SURROGATE SEARCH: A SIMULATION OPTIMIZATION METHODOLOGY FOR LARGE-SCALE SYSTEMS

by

Jyh-Pang Lai

BS in Industrial Engineering, Tunghai University, 2000

MS in Industrial Engineering, University of Pittsburgh, 2003

Submitted to the Graduate Faculty of

the School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This dissertation was presented

by

Jyh-Pang Lai

It was defended on

May 1, 2006

and approved by

Dissertation Director: Larry J. Shuman, Professor, Industrial Engineering Department

Dissertation Co-Director: Bopaya Bidanda, Professor, Industrial Engineering Department

Matthew D. Bailey, Assistant Professor, Industrial Engineering Department

Calvin C. Lai, PhD, FedEx Ground

Bryan Norman, Associate Professor, Industrial Engineering Department

Randall P. Sadowski, PhD, Rockwell Software

SURROGATE SEARCH: A SIMULATION OPTIMIZATION METHODOLOGY FOR LARGE-SCALE SYSTEMS

Jyh-Pang Lai, PhD

University of Pittsburgh, 2006

For certain settings in which system performance cannot be evaluated by analytical methods, simulation models are widely utilized. This is especially for complex systems. To try to optimize these models, simulation optimization techniques have been developed. These attempt to identify the system designs and parameters that result in (near) optimal system performance. Although more realistic results can be provided by simulation, the computational time for simulator execution, and consequently, simulation optimization may be very long. Hence, the major challenge in determining improved system designs by incorporating simulation and search methodologies is to develop more efficient simulation optimization heuristics or algorithms.

This dissertation develops a new approach, Surrogate Search, to determine near optimal system designs for large-scale simulation problems that contain combinatorial decision variables. First, surrogate objective functions are identified by analyzing simulation results to observe system behavior. Multiple linear regression is utilized to examine simulation results and construct surrogate objective functions. The identified surrogate objective functions, which can be quickly executed, are then utilized as simulator replacements in the search methodologies. For multiple problems containing different settings of the same simulation model, only one surrogate objective function needs to be identified. The development of surrogate objective functions benefits the optimization process by reducing the number of simulation iterations.

Surrogate Search approaches are developed for two combinatorial problems, operator assignment and task sequencing, using a large-scale sortation system simulation model. The

experimental results demonstrate that Surrogate Search can be applied to such large-scale simulation problems and outperform recognized simulation optimization methodology, Scatter Search (SS). This dissertation provides a systematic methodology to perform simulation optimization for complex operations research problems and contributes to the simulation optimization field.

Keywords: Automatic material handling system, heuristics, simulation optimization, Surrogate Search.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

1.1 PROBLEM STATEMENT

The difficulties of determining near optimal system designs using simulation approaches include long computational times and lack of effective methods to search the solution space. Because simulation captures the stochastic nature of the system, the performance of one scenario is typically unknown until it is simulated; further, multiple simulation replications are required for each simulated scenario. Although the goal of optimizing simulation results is to identify the optimal system design of the simulation model, obtaining the optimal design may require evaluating a large number or all system designs through simulation. In practice, the optimal system designs cannot be obtained for a large portion of simulation problems due to the long CPU time required to evaluate alternative system designs. In this dissertation, simulation optimization is defined as those methodologies that determine near optimal solutions of simulation models by evaluating a small number of solutions obtained through simulation.

Meta modeling is one simulation optimization method that utilizes mathematical functions to represent the relationship between independent and dependent variables of a simulation. By optimizing the simulation meta model, near optimal solutions can be identified in a relatively short amount of time without being restricted by unrealistic assumptions. For problems in which the relationship between independent and dependent variables cannot be determined, methodologies to identify near optimal solutions include mathematical programming

or meta heuristics that utilize simulation models as objective functions. In general, these methodologies require more simulation runs than simulation meta modeling to determine near optimal solutions. Hence, the major issue for simulation optimization using mathematical programming or meta heuristics is that a large number of simulation runs is required, which may result in long computation times.

For complex optimization problems in which objective functions cannot be identified, it is necessary to utilize a simulation model as the objective function. However, the number of solutions that can be examined by simulation models may be relatively small due to the long CPU time required for the simulation runs. This dissertation develops a new methodology, Surrogate Search, for simulation optimization. Surrogate Search utilizes an objective function identified by analyzing simulation results. Prior to simulation execution, a large number of system designs are evaluated by the surrogate objective function. By utilizing the surrogate objective function approach, solutions leading to statistically significant improvements can be determined in a relatively small number of iterations when compared to other simulation optimization techniques.

In this study, we investigate the methodology of utilizing simulation results to improve system settings. For the case study, a detailed simulation model for a complex Automatic Material Handling System (AMHS) based sortation system is built, a method for optimization is proposed that uses a surrogate objective function approach, and several problems are solved. The sortation system is one of the largest such systems in the U.S. and is owned by one of the major companies in the distribution industry. This system involves both continuous and combinatorial optimization problems; the continuous problems include system parameter settings and operating policies; the combinatorial problems include assignment and scheduling problems.

In summary, this dissertation research addresses the problem of utilizing heuristics to identify near optimal simulation results. The identified surrogate objective functions can effectively direct search procedures to identify improved solutions.

1.2 CHALLENGES OF SIMULATION OPTIMIZATION

For large-scale systems, it is difficult to evaluate performance by deterministic approaches, which do not have the ability to capture stochastic factors and the interactions among factors. As a result, modelers have turned to simulation. However, to accurately simulate complex systems, detailed operations must be programmed into the models. Simulation is widely used to analyze complex systems, since it's one of the methods that generate the most realistic results. Although more realistic results can be generated by complex simulation models, the CPU time needed to execute a large simulation is typically quite long. In contrast, Linear Programming (LP) and Mixed Integer Programming (MIP) can evaluate a large number of solutions within a short amount of time. For problems that contain a relatively small number of variables, LP and MIP can identify the optimal solutions relatively fast. The weakness of LP and MIP is that their formulations cannot represent the interactions among factors due to the resultant assumptions, which could lead to infeasible solutions for implementation. In addition, MIP may require substantially long CPU time to solve certain large-scale problems.

To capture the stochastic nature of systems, Stochastic Programming (SP) is designed to incorporate uncertainties in the problem formulation. Although SP presents probabilities of events using constraints and optimizes the expected objective values, the objective functions and relationships among variables still need to be determined. For large-scale systems, the objective

functions and relationships among variables often cannot be observed from the real system and may require utilizing simulation models to predict interactions.

Simulation models provide the functionality to replicate the real system and generate more information than simply a measure of the objective value. The information includes resource utilization, system breakdowns, process time, waiting time, and queue length. Additional information can be gathered by simulating different values for independent variables (e.g., system parameters and designs). An advantage of utilizing simulation models is the relatively low cost to change system designs. Adjusting system parameters and designs for a validated simulation model can be completed in a short amount of time without the cost of changing the real system and the risk of system failure. In addition, the required computational time is typically much shorter than the operating time for the real system. Hence, it requires less time and lower cost to observe system behavior utilizing simulation compared to changing the real system.

A major issue with utilizing simulation models to evaluate system performance is that they do not have the ability to improve system designs; rather they simply evaluate a given scenario. Consequently, developing a methodology to determine improved system designs using simulation is one of the most popular topics among simulation researchers, and a number of methodologies have been proposed.

The major challenge in determining improved system designs through simulation is the large number of iterations required for the optimization process. As noted, for complex simulation models, the computational time for execution may be very long. Consider a simulation model that requires one hour of CPU time for each execution. If there are three factors that each have ten possible values, the total number of possible settings would require

1,000 hours to determine the optimal system setting ($10 \times 10 \times 10 \times 1$ hour = 1000) if all possibilities are evaluated. This doesn't consider the necessary replications which could add another order of magnitude to the CPU time. In addition, the simulation results are only validated to specified simulation model inputs, which may be changed regularly. By the time that large-scale simulation models are optimized, the associated system inputs may no longer exist and the optimal system settings may be impractical. Consequently, large-scale simulation models that require long CPU times frequently cannot be optimized unless there are methods that can identify improved solutions relatively quickly.

Although simulation models generate a large amount of data, typical simulation optimization techniques only record and utilize the predefined result as objective values. For large-scale simulation models, detailed operations are programmed into the models in order to generate realistic results. When executing the simulation model, measures for all these operations can be obtained. However, the existing techniques do not fully utilize these simulation results. In contrast, these simulation results can be utilized by Surrogate Search to determine surrogate objective functions. Although approaches that utilize alternative formulas (e.g., linear relaxation for Integer Programming [1] and surrogate state for Stochastic Integer Programming [2]) have been developed by simplifying the problem structure, this dissertation focuses on observing system behavior and does not consider those approaches. The process of identifying surrogate objective functions may require a long time. If simulation models can be executed relatively fast, near optimal system designs can be identified by other existing methodologies without investing the large amount of time needed up front for surrogate search. Consequently, there is no need to analyze simulation results for these simulation problems since

the time to identify surrogate objective functions can be longer than solving these problems by other existing methodologies.

1.3 OVERVIEW OF THE DISSERTATION

Chapter 2 presents a literature review that includes the discussion of deterministic approaches, simulation modeling techniques, and simulation optimization methodologies. The advantages and disadvantages of deterministic and simulation optimization methodologies are presented.

In Chapter 3, the sortation system for the case study is introduced and the modeling approaches are presented. In Chapter 4, regression meta modeling is applied to two problems. A major cause of regression meta modeling failures is presented. These regression meta modeling failures provided the motivation to develop search methodologies to effectively identify improved system designs using simulation models.

In Chapter 5, Surrogate Search, a systematic approach that utilizes simulation results and then incorporates those results into current search methodologies is presented. Two types of problems that contain combinatorial decision variables in the sortation system simulation model are utilized as test problems. In Chapter 6, Surrogate Search is applied to solve assignment problems. In Chapter 7, Surrogate Search is applied to solve a task sequencing problem. These two chapters address the benefits of utilizing a Surrogate Search approach. Chapter 8 contains the summary and conclusion of this dissertation. Contributions and future research directions are also discussed.

2.0 LITERATURE REVIEW

In this chapter, literature related to simulation optimization and Automatic Material Handling System (AMHS) are discussed. Section 2.1 describes deterministic optimization methodologies, which includes mathematical programming and heuristics. In section 2.2, simulation modeling approaches for large-scale manufacturing and distribution systems are presented. These include the subsystem modeling and the flexibility to modify operating policies. In section 2.3, simulation optimization methodologies that utilize multiple linear regression, Artificial Neural Networks (ANN), and heuristics are discussed. Section 2.4 summarizes common meta heuristics that are applied to solve simulation problems.

2.1 DETERMINISTIC OPTIMIZATION METHODOLOGIES

Deterministic optimization methodologies, including optimization and heuristics, can be used to solve a variety of problems that contain only deterministic parameters. Linear Programming (LP) is typically utilized to determine optimal solutions of problems that are formulated using continuous variables and linear functions. Mixed Integer Programming (MIP) is a methodology developed to solve problems that contain both discrete variables and linear functions. For problems that contain nonlinear formulations or cannot be effectively solved by MIP due to their size, heuristics are commonly utilized to identify near optimal solutions.

For deterministic problems, a large number of solutions can be evaluated rapidly based on several assumptions. These assumptions are the additivity assumption, divisibility assumption, and certainty assumption [3]. The additivity assumption requires the objective function to be a linear function. The divisibility assumption allows fractional values for decision variables. The certainty assumption uses exact coefficients and parameters in the constraints and objective function. LP is based on these three assumptions. MIP is based on the additivity and certainty assumptions. For heuristics that are designed to solve deterministic problems, only the certainty assumption needs to be applied. This section describes deterministic optimization applications; more details and the theories underlying these methodologies can be found in [4] and [1].

The strength of LP and MIP is that they guarantee to find an optimal solution for the problem of interest. Although the certainty assumption enables deterministic optimization methodologies to solve problems quickly, large-scale combinatorial problems still pose a major challenge for optimization techniques, because a large number of feasible solutions need to be evaluated in order to find the optimal solution.

The major advantage of applying heuristics to nonlinear and combinatorial problems is that relative good solutions can be found when only a fractional portion of all of the solutions are evaluated. Consequently, the required CPU time to execute heuristics is relatively short in comparison to executing optimization techniques on the same problems. Although there are proofs that certain heuristics can identify global optimum by evaluating a finite number of solutions, the required long computational time is the major constraint to identify the global optimum in practice. In this dissertation, heuristics refer to search methodologies that can identify near optimal solutions by evaluating a small portion of solutions.

Holmberg and Hellstrand [5] developed a heuristic method based on Lagrangean relaxation to find the exact solutions for network problems without capacity constraints. However, their one origin and destination assumption limits the types of problems that can be solved by this method. Beschorner and Gluer [6] modeled an AMHS as a maximum flow problem. The upper and lower limits of the network problem are given by the AMHS's capacity. This technique provides a quick method to examine the AMHS's design and to initialize the AMHS's input parameters. The maximum flow model can react to the AMHS's failures and generate new optimal designs. Luna and Mahey [7] presented an approach for telecommunications and computer network expansion problems. The non-linear cost function in their model provides a better cost estimation than linear cost functions.

As noted, deterministic optimization methodologies are capable of evaluating and identifying improved solutions quickly. However, parameters (coefficients) for the models sometimes cannot be easily estimated. Simulation is one of the techniques to generate more realistic parameters for deterministic models. For example, Watson and Ter-gazarian [8] optimized a renewable electrical power system by combining simulation and deterministic planning models. The power plant operations and population's demand for electrical power were simulated on an hour-by-hour basis. The model was validated by comparing results to observed data. The deterministic model divides the power supply system into a network with demand and capacity constraints based on the simulation results. The optimal fuel cost is generated by solving the deterministic model. Bai, Bobba, and Hajj [9] optimized a power distribution circuit by determining the optimal locations for decoupling capacitances. The maximum voltage drop (input parameter) in the complex power distribution network was estimated using steady-state simulation. Accurate estimates of parameters for the deterministic model could be predicted by

simulating the power distribution circuit. In a similar manner, the failure rate function for an AMHS can be estimated by simulating the number of packages in each location and then fitting the results to a formula using multiple linear regression. This will be done in the sortation system simulation model described in Chapter 3.

Heuristics are designed to solve problems that cannot be easily solved by traditional optimization techniques. Most heuristics utilize certain properties of the problem structure in order to identify solutions, but their performance is limited to specified problem types. Heuristics that are designed to address a wide range of problem types are called meta heuristics. Meta heuristics are generally utilized to determine approximate solutions to complex problems since they are not limited by problem structure. Common meta heuristics include Simulated Annealing (SA), Genetic Algorithm (GA), Evolutionary Strategies (ES), Tabu Search (TS), and Scatter Search (SS).

For a variety of problem types, meta heuristics can identify improved solutions in a short amount of time. However, when only a fractional portion of the solutions are evaluated, there is no guarantee of solution quality. Further, performance varies among different meta heuristics. Lee compared the performance of GA, SA, and TS on multi-machine two-stage scheduling problems [10]. His results found that the TS could determine better solutions with less computational time then GA and SA.

Teghem, Tuyttens, and Ulungu [11] developed a SA based method to solve multi-objective combinatorial problems by assigning different weights to each objective function. Yip and Pao developed guided evolutionary SA that solves combinatorial problems [12]. Their algorithm utilizes the temperature mechanism to control SA population selection in an

Evolutionary Algorithm by the improvement of solution values. Their approach had better performance than SA for solving traveling salesman problems.

Chakraborty and Chakraborty solved network problems with budget constraints by GA and SA [13]. Their empirical results show that SA outperformed GA. Ohkura et al. developed a GA approach to solve newspaper advertising problems [14]. A concern with their study is the assumption that the proportion of people who notice the advertisement can be accurately formulated. Chen developed a SA approach to model turning operations and determine the optimal parameter setting [15]. Although the deterministic assumption for deterministic optimization methodologies may be unrealistic in many cases, Computer Numerical Control (CNC) machines can provide deterministic processing times. With the non-linear objective function and combinatorial constraints, Scatter Search (SS) was utilized to solve this type of problem.

2.2 SIMULATION MODELING

Simulation models are built to generate realistic results relative to actual systems. System failures, detailed operations, and uncertainties can be built into simulation models without resorting to unrealistic assumptions. In general, the more detailed models require relatively longer CPU time, but provide more realistic results. Consequently, the number of solutions that can be evaluated by simulation is reduced, and the time to determine an improved solution can be extremely long. The disadvantages of developing detailed simulation models are that modeler effort will be more time-intensive and the long computational time will reduce the number of solutions that can be examined by simulation models. In this section, techniques to develop

large-scale simulation models and their applications are discussed. A more comprehensive description of the theories and techniques related to simulation are presented in [16] and [17].

Simulation models are utilized because the complexity of systems cannot be described by formulas. For large-scale systems, techniques to develop flexible simulation modules are critical. By developing system components as modules, the simulation models of similar systems can be developed using the same modules with simple modifications. Hofmann summarized the major benefits of component-based simulation models [18]:

- An increase in the range of manageable complexity in system design and analysis.
- An increase in the model reliability by using validated simulation components.
- An accelerated model development process.
- Improved model maintainability.

Koopman discussed and categorized modeling decomposition approaches into strategies of structure, behavior, and goals [19]. The decomposition rules will help for model debugging and modification in large-scale system modeling.

Applications of utilizing flexible simulation modules include manufacturing facilities, military operations, and electrical systems [20]. For example, LeBaron and Hendrickson evaluated different scheduling policies in wafer fabrication facilities by utilizing a simulation module called "cluster tool," which simulates several machining processes and AMHS [21].

Cardarelli and Pelagagge [22] determined the relationship between delay time and utilization of a material handling and storage system. The system factors, and their interactions were analyzed by simulating the system. However, Cardarelli and Pelagagge used average delay time and utilization for the measures in this study. It would have been helpful if the variance were provided to show that the system had a stable process with small variance. Weigl [23]

measured the utilization of individual elements of a large-scale brewery distribution system by simulation. Here, the detailed simulation of the AMHS provides accurate results for alternative system designs. However, the author claimed that the simulation results, compared to the real output, were "100 percent accurate" without providing the necessary statistical analyses to justify this. Smith and Medeiros [24] simulated different control strategies for manufacturing systems. By separating the system into physical and control subsystems, the flexibility of the control strategies can be quickly and easily changed. However, this paper does not provide an actual case study. It would have been helpful if the authors had used their technique in modeling complex system.

Nazzal and Bodner [25] presented a simulation-based framework for AMHS facility design. In this paper, the AMHS modeling is completed by specifying the physical and logical AMHS subsystems. The modeling technique gives the flexibility to build alternative AMHS models by modifying the logical subsystem. The modeling technique to simulate the alternatives is fairly easy to implement. However, this paper did not provide methods to optimize the AMHS. Meinert, Taylor, and English [26] presented a modular simulation approach for the evaluation of an AMHS. A high-level modular simulator of alternative AMHS designs was completed by identifying possible types of material handling hardware, buffering strategies, and demand profiles of the AMHS.

Harit and Taylor [27] developed a simulation model of the controllers of large-scale material handling system. To keep the systems point of view, modeling was completed without simplifying assumptions or decomposing into sub-problems. Wang, Fuh, and Yee [28] presented an approach for AMHS based manufacturing system design. Their approach incorporates simulation and queueing network models. The major strength of their approach is

the queueing network model can eliminate potentially bad solutions from being evaluated by simulation. Although the experimental results showed improved system designs can be identified by their approach, the paper did indicate that statistical analysis were performed to determine differences among alternative solutions. Wu and Caves [29] simulated the rotation of aircraft in multiple airports. The detailed aircraft ground operations are modeled using Markovian concepts. In this paper, the simulated aircraft rotation problem is analogous to the AMHS simulation due to the network structure of both problems.

To utilize simulation as real time decision tool, Yaun, Carothers, Adali, and Spooner [30] developed a controlling logic that ran a simulator in parallel with real system to dynamically change the cache size for users. The simulator utilized the real-time data as the simulation input. The authors pointed out that it is critical to have fast simulation models when they are utilized as controlling logic in computer systems. The simulation result shows that significant improvements could be obtained by running such a parallel simulation. Shi, Watson, and Chen [31] developed a simulation model that captures the random cache memory demand of World Wide Web (WWW). The main purpose of their study is to determine a cache removal policy and associated system parameters that result in fast response to users.

Discrete event simulation is widely applied to supply chain design and evaluation. Lee, Cho, and Kim developed a supply chain simulation framework for operation, tactical, and strategic levels [32]. By combining continuous equations in the simulation model, both continuous and discrete processes in the supply chain can be presented in the same simulation model. Lendermann et al. developed the framework of supply chain simulation by simulating the operations in each facility and the transportation between facilities [33]. Each facility has

one simulation model and shares information using the Internet. A case study of a semiconductor supply chain is provided in the paper.

Simulation models can also be utilized as environment generators for training tools. For complex systems that have high operating costs or hazardous environments, experienced operators are typically required to avoid operating errors and injuries. To develop the training programs for inexperienced operators, simulation models are used to generate different events without affecting the real system or exposing operators in an unsafe environment [34,35].

Shikalger, Fronckwiak, and MacNair developed a detailed simulation model for a 300 millimeter wafer fabrication line [36]. Their simulation model includes several different types of tool processes linked by an AMHS. In a following paper, model refinements included several cluster tools that determined different production batch sizes [37]. The objective of their research is to determine the operating policies that result in a relatively small average work in process (WIP). However, the authors did not specify the system operating rules. Their results would be easier to appreciate if operator assignments and operational rules had been described in their study. In the paper, the authors mentioned that the average CPU time to complete one simulation run is 24 hours. As noted, the long CPU time is the major difficulty of simulation approaches for large-scale systems.

2.3 SIMULATION OPTIMIZATION

Accurate simulation models can generate realistic results when correct inputs are provided [38]. In contrast, simulation models are descriptive and cannot find optimal or improved solutions by themselves. The simulation running time for one replication is much longer than calculating one

objective value (iteration) in mathematical programming. In addition, to evaluate one alternative with a simulation model requires multiple replications. To search for improved solutions, simulation optimization methodologies have been developed to determine the designs that will give the optimal (best) performance. As noted, simulation optimization methodologies cannot guarantee to obtain optimal solutions for all problem types. The goal of simulation optimization is to determine near optimal solutions by simulating a relatively small number of scenarios instead of simulating all or nearly all scenarios. Even with these techniques, identifying improved optimal solutions for simulation models still requires a relatively large number of iterations. In practice, a significant improvement in system performance may not be found before the decision needs to be made for the real system. The development of optimization modules for simulation software has became one of the more popular and important topics in the discrete system simulation research area [39, 40].

Sadoun identified optimal operating strategies for traffic intersections by simulating the traffic system [41]. For these strategies, the optimal timing for traffic lights was solved by an analytical model and evaluated by a simulation model. In this paper, the author tested strategies that can stop vehicles from making left turns, but the simulation modeling approach to capture the interactions among vehicles moving in different directions was not specified. In addition, the traffic light timing was solved by an analytical model without explaining why the timing is an optimal or near optimal solution for the simulation model.

Lee and Chi applied the symbolic Discrete Event Specification (DEVS) simulation to the traffic light timing problem [42]. This DEVS simulation model was built by dividing the traffic system into two layers: vehicles on a road network and traffic light control devices. A case study

with two traffic intersections was provided in the paper. Similar to other traffic system simulation papers, the case study did not allow vehicles to make left turns in the traffic system.

Generally, there are two directions for simulation optimization. If the response of the simulation models can be estimated by a formula, the formula will be utilized for the simulation meta model. If the response cannot be estimated, simulation models are utilized as the objective function for heuristics or optimization techniques. Common simulation optimization methodologies include: i. Multiple linear regression, ii. Artificial Neural Networks (ANN), and iii. Heuristics.

2.3.1 Multiple linear regression

Multiple linear regression is a technique that analyzes the relationship between independent (decision) variables and dependent variables (objective values) [43]. The certainty assumption in mathematical programming assumes that objective functions are known and the parameters are deterministic. With a deterministic objective function, the optimization problem can be solved by identifying the decision variable values that results in the optimal objective value. Due to the stochastic nature of real world problems, the certainty assumption is not valid for the majority of complex problems. Instead of having the exact objective function, multiple linear regression can be applied to estimate the surface of the objective function based on a number of observations from the simulated system. If validated regression models are obtained, they can estimate the dependent (objective) variable's value for specified independent variable values within a predefined range.

For problems in which exact or approximate objective functions cannot be obtained, simulation models can be utilized as the objective function since they can capture the stochastic

behavior of the real system. For certain of these problems, multiple linear regression can be applied to analyze the simulation results with different independent variable values. If validated regression models are found, then the dependent variable values corresponding to input variables can be estimated. The ultimate objective of utilizing multiple linear regression is to develop a function that can replace the simulation model. The validated regression model based on the simulation results is an estimation model for the simulation model and called a simulation meta model [44].

Mackulak and Savory [45] applied simulation meta modeling and experimental design methods to optimize the performance of two AMHS systems in a semiconductor fabrication facility. The result showed that significant improvements were obtained by optimizing a meta model which has five factors (area and equipment utilization, processing time for two stations, and vehicle speed in the facility) for each of the AMHS systems. Durieus and Pierreval [46] developed a regression meta model for an AMHS in a manufacturing system. The regression analyses detected the significant factors of the AMHS. The optimal AMHS design is then determined by optimizing the regression model. However, the generic manufacturing processes in this paper are over-simplified by assuming there are no failures and no lack of parts to be processed. These assumptions may have a significant effect on the simulation results.

Irizarry, Wilson, and Trevino [47] developed a generic simulator that is capable of modeling various types of machines in cells. A general method to simulate cell manufacturing systems is developed in order to evaluate the manufacturing-cell design. In this paper, a generic cell simulator contains a number of machines arranged in a U-shape. The simulator allows personnel in the cell to operate multiple machines. Complex operations can be processed in a cell. In addition, response surface meta models based on the simulation results were applied to

optimize the manufacturing design [48]. Bose and Pekny [49] provided a framework for forecasting, optimization (Mixed Integer Programming), and simulation aspects of the supply chain. The optimization module provides the schedule for the simulation modules' input. However, the long computational time limits the number of tasks that can be scheduled by MIP. Dengiz and Akbay [50] compared the performance of push and pull systems for a PCB production line by simulation. In the pull system, batch size is optimized by building a regression model using the simulation results.

For certain simulation problems, system constraints can be related to dependent variables. Yang, Kuo, and Chou [51] developed a dual-response method to estimate multiple simulation dependent variables. The method categorizes dependent variables into primary and secondary responses. The primary response is the objective function, and the secondary response is the system constraints. Since the dependent variables' values cannot be obtained until a number of simulation runs are completed and the dependent variables are random variables, the primary and secondary responses are estimated based on simulation results for a number of simulation scenarios. Good estimation of primary and secondary responses can predict near optimal solutions in the feasible region. In the case study, the dual-response method found a near optimal solution based on a small number of samples compared to a commercial simulation optimization package called OptQuest®.

Multiple linear regression meta modeling can determine improved or near optimal solutions for a simulation model using a relatively small number of scenarios, but the solution quality is not guaranteed. A generation, evaluation, and selection of alternatives via simulation methodology that identifies near optimal solutions was developed by Otamendi [52]. By running a large number of scenarios and reducing the region of feasible solutions, the

methodology can identify optimal solutions by simulating less then 3.5 percent of the total feasible solutions. Although the optimal solution for a case study was found by the method, the author specified that it cannot be applied to large-scale simulation models due to the extremely long CPU time. In addition, complex simulation models with more constraints may require simulating a larger proportion of feasible solutions to determine the optimal solution.

2.3.2 Artificial neural networks

Like linear regression, Artificial Neural Networks (ANN) are widely applied in simulation optimization. The ANN is a network constructed by multiple layers of nodes. In ANN, there are input, hidden, and output layers connected with directed arcs. A validated ANN is developed by using one data set to train the ANN and another data set to test it. ANNs are suitable for problems with characteristics of noise, poorly defined characteristics, and changing environments [53]. Simulation models are utilized to generate data for training and testing ANN. The simulation meta models formed by ANN have the potential to accurately predict the simulation results for new scenarios. Kilmer developed a baseline ANN meta model approach that can predict mean values and variances for simulation scenarios in three phases [54]. These phases are: determining the simulation results for input information, backpropagation for ANN training, and evaluating the trained ANN. Backpropagation is divided into two stages. The first stage processes the training data in the ANN and results in target and error values. The second stage adjusts parameters in ANN to reduce the total squared error of the network [55].

Barton discussed existing techniques for forming meta models [56]. ANN is one of the techniques that is utilized to compute model coefficients. In this paper, all existing techniques were designed for problems with continuous variables. Laguna and Martí [57] compared online

training procedures: backpropagation algorithm, Simulated Annealing, Genetic Algorithm, Tabu Search, and Scatter Search. Of the six problems tested, five had only continuous values and one had absolute values. The empirical results show that Scatter Search outperformed other methods by having accurate predictive ability and using shorter CPU time.

Lee, Gupta, and Amar [58] developed a multi neural network approach to simultaneously solve lot sizing and sequencing problems for a job shop simulation. The results show that a multi neural network can determine better solutions than a Random Search or a neural network that only solves sequencing problems. In this paper, it is not clear how much data are required for the neural network training. Further, the simulation model only uses uniform distributions. It would have been more convincing if the authors had applied the technique to benchmark problems in order to compare it to other methodologies.

An approach that utilizes GA and ANN for simulation optimization was investigated by Caskey [59]. ANN analyzes the population generated by GA to determine an improved solution. Although the result in the case study showed the approach can identify good solutions, there is no comparison to other simulation optimization techniques. In addition, the paper did not specify the number of simulation runs for the test problems.

2.3.3 Heuristics

A heuristic is defined as a technique to search for near optimal solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality [60]. Heuristics are generally applied to real world problems where the computational time to solve them by optimization techniques are longer than the time needed for a decision.

The motivation for utilizing heuristics to identify near optimal solutions for simulation models is to reduce the number of simulation runs, resulting in shorter total CPU times. Azadivar and Lee developed a procedure that utilizes simplex and complex search techniques to explore feasible regions for simulation problems [61]. The procedure can be applied to problems with continuous and discrete decision variables, but combinatorial problems were not discussed in this paper. Andradóttir modeled discrete event systems by a Markov chain and then utilized gradient estimation to optimize the performance of the Markov chain [62]. In the later paper, a local search methodology that provides an "almost surely" local optimum was developed by Andradóttir [63]. The approach estimates the improving direction for a solution in a predefined neighborhood. The weakness of the approach is that formulas to model the system can be limited and the case study utilized formulas with random variables as the test problem instead of a simulation model.

A two-phased strategy for simulation optimization was developed by Bowden and Hall [64]. Their strategy utilizes a search method that changes one variable at a time to identify possible improved solutions based on solutions found by an evolutionary strategy. Although this approach determined good solutions for the case study, the size of the problem and required CPU time were not mentioned.

Arsham [65] presented a method for the detailed design, analysis, and operation of incorporating algorithms with discrete event system simulation results. Eight algorithms are discussed in this study that provides sensitivity information under the assumption that well built simulation models are available. However, these approaches did not present a justification for the necessary simulation runs. Hutchison and Hill [66] used gradient estimation and simulation

to minimize airline delays (20 percent of the delay time was reduced by introducing penalty functions into the model).

There are also project management decision making tools that utilize simulation. Subramanian, Pekny, and Reklatits developed a framework for research and the development of pipeline management [67]. The computing architecture "Sim-Opt" repeats the process of utilizing MIP and heuristics to solve stochastic programming problems and then simulates the next stage. In a following paper [68], heuristics designed for knapsack problems were utilized to resolve the capacity conflicts in the schedule. Nandi and Rogers developed simulation models to train operating rules for acceptance/rejection decisions for manufacturers' make orders [69]. The simulation model uses the current manufacturer's operations as the initial condition and then simulates the future operations for both acceptance and rejection decisions to determine if accepting the order can have positive contributions to the manufacturer. Instead of using the revenue of the decisions (acceptance or rejection) as the selection rule, the authors defined the selection rule for the order to be accepted when the total revenue for accepting an order exceeds a specified proportion of total revenue for rejecting the order.

For multi-criteria simulation optimization, there are three different structures based on the time to collect the required information. These structures can operate prior to the optimization, during the optimization, and after the optimization [70]. Pukkala and Miina developed a simulation optimization approach on stand management (forest products industry) [71]. Their approach applies a nonlinear programming algorithm to multi-objective simulation problems. No detailed discussion about the algorithm or the simulation model data was presented in the paper.

Heuristics combined with simulation are also utilized to solve production scheduling problems. Backward simulation approaches for component release [72] and order release were developed by Watson, Medeiros, and Sadowski [73]. By starting from the final state, the simulation model can identify the time to release components and orders. Similar to the critical path method, backward simulation examines schedule feasibility by incorporating randomly distributed setup, processing, and move times. A bi-directional simulation algorithm that combines forward and backward simulation was developed for order release planning [74]. In each iteration, the algorithm runs one forward and one backward simulation with a local search to improve the order release plans. The forward and backward simulations provide the feedback of the current solution in order to determine the solution for the next iteration. The case study in the paper shows that significant improvements can be obtained by the bi-directional simulation algorithm.

Job shop schedules are commonly evaluated by simulation models. A simulation optimization approach was developed to identify job shop schedules by testing scheduling rules based on due date, lead time, and machine utilization [75]. The approach can evaluate different scheduling rules based on an evaluation function that uses lead time and machine utilization as elements. One issue with this approach is that certain potentially good solutions will not be considered due to scheduling rules blocking them.

Turki, Andijani, and Shaikh developed scheduling rule for stochastic single-machine scheduling problem [76]. By estimating the time allowance for each job, the simulation results show that the both mean and variance of job completion time can be reduced. However, the defined measurements in this paper are not clearly explained and the rule is only validated for single-machine scheduling.

2.4 SIMULATION OPTIMIZATION USING META HEURISTICS

As noted, heuristics are search methodologies that find near optimal solutions. Meta heuristics are designed to solve broad classes of problems that include simulation optimization problems. In simulation optimization, each objective value calculation requires multiple simulation runs. The required CPU time for obtaining objective values through simulation is much longer compared to using deterministic objective functions for deterministic optimization problems. For simulation optimization techniques, problems with continuous values can typically be simplified into meta models that can be used to predict simulation results. For combinatorial problems, meta heuristics are widely used to identify good solutions. Two meta heuristics, Random Search and SA, are frequently utilized to solve combinatorial simulation optimization problems [77]. Since the relationship between decision variables and the objective value is unknown, the heuristics select solutions with certain probabilities and record the best solution found during the search process.

Applying meta heuristics to identify improved designs for simulation problems does not guarantee solution quality. The advantages of utilizing meta heuristics are that they can be applied to various types of problems, and improvements are found by evaluating only a small portion of all solutions. In contrast, methodologies that provide performance guarantees require large computational efforts that cannot be implemented for large-scale simulation models [78]. Banks et al. pointed out that common techniques to optimize simulation results by choosing system parameters include Genetic Algorithm, Tabu Search, and Random Search Algorithms [79]. Simulated Annealing and Scatter Search are meta heuristics frequently utilized for optimizing simulation results as well. The simulation model is utilized as the evaluation (objective) function in these heuristics. The major strengths of meta heuristics in simulation

optimization are that they can be used on a large set of problems and they don't require knowledge of anything in the simulation model [80]. The next sections describe certain of these meta heuristics.

2.4.1 Simulated Annealing

The concept of Simulated Annealing (SA) was first published in 1953 by Metropolis et al. Annealing is a material cooling process that first heats the material to melting point and then decreases the temperature. Material can be formed into the preferable structure by controlling the temperature change during the cooling process. SA is a variant of a local search that solves optimization problems using the same concept. SA gives flexible search directions by first assigning higher temperature and then reduces the flexibility to identify preferable solutions by decreasing the temperature. The SA algorithm for minimization problems is stated as follows [60]:

Select an initial solution s_0;

Select an initial temperature $t_0 > 0$;

Select a temperature reduction function α;

Repeat

Repeat

Randomly select $s \in N(s_0)$;

$\delta = f(s) - f(s_0)$;

If $\delta < 0$;

Then $s_0 = s$

Else

Generate a random number x uniformly in the range (0, 1);

If $x < e^{(-\delta/t)}$, then $s_0 = s$;

Until iteration number = *nerp*;

Set $t = \alpha(t)$;

Until stopping condition = true.

Output the best found solution s_0.

Empirical experiments show that near optimal results for both linear and non-linear problems can be found by SA; more detailed discussion of SA is given in [81] and [82]. A SA algorithm approach for simulation optimization was developed by Alrefaei and Andradóttir [83]. The methodology approximates the objective function of a simulation model and solves the objective function by the SA algorithm. In their papers, the underlining assumption is that the system can be modeled as a stochastic function or a Markov chain. However, one of the major reasons that simulation models are developed is because the system cannot be simply modeled by other techniques.

Handock and Mitenthal developed an SA approach for simulation optimization [84]. The paper demonstrated that SA can solve simulation problems with integer variables. In the case study, a flexible manufacturing system design problem was solved. However, the CPU time for large problems can be extremely long. Alkhamis et al. developed a modified SA algorithm that can converge to the global optimum when applied to simulation optimization [85]. Their proof shows that the global optimum can be reached using their SA approach with a finite number of iterations. However, the number of iterations to obtain the global optimum can be extremely large and cannot be achieved for many simulation problems.

Wieland and Holden developed a SA approach to solve aviation delay problems [86]. Although the approach can determine improved solutions, the simulation models in the paper were simplistic Monte Carlo simulation models that were built using deterministic functions and random variables following a normal distribution. In addition, the number of iterations required for this approach is large even though the simulation models were simple. For more detailed simulation models, the required computational time to determine an improved solution will be a major problem.

2.4.2 Genetic Algorithm

Genetic Algorithm (GA) is a meta heuristic that solves problems based on recombining solutions and assigning mutations to new solutions. It was developed by Holland and associated researchers in the 1960s and 1970s. The concept of GA originally comes from biology where offspring of plants and animals have a greater chance of survival due to certain desirable genes from parents. To present the structure of GA, a problem solution is encoded into components as genes of plants or animals. GA combines components of existing solutions to create new solutions in searching for characteristics that result in good solution quality [60]. The framework of GA is stated as follows:

Generate a population of solutions.

Generate offspring by combining components of existing solutions.

Perform mutation to offspring.

Evaluate fitness (objective) values of offspring.

If a better solution is found, record it.

Add offspring to population and delete a portion of population with worse fitness values.

Repeat until terminating condition occurs

GA starts by creating a group of solutions (population) that contain diversified decision variable values. The offspring (new) solutions are then generated by randomly recombining and modifying solutions in the population. At the end of a generation (iteration), the solutions that are associated with better objective values will remain in the population for next generation. After a number of generations, near optimal solutions will be identified. GA typically requires a fast method to evaluate the solution since the objective values for the population in every generation need to be collected. For a more detailed discussion of GA, one can refer to [87] and [88].

GA is one of the methodologies utilized in simulation optimization, where the evaluation function is replaced by the simulation model. Yunker and Tew [89] demonstrated the use of GA as a simulation optimization technique. A parameter setting problem of a university time-shared computer system is used as the case study. The authors concluded that GA can perform better than a local search and multiple linear regression. However, the size of the computational task for the case study was not mentioned, and the GA used approximately ten times the amount of CPU time as the other methods. A GA based simulation optimization method was developed by Al-Aomar [90]. It is intended to solve parameter design problems for production lines and business systems. The method combines the solution mean and variance as one of the measures.

The advantage of using GA for simulation optimization is that it can solve combinatorial problems. Azadivar and Tompkins developed a GA approach for production line design problems [91]. The results show that GA can perform better than a Random Search in terms of solution quality. However, the GA was required to evaluate a relatively large number of solutions, and this would be the major difficulty for large-scale simulation models.

GA approaches have been developed for distribution and manufacturing problems. For AMHS, developing effective operating rules is often a major challenge. Feyzbakhsh et al. developed a parameter setting approach for AMHS based production systems that allow operators to wait for incoming items before the next process starts [92]. A GA was utilized as the benchmark method to solve the simulation problem. However, only a single station simulation without breakdowns was considered.

Kochel and Nielander developed an approach that utilizes GA to optimize simulation problems related to Kanban manufacturing systems [93]. One concern in this paper is that the measurement is obtained by using steady-state simulation. For most manufacturing environments, it is hard to reach steady-state.

Another example of a distribution center scheduling problem solved by a GA based simulation approach is given by McWilliams et al. [94]. The decision variables in the scheduling problem are the job input sequences. Even with only 18 jobs in the problem, the number of possible solutions can be greater than 10^{10}. In the study, simulation results show that improved solutions can be determined by the GA approach. The major issue with this approach is that population needs to be evaluated by simulation for every generation. For large-scale simulation models, this requires an extremely long CPU time.

Similar to GA, Evolutionary Algorithm (EA) uses concepts of recombining and changing solutions. In general, EA emphasizes the process of randomly changing partial solutions (mutation) to determine improved solutions. Pierreval and Tautou developed a simulation optimization method that utilizes an EA for the optimization process [95]. A production planning problem is provided for the case study. The problem was sequentially solved by utilizing deterministic and stochastic simulation models. The solutions for both the deterministic

and stochastic simulation models had no statistically significant differences. Although the deterministic simulation models can reduce the number of replications to one, the deterministic simulation models cannot accurately represent random events such as system breakdowns or production failures.

Most simulation optimization methods are focused on parameter tuning. Pierreval and Paris determined the near optimal system configuration by an evolutionary simulation configuration method [96]. The method applies the EA to the simulation model. In this paper, an example of cell manufacturing system design is presented. The case study shows that the number of stations and the lot size for each station can be determined by the evolutionary simulation configuration method.

2.4.3 Tabu Search

Tabu Search developed by Glover is a variant of local search with the ability to solve nonlinear and combinatorial problems [97]. TS identifies solutions by evaluating a portion or all solutions in the neighborhood and tracking the solution's moving history. The general TS algorithm is given as follows [98]:

Initialization:

Select a starting solution $\boldsymbol{x}^{now} \in \mathbf{X}$

Define neighborhood $N(H, \boldsymbol{x}^{now})$ for current solution $\boldsymbol{x}^{now}$,

Define subset of neighborhood, $candidate\_N(\boldsymbol{x}^{now})$, that will be evaluated.

Record the current best known solution by setting $\boldsymbol{x}^{best} = \boldsymbol{x}^{now}$

Define $best\_cost = c(\boldsymbol{x}^{best})$.

Choice and termination:

Determine *candidate_N*($\boldsymbol{x}^{now}$) as a subset of $N(H, \boldsymbol{x}^{now})$. Select $\boldsymbol{x}^{next}$ from *Candidate_N*($\boldsymbol{x}^{now}$) to minimize $c(H, \boldsymbol{x}^{now})$ over this set. ($\boldsymbol{x}^{next}$ is called a highest evaluation element of *Candidate_N*($\boldsymbol{x}^{now}$).) Terminate by a chosen iteration cut-off rule.

Update

Reset $\boldsymbol{x}^{now} = \boldsymbol{x}^{next}$

If $c(\boldsymbol{x}^{now}) < best\_cost$, $\boldsymbol{x}^{best} = \boldsymbol{x}^{now}$ and $best\_cost = c(\boldsymbol{x}^{best})$.

Update the history record H.

Empirical experiments show that TS can perform well for both deterministic and stochastic problems with combinatorial decision variables. Details of TS variants are described in [99] and [100]. Dengiz and Alabas applied TS to optimize buffer size in Kanban-controlled systems [101]. The case study showed that TS can determine better solutions compared to a Random Search algorithm. In this paper, the authors found that the TS approach becomes very time consuming when problems contain too many decision variables. Problems containing many decision variables will result in a large number of solutions in a predefined neighborhood that cannot be effectively evaluated by TS approach.

Jacobson and Yuesan discussed issues of TS and SA for both deterministic optimization and simulation problems [102]. The authors proved that the methods for searching in a predefined solution neighborhood (neighborhood search) are NP-hard, which is the major reason why a large proportion of work on TS and SA performance evaluation are based on empirical tests.

A simulation optimization method that combines partitioning, selection procedure, and local improvement techniques was developed by Pichitlamken and Nelson [103]. The method

includes provable convergences and quality empirical results. However, this paper did not provide a comparison with other existing techniques, and the number of simulation replications appeared to be large. The experimental results show that 10,000 replications were carried out for both test examples.

2.4.4 Scatter Search

Scatter Search (SS) has been developed by Glover, Laguna, and Marti, the outline of it is as follows [104]:

1. Randomly generate a set of diversified trial solutions and select a subset as reference set.
2. Divide solutions into subsets and apply heuristics to determine improved solutions for each subset.
3. Collect best solutions in subsets as reference sets and generate new solutions by combining subsets in reference solutions. The combinations are
    a. Create feasible and infeasible solutions.
    b. Repair functions
4. Update the reference set by replacing old solutions in reference sets with best solutions found. Repeat 3 and 4 until reference set is not improving.
5. Restart from beginning until predefined iteration number is reached.

Scatter Search utilizes heuristics to improve solutions quickly and avoid local optimum by using a group of solutions, restarting step 1, and using penalty functions for infeasible solutions. Empirical tests show that Scatter Search can find improved solutions with a lower number of iterations compared to other existing meta heuristics. The comprehensive description of SS can be found in [105].

Martí, Lourenço, and Laguna [106] utilized Scatter Search to solve assignment problems with the objective of balancing the workload and fitting available time periods for school teaching assistants. However, the paper only compares SS with MIP optimization software, Cplex 6.5. It would be easier to assess the SS performance if other meta heuristics had used to solve the same test problems.

Even with the fewer number of required iterations in SS, the number of required simulation runs might still be large. As with to other meta heuristics, SS needs a relatively fast evaluation function to calculate objective values. For the simulation optimization that utilizes SS, simulation is still used as a "black box" that generates objective values [107]. As the size of the simulation increases, the run time will become a major constraint.

SS requires a large number of evaluations for each step. Campos, Glover, Laguna, and Martí used SS to solve linear ordering problems [108]. In this paper, there were ten diversification generators; a diversity measure is developed to evaluate the performance of these generators. These generators combine multiple Random Search and local search strategies including Greedy Randomized Adaptive Search Procedures (GRASP), random generator, and Tabu Search.

OptQuest®, a commercial optimization software package utilizes SS as its underlying algorithm. OptQuest is developed to solve nonlinear and combinatorial problems and incorporated in the commercial simulation software Arena® to solve simulation problems. Several studies have utilized the OptQuest in Arena as the bench mark simulation optimization algorithm for comparison purposes [40, 51, 109].

For simulation optimization, some meta heuristics use simulation meta models in the optimization process to eliminate solutions that meta model predicts to be bad [110]. The meta

heuristics are utilized to search for improved solutions and to maintain feasibility of the solution, while the simulation meta model is developed to identify bad solutions. The number of simulation runs can be reduced by combining the simulation meta model and meta heuristics. However, a valid simulation meta model is required to determine the solution quality of the meta heuristics. In general, meta heuristics are applied to simulation optimization because the problem structure is complex, and valid simulation meta models are hard to construct. Applying meta models that do not yield accurate predictions will lead to the risk of eliminating "good" solutions.

2.5 SUMMARY

A review of the subjects associated with this research has been presented. The topics discussed include simulation modeling, mathematical programming, and simulation optimization methodologies. The major challenges and strengths of deterministic approaches based on mathematical programming were discussed. Meta heuristics designed to solve combinatorial problems including Simulated Annealing, Genetic Algorithm, Tabu Search, and Scatter Search were introduced. The techniques of large-scale simulation modeling and laws of system decomposition were shown to simulate complex systems without making unrealistic assumptions. To obtain more realistic simulation results, more details must be incorporated into the simulation models. In contrast, the CPU times to execute complex simulation models are relatively long. This limits the number of system designs that can be evaluated by simulation models in a specified time period. Methodologies for utilizing optimization techniques to solve simulation problems were introduced. To solve simulation problems with continuous decision

variables, simulation meta modeling and ANN were presented. The major challenges of simulation problems with combinatorial decision variables were introduced. The advantages of dominant meta heuristics that incorporate simulation were discussed. Finally, the limitations of utilizing heuristics on simulation optimization support the needs of developing improved methodology.

3.0 SORTATION SYSTEM

In this dissertation, we research a sortation system in a distribution center that uses an Automatic Material Handling System (AMHS). This system is one of the largest sortation systems in the U.S. owned by a major company in the distribution industry. The system flow chart is shown in Figure 3-1.

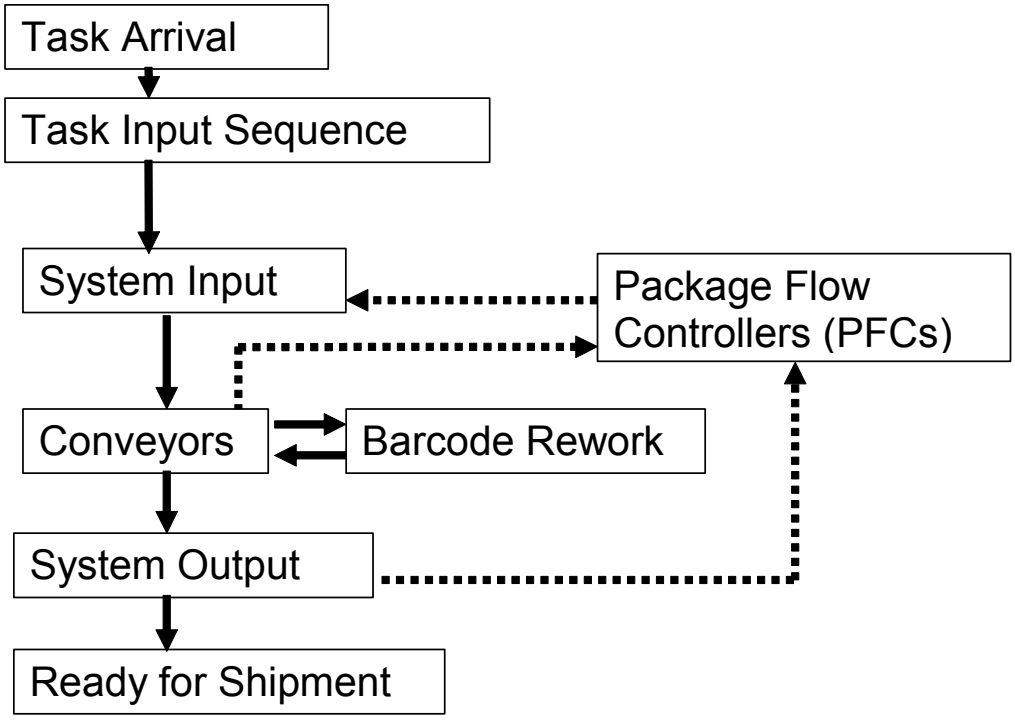

Figure 3—1 AMHS flow chart

3.1 SORTATION OPERATIONS

In the sortation system, tasks are defined as trailers loaded with packages. In the system input, packages in trailers are unloaded manually and put onto conveyors then transferred to the system output. In the system output, loaders load the sorted packages onto another group of trailers for different destinations.

In a typical shift, 80 to 85 tasks arrive at the system for processing. Each task contains between 750 and 900 packages that are assigned to 136 different destination load doors in the system output area. The number of packages assigned to each load door is unique for each task.

In the system input area, there are 20 to 23 unloaders working simultaneously to unload packages into the sortation system. Each unloads an individual task and no more than 32 tasks can be processed at the same time. Before tasks are processed, each of them is assigned a priority by facility management in order to determine the task input sequence for the unloading operation.

The Package Flow Controller (PFC) provides the controlling logic in the system monitoring the number of packages in certain conveyor segments. If there are too many packages in the system, the PFC will give feedback (warnings) to adjust the conveyors' speeds. One warning, the Chute Full (CF) signal, is defined as a load door in the system output that is fully occupied by packages. The duration of each warning condition is recorded as a measure of effectiveness.

The packages unloaded in system input area are transferred to the system output area by conveyors and then manually processed by loaders. The system output area contains eight physical subareas each with 14 to 17 load doors. The load doors in a subarea are divided into

two groups and each group is handled by a team of loaders. In the sortation system, there are 16 teams of loaders and the total number of loaders ranges from 65 to 75.

The sortation system overall performance for a shift is measured by the makespan which is defined as the time between when the first package is unloaded and the last package is loaded. Because a major portion of the operational cost is labor, the common goal for sortation system operations is to process tasks within the shortest makespan time.

3.2 SIMULATION MODELING

To simulate the distribution center, the sortation system is first analyzed and decomposed into subsystems. After the subsystem modules are developed, the full scale simulation model is constructed by combining the subsystem modules. The structure of the general distribution system simulation models includes five types of modules: system input module, system output module, conveyor module, operator module, and system breakdown module.

The system input and system output modules simulate the package unloading and loading processes. In both the unloading and loading processes, the processing times change based on the number of packages in the tasks. In order to accurately simulate distribution centers, system input and output modules must utilize actual operator assignments and operating policies which are modeled in the operator module. The input and output simulation modules also provide the function of tracking the number of packages in each task; i.e., counters are utilized to track the number of packages in each task and thus reduce the number of entities that represent packages in the simulation model. Consequently, the reduced number of entities in the simulation model will result in a shorter simulation run times.

The conveyor module transfers the packages from system input to system output at constant speeds (determined by the controllers). The function of re-directing packages due to system breakdowns is built into the conveyor module. The operator module controls the number of operators in the system input and output modules. Although it is difficult to modify system designs (the facility layout), operating rules and number of operators are easily changed. To provide the ability to efficiently modify operator assignments, the operator module is separated from the system input and output modules. The input and output modules will reference the operator module while executing the simulation. Most studies of large-scale facilities, both deterministic and simulation approaches, have not incorporated operator travel time into the model. The operator module in this study introduces operator moving time (i.e., delays) with predefined travel time functions.

In a distribution system, every subsystem has the potential of breaking down (jam). While breakdowns cannot be predicted, the possibility of system breakdown is a function of the number of packages on the conveyors. The system breakdown module simulates the PFC system warnings and breakdowns by monitoring every conveyor segment. All simulation modules have to frequently check the breakdown module. If a breakdown occurs in a certain location, the corresponding input (output) module and operator module will stop the current operations until the breakdown is cleared.

For the test problems in this study, the objective is to identify near optimal system designs for the AMHS sortation system through the simulation model. Due to the complexity of the system, the full scale simulation model requires 30 minutes of CPU time (on a Pentium IV, 3.0 GHz computer) to simulate one replication[*]. In practice, the system inputs change periodically,

[*] Models used for later Chapters are modified to require only 20 minutes of CPU time.

and because of the relatively long simulation run time, it is prohibitive to simulate all possible scenarios. Consequently, it is necessary to determine an improved design within a short amount of time.

3.3 SUMMARY

In this chapter, general sortation systems that utilize AMHS were introduced. These sortation systems are designed for distribution industry usage. The detailed operations in sortation systems including system input and output were discussed. An overview of the simulation modeling approach is given. The sortation system simulation model is based on subsystem modules. These provide the flexibility for expanding simulation models and modifying operating policies. The operator module in the simulation model provides the ability to simulate operators' traveling time penalty which has not been found in previous research. Finally, the overall performance measure of sortation systems is defined as the makespan, which will be used for test problems in this dissertation.

4.0 REGRESSION META MODELING

As noted, multiple linear regression is one of the simulation meta modeling techniques that utilizes simulation results as samples [111]. Although simulation models have the ability to estimate the impact of changing system designs, the number of alternative system designs can be extremely large and therefore cannot be simulated within a reasonable time. Multiple linear regression can be utilized to analyze the relationship among decision variables and objective values of simulation problems.

Regression models that contain k independent variables have the follow format:

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \varepsilon \qquad (4\text{–}1)$$

Independent variables (predictor/regressor) are represented as x_i and the dependent (response) variable is represented as y; ε is the measurement error, representing the stochastic nature of the system that cannot be captured by the regression model, where the expected value of ε is zero. Hence, for the expected value of y, ε will not be shown in the model. Regression models can be fitted to analyze the relationship between x_i and y. Given a number of samples of independent variable settings x_i and the response y in the system, a regression model can be formed to represent the system response by determining each independent variable's contribution (β value) to the response. The objective of regression is to predict the y values for the x_i for which we do not have samples within a predefined range of x_i.

Hence, a validated regression meta model can predict the simulator's output. The ultimate goal of utilizing multiple linear regression is to develop a function that can replace the simulation model. The regression meta model can predict dependent variable values without executing the simulation. The major strength of this technique is that a large portion of the computational tasks required for simulation can be reduced. In terms of computational efforts, the CPU time to run a simulation model is much longer than solving a regression model, although the meta model still requires a number of simulation runs for fitting.

Since the regression meta models are multiple linear regression models, the criteria to determine their predictive ability of regression meta models are the same as for regression models. Validated regression models are evaluated by: 1.) significance of regression, 2.) significance of coefficients, and 3.) R^2 values.

Significance of regression determines if there is a linear relationship between the response y and each x_i. If the regression is significant, at least one independent variable x_i has a linear relationship with y. Significance of coefficients can identify the contribution of x_i given the other independent variables in the model. R^2 is called the coefficient of determination. The proportion of variance in the system that can be explained by the regression model is determined by R^2. In practice, regression models are good predictors when coefficients are statistically significant and R^2 is high (i.e., close to 1).

4.1 ESTABLISHING SYSTEM PARAMETER SETTINGS

In the sortation system, the Package Flow Controllers (PFC) monitor the number of packages in the conveyors and load doors (chutes) and then give feedback to control the conveyors' speeds.

If the AMHS has too many packages, the risk of system failure will increase. When the AMHS has more packages than the predefined warning parameters, the PFC will receive warning signals and then reduce or stop the speeds for unloading operations' conveyors. Not until packages on the AMHS are loaded and the number of packages becomes less than the warning parameters, will the PFC warning signals be cleared and unloading operations' conveyors will be increased to full speeds. The warning parameters include local warning parameters, global warning parameters, and the chute full percentage. The local warning parameters limit the number of packages that can be unloaded in a time period. PFC receive warning signals when the number of packages that have been unloaded during a time period for each unloading area exceeds local warning parameters. The global warning parameters control the number of packages that can be sorted in the recirculation sorter (system buffer). If the recirculation sorter has more packages than the global warnings parameters, the PFC will receive warning signals. The chute full percentage is the portion of the doors that are fully occupied by packages.

4.1.1 Regression meta model development

For the AMHS in this study, there are four unloading areas, one recirculation sorter, and 136 load doors (chute full parameter) that are monitored by the PFC. The number of PFC parameters in the unloading areas and recirculation sorter are eight (two warning levels for each unload area) and two (two warning levels for the sorter) respectively. For the chute full warning, there are six parameters for the portion of the overall load doors that are fully occupied. If we used all PFC parameters as independent variables to fit a regression model, the regression would include 16 variables (x_1 to x_{16}). In addition, to fit a validated regression model, it is common to include the

second order and interaction terms among these independent variables. A regression model for this PFC parameter setting is shown as follow:

y: total processing time (makespan).

x_1 to x_8: unloading area parameters.

x_9 and x_{10}: recirculation parameters.

x_{11} to x_{16}: chute full parameters.

$$y = \beta_0 + \sum_{i=1}^{16} \beta_i x_i + \sum_{j=1}^{16} \sum_{i \geq j}^{16} \beta_{ij} x_i x_j + \varepsilon \qquad (4\text{–}2)$$

If all variables are accepted in the regression model, the total number of independent variables in the regression model including second order and interaction terms can be as large as 152. It is highly likely that this will result in an over-fitted model. In practice, regression models have a high risk of prediction failure when there are more than ten variables. To maintain the predictive ability of regression meta models, the number of variables needs to be reduced to a controllable level.

One search methodology to solve problems with too many variables is to fix the values of most variables and change only a portion of variables in each iteration [64,66]. Because only one variable value is changed in each iteration, the result can only give an improving direction for the specified variable and the interactions among different variables cannot be captured. The major issue with this search methodology is that it does not guarantee the solution quality and there is no predictive ability.

Another approach to handle problems with too many variables is to reduce the number of variables by batching and using multipliers. One technique is to multiply current values of a variable set that have similar functions by a continuous variable multiplier. The benefit of introducing multipliers into the regression model is that the number of variables can be reduced

to a manageable size. The tradeoff is that the values of each variable set have to move in the same direction as the specified multiplier. Consequently, the multipliers might not be able to find a setting that is as good as if all variables were considered independently.

For the PFC parameter settings, three multipliers are introduced into the regression. The PFC local warning multiplier combines eight parameters in unload areas (x_1 to x_8). The PFC global warning multiplier combines two parameters in the recirculation sorter (x_9, x_{10}). Lastly, the chute full warning multiplier combines six parameters in the chute full warning (x_{11} to x_{16}). After reducing the number of variables, the regression model is stated as follows:

Variables

XF_1: PFC global warning multiplier.

XF_2: PFC local warning multiplier.

XF_3: Chute full warning multiplier.

Regression model

$$y = \beta_0 + \sum_{i=1}^{3} \beta_i XF + \sum_{j=1}^{3} \sum_{i \geq j}^{3} \beta_{ij} XF_i XF_j + \varepsilon \qquad (4\text{–}3)$$

To utilize multipliers in the PFC control logic, warning parameters are set using default values (constants) and then multiplied by a multiplier according to the parameter type. For example, PFC control logic utilizes $x_1(XF_2)$ to $x_8(XF_2)$ as local warning parameters instead of x_1 to x_8, where the x_1 to x_8 are constants and XF_2 is a continuous variable. If XF_2 is set as 1.2, all PFC local warning parameters in PFC will be increased by 20 percent.

After introducing multipliers into the model, the maximum number of variables in this regression model is nine. That is, although the multipliers limit the values of parameters, the number of variables in the regression model is reduced from 152 to nine, a manageable level.

In a pilot simulation, we investigated two types of system designs related to task input sequences. The first type randomly selected task input sequences. The result showed that the R^2 value (0.40) for the regression model was low. For complex systems, it is common that a certain portion of the variance cannot be explained by regression models. For the AMHS in this study, one of the major causes of variance is the task input sequence. To identify the variance caused by the task input sequence, the second type of experiments fixed the task input sequence in the simulation model. The pilot simulation results show that the R^2 is increased to 0.66 and the standard deviation of total processing time is reduced by 59 percent. The regression model is:

$$Y = 6.915\text{-}1.274XF_1\text{-}1.996XF_2\text{+}0.323XF_1^2\text{+}0.718XF_2^2 \qquad (4\text{–}4)$$

Although the task input sequences vary in different sorting operations, the experiment shows that fixing the task input sequence can increase the R^2 values. In later chapters, methodologies for developing the task input sequence that result in short makespan and small variances are discussed. The optimal solution of the regression model is $XF_1 = 1.6$, $XF_2 = 1.389$. By simulating the optimal solution of the regression model, the total processing time is reduced by 14.1 minutes, from the original simulation time of 4 hour 33 minutes. For the remaining test problems in this chapter, task input sequences are randomly generated.

4.1.2 Computational results

In order to identify the performance of regression meta modeling in predictive ability, six PFC parameter sets were utilized as test problems. These test problems contain different number of loading operators and tasks. The settings for these problems are listed below in Table 1:

Table 1 Scenarios for PFC parameter setting problem

Problem	Number of loaders	Number of tasks
1	85	84
2	85	75
3	85	95
4	69	84
5	69	75
6	69	95

In Table 1, the problems are designed using two sets (69 and 85) of loaders and three sets (75, 84, and 95) of incoming tasks. As noted, task input sequences are different everyday. To determine the near optimal PFC settings for a range of events, task input sequences were randomly generated for the test problems. For each test problem, each multiplier had continuous values between 0.9 and 1.7. For problems with continuous variables, the total number of possible settings is infinite. To simulate samples for regression meta modeling, each multiplier uses the value of 0.9, 1.3, and 1.7. With three multipliers in the test problems, there were 27 scenarios that needed to be simulated. The list of scenarios is stated as follows in Table 2:

Table 2 PFC parameter samples for regression meta modeling

Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14
XF_1	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.3	1.3	1.3	1.3	1.3
XF_2	0.9	0.9	0.9	1.3	1.3	1.3	1.7	1.7	1.7	0.9	0.9	0.9	1.3	1.3
XF_3	0.9	1.3	1.7	0.9	1.3	1.7	0.9	1.3	1.7	0.9	1.3	1.7	0.9	1.3
Scenario	15	16	17	18	19	20	21	22	23	24	25	26	27	
XF_1	1.3	1.3	1.3	1.3	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	
XF_2	1.3	1.7	1.7	1.7	0.9	0.9	0.9	1.3	1.3	1.3	1.7	1.7	1.7	
XF_3	1.7	0.9	1.3	1.7	0.9	1.3	1.7	0.9	1.3	1.7	0.9	1.3	1.7	

Problems 1 and 4 were first tested for regression meta modeling. After fitting the simulation results into regression models, valid regression meta models were found. For both

problems, 20 replications were simulated for each scenario. Although the large number of samples can lead to validated regression models, it is computationally expensive to replicate large numbers of samples for each problem. The average CPU time to simulate one replication is approximately 30 minutes. Simulating 540 samples (27×20) for regression meta modeling requires more than 11 days of CPU time.

In order to reduce CPU time, it is critical to reduce the number of samples without decreasing the performance of the regression model. From the simulation results for problems 1 and 4, we selected five replications of each scenario as samples to fit into regression models. The regression models with 135 samples were similar to those that have 540 samples. For problem 2, 3, 5, and 6, each scenario was simulated for five replications. The experimental results for regression meta modeling are shown in Table 3 below:

Table 3 Regression meta modeling results of PFC parameter settings problem

Problem	Sample size of regression model	R^2	S. D. (minutes)	Significant Value of Regression
1	540	0.661	5.7	< 0.000
2	135	0.125	3.8	< 0.000
3	135	0.382	6.6	< 0.000
4	540	0.216	8.5	< 0.000
5	135	0.203	6.8	< 0.000
6	135	0.251	10.9	< 0.000
Problem	**Predicted optimal objective value**	**Simulated Optimal objective value**	**Sample size of simulation**	**Best solution found by OptQuest**
1	3:55:19	3:57:18	20	3:56:24
2	3:33:50	3:49:37	5	3:48:36
3	4:31:01	4:27:47	5	4:35:20
4	4:18:00	4:17:38	20	4:19:37
5	3:52:01	3:49:52	5	3:51:40
6	5:01:23	4:58:34	5	4:59:17

Table 3 lists the results of regression meta modeling for the six test problems. The sample size of regression is the total number of samples utilized in the regression model. There were 27 scenarios simulated in the six test problems and the number of replications was 20 for problems 1 and 4 and five for other problems. The resultant number of samples for the regression models is 540 for problems 1 and 4 and 135 for other problems.

As noted, the R^2 value is the proportion of variance in the system that can be explained by the regression models. Due to uncontrollable factors (e.g., task input sequence) in the sortation system, the R^2 values of the regression models are relatively low as shown in Table 3. Even with the low R^2 values, the computational results still showed that the regression meta models predict makespan times for PFC parameter settings problems as discussed below.

The optimal values of the regression meta models for test problems are also given in Table 3. The independent variable values associated with the optimal objective values were then simulated and presented in the section of simulated optimal objective values. The results in Table 3 show that the objective values obtained by simulation are relatively close to the optimal objective values of regression meta models (i.e., within ± 3.5 minutes) except for problem 2, which has the lowest R^2 value (0.125) among test problems.

For comparison purposes, these six testing problems were solved by OptQuest in Arena. Each test problem was solved for 30 iterations in OptQuest. OptQuest has the advantage of using more simulation output to find improved solutions by executing 30 iterations (27 scenarios for regression meta modeling). The results for OptQuest are also presented in Table 3. After performing statistical tests to the six test problems, OptQuest did not find significantly improved solutions compared to best solutions of regression modeling. In comparing the best solutions found by regression meta modeling and OptQuest, five of the test problems had differences that

are less than 2 minutes. For problem 3, the best solution found by regression meta modeling is 7.6 minutes less than the OptQuest solution. In addition, the best solution found by OptQuest for problem 2 (lowest R^2 value) is close to the solution found by regression meta modeling (the difference is one minute).

4.2 LOADING POLICY PROBLEM

For the loading operations, each operator has an assigned coverage. Because the input tasks contain different numbers of packages for each load door, it is common for loaders to have no packages to load during certain time periods and too many packages during other phases of the operation. In order to reduce the makespan, the simulated operating rules allow loaders to help load adjacent doors (±1) with a travel time penalty of ten to fifteen seconds. Even though the loaders' utilization can be increased by helping at adjacent doors, moving loaders too often will increase the total travel time for loaders. Consequently, the total time loaders spend in loading will be reduced.

In the sortation system, loading policy determines the conditions for loaders to move to adjacent doors and to go back to their coverage doors by utilizing the information about the number of packages at each load door. Since the package volume for each door is different, each of the load doors could have different conditions. However, the package volume and task input sequence change on a daily basis. To determine the improved performance by utilizing this information, a general policy for every loader is needed. The goal is to develop a general loading policy that results in a shorter makespan.

The loading policy problem has a structure similar to that for the "cycle stealing" problems, which are problems for determining computer network station operational rules for underutilized stations to process jobs for stations that are highly utilized. For the one-way cycle stealing problem with unlimited buffer size, near exact solutions of job waiting time can be approximated. After discussing this problem with Takayuki Osogami, a PhD candidate in the Computer Science Department at Carnegie Mellon University, whose research is on cycle stealing problems [112, 113, and 114], it was verified that there are no techniques to determine the exact solution or to approximate the solution without oversimplifying this loading policy problem. To approximate the solution for the loading policy, we have to assume that a loader can help at other doors with a given probability regardless of the number of packages in the loader's coverage doors and outgoing trailers. Another difficulty in approximating the solution lies in how to include the system breakdowns and trailer switching process into the model.

4.2.1 Current loading policy

For the sortation system, loaders have a ten to fifteen second travel time when switching among load doors. In order to reduce the travel time, loaders cannot leave their assigned doors until there are no packages at the door. The loading policy is based on the available information at the load doors. In this study, the available information is the number of packages at each load door. The loading policy problem is defined as follows:

Assignment doors: Loader's assigned coverage doors.

Adjacent doors: Load doors next to a loader's assignment doors.

XL_1: Number of packages in adjacent doors for loaders to move from assignment doors to an adjacent door. XL_1 must be an integer.

XL_2: Number of packages in adjacent door for loaders to go back to assignment door. XL_2 must be an integer.

Y: Total processing time.

Current loading policy

1. For a loader who does not have any package at the assignment door and an adjacent door (± 1) has more than XL_1 packages, go to help.
2. For a loader who is working at an adjacent door with the assigned loader, if there are less than XL_2 packages, the loader will go back to the assignment door.

Constraints

$XL_1, XL_2 \leq 20$ (physical constraint)

$XL_2 < XL_1$ (operating constraint)

Regression model

$$Y = \beta_0 + \sum_{i=1}^{k} XL_i + \sum_{j=1}^{k} \sum_{i \leq j}^{k} \beta_{ij} XL_i XL_j \qquad (4\text{–}5)$$

The loading policy defines the conditions when loaders should help at other load doors. Once the conditions are no longer existing, loaders will switch back to their assigned doors. In the second part of the loading policy, the variable XL_2 is applied to load doors that are operating under double loading conditions. For an adjacent door with only one loader working, moving the loader to his assignment door will have a travel time cost. For a loader working alone at an adjacent door, the loader will go back to his assignment door when there are no packages left in the adjacent door.

The physical constraint represents the maximum number of packages that can be contained at a load door, which is 20. The operating constraint eliminates the situation in which loaders move between assignment and adjacent doors without processing any packages (e.g., if

XL_2 is greater than XL_1, a loaders will go to an adjacent door based on the first part of loading policy and then immediately move to the assignment door following the second part of loading policy). Note that there can be two adjacent doors for a loader. To implement the loading policy in the simulation model, each load door is prioritized for a loader to sequentially search for an adjacent door to work at. To identify the efficiency of the loading policy, two simulation problems with different numbers of loaders were utilized to construct a regression meta model. The independent variables are XL_1 and XL_2. The dependent variable Y is the total processing time (makespan). The experimental result is shown in Table 4 below:

Table 4 Regression meta models for current loading policy

Problem	Sample size	R^2	S. D. (minutes)	Significant Value of Regression
85 Loaders	200	0.172	5.7	< 0.000
69 Loaders	200	0.026	8.7	0.024
Problem	**Optimal objective value: regression**	**Optimal objective value: simulation**	**Best Sample found**	
85 Loaders	4:05:13	4:05:17	4:04:08	
69 Loaders	N/A	N/A	N/A	

The result shows that R^2 values for both problems are relatively low. Although the optimal solution predicted by the regression is similar to the simulation output for the 85 loaders problem, it does not guarantee the regression meta model can predict well for other independent variable values. For the problem with 69 loaders, the low R^2 value (0.026) indicates that only a small portion of variation in the system can be explained by the regression model; hence, the regression model is not considered as a valid model.

4.2.2 Alternative loading policies

One of the most important functions in simulation is determining the system performance for new system designs. Currently, the loading policy allows loaders to work at the adjacent doors next to the assignment doors, and the available information is the number of packages at the load doors. Two alternative approaches were investigated for the loading policy problem:

1.) Increase loaders' adjacent doors to ± 2 doors, from the current assignment doors.

2.) Utilize the information on the number of packages coming into doors.

To identify the system performance, simulation models and meta models for both alternatives need to be constructed.

The current operating rules in the sortation system limit the loaders so that they can only help at adjacent doors (±1). In order to identify the effect of increasing this, a simulation model that allows loaders to help at additional load doors (±2) is utilized as a test problem. To determine the performance with an increased number of adjacent doors, the parameters in the new loading policy that incorporated additional adjacent doors also must be determined. The new loading policy problem is defined as follows:

Assignment doors: Loader's assigned coverage doors. Multiple doors can be assigned to one loader.

Adjacent doors ± 1: Load doors that are next to a loader's assignment doors.

Adjacent doors ± 2: Load doors that are next to a loader's adjacent doors ± 1, exclude assignment doors.

The relationships among assignment doors, adjacent doors ± 1, and adjacent doors ± 2 are illustrated in Figure 4-1 below:

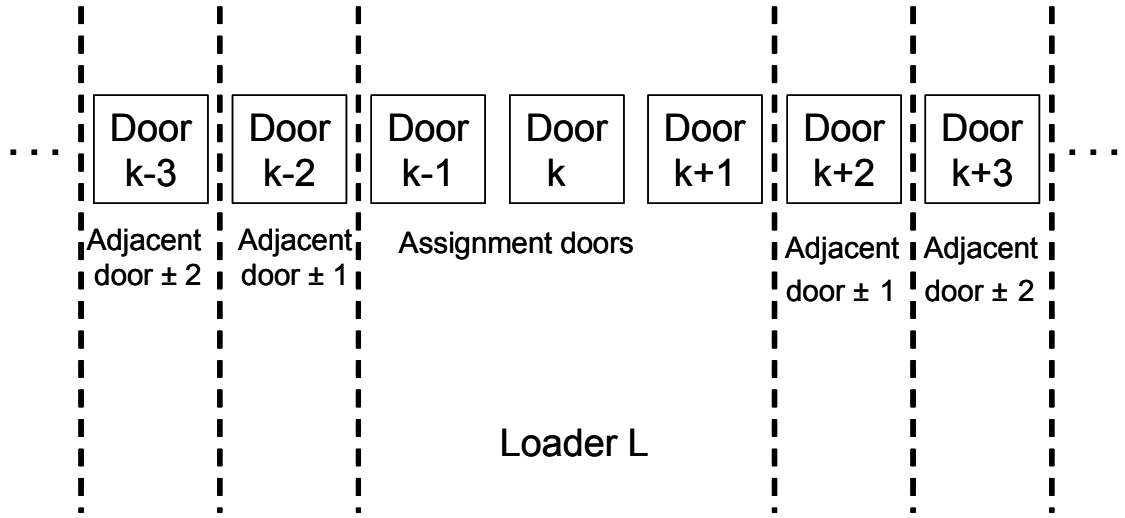

Figure 4—1 Loader-door relationship

In the figure, loader L's assignment doors are *k-1*, *k*, and *k+1*. The adjacent doors ±1 for loader L are doors *k-2* and *k+2*, and adjacent doors ± 2 are doors *k-3* and *k+3*.

Variables

XL_1: Number of packages in adjacent doors ± 1 for loaders to move from assignment doors to adjacent doors ± 1, XL_1 must be an integer.

XL_2: Number of packages in adjacent doors (± 1 or ± 2) for loaders to go back to assignment doors, XL_2 must be an integer.

XL_3: Number of packages in adjacent doors ± 2 for loaders to move from assignment doors to adjacent doors ± 2, XL_1 must be an integer.

Constraints

$XL_1, XL_2, XL_3 \leq 20$ (physical constraint)

$XL_2 < XL_1$ (operating constraint)

$XL_2 < XL_3$ (operating constraint)

Increased adjacent door loading policy

1. For a loader who is working at his assignment door:

   If there are no packages in the loader's assignment door and adjacent doors (± 1) have more than XL_1 packages, go to help.

   If there are no packages in the loader's assignment door and adjacent doors (± 2) have more than XL_3 packages, go to help.

2. For a load door that has two loaders (one is working at assignment door and the other is helping), the loader who is working at an adjacent door (± 1 or ± 2) needs to stop helping and then go back to his assignment door when there are less than XL_2 packages.

For the increased adjacent door loading policy, each loader can have multiple adjacent doors (± 1 and ± 2). In the simulation model, each adjacent door is prioritized for loaders to first search for adjacent doors ± 1 and then adjacent doors ± 2. The pilot simulation utilized for regression meta modeling contained 69 loaders. The samples generated through simulation show that allowing double loading at ± 2 doors can reduce the total time by eight to eleven minutes. The regression model is stated as follows:

$$Y = 4.759 - 0.021XL_1 + 0.046XL_2 - 0.035XL_3 + 0.0018XL_1^2 + 0.0014XL_2^2 + 0.0022XL_3^2 - 0.0025XL_1XL_2 - 0.0016XL_2XL_3 - 0.0004XL_1XL_3 \quad (4\text{–}6)$$

Given the loading policy that loaders can help other loaders only when there are more than two packages in adjacent doors and XL_2 must be less than XL_1 and XL_3, the regression model shows that minimum values occur when we set $XL_1 = 3$, $XL_2 = 2$, $XL_3 = 3$. This regression model was fitted using 860 samples. However, the R^2 is still low (0.346), even though all variables are significant. By simulating the optimal solution of the regression model, the average total processing time is 4:02:24, which is close to the best solution found in current loading policy (4:05:17). Another approach to develop an alternative loading policy is discussed below.

The sortation system has the ability to monitor the number of packages coming to the doors within a given period of time. In the current operations, the only information available to loaders is the number of packages at each of the load doors. An alternative approach is to utilize the information on the number of packages that will be coming into the load doors within a certain time period. The objective is to minimize the makespan by moving loaders among doors more efficiently based on this additional information. The loading policy problem with this additional information is defined as follows:

Assignment doors: Loader's assigned coverage door.

Adjacent doors ± 1: Load doors next to a loader's assigned door.

Adjacent doors ± 2: Load doors next to a loader's adjacent doors ± 1.

Variables

XL_1: Number of packages in adjacent doors ± 1 for loaders to move from assigned door to adjacent doors ± 1, XL_1 must be an integer.

XL_2: Number of packages in adjacent doors (± 1 or ± 2) for loaders to go back to assigned doors, XL_2 must be an integer.

XL_3: Number of packages in adjacent doors ± 2 for loaders to move from assigned door to adjacent doors ± 2, XL_3 must be an integer.

XL_4: Number of incoming packages for adjacent doors (± 1 or ± 2) for loaders to go back to assigned door, XL_4 must be an integer.

XL_5: Number of incoming packages for adjacent doors (± 1 or ± 2) for loaders to move from assigned door to adjacent doors (± 1 or ± 2), XL_5 must be an integer.

Constraints

$XL_1, XL_2, XL_3 \leq 20$ (physical constraint)

$XL_2 < XL_1$ (operating constraint)

$XL_2 < XL_3$ (operating constraint)

$XL_4 < XL_5$ (operating constraint)

Additional information loading policy

1. For loaders who are working at assigned doors with no packages:

   If the number of packages waiting at an adjacent door (± 1) is greater than XL_1 or the number of incoming packages is greater than XL_5, go to help.

   If the number of packages waiting at an adjacent door (± 2) is greater than XL_2 or the number of incoming packages is greater than XL_5, go to help.

2. For loaders who are working at adjacent doors (± 1 or ± 2) and there are two loaders working at the same door. If the number of packages waiting is less than XL_2 and the number of incoming packages is less than XL_4, the loader needs to stop helping and then go back to the assigned door.

The inclusion of variables XL_4 and XL_5 into the loading policy reduces the loader travel time. Because the incoming package information for each door is utilized, a loader will not leave a door that has a large amount of incoming packages to process few packages in adjacent doors. Consequently, the lower loader travel frequency will result in shorter total loader travel time. There are four loading policy problems utilized as test problems for regression meta modeling. These problems are:

a. 84 loaders with adjacent doors ± 1 and ± 2.

b. 84 loaders with adjacent doors ± 1.

c. 69 loaders with adjacent doors ± 1 and ± 2.

d. 69 loaders with adjacent doors ± 1.

The scenarios settings for XL_1 to XL_5 values to simulate regression samples are listed in Tables 5 and 6 below:

Table 5 Scenarios for loading policy problems a. and c.

Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
XL_1	2	2	2	2	2	7	7	7	15	15	15	16	16	16	16	16	16
XL_2	1	1	1	18	18	5	5	5	5	10	10	5	5	5	5	5	5
XL_3	2	2	2	2	2	7	7	7	15	15	21	7	7	7	7	16	16
XL_4	1	1	25	1	25	7	7	19	10	10	15	7	7	19	19	7	7
XL_5	2	30	30	30	30	9	23	23	20	15	20	9	23	9	23	9	23
Scenario	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
XL_1	16	16	16	16	16	16	18	19	20	20	20	20	20	20	20	20	15
XL_2	5	5	14	14	14	14	5	15	1	1	1	1	1	1	1	1	10
XL_3	16	16	16	16	16	16	18	19	2	2	2	2	20	20	20	20	15
XL_4	19	19	7	7	19	19	5	5	1	1	25	25	1	1	25	25	50
XL_5	9	23	9	23	9	23	15	10	2	30	2	30	2	30	2	30	50

Table 6 Scenarios for loading policy problems b. and d.

Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
XL_1	4	4	4	4	4	4	11	11	11	11	11	11	11	11	11	11	11	11
XL_2	2	2	2	2	2	2	2	2	2	2	2	2	9	9	9	9	9	9
XL_4	2	2	2	9	9	15	2	2	2	9	9	15	2	2	2	9	9	15
XL_5	5	13	21	13	21	21	5	13	21	13	21	21	5	13	21	13	21	21
Scenario	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
XL_1	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
XL_2	2	2	2	2	2	2	9	9	9	9	9	9	15	15	15	15	15	15
XL_4	2	2	2	9	9	15	2	2	2	9	9	15	2	2	2	9	9	15
XL_5	5	13	21	13	21	21	5	13	21	13	21	21	5	13	21	13	21	21

Tables 5 and 6 provide the parameter settings for test problems a, b, c, and d. To develop these scenarios, each of the independent variables was first assigned several predefined values and then all possible scenarios were listed. Among these possible scenarios, constraints in loading policies were applied to remove infeasible scenarios.

For scenarios in Tables 5 and 6, a large number of samples were simulated to develop regression meta models for the test problems. The results of regression meta models are listed in Table 7 below:

Table 7 Experiment results of loading policy problems.

Problem	Sample size	R^2	S. D. (minutes)	Significant Value of Regression
a	510	0.841	7.5	< 0.000
b	720	0.691	7.1	< 0.000
c	510	0.630	6.9	< 0.000
d	720	0.574	8.4	< 0.000

Problem	Optimal objective value of regression	Optimal objective value of simulation	Best Sample found
a	3:53:46	4:24:32	4:02:24
b	4:00:40	4:05:02	4:05:02
c	4:12:22	4:34:48	4:19:48
d	4:24:14	4:26:17	4:26:17

In Table 7, the R^2 values are higher than previous regression meta modeling experiments (PFC parameter setting problem). As previous noted, regression models can explain variances in systems better when R^2 values are high. However, as explained below these regression models cannot predict well with these high R^2 values. Among these test problems, regression meta models for problems a and c failed to predict optimal solutions. To further investigate problems a and c, OptQuest models were developed for these two problems. Each of the problems was solved for 60 iterations. The best solutions determined by OptQuest for the two test problems are relatively close to their best regression samples as given in Table 7. For the OptQuest solutions, the makespan times for problems a and c are 4:01:21 and 4:19:10, respectively.

4.3 RESTRICTIONS OF REGRESSION META MODELING

As mentioned earlier, regression meta models are multiple linear regression models that utilize simulation results. Montgomery et al. list reasons why regression coefficients have the wrong sign (fail to predict) [115]. These reasons are: 1. Ranges of some regressors are too small, 2. Important regressors are not included in the model, 3. Multicollinearity is present, and 4. Computational errors have been made.

In this study, there are six regression meta models (as shown in Table 3) constructed for PFC parameter settings and eight for loading policy. These fourteen regression meta models are developed following the same procedure. The ranges of the samples are defined as the ranges of predictive solutions prior to regression meta model fitting. To identify valid regression models, regressors (independent variables) and their interactions are examined using both forward and backward variable selection methods. Among the regression meta models, only two failed to predict primarily because of reason 3; here reasons 1, 2, and 4 are less likely to occur. Multicollinearity, therefore, appear to be the potential cause of regression model failures. The primary sources of multicollinearity were discussed by Montgomery et al. These sources are:

1. Data collection method.
2. Constraints of population/model.
3. An overdefined model (too many variables).

Multicollinearity is the near-linear dependency among independent variables. In the loading policy problem, there are constraints among the independent variables. These constraints can cause dependency among the independent variables. One common indicator to examine multicollinearity is the Variance Inflation Factors (VIF) of regressors [116]. In practice, the regression coefficients are poorly estimated due to multicollinearity if VIF values

exceed ten. To identify the cause of regression meta model failures, the VIF value of each independent variable in loading policy problems a and c were examined. The results show that VIF values for all independent variables in both models are higher than 20. It is concluded that there are near-linear dependencies among independent variables.

In this study, we demonstrated that regression meta modeling can solve certain simulation optimization problems for complex systems by predicting near optimal solutions. A number of experimental results show that regression meta models successfully predict the near optimal settings for the sortation system. Although regression meta modeling can solve complex simulation problems, the methodology is limited to problems with a small number of decision variables that are independent of each other. For problems with combinatorial decision variables, regression meta modeling cannot be applied due to the large number of variables and dependencies among variables. For simulation problems with combinatorial decision variables, validated regression models cannot be built, and other simulation optimization methodologies are required. In the next chapter, search methodologies for simulation optimization are discussed.

4.4 SUMMARY

In this chapter, there were two types of problems solved by the regression meta modeling methodology. The first type is the system parameter setting problem. A variable reduction technique is applied to maintain the predictive ability of the regression models. The second type is the loading policy problem. After identifying the loading policy for the current system design, two further variants of the loading policy problem, increased coverage and additional

information, were investigated. The reasons for the failures of regression modeling cases were identified.

One of the major challenges of regression meta modeling is that the number of independent variables is relatively large for many real world problems. A combinatorial problem with ten binary variables can result in 65 variables where the interactions among variables are considered. The regression cannot predict well when the number of variables is greater than ten. For combinatorial problems, it is common to have potentially hundreds of binary variables.

Another challenge of regression modeling is created by the dependency among independent variables. One assumption in multiple linear regression is that independent variables (regressors) are independent to each other. However, a large portion of real world systems have constraints. Although these constraints can be solved by other techniques, the constraints will lead to near linear dependency among variables. Once complex constraints are introduced into the problems, the multicollinearity will become the major reason that regression fails.

The failure of regression meta modeling demonstrated the needs of developing better simulation optimization methodologies. In the next chapter, a simulation optimization methodology, Surrogate Search, is introduced as a way to solve these problems.

5.0 SURROGATE SEARCH

In this study, a systematic approach called Surrogate Search is developed. Surrogate Search is designed to utilize simulation models to understand system behavior and search for improved system designs. The system behavior observed by analyzing simulation results is formulated into deterministic functions and then utilized as surrogate objective functions to replace the evaluation function, i.e., the simulation model. Because the surrogate objective functions can be rapidly evaluated by search methodologies, a large number of solutions can be evaluated using heuristics prior to making the simulation runs. The best solutions for the surrogate objective functions are then evaluated by simulation to obtain accurately estimations. In addition, for simulation problems utilizing the same model, only one surrogate objective function needs to be identified.

5.1 THE SURROGATE SEARCH ALGORITHM

For simulation models that can be executed relatively quickly, a large number of solutions can be evaluated by existing search methodologies. However, for large-scale simulation models, the number of solutions that can be evaluated is small and existing search methodologies may not be able to find solutions within an acceptable time period. Surrogate Search is developed to solve

simulation problems that require a long computational time to execute the model and contain combinatorial decision variables.

The major strength of using Surrogate Search is that the number of iterations to obtain improved solutions can be reduced compared to other simulation optimization methodologies. The Surrogate Search algorithm is stated as follows:

1. Define

   $\boldsymbol{X}$: Decision variable set.

   Z: Objective function.

   $\boldsymbol{Y_d}$: Dependent variable set

   $\boldsymbol{Y_c}$: A set of dependent variables that are correlated to Z, $\boldsymbol{Y_c} \subset \boldsymbol{Y_d}$.

   Z_s: Surrogate objective function.

2. Simulate a number of scenarios as samples.
3. Search for dependent variables $\boldsymbol{Y_c}$ among $\boldsymbol{Y_d}$ that are correlated to Z.
   a. Determine the relationships among $\boldsymbol{X}$, $\boldsymbol{Y_c}$, Z, and Z_s.
   b. If relationships are found such that Z is proportional to $f(\boldsymbol{Y_c})$ and $\boldsymbol{Y_c}$ is proportional to $g(\boldsymbol{X})$ in the same or opposite directions, define $Z_s = f(\boldsymbol{Y_c})$, $\boldsymbol{Y_c} = g(\boldsymbol{X})$ and go to 5. Else, go to 3.
   c. Determine stopping conditions and initial solutions for local search.
   d. Conduct a local search to find $\boldsymbol{X}$ that can optimize Z_s.
4. Update the local search parameters and execute the local search to determine an improved solution for Z_s.
   a. Evaluate the local search solution by simulation.
   b. If an improved Z is found, record Z and the associated $\boldsymbol{X}$.

c. If the stopping conditions are reached, go to 5. Else, go to 4.

5. Stop and output the best solution Z and $\boldsymbol{X}$.

The stopping conditions in step 4 need to be defined prior to initiating the local search procedure. These can be a predefined number of iterations, predefined computational time length, or a number of non-improving iterations. Note that the local search in step 4 is utilized to identify solutions for a surrogate objective function. Because the surrogate function uses only deterministic parameters, a large amount of solutions can be quickly evaluated by local search prior to the simulation runs. The flow chart of the Surrogate Search algorithm is given in Figure 5-1 below:

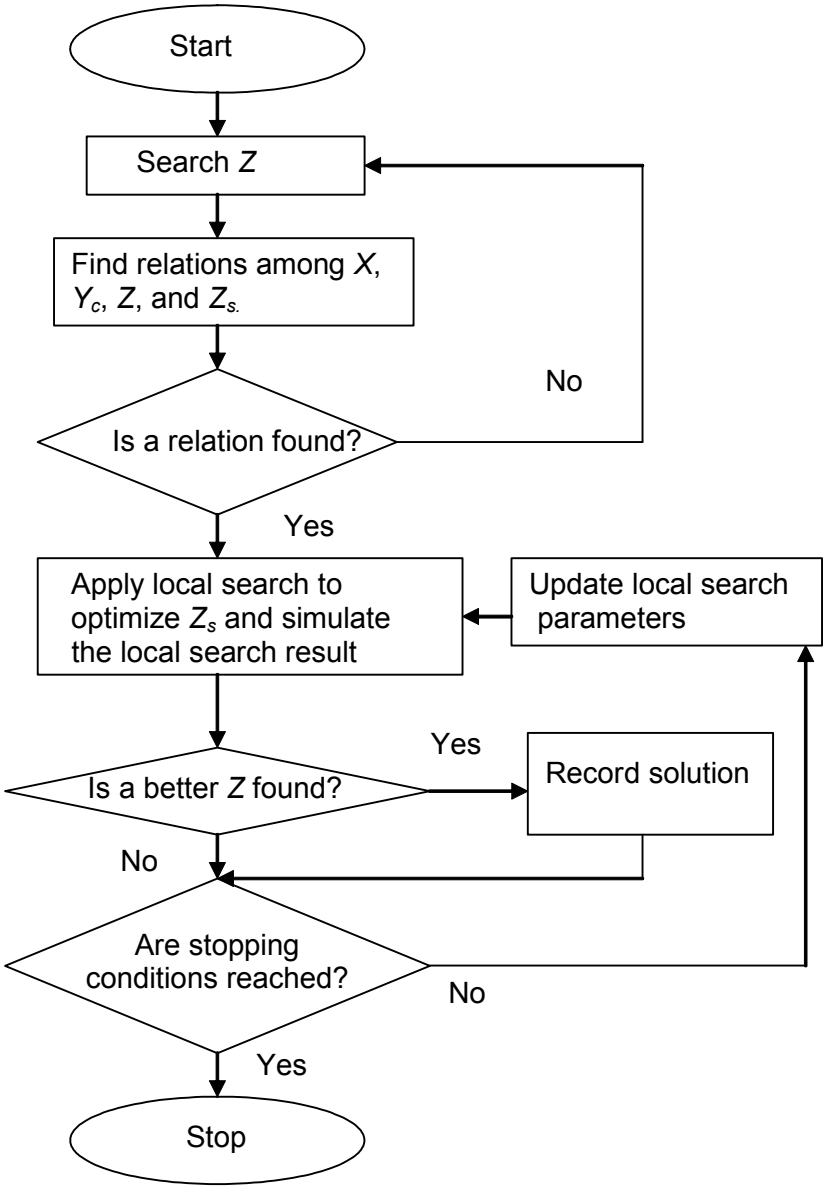

Figure 5—1 Surrogate Search flow chart

It is generally difficult to determine the relationship between the decision (independent) variables and the objective (Z) for simulation optimization problems with combinatorial variables. By observing the simulation results, dependent variables that are related to the objective can be identified. For dependent variables that are related to the objective and for which the relationship to the decision variables can be determined, improving directions can be

found by optimizing those dependent variables (surrogate objective). Consequently, improved objective values will be identified in the process of optimizing the surrogate objective values.

The simulation output provides the information to identify the surrogate objective. The major difference between Surrogate Search and other simulation optimization techniques is that the Surrogate Search observes and understands the system behavior by determining Z_s in step 3, and this occurs before the optimization process. By utilizing simulation results to understand system behavior, improved solutions can be found with fewer simulation runs than other simulation optimization approaches for the problems tested.

Similar to other simulation optimization methodologies, a difficulty in Surrogate Search is maintaining solution feasibility. The local search in step 4 provides a mechanism for maintaining solution feasibility and diversifying the search directions. Assuming that a good Z_s is found, the problem structure is similar to mathematical programming problems. Feasibility constraints are developed in order to determine feasible solutions. Besides maintaining solution feasibility, local search methods also provide the ability to avoid cycling during the optimization process. To illustrate the Surrogate Search, two example simulation problems, production line balancing and an inventory system, are given below:

5.1.1 Surrogate Search example 1: Production line balancing

For multiple station production lines where the objective is to maximize the throughput given a number of resources, the relationship between the number of resources at each station and throughput is unclear. The typical approach is to identify system bottlenecks by finding stations with high resource utilization.

In a multiple station production line simulation model, it is not straightforward to develop the operators' assignments that result in maximum throughput given stochastic system breakdowns, transportation times, and processing times. Instead of using the real objective, maximizing throughput, a surrogate objective would be to balance the capacity in each station by minimizing the utilization deviation from average among stations. That is, to reduce the total difference between utilization at each station and the mean utilization of all stations. After simulating an initial solution, the assignment could be improved by re-assigning operators from lower to higher utilization stations. Near optimal solutions could be identified when assignments that result in less deviation of utilization from the mean are found.

5.1.2 Surrogate Search example 2: Inventory system

Simulation models are also utilized to estimate an inventory system's performance. For inventory systems that are required to satisfy customer demands, the most common objective is to minimize the total cost. The challenge is in identifying relationships among such variables as the reorder point, the order batch size, and the total cost. By listing all components related to the total cost including ordering frequency, size of order, average inventory levels, and frequency of shortage, the contribution to the total cost of each component can be clearly observed.

Instead of using the real objective, minimizing the total cost, a surrogate objective function would be to minimize the difference between shortage cost and holding cost. The relatively low total cost can be obtained by minimizing the surrogate objective.

5.2 EXISTENCE OF SURROGATE OBJECTIVE FUNCTIONS

In general, simulation problems can be represented as several inputs that have to go through a number of processes. An objective value is calculated by using dependent variable values. The general structure of simulation models is shown in Figure 5-2.

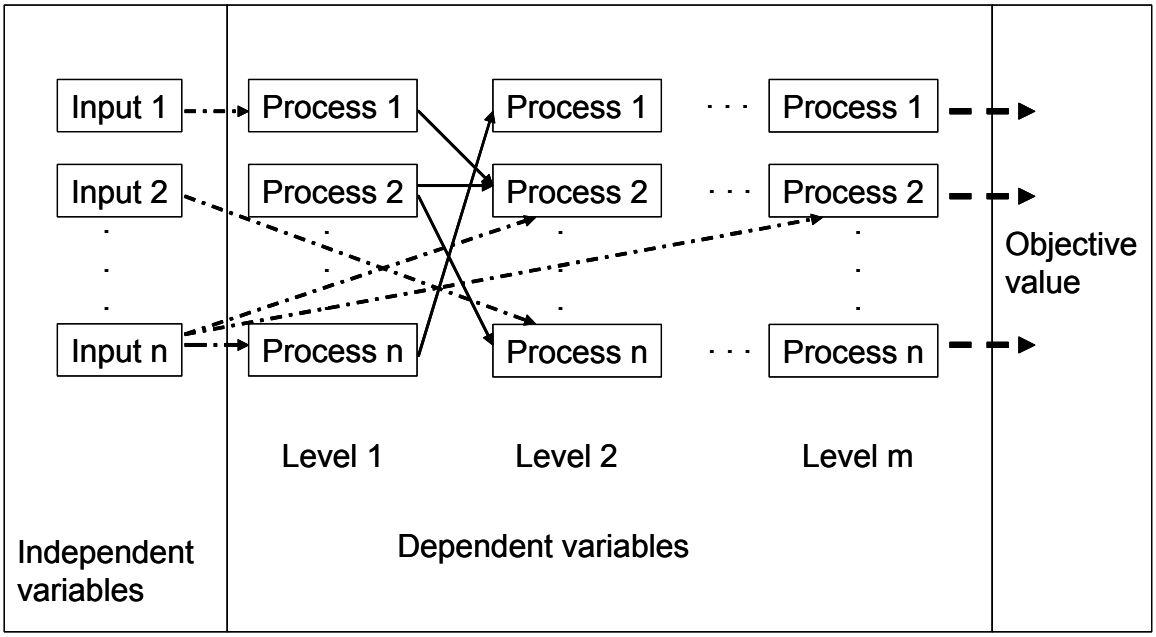

Figure 5—2 General description of simulation models

As shown in Figure 5-2, simulation models contain multiple levels of processes in order to model system events and their interactions. At the end of the simulation run, the objective value is calculated using values from the simulation results. As the number of processes and levels increase, there will be more entities in the simulation model and the computational time to execute the simulation model will become longer.

For any simulation problem, inputs are typically defined as independent variables and when processed each will generate dependent variable values at the end of the simulation. The

objective values are calculated using formulas that contain certain dependent variables. The relationships among objective values, independent, and dependent variables are shown in Figure 5-3.

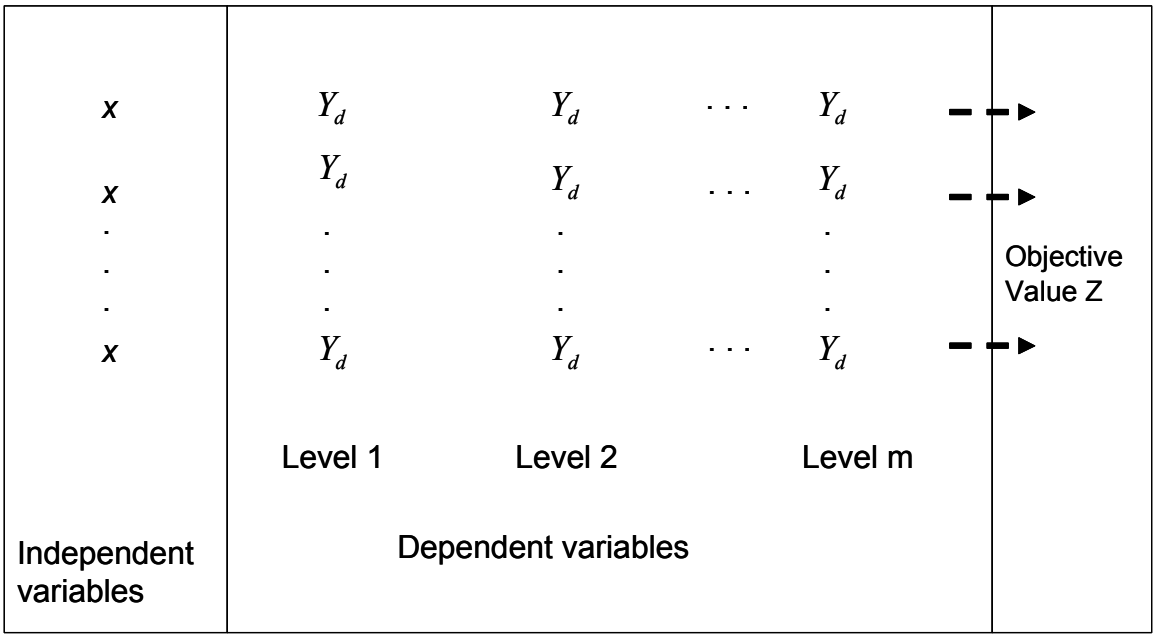

Figure 5—3 Relationships of general simulation models

Figure 5-3 shows the elements of the simulation models. In general, solving simulation problems is defined as determining the optimal or near optimal objective values for simulation models by changing specified independent variables. Because there is no direct relationship between independent variables and objective values, objective values cannot be obtained without running simulation models.

A large volume of results that contain all dependent variables are generated at the end of a simulation run. Instead of discarding these results, Surrogate Search utilizes the large amount of dependent variables ($\boldsymbol{Y_d}$) generated by the simulation model in order to identify the surrogate objective function Z_s. Although the process for identifying a surrogate objective function can be

time consuming, only one surrogate objective function needs to be determined for simulation problems that utilize the same model and different input data.

It cannot be proved that surrogate objective functions can be found for every type of simulation problem. Most simulation problems that do not have a surrogate objective function are the type of problems which the independent variables only have direct impact on objective values. The general structure of simulation models that the independent variables only have direct impact on objective values is illustrated in Figure 5-4 below:

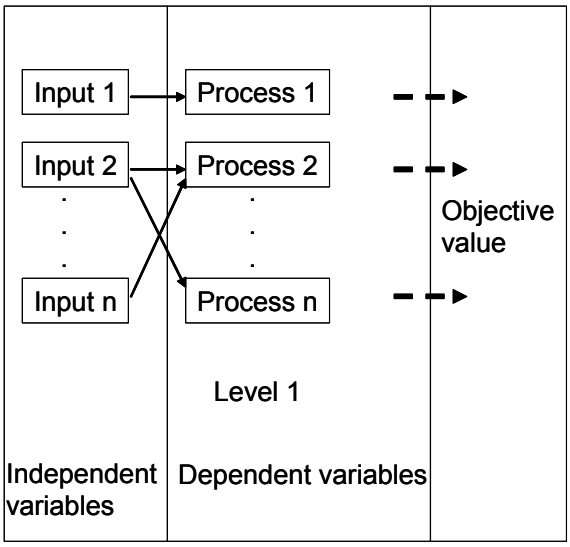

Figure 5—4 Direct impact structure of simulation model

There is only one level of processes existing in the simulation model because the independent variables have direct impact on objective values. That is, the independent variables are utilized by processes that do not relate to other processes and the objective values are calculated using the dependent variables generated by these processes. Although the relationship among independent variables, dependent variables, and objective values can be determined for

this type of simulation problem, the relationship will be a function that only contains independent (input) variables.

The simulation model is directly formulated by multiple random variables and a small number of processes. The structure of these simulation models is relatively simple and can be executed in a short amount of time. Therefore, these problems can be effectively solved by search methodologies without using surrogate objective functions.

5.3 IDENTIFY SURROGATE OBJECTIVE FUNCTION

In simulation optimization, simulation models are utilized to generate an objective value, which is defined as a formula containing one or more dependent variables from the simulation results. Although a large amount of data (dependent variables) are also generated when executing simulation models, existing simulation optimization techniques typically do not utilize them. To identify dependent variables that are correlated to objective values, step 3 of the Surrogate Search procedure is designed to determine relationships among the independent variables, dependent variables, and predefined objective.

To identify surrogate objective functions, multiple linear regression is utilized. As noted in the previous chapter, multiple linear regression is designed to determine relationships between response and independent variables. For Surrogate Search, linear regression can examine variables that are correlated with the objective values and form the surrogate objective functions. The procedure for identifying surrogate functions is as follows.

1. List all dependent variables in set $\boldsymbol{Y_d}$ and objective (Z).
2. Develop regression models using Z as the response.

a. Construct regression models with one variable in $\boldsymbol{Y_d}$ as the independent variable.
b. Construct regression models that contain multiple independent variables. The independent variables can be formulas containing multiple variables in $\boldsymbol{Y_d}$ if physical meanings can be given to these formulas.
c. Construct regression models using backward variable selection.
    i. Insert all variables in $\boldsymbol{Y_d}$ and Z into the regression model as independent and dependent variables, respectively.
    ii. Delete insignificant variables from the regression model one at a time until a validated regression model is constructed.
d. Record validated regression models and insert their independent variables into $\boldsymbol{Y_c}$.

3. Conduct pilot simulations to identify the surrogate objective function (Z_s) among the recorded regression models.

The procedure above describes how multiple linear regression can be utilized to identify surrogate objective functions. For simulation problems, it is possible that validated regression models cannot be constructed due to the problem's complexity. In step 2.a, regression models are developed for individual dependent variables in $\boldsymbol{Y_d}$. If simulation problems are complex and validated models cannot be constructed, step 2.b combines multiple dependent variables in $\boldsymbol{Y_d}$ to further investigate simulation results. If surrogate objective functions still cannot be obtained, it will require additional system expertise to identify the objective function. The process to identify surrogate objective functions using system knowledge is as follows:

1. List and interpret all variables in $\boldsymbol{Y_c}$.
2. Determine impact (increase or decrease) on objective values of changing independent variable values by interviewing system experts.

3. Conduct pilot simulations based on system experts' recommendations.
4. Identify and define the variable to be used as a surrogate objective function to improve objective values.

Due to the large number of dependent variables in simulation results, it is impractical to examine every dependent variable. In the process of developing regression models, a number of independent variables already have been examined to determine their significance levels. Although there is no linear model that can be developed, the significance levels provide a list of candidate dependent variables that can be defined as surrogate objective functions. By interviewing system experts and conducting pilot simulations, the dependent variables and improving directions should be identified to formulate surrogate objective functions. Improving the objective functions is effected by increasing or decreasing surrogate variable values. These directions provide feasible regions with better solution quality.

When a surrogate objective function is determined, the surrogate objective values and objective values will be updated during the process of executing the local search and simulation. Based on problem structures, appropriate heuristics or searching algorithms will be developed for the optimization process. For the optimization process, there are two types of local searches based on feedback from the simulation results. The first type of local search can receive feedback from the simulation results to adjust the surrogate objective functions' coefficients. The second type of local search determines the next iteration solution without analyzing simulation results from the previous iteration.

In terms of local search performance, the first type of local search can determine improved solutions more frequently in earlier iterations of the search. Because simulation results provide feedback for the next iteration's solution by updating the surrogate objective functions'

coefficients, the first type of local search tends to investigate neighborhoods of the current solution. For the second type of local search, surrogate objective functions are utilized without updating coefficients, solutions that result in relatively good surrogate objective values are identified by local searches. Hence, the moving path for the second type of local search does not follow a specified pattern. For the second type of local search, Surrogate Search can be performed on parallel computers. Each computer can execute a Surrogate Search independently by assigning different random number strings because the solution for the next iteration does not require feedback from the current iteration simulation results.

5.4 SURROGATE SEARCH APPLICATION FIELD

Although it cannot be concluded that Surrogate Search has the ability to solve all simulation problems, there are several problem types that have surrogate objective functions. These types of simulation problems are listed in Table 8, which gives the surrogate objectives for general simulation models found in the archived literature.

Table 8 Surrogate objectives of simulation problems

Problem types	Real objective	Surrogate objective
Vehicle assignment problem [117]	Max number of transported loads	Max vehicle utilization/ Min vehicle waiting time
Assignment problem [118]	Min labor cost	Remove low utilization laborers
Inventory problem [119]	Min Cost	Balance holding and shortage costs
Flow shop capacity designs [120]	Max throughput	Balance Machine utilization

For the types of problems listed in Table 8, surrogate objectives can be identified based on knowledge of the system. For the vehicle assignment (e.g., dump truck) problems, if the

objective is to maximize the number of loads in a defined period, the surrogate objective can be to maximize the vehicle's utilization or minimize the vehicle's waiting time. For assignment problems where the objective is to minimize the labor cost, the surrogate objective is to remove laborers (resources) that have low utilization. Although Table 8 does not list all types of simulation problems, one can still consider applying Surrogate Search for other types of problems.

Surrogate Search is designed for large-scale simulation problems that require a relatively long run time, since the process of searching surrogate objective functions is time consuming. For simulation problems that can be run quickly, the process of searching for a surrogate objective may take more time than using other methods to solve these problems.

The surrogate objective function is formed by understanding the system behavior. The behavior can be observed from either the system outputs of the real system or sample simulation runs. The real system should provide the most accurate estimate of the effects of changing parameter settings and system design. However, experimenting with the real system is typically expensive and may not be feasible.

5.5 ASSESSING SURROGATE SEARCH

Two typical measures used to evaluate the performance of optimization and heuristic methodologies are solution quality and computational efficiency. To evaluate the performance of the Surrogate Search approach, the same measures will be used to compare it with other search methodologies.

Solution quality relates to the best solution found by the methodology. For optimization methodologies, LP and MIP provide optimal solutions to the model. Although there are mathematical proofs for some heuristics that they will converge to optimal solutions, obtaining the optimal solutions is constrained by the long computational time in practice. Solution quality of heuristics is taken to be the difference between the best solution found and the optimal solution. If the optimal solution for the problem is not known, approximation techniques that relax certain constraints are often applied. To approximate optimal objective values for simulation problems, deterministic mathematical models can be utilized.

In terms of computational efficiency, the more effective methodologies will find improved solutions with less CPU time. For optimization methodologies, the time complexity analysis (i.e., determine the number of required iterations to obtain the optimal solution) is generally applied. For heuristics, the computational efficiency is the comparison of the number of iterations that will be required to obtain a solution with the same solution quality. Alternatively, the computational efficiency can be the comparison of the solutions' quality for a given number of iterations.

Large-scale simulation problems that cannot be solved by Surrogate Search, have neither clear objective functions that can be written directly nor indicators (from dependent variable values) to determine improved solutions. For these problems, it is obvious that Surrogate Search cannot find improved solutions based on surrogate objective values. Heuristics (and meta heuristics) are the current methods to solve these problems since no knowledge of the system is required. If Surrogate Search fails to replicate system behavior, it cannot be applied; other heuristics (search methodologies) are more appropriate to determine improved system designs for these problems.

For comparison, OptQuest in Arena which utilizes Scatter Search as the underlying algorithm was applied to solve the same problem set in this study. From the pilot simulations, it is found that both Surrogate Search and Scatter Search can identify the solution for the next iteration within a short amount of time (within 30 seconds). In this dissertation, both methodologies utilize the same simulation model that requires two hours (six replications) of CPU time to evaluate one solution, the computational time for both methodologies is referred as the number of iterations (simulation runs).

The test problems in this dissertation are solved by both Surrogate Search and Scatter Search. Note that Scatter Search is given the advantage of executing more iterations than Surrogate Search for all test problems. As noted, the two measures to determine the Surrogate Search performance are the solution quality and the computational efficiency. The solution quality is the best solution found. The computational efficiency is the number of iterations for both methods to identify relative good solutions.

5.6 SUMMARY

In this chapter, the Surrogate Search algorithm was introduced and details were discussed. Surrogate Search is designed to solve large-scale simulation problems effectively. The existence of surrogate objective functions for most general simulation problems was proposed. Simulation structures for which surrogate objective functions cannot be identified were discussed. These simulation models are simple simulation models that can be executed quickly.

A two-step approach to identify surrogate objective functions was investigated. First, multiple linear regression identifies significant dependent variables in the simulation results;

next, system expertise can be utilized. The Surrogate Search application to existing simulation problems in the literature was discussed. Two performance measures, solution quality and computational efficiency, are proposed for evaluating simulation optimization approaches.

6.0 AMHS WORK BALANCING PROBLEM

The sortation system output contains the loading operation described earlier. There are multiple loaders performing the loading operation in different loading areas. To obtain the shortest makespan, loaders must be assigned in a manner that results in balanced workloads. If certain loaders are assigned a heavier workload, they will require longer time to finish their work while other loaders are idle.

6.1 PROBLEM STATEMENT

For the sortation system in this study, there are eight loading areas, each divided into two subareas operated by a team of loaders. Each of the load doors is assigned to a specified loader on the team. For the operating constraints, each loader is assigned between one and three load doors, based on the distance between doors and the projected workload

Even if the work balancing problem is simplified by assuming that the number of loaders in each loading area is fixed, the problem of assigning load doors to loaders is still difficult to solve. For the case where every loading area has the same number of loaders, the number of feasible solutions for a loading area that has 17 load doors and nine loaders is 2,907. This is calculated by first dividing doors by loaders and then identifying the total number of combinations.

For all eight loading areas in the sortation system, the total number of feasible solutions will be the multiplication of the number of feasible solutions in all stations, which is 5.099×10^{27} (i.e., by changing loader assignments in one loading area, the total number of feasible assignments will be 2,907 to the power of eight since there are eight loading areas) in this case. However, the assumption that each loading area has the same number of loaders is unrealistic. It will dramatically increase the problem size to incorporate the number of loaders in each load area as decision variables. To reduce the number of feasible solutions to a solvable level, this chapter utilizes the assumption that a team of loaders will work in a subarea where every load door can be worked by any loaders on the team.

In the case study, the number of loaders ranges between 65 and 75. These loaders are divided into 16 teams. Each team cannot have more than nine or less than three loaders according to operational rules. The objective of the work balancing problem is to determine the number of loaders assigned to each team that result in the shortest makespan.

6.2 PROBLEM COMPLEXITY

To determine the optimal team assignment, one possible approach is to simulate all feasible assignments. The approach can be implemented by starting with a lower bound assignment (e.g., three loaders on each team) and add one loader at a time until all loaders are assigned to a team. For the case where 70 loaders need to be assigned to 16 teams, the lower bound assignment will assign 48 loaders to 16 teams. For the rest of the 22 loaders, there are 16 required scenarios (16 teams) to be simulated in order to determine the optimal assignment each time a loader is added. To find the optimal assignment for the 70 loaders, it will require at least 352 (16×22) scenarios.

If the optimal solution cannot be identified, it may be necessary to simulate all possible scenarios, which would be the worst case in determining the optimal assignment. The total number of feasible solutions does not have a closed form solution due to the constraint that each team has between three to nine loaders. Since each team has at least three loaders, the problem now becomes how to assign 22 loaders to 16 teams. To calculate the number of feasible solutions for the total of 70 loaders assigned to one of the sixteen teams where each team has between three to nine loaders, a C program was written. The result shows there are 6.94×10^9 feasible assignments. If each solution takes one hour (only three replications) to simulate, it would require 792,237 years to simulate all solutions.

The Surrogate Search approach was developed for the AMHS problem which utilizes the sortation system simulation model discussed in Chapter 3. Note the simulation model utilized in Chapters 6 and 7 has different inputs (operator assignments and incoming tasks) then the model in Chapter 4. It would be inappropriate to compare the makespan time in Chapters 6 and 7 to Chapter 4. For the simulation model in Chapters 6 and 7, the CPU time for each simulation replication was approximately 20 minutes. The details of Surrogate Search approaches for AMHS work balancing problem will be discussed in the next section.

Further, a variance reduction technique, antithetic variates, was applied to investigate the possibility of reducing the number of simulation replications for the work balancing problem. The preliminary experimental results showed that the variance can be reduced by assigning antithetic variates to task input sequences. After simulating 600 replications, it could not be demonstrated that antithetic variates reduced the variance by a factor that is greater than the square root of two. Since the variance of the sample mean is divided by the number of

replications (samples), it was concluded that assigning antithetic variates to task input sequences cannot significantly reduce the number of simulation replications.

6.3 SURROGATE SEARCH APPROACH – WORK BALANCING

The objective of the work balancing problem is to minimize the makespan by developing (near) optimal loader assignments. For this problem, the direct relationship between the number of loaders on each team and makespan cannot be determined. While increasing loaders on certain teams can reduce their long queue lengths, other teams might increase the makespan due to lack of loaders. This indicates that the interactions among different team assignments cannot be determined without executing the simulator.

6.3.1 Identify surrogate objective function

To solve the work balancing problem by Surrogate Search, dependent variables that are correlated to the real objective need to be identified. To observe the system behavior, 15 randomly generated assignments were simulated, and the simulation results were fitted as independent variables into a regression model. These input variables are:

J1: Frequency of system breakdowns.

F_k: Package Flow Controllers (PFC) total time for predefined warning level k, k = 1 to 6 (PFC provide six warning levels based on the number of packages on conveyors).

SCF: Sum of total chute full time in sortation system.

LT_i: Number of loaders/chute full time per team coverage, i = 1 to 16.

TL_i: Chute full time/number of loaders per team coverage, i = 1 to 16.

$LTST = \sum_{i=1}^{16} LT_i^2$: Sum of squares of LT_i. This is to determine the deviation among the teams.

$TLST = \sum_{i=1}^{16} TL_i^2$: Sum of squares of TL_i. This is to determine the deviation among teams.

Dependent variable:

Z: Makespan.

Table 9 Regression models of independent and dependent variables

Model #	Independent variables	Dependent variables	Significant level	R^2
1	SCF	Z	0.315	0.017
2	LT_i	Z	<0.000	0.836
3	$TLST$	Z	<0.000	0.811
4	$LTST$	Z	<0.000	0.602
5	TLT, TLT^2	Z	<0.000	0.736
6	$TLST$, $TLST^2$	Z	<0.000	0.842
7	LTT, LTT^2	Z	<0.000	0.716
8	$LTST$, $LTST^2$	Z	<0.000	0.769
9	F_k	Z	<0.000	0.946

From the list of independent and dependent variables in Table 9, there are several factors correlated to the total processing time. Model 9 has the highest R^2 value, and can explain the system variance well. However, the impact to F_k values by having different loader assignments cannot be accurately estimated. Among these variables, a linear relationship between $TLST^2$ (interaction) and Z is found with R^2=0.842. The regression model is $Z = 4.623 + 0.005(TLST^2)$ where the time unit for Z is in hours. The interpretation of this regression model is that good assignments result in short makespan, which can be identified by developing assignments with low $TLST^2$ values. That is, the sum of square of chute full time per loader for 16 teams.

The *TLST* and $TLST^2$ simulation results can determine which sorters need additional help. The chute full time is a simulation output that will not be zero for all teams without having additional operators. The best strategy to obtain low *TLST* values with a given number of loaders in the sortation system is to balance the chute full time among the teams.

For the work balancing problem, $\boldsymbol{X}$, $\boldsymbol{Y_c}$, and Z_s are team assignment, makespan, and PFC chute full warning time respectively. The chute full time per loading team (TL_i) can be interpreted as the time period that a team receives more packages than it can load. A large TL_i of a specified team collected from the PFC indicates that more loaders are needed on that team; it can be reduced by re-assigning loaders from other teams that have smaller TL_i values. To illustrate how the chute full time is related to the makespan, a simplified example is given below:

6.3.1.1 Work balancing problem: Two-team example

Assume that a sortation system contains only one loading area operated by two teams. The total number of loaders is ten and each team has five loaders. After simulating the loader assignment, the following result is found:

Table 10 Initial assignment simulation result

Variable	Team 1	Team 2
Finishing Time	3:15:00	5:01:05
Chute Full time	9:54:10	33:42:00
Number of Loaders	5	5
TL_i	1:58:50	6:44:24

The objective here is to minimize the longer finishing time between Team 1 and 2 because the makespan is ended when the last package is processed. From the simulation result, the objective *Z* is 5:01:05 (Team 2 finishes at time 5:01:05) and Team 1 has 1:46:05 of idle time (after 3:15:00). The chute full time for Team 1 and 2 is 9:54:10 and 33:42:00, respectively. The

TL_i values indicate that each loader in Team 2 has a heavier workload on average compared to Team 1.

According to the first iteration of the Surrogate Search, Team 2 requires additional loaders from Team 1. Since the total workload for each team is fixed, it is clear that sending one loader from Team 1 to Team 2 will reduce the finishing time for Team 2 and increase the finishing time for Team 1. After re-assigning loaders, the new assignment is simulated and the result is given in Table 11:

Table 11 New assignment simulation result

Variable	Team 1	Team 2
Finishing Time	4:11:06	4:32:53
Chute Full time	15:04:07	25:50:40
Number of Loaders	4	6
TL_i	3:46:02	4:18:27

The result showed that the new assignment leads to a smaller makespan (4:32:53). In terms of chute full time and TL_i, the difference between Team 1 and 2 is less than the initial assignment.

If the chute full time were fixed for each team, the loading team assignment would be a simple MIP to solve. However, chute full time is a simulation result (dependent variable) that is not constant and cannot be found until the assignment is simulated. Note that in the two-team example the chute full time does not change linearly with the number of loaders.

6.3.2 Local search approaches

Although chute full time changing directions (increasing or decreasing) for new assignments can be predicted, it is possible that the new assignments for certain teams can move from the upper bound to the lower bound during the optimization process. To prevent cycling, a local search with a predefined neighborhood is necessary for the optimization process because the interactions of chute full time and team assignments cannot be captured without a large number of simulation runs. The chute full time for a small number of simulation replications can be utilized to predict the direction of improvement within the neighborhood. For local search approaches for the work balancing problem, the neighborhood is defined as the current number of loaders on each team ± 1 with the constraint that the total number of loaders is given. The new assignments developed by local search will be simulated in order to observe the chute full time and makespan.

In this study, four local search approaches for the work balancing problem have been researched. Due to the large computational tasks involved, these local searches are implemented to the level where they can be executed automatically. Local searches 1 and 2 utilize MIP and were implemented using Microsoft Excel Solver and local searches 3 and 4 utilize Tabu Search and were implemented using Arena's Visual Basic for Applications (VBA) interface.

Local searches 1 and 2 use the MIP formulation, which can be solved to determine the optimal solution within the neighborhood. The objective of local search 1 is to minimize the sum of the total warning time per loader by removing loaders from certain teams and adding to other teams. The notation is given as follows:

Parameters

n: Total number of teams. There are 16 teams in the case study simulation model.

i: Team index number. $i = 1,\ldots, n$.

TC_i: Total chute full time for team i.

LS_i: Number of loaders in team i.

TL_i: Chute full time per loader for team i. Where $TL_i = \frac{TC_i}{LS_i}$

Decision variables

$X_{i1} = 1$, if team i needs to add one loader; 0, otherwise.

$X_{i2} = 1$, if team i needs to reduce one loader; 0, otherwise.

The objective of local search 1 is to minimize the sum of TL_i for all i. The neighborhood of local search 1 is defined as ±1 from the current solution. The formulation is stated:

$$\text{Minimize} \quad \sum_{i=1}^{n} \frac{TC_i}{LS_i + X_{i1} - X_{i2}} \tag{6–1}$$

$$\text{s. t.} \quad X_{i1} + X_{i2} \le 1, \forall i. \tag{6–2}$$

$$\sum_{i=1}^{n} \left(X_{i1} - X_{i2}\right) = 0 \tag{6–3}$$

$$LS_i + X_{i1} - X_{i2} \ge 3, \forall i. \tag{6–4}$$

$$LS_i + X_{i1} - X_{i2} \le 9, \forall i. \tag{6–5}$$

The number of loaders on one team can vary from three to nine. This is given by the facility's physical constraints. Local search 1 is implemented in Microsoft Excel since the problem structure is relatively simple to solve using Excel Solver.

However, local search 1 is a non-linear problem due to the objective function. In order to reformulate it as a linear problem, local search 2 replaces (6-1) with a new objective function $\sum_{i=1}^{n} W_i$. W_i is the number of loaders for the next iteration for team i divided by TC_i. Although solving the local search 1 formulation by evaluating all feasible solutions is relatively

fast, the linear formulation in local search 2 can be solved by most existing MIP software packages. The formulation of local search 2 is stated below:

$$\text{Minimize} \quad \sum_{i=1}^{n} W_i \tag{6–6}$$

$$\text{s. t.} \quad \frac{LS_i + X_{i1} - X_{i2}}{TC_i} = W_i, \forall i. \tag{6–7}$$

$$X_{i1} + X_{i2} \le 1, \forall i. \tag{6–8}$$

$$\sum_{i=1}^{n} \left(X_{i1} - X_{i2} \right) = 0 \tag{6–9}$$

$$LS_i + X_{i1} - X_{i2} \ge 3, \forall i. \tag{6–10}$$

$$LS_i + X_{i1} - X_{i2} \le 9, \forall i. \tag{6–11}$$

By introducing W_i into the model, the problem has a linear formulation. Note that local search 2 is a minimization problem, which is the same as local search 1. Because assigning too many loaders to one team will result in a small TC_i and a large W_i, minimizing (6-6) in local search 2 can remove loaders from teams that have too many operators and reassign them to teams that require additional help.

The results for local searches 1 and 2 indicate that cycling can occur after a number of iterations since there is no mechanism to prevent it. Local searches 3 and 4 utilize tabu lists to diversify search directions and prevent cycling.

Local searches 3 and 4 analyze the situation that a loading team can request a loader, provide a loader, or keep the same number of loaders. At the end of each iteration, 16 teams are divided into two lists: Type A list (need additional loaders) and type B list (need to reduce loaders). Each team is assigned to one of the two lists based on its TL_i. The Tabu Search restricts a team from adding or removing loaders continuously. For example, if team 15 goes to

seven loaders from six for this iteration, then team 15 cannot move to eight loaders for a specified number of iterations. The local search 3 algorithm is as follows:

Type A list: The list of teams that need help with TL_i in descending order.

Type B list: The list of teams that can provide help with TL_i in ascending order.

1. Calculate the TL_i for each team.
2. Store eight teams with higher TL_i in type A in ascending order.
3. Store eight teams with lower TL_i in type B in descending order.
4. Move loaders by:
    a. If list A or B is empty, stop.
    b. Remove the first team j in list A until a team with less than nine loaders and not tabu from adding loaders is found.
    c. Remove the first team k in list B until a team with more than three loaders and not tabu from removing loaders is found.
    d. If $\frac{TC_j}{LS_j+1}+\frac{TC_k}{LS_k-1}<\frac{TC_j}{LS_j}+\frac{TC_k}{LS_k}$, remove team j and k from type A and B after setting $X_{j1} = 1$ and $X_{k2} = 1$. Go to a.
5. Update the tabu list based on X_{i1} and X_{i2}.

In local searches 1 and 2, one type of constraint is that a team cannot both add and remove loaders ($X_{i1}+X_{i2}\le 1,\forall i.$). In local search 3, the solution feasibility is handled by splitting teams into two groups of lists. Since a team can be on only one of the two lists, the situation that both X_{i1} and X_{i2} are one will never happen. In step 5, the tabu list is updated using the solution for the current iteration. For one of the teams in the sortation, if $X_{i1} = 1$ ($X_{i2} = 1$) for

the current iteration, the tabu list will block the solution that $X_{i1} = 1$ ($X_{i2} = 1$) for a number of iterations.

Local search 3 was first implemented in C. The result showed that these improved solutions are as good as solutions found by local searches 1 and 2. By applying tabu lists, local search 3 provides a number of alternative solutions that have similar makespan times and can be utilized as alternative solutions for adjusting assignment in practice.

Similar to local search 3, the Tabu Search in local search 4 prevents a team from moving back to the previous solution after a move is made. In step 5 of local search 4, the tabu list update is modified as follows:

If a team has $X_{i1} = 1$ ($X_{i2} = 1$) for the current iteration, the tabu list will block the solution $X_{i2} = 1$ ($X_{i1} = 1$) for a number of iterations.

To obtain further diversified solutions by local search, another concept, simulated annealing, that accepts solutions with a specified probability is utilized. In local search 4, the step 4 of the local search 3 algorithm is changed to:

If $\frac{TC_j}{LS_j + 1} + \frac{TC_k}{LS_k - 1} < \frac{TC_j}{LS_j} + \frac{TC_k}{LS_k}$, remove team j and k from type A and B and then set $X_{j1} = 1$ and $X_{k2} = 1$ with a 0.2 probability. Go to a.

The pilot simulation results showed both local searches 3 and 4 could determine improved solutions within a small number of iterations. Once a good surrogate objective for a specified problem type is identified by Surrogate Search, the same surrogate objective can be applied to solve similar problems with different inputs. In the next section, a series of instances is solved by Surrogate Search.

6.4 COMPUTATIONAL RESULTS

A set of instances that includes different numbers of loaders and packages for work balancing problems have been solved by the four local searches discussed above. For comparison purposes, OptQuest models were built to solve the work balancing problem. Scatter Search is implemented in the commercial software OptQuest, which can utilize a simulation package, Arena [121], to calculate objective values.

Figure 6-1 compares the best solutions found by OptQuest and local search 1 for the instance of 69 loaders over 125 iterations (see Table 12). It shows local search 1 can determine better solutions with the same number of iterations executed by OptQuest. In addition, Surrogate Search found solutions that are substantially better than OptQuest in the first 20 iterations. Over these 125 iterations, OptQuest did not identify a solution that is better than the best solution found by local search 1 after the first ten iterations.

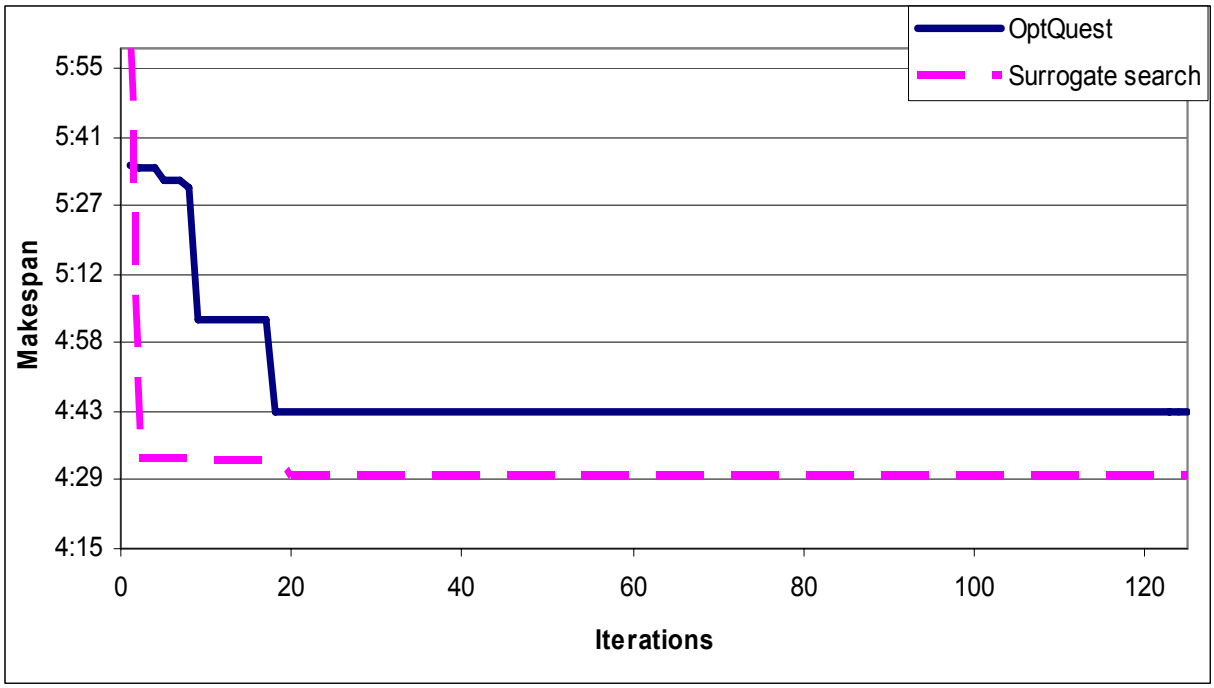

Figure 6—1 Best solutions found by OptQuest and local search 1

To determine the Surrogate Search performance, five instances with different numbers of loaders and package volumes were solved by OptQuest and the four local searches. The number of loaders and packages of test instances are listed in Table 12 belows:

Table 12 Scenario designs for work balancing problem

Instance	Number of loaders	Avg. package volume
1	69	64,800
2	64	64,800
3	74	64,800
4	69	48,600
5	69	81,100

The test instances in Table 12 are designed using different numbers of loaders and average package volumes. The numbers of loaders are 64, 69, and 74. The average package volumes for test instances are 48,600, 64,800, and 81,100.

The five test instances listed by Table 12 were solved by local searches 1 to 4 and OptQuest. Each of the search methods were executed for an average of 97 iterations. The experimental results are given in Table 13 below:

Table 13 Comparison of local search methods and OptQuest.

Instance	**1** (Lower bound 3:38:24)				
Search method	1	2	3	4	OptQuest
Replications	4	4	4	4	4
Number of iterations	95	65	93	93	125
Best solution (hr)	4:31:48	4:29:24	4:30:36	4:30:00	4:43:48
Instance	**2** (Lower bound 3:52:48)				
Search method	1	2	3	4	OptQuest
Replications	4	4	4	4	4
Number of iterations	100	98	89	86	117
Best solution (hr)	4:43:12	4:30:36	4:33:36	4:36:00	5:25:12
Instance	**3** (Lower bound 3:25:48)				
Search method	1	2	3	4	OptQuest
Replications	4	4	4	4	4
Number of iterations	125	43	91	90	131
Best solution (hr)	4:28:48	4:28:48	4:30:00	4:29:24	4:35:24
Instance	**4** (Lower bound 2:52:48)				
Search method	1	2	3	4	OptQuest
Replications	4	4	4	4	4
Number of iterations	124	64	90	88	178
Best solution (hr)	3:34:12	3:34:12	3:33:00	3:34:12	3:35:24
Instance	**5** (Lower bound 4:24:00)				
Search method	1	2	3	4	OptQuest
Replications	4	4	4	4	4
Number of iterations	90	60	90	89	101
Best solution (hr)	5:34:48	5:23:24	5:25:48	5:28:48	5:55:12

In Table 13, the best solutions found by local searches 1 to 4 have less makespan times than the best solutions found by OptQuest. Even with more iterations than the Surrogate Search approaches, OptQuest cannot identify a better solution for all instances. By re-running the best solutions for 20 replications, the largest standard deviation is 4.2 minutes. After performing paired T tests (common random numbers are used with paired replications) to compare the

difference between makespan times determined by Surrogate Search and OptQuest, it was found that the best solutions identified by Surrogate Search are significantly less for instances 1, 2, and 5. This indicates that the Surrogate Search can be utilized as a simulation optimization methodology.

In Table 13, the numbers of iterations vary (from 43 to 178) for each instance solved by the different search methods. Although the ideal strategy to compare performance among different methods is to execute a large number of iteration for each method, the amount of computational time is the major constraint here. The total number of executed iterations in Table 13 is 2,415, which utilized 134 days of CPU time (i.e., 2,415 iterations $\times$ 4 replications $\times$ 20 minutes simulation time per replication.)

Note that the surrogate objective functions for local searches were identified through the pilot simulation. The local searches utilized the same surrogate objective functions to solve the test instances. The results show that the same surrogate objective function for a simulation problem can be utilized to solve different instances of the same simulation model without the need to identify new surrogate objective functions.

Simulation optimization methodologies do not guarantee that the optimal solution will be found because the large amount of possible scenarios cannot be completely simulated. To understand the solution quality of the best solutions found, an approximating approach to calculate lower bounds is utilized. The lower bounds are calculated by a deterministic approach. Assuming all packages for each load door are available at the beginning of the sort and there are no system breakdowns, the loaders have a constant loading rate with a 10 percent estimated time penalty for loaders to travel between doors in the sortation system.

Among the five test instances, the best solution of instance 2 found by Surrogate Search is 55 minutes less than the best solution of OptQuest. To evaluate the performance of Surrogate Search, the best solutions of instance 2 found by both methods is shown in Figure 6-2 below:

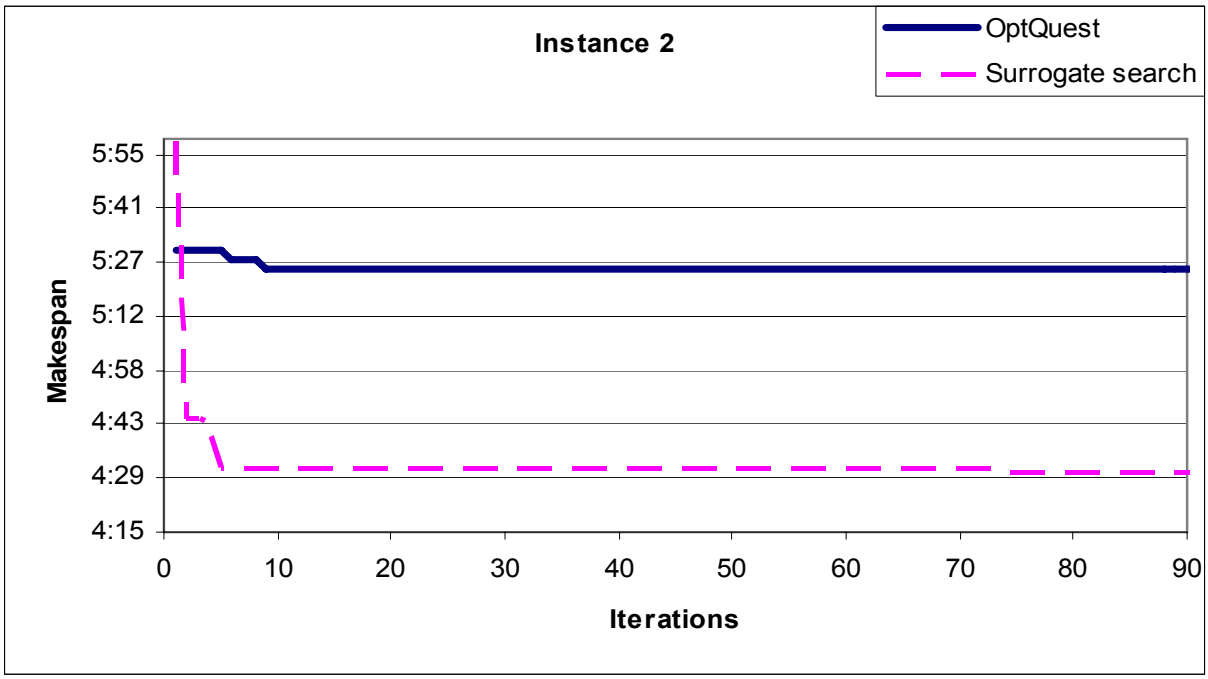

Figure 6—2 Best found solutions of instance 2 (OptQuest and local search 2)

Figure 6-2 compares the best solutions of instance 2 found by OptQuest and Surrogate Search over 90 iterations. It is shown that the best makespans found by Surrogate Search are substantially less than OptQuest solutions after the first five iterations. For instances 3 and 4, the best solutions found by both OptQuest and Surrogate Search are relatively close (differences within 10 minutes). The best found solutions of these two instances are given in Figures 6-3 and 6-4 below:

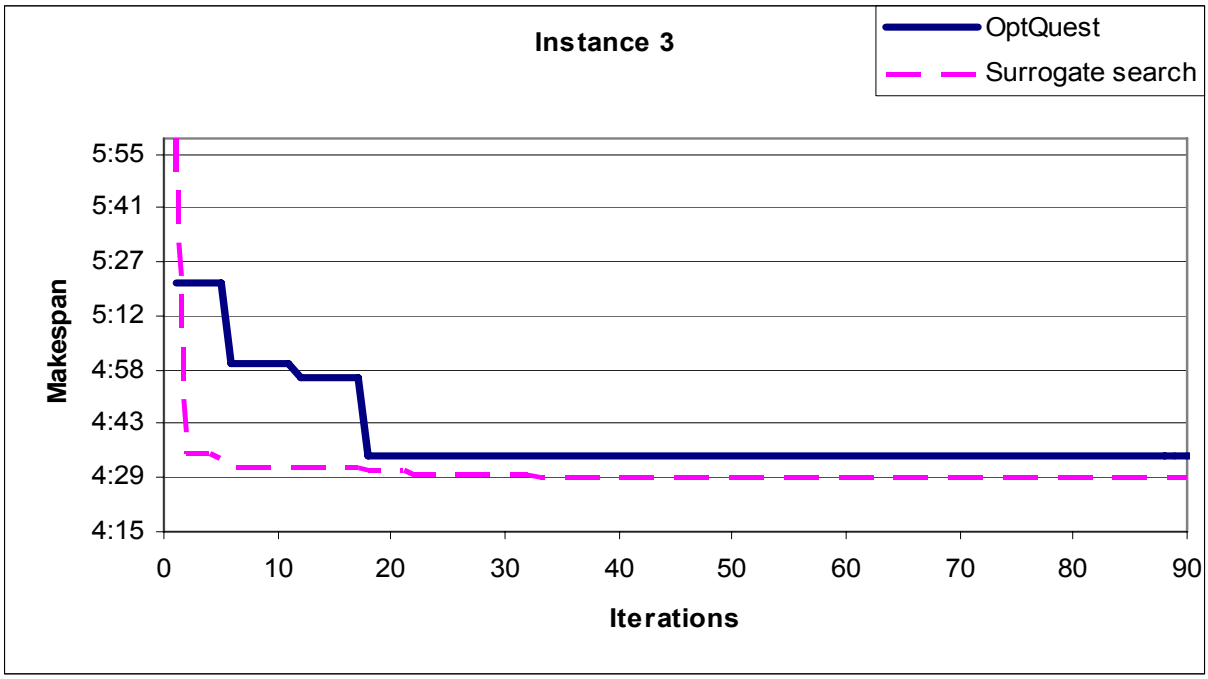

Figure 6—3 Best found solutions of instance 3 (OptQuest and local search 3)

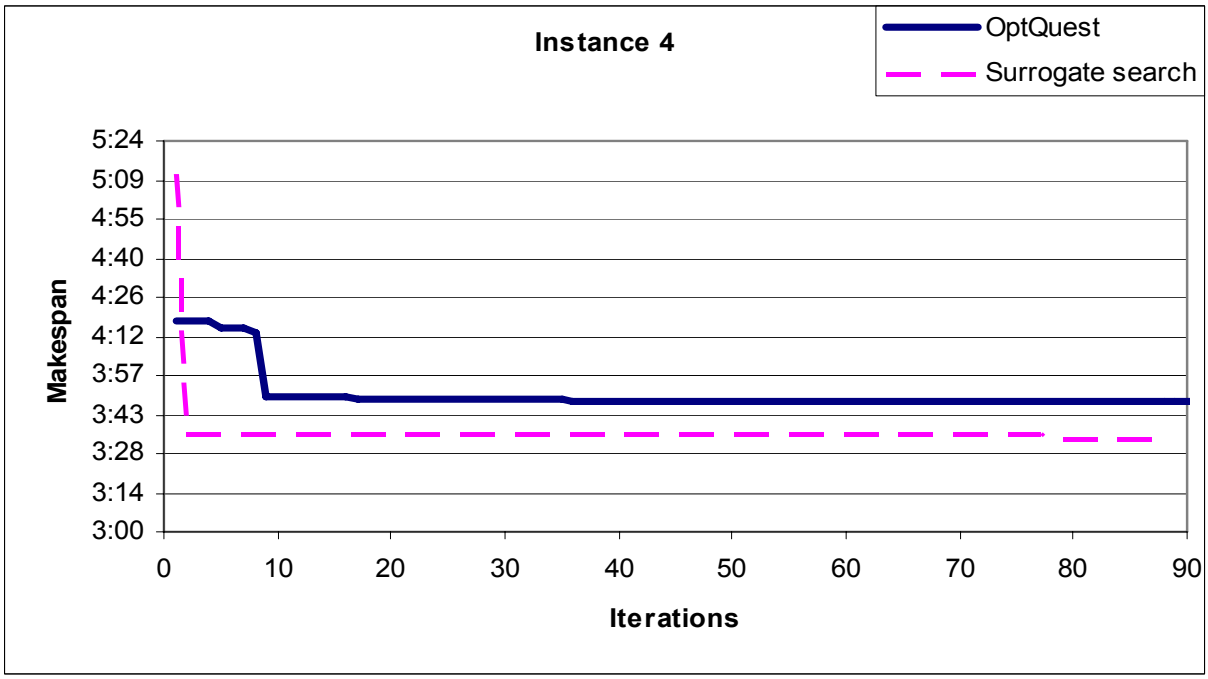

Figure 6—4 Best found solutions of instance 4 (OptQuest and local search 4)

As shown in Figures 6-3 and 6-4, although the surrogate solutions have shorter makespan times, the best solutions found by OptQuest and surrogate become closer as the number of iterations increased. Similar to instance 2, Surrogate Search identified relatively good solutions after the first five iterations for both instances 3 and 4. For OptQuest, it required twenty and ten iterations for instances 3 and 4 respectively to determine relatively good solutions.

6.5 SUMMARY

In this chapter, it is shown that a Surrogate Search can identify improved solutions from empirical experiments. Four local search methods were investigated to solve a relatively complex problem set. The tests indicate that these four methods can result in statistically significant improved solutions over a commercial package using Scatter Search. The fairly straightforward local search methods that utilize the surrogate objective functions have better performance than Scatter Search. Furthermore, local searches are implemented to the level where they can automatically search through the neighborhood.

In summary, Surrogate Search solves simulation problems by utilizing surrogate objectives to find improving directions for solutions in a predefined neighborhood. The major difference between Surrogate Search and other simulation optimization methodologies is that Surrogate Search identifies surrogate objectives by observing the simulation output prior to the optimization process. Once surrogate objectives are identified, similar problems can be solved using these surrogate objectives. The local search methods maintain the solution feasibility and avoid cycling in order to optimize the surrogate objective function. By applying a tabu list (short term memory) in Tabu Search, different search directions can be explored. There is no

additional computational cost to collect surrogate objective values since simulation results contain all required system information.

7.0 TASK INPUT SEQUENCING PROBLEM

In sortation system operations, each incoming task contains from 750 to 900 packages that are assigned to 136 load doors representing different destinations. For each of the incoming tasks, the number of packages assigned to each destination load door has a unique distribution. The package contents information (number of packages that will go to each load door) in each task is defined as Package Destination Distributions (PDDs). Ideally, if the PDDs are the same for all incoming tasks, loader assignments can easily be constructed, and the task input sequences will have no impact on system performance. However, the PDDs for incoming tasks cannot be controlled since they are based on customer demand.

As mentioned in the previous section, loader assignments cannot be easily developed due to system breakdowns and unbalanced workloads. For the system breakdowns, there is no available method that can predict the location and duration of the breakdowns. The major cause of the unbalanced workloads is that the packages in each task have different destinations.

The objective for the task input sequencing problem is to develop a methodology to identify task input sequences that result in short makespans. The task input sequence needs to be determined for every shift in the operation. The current unloading sequence is determined by task priorities (service types). Tasks that have a higher priority are more likely to contain more packages and need to be dispatched earlier, based on the decision maker's experience. By operating the current unloading sequence, the makespan often exceeds the planned time, and the

facility cannot reach its designed productivity. For the sortation system in this study, there are typically 80 to 85 tasks that need to be processed in one shift. Hypothetically, a short makespan can be obtained by utilizing the task content information to develop the task unloading sequence.

7.1 PROBLEM STATEMENT

In the sortation system, since all incoming tasks cannot be processed at the same time they need to be sequentially unloaded. If a large amount of packages for certain load doors are unloaded at the beginning of the operation, the loaders at these specified load doors will receive too many packages while other loaders will be idle. At the end of the operation, loaders that are over utilized at the beginning will be idle since most packages in the incoming tasks are already processed. Consequently, desired makespan times will be extended due to the large amount of idle time.

The objective of the task input sequencing problem is to develop waves of incoming tasks that result in the shortest makespan based on the total number of tasks and their PDDs. A wave is defined as a group of incoming tasks that will be unloaded in the same period of time. It is assumed that the total number of packages contained in each of the incoming tasks follow the same uniform distribution. In the unloading operation, the maximum number of tasks that can be unloaded at the same time is limited by the number of unloaders and operating policies. If 20 tasks are unloaded at the same time (i.e., a wave), there will be four waves of 20 tasks for a total of 80 tasks. The procedure of unloading task waves starts with the first wave. At the beginning, all first wave tasks are unloaded. Once a task is processed, a task in the second wave will be

randomly selected to replace the processed task. After all second wave tasks are selected, the next task that will be selected is in the third wave.

Although the PDDs for all trailers can be obtainable prior to the sort operation, current unloading sequences are not developed based on the task's package content information. The facility's major difficulties in utilizing PDDs include the scale of the computation and the lack of an objective function. To formulate the unloading sequence by MIP with 80 tasks assigned to 80 positions, the formulation can contain 6,400 binary variables (i.e., each of the 80 tasks has 80 binary variables to determine its position). In addition, each binary variable will have 136 parameters (i.e., one parameter for each load door) in the objective function to determine the impact a task will have on all 136 load doors. Even if MIP can solve the large problem, there is no objective function that clearly represents the relationship among unloading sequences and makespan.

The problem of sequencing incoming tasks in sortation facilities was introduced by McWilliams, Stanfield, and Geiger [122] and called "Parcel hub scheduling problem." In their paper, the authors developed a Genetic Algorithm (GA) approach that utilizes a simulation model to evaluate system performance. Similarly, the performance measurement is the time span to process all inbound trailers. Although the paper showed that the GA approach can provide relatively good schedules, as discussed below, there are two weaknesses in the paper: An over-simplified simulation model and limitation of GA in simulation optimization.

In their study, there are a number of assumptions that might lead to inaccurate simulation results. These assumptions are as follows: 1.) Trailers can be instantaneously replaced, 2.) Trailers are processed with equal and constant service rates, and 3.) All inbound trailers are fully

loaded. Although using these assumptions enable simulation models to be executed within a short amount of time, these assumptions are unrealistic for sortation operations.

The major challenge of the GA approach in simulation is the required long computational time to evaluate populations by simulation. In their paper, the population size was set at 50 and the GA was executed for 100 generations. To evaluate all solutions by simulation, there are 5,000 scenarios that need to be simulated. In their study, the computational time to solve one problem is from 20 to 360 minutes depending on the problem size. This relatively low time is because the required simulation time in their paper is relatively short. In contrast, if the CPU time to run one replication using a complex simulation model is 20 minutes, it would require 70 days of CPU time to simulate one replication for 5,000 scenarios.

7.2 PROBLEM COMPLEXITY

To develop a methodology to form the task input sequences that result in short makespans, it is necessary to consider the size of the computational tasks by calculating the number of possible solutions. Assuming that all 84 tasks are available at the beginning of the operation, and there is no constraint to limit the selection of the task unloading sequence, there are two possible approaches: 1) develop the exact task input sequence for 84 tasks, or 2) divide 84 tasks into four waves.

To determine the exact task input sequence of 84 tasks, 21 tasks will be selected for the first wave, and one task will be selected from the remaining tasks once a task is completely unloaded. The total number of the possible sequences is $\frac{84!}{21!63!} \times 63! = 6.4869 \times 10^{106}$ based on this method.

Hence, the 84 tasks are divided into four waves of 21 tasks each. Tasks in the same wave will have the same priority and probability of being processed. If 21 out of 84 tasks are selected as the first wave, the second wave will be formed by selecting 21 tasks out of the remaining 63 tasks. The total number of combinations to develop four waves from 84 tasks will be $\frac{84!}{21!63!} \times \frac{63!}{21!42!} \times \frac{42!}{21!21!} \times \frac{21!}{21!} = 4.8642 \times 10^{47}$. The number of feasible solutions using the wave approach is substantially lower compared to developing the exact task input sequence (6.4869×10^{106}).

Due to the lack of an objective (evaluation) function, meta heuristics and MIP cannot be directly applied to the task input sequencing problem. Hence, the objective function needs to be replaced by a simulation model. Utilizing simulation models as objective functions allow only a relatively small number of scenarios to be searched in the same amount of CPU time compared to deterministic approaches. For the simulation model in this study, each replication requires approximately 20 minutes of CPU time. Consequently, it requires two hours of CPU time to evaluate one solution for six replications using simulation. The simulation model will require 200 hours of CPU time to evaluate 100 solutions. In contrast, deterministic approaches typically evaluate thousands of solutions in one second of CPU time.

Intuitively, the approach that determines the exact unloading sequence can result in a lower objective value with smaller variance due to more control and less randomness in the system. The task wave approach has randomness built in when selecting the next task. However, the large problem size makes the task input sequencing problem extremely difficult even for the four task waves approach. In this study, the task wave approach will be applied to the problem stated at the beginning of this chapter.

7.3 SURROGATE SEARCH APPROACH

To identify surrogate objective functions, a Random Search algorithm is applied to the simulation model in order to generate samples. The algorithm randomly generates feasible task input waves (each task is assigned to one wave and each wave has 21 tasks) and then simulates four replications for each sequence. The average and best makespans of the Random Search results are shown in Figure 7-1 below.

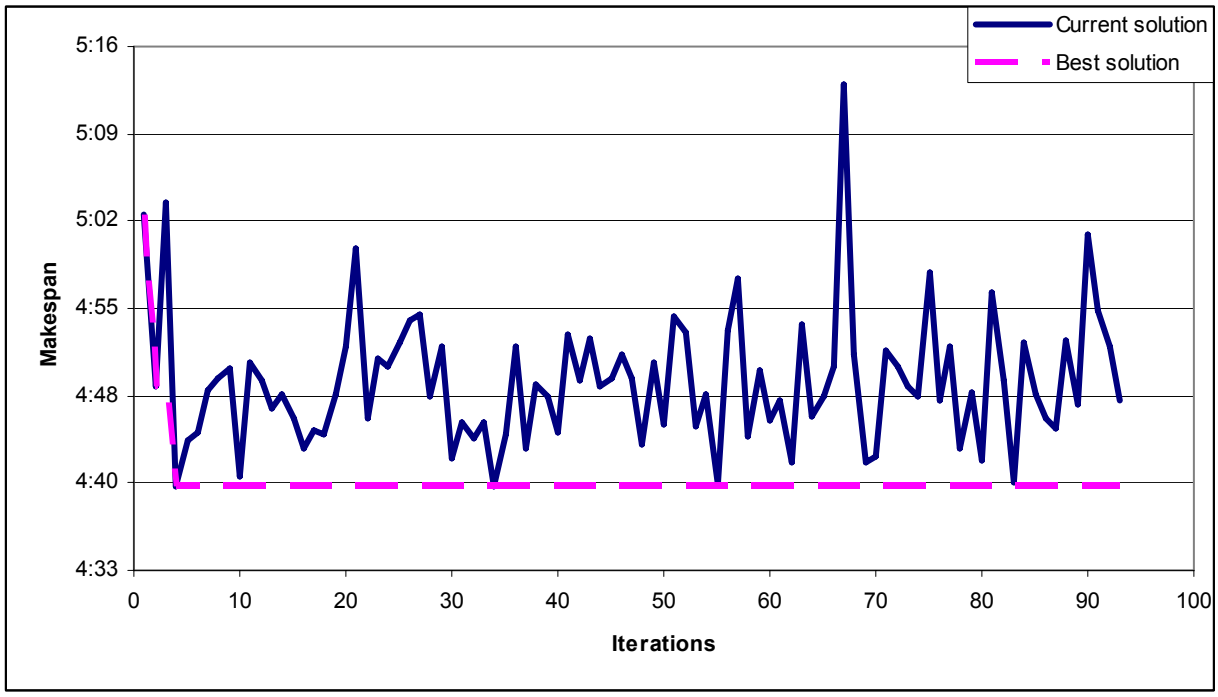

Figure 7—1 Simulation result of Random Search

Figure 7-1 is the Random Search for results of 93 iterations. Although 93 iterations is relatively small compared to the total number of solutions (4.86×10^{47}), the long CPU time required for simulation limits the number of iterations that can be executed. For the Random Search, the CPU time required for 93 iterations in Figure 7-1 is approximately five days. Figure

7-1 shows that by developing task input waves, a 33 minute difference in average makespan between the best and worst task input waves found.

The simulation outputs were analyzed to identify the surrogate objective function. Chute full time and unbalanced package flow for each door were identified as two types of dependent variables based on the data for 372 samples (93 iterations, replicated four times each). To utilize these two as independent variables in the multiple linear regression, the chute full time (CF_i) is divided into four different periods corresponding to each task wave. Unbalanced flow (FL_i) is defined as the difference between actual package flow in each wave and designed loading capacity. Although linear regression models with relatively high R^2 values (from 0.39 to 0.7) were identified, the pilot simulations that utilize these regression models as surrogate objective functions did not show consistent improvements.

In order to collect further samples for constructing surrogate objective functions, three scenarios were simulated. The first scenario utilizes the actual Package Destination Distribution (PDD) collected in the facility with the task input sequences randomly generated. For the second scenario, the PDDs are generated using the average package volumes for all incoming tasks. The third scenario uses the PDDs that match load door capacity. That is, for each task in the third scenario, the portion of packages assigned to each load door is the same as the load door's portion of overall loading capacity. The difference between scenarios 2 and 3 is that scenario 3 has the PDDs match the load door capacity, which is not the same as average package volumes for incoming tasks.

For scenarios 2 and 3, the PDD is the same for every task, and there is no difference when using any task input sequence. The direct impact of using different task input sequences

for the system is the number of packages that will be processed during different time periods of the operation. The simulation results are given in Table 14 below.

Table 14 Simulation results of different PDDs

Makespan	Scenario 1	Scenario 2	Scenario 3
Sample Mean	4:48:58	4:45:47	4:35:06
S. D. (minutes)	9.54	10.26	4.56
Number of Replications	280	30	30

From the simulation output, it is found that the sortation system has the best performance when package flows are the same as designed loading capacities. Although tasks' PDDs cannot be controlled, similar conditions can be obtained by developing task waves that provide package flows close to designed loading capacity. The surrogate objective function is defined as minimizing the difference between the average PDD of task waves and load door designed capacity.

7.3.1 Local search: problem constraints

To optimize the surrogate objective value, Surrogate Search requires local search methods to determine improved solutions. For different local search approaches, the problem structure provides the same types of constraints. The variables and constraints are defined as follows.

Parameters

i: task serial number, i = 1 to 84 (There are 84 tasks assigned to four waves).

j: wave number, j = 1 to 4.

Variable

T_{ij} = 1 if task number i is selected for wave j, otherwise 0.

Constraints

$$\sum_{j=1}^{4} T_{ij} = 1, \forall i \quad \text{(Each task is assigned to one wave)}$$

$$\sum_{i=1}^{84} T_{ij} = 21, \forall j \quad \text{(Each wave has 21 tasks)}$$

For the task input sequencing problem, there are two types of constraints. The first type of constraint assigns each task to one of the four waves, and the second type of constraint requires each wave to contain 21 different tasks.

7.3.2 Local search approaches

In the pilot simulation results, it was found that the sortation system can obtain short makespans when the unloading package flows match the load door designed capacity. The task input sequencing problem can be modeled in a MIP format by minimizing the largest package flow that exceeds load door capacity. Although the MIP formulation cannot handle a situation where certain load doors do not receive sufficient packages, minimizing package flow that exceeds the designed capacity for each of the load doors can reduce the unbalanced workloads. The MIP formulation of the task input sequencing problem is given below:

Parameters:

i: number of tasks, i = 1, 2,…,84.

j: number of groups, j = 1, 2, 3, 4.

k: number of load doors, k = 1, 2,…, 136.

P_{ik} : Portion of packages in task i to door k.

C_k : Designed loading capacity in door k (portion of total loading capacity).

Decision variables:

T_{ij} = 1 if task i is assigned to group j, otherwise 0.

G: the maximum package flow that exceeds designed capacity.

$$\text{Minimize} \quad G \tag{7–1}$$

$$\text{s. t.} \quad \sum_{i=1}^{84} T_{ij} = 21, \forall j \tag{7–2}$$

$$\sum_{j=1}^{4} T_{ij} = 1, \forall i \tag{7–3}$$

$$\sum_{i=1}^{84} (P_{ik} - C_k) T_{ij} \le G, \forall j, k \tag{7–4}$$

In the problem formulation, the variable G is created to measure the maximum overflow load door. By optimizing the problem, task waves that can result in the minimum G can be identified. The advantage of formulating the problem in a MIP format is that the there are existing methodologies that can be utilized to solve it. The weakness of the formulation is that only the largest package flow that exceeds designed capacity will be considered in the problem. The MIP formulation cannot detect load doors that are underutilized. In the study, optimization software, called Cplex 9.0, is utilized to solve the task input sequencing problem.

The pilot experiments contain two test problems utilizing different PDD data sets. The results show that the gap to lower bound is 20 percent for the problem that contains more unbalanced PDD data. For the problem that has more balanced PDD data, the gap is 0.07 percent. The empirical tests show that determining the optimal solutions requires more than one week of CPU time for Cplex. Although the MIP formulation can be solved faster than heuristics, Cplex cannot determine the optimal solutions in a reasonable amount of time. The major difficulty in determining the optimal solution is the large number of feasible solutions for the

problem. In the MIP formulation, each of the four waves gives equal contributions to the objective values. For any feasible solution, there will be 24 solutions that result in the same objective values. The MIP formulation is modified to eliminate the issue of symmetric solutions. There are three types of constraints added into the formulation. These constraints are stated as follows:

Type 1: $\sum_{i=1}^{l} T_{i1} \geq T_{l2}, \forall l = 1,2,...,22$

Type 2: $\sum_{i=1}^{m} T_{i2} \geq T_{m3}, \forall m = 1,2,...,43$

Type 3: $\sum_{i=1}^{n} T_{i3} \geq T_{n4}, \forall n = 1,2,...,64$

Because each of the four waves makes equal contribution to the objective function, a feasible solution can result in 24 symmetric solutions by rearranging the order of waves. These three types of constraints are utilized to form the waves in a certain order. Type 1 constraints force wave 1 to contain more tasks in the first 22 tasks than wave 2. Type 2 constraints force the wave 2 to contain more tasks from the first 43 tasks than wave 3. Finally, type 3 constraints force wave 3 to contain more tasks from the first 64 tasks. By implementing these constraints into the problem formulation, the symmetric solutions generated by of rearranging the order of waves for a feasible solution will result in infeasible solutions.

The symmetric solutions are eliminated by these constraints, and the number of feasible solutions is reduced by a factor of 24. However, the optimal solution still required substantial CPU time (more than 24 hours). The Cplex results were utilized as simulation input data and the result did not show significant improvement over the Random Search solution.

The surrogate objective function, determined by Surrogate Search, is to provide package flows that match the designed loading capacity. In the MIP approach, the objective function is replaced by G in order to maintain the linearity in the problem formulation. The surrogate objective function is a nonlinear function with dependent variables that are identified using simulation outputs. For problems with 336 binary variables, the surrogate objective function requires a large amount of computational efforts to solve it. For the local search approach 1, Cplex cannot solve the optimal solution within 24 hours. For nonlinear objective functions, there is no standard method to determine the optimal solution except to evaluate every feasible solution. In this study, heuristics are developed to identify improved solutions in a short amount of time. For local searches 2 and 3, Random Search and Tabu Search are utilized as underlying algorithms. The formulation to determine the waves for the task input sequences is stated as follows:

Parameters:

i: number of tasks, $i = 1, 2,\ldots,84$.

j: number of groups, $j = 1, 2, 3, 4$.

k: number of load doors, $k = 1, 2,\ldots, 136$.

P_{ik} : Portion of packages in task i to door k.

C_k : Portion of overall designed loading capacity in door k.

Decision variables:

$T_{ij} = 1$ if task i is assigned to group j, otherwise 0.

$$\text{Minimize} \qquad \sum_{k=1}^{136}\sum_{j=1}^{4}\left(\sum_{i=1}^{84}(P_{ik}-C_k)T_{ij}\right)^2 \tag{7–5}$$

$$\text{s. t.} \qquad \sum_{i=1}^{84} T_{ij} = 21, \forall j \tag{7–6}$$

$$\sum_{j=1}^{4} T_{ij} = 1, \forall i \qquad (7\text{–}7)$$

The surrogate objective function minimizes the deviation of package flows provided by task waves and designed loading capacity. Although the square term in the objective function could be replaced by an absolute value function, that would make it difficult to solve the problem.

For the problem of minimizing the deviation of the difference between package flows and designed capacity, there is no efficient methodology that can be applied. Local search 2 utilizes a Random Search algorithm to determine improved solutions of task waves. Because package flows are the results of task waves, the surrogate objective value can be identified when the waves are constructed. Local search 2 repeats the process of randomly generating and evaluating task waves. The algorithm for local search 2 is stated as follows.

1. Randomly generate a number of task waves and calculate the objective values using a surrogate objective function.
2. Simulate the solution associated with the best objective value in step 1 for a specified number of replications.
3. If a better makespan is found through simulation, record it.
4. Repeat steps 1 to 3 until the maximum number of iterations is reached.

The Random Search algorithm is utilized to generate diversified solutions to explore different search directions. Since the surrogate objective function can be evaluated deterministically, thousands of solutions can be evaluated before the simulation model is executed. After a predefined number of solutions are evaluated, the performance of the best task waves will be determined by simulation. The moving path for local search 2 is given in Figure 7-2 below:

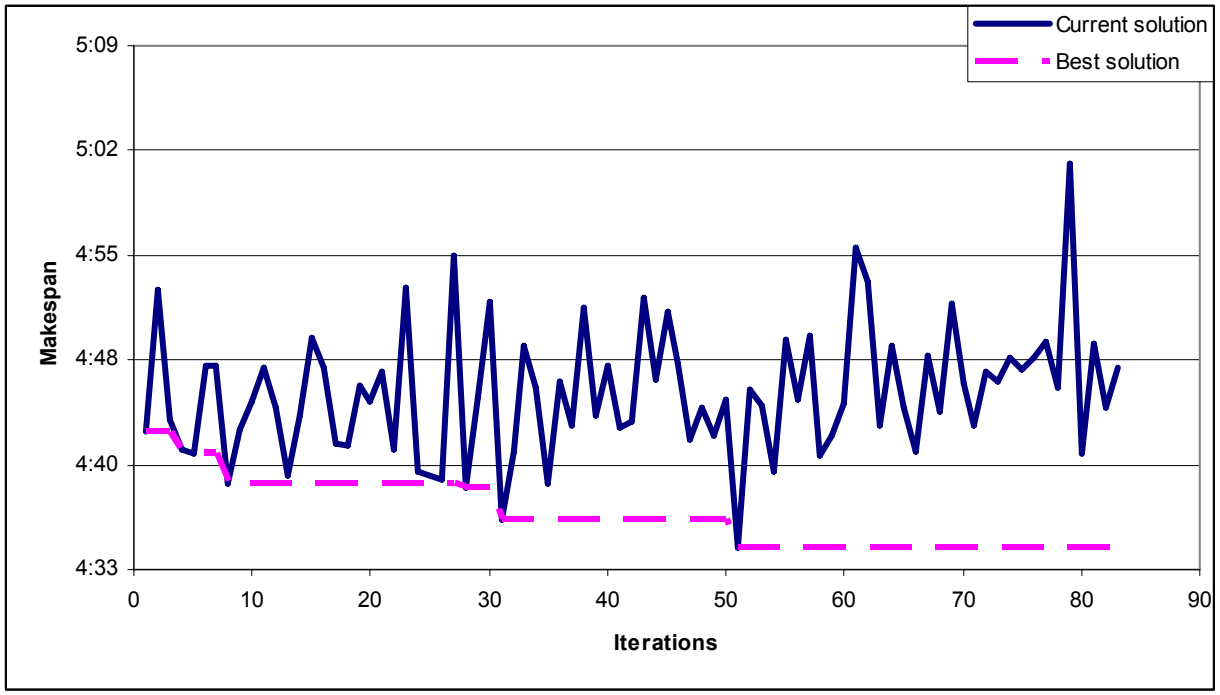

Figure 7—2 Local search 2 moving path

In pilot simulation runs, three variants were tested. These variants evaluate 1,000, 5,000, and 10,000 random solutions in step 1. Among these three variants, the best solutions and average solution qualities are very similar (the difference is within three minutes). One issue with local search 2 is that similar solutions were found once we increased the number of iterations. It is possible that the Random Search can get trapped in local optimums and cannot reach better solutions. Because the Random Search algorithm in local search 2 does not provide a mechanism to break away from the local optimum, the Random Search will be stopped at local optimums and cannot be improved. In local search 3, the mechanism of short term memory in Tabu Search is implemented in the algorithm to identify improved solutions. The algorithm is listed below.

Step 0. Define the maximum number of iterations, and the number of replications.

1. Randomly generate a number of solutions with wave 1, 2, 3, and 4.

2. Calculate objective function values for solutions in step 1 using the surrogate objective function.
3. Apply Tabu Search to the best solution found in step 2 to improve solution quality.
4. Select the solution with the best objective value in step 3 and simulate the solution for a specified number of replications.
5. If a better makespan is found in the simulation result, record it.
6. Repeat steps 1 to 5 until the maximum number of iterations is reached.

In local search 3, steps 1 and 2 that randomly generate feasible solutions are the same as for local search 2. In step 3, Tabu Search is utilized to improve the best solutions found by Random Search.

In the local search 3 pilot simulations, several settings for the Random Search and Tabu Search iterations in steps 1 to 3 were tested. The results show that local search 3 can identify task waves with short makespan times in the first 20 iterations. The moving path of local search 3 is shown in Figure 7-3 below:

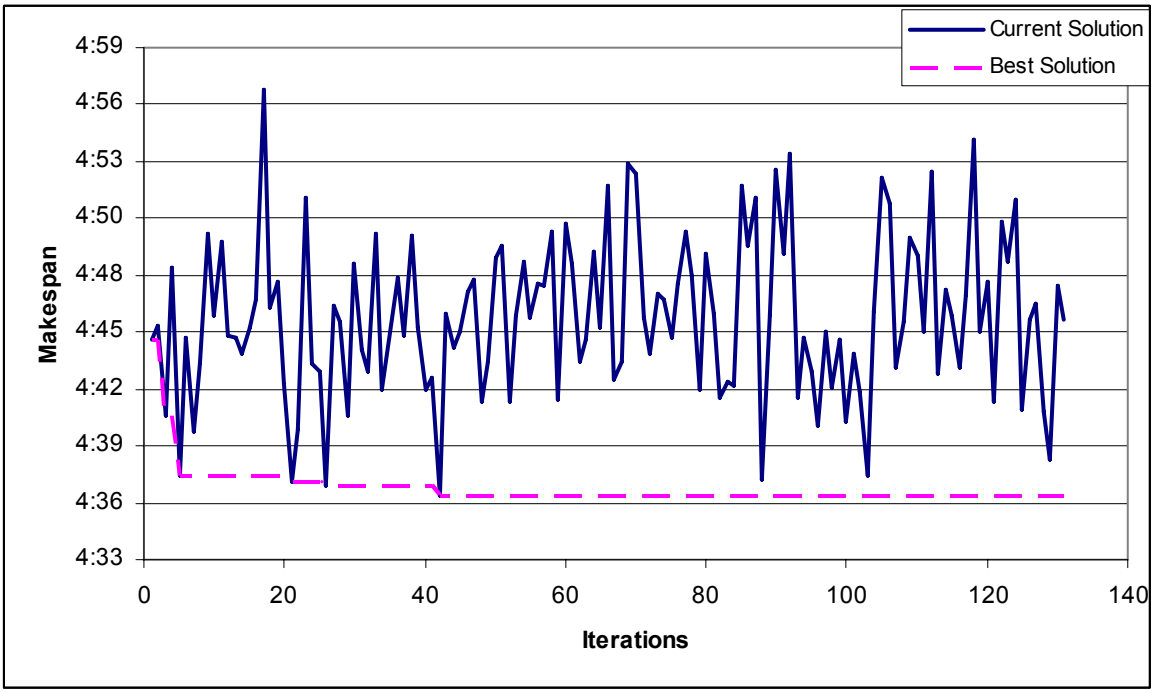

Figure 7—3 Moving path of local search 3

Figure 7-3 is the pilot simulation results for local search 3. Because local search 3 calculates the surrogate objective values prior to simulation executions and does not collect feedback from the simulation results, the moving path for local search 3 does not follow any pattern. For Surrogate Search approaches that receive simulation results as feedback, their moving paths will show more improved solutions more frequently.

In the pilot simulations, local search 3 identified task waves that led to an average makespan time of 4:36:36 when the number of Tabu Search iterations (prior to simulation runs) is greater than 100. The best and average makespan times are both less than those for local searches 1 and 2. The advantage of using the unbalanced flows as the surrogate objective function is that the objective values can be determined prior to simulation runs.

The major difference between local search 1 and local searches 2 and 3 is that local search 1 utilizes MIP to solve the problem and optimal solutions can be obtained. For local

searches 2 and 3, heuristics are utilized to determine improved solutions and there is no guarantee of the solutions' optimality. Although local searches 2 and 3 cannot determine optimal solutions, the near optimal solutions provided by local searches 2 and 3 can be determined within 300 iterations using the surrogate objective function, which requires less than 3 minutes of CPU time. In contrast, it would require longer than 24 hours to determine the optimal solutions by local search 1.

In the pilot simulation results for the three local searches, the best solution found is seven minutes less than the best solution found by randomly generating task input sequences. In addition, the average makespan time for multiple iterations by these three local searches are lower than randomly generated sequences. This suggests that the Surrogate Search can identify solution regions that have better solution quality, and improved solutions can be identified faster.

One remaining issue of the task input sequencing problem is that there are 24 symmetric solutions for any feasible solution due to the symmetric structure in the problem. Although these 24 solutions have the same objective values for surrogate objective function, the makespan times cannot be obtained until the simulation is executed. The estimated computational time to simulate 24 scenarios is 48 hours. In this study, the local searches utilize the strategy that only one solution will be simulated in order to explore more diversified solutions in the limited time.

7.3.3 Scatter Search approach

To evaluate the Surrogate Search performance, the task input sequencing problem is also solved by Scatter Search. As noted, Scatter Search is implemented in the software package OptQuest in Arena. For this study, the OptQuest model is developed for the task input sequencing problem given the same types of feasibility constraints. Each scenario is simulated for multiple

replications (four to six), and the average makespan is defined as the objective value. The example of the Scatter Search moving path is given in Figure 7-4 below.

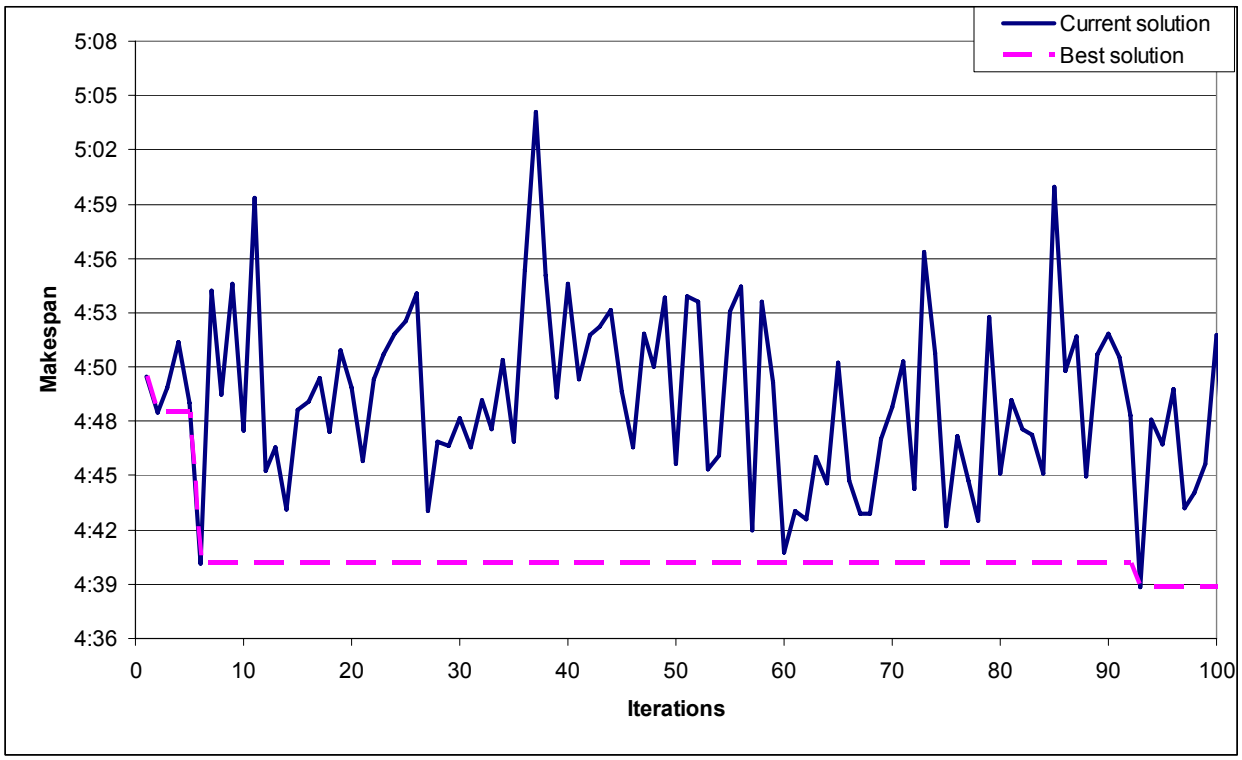

Figure 7—4 Scatter Search example

From the OptQuest results, the best found solution is 4:39:00 for the makespan. This is slightly longer than local searches 2 and 3. To compare the performance among local searches and OptQuest, a series of task input sequencing problems was designed. In the next section, the experimental results are discussed.

7.4 COMPUTATIONAL RESULTS

To evaluate Surrogate Search performance, multiple input data sets (PDDs) are utilized in test problems. For this study, there is only one PDD set collected from the facility. Although more data sets could be collected in the sortation system, the facility currently does not have an efficient method to collect PDD data sets. In addition, the data collected by the facility over a short time period (days or weeks) are similar and would not provide a variety of input ranges to test the various possibilities. In order to evaluate the Surrogate Search performance under different input data ranges, four additional problems were artificially generated for experiments.

To generate task PDDs for the experiment, the total package volumes for all load doors in the collected data were first calculated. A simulator was developed to generate task PDDs in this study. The packages for each of the load doors are randomly re-assigned to a specified number of tasks. For example, if the total number of packages for a load door is 1,000, these 1,000 packages would be re-assigned to five tasks with equal probability. The number of tasks that packages for a load door will be re-assigned determines the ranges of the input data. If the packages for a load door can only be re-assigned to five tasks, the PDD for each of the tasks will be relatively unbalanced compared to re-assigning packages to all tasks (84 tasks). In the study, four sets of task PDDs were generated by assigning packages in each of the load doors to 5, 20, 30, and 60 tasks. To guarantee the feasibility of the generated data, no more packages would be re-assigned to a trailer once it reaches the maximum package capacity. This decision rule is programmed into the PDD simulator.

In the experiment, there are five sets of task PDDs utilized to evaluate Surrogate Search performance. To further test the sortation system performance, there are two package volumes, the current package volumes and 125 percent package volumes, utilized in the test problems.

Hence, there are ten testing problems (five PDD sets and two package volumes). The method of generating PDDs ensures the feasibility and variety of the generated PDD sets. The settings of these ten testing scenarios are listed in Table 15 below:

Table 15 Scenario settings for experiment

Scenario #	**1**	**2**	**3**	**4**	**5**
PDD data	Actual data	Generated #1	Generated #2	Generated #3	Generated #4
Package Volume	65,000	65,000	65,000	65,000	65,000
Scenario #	**6**	**7**	**8**	**9**	**10**
PDD data	Actual data	Generated #1	Generated #2	Generated #3	Generated #4
Package Volume	84,000	84,000	84,000	84,000	84,000

From the pilot simulations for the four local searches, it is found that local search 3 can identify improved solutions faster than the other two local searches. Although they also determine improved solutions, local searches 1 and 2 cannot be easily applied. For local search 1, the required CPU time to solve the MIP formulation is longer than one day. For local search 2, cycling problems can occur due to the lack of a mechanism to break away from local optima. Hence, for the experiments in this study, Surrogate Search will utilize only local search 3, which combines Random Search and Tabu Search.

The most problematic element of the experiment is the long CPU time to execute Surrogate Search. For the experiment, each of the ten test problems is solved by Surrogate Search for 100 iterations. For the problem with a large number of feasible solutions (4.68×10^{47}), 100 iterations is relatively small. However, the simulation model in this study requires 20 minutes to execute one replication, and there are multiple (four to six) simulation replications executed for one iteration. The CPU time to execute 100 iterations using Surrogate Search is longer than eight days. In addition, Scatter Search is also utilized to solve these ten scenarios

with the advantage of executing more iterations than Surrogate Search. The advantage of executing more iterations is that additional solution regions can be explored, and there is a greater chance of finding a better solution. The tradeoff is that more CPU time is required due to the additional iterations. The computational results are listed in Table 16 below:

Table 16 Computational results of ten scenarios

Scenario #	**1**		**2**		**3**		**4**	
Search method	Surrogate	Scatter	Surrogate	Scatter	Surrogate	Scatter	Surrogate	Scatter
Number of iterations	75	88	126	229	101	139	101	141
Replications	6	6	4	4	4	4	4	4
Makespan (Avg)	4:45:36	4:46:48	5:04:48	5:19:12	4:49:12	4:52:48	4:46:48	4:49:12
Makespan (Best)	4:36:36	4:39:36	4:53:24	4:58:12	4:38:24	4:40:12	4:34:12	4:38:24
Scenario #	**5**		**6**		**7**		**8**	
Search method	Surrogate	Scatter	Surrogate	Scatter	Surrogate	Scatter	Surrogate	Scatter
Number of iterations	105	145	60	70	137	303	80	107
Replications	4	4	6	6	4	4	4	4
Makespan (Avg)	4:45:00	4:45:00	5:42:36	5:46:48	6:20:24	6:36:36	5:51:36	5:57:00
Makespan (Best)	4:35:24	4:35:24	5:33:00	5:34:12	6:06:00	6:13:12	5:37:48	5:42:36
Scenario #	**9**		**10**					
Search method	Surrogate	Scatter	Surrogate	Scatter				
Number of iterations	80	109	100	110				
Replications	4	4	4	4				
Makespan (Avg)	5:46:48	5:50:24	5:42:36	5:45:00				
Makespan (Best)	5:33:36	5:36:00	5:33:00	5:34:48				

For the computational results, the average makespan is defined as the average value of all solutions found by the search methods. For the search methodologies, the performance measurement is typically focused on the best solution found when a large number of iterations are executed. In this study, the largest number of iterations executed by Scatter Search is 303 (scenario 7). The average makespan can indicate the solution quality of the search methods.

In Table 16, Surrogate Search found better solutions for nine of the ten scenarios for both average and best makespans. Of these ten scenarios, eight of them utilize PDDs that were artificially generated. The advantage of generating PDDs is that the level of the unbalanced tasks can be controlled as input data. For the test problems with the most unbalanced input data (scenarios 2 and 7), simulation results indicate that the largest difference in makespan occurred between Surrogate and Scatter Search. The solutions for scenario 2 and 7 found by Surrogate Search are 14.4 and 16.2 minutes less, respectively, compared to the best solutions found by Scatter Search. For scenario 5, which has the most balanced input data, the difference between makespan times found by the two search methods is less than 0.5 minutes.

Another measurement to evaluate the Surrogate Search performance is the best solutions found during the search process, which can be utilized as an indicator of the speed of identifying improved solutions. If the best solutions found by a search method show that near optimal solutions can be found in the first 20 iterations, similar types of problems can be solved for a small number of iterations using the same method. The best solutions found by Surrogate Search and Scatter Search within the first 100 iterations for scenario 2 (most unbalanced) is shown in Figure 7-5 below:

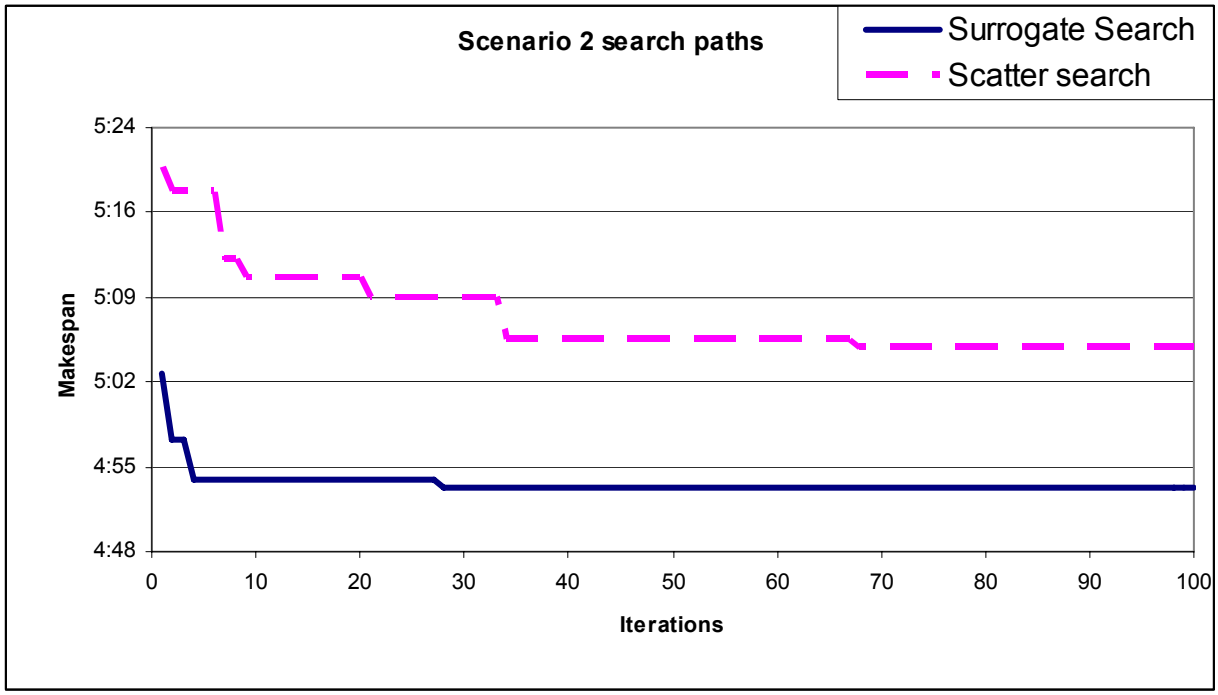

Figure 7—5 Best found solutions of scenario 2

Figure7-5 illustrates that Surrogate Search quickly found better solutions than Scatter Search solutions as indicated by the best solution paths. The best makespan times found by Surrogate Search is 4:54:00, while the best solutions found by Scatter Search is close to 5:06:00 hours. In addition, solutions found by Surrogate Search have makespan times that are close to 4:54:00 hours in the first 10 iterations. For scenario 5, which has the most balanced PDDs, the best solutions of surrogate and Scatter Search have similar makespan times. The scenarios 5 best found solutions are shown in Figures 7-6. Figure 7-6 shows that the best solutions for scenario 5 found by both surrogate and Scatter Search are similar after 80 iterations. Compared to the best solutions found in the earlier iterations, Surrogate Search reached solutions that have shorter makespan times than Scatter Search solutions after 20 iterations. In addition, Scatter Search did not identify solutions better than Surrogate Search solutions between iterations 20 and 75.

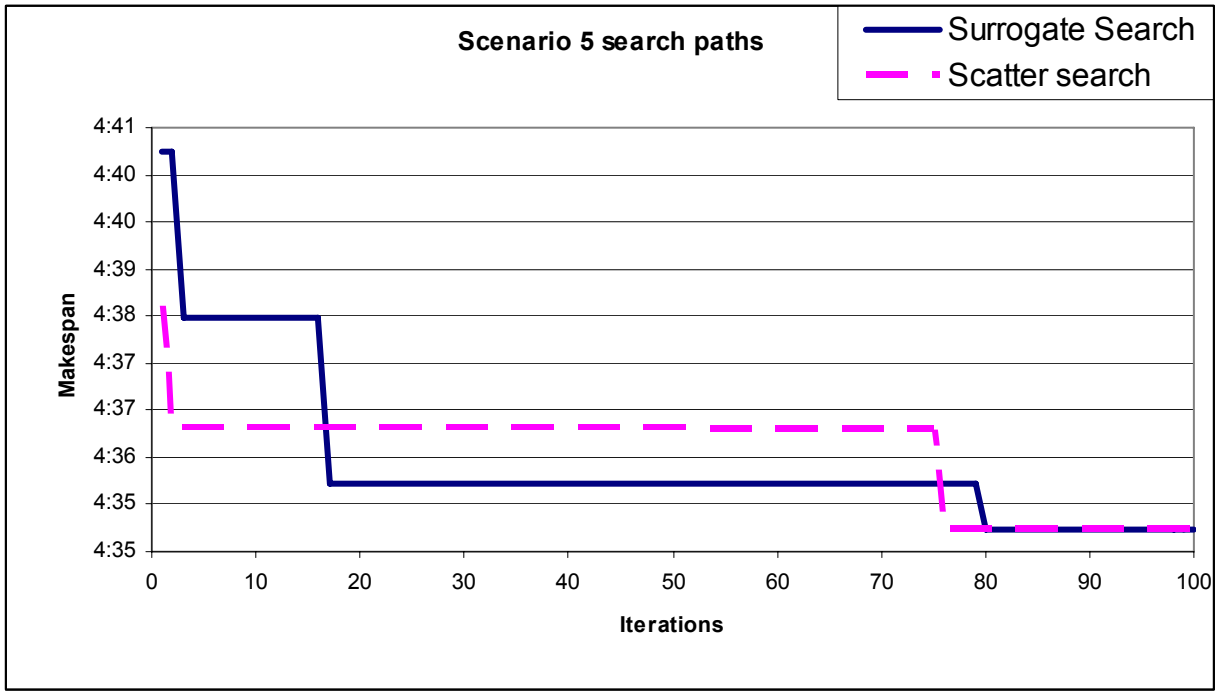

Figure 7—6 Best found solutions of scenario 5

The major difference between scenarios 2 and 5 is the tasks' PDDs. The task PDDs for scenario 2, which is the most unbalanced, is generated by re-assigning packages to five tasks. The task PDDs in scenario 5, which is the most balanced, are generated by re-assigning packages to 60 tasks. For the task input sequencing problem, Surrogate Search performed better than Scatter Search when the task PDDs were more unbalanced.

The sortation system processes all tasks in a short amount of time when all load doors can receive packages in proportion to their designed loading capacities. When task PDDs are more balanced, it is easier to develop task waves that result in shorter makespan times. A large portion of task waves that are arbitrarily generated by Scatter Search can have short makespan times because the balanced PDDs can provide package flows that are closer to the load doors' designed capacities. If task PDDs are extremely unbalanced, most of the feasible task waves will result in longer makespan times. Further, Scatter Search requires more iterations to determine solutions

that have the same makespan times as solutions found by Surrogate Search. The surrogate objective function can determine task waves that result in short makespan times by providing package flows that are close to the load doors' designed capacities.

7.4.1 Task input sequencing problem using imperfect information

In the previous section, it is demonstrated that Surrogate Search can identify task input sequences using tasks' PDD data set. Surrogate Search is utilized under the assumption that the PDD data set is 100 percent accurate (perfect information). As noted, one of the reasons that PDD data sets were artificially generated is because the facility cannot effectively collect the PDD data for trailers. In addition, Surrogate Search was executed for 97 iterations on average for each test problem, which requires more than five days of CPU time.

The goal of developing Surrogate Search approaches for the task input sequencing problem is to implement it as a decision tool for facility. To utilize Surrogate Search approaches, there are a number of issues that need to be addressed:

1. Will the facility be capable of collecting task PDD data sets prior to sort operations?
2. Will the PDD data set collected in the facility be 100 percent accurate?
3. Will there be enough time between when the PDD data set is collected and when the sort operations must start to solve the model?

For the issues listed above, the facility currently does not have the ability to provide the 100 percent accurate PDD data set within the time allowance prior to the actual start of operations. To resolve these issues, an alternative approach is to utilize historical PDD data. In the previous section, the experiments were conducted using historical and artificially generated PDD data sets. By using historical data, the PDD data set can be collected prior to the actual

operations and there will be sufficient time to use the Surrogate Search approach. That is, if the Surrogate Search approach requires the PDD data set 60 hours prior to the actual operations, the PDD historical data for the previous 20 days can be utilized. By using the historical data, the only issue left is that the historical PDD data set will not be 100 percent accurate. The PDD historical data is defined as imperfect information for task input sequencing problem.

A method to determine the impact of using imperfect information is to simulate the sortation system and determine the makespan time. For this research, a series of test problems were designed to evaluate the performance when using historical data in Surrogate Search. These test problems were designed by multiplying uniformly distributed random variables with the PDD data sets of scenarios 2, 3, and 4. Different levels of accuracy of the PDD data set can be generated by using different ranges for the uniform distribution. The test problems are listed below:

1. Original problem: test problems from previous section where the input sequence is solved based on 100 percent accurate PDD data.
2. 95 percent problem: problems where 95 percent of the PDD data are the same as in the original problem.
3. 87 percent problem: problems where 87 percent of the PDD data are the same as in the original problem.
4. 80 percent problem: problems where 80 percent of the PDD data are the same as in the original problem.

The PDD data set contains the probabilities for packages in each task going to each destination load door where the total probability for one task is one. To generate a 95 percent problem, the probabilities for each task from the original problem were first multiplied by

uniformly distributed random numbers within the range of 1 to 1.1. The probability for each destination load door was then normalized to match to the criteria that the summation of all load door probabilities in a task equals to one. The maximum package volume change for a destination load door can be as much as 10 percent by multiplying random numbers with PDD data. Based on an empirical test of modifying PDD data for 10,000 tasks, the largest difference for one destination load door between the original and the 95 percent problem is 5.7 percent. For the PDD data of the 95 percent problem, the error of a task's package volume to each destination door is within 5 percent.

Scenarios 2, 3, and 4 are utilized because there are different levels of imbalance in their PDD data. Among these three scenarios, scenario 2 has the most imbalanced PDD data set (only five incoming tasks contain packages for a specified load door) and scenario 4 has the most balanced PDD data set (20 incoming tasks contain packages for a specified load door).

To determine the performance of task input sequences based on imperfect information, the input sequences determined in the original problems will be utilized as input sequences for the 95, 87, and 80 percent problems. The approach to evaluate task input sequences is given as follows:

1. Determine the task input sequence of the original problem.
2. Simulate the 95, 87, and 80 percent problems using the task input sequence of original problem.
3. Determine the task input sequences for the 95, 87, and 80 percent problems using accurate PDD data.
4. Simulate the 95, 87, and 80 percent problems using their associated task input sequences found in step 3.

5. Determine the difference in makespan between the results of step 2 and step 4.

The test problems (95, 87, and 80 percent) will first utilize the input sequences determined by imperfect information (using original problems) and then use the sequences determined by Surrogate Search using there accurate PDD data. In step 3, the input sequences of the 95, 87, 80 percent problems are determined using their accurate PDD data sets. These PDD data sets were recorded during the process of generating the test problems. The Surrogate Search approach was executed for 100 iterations for each of the test problems.

For steps 2 and 4, each test problem was simulated for 20 replications. The experimental results are listed in Table 17 below:

Table 17 Computational results of imperfect PDD data

Scenario 2					**Scenario 3**			
Problem	**Original**	**95%**	**87%**	**80%**	**Original**	**95%**	**87%**	**80%**
Task input sequence of original problem					**Task input sequence of original problem**			
Avg. makespan	5:01:48	5:09:00	5:10:12	5:04:12	4:51:36	4:48:00	4:51:00	4:48:36
SD (minutes)	13.8	10.8	12.6	10.2	10.8	8.4	10.2	9.6
Task input sequence of each problem					**Task input sequence of each problem**			
Avg. makespan		5:00:00	5:03:00	5:04:48		4:51:00	4:54:00	4:45:36
SD (minutes)		10.2	8.4	8.4		9.6	10.2	10.2

Scenario 4				
Problem	**Original**	**95%**	**87%**	**80%**
Task input sequence of original problem				
Avg. makespan	4:48:36	4:47:24	4:44:24	4:48:00
SD (minutes)	9	9	9	10.8
Task input sequence of each problem				
Avg. makespan		4:45:00	4:43:12	4:49:48
SD (minutes)		7.8	4.8	8.4

In Table 17, each of the test problems was simulated for 20 replications. For these test problems, they were first simulated using the task input sequences of the original problems and then simulated using the input sequences determined by Surrogate Search. For 95, 87, and 80 percent test problems, the percentages only indicate that the amount of PDD data (95, 87, and 80 percent) can be accurately predicted by original problems. It is not guaranteed that 80 percent problems will always have longer makespan time then 95 percent problems. By changing the PDD data, test problems can result in shorter makespan time. For problems in scenarios 3 and 4, the difference of average makespan time between using task input sequences and their associated sequences are relatively small (within ± 3 minutes).

Among the test problems, scenario 2 test problems had the most unbalanced PDD data set. For the 95 and 87 percent problems in scenario 2, there were 9 and 7.2 minute improvements in average makespan time respectively by using individual task input sequences. For scenario 2 test problems, statistically significant differences were found by performing paired T tests (common random numbers are assigned to simulation model) to compare the difference between using the original problem sequence and the sequences based on actual PDD data (95 and 87 percent problems). Although there are only two instances that show significant difference in makespan when using imperfect information, these two instances are both in scenario 2 problem set, which has the most unbalanced PDD data. The results indicate that larger error for makespan time might occur when the PDD data is more unbalanced.

The simulation results indicate that it is feasible to use historical PDD data sets in determining the task input sequences. In terms of the PDD data quality, even when the data is only 80 percent accurate, the average makespan time can still be relatively close to the result of using 100 percent data.

7.5 SUMMARY

In this chapter, Surrogate Search approaches were developed for the task input sequencing problem. The task input sequencing problem is defined as constructing task waves for the sortation system input process. The experiments to obtain surrogate objective functions were based on the knowledge of sortation system operations. The surrogate objective function was identified by simulating package flows matched to designed capacity. For the optimization process, three local searches were researched. Surrogate Search utilizes Tabu Search to determine improved solutions. For nine of the ten test problems, Surrogate Search determined solutions that had better quality than the bench mark methodology, Scatter Search. Another set of test problems was designed to identify the possibility of using historical package destination distribution data for the task input sequencing problem. The result showed historical data can be utilized even when the data is only 80 percent accurate.

8.0 SUMMARY AND CONCLUSIONS

The Surrogate Search approach has been developed and demonstrated in this dissertation. Surrogate Search has been shown to be an effective and practical methodology to identify improved system designs for sortation system simulation problems.

8.1 SURROGATE SEARCH APPROACH

The AMHS sortation system simulation model discussed in Chapter 3 has been utilized as a large-scale simulation case study for this dissertation. Experiments for regression meta modeling were discussed in Chapter 4. When decision variables are not related (or constrained) to each other, near optimal solutions for simulation problems can be predicted and simulation models can be replaced by regression meta model under certain circumstances. It was shown that complex constraints contained within these systems are a major challenge for regression meta modeling.

The failure of regression meta modeling in Chapter 4 provided the motivation for developing Surrogate Search. The concept of Surrogate Search is to identify correlated information from simulation results to improve the effectiveness for optimizing large-scale problems by heuristics. The existence of surrogate objective functions is demonstrated for a variety of large-scale simulation problems.

The Surrogate Search approach is presented and discussed in Chapter 5. There are two major steps in Surrogate Search: identify surrogate objective functions and optimize surrogate objective values. The approach to identify surrogate objective functions is to utilize multiple linear regression to analyze simulation results. If multiple linear regression cannot provide a validated regression model as a surrogate objective function, it requires additional efforts to determine surrogate objective functions utilizing system knowledge.

Once surrogate objective functions are identified, the step of optimizing objective values needs to be developed. The optimization step first executes simulation models to generate system performance and surrogate objective values, and then utilizes local searches to determine the next solution that can improve surrogate objective values. A variety of methodologies can be used for the local searches to improve surrogate objective values. In the study, MIP, Random Search, and Tabu Search are used.

The use of multiple regression to define a surrogate objective function is described in Chapter 6 for determining operator assignments. In the optimization step, four local search procedures were researched for identifying improved solutions. The local searches analyze simulation results to provide feedback to adjust surrogate objective function parameters. They often provide consistent improvement in solution quality.

In Chapter 7, the Surrogate Search approach was investigated to solve the task input sequencing problem. In the process of identifying surrogate objective functions, linear regression failed to determine a formula that could be utilized to improve solution quality. However, a result of linear regression was a list of significant variables from the simulation results. These variables were further researched based on knowledge of the system to construct a surrogate objective function. With the identified surrogate objective function, three local

searches were developed to identify improved sequences. The local searches provide sequences that resulted in relatively good surrogate objective values prior to simulation execution. The sortation system simulation problem shows Surrogate Search is an effective methodology to identify improved system designs.

8.2 PERFORMANCE OF SURROGATE SEARCH

Surrogate Search experiments are presented in Chapters 6 and 7 using sortation system simulation models. The results of the operator assignment problem and task input problem showed that Surrogate Search consistently found solutions associated with good quality. The OptQuest module in Arena, which utilizes Scatter Search methodology as an underlying algorithm, is used to solve the same problems for comparison purposes. For the majority of tested instances, Surrogate Search outperformed Scatter Search by identifying solutions with the same quality in fewer iterations or found better solutions with the same number of iterations. This suggests that the process of searching for improved system designs can be dramatically shortened by identifying surrogate objective functions from the simulation results.

Although identifying surrogate objective functions is a time consuming process, the surrogate objective functions found can be applied to a series of problems based on the same simulation model. The objective function doesn't need to be re-defined when solving multiple instances of a simulation problem. For simulation problems in Chapters 6 and 7, the surrogate objective functions are identified through pilot simulations. The rest of the instances were solved using the same surrogate functions without re-running the pilot simulations.

Surrogate Search is flexible in terms of selecting search procedures. A variety of heuristics and optimization techniques can be used for the local searches in Surrogate Search. The local searches in Surrogate Search systematically find improved solutions. Multiple heuristics and optimization methodologies were utilized in both Chapters 6 and 7 to solve the same problem sets. The computational results showed that improved solutions could be obtained by multiple local searches when the same surrogate objective functions are utilized.

Although it is shown that surrogate objective functions exist for variety of simulation problems, an issue for Surrogate Search is the difficulty in identifying surrogate objective functions. While linear regression is utilized to examine significant level of dependent variables and to construct surrogate objective functions, there is not sufficient evidence to guarantee that surrogate objective functions can be found for all types of problems.

8.3 MAJOR CONTRIBUTIONS OF THE DISSERTATION

The major contribution of this study is the development of the Surrogate Search approach. Surrogate Search provides the framework for identifying surrogate objectives and constructing search procedures to solve large-scale simulation optimization problems. The methodology can be utilized by researchers interested in simulation optimization as a comparison methodology for new simulation optimization procedure developments. This contribution is needed in the field because the number of large-scale simulation models of real systems has dramatically increased and most of the existing methodologies are not designed for large-scale simulation optimization.

Another contribution of the dissertation is the simulation of a sortation system. The unique constrains and structure of sortation systems involves a series of problems in operations

research. This dissertation developed Surrogate Search approaches to research problems of system parameter setting, operational policy, operator allocation, and task scheduling. The simulation problems in this study utilized simulation models of one of the most complex AMHS sortation systems in the US.

Finally, this dissertation provides detailed Surrogate Search approaches to hub scheduling problem, which was introduced by McWilliams et al. [122]. The Surrogate Search approach is an alternative method to solve this problem. The computational results show that good solutions can be found in a small number of iteration (less than 100 scenarios) compared to the existing GA approach (5,000 scenarios). This methodology can be employed by operations research analysts who focus on distribution centers. This contribution is needed because distribution center operations managers typically have a short allowance time to determine solutions after the task information is available.

8.4 FURTURE RESEARCH DIRECTIONS

In this dissertation, the Surrogate Search approach is developed as the simulation optimization methodology that utilizes simulation results to formulate an objective function. There are a number of new research issues raised with the experiments performed in this study. These research issues can be divided into three areas: Improve the procedure to identify surrogate objective functions, develop surrogate constraints, and further applications on Surrogate Search.

8.4.1 Improve methods to identify surrogate objective functions

The process for identifying surrogate objective functions utilizes multiple linear regression and system knowledge. For this research, the variable selection of the linear regression approach was manually processed. One of the future research directions is to develop a general procedure for a linear regression approach to construct surrogate objective functions. The procedure can be developed using the measures of regression models include R^2 values, significance of regression model, and significance of coefficient.

For those simulation problems where surrogate objective functions cannot be determined by linear regression, methods to collect system knowledge such as interviewing field experts can be time consuming. The process of identifying surrogate objective functions involves understanding system behavior. Currently, there are a number of existing methodologies that are designed to analyze system behavior. These methodologies include data mining, artificial intelligence, and artificial neural networks. For large-scale simulation models, system behavior can be generated by simulating diversified scenarios. The surrogate objective functions can be identified by using methodologies that analyze the resultant system behavior.

One of the most important reasons that simulation is utilized as an objective function is the realistic results generated by simulation models. Surrogate Search provides a methodology to construct surrogate objective functions based on validated simulation models. By a more in depth analysis of system behavior, we would hope to develop methods to identify surrogate objective functions using less simulation runs, and that predict improved solutions more accurately.

8.4.2 Develop surrogate constraints

Another of the topics that require further investigation is to develop constraints using simulation results. In this dissertation, Surrogate Search developed surrogate objective functions to replace actual objectives. The same concept could be utilized to artificially construct constraints for simulation optimization problems. The major challenges of large-scale simulation optimization problems are the large number of feasible solutions and long computational time to execute simulation models. We want to research the approach for developing effective constraints for simulation problems to reduce the number of feasible solutions.

8.4.3 Further applications on Surrogate Search

An extension of Surrogate Search application is to convert simulation models into decision tools. As a result of Surrogate Search, improved system designs can be identified within a shorter amount of time.

The objective of large-scale simulation applications is to evaluate different system designs. Although the simulation models can provide detailed system information, the long computational time does not allow simulation models to determine optimal system designs for operational levels. Rather, these simulation applications can only be utilized for long term planning.

The goal of Surrogate Search is to identify improved solutions using a reasonable amount of computational time for large-scale simulation problems. For simulation models that can be executed relatively fast, Surrogate Search can be utilized to identify surrogate objective functions to enable these simulation models to be utilized as real time decision tools.

Another application is to develop common surrogate objective functions for a class of simulation problems. In the experiments, the process to identify surrogate functions has only been performed for the pilot simulation. Once a surrogate objective function is found, it can be repeatedly used for different instances. For a class of problems that have similar systems, common surrogate objective functions could be constructed and utilized by that class of problems. The common surrogate objective functions will benefit simulation problem by simplifying the process of identifying surrogate objective functions.

8.5 SUMMARY

The Surrogate Search approach is designed to optimize large-scale simulation models that contain combinatorial decision variables. The surrogate objective functions are identified by analyzing simulation results to observe system behavior. The development of surrogate objective functions can benefit the optimization process by reducing the number of simulation iterations. The experimental results showed that Surrogate Search performed well for complex sortation system simulation problems. This dissertation utilized the simulation model of a sortation system to demonstrate that Surrogate Search can be applied to large-scale simulation problems and contribute to the simulation optimization field. The Surrogate Search approaches and the experimental results are discussed. The dissertation contributions include the Surrogate Search framework for simulation optimization and approaches to solve sortation system simulation problems. Finally, future research directions for improving the method of identifying surrogate objective functions, developing surrogate constraints, and common surrogate objective function are discussed in this Chapter.

BIBLIOGRAPHY

[1] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. New York, NY: Wiley and Sons Inc., 1999.

[2] K. Gokbayrak and C.G. Cassandras. Generalized Surrogate Problem Methodology for Online Stochastic Discrete Optimization. *Journal of Optimization Theory and Applications* 114 (1):97-132, 2002.

[3] W.L. Winston. *Operations research: applications and algorithms*, 3rd ed.. Belmont, CA: Duxbury Press, 53-54, 1994.

[4] D. Bertsimas and J.N. Tsitsiklis. *Introduction to linear optimization*. Belmon, MA: Athena Scientific, 1997.

[5] K. Holmberg and J. Hellstrand. Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound. *Operations research* 46 (2): 247-259, 1998.

[6] A. Beschorner and D. Gluer. Flow theory for availability calculation of automated material handling systems. *Robotics and Computer Integrated Manufacturing* 19: 141-145, 2003.

[7] H.P.L. Luna and P. Mahey. Bounds for global optimization of capacity expansion and flow assignment problem. *Operations research letter* 26: 211-216, 2000.

[8] S.J. Watson and A.G. Ter-gazarian. The optimization of renewable energy sources in an electrical power system by use of simulation and deterministic planning models. *Int. Trans. Opl. Res.* 3, (3): 255-269, 1996.

[9] G. Bai, S. Bobba, and I.N. Hajj. Simulation and optimization of the power distribution network in VLSI circuits. *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design* 10A: 481-486, 2000.

[10] I. Lee. Artificial intelligence search methods fir multi-machine two-stage scheduling. *1998 ACM*: 31-35, 1998.

[11] J. Teghem, D. Tuyttens, and E.L. Ulungu. An interactive heuristic method for multi-objective combinatorial optimization. *Computers & Operations Research* 27: 621-634, 2000.

[12] P.P.C. Yip, and Y. Pao. Combinatorial optimization with use of guided evolutionary simulated annealing. *IEEE Transactions on neural networks* 6: 290-295, 1995.

[13] M. Chakraborty and U.K. Chakraborty. Applying genetic algorithm and simulated annealing to combinatorial optimization problem. *International Conference on Information Communications and Signal Processing*: 929-933, 1997.

[14] K. Ohkura, T. Igrashi, K. Ueda, S. Okauchi, and H. Matsunaga. A genetic algorithm approach to large scale combinatorial optimization problems in advertising industry. *IEEE*: 351-357, 2001.

[15] M.C. Chen. Optimizing machining economics models of turning operations using the scatter approach. *INT. J. PROD. RES*. 24: 2611-2625, 2004.

[16]A.M. Law and W.D. Kelton. *Simulation modeling and analysis*, 3rd ed. Boston, MA: McGraw-Hill Higher Education, 646-647, 2000.

[17] B. Banks, J.S.II. Carson, B.L. Nelson, and D.M. Nicol, *DISCRETE-EVENT SYSTEM SIMULATION*, 3rd ed. Saddle River, New Jersey: Prentice Hall: 489-459, 2001.

[18] M.A. Hofmann. Criteria for Decomposing Systems into Components in Modeling and Simulation: Lessons Learned with Military Simulations. *Simulation* 80 (7-8): 357-385, 2004.

[19] P.J. Koopman. A taxonomy of decomposition strategies based on structures, behaviors, and goals. *Design theory & Methodology* 95, 1995.

[20] B. Delinchant, F. Wurtz, D. Magot, and L. Gerbaud. A Component-Based Framework for the Composition of Simulation Software Modeling Electrical Systems. *Simulation* 80 (7-8): 347-356, 2004.

[21] H.T. LeBaron and R.A. Hendrickson. Using emulation to validate a cluster tool simulation model. *Proceedings of the 2000 Winter Simulation Conference*: 1417-1422, 2000.

[22] G. Cardarelli, P.M. Pelagagge, and A. Granito. Performance analysis of automated interbay material handling and storage system for large wafer fab. *Robotics and computer-integrated manufacturing* 3: 227-234, 1996.

[23] K. H. Weigl. Simulation of a large-scale brewery distribution system. *Proceeding of the 1998 Winter Simulation Conference*: 1255-1259, 1998.

[24] G.D. Smith and D.J. Medeiros. Simulation of flexible control strategies. *Proceeding of the 1995 Winter Simulation Conference*: 799-804, 1995.

[25] D. Nazzal and D.A. Bodner. A simulation-based design framework for automated material handling system, in 300mm fabrication facilities. *Proceeding of the 2003 Winter Simulation Conference*: 1351-1359, 2003.

[26] T.S. Meinert, G.D. Taylor, and J. R. English. A modular simulation approach for automated material handling systems. *Simulation Practice and Theory* 7: 15-30, 1999.

[27] S. Harit and G.D. Taylor. Framework for the design and analysis of large scale material handling systems. *Proceeding of the 1995 Winter Simulation Conference*: 889-894, 1995.

[28] L. Zuhang, Y.S. Wang, J.Y.H. Fuh, and C.Y. Yee. On the role of a queueing network model in design of a complex assembly system. *Robotics and computer-integrated manufacturing* 14: 153-161, 1998.

[29] C. Wu and R.E. Caves. Modeling of aircraft rotation in a multiple airport environment. *Transportation research part E* 38: 265-277, 2002.

[30] Y. Shi, E. Watson, and Y. Chen. Optimistic parallel simulation of a large-scale view storage system. *Proceedings of the 2001 Winter Simulation Conference*: 1045-1052, 2001.

[31] Y. Shi, E. Watson, and Y. Chen. Model-driven simulation of world-wide-web cache policies. *Proceedings of the 1997 Winter Simulation Conference*: 1045-1052, 1997.

[32] Y.H. Lee, M.K. Cho, and Y.B. Kim. A Discrete-Continuous Combined Modeling Approach for Supply Chain Simulation. *Simulation* 78 (5): 321-329, 2002.

[33] P. Lendermann, N. Julka, B.P. Gan, D. Chen, L.F. McGinnis, and J.P. McGinnis. Distributed Supply Chain Simulation as a Decision Support Tool for the Semiconductor Industry. *Simulation* 79 (3): 126-138, 2003.

[34] J. Jenvald and M. Morin. Simulation-supported live training for emergency response in hazardous environments. *SIMULATION & GAMING* 35 (3): 363-377, 2004.

[35] S. Kaltenhauser. Tower and airport simulation: flexibility as a premise for success research. *Simulation Practice and Theory* 11: 187-196, 2003.

[36] S.T. Shikalgar, D. Fronckwiak, and E.A. MacNair. 300MM wafer fabrication line simulation model. *Proceedings of the 2002 Winter Simulation Conference*: 1365-1368, 2002.

[37] S.T. Shikalgar, D. Fronckwiak, and E.A. MacNair. application of cluster tool modeling to a 300 mm fab simulation. *Proceedings of the 2003 Winter Simulation Conference*: 1394-1397, 2003.

[38] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. Topology modelling: On realistic network topologies for simulation . Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research: 28-32, 2003.

[39] J. Banks. Panel. Session: the future of simulation. *Proceedings of the 2001 winter simulation conference*: 1453-1460, 2001.

[40] A.M. Law. Simulation-based optimization. *Proceedings of the 2002 Winter Simulation Conference*: 41-44, 2002.

[41] B. Sadoun. An Efficient Simulation Methodology for the Design of Traffic Lights at Intersections in Urban Areas. *Simulation* 79 (4): 243-251, 2003.

[42] J. Lee and S. Chi. Using Symbolic DEVS Simulation to Generate Optimal Traffic Signal Timings. *Simulation* 81 (2): 153-170, 2005.

[43] R.V. Hogg and J. Ledolter. *Applied statistics for engineers and physical scientists*, 2nd ed. New Yprk, NY: Macmillan Publishing Company: 345-346, 1992.

[44] A.M. Law and W.D. Kelton. *Simulation modeling and analysis*, 3rd ed. Boston, MA: McGraw-Hill Higher Education, 646-647, 2000.

[45] G.T. Mackulak and P. Savory. A simulation-based experiment for comparing AMHS performance in a semiconductor fabrication facility. *IEEE transactions on semiconductor manufacturing* 14 (3): 273-280, 2001.

[46] S. Durieus and H. Pierreval. Regression metamodel for design of automated manufacturing system composed of parallel machines sharing a material handling resource. *International journal of production economics*: 1-10, 2003.

[47] M.A. Irizarry, J.R. Wilson, and T. Trevino. A flexible simulation tool for manufacturing-cell design, I: model structure, operation, and case study. *IIE Transactions* 33: 827-836, 2001.

[48] M.A. Irizarry, J.R. Wilson, and T. Trevino. A flexible simulation tool for manufacturing-cell design, II: response surface analysis and case study. *IIE Transactions* 33: 837-846, 2001.

[49] S. Bose and J.F. Pekny. A model predictive framework for planning and scheduling problem: a case study of consumer goods supply chain. *Computers and Chemical Engineering* 24: 329-335, 2000.

[50] B. Dengiz and K.S. Akbay. Computer simulation of a PCB production line: metamodeling. *International Journal of Production Economics* 63: 195-205, 2000.

[51] T. Yang, Y. Kou, and P. Chou. Solving a multiresponse simulation problem using a dual-response system and scatter search method. *Simulation Practice and Theory* 13, 2005. 356-369

[52] J. Otamendi. GESAS II: A Better Relationship between Efficiency and Efficacy While Experimenting with Simulation Models. *Simulation* 80 (2): 77-85, 2004.

[53] J.G. Taylor. *New Avenues In Neural Networks. Networks And Their Applications*. New York, NY: John Wiley & Sons Inc, 278-281, 1996.

[54] R.A. Kilmer, *Artificial neural network metamodels of stochastic computer simulations.* Industrial Engineering Department University of Pittsburgh, PhD dissertation: 160-162, 1994.

[55] R.A. Kilmer, *Artificial neural network metamodels of stochastic computer simulations.* Industrial Engineering Department University of Pittsburgh, PhD dissertation: 21-24, 1994.

[56] R.R. Barton. Simulation metamodels. *Proceedings of the 1998 Winter Simulation Conference*: 167-174, 1998.

[57] M. Laguna and R. Martí. Neural Network Prediction in a System for Optimizing Simulations. *IIE Transactions* 34 (3): 273-282, 2002.

[58] I. Lee, J.N.D. Gupta, and A.D. Amar. A multi-neural-network learning for lot sizing and sequencing on a flow-shop. *Proceedings of the 2001 ACM symposium on Applied computing*: 36-40, 2001.

[59] K.R. Caskey. A manufacturing problem solving environment combining evaluation, search, and generalisation methods. *Computers in Industry* 44: 175-187, 2001.

[60] C.R. Reeves and J.E. Beasley, *Modern Heuristics Techniques for Combinatorial Problems*, 1st ed., Great Britain, Black Well Scientific Publications, 1993.

[61] F. Azadivar and Y. Lee. Optimization of discrete variable stochastic systems by computer simulation. *Mathematics and computers in simulation* 30: 331-345, 1988.

[62] S. Andradóttir. Optimization of the transient and steady state behavior of discrete event systems. *Management Science* 42 (5): 717-737, 1996.

[63] S. Andradóttir. A method for discrete stochastic optimization. *Management Science* 41 (12): 1946-1961, 1995.

[64] R.O. Bowden and J.D. Hall. Simulation optimization research and development. *Proceedings of the 1998 Winter Simulation Conference*: 1693-1698, 1998.

[65] H. Arsham. Algorithms for sensitivity information in discrete-event systems simulation. *Simulation Practice and Theory* 6: 1-22, 1998.

[66] D.W. Hutchison and S.D. Hill. Simulation optimization of airline delay with constraints. *Proceeding of the 2001 Winter Simulation Conference*: 1017-1022, 2001.

[67] D. Subramanian, J.F. Pekny, and G.V. Reklaitis. A simulation-optimization framework for addressing combinatorial and stochastic aspects of an R&D pipeline management problem. *Computers and chemical engineering* 24: 1005-1011, 2000.

[68] D. Subramanian, J.F. Pekny, and G.V. Reklaitis. A simulation-optimization framework for research and development pipeline management. *AIChE Journal* 47: 2226-2242, 2001.

[69] A. Nandi and P. Rogers. Using Simulation to Make Order Acceptance/Rejection Decisions. *Simulation* 80 (3): 131-142, 2004.

[70] G.W. Evans, B. Stuckman, and M. Mollaghasemi. Multicriteria optimization of simulation models. *Proceedings of the 1991 winter simulation conference*: 894-900, 1991.

[71] T. Pukkala and J. Miina. A method for stochastic multiobjective optimization of stand management. *Forest ecology and management* 98: 189-203, 1997.

[72] E.F. Watson, D.J. Medeiros, and R.P. Sadowski. Generating component release plans with backward simulation. *Proceedings of the 1993 Winter Simulation Conference*: 930-938, 1993.

[73] E.F. Watson, D.J. Medeiros, and R.P. Sadowski. A simulated-based backward planning approach for order-release. *Proceedings of the 1997 Winter Simulation Conference*: 765-772, 1997.

[74] C.C. Ying and G.M. Clark. Order release planning in a job shop using a bi-directional simulation algorithm. *Proceedings of the 1994 Winter Simulation Conference*: 1008-1012, 1994.

[75] T. Aoki, S. Nakayama, M. Yamamoto, and J. Tanaka. Combinatorial scheduler: Simulation & optimization algorithm. *Proceedings of the 1991 Winter Simulation Conference*: 280-288, 1991.

[76] U. Al-Turki, A. Andijani, and S. Arifulsalam. A New Dispatching Rule for the Stochastic Single-Machine Scheduling Problem. *Simulation* 80 (3): 165-170, 2004.

[77] S. Andradóttir. A review of simulation optimization techniques. *Proceedings of the 1998 Winter Simulation Conference*: 151-158, 1998.

[78] S. Andradóttir. Simulation optimization: integrating research and practice. *INFORMS Journal on Computing* 14 (3): 216-219, 2002.

[79] B. Banks, J.S.II. Carson, B.L. Nelson, and D.M. Nicol, *DISCRETE-EVENT SYSTEM SIMULATION*, 3rd ed. Saddle River, New Jersey: Prentice Hall: 489-459, 2001.

[80] F. Glover, J.P. Kelly, and M. Laguna. New Advances and Applications of Combining Simulation and Optimization. *Proceedings of the 1996 Winter Simulation Conference*: 144-152, 1996.

[81] P.J.M. van Laarhoven and E.H.L. Arts. *Simulated Annealing: theory and applications*. Norwell, MA: Kluwer Academic Publisher, 1988.

[82] E. Arts and J. Korst. *Simulated Annealing and Boltzmann Machines*. New York, NY: John Wiley and Sons, 1-111, 1989.

[83] M.H. Alrefaei and S. Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science* 45 (5): 748-764, 1999.

[84] J. Haddock and J. Mittenthal. Simulation optimization using simulated annealing. *Computer ind. Engng* 22 (4): 387-395, 1992.

[85] T.M. Alkhamis, M.A. Ahmed, and V.K. Tuan. Simulated annealing for discrete optimization with estimation. *European Journal of Operational Research* 116: 530-544, 1999.

[86] F. Wieland and T.C. Holden. Targeting aviation delay through simulation optimization. *Proceedings of the 2003 Winter Simulation Conference*: 578-584, 2003.

[87] Z. Michalewicz. *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd ed.. New York, NY: Spring-Verlag, 1994.

[88] D.E. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[89] J.M. Yunker and J.D. Tew. Simulation optimization by genetic search. *Mathematics and Computers in Simulation* 37: 17-28, 1994.

[90] R. Al-Aomar. A robust simulation-based multicriteria optimization methodology. *Proceedings of the 2002 Winter Simulation Conference*: 1931-1939, 2002.

[91] F. Azadivar and G. Tompkins. Simulation optimization with qualitative variables and structural model changes: A genetic algorithm approach. *European Journal of Operational Research* 113: 169-182, 1999.

[92] S.A. Feyzbakhsh, M. Matsui, and K. Itai. Optimal design of a generalized conveyor-serviced production station: Fixed and removal item cases. *Internal journal of production economics* 55: 177-189, 1998.

[93] P. Kochel and U. Nielander. Kanban optimization by simulation and evolution. *Production Planning & Control* 13: 725-734, 2002.

[94] D.L. McWilliams, P.M. Stanfield, and C.D. Geiger. The parcel scheduling problem: A simulation-based solution approach. *Computer & Industrial Engineering* 49: 393-412, 2005.

[95] H. Pierreval and L Tautou. Using evolutionary algorithms and simulation for the optimization of manufacturing systems. *IIE Transactions* 29: 181-189(1997.

[96] H. Pierreval and J.L. Paris. From 'simulation optimization' to 'simulation configuration' of systems. *Simulation Modelling Practice and Theory* 11: 5-19, 2003.

[97] F. Glover. Tabu Search - Part I. *ORSA Journal on Computing* 1 (3): 190-206, 1989.

[98] F. Glover. Tabu Search - Part II. *ORSA Journal on Computing* 1 (3): 4-32, 1990.

[99] F. Glover and M. Laguna. *Tabu Search*. Norwell, MA: Kluwer Academic Publishers, 1997.

[100] F. Glover, and G.A. Kochenberger. *Handbook of Metaheuristics*. Norwell, MA: Kluwer Academic Publishers, 37-54, 2003.

[101] B. Dengiz and C. Alabas. Simulation optimization using tabu search. *Proceedings of the 2000 Winter Simulation Conference*: 805-810, 2000.

[102] S.H. Jacobson and E. Yucesan. Common issues in discrete optimization and discrete-event simulation. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 47: 341-345, 2002.

[103] J. Pichitlamken and B.L. Nelson. A Combined Procedure for Optimization via Simulation. *ACM Transaction on Modeling and Computer Simulation* 13: 155-179, 2003.

[104] F. Glover, M. Laguna, and R. Marti. Scatter search. *Advances in Evolutionary Computing: Theory and Applications*, A. Ghosh and S. Tsutsui, eds., Springer-Verlag, New York, NY: 519-537, 2003.

[105] F. Glover, and G.A. Kochenberger. *Handbook of Metaheuristics*. Norwell, MA: Kluwer Academic Publishers, 1-36, 2003.

[106] R. Martí, H. Lourenço, and M. Laguna. *Computing Tools for Modeling, Optimization and Simulation*. Kluwer Academic Publishers, Boston, MA: 215-227, 2000.

[107] F. Glover, M. Laguna, and R. Martí. Fundamentals of Scatter Search and Path Relinking. *Control and cybernetics* 29 (3): 653-684, 2000.

[108] V. Campos, F. Glover, M. Laguna, and R. Martí. An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem. *Journal of Global Optimization* 21: 397-414, 2001.

[109] C.D. Paternina-Arboleda and T.K. Das. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory* 13: 389-406, 2005.

[110] J. April, F. Glover, J. P. Kelly, and M. Laguna. Practical introduction to simulation optimization. *Proceedings of the 2003 winter simulation conference*: 71-78, 2003.

[111] A.M. Law and W.D. Kelton. *Simulation modeling and analysis*, 3rd ed. Boston, MA: McGraw-Hill Higher Education, 646-647, 2000.

[112] M. Harchol-Balter, T. Osogami, A. Scheller-Wolf, and A. Wierman. Multi-server queueing systems with multiple priority classes. *To appear in QUESTA* 51 (3-4): 331-360, 2005.

[113] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of Cycle Stealing with Switching Times and Thresholds. *Performance Evaluation*, volume 61 (4): 347-369, 2005.

[114] T. Osogami, M. Harchol-Balter, A. Scheller-Wolf, and L. Zhang. Exploring Threshold-base Policies for Load Sharing. *Forty-second Annual Allerton Conference on Communication, Control, and Computing,* University of Illinois, Urbana-Champaign, October, 2004.

[115] D.C. Montgomery, E.A. Peck, and C.G. Vining, *Introduction to linear regression analysis*, 3rd ed. Wiley Interscience Publication, New York: 120-122, 2001.

[116] D.C. Montgomery, E.A. Peck, and C.G. Vining. *Introduction to linear regression analysis*, 3rd ed. New York, NY: Wiley Interscience Publication, 337, 2001.

[117] B. Banks, J.S.II. Carson, B.L. Nelson, and D.M. Nicol, *DISCRETE-EVENT SYSTEM SIMULATION*, 3rd ed. Saddle River, New Jersey: Prentice Hall: 81-86, 2005.

[118] W.D. Kelton, R.P. Sadowski, and D.T. Sturrock. *Simulation with Arena*, 3rd ed. New York, NY: McGraw-Hill, 567-570, 2004.

[119]A.M. Law and W.D. Kelton. *Simulation modeling and analysis*, 3rd ed. Boston, MA: McGraw-Hill Higher Education, 60-62, 2000.

[120] W.D. Kelton, R.P. Sadowski, and D.T. Sturrock. *Simulation with Arena*, 3rd ed. New York, NY: McGraw-Hill, 583-589, 2004.

[121] F. Glover, J.P. Kelly, and M. Laguna. New advance for wedding optimization and simulation. *Proceedings of the 1999 winter simulation conference*: 255-260, 1999.

[122] D.L. McWilliams, P.M. Stanfield, and C.D. Geiger. The parcel scheduling problem: A simulation-based solution approach. *Computer and Industrial Engineering* 49: 393-412, 2005.