-1)*exp(-
(tf1/etta1)^betta1)),tf1,0,t1));

  syms tf2
  f(2)=  (cp2*(exp(-(t2/etta2)^betta2))+int(cu2*((betta2/etta2)*(tf2/etta2)^(betta2-1)*exp(-
(tf2/etta2)^betta2)),tf2,0,t2));

w1=0.4;
w2=0.6;
 fcombined=w1*f(1)+w2*f(2);
end
```

Case **3:**

```
 %Fitness Function for both Objective Functions
function fcombined = fitness3(x)

% x = [t1 t2 to1 to2];

t1 = x(1);       %Preventive time of gearbox 1
t2 = x(2);       %Preventive time of gearbox 2
to1= x(3);       %Ordering time of gearbox 1
to2 = x(4);      %Ordering time of gearbox 2
%-------------------------------------------------------------------------
% Failure Function Parameters (Weibull Distribution)
etta1=8;         %Scale parameter
betta1=3;        %Shape parameter
etta2=7.7;
betta2=2.5;
%-------------------------------------------------------------------------
M=10000;         %Extra charge for bidding on a part
tao1=0.013;      %Time for replenishment
tao2=0.0027;     %Time to perform preventive maintenance
tao3=0.01;       %Time to perform corrective replacement (tao3 > tao2)
P1=360000;       %Unit downtime cost due to preventive maintenance ($ per day)
P2=360000;
C1=720000;       %Unit downtime cost due to a failure
C2=720000;
Ch1=144000;      %Unit holding cost
Ch2=144000;
co=20000;        %Regular ordering cost
%-------------------------------------------------------------------------
```

```
for i=1:200,
tf1=wblrnd(etta1,betta1,1,1);
tf2=wblrnd(etta2,betta2,1,1);
%-------------------------------------------------------------------------
RUL1=etta1*gamma(1/betta1 + 1);   %Remaining useful life
RUL2=etta2*gamma(1/betta2 + 1);
%-------------------------------------------------------------------------
if RUL2<RUL1
  if abs(to1-to2)<tao1
    ro=1;
    tw1=max(tao1+to2-to1,0)+tao1;
    dcost1=C1*(max(tw1+to1-tf1,0)+tao3);
    if dcost1>M,
       delta=1;
    else
       delta=0;
    end

    tw1=ro*((1-delta)*max(tao1+to2-to1,0)+tao1)+(1-ro)*tao1;
    tw2=ro*(delta*max(tao1+to1-to2,0)+tao1)+(1-ro)*tao1;
%%%%
    dcostp1=P1*(max(tw1+to1-t1,0)+tao2);
    hcostp1=Ch1*(max(t1-tw1-to1,0));
    dcostu1=C1*(max(tw1+to1-tf1,0)+tao3);
    hcostu1=Ch1*(max(tf1-tw1-to1,0));

    dcostp2=P2*(max(tw2+to2-t2,0)+tao2);
    hcostp2=Ch2*(max(t2-tw2-to2,0));
    dcostu2=C2*(max(tw2+to2-tf2,0)+tao3);
    hcostu2=Ch2*(max(tf2-tw2-to2,0));
```

```
    cp1=dcostp1+hcostp1+co+delta*M;
    cu1=dcostu1+hcostu1+co+delta*M;
    cp2=dcostp2+hcostp2+co;
    cu2=dcostu2+hcostu2+co;

  else
      fcombined=inf;
      return
  end
 %-------------------------------------------------------------------------
elseif RUL1<RUL2
  if abs(to2-to1)<tao1
    ro=1;
    tw2=max(tao1+to1-to2,0)+tao1;
    dcost2=C2*(max(tw2+to2-tf2,0)+tao3);
    if dcost2>M,
        delta=1;
    else
        delta=0;
    end

    tw2=ro*((1-delta)*max(tao1+to1-to2,0)+tao1)+(1-ro)*tao1;
    tw1=ro*(delta*max(tao1+to2-to1,0)+tao1)+(1-ro)*tao1;
%%%%
    dcostp1=P1*(max(tw1+to1-t1,0)+tao2);
    hcostp1=Ch1*(max(t1-tw1-to1,0));
    dcostu1=C1*(max(tw1+to1-tf1,0)+tao3);
    hcostu1=Ch1*(max(tf1-tw1-to1,0));
```

```
    dcostp2=P2*(max(tw2+to2-t2,0)+tao2);
    hcostp2=Ch2*(max(t2-tw2-to2,0));
    dcostu2=C2*(max(tw2+to2-tf2,0)+tao3);
    hcostu2=Ch2*(max(tf2-tw2-to2,0));

    cp1=dcostp1+hcostp1+co;
    cu1=dcostu1+hcostu1+co;
    cp2=dcostp2+hcostp2+co+delta*M;
    cu2=dcostu2+hcostu2+co+delta*M;

  else
      fcombined=inf;
      return
  end
 else
     fcombined = inf;
    return
end
  %---------------------------------------------------------------------
 syms tf1
  f(1)=   (cp1*(exp((t1/etta1)^betta1))+int(cu1*((betta1/etta1)*(tf1/etta1)^(betta1-1)*exp(-
(tf1/etta1)^betta1)),tf1,0,t1));

 syms tf2
  f(2)=   (cp2*(exp((t2/etta2)^betta2))+int(cu2*((betta2/etta2)*(tf2/etta2)^(betta2-1)*exp(-
(tf2/etta2)^betta2)),tf2,0,t2));

 p1=RUL2/(RUL1+RUL2);
 p2=RUL1/(RUL2+RUL1);
```

```
 fcombined=p1*(f(1)+f(2))+p2*(f(2)+f(1));
end
```

Genetic Algorithm Code:

```
 function [x,fval,exitflag,output,population,score] = untitled(nvars,lb,ub,TimeLimit_Data)
% This is an auto generated MATLAB file from Optimization Tool.

% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options,'TimeLimit', TimeLimit_Data);
options = gaoptimset(options,'CrossoverFcn', {  @crossoverintermediate [] });
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', { @gaplotbestf @gaplotbestindiv
@gaplotdistance @gaplotexpectation @gaplotgenealogy @gaplotrange
@gaplotscorediversity @gaplotscores @gaplotselection @gaplotstopping
@gaplotmaxconstr });
options = gaoptimset(options,'OutputFcns', { [] });
[x,fval,exitflag,output,population,score] = ...
ga(@fitness3,nvars,[],[],[],[],lb,ub,[],options);
```

VITA

Faranak Fathi Aghdam is a graduate research assistant under Dr. Haitao Liao at the Industrial and Information Engineering of University of Tennessee, Knoxville. She plans to graduate from the University of Tennessee with a Master of Science degree in Industrial and Information Engineering in December 2011. Faranak received a Bachelor of Science degree in Industrial Engineering in 2010 from the University of Science and Technology (Tehran/IRAN).

Conference Presentations:

- Liao. H., Fathi Aghdam. F., Niknam. S.A., Predictive maintenance and service logistics for wind turbine fleet, IERC 2011 Conference.
- Fathi Aghdam. F., Prognostics-Based Two-Operator Competition for Maintenance and Service Part Logistics, INFORMS 2011 Conference.

Papers:

- Fathi Aghdam. F., Liao. H., Prognostics-Based Two-Operator Competition for Maintenance and Service Part Logistics, **working paper**.

Email Address: faranak.fathi@utk.edu