2015

SCHEDULE OPTIMIZATION WITH FLEXIBLE DURATIONS

Kevin R. Roy
University of Rhode Island, nivek10k@yahoo.com

SCHEDULE OPTIMIZATION WITH FLEXIBLE DURATIONS

BY

KEVIN R. ROY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2015

MASTER OF SCIENCE

OF

KEVIN R. ROY

APPROVED:

Thesis Committee:

Major Professor Manbir Sodhi

David Taggart

Cenk Undey

Nasser Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2015

ABSTRACT

Cost, quality, and time are the pillars of a popular and well known business model. What is also well known about this model is that it is very difficult to be the industry leader in all three categories. This is because these categories are naturally interconnected such that one category cannot be changed without affecting the others. This creates an environment where a balance must be established to determine the appropriate amount of effort to focus on each of these three categories.

This current research presents an application for the schedule optimization of operations with flexible durations. The objective of the problem is to find the optimal sequence of jobs to minimize cost, where cost is a function of makespan and work center processing durations. The constraints in the problem include variable job durations and a sequential flow of jobs through multiple successive work centers. Details of the implemented tools chosen in support of the optimization are described followed by a summary of the accuracy, robustness, and scalability of the proposed system on generic case studies.

Final results indicate the proposed approach can successfully be applied to scheduling optimization with flexible job processing times. Implementation of this approach proves to be accurate, robust, and scalable when evaluated against other approaches. The system is implemented via a user friendly graphical interface.

ACKNOWLEDGMENTS

Throughout my academic experience and writing this thesis, many have offered guidance, all of who deserve thanks. However the encouragement of a select few has led to the completion of this work.

Dr. Manbir Sodhi – your support and guidance throughout my undergraduate and graduate terms. Thank you for presenting me with a problem and allowing me the opportunity to develop my capability of learning. I have no doubt I would not be where I am today if we had never crossed paths and for that, I can never thank you enough.

Arash Nashrollahishirazi – your guidance and advisement truly broadened my academic horizon relating to python and genetic algorithms.

My family – your loving support has helped me through writing the chapters of this work. Without your love and encouragement, none of this would have been possible.

Amgen Inc. - your financial support during my graduate term.

TABLE OF CONTENTS

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1

INTRODUCTION

When used effectively, schedule optimization can be very valuable to industry. Two goals strived in manufacturing are to make products less expensive and at a faster production rate because often the customer wants the product as soon as possible and at the lowest price. The customer will likely purchase from a competitor if they feel they can get the product faster or for a lower price. Due to the complexities of manufacturing systems, schedule optimization is computationally expensive and difficult to implement. Some of the complexities of implementing schedule optimization are not considered within the scope of this study including work center downtime, labor capacity, product changeover tasks, interruptible operations, product demand fluctuations, job priorities, and resource utilization.

The design of the problem will include jobs that will be processed through work centers (WCs). There are a total of *j* jobs to be processed, *s* work centers in series, and *p* work centers in parallel. Each job must sequentially go through every serial work center. A job may sit idle between work centers if there is no subsequent work center available. Figure 1 is a graphical illustration of the design of the scheduling optimization problem with flexible durations. Case studies will be evaluated with different quantities of *j*, *s*, and *p*. The components of profit will include functions of makespan and work center process durations. The total profit will be evaluated to determine the quality of the solution.

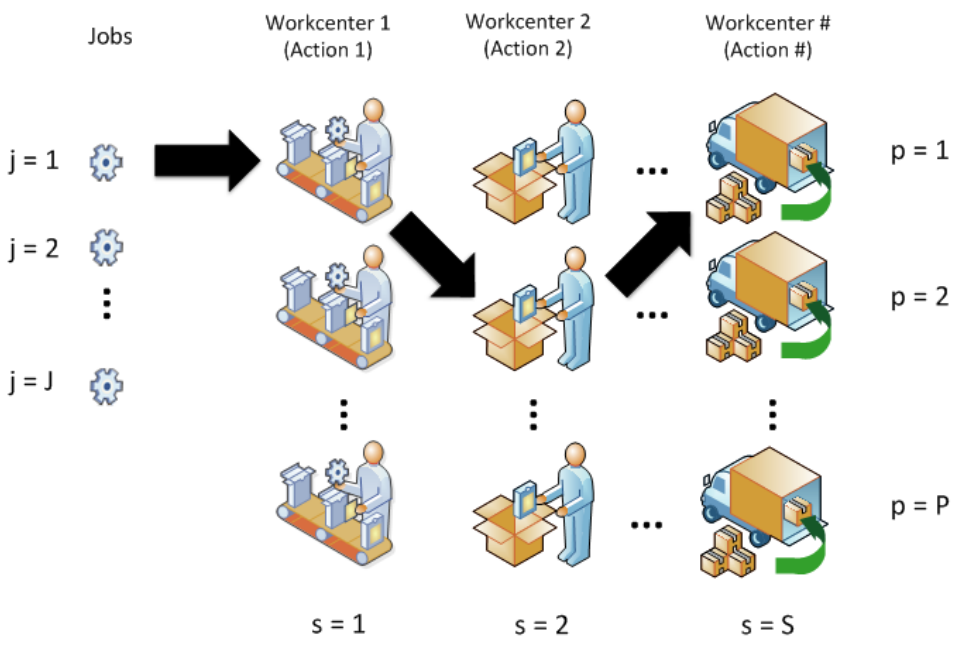

Figure 1. Graphical representation of the problem design.

For manufacturing systems observed by the author, typically there is one order of magnitude quantity of work centers in parallel, one or two orders of magnitude quantity of work centers in series, and anywhere between one to numerous orders of magnitude quantity of jobs. Scheduling optimization becomes increasingly computationally demanding as the size of a manufacturing system increases.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Schedule Optimization

There has been much research related to schedule optimization of manufacturing systems. Balasubramanian and Grossmann (2003) used a mixed integer linear programming (MILP) technique to optimize the minimum makespan of tasks with uncertainty in processing time durations. The approach presented by Balasubramanian and Grossmann worked well where the solution was more heavily structured by certain types of constraints but the models did not perform as well when tested against general problems. Li and Ierapetritou (2008) also evaluated schedule optimization with uncertainty in processing time durations but used three formulations of robust optimization (RO). The drawback of the approach taken by Li and Ierapetritou's was that it did not optimize an expected value objective function but rather promised to enforce the feasibility for the entire predefined uncertainty space of the problem. Ant colony optimization was presented by Kumar et. al. (2003) as an effective method of scheduling jobs in a flexible manufacturing system (FMS) but with non-flexible task durations. One of the most popular schedule optimization methods presented in this literature review is genetic algorithm (GA) optimization. Guo et. al. (2008) evaluated GA optimization with tardiness and earliness penalties built into the objective function which performed effectively when evaluated against two-order scheduling problems. Gao et. al. (2006) assessed GA optimization with machine availability constraints and Fanti et. al. (1998) studied GA optimization with a multi-criteria objective function by

weighing desired criteria such as makespan, resource utilization, and work in progress (WIP). To the author's knowledge, no specific work relating to schedule optimization with flexible processing durations using GA optimization or any other optimization method is published in literature.

2.2 Model Research Applications

There does not seem to be any published research devoted to improving schedule optimization with operations that have flexible durations. This may be due to the fact that scheduling optimization with static work center durations is by itself challenging thus allowing work center durations to be dynamic only increases the difficulty to find the optimal solution. It could also be a consequence of assuming that most of the cost and speed benefits are obtained by solving a scheduling problem with static durations however, under certain circumstances it may be desirable to allow the optimization algorithm to determine task durations to increase profit margins. The remainder of this paper will examine this problem, evaluate the performance of a GA approach against case studies, and conclude with a summary of the results.

CHAPTER 3

METHODOLOGY

3.1 Process Simulation and Modeling

Simulations provide the user with a means to better understand a system by allowing the user to conduct experiments and analyze a system without having to deal with the complications inherent with many real world experiments. Simulation package design can range between being tailored for a specific process with specific equipment to having a generic purpose with generic equipment. The advantage to having the more specific simulation package is the developer can spend less time creating the framework for the simulation and more time with the details unique to the system. The disadvantage to having the more specific simulation package is that the user is typically limited to the constraints of the pre-established framework and thus can lose flexibility. In contrast, a generic simulation package generally allows the user more flexibility for development yet typically requires additional time to create the framework. Ideally, an off the shelf simulation package should provide the user adequate customizability to add, remove, or modify features of the simulation with minimal development effort.

3.2 Components of Profit

One component of profit (as defined by this paper) is the cost to operate work centers. Maintenance is a significant input to the cost to operate work centers. Scheduled equipment maintenance (also known as preventative maintenance) is

generally correlated with the quantity of occurrences of using a piece of equipment (also known as cycles or intervals). Preventative maintenance is typically scheduled prior to the wear zone which is the point where the probability of corrective maintenance (also known as unscheduled maintenance) becomes increasingly (sometimes exponentially) more likely to occur. Figure 2 shows the general correlation between equipment failure rate and equipment interval, including examples of preventative and corrective maintenance occurrences relative to equipment interval and equipment failure rate.

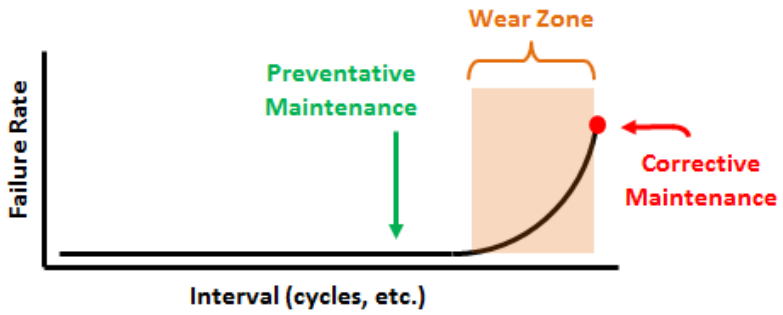

Figure 2. Correlation of maintenance and equipment usage cycles.

Boeing conducted a statistical analysis to evaluate the economics of conducting maintenance on their commercial airplanes (McLoughlin, Doulatshahi and Onorati 2011). This analysis was used to determine the optimal number of cycles (flight hours) to minimize costs associated with preventative maintenance and corrective maintenance. Figure 3 shows an economic analysis of maintenance to determine the optimal maintenance interval for Boeing commercial airplanes by evaluating the total cost of scheduled and unscheduled maintenance.

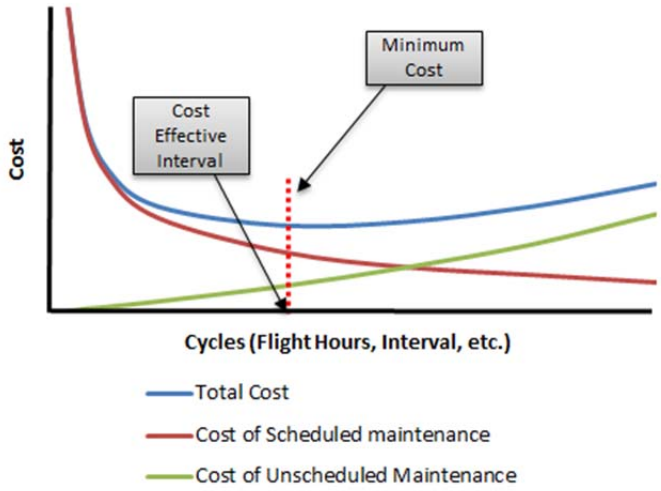

Figure 3. Economic analysis of Boeing commercial airplane maintenance.

This paper will assume maintenance will occur at the cost optimal quantity of cycles with no direct correlation to time. Given maintenance is correlated to equipment cycles, the cost of maintenance increases per unit of time as the number of cycles increase per that same unit of time. In addition to this maintenance cost, the paper will also assume labor and utilities is a linear addition to the work center cost. Therefore as a work center production rate increases, so does the cost to run that work center.

Work center cost constitutes half the profit equation whereas the other half comprises of the makespan cost. The components of makespan cost are more indirect than the components of work center cost. The carrying cost of inventory and stock out cost significantly impact the cost of makespan. Some of the components of carrying cost of inventory include taxes, insurance, depreciation, and inventory cost. The cost of inventory is money that is tied up in unfinished product which cannot be used to invest elsewhere until the product is sold (known as cost of capital or the opportunity cost of the money). Therefore, reducing the amount of unfinished product by reducing

makespan duration will liquidate inventory assets and allow cash to be invested in something else, such as mutual funds, treasuries, or even money market accounts. Reducing the amount of unfinished product by reducing makespan duration will also reduce taxes, insurance, and depreciation associated with work-in-progress inventory. Shorter makespan durations also reduce the risk of long stock out durations because raw materials can be turned into finished product faster. The cost of long stock out durations is a risk based calculation and is correlated with the inventory level of finished goods.

In practice, the cost equations for work centers and for makespan may be a challenge to define and will likely vary between industries and between companies. However for this paper, the cost equations for work centers and for makespan will be evaluated as quadratic and exponential.

3.3 Area of Optimization

Optimization is an attempt to select the best solution out of a collection of solutions, generally bounded by constraints. One area of optimization is schedule optimization which is an area of operations research that can be used widely among commercial industry. Almost every company has to do some form of scheduling in one way or another. Not every scheduling operation necessarily warrants formulating the problem and solving it using an optimization algorithm but when the problem becomes complex and having a sub-optimal solution may significantly impact cost or speed, then it may be appropriate to leverage schedule optimization.

3.4 P VS. N

The types of optimization problems are commonly categorized between P and NP. P is any type of problem that can be solved in polynomial time and NP is any type of problem that can be solved by a non-deterministic Turing machine in polynomial time (Garey and Johnson 1979). All P problems are NP problems ($P \in NP$) but not all NP problems are P problems. It takes significantly more time to solve an NP type problem than it does to describe the problem so consequently large NP type problems are typically computationally expensive (Garey and Johnson 1979). There are sub classes for NP type problems including NP-hard, NP-complete, NP-easy, and NP-equivalent. Job-shop schedule optimization typical falls into the NP-complete category (Garey and Johnson 1979) (Garey, Johnson and Sethi 1976) (Gonzalez and Sahni 1978) (Lenstra, Kan and Brucker 1977). Figure 4 is a Venn diagram showing the relationship between P, NP, and NP-complete.

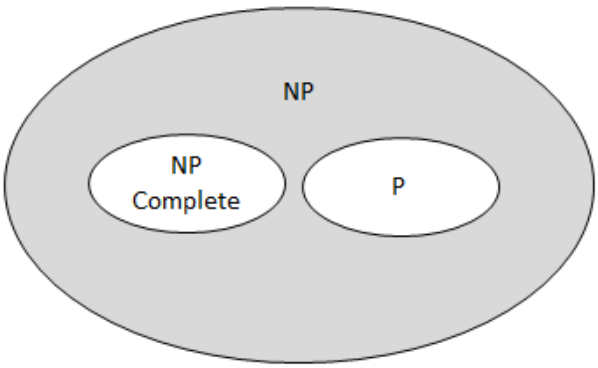

Figure 4. Venn diagram showing the relationship between P, NP, and NP-complete.

3.5 Mixed Integer Program Formulation

A formulation is presented in the next subsections for the problem of schedule optimization with flexible durations. The minimum degree of complexity of this

formulation is quadratic as a consequence of the product of the inputs parameters and decision variables. This is one of many ways to formulate this problem and depending on how the problem is formulated it may impact the appropriate optimization algorithm to use. For example, there may exist a formulation that replaces the input parameters start and end times with durations, such is the case for the GA approach discussed later in the paper.

3.5.1 Variables and Inputs

Sets and indices:

$j = job\ index\ where\ j = \{1, 2, 3, \ldots, J\}$

$p = work\ center\ in\ parallel\ index\ where\ p = \{1, 2, 3, \ldots, P\}$

$s = work\ center\ in\ series\ index\ where\ s = \{1, 2, 3, \ldots, S\}$

Input parameters:

$et_{max} = Maximum\ end\ time\ for\ all\ jobs.$

$c_s(st, et) = Cost\ function\ for\ the\ work\ center\ in\ series\ s.$

$C(et) = Cost\ function\ for\ the\ makespan.$

Decision variables:

$st_{j,p,s} = Start\ time\ for\ job\ j\ performed\ in\ WC\ in\ parallel\ p\ and\ in\ series\ s.$

$et_{j,p,s} = End\ time\ for\ job\ j\ performed\ in\ WC\ in\ parallel\ p\ and\ in\ series\ s.$

$$x_{j,p,s} = \begin{cases} 1, if\ job\ j\ will\ be\ performed\ by\ WC\ in\ parallel\ p\ and\ in\ series\ p. \\ 0, if\ job\ j\ will\ not\ be\ performed\ by\ WC\ in\ parallel\ p\ and\ in\ series\ s. \end{cases}$$

3.5.2 Objective function

The objective function is the equation that the optimization algorithm is going to attempt to maximize or in this case minimize by adjusting the variables. The following is the objective function of the formulation.

Minimize:

$$\sum_{s=1}^{S}\sum_{p=1}^{P}\sum_{j=1}^{J} c_s\left(et_{j,p,s} - st_{j,p,s}\right)x_{j,p,s} + C\left(\sum_{p=1}^{P} x_{J,p,S}et_{J,p,S}\right)$$

3.5.3 Constraints

The constraints establish bounds on the variables included in the objective function. The following are the constraints of the formulation.

$$\sum_{p=1}^{P} x_{j,p,s} = 1, \forall j, \forall s \quad \text{(a)}$$

$$x_{j,p,s}st_{j,p,s} \leq x_{j,p,s}et_{j,p,s}, \forall j, \forall p, \forall s \quad \text{(b)}$$

$$\sum_{p=1}^{P} x_{j,p,s}et_{j,p,s} \leq \sum_{p=1}^{P} x_{j,p,s+1}st_{j,p,s+1}, \forall j, \forall s \quad \text{(c)}$$

$$x_{j,p,s}et_{j,p,s} \leq x_{j+1,p,s}st_{j+1,p,s}, \forall j, \forall p, \forall s \quad \text{(d)}$$

$$0 \leq st_{j,p,s} \leq et_{max}, \forall j, \forall p, \forall s \quad \text{(e)}$$

$$0 \leq et_{j,p,s} \leq et_{max}, \forall j, \forall p, \forall s \quad \text{(f)}$$

$$\sum_{p=1}^{P} x_{j,p,S}et_{j,p,S} \leq \sum_{p=1}^{P} x_{j+1,p,S}et_{j+1,p,S}, \forall j \quad \text{(g)}$$

Constraint (a) ensures only one work center can be used per job per operation (work center in series). The duration of any operation must be greater than or equal to zero which is enforced through constraint (b). Constraint (c) ensures the end of an operation must finish before the start of the next work center in series. To ensure the end of one operation must finish before the start of the next operation at the same work center, constraint (d) was included in the formulation. Constraint (e) ensures no

job can start before time zero and no job can start after the maximum end time. Similarly, constraint (f) ensures no job can end before time zero and no job can end after the maximum end time. The last constraint (g) ensures the end time for job j cannot be later than the end time for job $j + 1$. Constraint (g) is enforced through constraint (d) when there is only one work center in parallel but is required when there are more than one work centers in parallel.

3.6 Optimization Techniques

There are a vast number of optimization techniques available, some being more appropriate for certain problems than others. Some of these techniques include simplex, combinatorial, iterative methods (e.g. quasi-newton, interior point, gradient descent, etc.), and heuristics (e.g. genetic, hill climbing, ant colony, etc.). Selecting an appropriate optimization technique to obtain a desired level of accuracy and robustness is very much dependent on the class of the problem. It's important to note that not every optimization technique will necessarily provide the optimal solution to a problem but rather an approximate solution relative to optimal.

3.7 Algorithm Selection

As mentioned previously, job-shop scheduling typically falls into the complete combinatorial optimization problem which is a category of problem that is challenging to find a method capable of providing the optimal solution. Therefore, it is appropriate to look for heuristic methods for finding a time schedule solution for the problem. In studies, genetic algorithms have been shown to be efficient and robust in comparison with other methods for solving these scheduling problems because of its stochastic nature (Chen, Ihlow and Lehmann 1999) (Biegel and Davern 1990). The author

decided to leverage the genetic algorithm for the problem discussed in this paper and implementation of this algorithm is presented in the next subsection.

3.8 Algorithm Implementation

The first step for genetic algorithm is to randomly generate *N* number of time duration ($d_{j,p,s}$ where $d_{j,p,s} = et_{j,p,s} - st_{j,p,s}$) vectors to make the population. The durations are for a particular job on a specific work center based on a permutation schedule. A permutation schedule is where the jobs are sequenced (i.e. processing order) the same way for all work centers. These duration vector elements (also known as chromosomes) are then sorted from smallest to largest (shown in Figure 5) and represent the duration for each job for every work center in series.

Figure 5. GA chromosome of work center durations.

Figure 6 illustrates an example of three jobs performed by two work centers in series and two in parallel. The makespan time to finish all jobs is *T* (where $T = \max_{1 \le p \le P} et_{J,p,S}$).

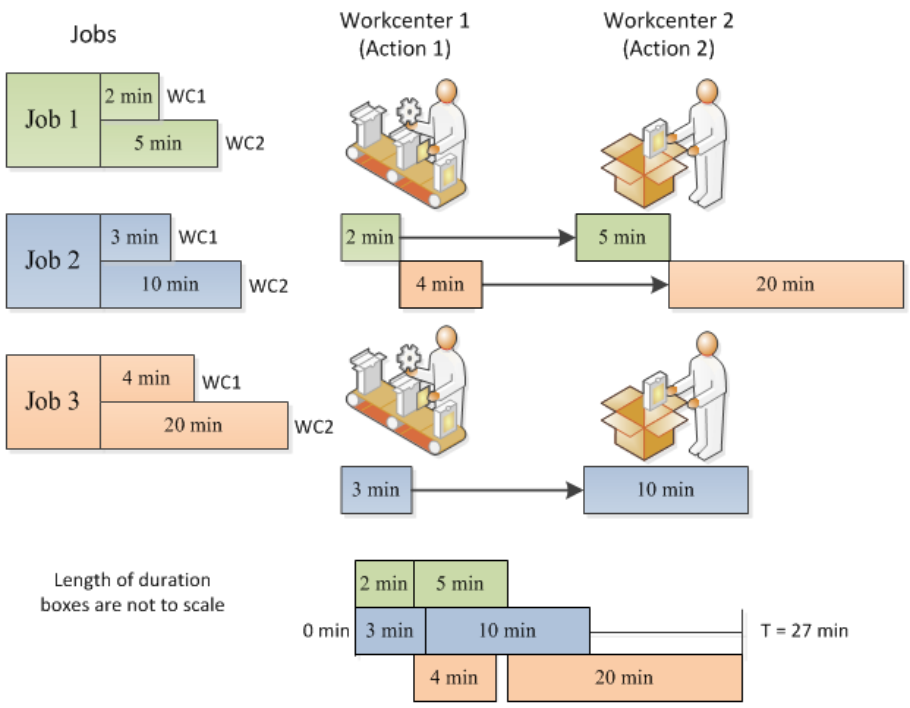

Figure 6. Graphical representation of an iteration within example problem.

Main genetic processes begin after chromosomes generation. The genetic process iteratively adjusts the work center durations in such a manner that it produces desired results (i.e. lower cost). Crossover (also known as homologous recombination) and mutation are two main processes in natural genetics. Meiosisrate is the size of the randomly selected subset of the main population used to perform crossover. Meiosisrate is defined as:

$$Meiosisrate = (n|\, n \leq N) \quad N{:}\, population\ size$$

Homologous recombination is the biological way for exchanging genetic material between parent chromosomes. The genetic diversity will increase by exchanging more genetic information. Figure 7 shows the crossover process in the algorithm for two parent chromosomes. Crossover occurs on two randomly selected chromosomes from

the meiosisrate subset and this pair of randomly selected chromosomes is called parent chromosomes. There is a quantity of M parent chromosomes.

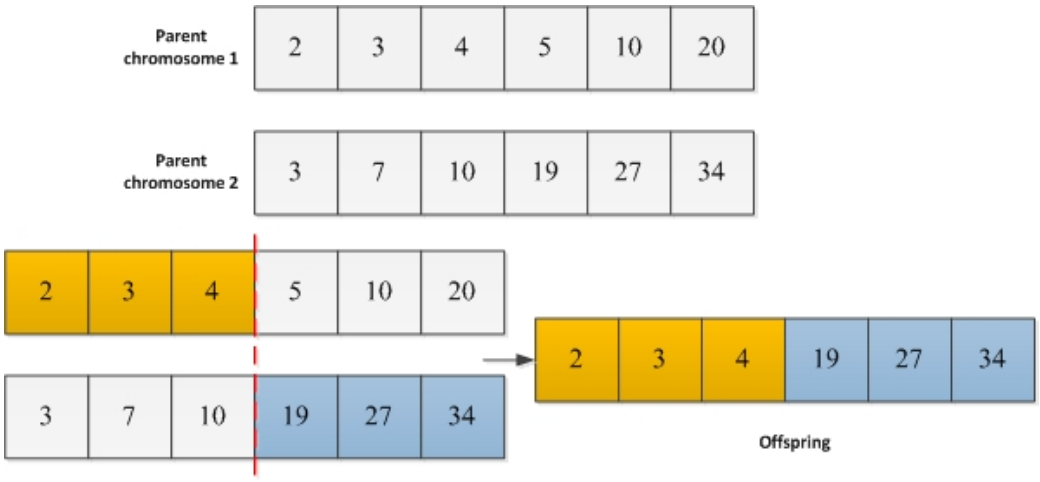

Figure 7. Crossover process within GA.

Mutation is the second main genetic process and is performed on every offspring produced from crossover. In this process the algorithm will randomly select one element from the crossover offspring time duration vector and copy to a randomly selected mutation element within the same chromosome. This process imitates the mutation process of DNA. Mutation may improve the final solution by searching globally rather than diverging to a local optimum. Figure 8 shows the mutation process in the algorithm for one offspring chromosome.

Figure 8. Mutation process within GA.

The work center fitness function (c_s) is used to evaluate all M mutated chromosomes. The fitness function value for the M^{th} chromosome is calculated as a linear combination of the makespan cost function (C_M) and cost functions for each work center (c_s^M). The fitness function value is defined as:

$$(Fitness\ function\ value)_M = C_M(T) + \sum_{s=1}^{S} \sum_{p=1}^{P} \sum_{j=1}^{J} c_s^M \left(d_{j,p,s}\right)$$

The mutated offspring and associated parent chromosomes are evaluated using their relative fitness function values. Mutated offspring with a lower fitness function value than either of the associated parent chromosomes will replace the parent chromosome with the largest fitness function value. This iterative process will continue until the program reaches the termination condition of a predefined number of iterations. Finally, the chromosome with the lowest fitness value (for a minimization problem) will be chosen as the solution for the problem. The optimization engine uses a different seed for the random number generator and thus the solution to the problem may vary between calculation executions. Increasing the number of iterations would increase the likelihood that different calculation instances

would produce the same solution and more importantly it would also increase the likelihood of a better solution. The pseudocode of the GA algorithm discussed in this paper is outlined in the appendix.

3.9 System Design

A C# graphical user interface (GUI) application was developed to act as the conduit between user inputs and the optimization engine as well as a graphical means to display the results. Using the interface, the user can enter all of the variables of the problem including number of jobs, number of work centers in series, number of work centers in parallel, cost per work center, and cost for makespan. Figure 9 shows the interface for providing the problem inputs with example inputs prepopulated.

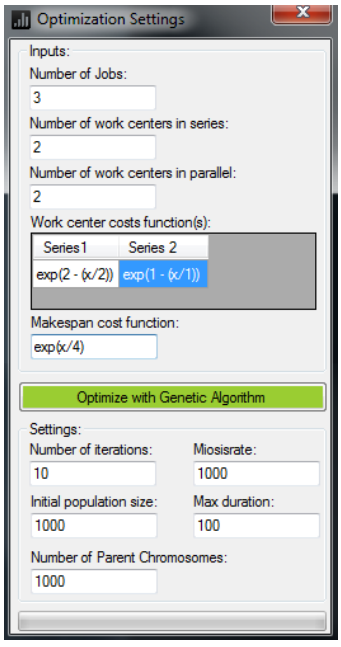

Figure 9. C# GUI interface for providing optimization problem inputs.

The C# GUI collects the necessary information to formulate the problem and then sends this information to an embedded python GA script. The GA is then executed with the problem until the termination condition has been meet at which point it then

sends the solution with the lowest fitness value back to the C# GUI to be graphically displayed via the Gantt chart. The name of the GUI is called “scheduler.” The front end is similar to Microsoft® Project in that time is expanded across the x-axis and resources (equipment) are listed along the y-axis. Events (also known as jobs) and dependencies between events can be added manually. Figure 10 shows what the scheduler looks like without any events added to the scheduler and Figure 11 shows a summary of the system process flow.

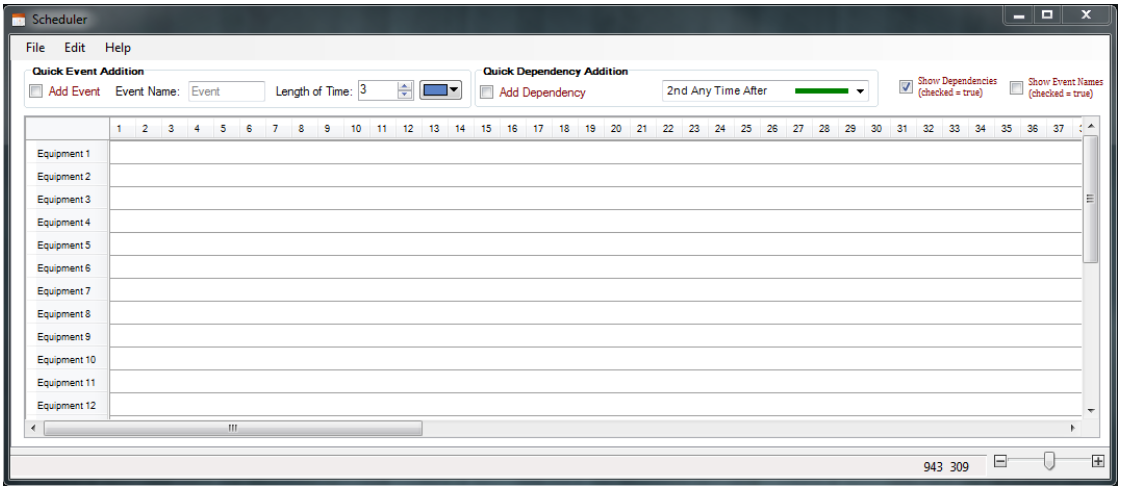

Figure 10. C# GUI interface for displaying solution.

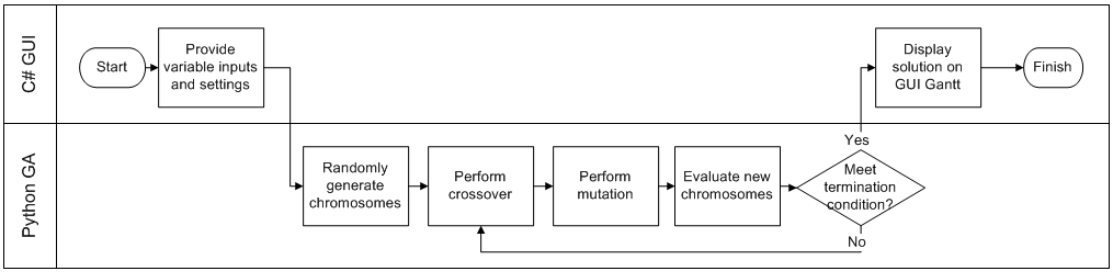

Figure 11. Process flow of the C# and python system.

3.10 Additional Features

Schedules created with the scheduler can be saved as multiple different file formats including CSV, TXT, XLSX, and BMP. Models saved to CSV, TXT, and XLSX can also be loaded back into the scheduler. There is a zoom track-bar that allows the user to adjust the zoom of the Gantt chart approximately between 10% and 200% the default size. Tooltips can be enabled or disabled to allow the user to quickly view each event's name, associated equipment, start time, and duration. Lines showing the dependencies between events can also be enabled or disabled. Events can be dragged and dropped anywhere within the preconfigured workspace. Event durations can also be adjusted simply by dragging the ends of any event. The progress of the python GA script, as a percent completed, is displayed on a progress bar within the C# GUI while the python script is executing.

The scheduler also allows the user to "re-optimize" a previously evaluated problem. After a solution has been graphically displayed via the Gantt chart, the user has the ability to adjust and then lock any event duration (i.e. set an event duration constraint). Duration constraints can be applied to any number of events simultaneously. Re-optimization then evaluates the problem with the new event duration constraint(s) included. This new solution will likely have a higher cost than the original solution but this functionality is intended to provide a means to respond to instances where event delays occur. Figure 12 shows iterations of scheduler throughout a re-optimization example. Part "a" of Figure 12 shows the optimized solution for a problem with no event duration constraints. Part "b" shows the scheduler with a duration constraint of 4 time units manually applied to job 1 on work

center in series 1 and work center in parallel 1 (equipment S1P1). Part “c” shows the new re-optimized solution for the problem with this new event duration constraint included. The re-optimized solution adjusted some of the non-duration constrained events from part “b” to minimize cost.

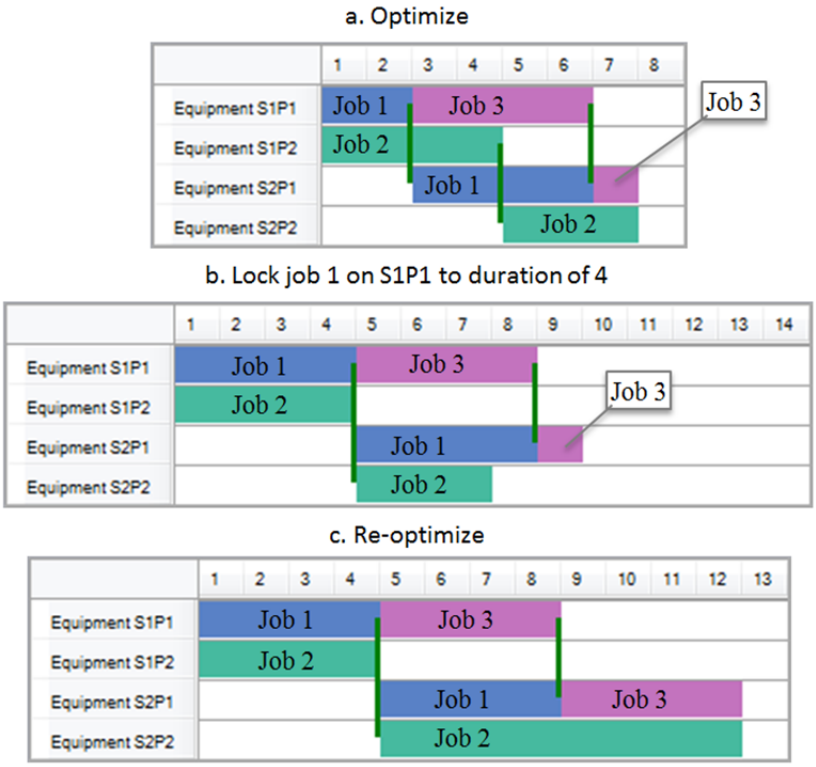

Figure 12. Scheduler graphical display for re-optimization example.

3.11 Limitations

One limitation to the optimization engine is a maximum duration for any given operation must be provided. Another limitation of the GA is that it is designed to follow a permutation schedule. The last limitation of the GA is that it assumes there are an equal number of work centers in parallel for each work center in series. This assumption could be removed allowing the optimization engine to solve a problem

with unequal number of work centers in parallel for each work center in series but this would require modification to both the GA and the GUI.

The limitations of the GUI include it cannot display more than 100 work centers in series, has a maximum makespan of 650 units of time, and time units must be natural numbers (i.e. cannot be decimal or negative). The optimization engine is capable of executing on multiple processers but is limited to single processing in order to integrate with the C# GUI.

CHAPTER 4

FINDINGS

4.1 Case Studies

There are five case studies discussed in this chapter. The first two are relatively simple case studies to evaluate the accuracy of the approach. The next three are more complex case studies to assess robustness and scalability. For each of the case studies, the following three methods will be evaluated.

- Graphical solution: 3D and contour plots using Wolfram Mathematica
- IBM® CPLEX Optimizer: Mixed integer quadratic program (MIQP)
- C# and Python: Genetic algorithm

4.1.1 Case Study 1

The first case study is a very simple case study with the intent to evaluate accuracy. There are eleven inputs to this particular problem (shown in Table 1) including cost functions which are dependent upon the number of work centers (WC) in series. There will always be one makespan cost function and one cost function for every WC in series. The cost functions for this case study are quadratic.

Jobs	WC in Series	WC in Parallel
1	2	1

WC Series 1 Cost Function	WC Series 2 Cost Function	Makespan Cost Function
$c_{1,1} = \left(-2 + \frac{d_{1,1,1}}{2.5}\right)^2$	$c_{1,2} = \left(-3 + \frac{d_{1,1,2}}{5}\right)^2$	$C = \left(\frac{d_{1,1,1} + d_{1,1,2}}{6}\right)^2$

Iterations	Meiosisrate	Initial Population	Max Duration	Parent Chromosomes
100	1000	1000	100	1000

Table 1. Case study 1 parameter inputs.

The objective function (minimize total cost) for case study 1 is defined as:

$$c_{1,1} + c_{1,2} + C = \left(-2 + \frac{d_{1,1,1}}{2}\right)^2 + \left(-3 + \frac{d_{1,1,2}}{5}\right)^2 + \left(\frac{d_{1,1,1} + d_{1,1,2}}{6}\right)^2$$

Figure 13 and Figure 14 show the 3D and contour plot graphical solutions (respectively) to case study 1. The graphical solutions indicates the minimum total cost at approximately $d_{1,1,1} = 3$ and $d_{1,1,2} = 8$.

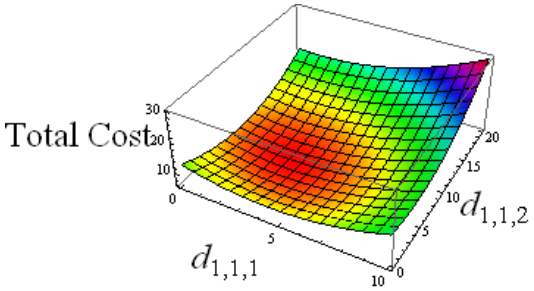

Figure 13. 3D plot presenting the graphical solution for case study 1.

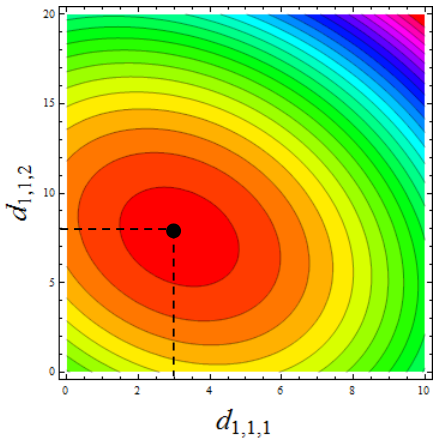

Figure 14. Contour plot presenting the graphical solution for case study 1.

Figure 15 is a graph of the cost and makespan calculated by the GA algorithm for each of the first 50 iteration. This graph shows rapid divergence to a solution with an approximate solution in as little as 12 iterations. Figure 16 shows the Gantt chart output from the scheduler when provided the inputs to case study 1.

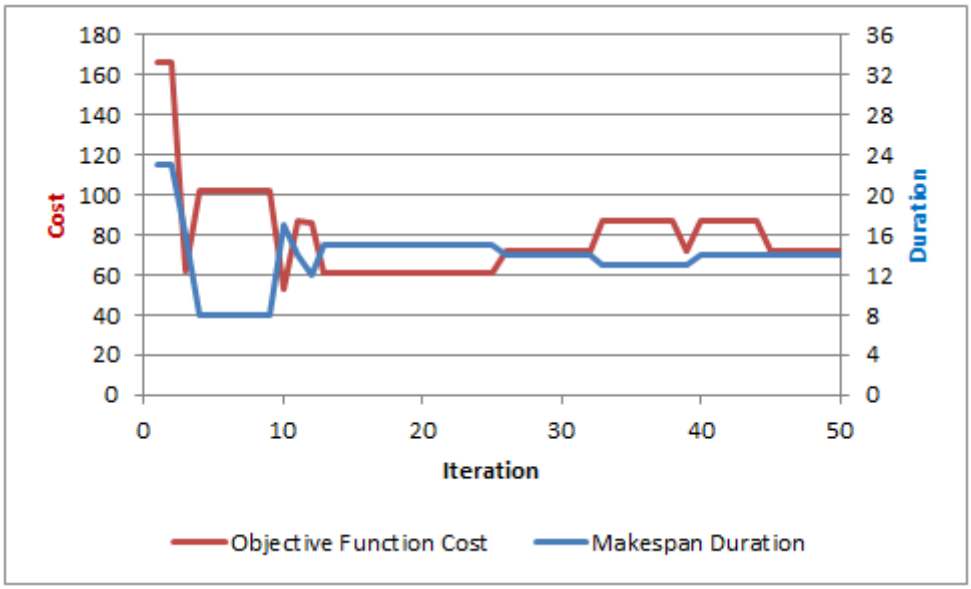

Figure 15. Chart showing divergence to a solution for case study 1.

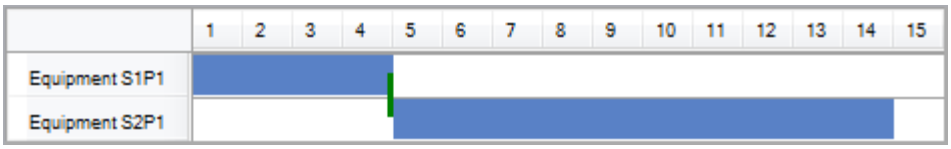

Figure 16. Scheduler graphical display for case study 1.

The performance of the three different methods (shown in Table 2) is relatively similar. Graphical and IBM® CPLEX had slightly more desirable total cost solutions of 6 relative to the GA solution of 7. The IBM® CPLEX solution required a permutation schedule formulation otherwise the problem would have been a polynomial of degree 3 and consequently unsolvable by CPLEX.

Method	Iterations	$d_{1,1,1}$	$d_{1,1,2}$	Total Cost
Graphical	N/A	Approx. 3	Approx. 8	6
IBM® CPLEX – MIQP	100	3	8	6
C# and Python – GA	100	4	10	7

Table 2. Case study 1 method performance comparison.

4.1.2 Case Study 2

The second case study is again designed to be simple with the intent to evaluate the accuracy. This case study will also evaluate robustness because this problem involves exponential cost functions rather than the previously evaluated quadratic cost functions. There are again eleven inputs to this particular problem (shown in Table 3).

Jobs	WC in Series	WC in Parallel
1	2	1

WC Series 1 Cost Function	WC Series 2 Cost Function	Makespan Cost Function
$c_{1,1} = exp\left(5 - \frac{d_{1,1,1}}{5}\right)$	$c_{1,2} = exp(1 - d_{1,1,2})$	$C = exp\left(\frac{d_{1,1,1} + d_{1,1,2}}{8}\right)$

Iterations	Meiosisrate	Initial Population	Max Duration	Parent Chromosomes
50	1000	1000	100	1000

Table 3. Case study 2 parameter inputs.

The objective function (minimize total cost) for case study 2 is defined as:

$$c_{1,1} + c_{1,2} + C = exp\left(5 - \frac{d_{1,1,1}}{5}\right) + exp(1 - d_{1,1,2}) + exp\left(\frac{d_{1,1,1} + d_{1,1,2}}{8}\right)$$

Figure 17 and Figure 18 show the 3D and contour plot graphical solutions (respectively) to case study 2. The graphical solutions indicates the minimum total cost at approximately $d_{1,1,1} = 17$ and $d_{1,1,2} = 1$.

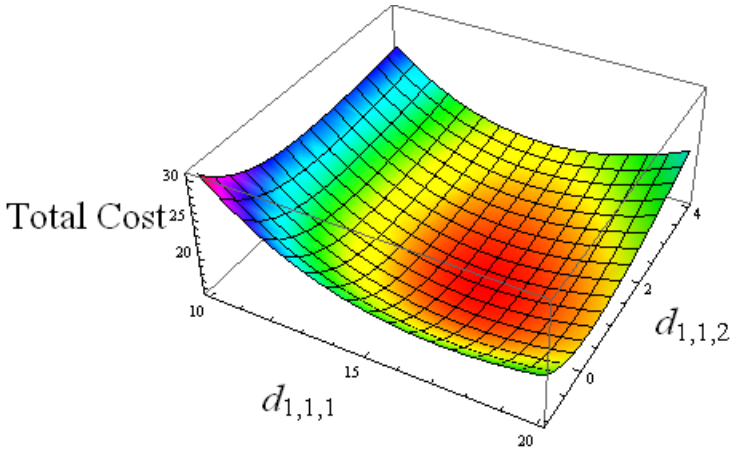

Figure 17. 3D plot presenting the graphical solution for case study 2.

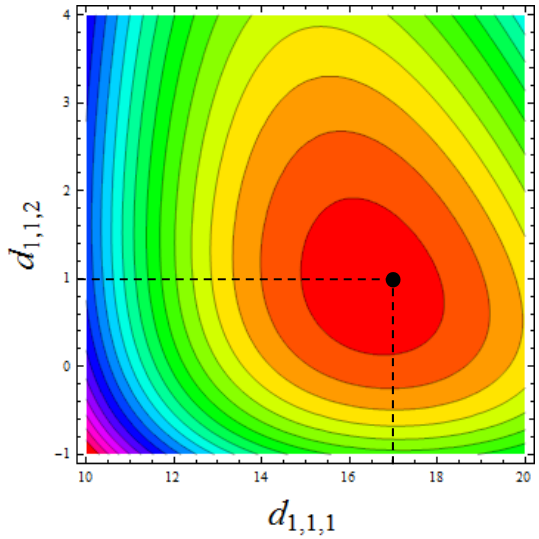

Figure 18. Contour plot presenting the graphical solution for case study 2.

Figure 19 is a graph of the cost and makespan calculated by the GA algorithm for each of the first 50 iterations. This graph shows rapid divergence to a solution with an approximate solution in as little as 19 iterations. Figure 20 shows the Gantt chart output from the scheduler when provided the inputs to case study 2.

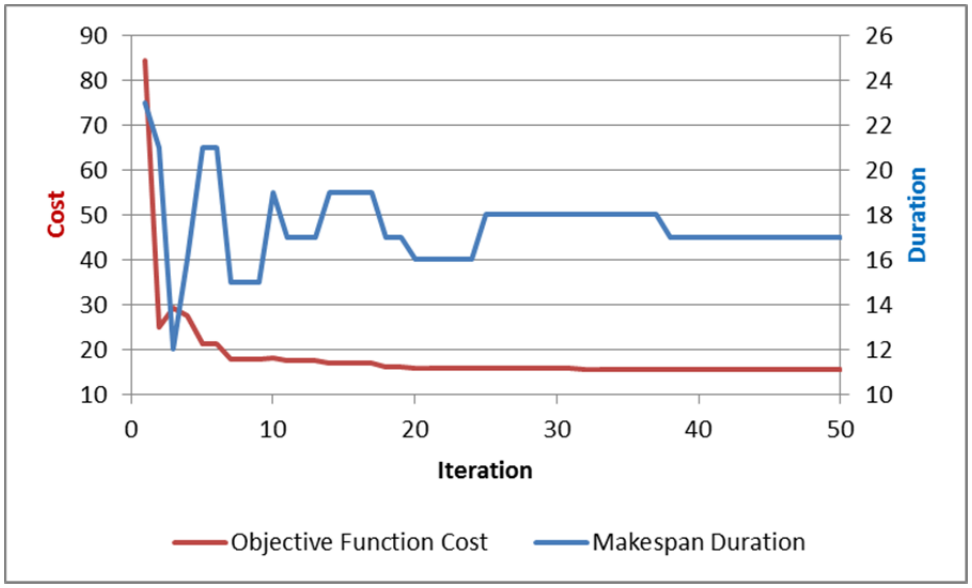

Figure 19. Chart showing divergence to a solution for case study 2.

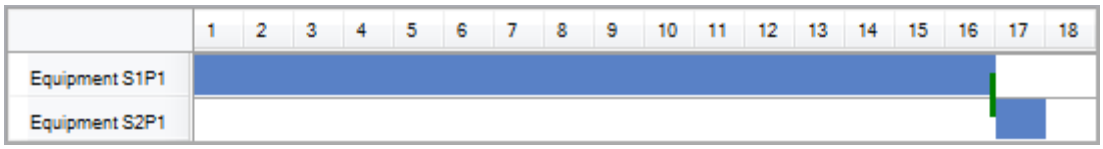

Figure 20. Scheduler graphical display for case study 2.

The performance of the three different methods (shown in Table 4) is the same for those that could be evaluated. A total cost solution of 15 was calculated for both the graphical and GA methods. CPLEX has the limitation that it "cannot solve problems where a decision variable is an argument to the exponential function" (IBM Corp. 2010).

Method	Iterations	$d_{1,1,1}$	$d_{1,1,2}$	Total Cost
Graphical	N/A	Approx. 17	Approx. 1	15
IBM® CPLEX – MIQP	CPLEX cannot solve problems where a decision variable is an argument to the exponential function (IBM Corp. 2010)			
C# and Python – GA	50	16	1	15

Table 4. Case study 2 method performance comparison.

4.1.3 Case Study 3

The third case study is designed to show robustness and scalability against a slightly larger problem than the first two case studies and with quadratic cost functions. There are eleven inputs to this particular problem (shown in Table 5).

Jobs	WC in Series	WC in Parallel
3	2	2

WC Series 1 Cost Function	WC Series 2 Cost Function	Makespan Cost Function
$c_{1,1} = \left(-6 + 0.8d_{j,p,1}\right)^2$	$c_{1,2} = \left(-3 + 0.5d_{j,p,2}\right)^2$	$C = (0.5T)^2$

Iterations	Meiosisrate	Initial Population	Max Duration	Parent Chromosomes
100	1000	1000	100	1000

.

Table 5. Case study 3 parameter inputs.

Figure 21 is a graph of the cost and makespan calculated by the GA algorithm for each of the first 50 iteration. This graph shows rapid divergence to a solution with an approximate solution in as little as 17 iterations. Figure 22 shows the Gantt chart output from the scheduler when provided the inputs to case study 3.

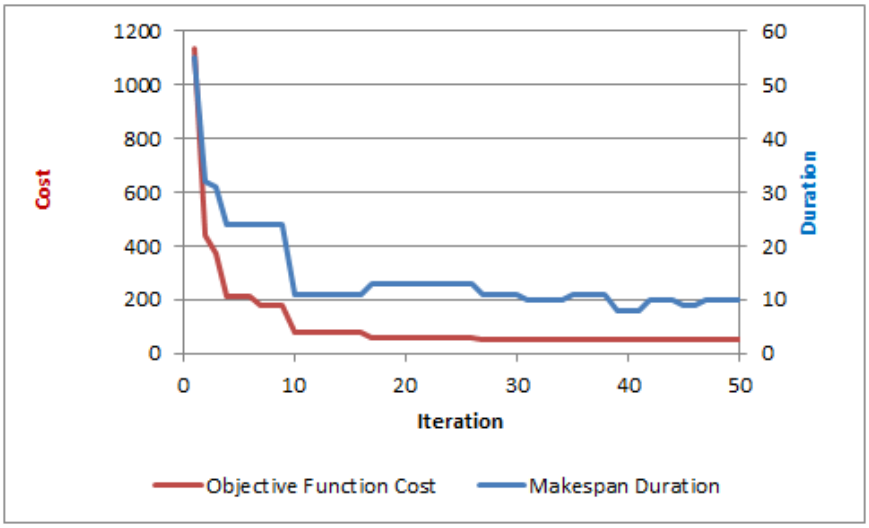

Figure 21. Chart showing divergence to a solution for case study 3.

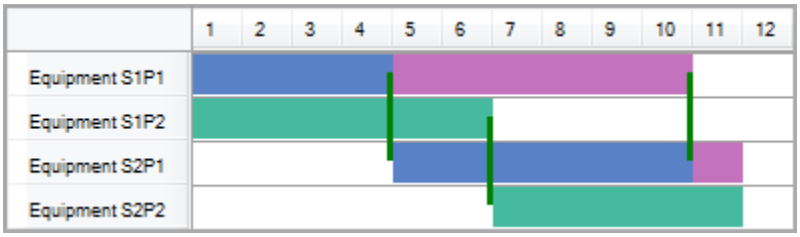

Figure 22. Scheduler graphical display for case study 3.

The performance of the three different methods is shown in Table 6 for those that could be evaluated. The IBM® CPLEX had slightly more desirable total cost solution of 45 relative to the GA solution of 47. The IBM® CPLEX solution again required a permutation schedule formulation otherwise the problem would have been a polynomial of degree 3 and consequently unsolvable by CPLEX.

Method	Iterations	Total Cost
Graphical	Greater than three dimensions.	
IBM® CPLEX – MIQP	100	45
C# and Python – GA	100	47

Table 6. Case study 3 method performance comparison.

4.1.4 Case Study 4

The forth case study is again designed to show robustness and scalability with a problem similar to case study 3 but with exponential cost functions. There are eleven inputs to this particular problem (shown in Table 7).

Jobs	WC in Series	WC in Parallel
3	2	2

WC Series 1 Cost Function	WC Series 2 Cost Function	Makespan Cost Function
$c_{p,1} = exp\left(2 - \frac{d_{j,p,1}}{2}\right)$	$c_{p,2} = exp\left(1 - \frac{d_{j,p,2}}{1}\right)$	$c_{p,1} = exp\left(\frac{T}{4}\right)$

Iterations	Meiosisrate	Initial Population	Max Duration	Parent Chromosomes
50	1000	1000	100	1000

Table 7. Case study 4 parameter inputs.

Figure 23 is a graph of the cost and makespan calculated by the GA algorithm for each of the first 50 iteration. This graph shows rapid divergence to a solution with an approximate solution in as little as 15 iterations. Figure 24 shows the Gantt chart output from the scheduler when provided the inputs to case study 4.

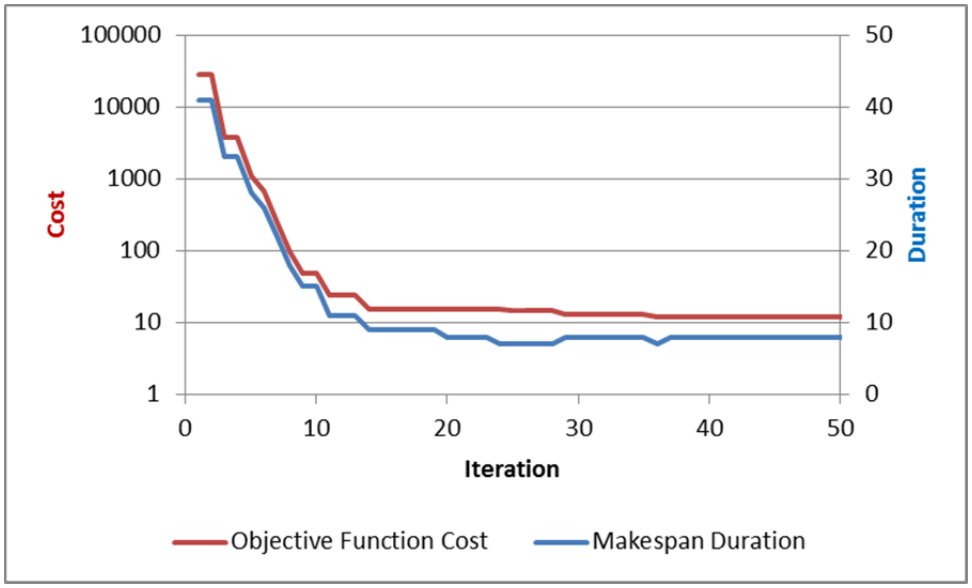

Figure 23. Chart showing divergence to a solution for case study 4.

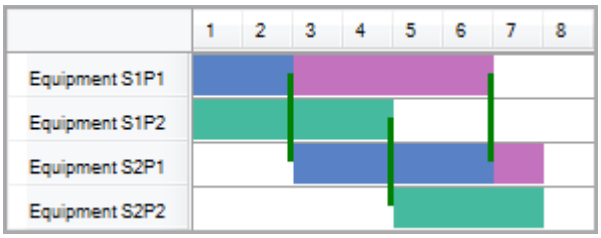

Figure 24. Scheduler graphical display for case study 4.

The performance of the three different methods is shown in Table 8 for those that could be evaluated. A total cost solution of 12 was calculated for the GA method.

Method	Iterations	Total Cost
Graphical	Greater than three dimensions.	
IBM® CPLEX – MIQP	CPLEX cannot solve problems where a decision variable is an argument to the exponential function (IBM Corp. 2010).	
C# and Python – GA	50	12

Table 8. Case study 4 method performance comparison.

4.1.5 Case Study 5

The final case study is designed to show the scalability of the GA algorithm by evaluating a larger size problem than any of the previous case studies. There are 15 inputs to this particular problem (shown in Table 9). The cost functions for this case study are exponential.

Jobs	WC in Series	WC in Parallel
25	6	5

WC Series 1 Cost Function	WC Series 2 Cost Function
$c_{p,1} = exp(4 - 0.7d_{j,p,1})$	$c_{p,2} = exp(3 - d_{j,p,2})$

WC Series 3 Cost Function	WC Series 4 Cost Function
$c_{p,3} = exp(1 - 0.8d_{j,p,3})$	$c_{p,4} = exp(2.5 - 0.95d_{j,p,4})$

WC Series 5 Cost Function	WC Series 6 Cost Function
$c_{p,5} = exp(3.25 - 0.75d_{j,p,5})$	$c_{p,6} = exp(3 - 0.65d_{j,p,6})$

Makespan Cost Function
$c_{p,1} = exp(0.8T)$

Iterations	Meiosisrate	Initial Population	Max Duration	Parent Chromosomes
1000	1000	1000	100	1000

Table 9. Case study 5 parameter inputs.

Figure 25 is a graph of the cost and makespan calculated by the GA algorithm for each of the first 1000 iterations. This graph shows divergence to a solution with an

approximate solution to this larger size problem after 900 iterations. Figure 26 shows the output from the schedule for case study 5 after 250 iterations and Figure 27 shows the output from the scheduler for case study 5 after the full 1000 iterations. The makespan reduced from approximately 100 time units down to 42 between the first 250 iterations and the 1000th iteration.

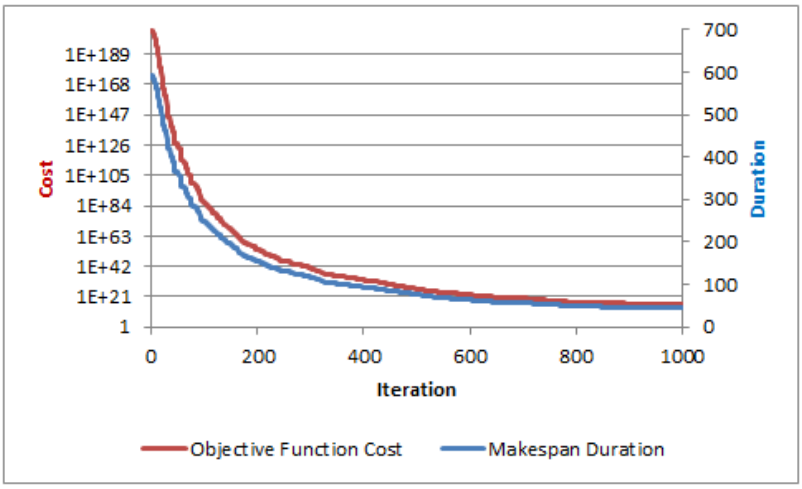

Figure 25. Chart showing divergence to a solution for case study 5.

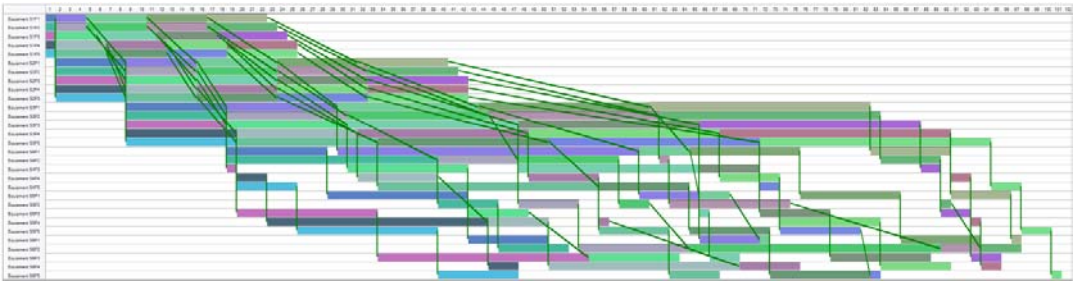

Figure 26. Scheduler graphical display for case study 5 after 250 iterations.

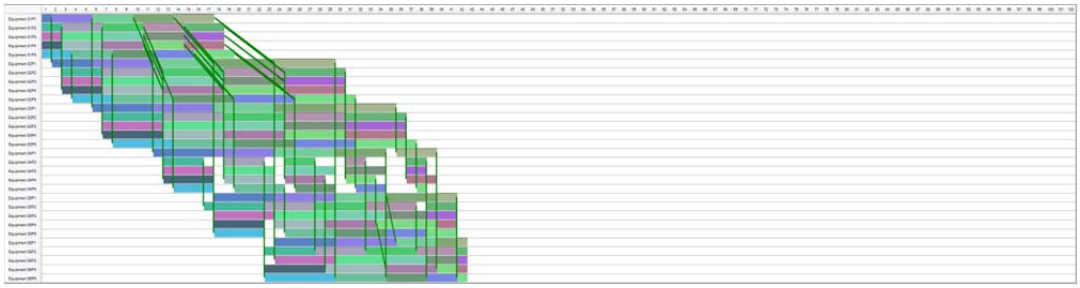

Figure 27. Scheduler graphical display for case study 5 after 1000 iterations.

The performance of the three different methods is shown in Table 10 for those that could be evaluated. A total cost solution of $3.9x10^{14}$ was calculated for the GA method.

Method	Iterations	Total Cost
Graphical	Greater than three dimensions.	
IBM® CPLEX – MIQP	CPLEX cannot solve problems where a decision variable is an argument to the exponential function (IBM Corp. 2010).	
C# and Python – GA	1000	$3.9x10^{14}$

Table 10. Case study 5 method performance comparison.

CHAPTER 5

CONCLUSION

This work has demonstrated that the genetic algorithm can be successfully applied to scheduling optimization with flexible durations. Solving NP combinatorial optimization problems can be computationally expensive but the proposed approach has shown accurate, robust, and scalable when evaluated against multiple generic case studies of different variety and sizes. Leveraging this approach can be valuable to industry by minimizing cost and consequently increase profit margins.

This work serves as the beginning. There is incredible potential for future work related to the problem discussed in this paper. Future work should attempt to solve the problem without the limitation of permutation scheduling, which will likely result in lower cost solutions. Also, work focused on the linearization of the mixed integer program formulation may yield desirable opportunities for other approaches, in addition to the approach proposed in this paper.

APPENDIX

GA PSEUDOCODE

Procedure: overall adaptive multi objective Genetic Algorithm

Input: number of jobs (j), number of work centers in parallel (p), number of work centers in series (s), population size (N), parent chromosomes (M) and meiosisrate (n), c_s^M , C_M

Output: a complete schedule

var: n' (number of chromosomes which do the procedure in parallel)

begin

$t \leftarrow 0$

initialize the population *P(t)* by encoding pattern in to chromosome M_i; $\forall\ i \in N$

evaluate *P(t)* by fitness function (by using c_s^M , C_M);

multiply chromosomes' fitness values *with -1* (for minimization);

sort *P(t)* in heapq tree;

while (not termination condition) **do**

begin

select M parent chromosomes from *P(t)* and pair *n* of them and put them in *B(t);*

create n'*C(t)* from *B(t)* by crossover procedure;

create n'*C(t)* from *B(t)* by mutation procedure;

improve n'*C(t)* from *B(t)* by local search;

evaluate n'*C(t)* by using fitness function;

select *P(t+1)* from *P(t)* and *C(t)* by selection procedure;

$t \leftarrow$ *t+1;*

end(while)

output a complete schedule

end

Figure A.28. Proposed GA pseudocode for the problem. (Nashrollahishirazi, Roy and Sodhi n.d.).

BIBLIOGRAPHY

Balasubramanian, J, and I.E. Grossmann. "Scheduling optimization under uncertainty-an alternative approach." *Computers and Chemical Engineering* 27, no. 4 (2003): 469-490.

Biegel, John E, and James J Davern. "Genetic algorithms and job shop scheduling." *Computers & Industrial Engineering* 19, no. 1 (1990): 81-91.

Chen, Haoxun, Jürgen Ihlow, and Carsten Lehmann. "A genetic algorithm for flexible job-shop scheduling." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference* 2 (1999): 1120-1125.

Fanti, M.P., B. Maione, D. Naso, and B. Turchiano. "Genetic multi-criteria approach to flexible line scheduling." *International Journal of Approximate Reasoning* 19, no. 1-2 (1998): 5-21.

Gao, Jie, Mitsuo Gen, and Linyan Sun. "Scheduling jobs and maintenances in flexible job shop with a hybrid gentic algorithm." *Journal of Intelligent Manufacturing* 17, no. 4 (2006): 493-507.

Garey, M. R., D. S. Johnson, and Ravi Sethi. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1, no. 2 (May 1976): 117-129.

Garey, Michael R., and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Edited by Victor Klee. San Francisco: W.H. Freeman and Company, 1979.

Gonzalez, Teofilo, and Sartaj Sahni. "Flowshop and Jobshop Schedules: Complexity and Approximation." *Operations Research* 26, no. 1 (February 1978): 36-52.

Guo, Z.X., W.K. Wong, S.Y.S. Leung, and J.T. Fan. "A genetic-algorithm-based optimization model for scheduling flexible assembly lines." *The International Journal of Advanced Manufacturing Technology* 36, no. 1 (2008): 156-168.

IBM Corp. 2010. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.2.0/ilog.odms.cplex.help/Content/Optimization/Documentation/CPLEX/_pubskel/CPLEX1248.html (accessed January 15, 2015).

Kumar, R., M.K. Tiwari, and R. Shankar. "Scheduling of flexible manufacturing systems: an ant colony optimization approach." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 217, no. 10 (2003): 1443-1453.

Lenstra, Jan Karel, AHG Rinnooy Kan, and Peter Brucker. "Complexity of machine scheduling problems." *Annals of discrete mathematics* 1 (1977): 343-362.

Li, Zukui, and Marianthi G. Ierapetritou. "Robust Optimization for Process Scheduling Under Uncertainty." *Industrial & Engineering Chemistry Research* 47, no. 12 (2008): 4148-4157.

McLoughlin, Brian, Farshad Doulatshahi, and Jason Onorati. July 2011. http://www.boeing.com/commercial/aeromagazine/articles/2011_q3/3/ (accessed December 28, 2014).

Nashrollahishirazi, Arash, Kevin Roy, and Manbir Sodhi. "A genetic algorithm to compute non-permutation schedules for flow-lines with flexible job processing times." *Working Paper*, n.d.