LEAN RESOURCE SCHEDULING ALGORITHM WITH MAXIMIZED RESOURCE UTILIZATION USING ITERATIVE LOCAL SEARCH

A Thesis

by

RYAN CHRISTOPHER SKINNER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jorge Leon
Committee Members,	Amarnath Banerjee
	Bimal Nepal
Head of Department,	Mark Lawley

December 2018

Major Subject: Industrial Engineering

ABSTRACT

This thesis presents a lean resource scheduling algorithm which merges traditional machine scheduling problems with Lean Manufacturing concepts to determine the resource levels, such as employee headcount or number of machines used in production, and the corresponding schedule which minimize resource idle time while keeping scheduled makespan within a neighborhood around the takt-time. The algorithm begins by solving a relaxed problem to find a satisfactory makespan via iterative local search, then solving a secondary problem to minimize the idle time subject to a makespan neighborhood constraint.

Experiments were conducted on a randomly generated dataset with six different factors, and both the overall program run time and the amount of idle time reduction between the first feasible solution and final solution were measured. The algorithm executes in a relatively short time, even for moderately large problem instances, and the idle time reductions are promising at a grand average of twenty-five percent reduction.

The results of the algorithm are promising on the test sets, although the method has not been tested in a practical case study. Given the promising results, further study on the underlying model, algorithm performance, and testing in a practical application are recommended.

ACKNOWLEDGEMENTS

Thank you to my mother, father, and sister for their never-ending support during my time at Texas A&M. Nothing in my life would be possible without their love and support. Additionally, I'd like to thank Gail Smith for her guidance in my professional career, support, and generosity during this process.

Thank you to Dr. Jorge Leon for his mentorship throughout my undergraduate and graduate research and Dr. David Beirling for his mentorship throughout my early academic career and providing me opportunities to grow and develop at TTI. My experiences under Dr. Leon's and Dr. Beirlings' mentorship have had an enormous impact on my academic and professional career.

To Mr. Brent Lyon and Mr. Scott Swann, thank you for allowing me to continue my professional career while completing this thesis, and for your flexibility and understanding during the process. Thank you to Ms. Lauren Gibbens and Mr. Justin Loos, for the inspiration for this work and serving as this works' impromptu "Industry Panel".

Finally, thank you to my all my friends and family, particularly Kate, Cullen, Davis, and Lauren for their support during graduate school.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a dissertation committee consisting of Dr. Jorge Leon and Dr. Amarnath Banerjee of the Department of Industrial Engineering and Dr. Bimal Nepal of Industrial Distribution. All work for the thesis was completed independently by the student. No outside funding was received for the research and compilation of this document.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

This thesis presents a methodology that aims to finds the resource levels and corresponding schedule that minimizes the total cost weighted idle time of the schedule, given that the schedules makespan is within a tolerable window about a target value. This work was motivated by the Lean Manufacturing principles and scheduling heuristic optimization solutions for sequencing and scheduling problems within the aerospace assembly environments. Solving a first-stage search to find a resource set with a corresponding feasible completion time and then performing a second search to minimize the idle time of the schedule within the feasible region strikes a balance between completing a planned scope of work within the allowable time while still minimizing the idle time.

The Lean manufacturing paradigm was developed in the early 1940's and made famous the Toyota Production System (TPS). Lean manufacturing can be broadly characterized by delivering what a customer values, while using the least amount of resources possible. The concept of lean manufacturing is well-presented by Womack and Jones' [7] who identified five lean principles, Identify Value, Map the Value Stream, Create Flow, Establish Pull, and Seek Perfection. Lean Manufacturing drives manufacturers towards creating high value, in-demand products for their customers while using the least amount of effort, such as raw materials, labor, or capital, as possible. A central concept in lean manufacturing is the takt time, which is the time between a fully completed job. Work balancing, workstation capacity, and employee staffing levels are all adjusted in lean planning to ensure that jobs are completed near the takt time.

Implementation of lean manufacturing principles in industries with high product variability and low product volume, such as the aerospace industry, has had limited success [9]. Despite the limited success to date, Aerospace manufacturers are striving to be leaner since successful implementation of Lean Manufacturing principles can result in a large competitive advantage by increasing the overall production throughput and reducing setup and change-over times, which in turn allows greater schedule stability and lower rework costs [10].

This methodology bridges the gap between traditional sequencing and scheduling works and Lean Manufacturing principles by minimizing the total idle time of resource over the takt time, instead of the completion time of the schedule. Extending the definition of idle time to penalize schedules that are completed before the takt-time ensures the methodology finds employee and machine resource levels that result in a makespan that is near the takt time with a minimum resource idle time. This methodology applies methodologies and techniques common in machine scheduling to Lean manufacturing scheduling.

2. MOTIVATION

In modern manufacturing and service environments decision makers face conflicting pressures to complete the planned work within an allotted time while keeping the overall manufacturing costs as low as possible. Generally, these two objectives are in competition with each other, as employee staffing can have a significant impact on the rate of production or service. This work aims to address these conflicting objective functions by creating schedules that satisfy completion time objectives while minimizing the total idle time or costs of utilizing the resources needed during the schedule. These two objectives, completing work within on a consistent, planned time and minimizing the amount of resources aid in achieving Lean Manufacturing's primary objective, using resources efficiently. This work is applicable to environments where the planned scope of work is consistent and stable across a longer, strategic planning horizon.

In Aerospace assembly environments, the overall job variability is low, and orders are known well in advance and can be adequately planned for. In aerospace assembly, the full assembly of the airplane is broken up into different areas, with well-defined and appropriately scoped job content, where a specific set of tasks or operations are completed to finish a subassembly of the overall aircraft. In this environment, the decision maker can use the proposed methodology to make a strategic decision on what long term staffing levels and schedule combinations result in a minimized idle time or idle cost.

This methodology is not limited only to traditional manufacturing environments, as this methodology can be effectively applied in other fields where job variability is low, and the scope of work is large and well defined. Large-scale construction environments share many similarities with the aerospace manufacturing environment, low job variability, high number of distinct operations with various resource requirements, and this methodology can be applied to reduce the overall employee costs associated with completing a major construction project.

3. LITERATURE REVIEW

The concept of Lean Manufacturing, made popular by the Toyota Production system, was developed and refined in Toyota's in "Lean" supply and distribution base in the 1970's and 1980's [8] and has developed into a central manufacturing ideology in the world's automotive industries. Two central ideals in Lean Manufacturing are converting the manufacturing to single-piece flow, where each job is processed sequentially with no build-up of work in progress (WIP) in-between manufacturing steps and completing jobs a steady and continuous rate. Successful implementation of single-piece flow and completing jobs on the takt-time has a large impact reducing excessive WIP within a system [11]. Companies that can successfully schedule work in accordance to the takt time and execute to the schedules can see significant reductions in WIP, both reducing the capital invested in in-work goods and improving the visibility of the manufacturing process.

The field of machine scheduling has been well-studied and has been applied to many different industries and problems. A.H.G Rinnooy Kan's '*Machine Scheduling Problems: Classification, Complexity and Computations'* [4] provides an insightful overview of machine scheduling problems. French's '*Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop (Mathematics and its Applications)* [1] provides a similar background in machine scheduling. The Job Shop Scheduling Problem (JSP) is traditionally formulated as a mixed-integer program (MIP), known in the field of machine scheduling as an exact method, and modeled and solved with various models and commercial solvers. [5] provides an overview of the different modeling approaches, and the speeds of different commercial solvers. When modeled as an MIP, there are several restrictive assumptions that prohibit modeling realistic

manufacturing systems. Most MIP models have prohibitions against using more than one unit of resource capacity during a given operation and many models give an explicit 1:1 assignment between jobs and machines causing each operation to be assigned to one and exactly one machine.

In contrast with the exact method, an approximate or heuristic method can model a more realistic production system where each operation may require more than one unit of a given resource type and require more than one resource type at a time. For example, a given operation may need two units of one resource type and three units of the second. To have a more robust solution methodology an approximate method was chosen.

Iterative Local Search (ILS) a popular approximate method used to solve optimization problems that, due to problem size or restrictive constraints, cannot be reasonably solved using exact methods. Lourenco [12] provides an overview ILS methods and applications to a variety of problems. In machine scheduling, ILS is used in conjunction with heuristic scheduling methods to solve various machine scheduling problems. Panwalkar and Iskandar [6] provide a survey of the various scheduling rules or heuristics that may be applied to scheduling problems. Scheduling heuristics have been successfully applied by Leon and Balakrishnan [2], who combine scheduling heuristics with a special case of ILS known as problem-space based neighborhood search to efficiently solve the resource constrained scheduling problem in job shop environments. Additionally, Leon and Balakrishnan [3] have shown this procedure develops strong neighborhoods with strong computational results.

4. PROBLEM DESCRIPTION

Consider a workstation with two different employee types that must complete two different jobs, where each job can be broken down into individual operations. Before work can begin on an operation, all the operations' predecessors must be completed. For example, Job 1 consists of four different operations. Before work can begin on Operation 2 and Operation 3, Operation 1 must be completed. Once Operation 2 has been completed, Operation 4 may be started. The job shop must process an additional job, Job 2, which consists of two operations, Operation 1 and Operation 2, which must be completed sequentially. Each combination of operation *i* and job *j*, represented as o_j^i, has an associated processing time $d(o_j^i)$ and requires $r_1(o_j^i)$ employees of employee type 1 and $r_2(o_j^i)$ employees of employee type 2. Figure 1 shows a network representation of the jobs one and two.

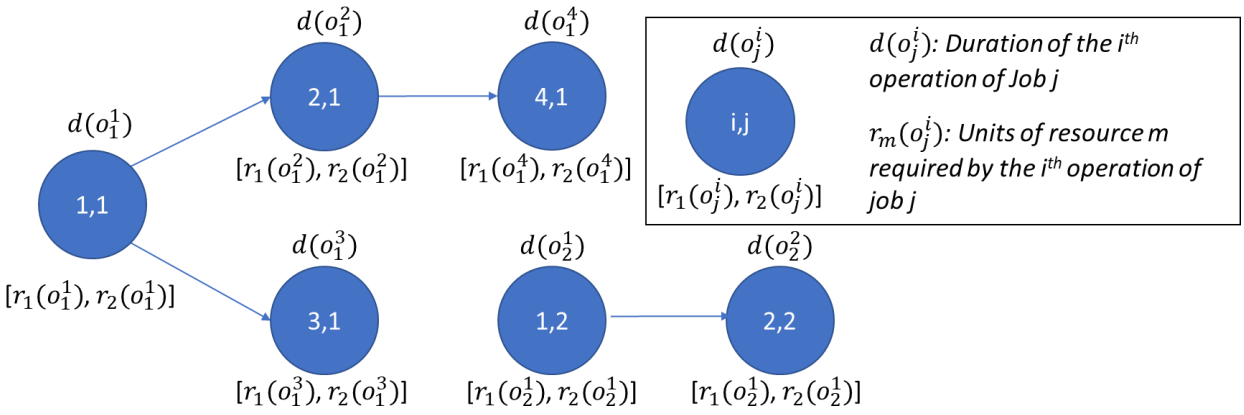


Figure 1: Network Representation of Job A and Job B

Since this workstation is part of a larger production facility with low product mix the planned work, the set of operations and operation parameters, in this workstation is stable for the foreseeable future. Given that this workstation is part of a larger assembly-line environment it is critical that both Job 1 and Job 2 are completed near a target completion time, *T*. Management would like to determine the staffing levels and associated job completion sequence that minimizes the total cost of idle time of the resources, while keeping the completion time of the planned work to the target makespan *T*, which represents the takt-time of the planned work

5. PROBLEM MODEL

This section presents a generalization of the problem described above. Define a set of replenishable resources, *R*, where each resource $m \in R$ has an integral resource capacity $R_m \in Z^+, \forall m \in R$. All values are assumed to be known and deterministic. Given a set of jobs *J*, where each job has an associated sequence of operations O_j consisting of least one operation $o_j^i, \forall j \in J, 1 \leq i \leq n_j$, where o_j^i represents the i^{th} operation of job *j* and n_j represents the number of operations in job *j*. There is a non-negative duration $d(o_j^i)$ and start time $\sigma(o_j^i)$, a set of direct predecessors $P(o_j^i)$, and a vector of replenishable resource requirements $\boldsymbol{r}(o_j^i) = [r_m(o_j^i)]$ with $r_m(o_j^i) \in Z^+$ associated with each operation. It is important to note that the operation o_j^i requires exactly $r_m(o_j^i)$ units of m and requires exactly $d(o_j^i)$ units of time to complete. Adding additional resources to the operations is prohibited and does not expedite jobs completion time. Define the makespan of the schedule $C_{max}(R_m) = \max\{\sigma(o_j^i)\} \ \forall \ o_j^i$. Define the idle time of resource type *m* to be $I_m = R_m \max\{C_{max,}\mathrm{T}\} - \sum_{j\in J}\sum_{i\in[1,n_j]} d(o_j^i) r_m(o_j^i)$ and cost weighted sum of idle times to be $I = \sum_{m\in R} c_m I_m$, with c_m representing the cost unit cost of resource m per unit time.

The idle time formulation used in this problem differs from the traditional idle time show in [2,3] as the idle time is computed over the maximum of target completion time and the scheduled makespan. This change in formulation adds a penalty to schedules that complete work well before the target completion time, as the resources used in production will necessarily be idle during the time between schedule completion, C_{Max} and the target completion time T.

The scheduled start time of all operations must adhere to both technological and resource feasibility constraints. A schedule is technologically feasible if the start time of any given job is less than or equal to the start time of all its successors, $\sigma(o_j^i) + d(o_j^i) \leq \sigma(o_j^k), \forall k \in P(j), \forall j \in J$. A schedule is said to be resource feasible, for any given time and resource type, the sum of resource requirements for all in-work tasks $O_p(t) = \{o_j^i | o_j^i : \sigma(o_j^i) \leq t \leq \sigma(o_j^i) + d(o_j^i)$ is less than or equal to the resource capacity for all resources $\sum_{o_j^i \in O_p(t)} r_m(o_j^i) \leq R_m \forall m, \forall t$.

The objective of the problem is to find the resource set **R** and associated sequence of operation start times that minimizes the cost-weighted idle time, while keeping the scheduled makespan $C_{Max}(\boldsymbol{R})$ with in a neighborhood around the target makespan defined by two parameters $T_{LB} = \max\{T(1 - \alpha), 0\}$ and $T_{UB} = T(1 + \alpha)$.

6. PROBLEM FORMULATION

The minimum weighed idle time problem can be defined as follows:

$$\min_{\boldsymbol{R}} \sum_{\forall m} c_m \left(R_m \max\{\max\{\sigma(o_j^i) + d(o_j^i)\}(\boldsymbol{R}), T\} - \sum_{j \in J} \sum_{i \in [1, n_j]} d(o_j^i) r_m(o_j^i) \right)$$

subject to:

$$T_{LB} = \max\{0, T(1-\alpha)\} \quad (1)$$

$$T_{UB} = T(1+\alpha) \quad (2)$$

$$T_{LB} \leq C_{max} \leq T_{UB} \quad (3)$$

$$\alpha \in [0,1] \quad (4)$$

$$R_m \in Z^+, \forall m \quad (5)$$

$$r_m(o_j^i) \in Z^+, \forall o_j^i \quad (6)$$

$$\sum_{o_j^i \in O_p(t)} r_m(o_j^i) \leq R_m, \forall m, \forall t \quad (7)$$

$$d(o_j^i) \geq 0, \forall o_j^i \quad (8)$$

$$\sigma(o_j^i) + d(o_j^i) \leq \sigma(o_j^k), \forall k \in P(j), \forall o_j^i \quad (9)$$

The objective function is an expansion of the more compact objective function, $\min(\boldsymbol{c^T I})$, where the vector **c** represents the cost, typically in dollars or other currency, of using one unit of a resource for one unit of time and $\boldsymbol{I}$ represents the vector of idle times for each resource type. For example, if $c_m = \$3\ per\ hour$, each and $I_m = 4\ hours$ the total cost of idleness for resource type m is $12. The vector **c** = 1 represents the unweighted idle time

minimization problem. The addition of an idle-time cost allows the modeler to specify critical, expensive resources, and more heavily penalize idle time on those resources.

Constraints (1) and (2) define the makespan neighborhood sizes, while constraint (3) ensures all feasible solutions have a makespan within the target neighborhood. Constraint (4) bounds the neighborhood size parameter α. Constraints (5) and (6) restrict the resource capacities and resource requirements for all operations to the positive integers. Constraint (7) restricts all operation durations to the non-zero reals. Constraint (8) represents the technological precedence constraints addressed in the previous section.

7. SOLUTION APPROACH

The problem is decomposed into two different problems. The first problem stage, referred to as the first-stage search, finds an initial feasible solution that satisfies the target makespan constraints. After an initial feasible solution has been found, a second-stage search is performed that minimizes the total cost of idle-time associated with a feasible schedule.

The first-stage search begins with the smallest resource set **R** that results in an executable schedule. This resource set $\boldsymbol{R^{Min}} = \max\{r_m(o_j^i)\}\,\forall m, \forall o_j^i$ is set as the initial seed for the search, and the resource set neighborhood is searched for solutions with a smaller absolute difference between the resultant makespan $C_{Max}(R)$ and the target makespan T. The search continues until a resultant makespan is found that satisfies the problem's makespan constraints. Once found, this solution represents the first observed feasible solution and the algorithm continues to the second-stage search.

The initial feasible solution is set as the first seed and the neighborhood is searched for solutions with smaller idle cost values, $\boldsymbol{c^T I}$. If a resource set is found that has a smaller idle cost than the initial feasible solution, the seed is updated, and the neighborhood search continues. The search is executed until the total number of iterations have been completed, and the best-found solution is reported. Both the makespan in the first-stage search and the idle time in the second stage search are outputs of the solution generation heuristic, g(R,J,h), explained in the section below.

7.1 Schedule Generation Procedure

A schedule generation heuristic g(R,J,h) receives a set of resources (R), a set of jobs (J), and a scheduling heuristic (h) and computes the start and finish time for every operation. The schedule generation heuristic used to solve this problem is very similar to the schedule generation procedure used by Leon, et. al [2], without the neighborhood search to minimize makespan. The search procedure used in [2] is omitted from this work and the construction heuristic is all that is used to generate schedules.

The schedule generation procedure iteratively chooses an operation from the set of schedulable operations $P_s(t)$, that is operations that have all their predecessors completed and enough resources available at the time of decision to schedule the operation. When there is more than one operation in consideration, a scheduling heuristic, h, is applied which assigns a priority to the competing operations. The operation with the highest priority is scheduled, and the start and finish times of the operation are updated, along with the schedules' resource availability is updated. After scheduling an operation, $P_s(t)$ is updated to reflect the new resource availabilities. If there are more schedulable activities after the previous operation has been scheduled, the schedule generation heuristic is reapplied to the new set of schedulable activities. This procedure continues until the set of schedulable activities is empty, after which the algorithm moves next timestep t, where the procedure is applied again.

7.2 Schedule Generation Algorithm

A critical operation in the schedule generation algorithm is determining the set of schedulable activities at a particular time. The *DetermineSchedulable* algorithm returns the schedulable operations, operations that satisfy both technological and resource feasibility The algorithm first resets the set of schedulable activities, then the algorithm compares each operations total number of predecessors, $NumPredecessor(o_j^i)$, and the number of completed predecessors at time t, $CompletedPredecessors(o_j^i)$. If the number of completed predecessors matches the total number of predecessors, and $r_m(o_j^i)$ is less than the resource availability of resource m at time t, add o_j^i to the partial schedule.

DetermineSchedulable(O,R(t))

Initialize: $P_s(t) = \emptyset$

For all $o_j^i \in O$:

If $NumPredecessor(o_j^i) == CompletedPredecessors(o_j^i)$:

If $r_m(o_j^i) \leq R_m(t)\ \forall m$:

Add o_j^i to $P_s(t)$

End if

End if

The schedule generation algorithm described in this section is used to determine the start time for each operation in the planned work J. The algorithm is initialized with all operations in the set of operations, O and two sets, the set activities that are schedulable at a given time $P_s(t)$, and the set of activities that have scheduled, S'. While the set of unscheduled operations is not

empty, apply the *DetermineSchedulable* procedure to determine which tasks are schedulable at the current simulation time. If there are no schedulable tasks, advance the time to the next finish time of a task and re-apply the *DetermineSchedulable* algorithm. If the set of schedulable activities is non-empty at time t apply the schedule heuristic *h* to the set of schedulable activities and choose the task with the highest priority. The chosen operation is then moved to the set of scheduled operations, and the operations start and finish time are updated along with the resource availability at time t. Once the updates are made the *DetermineSchedulable* procedure is applied again. After all the jobs have been scheduled, the algorithm terminates and the various schedule measures, such as makespan or idle time, can be calculated.

GenerateSchedule(R,J,h)

Initialize: $O = \{o_j^i\}, P_s(t), = \emptyset, S' = \emptyset, t = 0$

While $O \neq \emptyset$:

DetermineSchedulable(O,R(t))

While $P_s(t) \neq \emptyset$:

Apply h, and select operation with the highest priority (o^*)

$\sigma(o^*) = t$, add o^* to *S*, remove o^* from *O*, update *R(t)*

DetermineSchedulable(O, R(t))

NextTime = $\min\{\sigma(o_j^i) + d(o_j^i) : \sigma(o_j^i) \geq t\}$

t = NextTime

Upon conclusion of the algorithm, all activities will be scheduled. The DetermineSchedulable procedure ensures that all generated sequences are technologically feasible, since only operations with all completed successors may be in the set of schedulable activities. Additionally, all generated sequences will adhere to resource constraints, since any operation with resource requirements exceeding the resource availability at the time of the DetermineSchedulable process. Finally, the algorithm will necessarily terminate if the resource schedule resource capacities exceed the resource requirements for all operations. The guaranteed technological and resource feasibility, guaranteed convergence, and the algorithms worst-case complexity, O(N^2)[3], ensures the schedule generation procedure quickly creates feasible sequences with modest computational effort. See Appendix A1 for a process diagram describing the schedule generation algorithm.

7.3 First Stage Search

To find a feasible solution, the problem is solved with the makespan constraint relaxed. The search is initialized with the schedule generation procedure applied to R^{Min}, the smallest resource set that results in a feasible solution to the relaxed problem. The resource set and corresponding makespan are used as the initial seed and makespan for a neighborhood search algorithm with the objective of minimizing the absolute difference between the schedules' makespan and the schedules associated target makespan. Within the first neighborhood, a perturbation vector, p is added to R^{Min} where each element in p is drawn from a (0,1) uniform distribution. The stochastic vector is drawn from a uniform distribution between zero and one to eliminate searching infeasible solutions. The initial seed represents the smallest resource set that results in a feasible solution and reducing any resource value in R^{Min} will result in an infeasible

schedule when the schedule generation technique is applied. The stochastic is multiplied by a step-size δ^1 which creates the perturbation vector. This perturbation vector is added to the seed, and the schedule generation procedure is applied to the new test set. If the test set has a makespan that is closer to the schedules' target makespan, the seed is updated, and a new perturbation vector is drawn from a (-1, 1) random uniform distribution. This search procedure continues until finding a resource set with a makespan that satisfies the original problems' makespan constraints. This resource set is then set as the initial seed for the second-stage search which minimizes the total idle cost of the feasible schedules. See Appendix A2 for a process diagram describing the first-stage search.

7.4 Second Stage Search

Once a feasible solution is found, a secondary search is conducted to find a resource set with the smallest idle cost among the feasible solutions. The first-stage search finds the first feasible solution, R^0 with a corresponding idle cost I^0. For each iteration of the search, a random perturbation vector is created with elements drawn from a (-1, 1) random uniform distribution and scaled by some parameter delta, resulting in a perturbation vector $p = \delta * U(-1,1)$, which is added to the initial seed. The schedule generation procedure is applied to the test set and the resulting idle cost is compared with the seed's idle cost. Test set's idle cost is less than the seed's, the seed is updated, and the new neighborhood is searched. This procedure continues until the maximum number of search iterations is reached. See Appendix A3 for a process diagram describing the second-stage search.

8. PROBLEM GENERATOR & PARAMETERIZATION

8.1 Problem Generator

To test the idle time reduction algorithm experiments will be run on randomly generated problems that are created using a problem generator. The problem generator creates a specified number of operations, each with associated durations and resource requirements. Each operations duration is drawn from a continuous random uniform distribution $D \sim U_{Cont.}(a, b)$ with interval width characterized by the coefficient of variation and expectation for the task duration $(b - a) = (12E[D]C[D])^{.5}$. Given an interval width, the values of a and b are easily computed $a = \max\{0, E[D] - .5(12E[D]C[D])^{.5}\}$, $b = E[D] + .5(12E[D]C[D])^{.5}$.

Similarly, the each operation's resource requirements are generated from m independent samples from a discrete random uniform distribution $R \sim U_{Discrete}(c, d)$ where the interval width is defined as above, with the lower-bound value a defined as the maximum of zero and floor of the expectation minus half the interval width, $a = \max\{0, \lfloor E[D] - .5(12E[R]C[R])^{.5} \rfloor\}$, and b defined as the ceiling of the expectation plus half the interval width $b = \lceil E[R] + .5(12E[R]C[R])^{.5}$.

During the creation of the operation, each operation is assigned a "TaskID" which is then used while building the successors for each operation. Once an operation is created the problem generator draws from a continuous standard uniform distribution, $U \sim (0,1)$, for all operations with a strictly greater TaskID. If this random variate associated with TaskID j for j > i is greater than an input parameter, probability of connectedness, then j is assigned as a successor of i. The maximum number of successors of any node is controlled by an additional problem generation parameter, to decrease over-connectedness that tends to create serial predecessor networks. This

successor generation procedure ensures that a topological ordering exists within the set of operations, resulting in a production feasible sequence.

After creating all operation level characteristics, the problem generator then finds the minimum resource capacity set, $R^{Min} = \max\{r_m(o_j^i)\}\,\forall o_j^i, \forall m$ that is needed to complete any operation. The schedule generation procedure is applied to this resource set to define an upper-bound on the makespan C_{Max}^{UB}. The non-resource scarce schedule, also known as the critical path, is found and used to define a lower-bound on the makespan C_{Max}^{LB}. The target tightness, t, is used to define the target makespan $T = L + t * (C_{Max}^{UB} - C_{Max}^{LB})$. The target-neighborhood which defined the interval of feasible makespans is calculated, $T_{LB} = T(1-\alpha), T_{UB} = T(1-\alpha)$. Table 1 shows a description of the different parameters used in schedule generation.

Table 1: Problem Generation Parameters

Parameter	**Description**
Number of Operations	The total number of operation created
Duration Mean	Expected value of the operations duration.
Coef. Variation (Duration)	The coefficient of variation for the duration $C_D = \frac{Var[D]}{E[D]}$
Number of Resources	The number of resource types in use, m
Resource Mean	Expected value of $r_m(o_j^i)$
Coef. Variation (Resource Req.)	Coefficient of variation for resource requirements, $C_R = \frac{Var\,[R]}{E[R]}$
Probability of Connectedness	The probability that Operation B is a successor to Operation B, B > A
Connected Max	The maximum number of successors for any operation
Target Tightness (t):	Defines the target makespan from Relative distance $T = C_{Max}^{LB} + t(C_{Max}^{UB} - C_{Max}^{LB})$
Alpha	Size of the target neighborhood

8.2 Search Delta Parameterization

A neighborhood search is defined by two main characteristics, the size of each individual neighborhood and the total number of iterations examined. To select a delta that provides good solutions an experimental study was performed to measure which delta minimizes the average of the scheduled idle time across ten replications. To more easily extend the results to a more general problem, the delta-levels tested were expressed as a percentage of the resource mean. The experimental problem parameterization is show in Table 2.

Table 2: Parameter Values for Delta Experiments

Parameter	**Value**
Number of Operations	50
Duration Mean	30
Coef. Variation (Duration)	.25
Number of Resources	4
Coef. Variation (Resource Req.)	.3
Probability of Connectedness	.5
Connected Max	4
Target Tightness:	.5
Alpha	.15

Problem instances with the above parameterization and resource means of ten, twenty, and thirty were created. Five experimental delta values, 10%, 25%, 50%, 75%, 90% of the experimental mean, were tested to measure which delta values minimize the idle time. Ten replications were run for each combination of delta value and resource mean. Selecting a delta value of 50% of the resource mean resulted yielded consistently good results in this experimental study and was chosen as the rule for delta parameterization for further experiments.

8.3 Search Iteration Parameterization

Prior to experimentation, a preliminary experiment was run to determine an appropriate number of iterations to perform in the second-stage idle time minimization search. The second stage search was run over multiple replications and allowed to run for eight-hundred iterations and the idle time and iteration number was recorded for any improvement. Figure 2 shows the relationship between idle time and iteration number for a fixed problem. Improvement in makespan is most significant in first 200 iterations with little to no improvement found in the subsequent six-hundred iterations and the maximum number of iterations evaluated in the second stage search is set at two-hundred iterations for subsequent experiments.

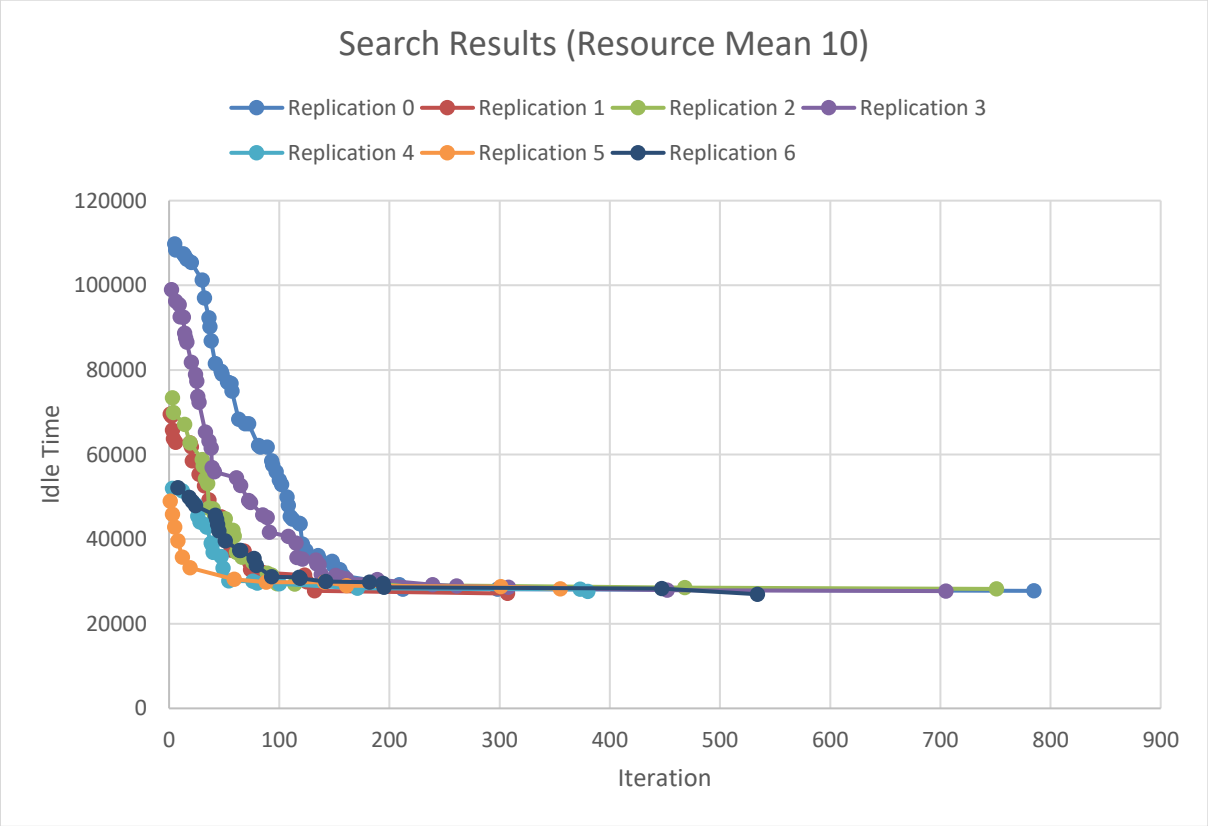


Figure 2: Iteration Parameterization Results

9. EXPERIMENTAL DESIGN

To measure the effectiveness of the algorithm two measures have been chosen. The first measure, solution improvement, represents the percent change between the initial feasible solution and the best solution found by the second-stage search. The second measure is the overall program runtime, which is defined by the total elapsed time between the start of the first-stage search and the end of the second-stage search.

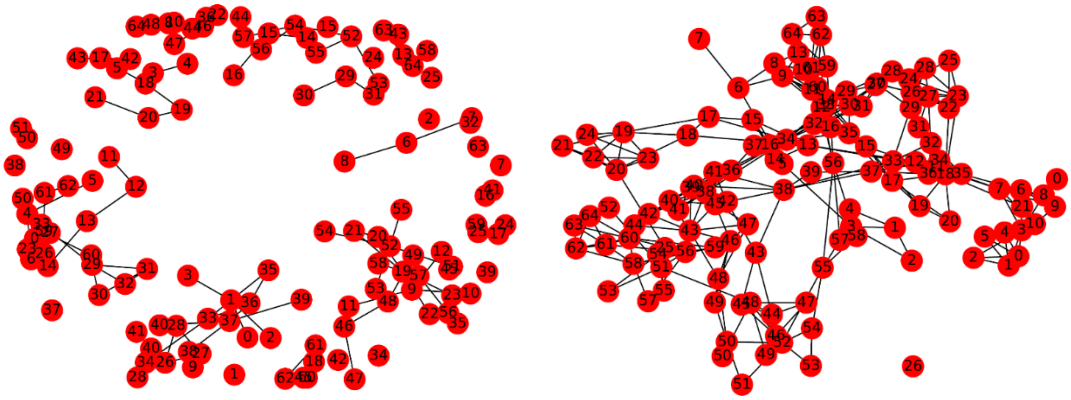


Figure 3: Network Representations of Connectedness Settings

Six different factors with two levels are analyzed, resulting in a 2^6 factorial design. The first two factors, Resource Variation and Resource Mean, represent the different combinations of resource types, with high and low resource means and high and low resource variations. The third factor, duration variation, represents the amount of variance in duration time for the

operations. The fifth factor, target tightness, represents problems with target makespans close to the lower-bound and upper-bound for the respective high and low levels.

The final two factors, Number of Operations and Connectedness, represent large and small sizes of the fundamental problem types, with the high connectedness representing jobs with stricter technological build precedencies, while low connectedness problems have more freedom in the build order. Figure 3 shows representative networks for the low (left) and high (right) connectedness settings and different factors and their corresponding aliases and levels are summarized in Table 3.

Table 3: Experimental Design Factors

Factors	**Alias**	**Low Level**	**High Level**
Resource Variation	A	.4	.75
Resource Mean	B	10	25
Duration Variation	C	.4	.75
Target Tightness	D	.25	.75
Number of Operations	E	50	150
Connectedness	F	{p_connected = .4, connect_max = 4}	{p_connected = .8, connect_max = 5}

Five replications were performed for each experimental scenario. In preliminary experimentation, five replications resulted in a standard error for mean estimates of approximately 6%, which was deemed an acceptable level of uncertainty. If the first-stage feasibility search failed to find a feasible solution on a problem instance a new randomized problem was generated using identical parameters and searched again until the first-stage solution found a feasible solution. This approach was chosen to ensure that the solution improvement resulting from the second-stage search could be measured and was only used for experimentation purposes.

Given that this work is intended to be performed on at a strategic level, if the first-stage search failed in a practical setting, additional search iterations should be performed to ensure that a feasible solution does not exist or cannot be practically found. If no solution exists, or no feasible solution can be practically found, the problem's target makespan may be unrealistic or infeasible, and a new target makespan should be found or a redefinition of the job content may be needed.

10. EXPERIMENT: SOLUTION IMPROVEMENT

During the analysis of results, only main effects and second order interaction effects were considered. Target Tightness (D), Connectedness (F), and Number of Operations (E), the two-factor interaction between Resource Mean and Connectedness (BF) and the two-factor interactions between Resource Mean and Number of Operations (BE) were determined to have a significant effect at 95% confidence. Target Tightness and Connectedness and the two-factor interaction between Resource Mean and Connectedness are all are significant with a p-value of 0.000, indicating that the effects are almost certainly significant. Other factors considered in experimentation did not have an impact on the solution improvement at 95% confidence. The significant factors are show in Figure 4.

10.1 Main Effects

The amount of improvement found by the secondary search is significantly higher on problems with low target tightness (D), that is problems where the targeted makespan is closer to the makespan upper-bound than problems where the target makespan is near the makespan lower bound. Across all experiments scenarios and replications, problems generated with low target tightness saw and average of 32.46% reduction in idle time when compared to the initial feasible solution. In in contrast high target tightness scenarios, the solution improvement was only 16.02%. This result is consistent with the expectation, as the feasible makespan range is closer to the makespan lower-bound, reducing the number of feasible schedules that can be examined.

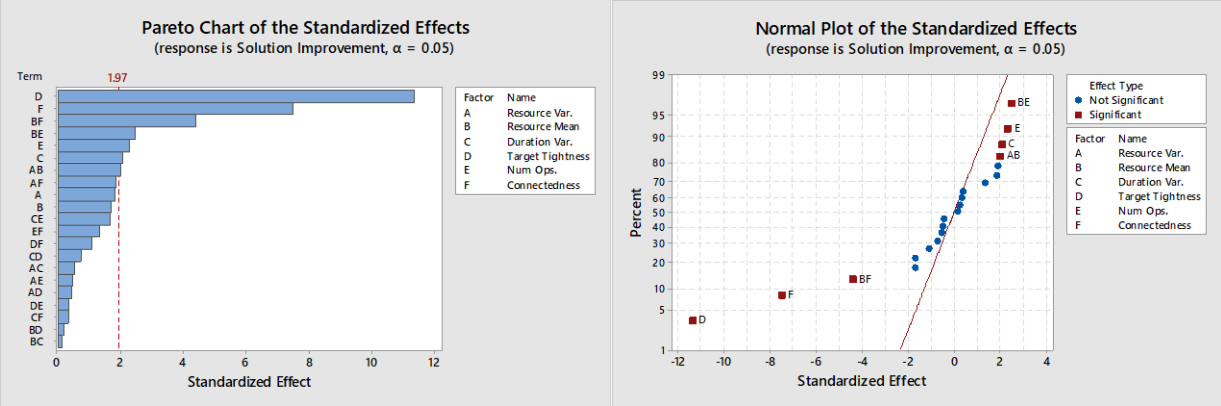


Figure 4: Pareto and Normal Plot for Solution Improvement

The connectedness settings of the network also have a significant impact on the amount of improvement found by the second stage search. Across all experiment scenarios and replications, the Low connectedness settings found an average of 29% improvement from the initial feasible solution. Scenarios with High connectedness settings found an improvement of 18.82% improvement. Schedules with higher connectedness settings have a smaller number of feasible schedules. Consider the extreme case, where the connectedness setting is 1 (e.g. an entirely serial build precedence consisting of n operations). There exists only one sequence which satisfies technological precedence. Contrast this extreme high connectedness scenario with the lowest possible connectedness of 0, where no operation has any technological predecessor, where n! feasible schedules exist. The connectedness setting has an enormous impact on the number of feasible sequences which are evaluated by the algorithm. Schedules with low connectedness settings have larger neighborhoods with more feasible solutions, resulting in better second-stage search results when compared to the high connectedness schedules.

The final significant main effect, Number of Operations, has a much smaller impact on the Solution Improvement than the previous two main effects. The second-stage search has a more significant improvement with the higher levels of number of operations, at a 26.14% improvement with 150 operations and a 22.56% improvement with 50 operations. Like the connectedness settings, the number of operations that are scheduled largely controls the number of feasible sequences, which has a similar effect on the second stage search.

The main effects plot, shown in Figure 5 , shows that the average solution improvement is greatest when the duration variation and number of operations are high, while the target tightness and connectedness is low. In summary the method will produce the best results in larger networks that have more variance on the operation duration, a target makespan that is closer to the upper-bound on makespan, and low connectedness. The experimental results are consistent with the expectation, as the algorithm has more freedom in build order and has more opportunities for scheduling contemporaneous operations in networks with lower connectedness settings.

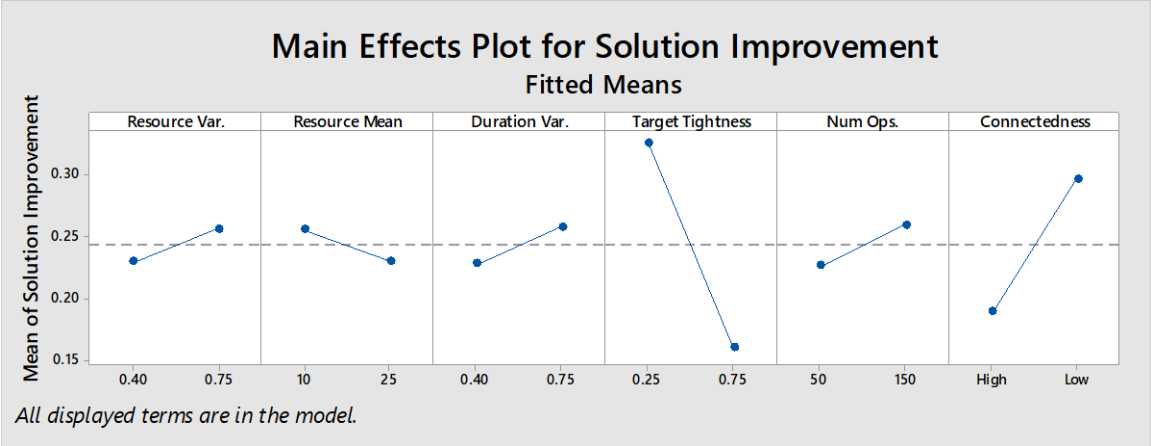


Figure 5: Main Effects for Solution Improvement

10.2 Two-Factor Interactions

Of the two-factor interactions, shown in Figure 6, three are significant at 95% Confidence Level. The interaction between Resource Mean and Connectedness (BF), with a p-value of 0.00, is almost certainly significant, while the interaction between Resource Mean and Number of Operations (BE), p-value 0.012, also has a strong degree of significance. The final interaction effect with significance, Resource Variability and Resource Mean (AB), p-value .043, is considered significant at an alpha level of .05, but it should be noted that the significance of the effect is much less pronounced than BF or BE.

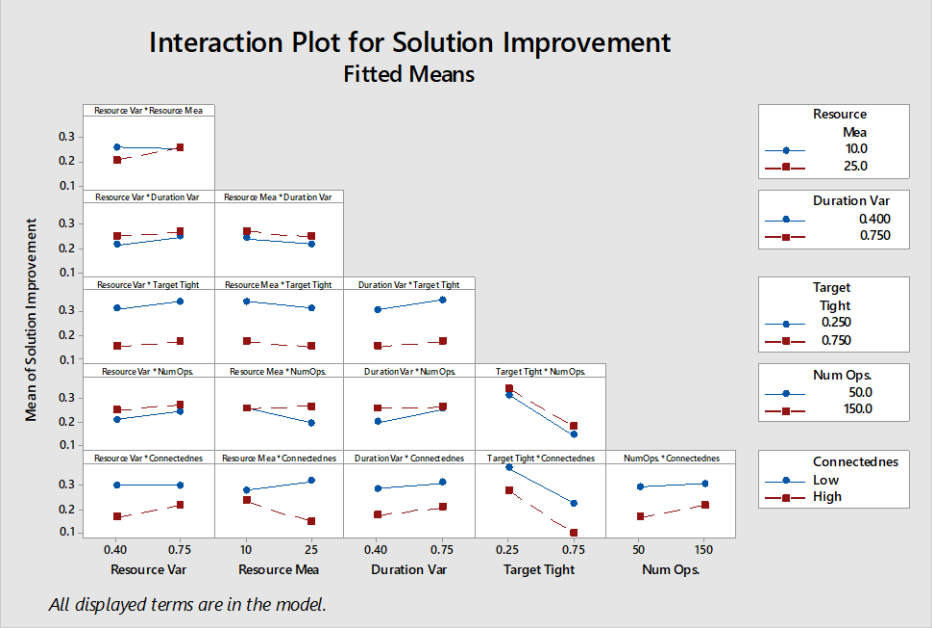


Figure 6: Two-Factor Interaction Effects for Solution Improvement

The BF interaction demonstrates that the method finds that connectedness has a major impact on the method performance when the resource means are large. Table 4 summarizes the average solution improvement results for all combinations of the BF Interaction.

Table 4: Interaction Effect between Resource Mean and Connectedness

Resource Mean	**Connectedness**	**Avg. Solution Improvement**
10	Low	27.71%
10	High	23.23%
25	Low	31.61%
25	High	14.41%

The connectedness setting has a major impact on the methods' efficacy in situations with high mean resource levels. Experimental results with low connectedness settings have an average solution improvement that is approximately 4% better than the high connectedness settings, while the connectedness difference with higher resource means is much more pronounced at approximately 15%. The interaction effect between connectedness and resource mean results from variance about the mean.

With low coefficient of variation on resource mean, the low Resource Mean scenario uniformly generates operations with resources requirements between six and fourteen units, while the high resource mean scenarios uniformly generates operations with resource requirements between nineteen and thirty units. The main effects results show idle time reduction is greatest in scenarios with low connectedness. Schedules with higher resource means have more variance in the operation's resource requirements when compared to schedules with low resource means but the same coefficient of variation. The higher resource requirement variance creates larger differences in objective value between sequences, which magnifies the main effect of the connectedness setting.

The interaction between Resource Mean and Number of Operations (BE) has a significant interaction as well. Like the connectedness settings, the impact the number of operations has on the average solution improvement is strongly impacted by resource mean. Table 5 shows the average solution improvement for the interaction between resource mean and number of operations.

Table 5: Interaction Effect between Resource Mean and Number of Operations

Resource Mean	**Number of Operations**	**Avg Solution Improvement**
10	50	25.58%
10	150	25.36%
25	50	19.55%
25	150	26.47%

The number of operations has almost no impact on the average solution improvement when the resource mean is at the low setting, while the solution improvement is approximately 7% better on average for 150 operations while resource mean is at the high setting. The interaction between Number of Operations and Resource Mean has the same underlying cause as the interaction between Connectedness and Resource Mean. Like the interaction effect between connectedness settings and resource mean, the higher operation resource requirement variance increases the difference in objective value between sequences, which increases the impact of the Number of Operations main effect.

10.3 Experimental Conclusions

The results of experimentation show that the if the second-stage search methodology was applied to a procedure that only finds a feasible solution within the target window there exists a significant opportunity for improvement in total idle cost while maintaining an acceptable makespan. The grand average of across all scenarios shows an idle time reduction of 24.24%. In the worst-case experimental scenarios, the procedure shows promising results and would result in significant savings of manpower and machine time which can have a significant impact on production cost.

11. EXPERIMENT: RUNTIME

The following section presents the results from the runtime experimentation. This experiment was conducted in parallel with the solution improvement results to characterize the run time of the program and identify any factors other than number of operations and connectedness that have a significant impact on the program run time.

11.1 Main Effects

As expected, the Number of Operations (E) and the Connectedness (F) dominate the significant main effects for the algorithms' run time. The most significant two-factor interactions are the interaction between Target Tightness and Connectedness (BF), and the interaction effect between Number of Operations and Connectedness (EF). Both main effects are clearly significant, with associated p-values of 0.00, indicating their significance is almost a certainty. There are a number of other effects that are significant at a 95% confidence level, but the experimental analysis has been focused on the most significant effects, due to their prominence. Figure 7 shows the Pareto Chart and Normal Plot for the Standardized Effects.

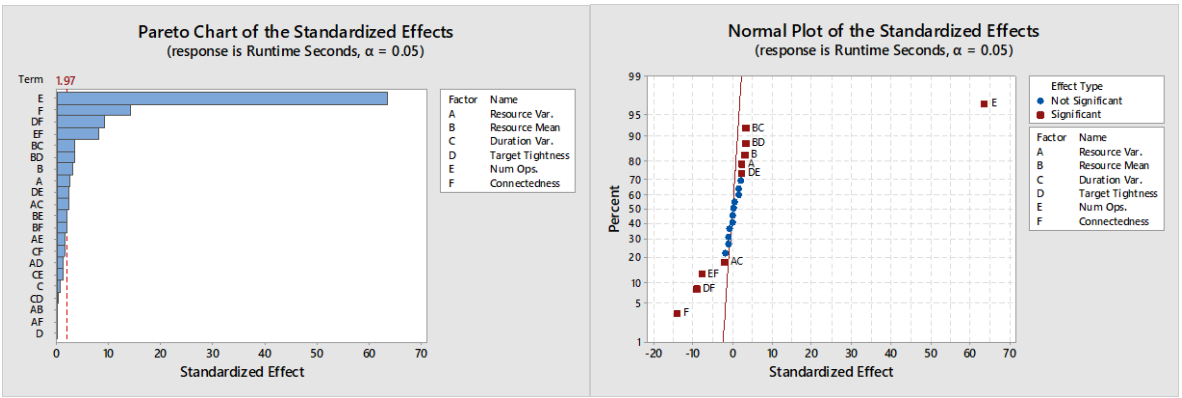


Figure 7: Pareto and Normal Plot for Runtime

The previous section demonstrates which experimental factors have a significant impact on the algorithms' runtime, while this section characterizes the impact each factor has. Figure 8 shows the program runtime (seconds), for each of the main effects and level combinations. The results for the main effects analysis are consistent with the significance analysis, it's clear that changing the levels of two parameters, namely the Number of Operations (E) and Connectedness (F) have the largest impact on the programs runtime.

It is intuitive and expected that the program runtime is driven by the number of activties that are scheduled. Additionally, program runtime is longer for networks with low connectiveity, since networks with low connectivity generally have more elements in the set of schedulable activities and the algorithm spends more time evaluating which operation to schedule when compared to networks with high connectedness. In the extreme case, where the network is entirely serial, there is only one technologically feasible operation per iteration of the algorithm, and little to no computational effort is expended comparing competing schedulable operations.

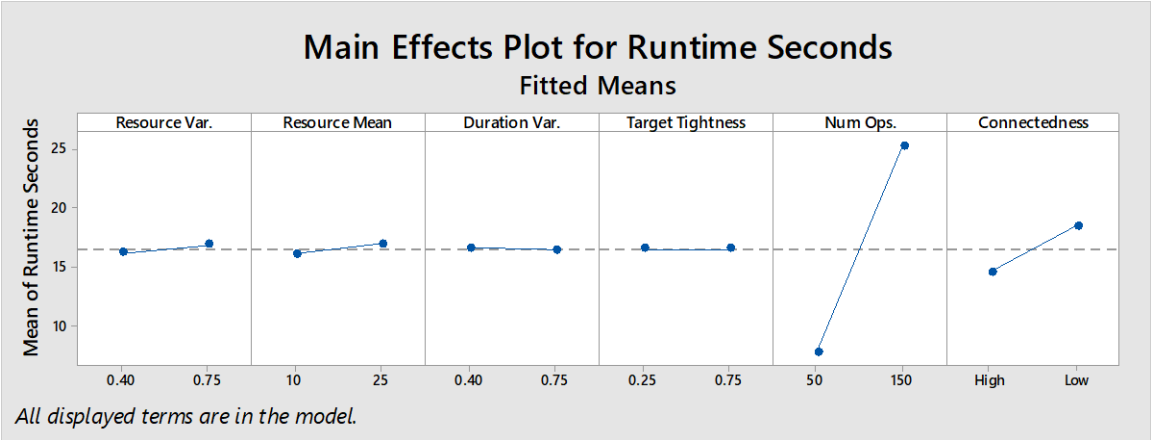


Figure 8: Main Effects for Runtime

11.2 Two-Factor Interactions

Two different two-factor interaction effects are considered in this work. The interactions between Target Tightness and Connectedness (BF), and the interaction effect between Number of Operations and Connectedness (EF). These effects, shown in Figure 9, have a less of on the program runtime when considered alongside the dominant main effects.

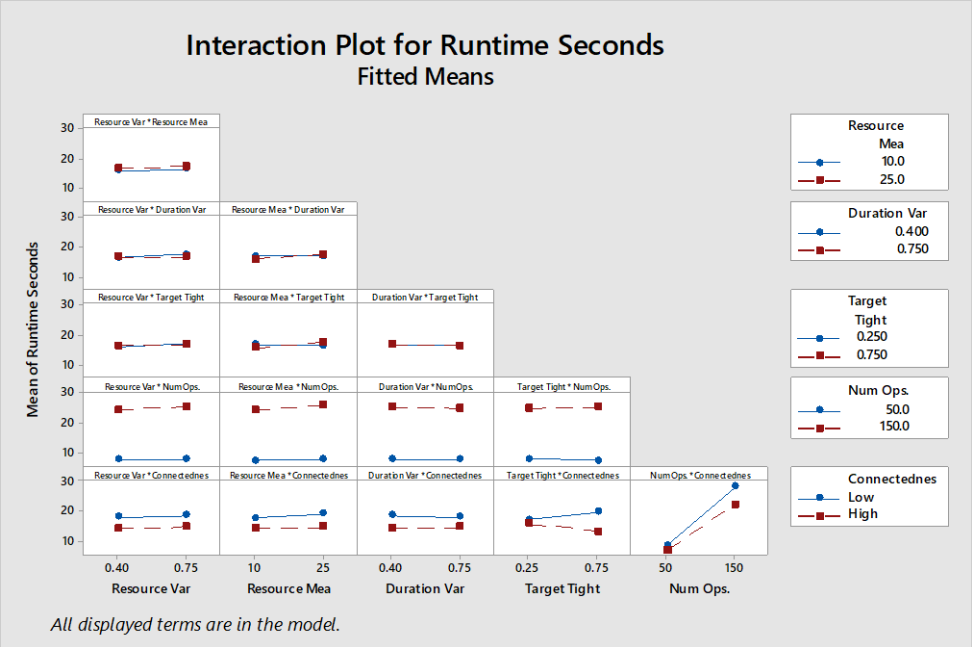


Figure 9: Two-Factor Interaction Effects for Runtime

The most significant two factor interaction is between Target Tightness and Connectedness (EF). Table 6 demonstrates that the network Connectedness has a much more pronounced impact on the program runtime while the Target Tightness is high than when the Target Tightness is low. There is approximately a 2 second difference in runtime between the

Low and High Connectedness settings while the Target Tightness is low, and a 6 six second difference in runtime while the Target Tightness is high. This difference is expected to be more pronounced in problems with larger number of operations.

Table 6: Two-Factor Interaction Results (BE)

Target Tightness	**Connectedness**	**Runtime (Seconds)**
.25	Low	17.26
.25	High	15.90
.75	Low	19.75
.75	High	13.39

The second two-factor interaction between Target Tightness and Connectedness has a less pronounced, yet still significant impact on the program run time. While the Target Tightness is high, the impact of the connectedness setting has a much larger impact on the program runtime. Programs with Low Connectedness settings have longer run times on networks with higher number of operations than expected from the main effects alone. This difference, demonstrated in Table 7, shows the algorithm will be less effective, in terms of runtime, for problem instances with high target tightness and low connectedness settings.

Table 7: Two-Factor Interaction Results (EF)

Target Tightness	Connectedness	Runtime (Seconds)
.25	Low	17.26
.25	High	15.90
.75	Low	19.75
.75	High	13.39

11.3 Additional Testing

To test the methods effectiveness on larger problem instances, an additional experiment was conducted to measure how the number of operations influences program runtime. The results show in Figure 10 are promising, demonstrating that with appropriate search parameterization large problem instances can be quickly solved. The program runtime was tested on low connectedness problem sets from 100 operations up to 1200 operations. The runtime testing results demonstrate the algorithm runtime estimate is well fit by a second order polynomial. The results show that the algorithm can be effectively used on large problem sets, even with the pessimistic connectedness settings.

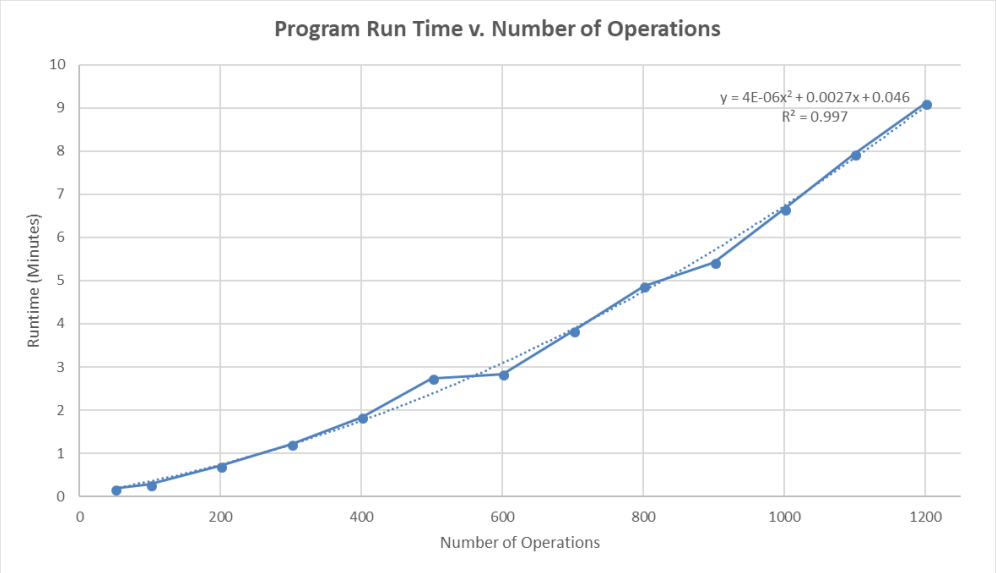


Figure 10: Algorithm Runtime Results

12. FUTURE WORK AND LIMITATIONS

12.1 Future Work

The Lean Resource Scheduling algorithm proposed in this thesis has demonstrated that there may be opportunities for significant savings in resource usage if applied to appropriate problems. The first major opportunity for future work is testing the algorithms performance in a case study with real data, instead of randomly generated data sets.

This work only uses the schedule generation procedure described in section 7b to determine the resource utilization, idle time, and makespan of a schedule given a fixed resource level. Since the algorithm evaluates the schedule performance of a significant number of resource leave the schedule generation procedure only utilizes a construction heuristic and no local search phase. In general, a construction heuristic is not guaranteed or expected to find an optimal solution and there is an opportunity further reduce a schedules idle time by adding a local search phase once the proposed algorithm terminates.

Extending the model to assign different resource levels to specific jobs would improve the model fidelity in many situations. Currently, resources aren't assigned to a specific job, and are able to work on any operation of any job, assuming they are available. In many practical environments resources are assigned to specific areas or teams and extending the modeling to account for individual team assignments will significantly increase the accuracy of the underlying model.

12.2 Limitations

One major theoretical and practical limitation of this work is all of the operation parameters, such as duration, resource requirements, and are assumed to be deterministic. This assumption limits a practical implantation of the methodology, as almost all real-world scheduling environments involve some degree of stochasticity. Additionally, a practical implementation of this methodology would require an extremely agile staffing plan, where employees are routines switching between different job, with a high degree of cross-training.

13. CONCLUSIONS

The runtime experiment results show methodology can be applied to large problem sizes while still maintaining reasonable runtimes. The algorithms' runtime performance was on problem sets with up to 1,200 individual operations and the runtime results are promising even for large problems. The relatively short runtimes, coupled with promising idle time reduction result, show implementation of a second-stage idle time minimizing search could produce significant savings in large operations.

REFERENCES

[1] Simon French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop (Mathematics and its Applications).* Hemel Hempstead: Ellis Horwood Ltd, 1982.

[2] Leon, Balakrishnan. *Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling*. Operations-Research Spektrum, 17:173. Springler-Verlag, 1995.

[3] Leon, Balakrishnan. *Adaptable problem-space-based search methods for flexible flow line scheduling*. IIE Transactions. 29:115, 1997.

[4] Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations.* Nijhoff: 1976.

[5] Ku, Bec. *Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis*. Computers & Operations Research, 73:165. 2016

[6] Panwalkar, Iskander. *A Survey of Scheduling Rules*. Operations Research, 25:45. 1997

[7] Womak, Jones. *Lean Thinking : Banish Waste and Create Wealth in Your Corporation*. Simon & Schuster, 1996

[8] Melton. *The Benefit of Lean Manufacturing: What Lean Thinking has to Offer The Process Industries.* Chemical Engineering Research and Design. 83:622, 2005

[9] Crute, Ward, Brown, Graves. *Implementing Lean in aerospace – challenging the assumptions and understanding the challenges.* Technovation. 12:917, 2003

[10] Jina, Bhattacharya, Walton. *Applying lean principles for high product variety and low volumes: Some issues and propositions*. Logistics Information Management. 10:5, 1997

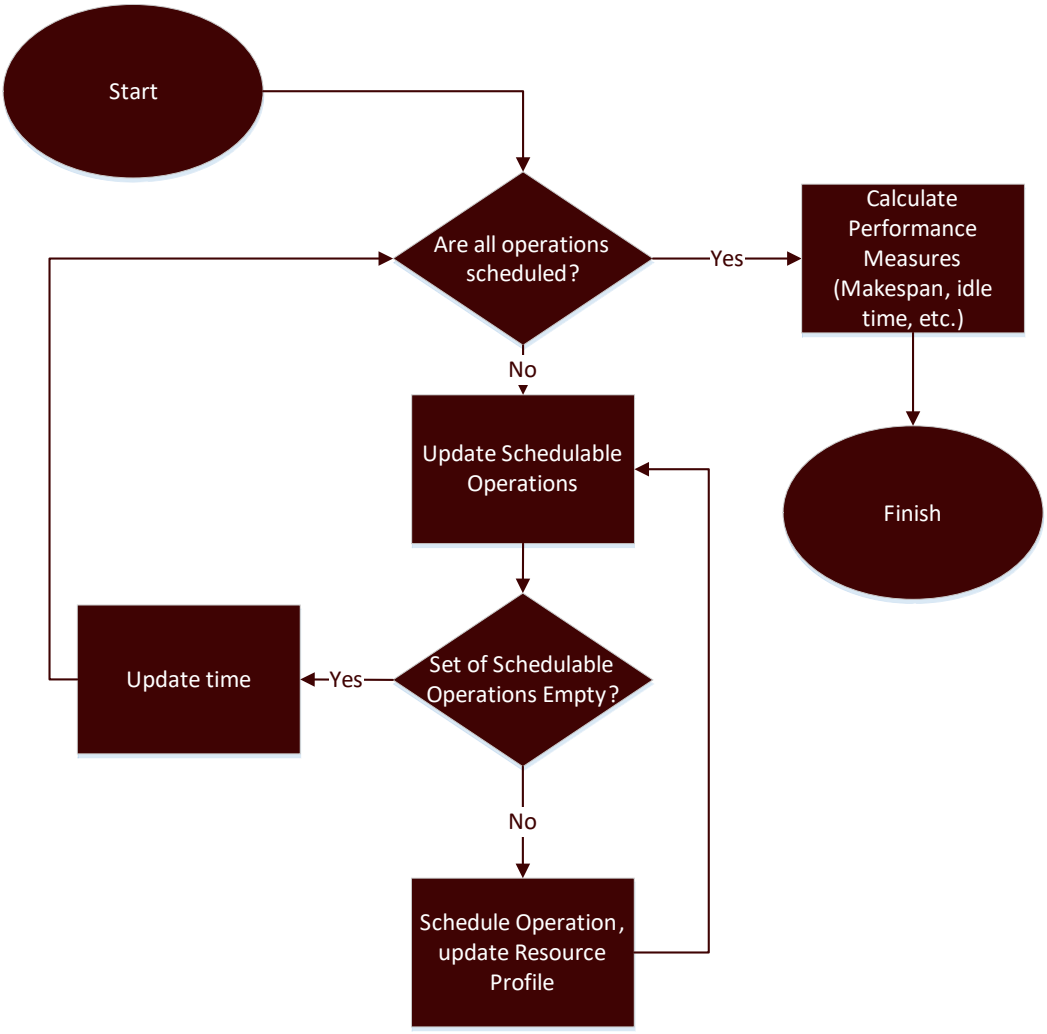
[11] Ali, Deif. *Dynamic Lean Assessment for Takt Time Implementation*. Variety Management in Manufacturing. Procedia CIRP. 17:577, 2014

[12] Lourenço, Martin, Stützle. (2003) *Iterated Local Search*. In: Glover, Kochenberger, (eds) *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer, 2003

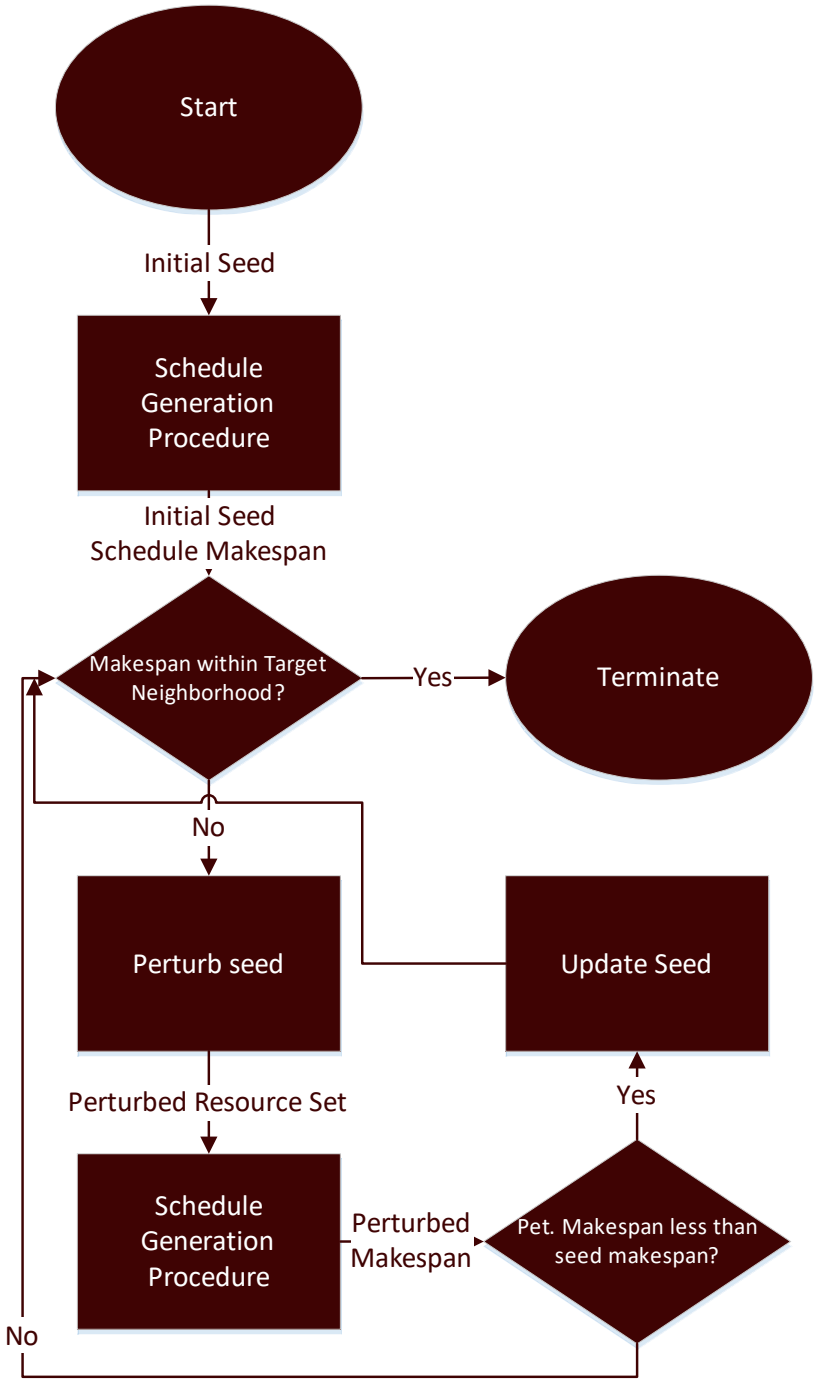
APPENDIX A

PROCESS FLOW DIAGRAMS

Schedule Generation Process



First Stage Search Process



Second Stage Search Process