Robust Solutions in Resource Leveling via Genetic Algorithm Populations

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

David F. Dunham

May

2015

APPROVAL SHEET

The thesis

is submitted in partial fulfillment of the requirements

for the degree of

Master of Science

_____

AUTHOR

The thesis has been read and approved by the examining committee:

Prof. Donald E. Brown

_____

Advisor

Prof. Gerard P. Learmonth

_____

Prof. William T. Scherer

_____

_____

_____

_____

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

May

2015

Abstract

Algorithms for solving the resource leveling problem (RLP) in construction projects are proven to increase efficiency, create predictability, and balance resource demand across adjacent time periods of the project's duration while observing time and resource constraints. Leveling resources reduces the amount of change between one time period and the next in the project's time-use resource histogram. Conventional optimization methods of the RLP can become difficult as the number of decision variables grow, because the solution space increases exponentially as variables are added. Genetic algorithms are very capable when applied to large-scale instances of the RLP, and here the author applies a genetic algorithm testing multiple objective functions from literature with different performance measures, including kurtosis, the Resource Improvement Coefficient, and non-parametric statistical analysis. Results show that given a large problem based on practical data, genetic algorithms produce an unexploited large range of robust options via their embedded search population for decision-makers and planners. These alternatives can even be selected mid-project up to a certain point, while maintaining solution strength. There is a demonstrated tradeoff between number of time-feasible alternate solutions and variance of the alternate decision variables.

Contents

1 Introduction

1.1 Resource leveling

Second only to resource scheduling, resource leveling has the potential to dramatically reduce costs and improve efficiency in any set of tasks based on precedence relationships and a finite number of resources [1]. Based around an assumed critical path established during a scheduling algorithm, resource leveling then takes non-critical tasks (tasks that despite their change in start time will not change the overall outcome of the project duration) and reorders them by shifting their start times in order to reduce the number of resources or expenses incurred during a project. In small projects (e.g. 5 to10 tasks), integer programming can be used as a tool to arrive at a feasible, optimal solution. Using different exact search methods and heuristics to level resources in [2] and [3] are earlier attempts to solve this problem. The difficulty of the resource leveling problem (RLP) lies in combinatorial explosion of the solution space as resources and tasks are added. This makes pure optimization techniques ideal for smaller problems only. [2] suggests that the larger problems are non-deterministic polynomial-time hard [4], and therefore are ideal for search heuristics to examine the solution space. Many heuristics attempt to arrive at a near-optimal solution by establishing an algorithmic search using a large population of search solutions that return the top solution from the group. The following methods combine the robust solution features of the population and offer comparisons between nine different objective functions using a normalized comparison between resources of different capacities with the Resource Improvement Coefficient [5], as well as statistical comparison of the effect of the mutation parameter using Friedman's test and an Bonferroni-Dunn adjusted p-value. This paper will demonstrate that during a construction project, genetic algorithm solution populations are capable of providing robust alternatives to the best solution using the population of near-optimal solutions and compare their statistical and normalized performance across different objective functions.

1.2 Theater Construction Management System

The goal of many research efforts in this area is the expand the methodology by accessing smaller benchmarks for comparison, but leaving the suggestion of further research on large specific problems to other papers. This research seeks to remedy this issue by applying RLP methods to real-world data acquired from the US Army's Theater Construction Management System. This repository of information contains data on the man-hours and resources required to fulfill given construction tasks, altered for reproducibility here and for the analysis of these methods.

2 Literature Review

2.1 Resource Leveling Methods

Resource leveling can be described as a depiction of the time-resource histogram which has tasks restructured in order to closely resemble a rectangle as much as possible. This rectangular shape can also be described as the kurtosis, or moment of the resource usage over time. A flat surface on the histogram indicates a minimum fluctuation of resources throughout the construction project. [3] established a method based on several assumptions in the construction project. First, activities are assumed to be continuous, and resources are applied at a constant rate over the entire task. Second, the duration of activities and the network logic is fixed in order to preserve the logical relationship that exists in the critical path. Finally, the project termination time is fixed, so that the search heuristic may not change the overall time of the project. Every set of resources will have some constraints which prevent the formation of a perfect rectangle, and the solution space itself relies on software to provide the critical path based on tasks and resources present in the project. [Harris] methodology revolves around the representation of the histogram itself, which is first provided by the proper data for tasks and resources. A matrix detailing the days of operation for each tasks as columns and the individual activities themselves as rows. Each matrix position that must be used by a task is assigned a 1, and a position that may be occupied is given a 2. Matrix positions that are unavailable are assigned a 0. Each day's resource use is then calculated to form a base case for the histogram. A list of activities ordered on the highest resources used is then formed, which has several tie breaking procedures based on total float and sequence length. Each activity on this list is processed in order to find the best reduction in resource consumption for the possible days where that activity may be scheduled based on preceding and following activities. The matrix of days active for tasks is updated with the new position information after each task is processed, and the final matrix is used to redraw the histogram representing the leveled resources. The objective function is the reduction in each day's change in moment m from the prescribed shifts $y_1 - y_2$ based on the resource rate r for the equation $\Delta m = r(y_1 - y_2)$. This heuristic is relatively simple in application because of its methodical step-by-step approach, and can therefore be used to tackle large problems while avoiding the difficulty of being NP-complete in practice when applying optimization techniques.

An issue that is covered with assumptions by many pieces of RLP literature is the existence of multiple resources. Many papers will assume a renewable resource, or equate resource usage in to unit costs such as currency in order to simplify the solution space by constraint reduction. [6] attacks the multi-resource problem using a genetic algorithm, which is uniquely suited to exploring this search space. These resources are optimized simultaneously resulting in reduced resource fluctuation.

One of the first optimization approaches to the problem by Easa in [2] also starts from an established

critical path. The model's assumptions are that only one resource is being leveled at a time. An alternative to leveling multiple resources is to express every resource in the same units (e.g. dollars or time) and apply the method. This integer programming approach involves minimizing an objective function that uses the difference between the ideal resource rate expressed as the total resource available over time of the project against the actual value, $z = \sum_{i=1}^{T}(e^- + e^+)$, where e^-or e^+ takes on the real positive value of the resource rate ($e^-$for below desired rate or e^+positive for above desired rate), with the other being zero. The specified constraints ensure that the resource rates are not exceeded in any one time period i, and that each activity y_i is completed by the assigned resources. Additional constraints ensure that the shift x_k for each non-critical activity k does not fall outside the range of the project from time zero to the end of the total float TF_k. A second objective function that instead minimizes the absolute difference between some other variables associated with each activity y_i, $z = \sum_{i=1}^{T}(u^- + u^+)$. Variables u^- and u^+ are implemented as programmable variables that minimize the difference between consecutive days of operation in order to decrease the total fluctuation of resource usage over time. Two cases are examined, the first which attempts to level the histogram of resource usage across the entire project, and the second case that attempts to minimize the change in resource usage from one day to the next (consecutive fluctuations). A third case is a small restructuring of the constraints which has the goal of conforming the resource histogram to a specific desired response. The main drawback to this optimization method is that a single resource is optimized at a time, or the resources must be expressed in uniform units in order to tackle a problem with multiple resources.

Al-Sayegh and Hariga propose a method for splitting activities during the optimization process in [7] by assigning costs to starting and stopping activities as well as the cost per-use for all resources executing assigned activities. The authors then take this concept further in [8] where they sidestep the difficulties associated with NP-completeness for large problems involving many resources and tasks with a "meta-heuristic" by combining several techniques. The heuristic method used is Particle Swarm Optimization (PSO) . This method initializes a number of solutions whose goal is to emulate the natural flow of a swarm such as a group of bees or a school of fish, with the idea that information about the best solution in nature (location of food) is assessed by the swarm through the sharing of information between members of the swarm. Each solution in the swarm carries information about its velocity in the solution space, best local position based on the objective function, and the best global solution in the swarm. The k^{th} search position for each solution i is related as an $N-$dimensional vector $X_i(k) = [x_{i1}(k), ..., x_{iN}(k)]$. The system is initialized such that the starting binary solutions for each swarm member are feasible project schedules. Each iteration of the search sees the members of the swarm update their position via their velocity (rate of change between previous position and the current position) using the equations

$$V_i(k) = wV_i(k-1) + c_1 r_1 \left[X_1^L - X_i(k-1)\right] + c_2 r_2 X^G - X_i(k-1) \tag{1}$$

$$X_i(k) = V_i(k) + X_i(k-1) \tag{2}$$

Where X_i^L is the best local solution for the i^{th} member of the swarm, X_i^G is the best global solution, and w, c_1, and c_2 are adjustable weights representing inertia, strength of local learning (or local gravity), and strength of global learning, respectively. Both r_1 and r_2 are uniform random numbers between 0 and 1. A maximum and minimum range is imposed on velocity in the form of $[V_{min}, V_{max}]$, which are also confined to a range between 0 and 1. Each bit or dimensional position information value $x_{in}(k)$ is represented as a string of binary values denoting whether or not an activity is active during a time period. So an example of this is to have the swarm solution be represented by $X_i(k) = \{(1,1,0), (0,1,1), (0,0,1)\}$, which would be three noncritical tasks occupying different time periods. After each update, the PSO algorithm in question may or may not have a feasible solution. If it exceeds resources, the task is either moved backward if there are non-critical prerequisite activities, or it is randomly moved with equal probability among other feasible time states until a feasible solution is found. This process continues until a solution cannot be improved upon, or a steady-state condition for the swarm members is found. The authors also note that a deficiency of the PSO method is individual members may trap themselves in local optima, and that the computational complexity of the search algorithm must be balanced against the search strength in the number of swarm members initialized.

Evolutionary algorithms like genetic algorithms and particle swarm optimization have led to more biology-based heuristics. In [9], an improved ant-colony optimization mimics the pheromone trail left by ants in order to search a solution space consisting of a directed graph. The solution strength of each ant affect the stochastic process of choosing a node-arc path along the graph. Another element of leveling resources in a practical application is simulated by adding stochastic processes to the duration of the activities as described in [10]. The authors also use a genetic algorithm, but take the model further by employing a Monte Carlo simulation model by selecting predetermined values for the task lengths using a cumulative distribution function. By analyzing project length in a stochastic fashion, this study allows for sensitivity analysis which can further be incorporated in to risk analysis and decision making.

2.2 Comparison of Optimization Solutions and Robustness

The different solutions provided by the population of the any algorithm which contains such a population

over the course of its search allows for further analysis of each member of the population. No solution in a heuristic search is guaranteed to be optimal, but due to the evolutionary nature of the algorithm carrying parts of the best solutions forward in to each generation, we may see multiple solutions that are very near-optimal but do no necessarily have similar decision variable values. In addition to having a population of near-optimal solutions for comparison, the robustness of these solution populations may be of some utility. Robustness is defined in The attractiveness of acquiring a robust solution stems from the natural world's changing features which can affect measurements and observations. [11] apply the concept of robustness to a single solution of an optimization problem in that the solution is resistant to small changes in the values of the decision variables. In other words, the optimum point that the solution rests in or is near to resides in a space that is closer to flat than sharply convex, or peaked. This ensures that the variables may be moved without throwing the solution too far from its optimum (or in to infeasible space). The motivation for the desire of this kind of robust solution is based in the real world idea of noise. Signals, measurements, and uncertainty all play a part in every day solutions, and optimization methods for the practicing engineer must be able to cope with these perturbations.

The kinds of noise that [12] discuss come from four different areas. The first kind of noise is applied to the objective function for the algorithm in question, such that an objective

$$F(X) = \int_{-\infty}^{\infty} [f(X) + z]p(z) \tag{3}$$

Where $z \sim N\left(0, \sigma^2\right)$. This objective function results in a random change for each solution for every calculation, which simulates some noise that is encountered for the problem at hand. The second kind of noise discussed by [12] and is change applied directly to the decision variables via some small δ_i,

$$F(X) = \int_{-\infty}^{\infty} f(X + \delta_i)p(\delta_i)d\delta_i \tag{4}$$

This solution is considered robust if it maintains near-optimality despite these δ_i changes in the decision variables. In other words, δ_i does not carry the new value of the objective $F(X)$ very far from its original value. While this noise is similar to the uniformly distributed noise, the key difference is the same continuous application of the objective function over $X + \delta_i$, instead of changing the value of the objective function after the measurement was taken.

Since the solutions provided are stochastic in nature, they cannot be reliably compared using conventional methods, such as comparing a single objective function value to another [11]. An algorithm like genetic algorithms that contains a measure of uncertainty in every evaluation, be it decision variable or objective function, should include a confidence interval for the values of the decision variables or solutions achieved. In

this way, the amount of variance provided in the solution can be measured and compared to make a better judgment as to the efficiency of the method in question.

Another special case of problem is the dynamic multi-objective optimization problem [13]. The introduction of time as a variable has a special relationship with the RLP, in that once time has passed for certain variables, the problem becomes simpler, as there are less decision variables to optimize. That is, once the start time for a given construction task has passed, that task is no longer available for adjustment by the algorithm. At the same time, [13] considers the important implication of considering the evaluation of not only the objective function value as fitness, but also the real values of the decision variables themselves. An additional level of complexity to this problem is time scales. If resources are being applied hourly in varying rates, does it make sense to re-compute the possible solutions at every hourly time step? What step value does it make sense to compute at? The authors answer is that more efficient methods are needed to compute in a time-sensitive environment to ensure optimality and decision making.

Lastly, [14] sheds further light on comparison techniques for different classes of algorithm comparison. The main two tests considered are the Friedman Rank-Sum Test and the Quade Test. Both of these tests are similar to analysis of variance (ANOVA) in that they test for differences between groups of data (treatments) separated by factors. In this case, it is different evolutionary algorithms (EAs). Because the EA solutions lack normality, the Friedman and Quade Tests are utilized because they claim no Gaussian assumptions with respect to distribution. The downside to those relaxed assumptions, however, is a loss of hypothesis strength. The Friedman test statistic treats all of the data sets as if they were equal, while the Quade test takes each data set and ranks them according to the largest difference between their maximum and minimum values. This is a good feature when we are trying to preserve the diversity (robustness) of the solution.

3 Problem Description

The RLP remains a challenging problem for practicing engineers who manage a large quantity of resources. As stated previously, the exponential increase in solution space from a modest increase in resources and tasks to be optimized results in a problem that is difficult to attack computationally. Once a problem solution has been identified however, the possibility that a single optimized solution for a large, complex operation is going to suffice is an unlikely proposition. Changing conditions, or noise, as described in [15] are a common reason to seek robustness in an algorithm's ability to find a solution. To that end, genetic algorithms provide some unique abilities in terms of noise addition by randomly mutating decision variables as well as randomly swapping variables between solutions. I seek to analyze what robustness lies between the different members of a genetic algorithm population solutions of different objective functions and how that may be exploited in the RLP.

The maintenance of heavy equipment and the overhead of carrying many construction workers highlights the need for effective management of resources during the execution of the project to control costs. Resource leveling reduces costs and complexity of projects by reducing the change in resources necessary over time for a given project. Many past papers have not used real-world data, but taken a theoretical approach to implementing resource leveling algorithms. Instead, this research focuses on data from military construction projects in order to accurately test genetic algorithms and various objective functions in a large, multi-resource projects. This research applies the concepts of variable similarity and model robustness to genetic algorithms, whereby a set of many feasible, near-optimal solutions provide stakeholders with a varied array of options to meet their prioritized demands in a complex project. Should the chose solution become infeasible during the course of the project, the project planners have multiple alternative solutions to choose from when capitalizing on the search populations provided by genetic algorithms. Many different search methods explored in literature find a way to either identify the shortcomings of exact search [2], or to discover a clever heuristic to overcome the shortcomings of exact search when applied to large problems [3]. The Resource Leveling Problem (RLP) is a class of problem that has the feature of being solvable in Non-deterministic Polynomial time, or NP-hard. NP-hard problems are characterized by a combinatorial explosion of the solution space as resources and tasks are added. This attribute makes pure optimization techniques ideal only for smaller problems. Larger problems are NP-hard, and therefore is ideal for search heuristics to examine the solution space. An area that is not well-explored is the robustness of genetic algorithms. Because genetic algorithms use a population of solutions to explore the search space by chromosomal recombination and mutation, at the end of the search a number of semi-optimal solutions exist. While no genetic algorithm is guaranteed to be exact, having a number of feasible, high-quality solutions to offer stakeholders is an advantage that population-based searches like genetic algorithms hold over more exact or single-solution search techniques.

4 Data and Methodology

4.1 Data and Objective functions

The construction tasks associated with a 30 to 60 soldier combat outpost have been pulled from the Theater Construction Management System utilized by the US Army. These tasks begin with site clearing, or horizontal construction, in which the site is bulldozed and graded in preparation for further vertical construction. The horizontal constructions unit provides the machinery and soldiers to conduct these tasks. After the horizontal leveling is completed, the vertical construction unit begins using human resources composed of carpenters, masons, electricians, and plumbers to execute work on construction activities once

Table 1: Objective Functions

Number	Function	Description
1	$Z = min \sum_i^T \mid Y_{i+1} - Y_i \mid$	Minimize absolute difference between time periods
2	$Z = min \sum_i^T \mid Y_{i+1}^+ - Y_i \mid$	Minimize positive difference between time periods
3	$Z = min \sum_i^T \mid Y_i - \overline{Y} \mid$	Minimize absolute difference between time periods and the mean
4	$Z = min[max(Y_i)]$	Minimize the max resource usage per time period
5	$Z = min[max \mid Y_{i+1} - Y_i \mid]$	Minimize the maximum absolute difference between time periods
6	$Z = min[max \mid Y_i - \overline{Y} \mid]$	Minimize the max absolute difference between time period and the mean
7	$Z = min \sum_i^T (Y_i)^2$	Minimize the square of the time period resource usage
8	$Z = min \sum_i^T (Y_{i+1} - Y_i)^2$	Minimize the square of the difference between time periods
9	$Z = min \sum_i^T \left(Y_i - \bar{Y}\right)^2$	Minimize the square of the difference between time period and the mean

horizontal construction is complete. The tasks for vertical construction include force protection towers, HESCO barrier walls, a security gate, Southeast Asia huts (SEAHUT) for housing, an ammunition storage building, an operations building, and a flat gravel helipad. These tasks use construction sub tasks from the Army's Theater Construction Management System to draw information about construction times, resources required, and the logical network path for each construction sub task. These sub tasks are then placed in to Microsoft Project to gain the critical path information necessary for the resource-leveling of non-critical tasks. With 59 tasks completed by 21 resources, viewing each graph visually (Appendix B) can be tedious. Relying on kurtosis to help demonstrate the overall shape of the graph numerically help us quickly evaluate candidate solutions.

The objective function in use minimizes the sum or squared difference of each t^{th} resource on every i^{th} day from the expected average resource level, $\bar{Y}_i$, across the project duration. Each solution is evaluated for its value of X which is then returned to the algorithm in question. In total there are nine objective functions from [16]

These objective functions are compared using kurtosis and a normalized version of kurtosis called the resource improvement coefficient (RIC) [5]. The RIC is normalized for the quantity of resources being use, which, unlike kurtosis, allows for direct comparisons between individual resources for the strength of the leveling solution. Resources with a high capacity will still have better scores, as it is easier to level them out by spreading more units of resource around. Resource with only a few or even a single unit of capacity or assignment will fare the worst, but RIC is a fair comparison due to the normalized measure.

$$RIC = \frac{n \sum Y_i^2}{\left(\sum Y_i\right)^2} \tag{5}$$

Other objective functions have included expressing the unit cost of a resource in a specific time to order all

resources, while other functions have minimized the objective expressed as resource usage fluctuations over a sequential time period. In the case of universal costs, I believe that it is not as realistic to express a problem in terms of cost in a military operation implied in the data, due to the nature of unit structure and equipment allocations. The budgeting requirement for deploying military units mostly comprises the transportation to the site of the counter-insurgency as well as the necessary logistics of keeping those units operating once they have arrived. These cost-planning procedures are generally unconsidered by individuals planning military construction operations, whose priorities are usually meeting mission construction objectives while minimizing or avoiding casualties. Therefore, considering the resources as uniform costs is not appropriate for this research. Minimizing fluctuations of day-to-day use, on the other hand, may be especially useful for high-priority items like heavy equipment or highly technical fields, like electricians or technical supervisor.

4.2 Methods

4.2.1 Genetic Algorithm

Genetic algorithms similar to [6] are a class of search heuristics which attempt to emulate the natural progression of evolution prescribed by the mating of surviving pairs of a species and the possible mutations of genes. This process is achieved by first initializing the genes, or real-valued sequences of start times for the tasks within bounding constraints that imply their network precedence, and then calculating the fitness of the initial sequence in the gene. A population of the user's preference is set where each member of the population is a possible solution that is initialized with random values that fall within the parameters of the search space. A prioritized list of the solutions is sorted with the best scoring genes at the top. The solutions, or "genes," all perform crossover at that point where each gene is randomly (or not randomly) paired with another gene, and they swap chromosomes, which are represented as different start times for different tasks. The point at which each gene is split in half is random. The start times after this split point are swapped to form two new solutions. After all the genes have completed crossover, there is the possibility of mutation, which can be set as a variable before the genetic algorithm search is initiated. After the mutation, the objective scores for each gene is recalculated and all the genes are re-sorted according to their objective score. Then elitism is applied, whereby only a certain number or percentage of the population continue on to the next iteration. The remaining population is discarded, similar to survival of the fittest. The search sequence then repeats according to either a minimal increase in population strength (i.e. the best solution does not get better), or a certain number of iterations have been run.

4.2.2 Robustness

Each solution in a genetic algorithm lies in a convex optimum space which is usually near-optimal without any guarantees of being the global optimum. While it is easy to identify robustness graphically in a one- or two-dimensional setup like [15], it becomes difficult to describe robustness graphically beyond three dimensions (two decision variables and one objective function), or in the case of the given decision variables, 21 dimensions. Coupling that with the nine objective functions of interest (Table 1), and the problem becomes difficult to imagine as a picture. In looking at a graph of solutions plotted against a solution strength in a one-dimensional case from [15], one could summarize robustness as maintaining as close a value as possible to the optimal objective f while allowing for a diverse range in x. Or, using the normalized values of RIC from [5], robustness among solutions R_{GA} maintaining the best possible RIC values while simultaneously seeking to maximize the amount of variance described the different decision variable values across a genetic algorithm population solution. The best scores for (6) would be those that are closer to zero. Because the RIC for each solution a among all objective functions is normalized and each decision variable x_b occupies the same range for each solution, (6) may be used to compare the robustness of each genetic algorithm's population.

$$R_{GA} = \frac{\left(\sum_{a=1}^{A} RIC_a - RIC_{best}\right)^2}{\sum_{a=1}^{A} var(x_a)} \tag{6}$$

5 Experimental Setup and Preliminary Results

The genetic algorithm was initially computed different values based on the inputs in Table 2, using a starting seed of 1234 and R's default Marsenne Twister for random number generation. The initial population is drawn from the critical path method start times, and the solution space is the available slack to the non-critical tasks. In this way, the duration of the project is not increased while exploring the full range of values for each decision variable. The GenAlg package from R was used to supply the algorithm and return the best solution in each model population. Each population's size and the number of iterations remained constant, as I discovered that after a certain point (approximately 7-12), increasing the number of iterations has diminishing improvement on the average solution in some cases. The best solution was unmoved after 7-12 iterations, and each run was completed with 25 iterations on 50-solution population with the initial results in Table 2. Additionally, increased elitism, where less solutions are carried over from the previous generation, decreases the computation time shown in Table 2 due to a lower computational requirement on the objective function by not computing the scores for the culled solutions. Mutation is kept low to

avoid having wild solutions affect the overall progression of each population towards solutions. Mutation was initially run at levels of 0.05, 0.10, and 0.15, and then later changed to 0.10, 0.20, and 0.30 for the final runs to generate a large data set of solutions (N=1350). The limited-memory BFGS-bounded method [17] was also initially examined (Table 2), confirming that the solution space is in fact filled with many local optima. This preliminary setup allowed insight for further exploration of the algorithm's performance on the problem data.

Table 2: Objective Function and Kurtosis Values

Model	Mutation Chance	Elitism	OBJ -SSD from Average	Average Kurtosis
1	0.05	10	34657.4	22.3522
2	0.10	10	34687.01	20.14292
3	0.15	10	35007.91	21.78503
4	0.05	20	35173.64	18.26831
5	0.10	20	34891.79	20.66061
6	0.15	20	34693.12	17.9332
7	0.05	30	34817.02	21.26747
8	0.10	30	34937.89	15.36339
9	0.15	30	35218.36	19.17767
10	L-BFGS-B	NA	60277.47	54.94959

After analyzing the results from Table 2, a new set of runs using the objective functions from Table 1, the mutation levels at 0.10, 0.20, and 0.30, and a constant elitism of 25. Each objective function was tested against the three runs from the different mutation levels, for a total of 27 runs. These 27 runs together provided a net 1,350 solutions for the objective function performance to be analyzed. These were compiled in to data sets for individual resource values for kurtosis and RIC with the appropriate factors indicated for the run as a record.

6 Experimental Results

6.1 Objective function sensitivity to mutation

[16] supplies 9 such objective functions which are variations on the main objective functions found in literature. Simple scatter plots of RIC and Kurtosis values across all solutions showed a high degree of integration between many of the objective functions across the different resources (Figure 1). To compensate, I attempted to aggregate the data across all objective function to see if there were Gaussian characteristics, but many of the data became multi-modal at that point (Figure 2). After compiling the data for both kurtosis

Figure 1: RIC and Kurtosis Scatter plots for Carpenters and General Labor

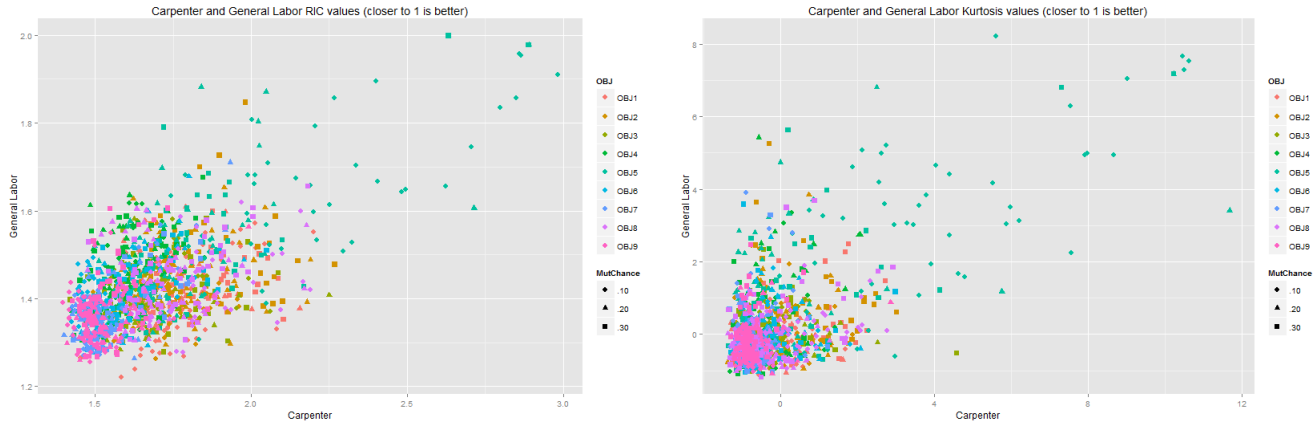
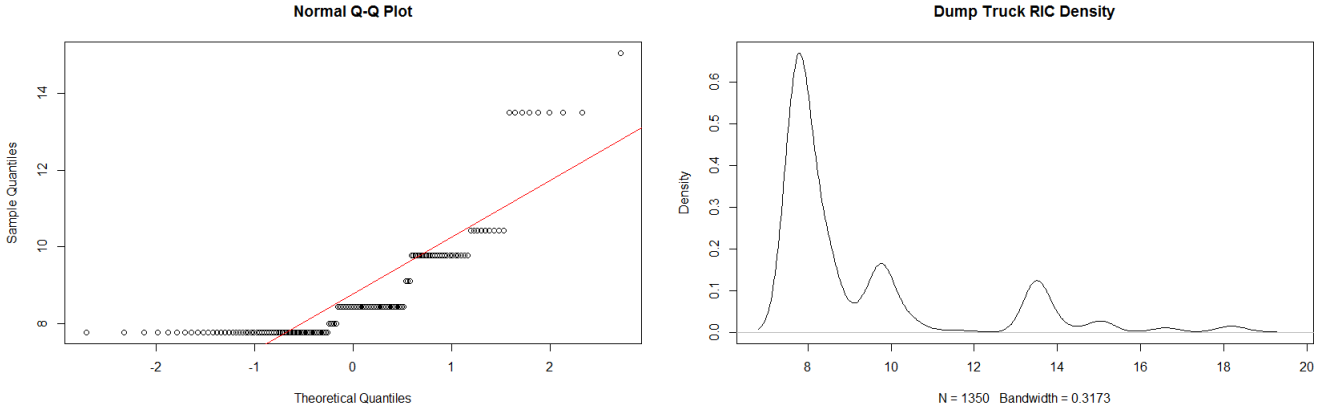


Figure 2: RIC QQ and Density Plots for Dump Trucks



and RIC, density plots, quantile-quantile plots, and chi-square tests for normality were run on all groups within each objective function treatment (i.e. 150 solutions). Each group of data failed all of the tests. With statistical test based on Gaussian assumptions unavailable, a recommended alternative to comparing optimization solutions is non-parametric analysis using the Friedman Rank-Sum Test [14].

The Friedman rank-sum test uses a test statistic

$$\chi_F^2 = \frac{12n}{k(k+1)} \left[\sum_j R_j^2 \frac{k(k+1)^2}{4} \right] \tag{7}$$

Table 3: Friedman's Test (Y < Adjusted P-value) Objective Function factored by Mutation Rate

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
OBJ1											Y	Y	Y		Y				Y		
OB2						Y		Y	Y	Y	Y										Y
OB3											Y					Y	Y			Y	
OBJ4												Y					Y				
OBJ5		Y	Y		Y	Y	Y	Y	Y	Y			Y	Y		Y					
OBJ6			Y	Y			Y	Y			Y	Y		Y			Y	Y	Y		Y
OBJ7			Y	Y		Y						Y	Y				Y	Y	Y	Y	Y
OBJ8											Y	Y			Y						
OBJ9		Y	Y	Y	Y	Y		Y			Y		Y		Y		Y	Y		Y	

Which has a χ^2_F distribution with $k-1$ degrees of freedom. The authors in [14] recommend that the different non-parametric analysis techniques represented by the Friedman Test, the Friedman Aligned-Sum test and Quade test are similar, but that the latter two should not be used when comparing more than 4-5 treatments as I am here. Therefore, the Friedman test is used to compare the differences among the different objective functions to test for significant differences among the solutions. I then perform post-hoc analysis as recommended in [14] by selecting the Bonferroni-Dunn adjustment to the p-value to add some power to the test. The adjusted p-value was 0.0018, which a number of groups within the mutation factors fell below for significance (Table 3). This table indicates which objective functions may have mutation chance play a significant role in the performances of the genetic algorithm. Objectives 5, 6, 7, and 9 all had a higher incidence of a statistical difference between the within objective function runs of a given mutation chance. In other words, at a 95% confidence interval with Bonferroni-Dunn adjustment with 8 degrees of freedom, these objective functions are sensitive to mutation when comparing RIC scores among solutions for the objective function and resource combinations indicated by a Y (for yes) in Table 3. Additionally, the Grader, Heavy Equipment Operator, and Plumber (resources 11, 12, and 17, respectively) are also more affected by difference in the mutation chance assigned to the genetic algorithm. These are all low-capacity resources that are not only assigned the critical path tasks, which indicates that there will be a high amour of variability when it comes to optimizing them in the RLP.

6.2 Optimality

Meeting the unique challenges of analyzing a non-parametric set of solutions requires visualization and an understanding of the ultimate objective of the genetic algorithm robustness. Stakeholders desire multiple options to suit a variety of changing constraints and goals, and a body of solutions which maintains a high degree of variance while still preserving solution strength may be defined as robustness. With a tool like

RIC to compare normalized resources, it is not necessarily prudent to devolve all resource management in to currency in order to make the RLP easier. It will still be difficult to compare solution statistically due to the weak assumptions that the Friedman's test rests on. The ultimate goal of resource leveling is the charts in Appendix B, where resource usage is leveled over time to turn the time-use histogram in to a square box. Achieving robustness and preserving the RLP solution strength at the same time with a genetic algorithm population ensures that there are alternate options should there be contingencies during the project. Applying the Friedman's test with Bonferroni-Dunn adjustment (.0167), this time to the mean RIC values supplied by each objective function and mutation combination as described in Table 4, yields a p-value of 0.007, stating that the objective functions when compared with mean RIC are sensitive to different mutation values at 95% confidence with Bonferroni-Dunn adjustment on 8 degrees of freedom. Based on this information and Table 4, objective function number nine with a mutation probability of 0.10 appears to be the best performing genetic algorithm for this method. Examining Figure 3 allows a comparison between mean RIC and variance, where it can be seen that the OBJ9 solution at 0.10 mutation lies firmly in the middle of the pack when it comes to overall variance. If mean RIC is a good metric of the population's robustness and more variance is desired, a better solution might be objective function number 4 at 0.30 mutation. Also worthwhile to note is that the solutions with higher mutation (0.30) are generally on the right side of the graph, while the opposite is true for the lowest mutation value of 0.10. Additionally, objective function number 5 performs quite poorly compared to the others, as all three of its solutions mean RIC are above its neighboring solutions.

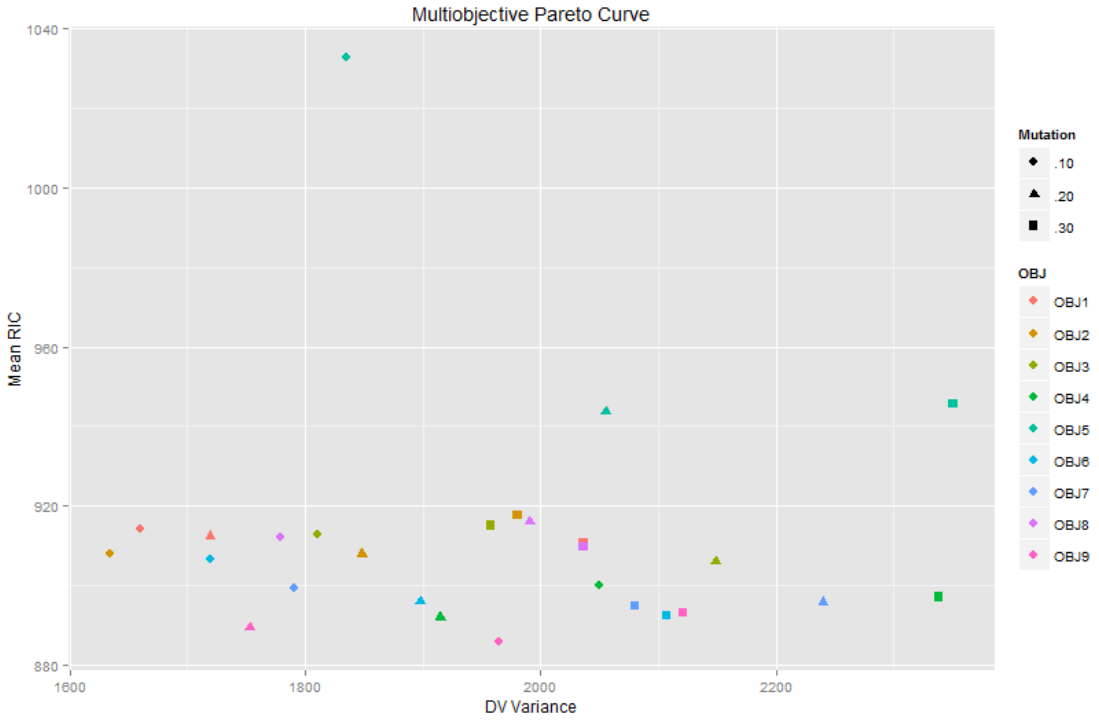
6.3 Robustness among the population

Using (6) to analyze the summed differences in robustness between all of a population's respective RIC solution values and the best RIC value for that solution yields Figure 4. What we see is a vertical stretching of Figure 3's graph as the differences between the best solution's RIC values and those of the population are highlighted in models that do not maintain solution strength across their populations. Especially in the case of objective function four's 0.30 mutation population, there is a huge shift from being a non-dominated solutions (Figure 3) to being outperformed by all other options in terms of difference in the mean RIC between the best and the other population solutions. Objective function seven's solution at 0.20 mutation chance is much more attractive in this case, ensuring a high amount of choice among the population while maintaining solution strength as measured by RIC. Table 5 illustrates the difference in scores between the RGA metric

Table 4: Model values for mean RIC and variance

Objective Function	Mutation	Mean RIC	Variance
1	0.10	914.26	1659.27
1	0.20	912.47	1719.37
1	0.30	910.62	2036.02
2	0.10	908.12	1634.60
2	0.20	907.88	1848.24
2	0.30	917.73	1980.37
3	0.10	912.78	1809.86
3	0.20	905.98	2148.49
3	0.30	915.06	1957.72
4	0.10	900.21	2049.20
4	0.20	892.05	1914.53
4	0.30	897.17	2337.90
5	0.10	1035.88	1834.70
5	0.20	943.77	2055.12
5	0.30	945.71	2350.03
6	0.10	906.64	1718.75
6	0.20	895.96	1897.36
6	0.30	892.43	2106.93
7	0.10	899.35	1790.44
7	0.20	895.93	2240.05
7	0.30	894.83	2080.14
8	0.10	912.35	1778.66
8	0.20	916.00	1990.66
8	0.30	909.84	2036.36
9	0.10	885.94	1964.35
9	0.20	889.40	1752.80
9	0.30	893.25	2121.09

Figure 3: Pareto Optimal Frontier for Mean RIC ~ var(Decision Variables)



and Table 4. Some mean RIC values are exposed to have large differences between the solution and best RIC values in the population. Objective function seven's solution at 0.20 mutation chance is highlighted in Figure 5, demonstrating the wide range in decision variable values that exist across the population while maintaining a relatively low RIC difference score.

6.4 Choosing Alternative Solutions

Examining the different solutions that a model offers to meet stakeholder goals is an excellent advantage of having different solutions, but once a project is in process, choosing an alternative solution from within the population of solutions is feasible only when the alternative solution also starts the same tasks prior to the change point. In other words, the same construction tasks must be started by the alternative solution as well as the current solution prior to the given point in time at which an alternative is sought. If no such solution exists, then an alternative solution from the present population does not exist and the algorithm

Figure 4: Pareto Optimal Frontier for R_{GA} Numerator and Variance of Decision Variables

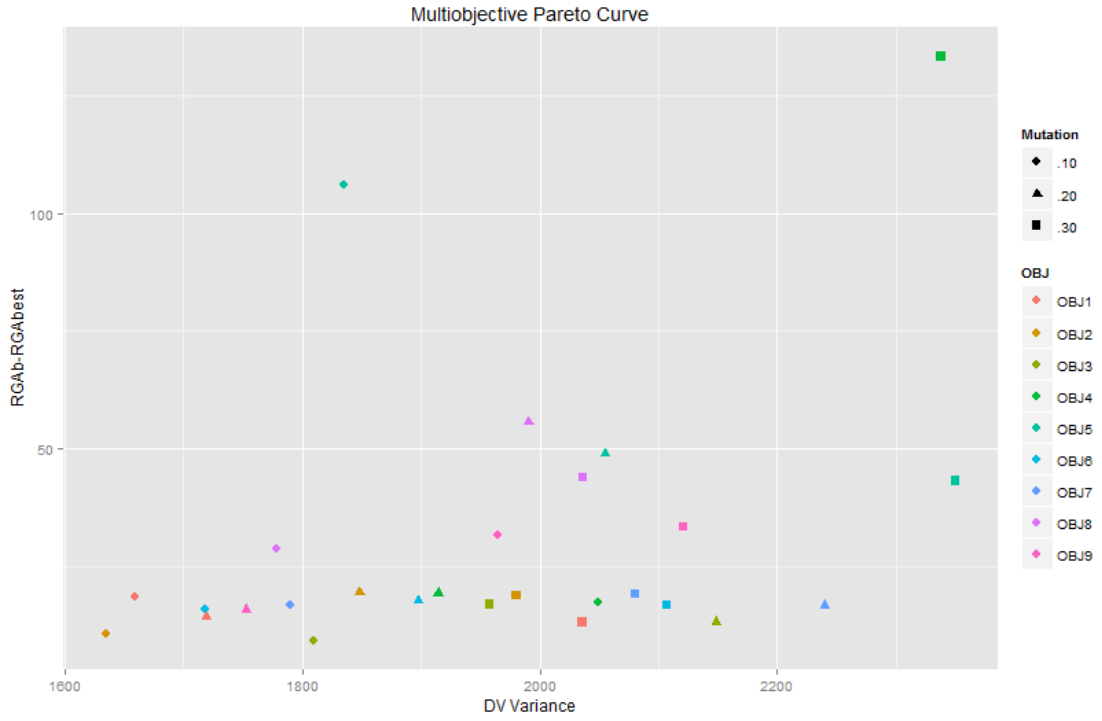


Figure 5: Decision Variable Values - First 10 Solutions

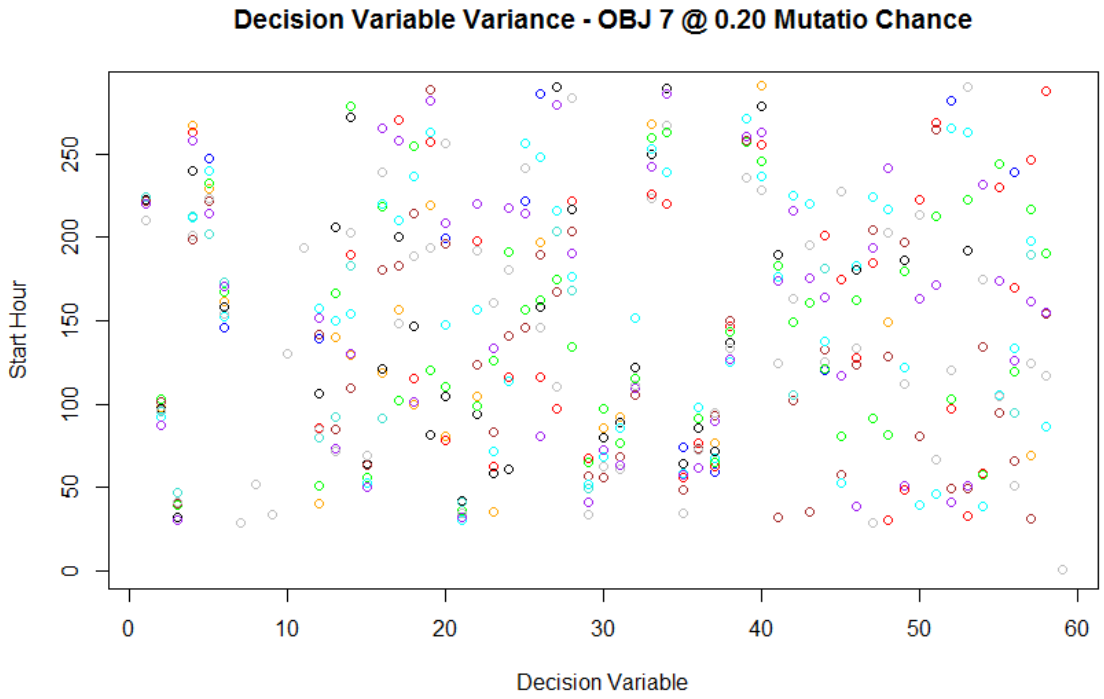


Table 5: Model values for Best RIC, Variance, and R_{GA}

Objective Function	Mutation	$\sum_{a=1}^{A} RIC_a - RIC_{best}$	Variance	R_{GA}
1	0.10	18.80	1659.27	0.2129
1	0.20	14.49	1719.37	0.1221
1	0.30	13.33	2036.02	0.08733
2	0.10	10.86	1634.60	0.0722
2	0.20	19.58	1848.24	0.2075
2	0.30	18.89	1980.37	0.1802
3	0.10	9.38	1809.86	0.0486
3	0.20	13.41	2148.49	0.0837
3	0.30	17.02	1957.72	0.1481
4	0.10	17.49	2049.20	0.1492
4	0.20	19.46	1914.53	0.1977
4	0.30	133.18	2337.90	7.586
5	0.10	106.08	1834.70	6.132
5	0.20	49.01	2055.12	1.168
5	0.30	43.29	2350.03	0.7974
6	0.10	16.05	1718.75	0.1498
6	0.20	17.96	1897.36	0.1700
6	0.30	16.88	2106.93	0.1351
7	0.10	16.80	1790.44	0.1575
7	0.20	16.92	2240.05	0.1279
7	0.30	19.27	2080.14	0.1785
8	0.10	28.85	1778.66	0.4679
8	0.20	55.64	1990.66	1.556
8	0.30	44.08	2036.36	0.9542
9	0.10	31.91	1964.35	0.5186
9	0.20	16.03	1752.80	0.1467
9	0.30	33.59	2121.09	0.5320

must be re-initiated. Distance measures such as a euclidean or Canberra distance offer a readily accessible method for comparing solutions at a decision point, but I discovered that many of the solutions identified were only near-feasible due to not starting some of the tasks prior to the change point. A simpler way of examining feasible solutions is by performing a logical test. In this logical test we will determine if both elements of the current and each alternative solution's decision values are on the same side of the change point. If they are the same, then this solution can be a possible alternative moving forward in the project. For each integer time point i in the project, the logical test in (8) is performed for all m decision variables from the best solution a to all other 49 solutions.

$$x_{a,m} \,\&\, x_{a+1} \leq i \;\|\; x_{a,m} \,\&\, x_{a+1} \geq i \tag{8}$$

Those instances resulting in answer of *true* for all i integer time periods from zero to the change point are summed across each time period for the model, which is displayed in Figure 6. This graph displays the rapid decay of solutions at approximately-one-third of the way through the project. At the outset of the project the planner or stakeholder may consider the alternate solutions that remain feasible up until such a time that (8) no longer results in any response of *true*. The area under the curve from Figure 6 represents the total number of time-sensitive solutions available from a given model in this project, and is compared against the variance of the model (Figure 7) and the numerator from R_{GA} (Figure 8). Objective function 3, $Z = min \sum_i^T \mid Y_i - \overline{Y} \mid$,with 0.20 mutation chance was the only model from both Figures 7 and 8 that is non-dominated with respect to feasible functions over time, variance of the decision values, and robust solution strength as measured by the metric $RIC_a - RIC_{best}$. This objective function reduces the absolute difference between each time period and the average resource usage for that resource type over the duration of the project.

While this objective function has particularly good scores across the population with respect to RIC, the contrast between model variance and model feasibility is highlighted between Figures 7 and 8. This tradeoff is important in that seeking a population that has high feasibility across the project (robust population) may result in lower variance among the decision variables, which has consequences for elitism and mutation rate. A lower elitism, where less solutions are culled from the population in each generation, would certainly increase feasibility of different solutions since many of the solutions would be related by crossover from one generation to the next. Additionally, a lower mutation rate would also lower the variance of the decision variables and possibly increase alternate solution feasibility, but at a cost to the genetic algorithm's ability to explore large solution spaces, which is why it is selected to solve the RLP in the first place.

Figure 6: Feasible Alternate Solutions by Model over Project Length

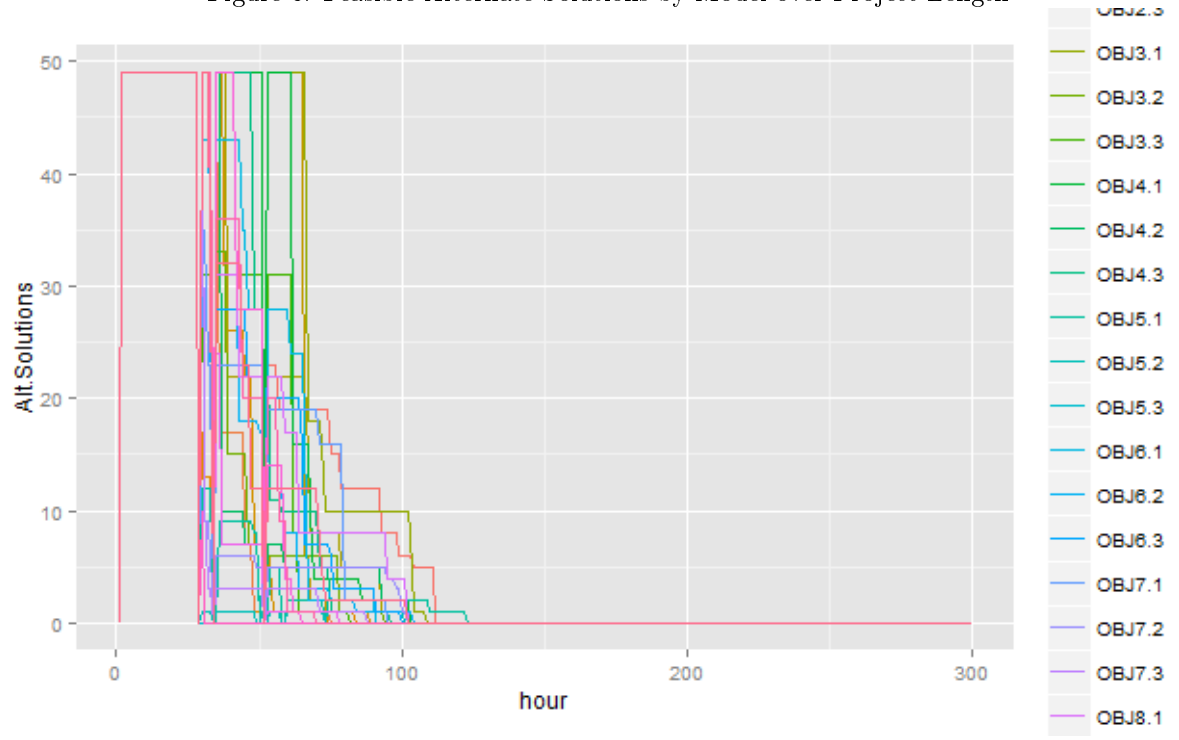


Figure 7: Pareto Optimal Frontier AUC from (8) and Variance of Decision Variables

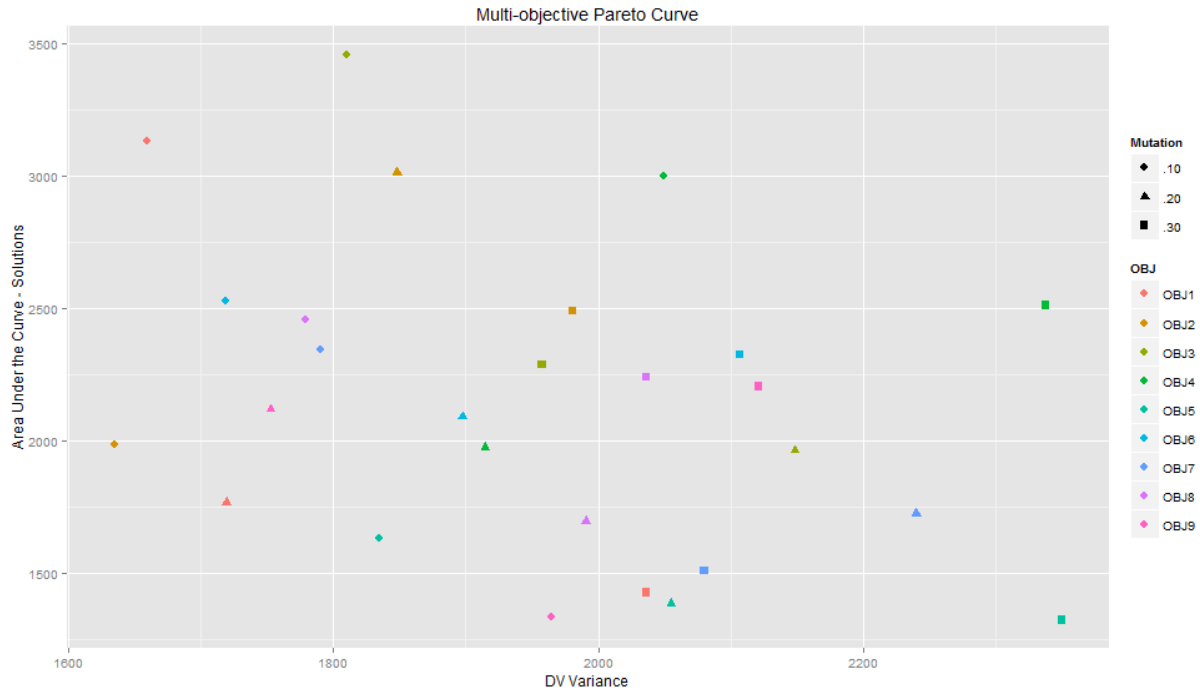
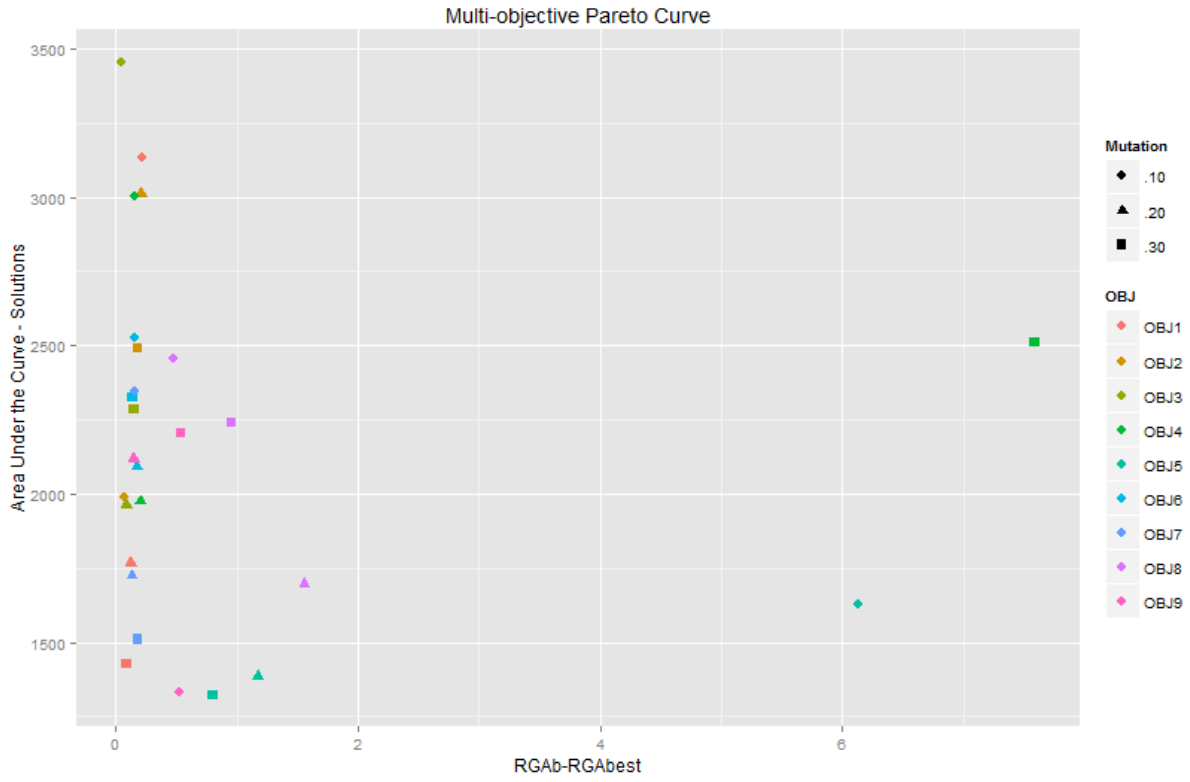


Figure 8: Pareto Optimal Frontier AUC from (8) and R_{GA} Numerator



7 Conclusion

Planning and managing a construction project is an operation that contains many variables outside of the control of any one person. Weather, worker health, equipment failure, and other stochastic elements may affect the project at any point in time and require the manager to make a change to the plan. In searching for effective ways to meet the computationally difficult goal of resource leveling during a construction project, genetic algorithms are shown to offer a population which has been unexploited for provided alternative solutions. These alternates offer multiple scheduling options which preserve near-optimality. The alternate solutions are shown to remain feasible after the project has started, up to a certain point. After this point, the algorithm must be re-initiated with updated parameters. Combining the features of robustness, variability, and normalized comparison between resources of different capacities with the Resource Improvement Coefficient established by [5] allows for a powerful comparison of the solutions within the population from

genetic algorithms not previously made in resource leveling. A population with many alternate solutions over the course of the project therefore maintains a strong robustness to its best solution. Comparing these solutions shows that there exists a marked tradeoff between the variability of the decision variable values and the amount of time-feasible solutions that may be found in a model.

This lowering of variability to achieve a measure of robustness in the population by selecting alternative solutions from the model population represents specific consequences for the genetic algorithm as a search heuristic. Genetic algorithms rely on mutation and crossover to spread the population across the solution space. An effort to reduce this variability in order to achieve more feasible alternative solutions within the population would reduce the genetic algorithm's ability to get as close to optimal as possible. More likely, the best solution returned and the other solutions would be stuck closer together in local optima and could be victims of a higher rate of genetic drift, where all solutions begin to resemble one another. Further research in this area could include the use of specific objective functions to simultaneously encourage feasibility and solution strength while still capitalizing on the strengths of the genetic algorithm to examine a large search space thoroughly. Multi-objective genetic algorithms could be used in order to achieve a composite score in order to satisfy this goal, and merits further investigation. Another area of interest is the application to extremely large-scale projects using very large populations, in order to gauge if population size will have a beneficial effect on increasing the length of time during the project that feasible solutions might still be found within the population. In this way, the genetic algorithm's expensive computational requirements in terms of the large population could be exploited for alternatives.

Bibliography

[1] D. E. Brown and W. T. Scherer, Eds., *Intelligent Scheduling Systems*. Boston: Kluwer Academic Publishers, 1995.

[2] S. EASA, "Resource Leveling in Construction by Optimization," *Journal of Construction Engineering and Management-ASCE*, vol. 115, no. 2, pp. 302–316, JUN 1989.

[3] R. HARRIS, "Packing Method for Resource Leveling (PACK)," *Journal of Construction Engineering and Management-ASCE*, vol. 116, no. 2, pp. 331–350, JUN 1990.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Bell Laboratories, 1979.

[5] R. B. Harris, *Precedence and Arrow Networking Techniques for Construction*. New York: John Wiley and Sons, Inc., 1978.

[6] J. Luis Ponz-Tienda, V. Yepes, E. Pellicer, and J. Moreno-Flores, "The Resource Leveling Problem with multiple resources using an adaptive genetic algorithm," *Automation in Construction*, vol. 29, pp. 161–172, JAN 2013.

[7] M. Hariga and S. M. El-Sayegh, "Cost Optimization Model for the Multiresource Leveling Problem with Allowed Activity Splitting," *Journal of Construction Engineering and Management-ASCE*, vol. 137, no. 1, pp. 56–64, JAN 2011.

[8] H. Alsayegh and M. Hariga, "Hybrid meta-heuristic methods for the multi-resource leveling problem with activity splitting," *Automation in Construction*, vol. 27, pp. 89–98, NOV 2012.

[9] J.-q. Geng, L.-p. Weng, and S.-h. Liu, "An improved ant colony optimization algorithm for nonlinear resource-leveling problems," *Computers and Mathematics with Applications*, vol. 61, no. 8, SI, pp. 2300–2305, APR 2011, 3rd International Symposium on Nonlinear Dynamics, Shanghai, PEOPLES R CHINA, SEP 25-28, 2010.

[10] S. Leu and T. Hung, "An optimal construction resource leveling scheduling simulation model," *Canadian Journal of Civil Engineering*, vol. 29, no. 2, pp. 267–275, APR 2002.

[11] M. Mlakar, T. Tusar, and B. Filipic, "Comparing Solutions under Uncertainty in Multiobjective Optimization," *Mathematical Problems in Engineering*, 2014.

[12] Y. Jin and H. Branke, "Evolutionary optimization in uncertain environments - A survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, JUN 2005.

[13] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, OCT 2004.

[14] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, SI, pp. 2044–2064, MAY 15 2010.

[15] S. Tsutsui and A. Ghosh, "Genetic Algorithms with a Robust Solution Searching Scheme," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, pp. 201–208, SEP 1997.

[16] A. Damci and G. Polat, "Impacts of Different Objective Functions on Resource Leveling in Construction Projects: A Case Study," *Journal of Civil Engineering and Management*, vol. 20, no. 4, pp. 537–547, AUG 2014.

[17] C. Zhu, R. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, DEC 1997.

[18] Y. Tang, R. Liu, and Q. Sun, "Two-Stage Scheduling Model for Resource Leveling of Linear Projects," *Journal of Construction Engineering and Management*, vol. 140, no. 7, JUL 2014.

[19] S. Adabi, A. Movaghar, and A. M. Rahmani, "Bi-level fuzzy based advanced reservation of Cloud workflow applications on distributed Grid resources," *Journal of Supercomputing*, vol. 67, no. 1, pp. 175–218, JAN 2014.

[20] A. J. Nyberg, T. P. Moliterno, D. Hale, Jr., and D. P. Lepak, "Resource-Based Perspectives on Unit-Level Human Capital: A Review and Integration," *Journal of Management*, vol. 40, no. 1, pp. 316–346, JAN 2014.

[21] A. Damci, D. Arditi, and G. Polat, "Resource Leveling in Line-of-Balance Scheduling," *Computer-aided Civil and Infrastructure Engineering*, vol. 28, no. 9, pp. 679–692, OCT 2013.

[22] R.-C. Lin, M. Y. Sir, and K. S. Pasupathy, "Multi-objective simulation optimization using data envelopment analysis and genetic algorithm: Specific application to determining optimal resource levels in

surgical services," *Omega International Journal of Management Science*, vol. 41, no. 5, pp. 881–892, OCT 2013.

[23] L. He and L. Zhang, "Dynamic priority rule-based forward-backward heuristic algorithm for resource levelling problem in construction project," *Journal of the Operational Research Society*, vol. 64, no. 8, SI, pp. 1106–1117, AUG 2013.

[24] I. Nieuwoudt and J. H. van Vuuren, "Algorithms for a shared resource scheduling problem in which some level of conflict is tolerable," *Journal of Scheduling*, vol. 15, no. 6, pp. 681–702, DEC 2012.

[25] S.-Y. Lin and C.-Y. Chang, "Genetic Algorithm Based Iterative Two-Level Algorithm for Resource Allocation Problems and Applications," *International Journal of Innovative Computing Information and Control*, vol. 8, no. 10B, pp. 7157–7168, OCT 2012.

[26] T. Gather, J. Zimmermann, and J.-H. Bartels, "Exact methods for the resource levelling problem," *Journal of Scheduling*, vol. 14, no. 6, pp. 557–569, DEC 2011.

[27] M. Drotos and T. Kis, "Resource leveling in a machine environment," *European Journal of Operational Research*, vol. 212, no. 1, pp. 12–21, JUL 1 2011.

[28] S. H. H. Doulabi, A. Seifi, and S. Y. Shariat, "Efficient Hybrid Genetic Algorithm for Resource Leveling via Activity Splitting," *Journal of Construction Engineering and Management-ASCE*, vol. 137, no. 2, pp. 137–146, FEB 2011.

[29] M. Gorczyca and A. Janiak, "Resource level minimization in the discrete-continuous scheduling," *European Journal of Operational Research*, vol. 203, no. 1, pp. 32–41, MAY 16 2010.

[30] S. E. Christodoulou, G. Ellinas, and A. Michaelidou-Kamenou, "Minimum Moment Method for Resource Leveling Using Entropy Maximization," *Journal of Construction and Engineering Management*, vol. 136, no. 5, pp. 518–527, MAY 2010.

[31] L. Zhang, J. Du, and S. Zhang, "Solution to the Time-Cost-Quality Trade-off Problem in Construction Projects Based on Immune Genetic Particle Swarm Optimization," *Journal of Management in Engineering*, vol. 30, no. 2, pp. 163–172, MAR 1 2014.

[32] J. Rieck, J. Zimmermann, and T. Gather, "Mixed-integer linear programming for resource leveling problems," *European Journal of Operational Research*, vol. 221, no. 1, pp. 27–37, AUG 16 2012.

[33] G. K. Koulinas and K. P. Anagnostopoulos, "A new tabu search-based hyper-heuristic algorithm for solving construction leveling problems with limited resource availabilities," *Automation in Construction*, vol. 31, pp. 169–175, MAY 2013.

[34] A. Senouci and N. Eldin, "Use of genetic algorithms in resource scheduling of construction projects," *Journal of Construction Engineering and Management-ASCE*, vol. 130, no. 6, pp. 869–877, NOV-DEC 2004.

[35] D. o. t. A. Headquarters, "The Operations Process," *Army Doctrine Publication 5-0*, MAY 2012.

[36] S. A. McChrystal, "COMISAF's Initial Assessment," AUG 2009.

[37] D. Patraeus and J. Amos, "Counterinsurgency," *FM3-24*, DEC 2006.

A Github Repository

The following Github page https://github.com/dfd3mt/Robust-Solutions-in-Resource-Leveling-GA-Populations is a repository for the input file and R scripts for each objective function, genetic algorithm run, metric function, and graphs used herein.

B Kurtosis Values

Table 6: Kurtosis of individual resources by model

Model	1	2	3	4	5	6	7	8	9
Auger	144.0216	144.0216	144.0216	144.0216	144.0216	144.0216	144.0216	144.2456	144.0216
Backhoe	4.3137	3.7904	1.8294	2.2026	2.898556	3.461732	3.9848	1.6414	3.9371
Carpenter	-0.3361	-1.1277	-0.7261	-0.8854	-0.504387	-0.8976865	-1.1166	-1.0129	-0.9463
Supervisor	45.0478	22.1423	45.0478	28.1555	28.15551	28.15551	24.8479	25.9921	28.1555
Water Distributor	51.9208	31.7496	32.7469	5.6527	4.166488	5.999615	50.7804	5.5411	5.9996
Dozer w/heavy Ripper	37.6089	32.2909	37.6089	44.7008	44.70081	44.70081	37.6089	41.3075	44.7008
Dozer w/medium Ripper	30.3788	31.5295	30.3788	29.2899	29.2899	28.0647	30.3788	9.8530	29.2899
Excavator	18.6878	17.7363	16.4909	15.6693	14.7068	15.2965	21.5119	20.0629	18.6146
Forklift	-0.4031	-0.8273	-0.9600	-0.4418	-0.7351	0.0370	-0.0566	-0.2061	-0.4166
General Labor	0.0217	0.0186	1.5031	-0.6432	-0.7007	0.1341	-0.5488	-0.1327	0.9866
Grader	4.0813	0.0075	3.2935	4.2537	1.2889	0.2283	1.8337	3.3065	5.1667
Heavy Equipment Operator	18.2086	18.2086	19.8883	18.2086	19.8883	19.8883	18.2086	19.3025	18.2086
Electrician	-1.2209	-1.2051	-1.0678	-1.2170	-1.1806	-1.1766	-1.1232	-1.1400	-1.1348
Loader	-0.3710	-0.0531	-0.3710	-0.3131	-0.3131	-0.9402	0.3353	0.0099	-0.3131
Metal Worker	7.0237	7.0237	7.0237	5.7585	11.7249	16.5927	7.0237	10.9922	16.5927
Paver	54.3441	60.4501	64.6436	28.2134	33.8493	13.9676	39.1883	12.4298	29.5632
Plumber	3.4266	10.7648	3.4266	7.3533	3.4266	10.7648	3.4266	5.6906	3.4266
Roller Vibratory	31.2983	22.4698	31.2983	32.4400	35.2254	24.2662	44.6305	11.2306	35.2254
Scraper	13.9724	13.9724	13.9724	13.3869	13.9724	13.9724	13.9724	3.9082	13.9724
Sweeper	2.4250	8.5766	2.4336	2.4625	2.3164	8.8381	2.4410	2.8139	2.3149
Dump Truck	4.9458	1.4614	5.0025	5.3653	4.8378	1.2211	5.2673	6.7944	5.3653

C Sample hourly use plots

Figure 9: Auger use from initial state (left) to Model 6 (right)

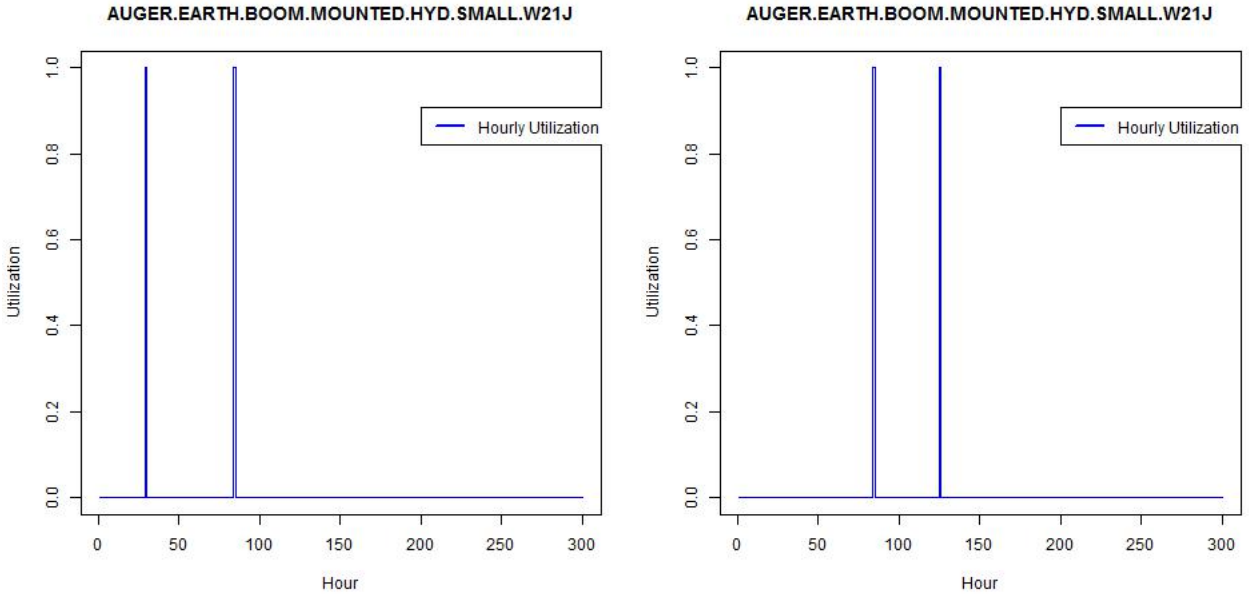


Figure 10: Backhoe use from initial state (left) to Model 6 (right)

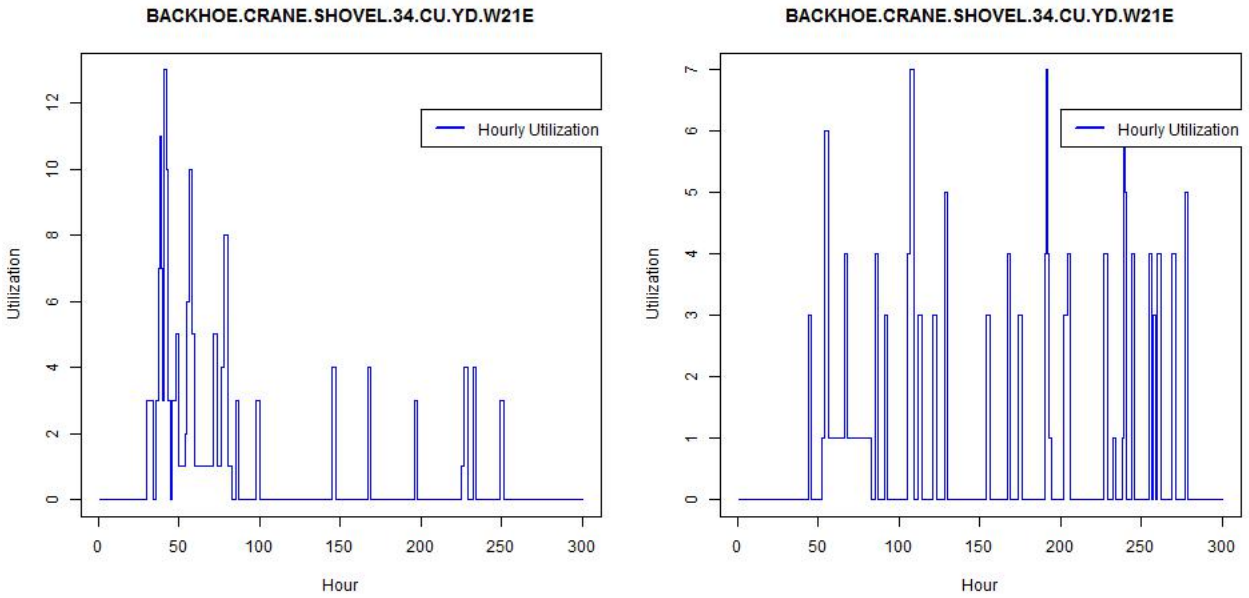


Figure 11: Carpenter use from initial state (left) to Model 6 (right)

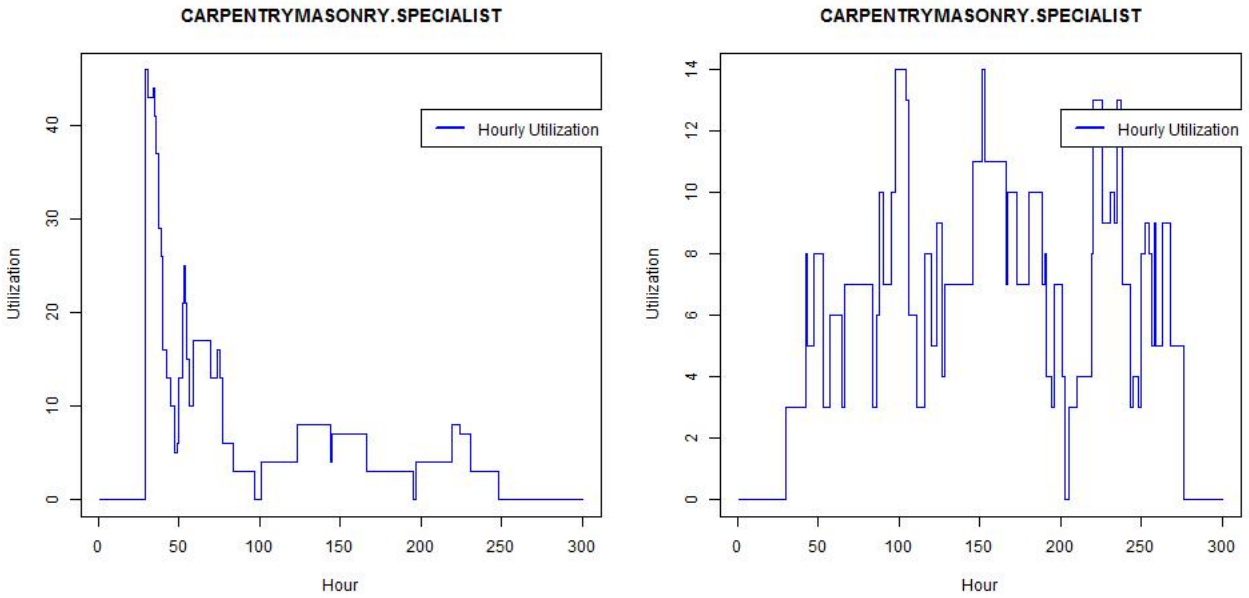


Figure 12: Supervisor use from initial state (left) to Model 6 (right)

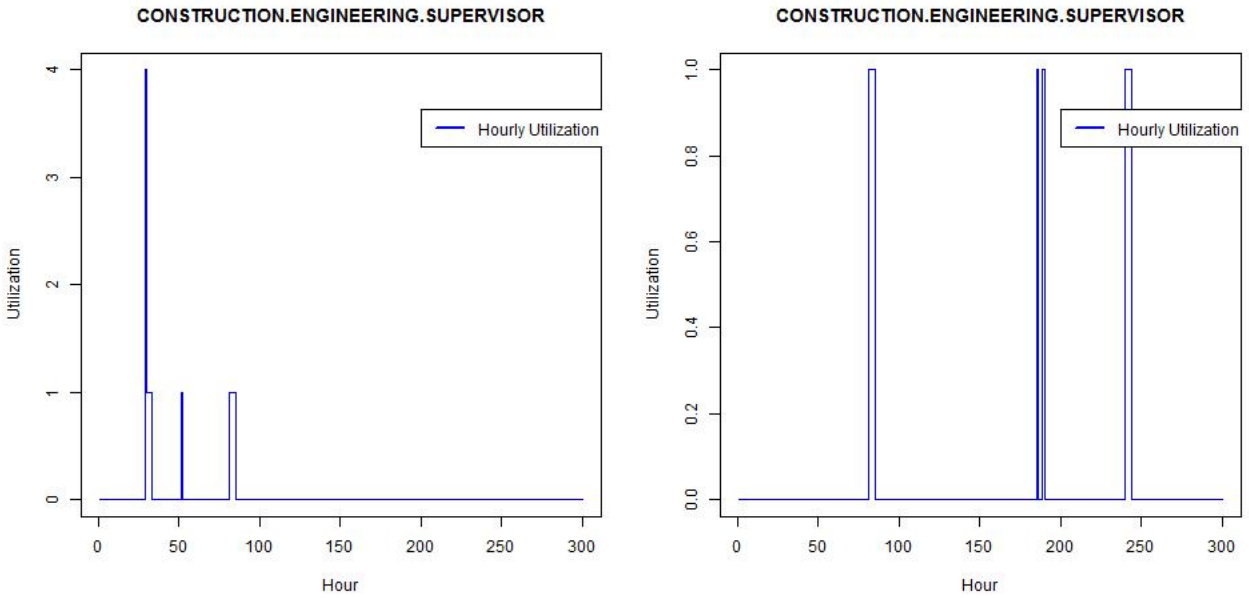


Figure 13: Water Truck use from initial state (left) to Model 6 (right)

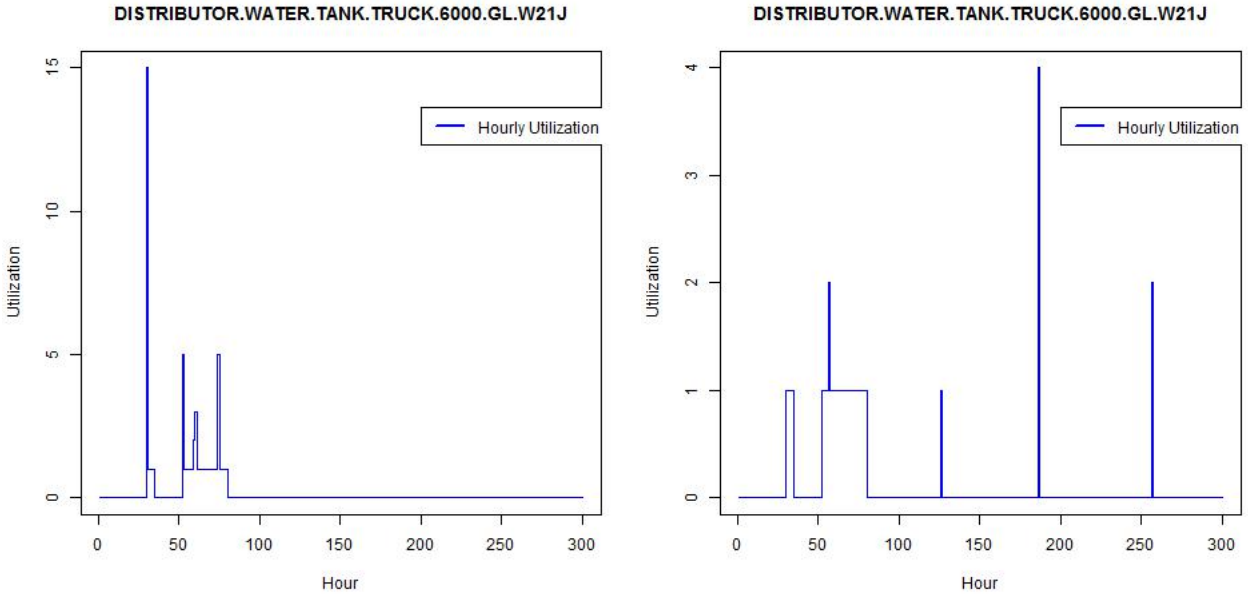


Figure 14: Excavator use from initial state (left) to Model 6 (right)

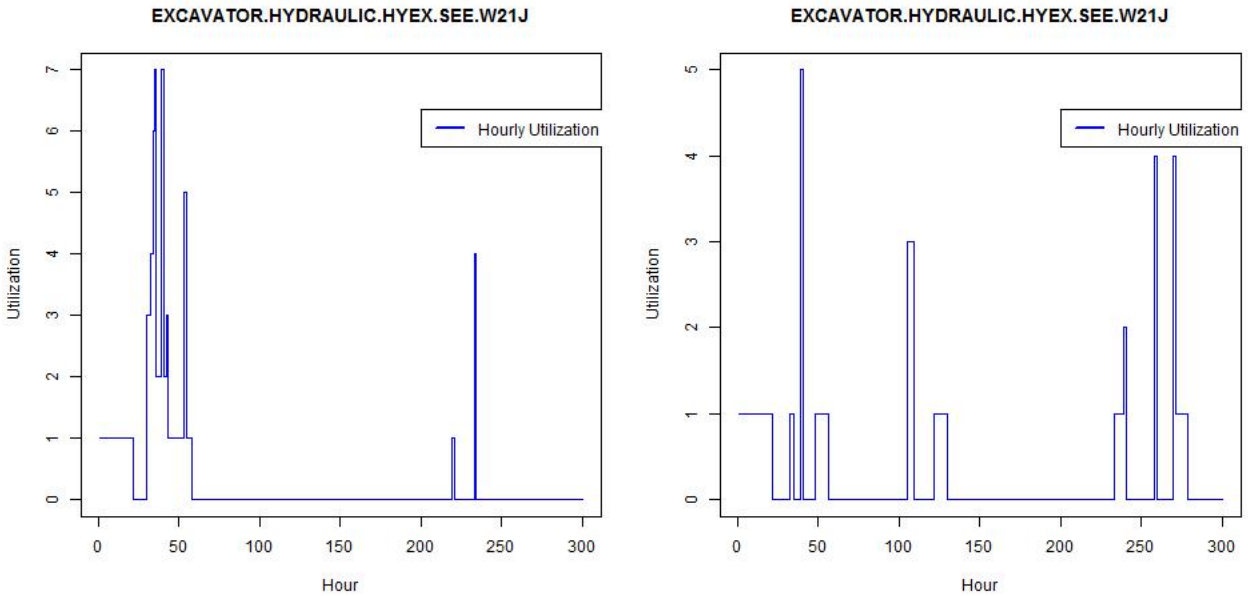


Figure 15: Forklift use from initial state (left) to Model 6 (right)

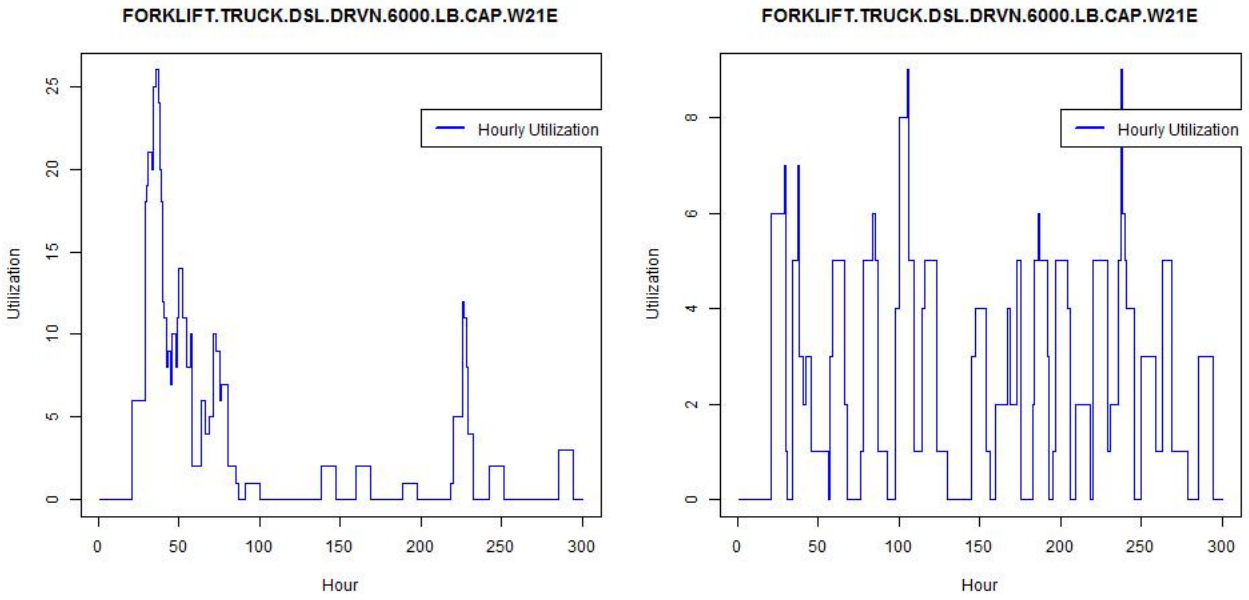


Figure 16:

Figure 17: General labor use from initial state (left) to Model 6 (right)

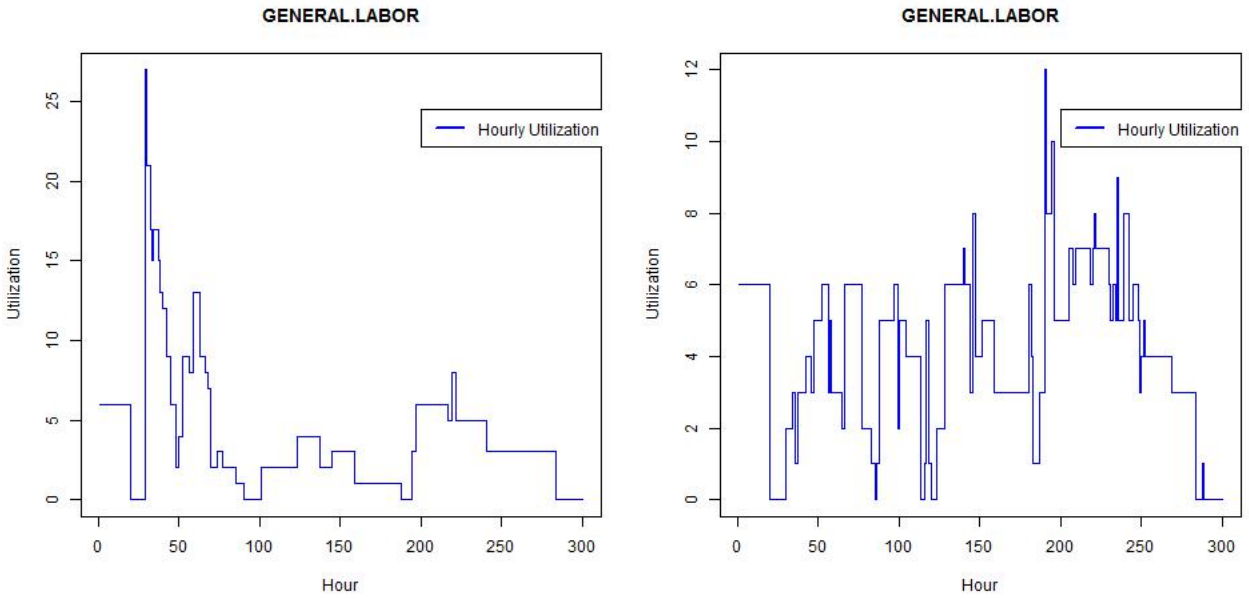


Figure 18: Towed sweeper use from initial state (left) to Model 6 (right)

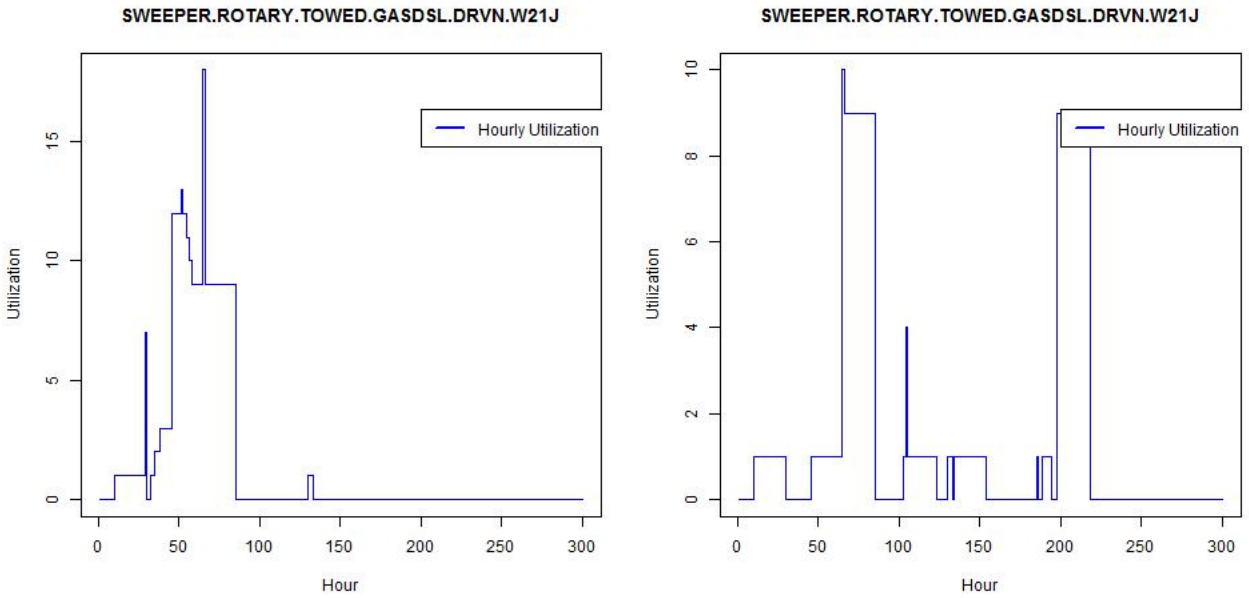


Figure 19: Dump Truck use from initial state (left) to Model 6 (right)