INTEGRATED DECISION MAKING FOR PLANNING AND CONTROL OF DISTRIBUTED MANUFACTURING ENTERPRISES USING DYNAMIC-DATA-DRIVEN ADAPTIVE MULTI-SCALE SIMULATIONS (DDDAMS)

By

Nurcin Celik

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 1 0

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Nurcin Celik entitled Integrated Decision Making For Planning and Control of Distributed Manufacturing Enterprises using Dynamic-Data-Driven Adaptive Multi-Scale Simulations (DDDAMS) and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

_____ Date: July 8th, 2010
Young-Jun Son

_____ Date: July 8th, 2010
Ferenc Szidarovszky

_____ Date: July 8th, 2010
Guzin Bayraksan

_____ Date: July 8th, 2010
Sudha Ram

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: July 8th, 2010
Dissertation Director: Young-Jun Son

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Nurcin Celik

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the guidance of my professors, support of my friends and the love of my family.

I express my sincere thanks to those who made my foray into the world of graduate studies possible. I am grateful to Drs. Young-Jun Son, Ferenc Szidarovszky, Guzin Bayraksan and Sudha Ram for serving on the committee and providing me with many invaluable comments, suggestions, and questions that made this dissertation possible. I would like to especially thank my advisor, Dr. Young Jun Son, for his guidance, contagious enthusiasm and encouragement in all aspects of research and academic life. He has been an irreplaceable role model for my academic career, and the knowledge he has provided me extends beyond what can be found in any textbook.

I extend my thanks to all former and current members of the CIMSLAB. Among them, I especially would like to thank Karthik Vasudevan and Parag Sarfare for their valuable assistance during the early stages of this dissertation in formulating and clarifying the problem. I would also like to thank Dr. Seungho Lee, who has been a big brother to me from the second I joined the CIMSLAB, for friendship, and valuable feedback. I also feel very glad to have Esfand, John, Melody, Srinivas, Hui, and Dong as my colleagues as well. Their company made the long hours in the lab, quite frankly, fun.

I would like to specially acknowledge all my friends, Seher, Zeynep, Debora, Pierre, Perihan, Juana, Adiba, Esra, Ozlem, Ozlem Junior, Lauren, and all others for making my stay in Tucson most memorable. I thank them all for making me truly feel at home. We shared some unforgettable time together - our Friday night outs, last minute coffee breaks, dinners, or just hanging out; for all of which I am grateful.

Finally, I wish to thank to my beloved sisters and very best friends Gul, and Gulcin; and my brothers Olgun and Dogan, who have never quit being beside me and checking out on me, my sister-in-laws, and my brother-in-laws for providing a loving environment and for believing in me. I am forever indebted to my parents, Hamide -

Rafet Koyuncu, and Adile - Osman Celik. They bore me, raised me, taught me, supported me and loved me. I know I can never reciprocate their love in full. I hope I have made them proud. I would also like to thank to my extended family including all my aunts, uncles, and cousins, especially to my nephew Neset and niece Tugba who have become a little brother and sister to me. Finally, to the love of my life, to my husband Emrah Celik, I would like to thank deepest for the love, patience, care, and joy that he brought to my life. To them, I dedicate this dissertation.

DEDICATION

Let the beauty of what you love be what you do.

--Mawlana Jalal ad-Din Rumi

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS - *Continued*

LIST OF ILLUSTRATIONS - *Continued*

LIST OF ILLUSTRATIONS - *Continued*

LIST OF TABLES

ABSTRACT

Discrete-event simulation has become one of the most widely used analysis tools for large-scale, complex and dynamic systems such as supply chains as it can take randomness into account and address very detailed models. However, there are major challenges that are faced in simulating such systems, especially when they are used to support short-term decisions (*e.g.*, operational decisions or maintenance and scheduling decisions considered in this research). First, a detailed simulation requires significant amounts of computation time. Second, given the enormous amount of dynamically-changing data that exists in the system, information needs to be updated wisely in the model in order to prevent unnecessary usage of computing and networking resources. Third, there is a lack of methods allowing dynamic data updates during the simulation execution. Overall, in a simulation-based planning and control framework, timely monitoring, analysis, and control is important not to disrupt a dynamically changing system. To meet this temporal requirement and address the above mentioned challenges, a Dynamic-Data-Driven Adaptive Multi-Scale Simulation (DDDAMS) paradigm is proposed to adaptively adjust the fidelity of a simulation model against available computational resources by incorporating dynamic data into the executing model, which then steers the measurement process for selective data update. To the best of our knowledge, the proposed DDDAMS methodology is one of the first efforts to present a coherent integrated decision making framework for timely planning and control of distributed manufacturing enterprises.

To this end, comprehensive system architecture and methodologies are first proposed, where the components include 1) real time DDDAM-Simulation, 2) grid computing modules, 3) Web Service communication server, 4) database, 5) various sensors, and 6) real system. Four algorithms are then developed and embedded into a real-time simulator for enabling its DDDAMS capabilities such as abnormality detection, fidelity selection, fidelity assignment, and prediction and task generation. As part of the developed algorithms, improvements are made to the resampling techniques for sequential Bayesian inferencing, and their performance is benchmarked in terms of their resampling qualities and computational efficiencies. Grid computing and Web Services are used for computational resources management and inter-operable communications among distributed software components, respectively. A prototype of proposed DDDAM-Simulation was successfully implemented for preventive maintenance scheduling and part routing scheduling in a semiconductor manufacturing supply chain, where the results look quite promising.

CHAPTER 1

INTRODUCTION

In today's global and competitive market, different companies (*e.g.*, suppliers, manufacturers, retailers, distributors, and transporters) form a supply chain to transform raw materials into finished goods and distribute the finished goods to the customers in a supportive manner. For success in a supply chain, coherent planning and control across as well as within each of strategic, tactical, and operational issues are of critical importance (Disney et al., 2008; Dreyer et al., 2009; Longo and Mirabelli, 2008; Rupp and Ristic, 2000; and Towill, 1991). In the decision making process of coherent planning and control, the latest information reflecting immediate supply chain status has to be used in the best possible harmony with current systems capabilities. However, the large-scale, dynamic and complex nature of supply chains makes coherent planning and control very challenging. While it is true for the strategic and tactical levels, it becomes even more so at the operational level as the number of parameters as well as the frequency of update for each parameter grow significantly.

Most decisions in a supply chain are made through various departments over a variety of supply chain echelons which are positioned apart from each other, one at a time or all at once, including interaction between them. For instance, when a tool is unexpectedly broken at a machine shop, the production cycle time of that facility may go up severely affecting 1) the level of work in process as well as finished goods inside that facility, 2) choice of shift of the production (to other facilities or outsourcing), and 3)

transportation and shipment sizes. Due to this interdependence among supply chain members, they support, interact or compete with each other to arrive at an overall optimum or equilibrium (Venkateswaran and Son, 2005). Such segregation and subsequent cooperation of decisions distributed over a range of business units is referred to as distributed decision making.

Successful implementation of the above mentioned distributed decision making in supply chains requires clear understanding of the inherent characteristics of them. First, supply chains are large-scale in nature due to the involvement of multiple departments over geographically dispersed facilities with large-scale demand data. Second, such systems are significantly complex as a consequence of the interdependencies among various components. Third, they are highly dynamic due to random variations in demand, equipment status and facility conditions, re-entrant flow of products, and changing sizes of customer orders as well as priorities.

To address the above mentioned challenges of decision making in supply chains, various modeling and analysis techniques have been proposed, where the goal is to avoid 1) delays in production times during every stage of production, 2) inability of the manufacturers to fulfill the demand within due date, and 3) failure to deliver the finished goods on time. The modeling and analysis techniques available in the literature can be categorized into two main classes: analytical models (deterministic or stochastic) (Geoffrion and Graves, 1984 and Tsiakis et al., 2001), and simulation models (Shapiro, 2000; and MirHassani, 2000).

Between those two classes of approaches, analytical models are preferred to obtain decent results on time when the system scale is relatively small and the system behavior is rather static and predictable. On the other hand, when supply chain components need to respond to highly dynamic changes in demand or in their system, analytical models are not ideal. Instead, discrete event simulation models are capable of handling such dynamicity even in large scale system settings. Also, they allow us to 1) point out strategic supply chain hassles together with certain product-specific supply chain hassles, 2) manage scalability and complexity issues of gigantic supply chain systems in modeling, 3) take randomness into account, and 4) address very detailed models.

In a simulation-based planning and control framework, timely monitoring, simulation analysis, and control is important not to disrupt a dynamically changing supply chain system. Yet, there are three major challenges that we face while using discrete event simulation for this purpose. First, an accurate and complete representation of a real supply chain system may end up being a remarkably detailed model, especially when it is aimed to support short-term decisions (*e.g.* operational decisions or maintenance and scheduling decisions considered in this research). Such a detailed simulation requires significant amounts of computation and execution time. Second, given the enormous amount of dynamically-changing data that exist in the supply chain system, data needs to be updated wisely into the simulation model in order to prevent unnecessary usage of computing and networking resources. Third, there is a lack of methods allowing dynamic data updates during the simulation execution. To illustrate

the impact of system complex on the execution time of the corresponding simulation, the simulation run times of relatively undemanding models representing 1-day worth operation of facilities with various numbers of machines are shown in Figure 1.1. The run time of a simulation model grows exponentially as the number of machines contained in the model increases. Considering the fact that a medium size of a facility is usually comprised of hundreds of machines and a medium size of a supply chain is comprised of several echelons, the run time of the above mentioned simulation model may take hours to represent 1-day worth operation.

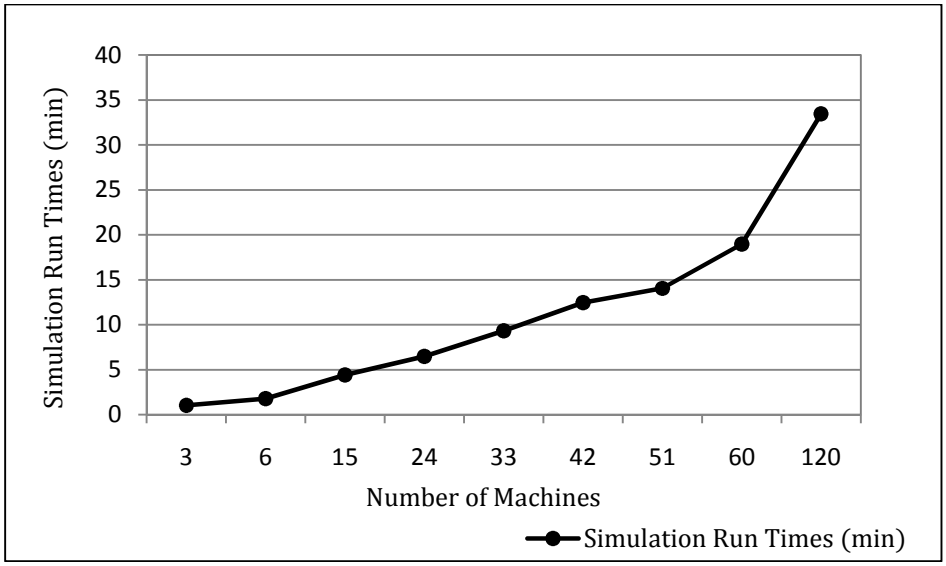


Figure 1.1: Simulation run times for supply chain systems with varying number of machines

To address the above-mentioned challenges and to enable timely monitoring, simulation-based analysis, and control of these supply chains in an economical and effective way, a distributed adaptive simulation scheme, including architecture and

applications, is proposed. The proposed scheme is unique for several reasons. First, simulation adapts its level of detail appropriately to the dynamically changing system conditions (e.g. within the facility or in the partnering facilities) without human intervention via incorporated algorithms (developed in this research). Second, the simulation model adjusts its level of detail efficiently against varying computational resource availabilities considering the accuracy limits of the model required for proficient decision making in the supply chain system. Third, dynamically-changing data is used to update the executing supply chain simulation. Finally, the proposed scheme allows for superior enterprise integration in terms of communication and information synchronization. It is noted that the proposed adaptive scheme above is extensible not only to other areas of supply chain systems but also to a multitude of domains where there is a need to handle a large volume of highly dynamic data.

The components of the proposed comprehensive DDDAMS framework and methodologies include 1) real time DDDAM-Simulation, 2) grid computing modules, 3) Web Service communication server, 4) database, 5) various sensors, and 6) real system. Four algorithms are then developed and embedded into a real-time simulator for enabling DDDAMS capabilities such as abnormality detection, fidelity selection, fidelity assignment, and prediction and task generation. The first embedded algorithm is data filtering algorithm involving control charts. In this work, Algorithm 1 has been implemented in Visual Basic Application (VBA) block and plugged into the Arena simulation model. The Algorithm 1 module is called with a timestamp and data point, and then obtains the measurement from the real system and returns a single filtered data

value. It also stores past data points in the database, which is constantly updated at the rate of which new stream of measurements are taken.

Second is fidelity selection algorithm which is initially proposed and developed using the Bayesian Belief Network and subsequently, enhanced using Sequential Monte Carlo sampling technique in order to make efficient inferences to determine the sources of abnormality in the system (shop floor in this research). This algorithm which works on the main principles of Bayesian inferencing is embedded into the simulation to enable its ideal fidelity selection given massive datasets. As part of this algorithm, two new resampling schemes (one based on the minimized variance and other one based on the minimized bias)are first developed theoretically and then benchmarked against the resampling rules existing in the literature. In DDDAMS for a large-scale system, inference may involve hundreds of sensors for a variety of quantity of interests, which makes real-time inferencing a challenging task considering limited computational resources. Computational intensity would go even worse when evaluation of integrals for continuous variables is considered. To resolve these issues, a parallelization path is developed for the proposed algorithm to reduce the number of data accesses while maintaining the accuracy of parameter estimates.

Third, fidelity assignment algorithm involving mathematical programming is developed to opt for the available fidelity level of each component by taking the system level computational resource constraint into account. This algorithm obtains a matrix which encapsulates the proper (desirable) fidelity level of each component discussed in Algorithm 2 as an input as well as the available computational resource capacity from the

grid computing service, and returns a new matrix which holds the assigned fidelity level for each component in the system that was evaluated at the current time point. This algorithm is based on the well-known Knapsack problem.

Last of all, multi-linear regression and distributed discrete-event simulation models running on the fast mode and operating under decision rules (*e.g.* part selection rules or machine selection rules) are developed with the purpose of online prediction and task generation. It provides a real system with 1) near optimal preventive maintenance (PM) schedule and 2) near optimal part routing (PR) recommendations for operational efficiency of jobs (parts) to be processed. This algorithm is comprised of three main steps, where Step I predicts mean time between failures (MTBF), Step II updates PM schedules based on the information obtained from Step I, and Step III updates PR schedules based on the updated PM schedules.

Grid computing and Web Services are used for computational resources management and inter-operable communications among distributed software components, respectively. A prototype of proposed DDDAM-Simulation was successfully implemented for preventive maintenance scheduling and part routing scheduling in a semiconductor manufacturing supply chain (see Chapter 4 for details), where the results look quite promising.

1.1 Detailed Objectives

As mentioned above, the purpose of this research is to develop a distributed adaptive system structure as an accurate representation of the actual supply chain system

in the most cost-effective manner in order to be able to monitor it, to conduct various what-if analysis over its complex organizational configuration, and to direct the supply chain system to reach or to keep its near optimum or equilibrium by controlling the subsequent distributed decisions over a range of business units. The purpose is divided into the following detailed objectives.

The first objective is *to develop a robust architecture that monitors and controls different and distributed operations as well as interactions of a supply chain system at various levels*. A novel adaptive dynamic data driven simulation-based architecture and methodology is proposed, which will allow us to mimic the supply chain operations in the most accurate manner while enabling 1) efficient usage of computational resources, 2) access to the dynamically-changing data in a supply chain, and 3) superior enterprise integration in terms of communication and information synchronization.

The second objective is *to uncover the sources of abnormality in the shop floor as soon as they occur so that appropriate actions can be taken in a similar timely manner*. Monitoring and control, hence the determination of the sources of abnormality, are performed using advanced sampling techniques including the Bayesian Belief Networks and Sequential Monte Carlo sampling. The latter technique is further enhanced via the developed stratified resampling rules (one based on the minimized variance and other one based on the minimized bias) in order to make inferences efficient given massive datasets. These new resampling schemes, which are first developed theoretically and then benchmarked against the resampling rules existing in the literature. This algorithm

which works on the main principles of Bayesian inferencing is embedded into the simulation to enable its ideal fidelity selection.

The third objective is *to determine short term operational/planning decisions in the supply chain via intelligent algorithms*. In this study, look-ahead algorithms are implemented via fast mode simulation to determine the best possible values of various system parameters, with a focus on the impact of the decisions made onto the operational efficiency of the supply chains. Different optimization techniques (including meta-heuristics and multi-linear regressions) are employed to determine the plans and schedules depending on the specific need in the supply chain.

The fourth objective is *to identify a parallel and distributed implementation path for the proposed methodology due to the diverse nature of computational resources which are spread across the supply chain*. In DDDAMS for a large-scale system, inference may involve hundreds of sensors for a variety of quantity of interests, which makes real-time inferencing a challenging task considering limited computational resources. Computational intensity would go even worse when evaluation of integrals for continuous variables is considered. To resolve these issues, a parallelization path is developed for the proposed algorithm to reduce the number of data accesses while maintaining the accuracy of parameter estimates.

The fifth objective is *to evaluate the effect of interactions between the decision models in different levels and across different members*. This is performed to better understand the global consequences of the locally optimal decisions determined at each decision model.

The sixth objective is *to develop an infrastructure to enable distributed analysis of systems of systems*. The decisions models at different supply chain members by themselves are complex systems, and this research work is concerned with the functioning of the systems of systems.

The seventh objective is *to implement and demonstrate the proposed Dynamic Data Driven Adaptive Multi-scale Simulation (DDDAMS) paradigm in a simulated environment (e.g. distributed-computing environment).* DDDAMS is to steer the measurement process for selective data update and incorporate the real-time dynamic data into the executing model.

The eighth objective is *to demonstrate functioning of the proposed architecture for different supply chain configurations*. The proposed DDDAMS architecture and algorithms are used to find near optimum 1) preventive maintenance schedules and 2) part routing rules in a semiconductor supply chain in a distributed computing environment.

1.2 Organization of the Remainder of the Dissertation

The remainder of the dissertation is organized as follows keeping in mind readability and easy understanding of the various methodologies proposed as parts of this study. In Chapter 2, an introduction to supply chain systems including various configurations in its structure, decision levels in its management and information sharing methods in its operations is provided. Then literature survey is presented focusing on the decision-making models of the supply chains and various Bayesian inferencing

techniques. In Chapter 3, structural and functional characteristics of the supply chain under study are presented. The assumptions made while developing the proposed decision-making framework is also discussed in this section. Then, details of the proposed system architecture along with all the algorithms developed to realize its sub-modules as well as their interactions with each other are described in Chapter 4. Integration of the various modules of the proposed decision-making framework in a distributed virtual setting based on Web services and Grid computing technologies is explained in Chapter 5. The results and analysis procedures are detailed in Chapter 6. Methodology developed for partitioning of large-scale simulations as part of the extended DDDAMS framework is presented in Chapter 7. Chapter 8 includes a summary of the findings and the conclusion of the thesis. The directions of future research are also discussed in this section.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, we first provide a brief background on supply chain systems involving configurations of supply chains, decision levels and information sharing among its echelons. Next, we review the literature on the modeling of supply chains focusing on simulation modeling. Also, previous research works, on new venues of simulation involving simulation-based planning and control, distributed simulation, and Bayesian analysis in simulation motivating part of the research questions of this dissertation, are discussed in detail.

2.1 Background on Supply Chains

Traditionally, various business units along the supply chain, such as suppliers/vendors, distribution centers, manufacturing plants, transportation network, warehouses, and retailers/customers operate independently. These units have their own, often conflicting, objectives (Ganeshan and Harrison, 1993). The critical supply chain management seeks to integrate performance measures over multiple firms or processes, rather than taking the perspective of a single firm or process (Lambert at al., 1998). In this research, against the majority of the literature aiming to find ways to enable effective supply chain management at an aggregate mode (*i.e.*, in the strategic level of decision making), we develop a coherent monitoring and control mechanism at detail levels of decision making (*i.e.*, tactical and operational levels of decision making) enabling the

management of these large-scale, dynamic and complex systems operate as efficiently as possible.

2.1.1 Structure and Configuration of Supply Chains

This section discusses a variety of supply chain configurations and inter-organizational relationships formulating those configurations. Some of these configurations are used to illustrate and demonstrate the proposed approach in this research.

Configuration wise, supply chain systems can be of two main types: 1) collaborative supply chain or 2) competitive supply chain. Collaborative supply chain systems are usually comprised of echelons (or partners) which belong to the same company, and therefore significant amount of information related to operations are shared among these echelons. On the other hand, the competitive supply chain systems are composed of echelons which in general belong to different companies or organizations. Therefore, access to other members' data is restricted to either partially or to the full extent by the regulations of the corresponding competitor partners within the supply chain. Van de Ven and Ferry (1980) identifies the underlying inter-organizational relationship patterns forming these configurations as follows:

(a) A dyadic network involves the interaction between two firms (*e.g.* 1 supplier and 1 manufacturer).

(b) A multiple dyadic network involves the interaction of one firm with several other firms (1 to *N* or *N* to 1). This can take the form of 1 supplier and *N* manufacturers

or *N* suppliers and 1 manufacturer. Here the *N* participants can also be competitors. An example of the 1 to *N* interaction is the relationship between an airline and several independent travel agencies.

(c) A many to many (or multi-channel) network is one where several firms interact with several other firms (*M* to *N*). Here we could have *M* suppliers linked to *N* manufacturers with competition possible within the *M* and the *N* groups.

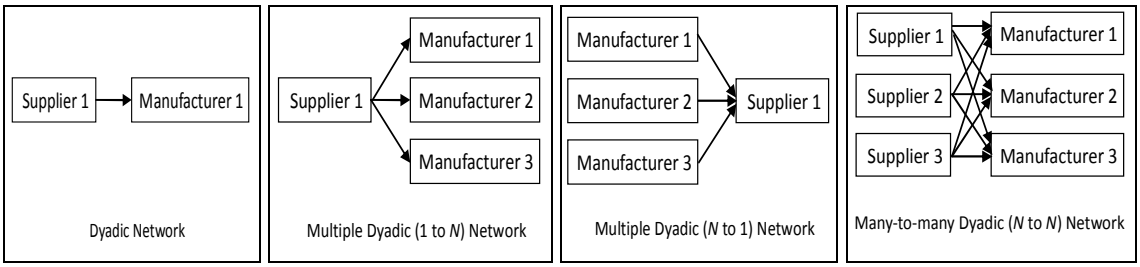


Figure 2.1: Configurations of supply chains with varying inter-organizational relations (courtesy from Van de Ven and Ferry, 1980)

In the above configurations, we have only considered patterns for a single stage relationship between two layers of the network. The number of suppliers contributes to the complexity of the supply network (Beamon, 1999), hence in a single stage relationship, the dyadic network will experience the least complexity, and the multi-channel networks the most (Samaddar et al., 2006).

Typically, simple dyadic supply chains (see the first graph in Figure 2.1) become highly collaborative. Convergently and divergently structured ones (see the second and third graphs in Figure 2.1), on the other hand, develop into competitive supply chains where the degree of competitiveness of the supply chain depends on the number of

networks included as well as their complexities, the regulations among the firms forming these networks, and the pressure of overall industry in which the considered supply chain operates.

2.1.2 Decision Levels in Supply Chains

The present literature identifies three key dimensions to describe complex supply chains-vertical structure (within the supply chain), horizontal structure (in between supply chain echelons), and location in the network (Harland, 1996; Lambert et al., 1998; and Spens and Bask, 2002). A clear understanding of interactions occurring in between network members and within each supply chain echelon can be achieved by studying these factors. As Samaddar et al., (2006) mentions, for these interactions, goods and services flow in one direction; payments flow in the opposite direction; and information flows in both directions. In this section, vertical structure of supply chain partners is examined whereas next section is concerned with the horizontal structure of the overall supply chain network together with the location aspect of each individual echelon inside the network.

Decisions in a supply chain partner often belong to one of the three categories or levels – the strategic, the tactical or the operational level. The levels of supply chains can be represented as a pyramid shaped hierarchy (see Figure 2.2). The decisions on a higher level in the pyramid will set the conditions under which lower level decisions are made.

Vankateswaran and Son (2005) summarize the characteristic of those levels as follows. On the strategic level, long-term (annually/half-yearly) decisions are made.

These are closely linked with the corporate strategy. Tactical and operational levels are concerned with medium-term (quarterly/monthly) and short-term (weekly/daily) decisions, respectively. Since there is no clear demarcation between tactical and operational level, they are frequently combined and referred to just as operations level.

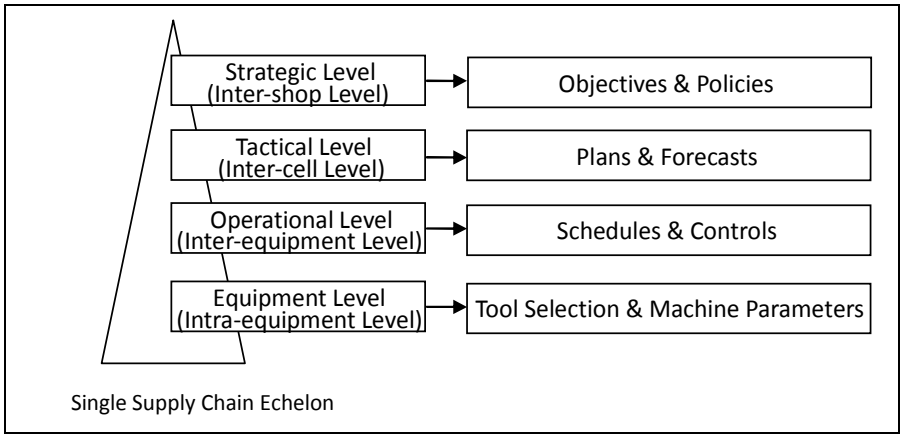


Figure 2.2: Supply chain decision levels

Decisions such as those concerned with capacity planning, facility location and choice of transportation are considered as strategic level decisions since they involve a commitment of resources to a long-term plan. Once the size, number, and location of the facilities are determined, so are the possible paths by which the product flows through to the final customer. These decisions are of great significance to a firm since they represent the basic strategy for accessing customer markets, and will have a considerable impact on revenue, cost, and level of service. Decisions related to outsourcing, bidding and allocation of workload are defined as tactical level decisions. Furthermore, replenishment policy, planning, scheduling and shop-floor control decisions are

commonly thought of operational level decisions. Equipment level decisions mostly handles selection of appropriate tools and machine specific parameters within the facility. Finally, production decisions, inventory decisions, choice of shipment size and routing decisions are considered to be on strategic, tactical as well as operational level. Ganeshan and Harrison (1993) explains that the strategic decisions include what products to produce, and which plants to produce them in, allocation of suppliers to plants, plants to distribution centers, and distribution center's to customer markets. Operational decisions focus on detailed production scheduling. These decisions include the construction of the master production schedules, scheduling production on machines, and equipment maintenance. Other considerations include workload balancing, and quality control measures at a production facility. These refer to the means by which inventories are managed. Inventories exist at every stage of the supply chain as either raw material, semi-finished or finished goods. Their primary purpose is to buffer against any uncertainty that might exist in the supply chain. Their efficient management is critical in supply chain operations. Inventory decisions are strategic in the sense that the top management sets goals. However, most researchers have approached the management of inventory from an operational perspective. These include deployment strategies (push versus pull), control policies - the determination of the optimal levels of order quantities and reorder points, and setting safety stock levels, at each stocking location.

In this thesis work, a three echelon conjoined supply chain is analyzed. Also, the decisions of interest include the production scheduling and preventive maintenance decisions at the tactical and operational levels.

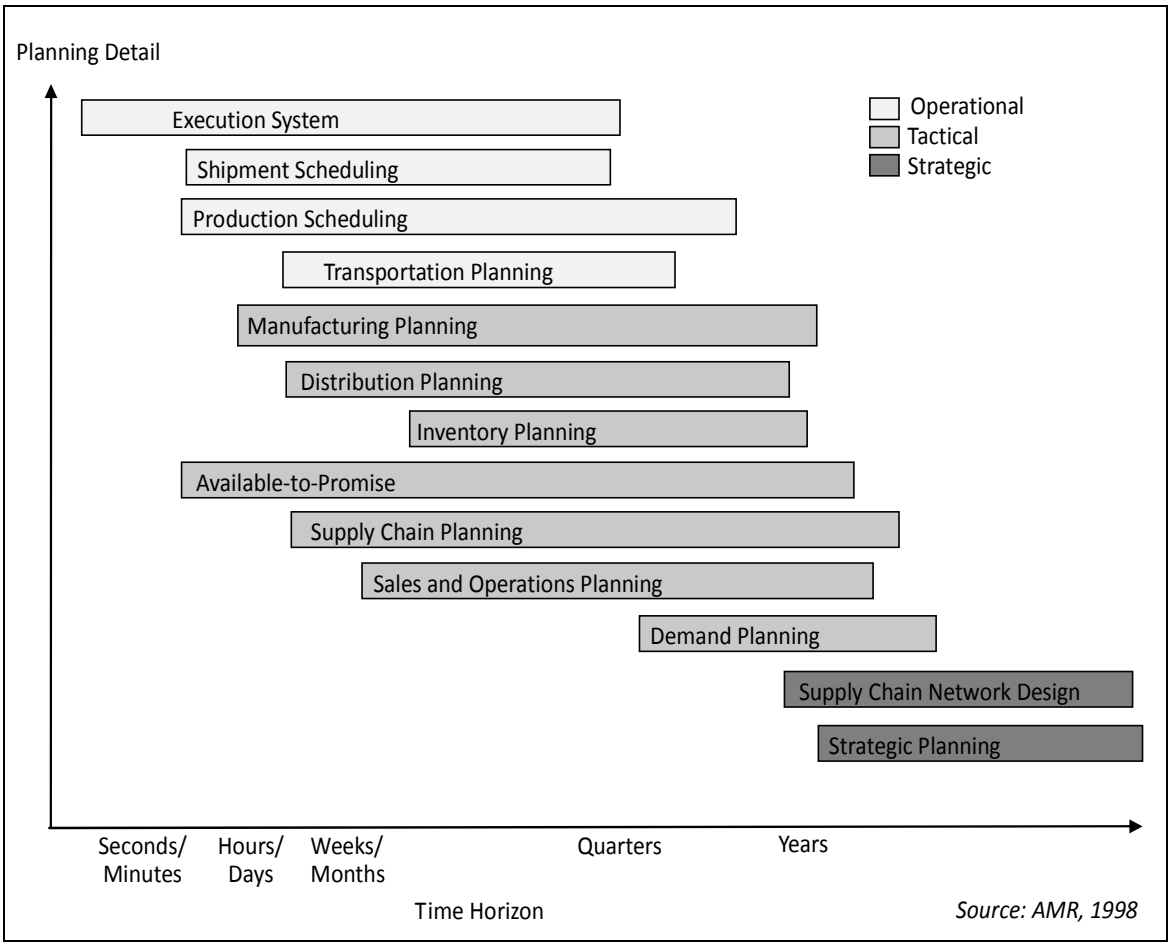


Figure 2.3: Scope of advanced planning and scheduling levels (courtesy from AMR, 1998)

2.1.3 Information Sharing in Supply Chains

Information sharing in supply chains enables coordination across its partners and is known to help allocate inventories efficiently across different stages. Information sharing has also been found to reduce the bullwhip effect (order variability) and add value to the chain as a result of various studies (Chen et al., 2000, Gaur et al., 2005, Lee et al., 1997, and Ouyang, 2007).

Inter-organizational information sharing involves sharing across firm boundaries, and is needed since organizations are unable to generate all of their required resources internally. Firms must, therefore, interact with other organizations that control these critical resources so that they can compete effectively in their environment. When information is shared in inter-organizational networks, it can result in a more efficient flow of goods and services, reduced inventory level, and lower costs, which benefits the overall network. For example, Wal-Mart shares point-of-sale information with their suppliers and transmits orders electronically to the relevant supplier when inventory for an item falls below a predetermined minimum level of stock (Lancioni et al., 2000). This inter-organizational information sharing reduces carrying costs of inventory, facilitates quick response for inventory replenishment and allows suppliers to better plan their production schedules, and reduces lead times (Stevenson, 1994).

While several research, as detailed above, have shown that inter-organizational information sharing leads to improved performance of the supply network, it is difficult for firms to reap these benefits unless they have a better grasp of the antecedents of inter-organizational information sharing that influence its effectiveness. This is an area in which firms need guidance so that they can effectively channel resources to their knowledge and information sharing activities. The ability of managers to coordinate the complex network of business relationships that exist between parties involved in the supply network is critical to the success of a firm (Lambert et al., 1998).

The network patterns, which can have one or more stages, and range from the simple dyadic to the multi-channel network within a single stage, create varying levels of

complexity, and hence differing environments for information sharing. Furthermore, location of the firm(s) within a network creates different information needs and different consequences due to the distortion in the flow of information. Managing the flow of information effectively requires close attention to the coordination mechanism established among the member firms in a network. The degree of centralization is likely to affect the nature and amount of information that gets shared across the network. Similarly, the degree to which the partner firms perceive a match in their goals may impact the nature and amount of information they are willing to share with each other (Samaddar et al., 2006).

Zhou and Benton (2007) summarize the critical aspects of the information to be shared in three main categories which are namely 1) supportive technology used for information sharing, 2) content of the shared information, and 3) quality of the shared information. Supportive technology used for information sharing includes the hardware and software needed to support the sharing process. Content of shared information refers to the information shared between various members of supply chain and its end-customers. Quality of shared information, as the name implies, measures the quality of information shared between various partners of supply chain and customers.

In this research, in addition to the three aspects mentioned above, one more aspect of information sharing, which is degree of information sharing, is considered. The degree of information shared can be divided into three aggregated levels: No information sharing, partial information sharing and full information sharing. Samaddar et al., (2006) briefly explains these levels in their paper. In the case of no information sharing, the

supplier only has information on the orders received from the manufacturer and must utilize historical data to augment the order information when preparing demand forecasts. With partial information sharing the demand distribution faced by the retailer and the retailer's inventory policy are known. Finally with full information sharing, the supplier also receives instantaneous information on the retailer's demand. In addition, the focus on only the amount of information shared (none, partial, full) does not provide sufficient tools to make any assertions on the impact of sharing different types of information. For instance, are there scenarios under which the sharing of strategic information is advantageous to the supplier or the manufacturer, or to the total network? Does this change when competitive information is shared?

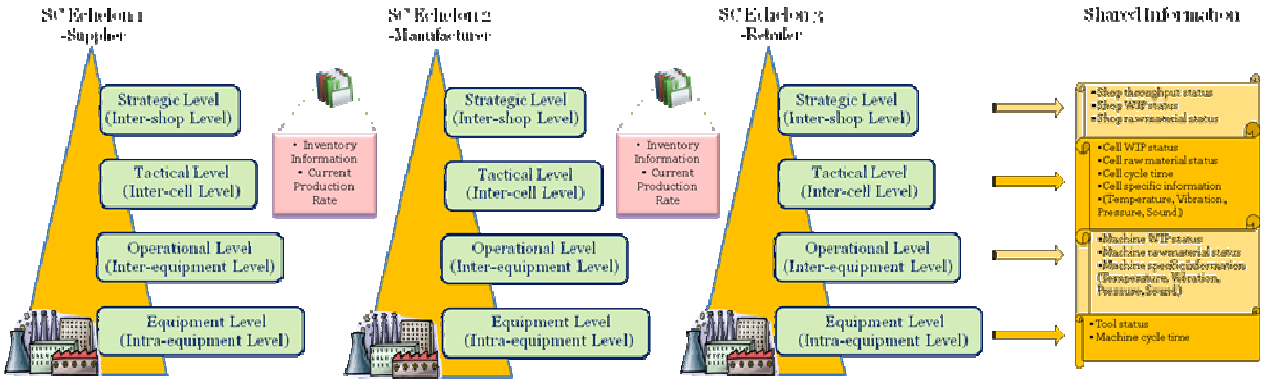


Figure 2.4: Information sharing among supply chain partners

2.2 Models for Supply Chain Decision Making

Decision making in supply chains is a complicated process because supply chain systems are large scale, complex, and inherits dynamicity due to various interactions among its echelons by their nature. The intrinsic complexity of decision making grounds

a growing need for efficient modeling techniques enabling operations of high performance supply chains. Because of this complexity, much of the past research in this area focused on individual echelons of supply chains. However, recently placed works (Wikner et al., 1991; Towill, 1991; Lee and Billington, 1993; Towill and Del Vecchio, 1994; and Venkateswaran and Son, 2004) pay considerable attention to the design, analysis and performance of the supply chain as a whole. Beamon (1998) categorizes the modeling approaches for supply chain systems in the following main classes: 1) deterministic analytical models, 2) stochastic analytical models, and 3) simulation.

In the past, significant amount of research has been conducted to develop linear programming, mixed integer linear programming and dynamic programming models with the aim of addressing various decision making problems that exist in supply chain networks such as optimal location selection of distribution facilities between plants and customers (Geoffrion and Graves, 1984; Cohen and Lee, 1988), routing in transportation systems (Camm et al., 1997), and strategic planning for multi-echelon supply chain networks (Tsiakis et al., 2001). Deterministic optimization problems are formulated with deterministic parameters. Real world problems, in contrast, almost invariably include some uncertain parameters. Hence, deterministic models are not fully capable of representing the actual real system. Another important issue regarding the deterministic programming formulations is that of computational complexity, especially in the context of scenario planning approaches for the handling of uncertainty. Most practical optimization problems related with supply chain networks, involve quadratic number of constraints with huge number of variables leading to NP-hardness. The solving process

(simplex method for liner programming models and branch-and-bound technique for mixed integer programming models) becomes very inefficient as the number of variables increases. Moreover, these types of modeling approaches lack the ability to incorporate dynamically changing data into the executing model meaning that reached solution of the model, is already outdated when it is put into real-world implementation. Hence, it is only suitable to use these models when the scope of the system is relatively small, the computational resources are not scarce and the level of decision is aggregate.

While deterministic programming models are formulated with only known parameters, stochastic programming is a framework for modeling optimization problems that involve uncertainty. Stochastic programming models take advantage of the fact that probability distributions governing the data are known or can be estimated. The goal here is to find a policy that is feasible for all (or almost all) the possible data instances and maximizes the expectation of functions of the decisions and the random variables. Supply chains, in their dynamic and stochastic environment, can be studied as stochastic integer programs (MirHassani et al., 2000; and Wagner, 1995), Markov chains (Choi and Lee, 2006), stochastic Petri-nets and queuing network models (Kerbache and Smith, 2003; and Hennet and Arda, 2007).

MirHassani et al., (2000) points out the fact that mixed integer programming structure of supply chain networking problem makes it practically intractable for applications, which usually involves multiple products, factories, warehouses and distribution centers. The same problem studied with uncertainty makes it even more so. One advantage of the dynamic stochastic programming formulation is that it allows the

decision maker to respond to changes in demand. Another advantage is that it can be formulated with a broad range of ways of representing the demand uncertainty, including the use of continuous probability distributions. Unfortunately, the dynamic stochastic programming formulation is large and very difficult to solve. The computer resources needed to solve such large formulations of stochastic models are of significant amount.

2.2.1 Simulation Models

Models discussed above are high abstraction models for various processes under serious simplifying assumptions such as Markovian dynamics. Among the approaches taken for modeling and analyzing supply chain systems, simulation provides a practical methodology for representing complex interdependencies between organizations, and help to realistically analyze the performance tradeoffs associated with different organizational decision making assumptions (Biswas and Narahari, 2003; and Swaminathan et al., 1995).

Bhaskaran (1998) demonstrates the importance of supply chain analysis by tackling the unstable production scheduling and inventory fluctuation problems in an aggregate level with the usage of a simulation with a user-friendly software which is originally developed to General Motors's specification, by David Kletter and Steve Graves of MIT. They study the supply chain as a whole with a clear understanding of how the various parts interact with each other. In contrast to the previously completed studies, they include material shortages, production capacity constraints and large container sizes in their simulation. The changing inventory levels in various buffers

throughout the supply chain are recorded under two control strategies which are Material Requirements Planning (MRP) and Kanban. Limitations exist for the lack of adequate impact analysis on capacity constraints and material shortages. Further analysis can be conducted on cases where control strategies are hybrid Kanban-MRP policies and/or some tiers are MRP and some tiers are Kanban.

Van der Zee and Van der Vorst (2005) analyze simulations of supply chains not from implementation point of view but from design point of view. They discuss the flaws of current modeling approaches which hinder the successful support of decision making. For instance, decision makers, control rules and their interactions mostly found to be hidden within a manufacturing environment rather than being explicitly visualized. Hence, not only realism but also modeling flexibility and modularity harmed. To resolve these issues, a new generic (nonsoftware-specific) modeling framework for supply chain simulation is proposed. This framework, regardless of the level of strategy (strategic, tactical or operational), intends to improve the efficiency and effectiveness of supply chain process on an object oriented format. Main concepts of the approach involve agents to model supply chain entities and distribution systems as intelligent decision makers, jobs to define chain activities and flows to describe movable objects processed by agents such as goods, data resources or job definitions. Model dynamics is realized by job execution. Areas of improvement of their study may include linkage with Petri-nets for strong mathematical background within the framework and incorporation of artificial intelligence to help increase modeling efficiency and effectiveness.

Another special purpose simulation tool for supply chain analysis is Supply Chain Simulation (SCSIM) by presented Petrovic (2001). Uncertainties among customer demand, external supply of raw material and lead times to the facilities are modeled through fuzzy sets within this tool. Fuzzy analytical model chooses the optimum order-up to level for all inventories in a fuzzy environment and then simulation model evaluates the performance results those could be achieved by implementation of this decision. Primary results are obtained from small scale numerical examples however supportive experimentation needs to be done with the inclusion of continuous review inventory models and other sources of uncertainty in a supply chain for validation purposes.

As examples are provided above, researchers have used simulation models (*i.e.*, discrete-event models or system dynamics models) to study different aspects of the supply chain, such as the instability of the chain (Bhaskaran, 1998), the performance effects of operational factors (Beamon and Chen, 2001), demand amplification effects (Wikner et al., 1991), and the impact of modeling fidelity on the analysis of supply chain dynamics (Venkateswaran and Son, 2004). In many cases, discrete event simulation is a natural approach in studying supply chains as their complexity restrains analytic evaluation. During past two decades considerable research has been conducted on discrete event simulation of supply chains attacking a wide range of problems that exist in these large scale, complex and dynamic systems. In addition to discrete-event simulation (DES) modeling, agent-based (AB) modeling has also been employed to simulate supply chain systems (Van der Zee, 2003; Labarthe et al., 2007; Chatfield et al., 2007). Later, researchers have proposed hybrid simulation approaches. For example,

Lee et al. (2002) represented the continuous aspects of the supply chain (*i.e.*, ordering rate, shipping rate and inventory) using mathematical models, which are then integrated with the discrete aspects of the supply chain such as transportation activity. Similarly, Rabelo et al. (2003) presented the potential merit of integrating SD and DES models to evaluate the impact of local production decision on the global enterprise. Similarly, Venkateswaran and Son (2005) described a two-level Hierarchical Production Planning (HPP) architecture consisting of SD components at the higher decision level and DES components at the lower level. While much of the previous work served as a basis for future supply chain research, an adaptive control system that can respond to the unpredictable changes taking place within a single facility or the whole supply chain in the most efficient manner in any existing type of simulation approach has not been established yet. To the best of our knowledge, our study is unique for being the first to develop an adaptive control system framework for large-scale supply chains which is able to dynamically react to system abnormalities in the most cost-effective manner. While in this research, the studied supply chains and developed simulation models are discrete-event ones, future extension of this work concerns itself with realization of the proposed architecture in continuous simulation environments.

2.2.2 Survey of Dynamic Data Driven Simulations

Dynamic data driven application systems (DDDAS) has been conceived as a powerful tool that allows more effective measurement processes in a variety of application areas which brings along challenges in the aspects of applications,

mathematical algorithms, systems software, and in the way data is collected (Darema, 2004). In a study of contaminant tracking, numerical procedures for multi-scale interpolation are introduced in order to map sensor data and allow continuous update of the simulation (Douglas et al., 2004). Another study performed by Mandel et al. (2004) introduces the use of online data acquisition and a filtering control to recognize out-of-order data. Carnahan and Reynolds (2006) draw attention to the challenges of automatically adapting simulations when experimental data indicates that a simulation must change. To this end, a simulation is first run to gain insight about a phenomenon. Then, this insight is used to determine what new observations should be collected, and the simulation is adapted to reflect these observations. Generalizing software to anticipate all possible ways it could change is difficult, and attempting to do so usually comes at the expense of performance, as well as makes the code unmanageably complex (Parnas, 1979). However, software engineers have observed that the problem of software adaptation can be simplified by taking advantage of the flexibilities and constraints of a simulation at the same time. Without flexibility, automatic adaptation is impossible because there is no way to know which alternatives should be considered. Without constraints, automatic adaptation is infeasible because there are too many alternatives to consider them all in a timely fashion. Carnahan and Reynolds, therefore, propose a semi-automated adaptation approach that exploits the flexibility and constraints of model abstraction opportunities to automate simulation adaptation. Although their study does not involve manual modification of the code or application of optimization methods which can make the software extremely complex to control, it is still in need of human

intervention to determine the most likely places of the code in need to be changed. In our research, changes in level of detail of data acquisition and the choice of certain parameters over others allow the automatic multi-fidelity adaptation in the simulation model. In addition, the development of interfaces to physical devices and the creation of an infrastructure to support the communication and data requirements are also addressed. Through a networked sensor-driven control system and integrated grid architecture for distributed computing, a methodology is proposed for up to date data to be transferred, thereby allowing the models to be updated in real time.

Another recent contribution to DDDAS community has been made by Patrikalakis et al. (2004) where they provide an overview of a DDDAS for rapid adaptive interdisciplinary ocean forecasting. Information technology allows the development of an Internet-based distributed system that enables the seamless integration of field and remote observations, as well as data assimilation in order to effectively estimate the uncertainties in oceanic fields. They also present illustrative examples of interdisciplinary modeling and forecasting and progress to date towards an integrated ocean science DDDAS. In their study, automated adaptive modeling and sampling, fully coupled physical-acoustical-biological oceanography and grid computing application aspects are recommended for further research most of which comprise the focus of this dissertation.

Smith et al. (1994) and Son et al. (2002) proposed a simulation-based shop floor planning and control system, where the same simulation model (executing in the fast mode) used at the planning stage after some modification is used as a real-time task

generator (real-time simulation) during the control stage. In their approaches, the real-time simulation drives the manufacturing system by sending and receiving messages using socket-based communication links. While the use of real-time simulation as a task generator in their approaches is similar to the previously proposed DDDAS concepts to some extent, the adaptive simulation scheme proposed in our research steering the measurement process for selective data update and incorporating the real-time dynamic data into the executing simulation model is novel. Such adaptivity allows us to save computational power usage while keeping the model accurate enough by wisely drawing conclusions via the embedded algorithms which were developed in this research.

2.2.3 Simulation-based Control

Among all of the analysis tools explained so far, most of the deterministic and stochastic models seek to prototype the supply chain systems for strategic and/or tactical planning purposes since they cannot delve into much detail in these models. When they do, the payback becomes tremendous in terms of the computational resource usage and efficiency. Simulation models, on the other hand, especially distributed simulations, are able to represent the actual real system with the highest detail. Hence, they are widely used for operational planning and/or shop floor control purposes. The models built to perform what-if analysis to determine values of design variables, control strategies and performance estimation are usually discarded after the initial plans are finalized. Wu and Wysk (1989) and Drake and Smith (1996) have presented the expanded role of

simulation to include “real-time” scheduling as part of intelligent simulation to dynamically select scheduling policies based on real-time shop floor status.

Son et al., (2002) use discrete event simulation not only as a traditional analysis and evaluation tool but also as a task generator that drives shop floor operations in real time. Their control simulation reads process plans and master production orders from external databases that are updated by a process planning system. Multipass scheduling algorithms created by Wu and Wysk (1989) attempt to solve the scheduling problem by selecting the best dispatching rule, among its alternatives. This multipass framework is applied to the proposed simulation-based control environment and implemented. Their framework is comprised of two models where one is used primarily as a real-time task generator for the shop floor and the other is used as the look-ahead manager. Once this manager received a request, it gets several fast-mode simulations runs stored in a file. This file is then sent to the real-time simulation for execution of the best control strategy on the system. Overall system implementation conducted with Arena® simulation software gives promising results. However, the level of detail between the scheduler and controller is different since in scheduling dynamically-changing data are not considered. Providing such schedules to real system may cause serious problems such as collisions or deadlocks in the shop floor. A challenging extension of this study may be determining which approach should be taken under what circumstances.

Further studies on shop floor control have been conducted by Son and Wysk (2001) in an extended manner. Simulation models for shop floor control need to be as detailed as possible since real time action will be affected if any error occurs. In order to

build such detailed models in a compact way, knowledge regarding the system to be modeled and skillful modeler in terms of the simulation language as well as statistical methods is needed. Even for the experienced simulation modeler, the process can be very time-consuming. To resolve this problem, Son and Wysk (2001) present a structure and architecture for automatic simulation model generation of very detailed simulation models intended to be used for real-time simulation-based shop floor control. Although many researchers have worked on automatic simulation model generation, this study is novel for being the only one focusing on generation of simulation models for real-time shop floor control. Generated simulations can be used both for traditional system analysis and manufacturing system control by sending and receiving messages using an Ethernet communication link to a high-level task executing system. Models (in Arena®), are automatically generated from a shop floor resource model (MS Access 97®) and a control model (message-based part state graph). The overview of the control is shown in Figure 2.7 where Arena® simulation model obtains part orders and part process plans using an SQL connection to an interactive database. The BigE receives instructions from the simulation and sends messages to machine level controllers based on the current system status. Implementation is performed for 6 different manufacturing configurations and promising preliminary results are obtained. However, the study is limited to address only discrete part manufacturing systems that operate with a single part unit loads. Also, the movement of parts is assumed to require material handlers. Finally, machine and robot capacities are also assumed to be one. Further research may focus on relaxing these strict assumptions.

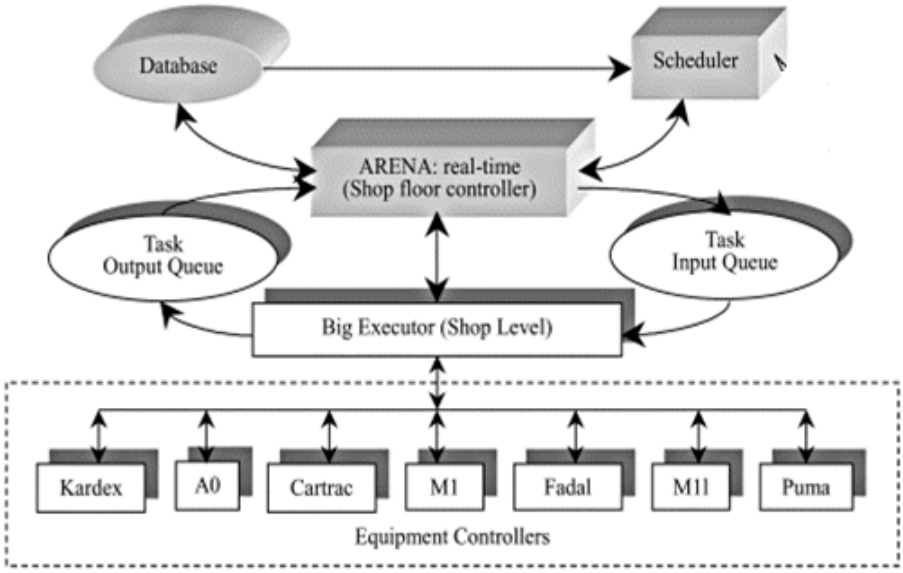


Figure 2.5: Architecture for simulation-based shop floor control (courtesy from Son and Wysk, 2001)

Madhusudan and Son (2005) perform even further studies where they develop a simulation-based approach for dynamic process management at web service platforms. They first draw attention to the volatile and open nature of the web during execution of composite services at service platform and how significant is the delivery of reliable and acceptable performance results. Then, they identify the clear analogy between the discrete manufacturing environment and web services process environment (see Table 2.1). Thirdly, they propose to solve the above mentioned problem by using an online simulation for scheduling service requests at service platform and a look-ahead simulation to provide guidelines for service execution in the volatile system environment.

Their integrated simulation-based control and execution module consists of five components. Given service plans from the planning module and current state information

from the web services world, look-ahead manager develops a control policy. The statistical analyzer analyzes simulation results. Then, the decision-maker uses the control policy to make real-time decisions and calls the decision-planner when the actual performance deviates substantially from the predicted performance and when significant, unforeseen events, such as service provider failures, network failures, and new request arrivals, occur. The decision-maker sends the appropriate messages to the executor dispatch mechanism. Service responses are received at the status manager and processed appropriately. As it is also mentioned in their own self-criticism, future studies may involve improving the interface between the simulation and the decision-maker modules in the service execution component, exploring the role of simulation in guiding re-planning versus re-execution, dynamically updating the simulation and underlying distributions (based on change points in the real-world), and performing large-scale real-world studies (including stress testing the implementation).

Another simulation based control perspective is carried by Schwartz at al. (2006) when they study inventory management in supply chains. They establish simulation-based optimization framework involving simultaneous perturbation stochastic approximation (SPSA) for optimally specifying parameters of internal model control (IMC) and model predictive control (MPC)-based decision policies under supply and demand uncertainty. SPSA realizes the same level of accuracy compared to other methods with fewer evaluations of the objective function. The SPSA algorithm is comprised of 5 main steps: 1) initializing input vector, 2) generating perturbation factor, 3) evaluating the objective function, 4) approximating the gradient and 5) updating the

estimate. Their results obtained from case studies involving semiconductor supply chain demonstrate that safety stock levels can be significantly reduced while maintaining satisfactory operating performance in the supply chain.

Among various supply chain elements, warehouse system holds a significant place as it takes care of fluctuation of demands from customers and provides just in time delivery of materials. Ito and Abadi (2002) develop an agent based model for warehouse system, which is called AWAS (Agent based model for Warehouse System). It facilitates the information exchange between a warehouse system and its customers and suppliers. In addition, it also arranges AGV assignments and flow of materials inside the warehouse for improved performance of material handling system. AWAS is comprised of three subsystems where ACS is responsible for communication between distributed customers and suppliers, AIPCON is to assign inventories to be sent to customers and AMATH is for management of AGV's. The prototype system based on AWAS is implemented using Java and the initial results show the validity of the overall framework. However, a mechanism for negotiation of AWAS with its suppliers and customers, selection of appropriate supplier among alternatives, effective job-allocation to AGV's and scheduling jobs of each AGV is among the future works of this study.

In this dissertation, we make use of both traditional fast mode simulations and recently studied real time simulations. The traditional fast mode simulation models are proposed in order to facilitate the analysis of the system under investigation once (or if) the system specific schedules are changed. On the other hand, the real time simulation

model involving decision making units (*i.e.*, algorithms) is proposed to facilitate the simulation-based online control of the considered system.

Table 2.1: Selected works on simulation

Authors	Year	Application Industry	Problem Assessed	Output	Solution Methodology	Limitations/Areas of Improvement
Bhaskaran S.	1998	Automobile Supply Chains	Instable Production Schedules and Fluctuating Inventory	Methodology to analyze specific real-world supply chains for areas of improvement	Simulation with a user-friendly software originally developed to GM's specification, by David Kletter and Steve Graves of MIT	• Lack of adequate impact analysis on capacity constraints and material shortages • Considered material strategies are limited to MRP and Kanban only.
Van der Zee D.J. and Van der Vorst J. G.A.J.	2005	Chilled Salads Supply Chain	Flaws of current modeling approaches hindering explicit control rules and member interaction	Generic (high-level and software independent) modeling framework for supply chain simulation	Object-oriented format with three key elements: agents, jobs and flows	• For strong mathematical background, modeling framework proposed can be linked with Petri Nets • Artificial Intelligence may help increasing modeling efficiency and effectiveness
Forget P.,D'Amours S.and Frayret J.M.	2008	Lumber Industry	Supply chain agility and synchronization	Multi-behavior planning agent model using different planning strategies	Agent-based simulation for planning	• Multi-behavior agents ability to learn may be improved through experience and incorporated algorithms
Petrovic D.	2001	Serially Linked Supply Chains	Assessment of Supply Chain Uncertainty	Order-up-to levels for all inventories in supply chain system	Supply Chain Simulation (SCSIM) tool comprised of fuzzy analytical model and simulation model	• Supportive experimentation needs to be done with the inclusion of continuous review inventory models and other sources of uncertainty in a supply chain for validation purposes
Iannone R., Miranda S. and Riemma S.	2006	General Supply Chains with Several Echelons	Time and Information Exchange Synchronization in Distributed Supply Chain Simulations	Architecture (SYNCHRO) which is able to synchronize distributed simulation models both time-wise and information-wise simply and securely	Models created in Visual Basic 6.0 together enabling server communication through TCP	• Overload of data exchange requests from several federates cause considerably slow down in model • The completeness and speed of the framework should be concisely established in a real supply chain environment
Son Y.J., Joshi S.B., Wysk R.A., Smith J.A.	2002	Penn State CIM Lab Shop Floor	Scheduling	Real-time shop floor task generator	Discrete event simulation implemented in Arena® (real-time simulation for controller, fast mode simulation for scheduler)	• Level of detail between the scheduler and controller is different since in scheduling buffers and material handling are not considered. Providing such schedules to real system may cause serious problems such as collisions or deadlocks in the shop floor

Authors	Year	Application Industry	Problem Assessed	Output	Solution Methodology	Limitations/Areas of Improvement
Son Y.J, and Wysk R.A	2001	Discrete Part Manufacturing Systems	Time consuming modeling of detailed simulations for real-time shop floor control	An architecture for automatic simulation model generation of detailed simulation models for real-time shop floor control	Discrete event simulation implemented in Arena® with a resource model (MS Access 97) and a control model (message-based part state graph)	• The study is limited to address only discrete part manufacturing systems that operate with a single part unit loads • The movement of parts is assumed to require material handlers • Machine and robot capacities are also assumed to be one Further research may focus on relaxing these strict assumptions
Madhusudan T. and Son Y.J.	2005	Web Service Platforms	Dynamic Process Management	A simulation-based approach for dynamic process management at web service platforms	Service composition is enabled using the Hierarchical Task Network (HTN) planning technique ,the execution module is implemented as JAVA servlets and simulation engine is based on Arena®	• Improving the interface between the simulation and the decision-maker modules in the service execution component, • Exploring the role of simulation in guiding re-planning versus re-execution, • Dynamically updating the simulation and underlying distributions • Performing large-scale real-world studies
Schwartz J.D., Wang W.L. and Rivera D.E.	2006	Semiconductor Supply Chains	Process Control Policies for Inventory Management	A simulation-based optimization framework involving simultaneous perturbation stochastic approximation (SPSA)	Derived internal model control (IMC) and model predictive control (MPC) together with SPSA optimization method	• Further experimental analysis and their contribution might be performed for various supply chain systems
Ito T. and Abadi S.M.M.J.	2001	Warehouse systems	Inventory Planning in Warehouse	Agent-based model for warehouse system	Prototype system based on AWAS is implemented using Java	• A mechanism for negotiation of AWAS with its suppliers and customers, • A mechanism for selection of appropriate supplier among alternatives, • A mechanism effective job-allocation to AGV's and scheduling jobs of each AGV may be among the future works of this study

2.2.4 Distributed Simulations and Their Synchronization

Recently, constructing distributed simulations (federations) integrating existing federates is widely discussed in the literature (Kuhl et al., 1999; Fujimoto, 2000; Riddick and McLean, 2000; Venkateswaran and Son, 2004; McGinnis et al., 2005). Here, the utmost benefit (in terms of reduction of simulation execution time) can be gained if the federates (individual simulations) comprising the federation are designed or selected to maximize parallelism. To select the most promising alternatives, McGinnis et al., (2005) evaluated the design alternatives (generated using a hierarchical decomposition approach and a material handling approach) based on various criteria prior to implementation. In the hierarchical decomposition approach, the decomposition design is based on the hierarchical structure of 'facility - work area - work station - equipment'. In the material handling approach, three federates are considered including the factory federate, the material handling federate, and the facility control federate. In their study, designs generated by the material handling approach computationally outperform designs generated by the hierarchical decomposition approach in a distributed setting.

Another benefit of the distributed simulation technology is its ability to allow each federate to hide any proprietary information, but still provide enough information to evaluate the entire federation as a whole. Venkateswaran and Son (2004) proposed a prototype distributed simulation system to evaluate viability of a potential supply chain, which allows potential partners evaluate whether it will be profitable for them to participate in the proposed supply chain without sharing much of their proprietary information.

However, realizing distributed simulation requires a reliable and efficient computational infrastructure. HLA/RTI (High Level Architecture/Run Time Infrastructure, IEEE Standard 1516-2000) is one of the most widely known frameworks. While it is generic and effective, its computational overhead is high, especially for a small scale federation. Therefore, Lee et al. (2008) employed web services technology to implement a subset of the HLA/RTI functions to offer a simplified set of services in a platform independent setting. Similarly, Taylor et al. (2002) introduced a generic runtime infrastructure for distributed supply chain simulation that promotes user transparency and extensible service provision.

Time synchronization, another challenge to be addressed in distributed simulation, may be enabled in three ways. First, it can be enabled by *Conservative synchronization*, where no federate advances its logical time until it is guaranteed not to receive events which are supposed to happen in its past (Fujimoto, 2000; Ayani and Rajaei, 1992; and Nicol, 1993). This type of synchronization can be realized in either centralized or decentralized manner. In the centralized case, there is a coordinator recording every partitioned simulations clock and their notification of "done" messages with their current event and letting the whole system move to the next phase all together if everybody is done. Whereas, in the decentralized case, each simulation has to send its notification of "done" messages with its current event to the other partitions and once every simulation sends this notification, they all move to their next phase. Second, it can be enabled by *Optimistic synchronization*, where a federate may compute into the future, but may also receive events that cause it to roll back state which is troublesome and can

cause computationally intense operations (Jefferson, 1985; Fujimoto, 2000; Vardanega and Maziero, 2001; and Wang et al., 2005). Finally, time synchronization among partitioned simulations can be realized via *Epoch synchronization*, where federates progress through episodes in which they exchange messages at the same time until they agree to advance together (Rathore et al., 2005). This way, each federate advances its time independently and only synchronizes at every time interval (*i.e.*, Δt). Epoch synchronization allows for a computationally lighter scheme than the conservative synchronization while enabling accurate results. In this study, an Epoch synchronization scheme is employed, where time intervals (Δt) are determined based on off-line simulation analyses. Further studies available in the literature regarding these time synchronizations attempts are summarized in Table 2.3 in terms of their applications, specific problems, contributions to the literature, and extension opportunities.

While increasing availability of network computing resources (*e.g.* high-speed networks and high performance machines) presents an opportunity for delivering good performance on a range of parallel applications including distributed simulations, developing applications to run efficiently in such an environment however is another challenge to be overcome (Weissman and Grimshaw, 1994). Omari et al. (2004) discuss an effective technique for the distribution of the processes on multi-computers to achieve several performance goal(s), such as improving the average completion time of a group of independent tasks or subtasks of a parallel program, minimizing communication delays, and/or maximizing resource utilization. In their approach, completion time is

determined by two factors: load balance and computation granularity, where load balance intends that all processors will finish around the same time. In our work, partitioning is

Table 2.2: Selected works on time synchronization of distributed simulation

Authors and Year	Application	Problem Assessed	Contribution to Literature	Extension Opportunities
Nicol, 1993	A domain composed of sites (an atomic simulatable unit). *i.e.*, a G/G/k queue is modeled as a site with k servers.	Performance of a synchronous conservative parallel discrete-event simulation protocol.	A new approach for the analysis of parallel discrete-event simulations is developed.	The analyzed conservative synchronization protocol is a simple one and can be extended to more complex protocols for its robustness test.
Namekawa et al., 1999	Road-traffic simulation using a microscopic model including both the discrete-change model and continuous-change model.	Lack of a method for decreasing synchronous processing between processors as much as possible.	A conservative time window based algorithm predicting the simulation clock allowing synchronization between a subsystem and the neighboring subsystems is proposed.	The algorithm can be refined to reduce the gap between the forecast event time and the actual one in the next sub-area, and to carry out parallel simulation using more than two processors.
Ayani and Rajaei, 1992	Feedforward networks and networks with feedback loops.	Lack of a conservative synchronization scheme where different windows may have different sizes if the nodes are advancing heterogeneously.	Conservative Time Window (CTW) algorithm for parallel simulation of discrete event systems is presented.	CTV-algorithm requires that size of application is much larger than size of hardware. Improving its performance for heterogeneous applications is an extension item.
Vardanega and Maziero, 2001	A simple federation involving several optimistic federates was developed and used where the federate state size was 200 bytes.	Some HLA services are very low-level and hard to use when building simulation models that use peculiar time synchronization schema. Also a generic rollback manager to detect causality violations is needed by optimistic simulation entities.	The addition of an extra piece of software, a rollback manager, to implement state saving and rollback management for optimistic federates in the High Level Architecture (HLA).	The demonstrated impact of the proposed mechanism on the system performance can be extended involving more extensive measurements.
Wang et al., 2005	A model based on the interoperability reference models proposed by the HLA Commercial-Off-the-Shelf Simulation Package.	Performance of an optimistic synchronization in HLA-based distributed simulations.	A rollback controller using a middleware approach to handle the complex rollback procedure on behalf of the simulation model is introduced.	Investigating a mechanism to handle simultaneous events and efficient algorithms to optimize the handling of state saving, rollback, and fossil collection is an extension item.

modeled as a group technology configuration problem considering both load balancing and computation granularity.

The load balance problem has received considerable attention in parallel and distributed applications because good solutions are critical to achieving speedups. To date, most of the existing graph emulation works do not provide systematic techniques to achieve load balance (and therefore good scaling) (Liu and Chien, 2003). A number of these works use simple hierarchical graph partitioners (Liu and Nicol, 2001), while others use randomized clustering techniques based on graph topologies (Vahdat et al., 2002) which have not been demonstrated to give broadly robust results. Also, most of them provide no automated solution, requiring users to manually partition the graph. Manual or simple heuristic approaches presented in the literature are unsuitable for large-scale graph emulations of thousands of graph entities (Liu and Chien, 2003). This has motivated our proposed research on developing a partitioned (distributed) extended DDDAMS solution which intends to minimize the interaction events (time synchronization requirements between the partitions. Other approaches on partitioning various types of simulations are detailed in Table 2. To the best of our knowledge this study is the first attempt to solve the problem of partitioning discrete event simulations in grid computing environment using a polynomial-time heuristics which considers both load-balancing and computation granularity issues (see Chapter 7 for details).

Table 2.3: Selected works on methods developed to partition simulations

Authors and Year	Main Partitioning Approach	Details of Approach	Extension Opportunities
Kim et al.,	Hierarchical	Approach is based on the	The model used in the experiments

1998	partitioning algorithm for optimistic distributed simulation of Discrete Event System Specification (DEVS) models	assumption that when a system is modeled, a modeler naturally partitions the system into a set of subsystems while considering parallelism between them. Thus, by utilizing such hierarchical structural information of models, the method performs the partitioning.	is an open queuing network and therefore includes no feedback of messages. Also, proof of the generality of the method can be extended for other formalisms apart from DEVS.
Boukerche, 2002	Simulated annealing to partitioning conservative simulation on multiprocessor machines	This method is derived by mapping the complex simulation system into a physical system and is based upon realistic estimates of computation and communication load.	Resolving long computation of the simulated annealing in case of large-scale networks is an extension item.
Bokhari, 1988	A sum-bottleneck path algorithm for optimally assigning the modules of a parallel program over the processors of a multiple-computer system	Several variants of the partitioning problem for parallel and pipelined programs are addressed. Under certain constraints on the structure of the program and/ or the multicomputer system, this problem can be solved optimally using the proposed algorithm.	An appropriate assignment graph must first be discovered prior to application of this algorithm. Each assignment graph has to be designed very carefully in order to capture all information about the problem. Applying the SB path algorithm to other problems (for generality) is an extension item.

2.2.5 Bayesian Filtering in Simulation

Real world is comprised of systems of systems in a wide range of scale. These systems can be classified as either static (time-invariant) or dynamic (time-variant). In a dynamic system, its behavior changes over time due to its inherent dynamicity, randomness and complexity (*e.g.* economics, weather, a moving object, signal processing); whereas in a static system, its behavior does not change over time (Niculescu, 2001; Middleton and Goodwin, 1988; and Zhang and Ioannou, 1996). A dynamic system considered in this study is a large-scale and complex supply chain, where we are interested in analyzing its evolving behavior to make critical decisions when necessary.

Usually, dynamic systems can be characterized by their input, state and output (observation) classes, and a state of a dynamic system can be represented using state variables, rate variables, and parameters. If a state of a dynamic system depends on only its previous state, its state sequence is called to follow a first order Markov random process, and can be represented by state space models using state and observation equations (Markov, 1971; and Meyn and Tweedie, 2009). The state space approach is convenient for handling multivariate data and nonlinear/non-Gaussian processes, and it provides a significant advantage over traditional time-series techniques such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA) and univariate transfer function models (UTF) (West and Harrison, 1997). Taking the time dependencies between the observations of a time series data into account explicitly results in residuals that are much closer to independent random values than in classical models such as linear regression or deterministic models (Commandeur and Jan Koopman, 2007). The general state (motion) and measurement (observation) equations in the state space approach (*i.e.*, Hidden Markov Model) are given in Eq. (2.1) (Arulampalam et al., 2002; and Muhlich, 2003), where $\boldsymbol{x}_k$ is a state vector at time instant k, f_x is a state transition function, $\boldsymbol{u}_{k-1}$ is a process noise with a known distribution, $\boldsymbol{z}_k$ is observations at time instant k, f_z is an observation function, and $\boldsymbol{v}_k$ is an observation noise with a known distribution.

$$\boldsymbol{x}_k = f_x(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}), \quad \boldsymbol{z}_k = f_z(\boldsymbol{x}_k, \boldsymbol{v}_k) \qquad \text{Eq. (2.1)}$$

The state equation based on $f_x(.)$ and density $\boldsymbol{u}_{k-1}$ characterizes state transition probabilities $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ for every *k*-1 given $p(\boldsymbol{x}_0)$, whereas the measurement equation based on $f_z(.)$ and density $\boldsymbol{v}_k$ describes measurement probabilities $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$.

Dynamic systems mentioned above can be categorized as either discrete or continuous based on their time and state spaces. In the state space, the state variables of a continuous system change continuously over time whereas in a discrete system, changes of state variables are limited to a number of different states. The change that occurs in continuously-changing state variables can be expressed via differential equations, and continuous-change state processes can be modeled via Brownian motion processes when the time space is also continuous. When the time space is discrete, the said changes can be expressed using different formalisms such as cellular automata and Petri-nets. On the other hand, continuous time Markov processes are mostly used to model the change in system parameters when the system is classified as discrete in state space and continuous in time space. Similarly, many other formalisms such as linear discrete time networks, cellular automata, Petri-nets, finite state machines, discrete-time Markov chains are used to model the said change when the system is classified as discrete both in state and time spaces. It should be noted that the formalisms those comprise the basis of discrete-event modeling (*e.g.* cellular automata, Petri-nets) can cover both discrete and continuous state space although they always move forward at discrete times. Moreover, as Klingener (1996) mentions, just like in practice where few systems are completely discrete or continuous, simulation models can also be a combination of both continuous-time and discrete-event models.

Xia and Lin (2003) states that in practice, combined (hybrid) models are usually built through merging continuous-time models into a discrete-event modeling framework, where the following four basic interactions occur:

- A continuous variable value may suddenly increase or decrease as a result of a discrete event.
- An initiation of a discrete event may occur as a result of reaching a threshold value of a continuous variable.
- The change rate of a continuous variable may be altered as a result of a discrete event.
- An initiation or interruption of change in a continuous variable may occur as a result of a discrete event.

Hybrid models entails less computation compared with pure continuous models while providing more accurate system representation compared with pure discrete models. On the other hand, the complexity and computational intensity of the simulation is higher when compared to that of the pure discrete event models. Usually, such cross-implementations limit the accuracies of models compared to that of pure continuous models. Therefore, DDDAMS methodology is proposed to build hybrid models carefully in order to provide necessary details of the considered system while saving from computational resources.

The initial realization of DDDAMS in Celik et al. (2010) was carried out via discrete event simulation involving three model fidelities (abstraction levels of the system model). The later enhanced approach, however, focuses on hybrid simulations in order to represent the system under study more accurately. In the proposed hybrid simulation for

DDDAMS, continuous modeling is implemented in the discrete event simulation framework leveraging all four interactions mentioned above. For instance, in a semiconductor supply chain simulation, manufacturing systems (*e.g.* flow of batches of wafers) are modeled via discrete event simulation, and the chemical processes for wafer production are represented via continuous models. Moreover, in this approach, high accuracy of this hybrid representation of the system is enabled via a substantially increased number of fidelities covering a wide range across macroscopic (purely discrete) and microscopic (purely continuous) levels. Here, in determining a system fidelity, continuous time Markov chains are used to model the overall behavior of the system components (*i.e.*, machine, cell, shop) since the time that they spend in a state changes dynamically. The selection of the fidelity should be made wisely depending on the need and the computational resource availability, which is a major purpose of the proposed algorithms in this work.

2.2.5.1 Bayesian Filters

In a dynamic system that we have discussed in previous section, a Bayesian approach uses probability distributions to describe all types of uncertainties accrued from various sources, such as 1) randomness from stochastic behaviors of simulated systems, 2) input parameters, 3) output performance measures including their random errors, and 4) unknown metamodel parameters (Chick, 2006). In that approach, Bayesian filters are designed to extract relevant information of a system via investigating the observations.

They provide a convenient way to represent uncertainty about performance of systems and input parameters. In this section, various Bayesian filtering techniques available in the literature are detailed in terms of their assumptions, advantages, and disadvantages (see Table 2.5 for the comparison of various techniques and Table 2.6 for the assessment of representative literatures for each filtering technique, respectively).

The Bayesian approach generates more enhanced performance analyses compared to those of classical models (*e.g.*, linear regression and Markov localization approaches) as it takes into account what is actually known and what is not regarding the system to be simulated during its execution. It incorporates prior information (*i.e.*, available information before a latest data is collected) into the analysis in a precise and rational manner by making use of the hidden Markovian models (which can be both discrete and continuous). In addition, it is particularly beneficial under the circumstances, where data is sparse and data collection is costly. Especially in two specific simulation cases, where 1) gaining detailed simulation data is extremely costly due to need for large numbers of data points for numerous parameters and 2) obtaining each data point is costly, a Bayesian technique allows for obtaining a desired level of precision in knowledge with the minimal amount of data. Basically, it reduces the confidence interval about the output quantity of interest by using a dependable source of prior information about that quantity. Furthermore, one can obtain significant savings in computation time by using the prior information.

In the Bayesian framework (Chick, 2006), *prior distributions* (*i.e.*, $\boldsymbol{p}(x_0)$) for unknown quantities (such as unknown outputs, unknown input parameters, or unknown

metamodel parameters) are defined to specify initial uncertainty about unknown parameters; *likelihood models* ($\boldsymbol{p}(\boldsymbol{z}_k|\boldsymbol{x}_k)$) to relate unknown parameters to observable data, and *numerical tools* (*i.e.*, algorithms or filters) to update beliefs about unknown quantities as data becomes available using *Bayes' rule* to obtain posterior distributions for unknown quantities. The posterior probability ($\boldsymbol{p}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$) of $\boldsymbol{x}_k$, based on all available measurements up to time *k* (abbreviated as $\boldsymbol{z}_{1:k}$), summarizes uncertainty about $\boldsymbol{x}_k$ via the likelihood model and prior distribution and performs probabilistic state estimation for a dynamic system assuming that an initial probability density function (prior distribution, $\boldsymbol{p}(x_0)$) is known. Since in Bayesian filters, all available information is used via the probability distribution functions, an optimal estimate can be found theoretically for any criterion within the dynamic system.

In our study, we intend to estimate the posterior distribution of a dynamic supply chain system, which will then help us compute various estimates for the current state or for future observations including an expected value, median, modes, confidence interval, and kurtosis. For instance, let the state transition function $\boldsymbol{f}$ be any arbitrary, integrable function that can depend on all components of a state $\boldsymbol{x}$ (*i.e.*, a set of variables defining a state $\boldsymbol{x}$) and on the whole trajectory in the state-space. Then knowing the posterior distribution, an expected value of $\boldsymbol{f}$ can be computed by Eq. (2.2). However, in order to estimate the state of a dynamic model, which evolves over time, the Bayesian filtering should be applied in a recursive manner.

$$E[\boldsymbol{f}(x_{0:k})] = \int \boldsymbol{f}(x_{0:k})\boldsymbol{p}(x_{0:k}|z_{1:k})\, dx_{0:k} \qquad \text{Eq. (2.2)}$$

Recursive Bayesian filters enable a sequential update of previous estimates as new data becomes available. In any update, they allow for batch processing of data (computation with all readily available data in one step) by which not only the state of a dynamic system is derived faster, but also rapid adaptation to changing signal characteristics are enabled through on-line processing of data. Recursive Bayesian filters (Wan and Van der Merwe, 2000; Liu, 1996; Arulampalam, 2002; Doucet et al., 2000; and Ridgeway and Madigan, 2003) essentially consist of two steps (see Figure 2.6): 1) the prediction step where the next state is predicted using a probability density function (or probability mass function) by deducting $\boldsymbol{p}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$from $\boldsymbol{p}(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$ and 2) the update step where the likelihood of current measurement is combined with the predicted state by deducting $\boldsymbol{p}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ from $\boldsymbol{p}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$ and $\boldsymbol{z}_k$.

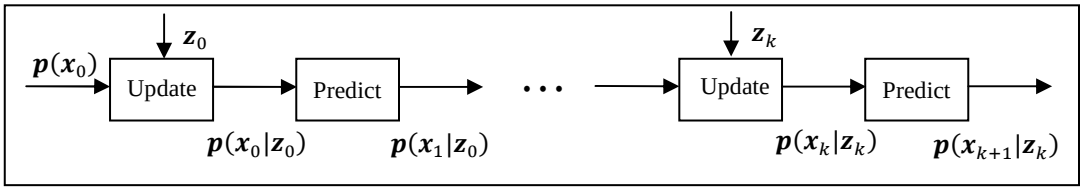


Figure 2.6: Two major steps (update and prediction) in recursive Bayesian filtering

The structure of the update equation is given by Eq. (2.3), where prior is given by a prediction equation, likelihood is given by a measurement model, and evidence is the normalizing constant in the denominator.

$$posterior = \frac{likelihood.prior}{evidence} => p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \qquad \text{Eq. (2.3)}$$

where $p(z_k|z_{1:k-1}) = \int p(z_k|x_k)\, p(x_k|z_{1:k-1}) dx_k$.

2.2.5.2 Kalman Filters

Bayesian analysis, in principle, can be applied to various situations, in particular for cases, where simple closed-form (*i.e.*, natural conjugates) results are available. In a linear state space model, when the posterior distribution of an unknown at time $k-1$, $p(x_{k-1}|z_{k-1})$ and the measurement noise are both Gaussian, then the new posterior $p(x_k|z_k)$ also becomes Gaussian and can be computed exactly in a linear state space using Kalman filter. In the literature, Kalman filters have been effectively applied to various research areas including economy, radar, computer vision, navigating and tracking a target, robot localization, and audio and video localizations (Cheung, 1993; Bahmani and Brown, 2004; and Gehrig et al., 2005).

While Kalman filter presents an exact optimal solution for the posterior distribution, required assumptions (*i.e.*, Gaussian noise process and linear state space model) are highly restrictive and not realistic for many real world systems. When the process to be estimated or the measurement relationship to the process is non-linear, Extended Kalman filter (EKF) can be used to approximate the exact optimal solution. The EKF, basically, linearizes the non-linear process model about the current first two moments of the state (*i.e.*, mean and covariance). Unfortunately, the linear approximators to the nonlinear functions (*i.e.*, derivation of the Jacobian matrices) can be

Table 2.4: Comparison of Bayesian Filters

	Kalman Filter (Weng et al., 2006; Linsker, 2008)	**Extended Kalman Filter** (Julier et al., 1995; Ozbek and Ozlale, 2005)	**Unscented Kalman Filter** (Wan and Van der Merwe, 2000; Kandepu et al., 2008)	**Grid Based Filter** (Martinez et al., 2007)	**Particle Filter** (Liu, 1996; Arulampalam, 2002; Doucet et al., 2000; Ridgeway and Madigan, 2003)
Assumptions	• Gaussian noise process • linear state space model	• Gaussian noise process and non-linear state space model • all transformations are quasi-linear	• Gaussian noise process • non-linear state space model	• state space is discrete • number of different states is limited	• any state-space model (Gaussian, Non-Gaussian, linear, non-linear)
Solution	• exact, optimal	• suboptimal, approximate	• suboptimal, approximate	• exact, optimal	• suboptimal, approximate
Drawbacks	• assumptions too restrictive	• linear approximators to the nonlinear functions can be complex • linearizations can lead to filter instability if the time step intervals are not sufficiently small • difficult to implement and tune	• fails if we have bimodal / multimodal pdfs and/or heavily skewed pdfs	• as dimensionality of state space increases, computational cost of the approach increases dramatically • state space must be predefined and, therefore, cannot be partitioned unevenly to give greater resolution in high probability density regions, unless prior knowledge is used	• degeneracy problem: after a few iterations, most particles have negligible weight (the weight is concentrated on a few particles only)
Sensor Variety	• low	• low	• medium	• high	• high
Ease of Implementation	• medium	• low	• high	• medium	• high

complex causing implementation difficulties and can lead to filter instability if time step intervals are not sufficiently small (Julier et al., 1995).

2.2.5.3 Sequential Monte Carlo Methods (Particle Filters)

Sequential Monte Carlo methods update the estimates of posterior distributions as new data arrives. Particle filters are defined as sequential Monte Carlo methods based on point mass (or "particle") representations of probability densities, which can be applied to any state-space model and generalize the traditional Kalman filtering methods (Arulampalam et al., 2002). Major comparison of these traditional Kalman filtering techniques is provided in Table 2.5. Although the aforementioned filters are not explained in a detail, particle filter is explained in a detail in this section as it is a major technique leveraged in our work.

If a natural conjugate prior does not exist or there is not a suitable way of representing the available prior knowledge in a particular case, the Bayesian analysis can be carried out by numerical integration, which is computationally time consuming. In such cases, Particle filters represent the posterior probabilities by a set of randomly chosen weighted samples, where random selection of samples is enabled by Monte Carlo simulations (to approximate integrals). The approximation depends on the ability to sample from the original distribution depicting a system behavior. Almost sure convergence to a true probability density function is assessed as the number of samples increases (Muhlich, 2003).

Particle filtering uses importance sampling (sequential) to filter out those "particles" that have the least posterior mass after incorporating the additional data. Importance sampling is a general Monte Carlo method for computing integrals. When a sampling mechanism is not readily available for the "target density", but one is available for another "sampling density", we can use importance sampling where the relevant distribution is contained in the pairs of samples and weights (*i.e.* particles). It has been demonstrated in the literature (*e.g.*, Liu, 1996; Arulampalam et al., 2002; Doucet et al., 2000; and Wan and van der Merwe, 2001) that the particle filter approach greatly reduces the number of data accesses while maintaining accurate parameter estimates.

During the past two decades, several Sequential Monte Carlo algorithms have been developed to attack different problems from various aspects. Some of these algorithms emphasize solely importance sampling or Markov Chain Monte Carlo (MCMC) sampling, whereas some others emphasize their combination to obtain faster and better results. Practical applications of these studies include target tracking (Gordon et al., 1993), enhancement of speech and audio signals in digital environment (Godsill and Rayner, 1998) and integrating multi-resolution metrology data (Xia et al., 2008). As an example of these studies, Vo et al. (2005) first establishes a relationship between finite set statistics and conventional probability. Then, they build a principled and computationally tractable Sequential Monte Carlo multi-target filter that makes use of data coming from various sensors. Here, the importance distributions should be chosen to minimize the conditional variance of the weights. Although their initial results look

promising, the viability of the proposed approach needs to be tested in real and complex applications (see Table 2.6 for comparison of the above mentioned studies).

As mentioned earlier, in our work, we are interested in preventive maintenance and part routing scheduling problems in supply chains. Effective online scheduling mechanism based on DDDAM-simulations requires estimation of the state of a system that changes over time using a sequence of noisy measurements made on the system. In this study, we attempt to estimate the posterior probability distribution of the states of our dynamic system (*i.e.*, manufacturing supply chain) in the best possible way given measurements. Continuous-time Markov chains (discrete state space) are leveraged to model the states of the components of the dynamic systems considered in this study. This estimation, which is made via Sequential Monte Carlo method, is crucial as it enables near optimal fidelity selection in the proposed DDDAM-simulations.

Table 2.5: Selected works on Bayesian filters

Authors	Year	Application	Problem Assessed	Output	Benefits	Limitations/Areas of Improvement
Andradottir and Bier	2000	Simulation models	Application of Bayesian ideas to simulation	Bayesian generalization of the confidence interval approach to be used when data on actual system are not available	• Bayesian ideas used in verification and validation of simulation models (output analysis) and input analysis	• Results are checked by their conformance to prior expectations/distributions, hence approach is inapplicable to cases where they are not known
Ridgeway and Madigan	2003	Modeling of web traffic and robotics	Bayesian analysis of massive datasets	Methodology for making Bayesian analysis of massive datasets feasible while using importance sampling and rejuvenation ideas	• Allows to stop when parameter uncertainty drops below a tolerance limit • Parallelization is possible with ease • Convergence of MCMC sampler	• More empirical work needs to undergo to test its limitations • Approach might not be applicable to high dimensional applications where throwing data away might be costly
Xia , Ding and Mallick	2008	Modeling of multilevel or relational data	Integrating multi-resolution metrology data	Bayesian hierarchical model for integrating multi-resolution metrology data	• Enables use of both low and high resolution data • Saves from computational resources while predicting accurate results (efficient)	• Misalignment between data points of different resolutions should be accessed in detail • Study can be extended to involve various resolution levels (low, medium, high etc.)
Doucet, Godsill, and Andrieu	2005	Multi-sensor Multi-target Filtering	Establishing the relationship between finite set statistics (FISST) and conventional probability that leads to SMC multi-target filter	SMC implementation of the probability hypothesis density filter	• Demanding task of computing probability densities of random finite sets is achieved via FISST • A principled and computationally tractable SMC implementation of the Bayes multi-target filter is developed	• Viability of the proposed approach needs to be tested in real applications • Importance distributions should be chosen so as to minimize the (conditional) variance of the weights
Linsker	2008	Neural Networks (NN)	Learning via Kalman prediction or control in recurrent Neural Network algorithm	Artificial neural circuit and algorithm that operates in parallel systems consisting of simple processors and has learning capability via Kalman prediction and control	• Circuit is built to enable comparison with aspects of biological neural networks, particularly in cerebral cortex (CC) • Design enables analyzing the restrictive effect of computational tasks on the resulting NN architecture, circuitry and signal flows	• The computational tasks considered are in primitive stage • Association between certain types of computational tasks and specific architectural features in NN's can be studied in broader context • CC signal flows and their sequencing used in the study as well as CC connectivity are incomplete

CHAPTER 3

SEMICONDUCTOR MANUFACTURING SUPPLY CHAINS

In this section, we first discuss two different types of supply chain configurations (collaborative and competitive) and explain how the proposed DDDAMS methodology can be applied to them. Next, we provide an overview on semiconductor industry, manufacturing processes, and semiconductor supply chains, which have been selected as a specific application to illustrate and demonstrate the proposed work in this dissertation.

Supply chain systems, in general can be of two types: 1) collaborative or 2) competitive. Collaborative supply chain systems are usually comprised of echelons, which belong to the same company, and therefore sharing of information is not constrained by security issues and/or confidentiality standards among the echelons. On the other hand, competitive supply chain systems are composed of echelons, which in general belong to many different vendors or companies. Construction of dynamic data driven simulations (using grid computing) for those different types of supply chains involves two major differences. First, in simulations of collaborative supply chain systems, as the level of the fidelity increases, the sensor data accessed from the other echelons of the supply chain increases as well without any limitation on access to data (see Figure 3.1(a)). However, in simulations of the competitive supply chain systems, access to the required sensor data is restricted by the regulations and confidentiality policies of the other competitors (see Figure 3.1(b)). Second, in collaborative supply chain systems the computational resources from the partners are pooled and then by help

of grid computing, these resources are distributed among the elements as needed; whereas in the competitive supply chains, computational resources are being shared only among particular parties who are registered for collaboration to some extent. Hence, in terms of managing the computational resources, a single resource manager is adequate for all of the partners of the collaborative supply chain while distinct resource managers are needed for each group of parties that share the resources in the competitive supply chain systems. In this dissertation research, collaborative semiconductor supply chain systems are used to illustrate the proposed approach. However, the proposed framework can be applicable to both configurations (and to more general cases) when the distinctions explained above are taken into consideration.

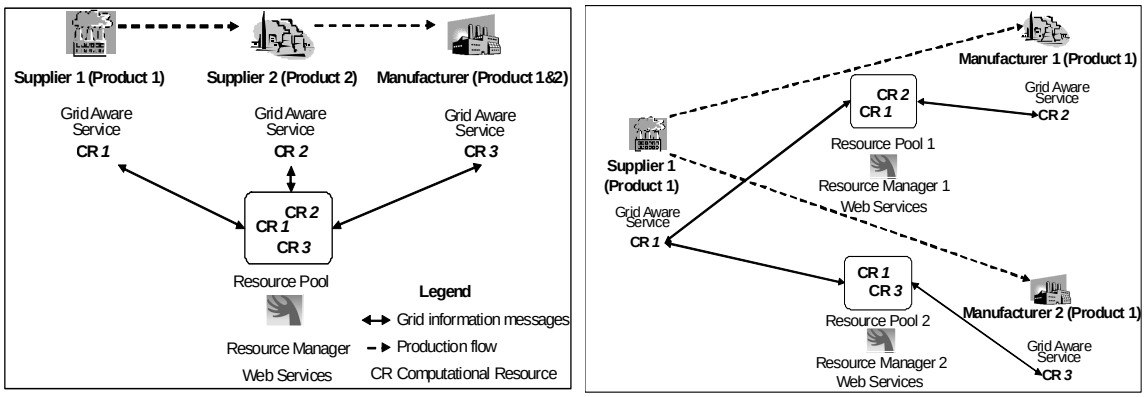


Figure 3.1 (a): Collaborative supply chain; (b) Competitive supply chain

3.1 Semiconductor Manufacturing and Supply Chains

Semiconductor industry is formed based on the operations related to the design and fabrication of semiconductor devices. Semiconductor devices have become a

significant part of our lives and their use has increased dramatically during the last 15 years. Today, semiconductor devices are used in computers, phones, automobiles, military applications, and manufacturing industries, and their further use is predicted to grow larger in next several years, and most probably in decades. The semiconductor industry was formed around 1960's, and outgrown to around $300 billion market as of 2009 according to the KPMG report. Intel Corporation has dominated the worldwide market with around 13% of the market share followed by Samsung electronics with a 7% market share, Toshiba Semiconductors with 4%, Texas Instruments with 4% and STMicroelectronics with 4%.

Overall, the semiconductor industry stands out from other manufacturing industries due to its strong affect in the United States as well as world economy. It constitutes the backbone of technological progress for the whole electronics value chain. However, this requires a rapid change and constant improvement in production performance in the market (*i.e.*, short life cycle of products). For instance, the size of a wafer has increased from 50mm to 300mm in diameter from 1965 to 2000, where the number of circuits (also known as die or chip) integrated on a wafer has increased from 2 to more than 1,000,000 in about the same timeline (see Figure 3.2 and Table 3.1).

Table 3.1: Circuit integration of semiconductors (courtesy from Quirk and Serda, 2001)

Circuit Integration	**Semiconductor Industry Time Period**	**Number of Components per Chip**
No integration (discrete components)	Prior to 1960	1
Small scale integration (SSI)	Early 1960s	2 to 50
Medium scale integration (MSI)	1960s to Early 1970s	50 to 5,000

Large scale integration (LSI)	Early 1970s to Late 1970s	5,000 to 100,000
Very large scale integration (VLSI)	Late 1970s to Late 1980s	100,000 to 1,000,000
Ultra large scale integration (ULSI)	1990s to present	> 1,000,000

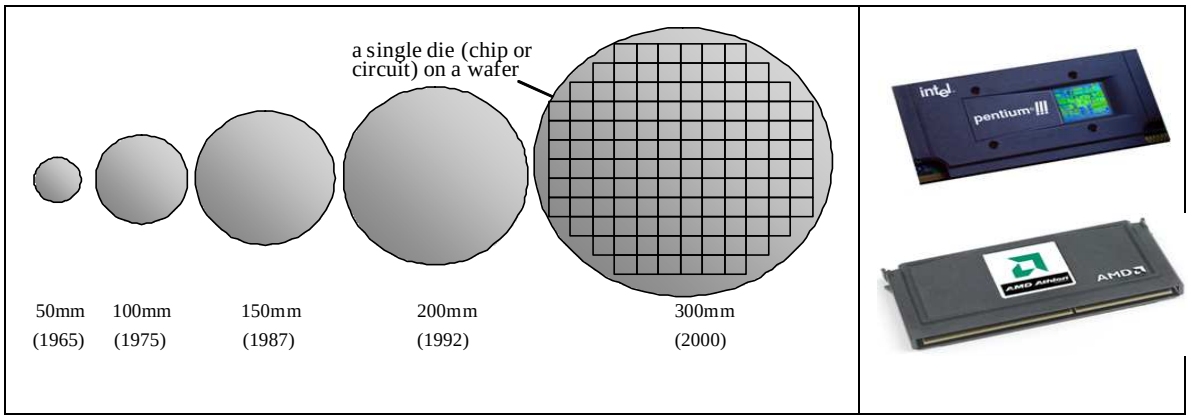


Figure 3.2: Microprocessor chips (courtesy from Intel corporation and Advanced Micro Devices)

A brief overview of the considered semiconductor supply chain components are depicted in Figure 3.3 (a). The first echelon is a wafer fab, where the raw silicon wafers are formed. The wafers are then sent to the semiconductor die fabs, where multiple layers of circuits are developed on the silicon wafer, and they are cut into individual chips called dies. The dies are transferred to an assembly and packaging fab to be packed into integrated circuits. In the considered supply chain, we aim to find best possible preventive maintenance schedules and part routing schedules within a die manufacturing fab in particular while its extended information flow network covers both the downstream and upstream members of the same supply chain. Figure 3.3 (b) depicts the highly re-entrant flow (a major cause for dynamicity in the work area) among the following major areas in this die manufacturing fab:

- Diffusion – injection of ions/atoms into the silicon wafer
- Photo – use a mask to embed a circuit onto the impure silicon wafer
- Etch – dissolve unexposed silicon layers to create a circuit
- Metals – deposit metals to enable contacts and conductive paths
- Probe – test area to check quality of die on wafer

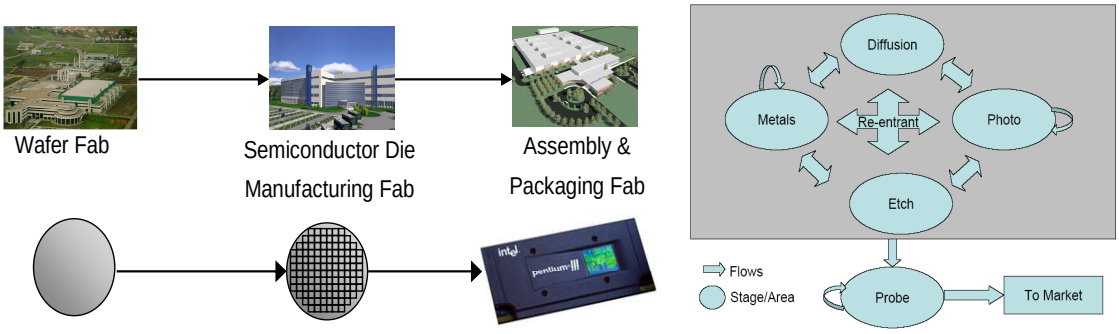


Figure 3.3: (a) Semiconductor supply chain; (b) Re-entrant flow in a die fab

The die manufacturing fab considered in this study consists of ten production cells each with ten machines (100 machines in total), where each machine utilizes nine measurement sensors of various types. The facility manufactures six product families (see Table 3.2). Each family is assumed to have different process flow. It should be noted that the processes mentioned above can be repeated up to 20 times when producing a single wafer depending on the complexity of the circuits involved. The process plan itself varies across each of the product types. Wafers are transported between machines in lot boxes with capacity of 24 wafers. Each process is carried out in one of the ten areas and is served by a central maintenance team. There are specific area experts within

the team who cater to specific areas such as diffusion. Each area is responsible for avoiding possible accumulations or rapid increases in its own work-in-process (WIP). These experts are also responsible to reduce overall WIP in the facility, which reinforces the importance of the communication within the maintenance team in order to prevent avoidable delays due to machine break-downs through preventive maintenance (PM).

Table 3.2: Different product families with re-entrant flows

Process	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Product Family 1	D	M	M	Ph	E	M	D	E	Pr											
Product Family 2	D	Ph	E	D	M	E	Pr	Pr												
Product Family 3	D	Ph	D	M	M	E	Ph	Ph	D	E										
Product Family 4	D	E	Ph	M	D	Ph	E	M	M	E	Pr	D	M	Ph	M	E	Ph	M	E	Pr
Product Family 5	D	M	M	Ph	E	M	D	Ph	E	Pr	Pr									
Product Family 6	D	E	Ph	M	Ph	Ph	D	E	M	M	D	E	Pr							
Diffusion (D); Metals (M); Photo(Ph); Etch (E); Probe (Pr)																				

3.2 Specific Problems of Our Interest in Semiconductor Supply Chain

Preventive maintenance (PM) schedules establish specific time frames for machine repairs in order to avoid unscheduled breakdowns. Such scheduled downtimes have lower costs associated with them compared to those of the unscheduled downtimes. An effective PM schedule can therefore greatly reduce operational costs of a facility since it impacts the occurrence of machine break-downs, deadlocks, setup costs, and associated WIP costs. This way, PM is scheduled with the objective of minimizing

maintenance and associated costs for all equipment over time. This interdependence between maintenance policy and costs leads to complex interactions between the other components in the supply chain and the machine states.

In this dissertation work, the DDDAMS-based preventive maintenance scheme is built for a die manufacturing fab in the semiconductor manufacturing supply chain (see Chapter 3.1). Afterwards, part routing schedules are developed based on the updated preventive maintenance schedule and machine availability. Current PM schedules are built from historic data and followed over prescribed periods of time. However in a dynamically changing environment, static operations and PM schedules derived from historical data are not efficient and flexible. Dynamically changing equipment status and facility conditions, varying priorities and sizes of customer orders result in exposing a number of deficiencies in this system. This justifies the need for dynamic generations of a near optimum preventive maintenance schedules based on the most updated data. To meet certain performance criteria under a particular system configuration such as better utilization of facility machinery and fulfillment of customer orders with lesser amount of lead times, it is necessary to create PM schedules that use the most current data and direct job orders to right locations at right times in the shop floor by generating and adapting their routing plans depending on the dynamic manufacturing environment. Exemplary pictures of the technicians (who also constitute the maintenance team) monitoring wafers in an automated wet etch tool and monitoring the wafers' progress through the fab using automated measurement tools are provided in Figure 3.4 by the photo courtesy of Intel Corporation.

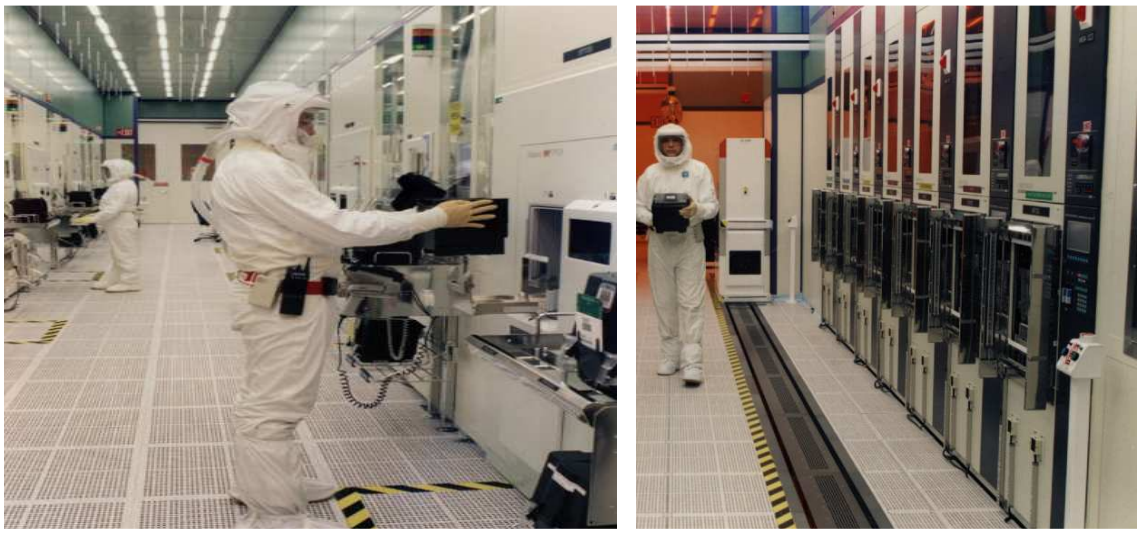


Figure 3.4: Technicians monitor wafers in an automated wet etch tool and monitor the wafers' progress through the Fab using automated measurement tools (photo courtesy from Intel Corporation)

3.3 Considered Supply Chain Performance Measures

The past literature presents many metrics of supply chain performance each of which helps us identify and understand different functioning or processes of complex supply chains. These metrics can be mainly classified into two classes of metrics which are namely, 1) functional performance measures and 2) process performance measures. Functional performance metrics are the ones which are solely specific to functions of different units in a supply chain such as purchasing, machining, distribution, and sales and marketing without any cross-functional considerations. Process performance measures on the other hand, maps the performance of the entire process of servicing or manufacturing to key metrics. To this end, functional measures provide only a partial picture; hence functional excellence does not imply process excellence. In addition,

function-based optimization can be disastrous in many cases. Process performance measures are also composed of two sets of metrics including financial process measures and operational, non-financial process. Financial measures such as market share, stock, valuation, profits, return of income, and inventory cost status are lagging metrics as their values are determined as a result of past decisions. Hence, the earliest update of such measures takes at least months. Operational, non-financial measures are on the other hand, excellent indicators of process health because their values are determined as a result of latest decisions made within the supply chain and their updates can be obtained as short as weeks, days, and even hours. Therefore, this dissertation study focuses on operational, non-financial supply chain process performance measures rather than functional performance measures. Table 3.3 depicts the most practically used operational, non-financial measures.

Table 3.3: Operational, non-financial performance measures of supply chains

Operational, Non-financial Measures
▪ Cycle time
▪ Customer service level ❖ order fill rate ❖ stock-out rate ❖ backorder level ❖ probability of on-time delivery
▪ Inventory levels
▪ Resource utilization
▪ Capacity/Throughput
▪ Quality
▪ Reliability
▪ Dependability/Performability
▪ Flexibility ❖ volume

- product mix
- routing
- delivery time

Selection of the most appropriate operational, non-financial measures is subject to the specific supply chain considered. In this research, cycle time is chosen as a supply chain process performance measure for four reasons. First, cycle time is an all-encompassing measure as it denotes either supply chain end-to-end lead time or order-to-delivery lead time. Second, minimized cycle time leads to increased customer satisfaction together with reduced inventory, reduced obsolescence and increased quality. Third, it involves procurement lead time, manufacturing lead time, distribution lead time, logistics lead time, setup times for machines, waiting times, decision-making times and synchronization times. Inventory reduction and optimal utilization of resources through accurate forecasting and intelligent use of information is the key to lead time reduction. In addition, load balancing, optimal resource allocation and strict control of processing times reduce lead times considerably. In this dissertation work, in order to minimize the cycle time throughout the supply chain, the proposed DDDAMS methodology is applied to operations scheduling via online part routing and maintenance via preventive maintenance scheduling. Equation 3.1 depicts the overall system objective in a mathematical form.

$$min \sum_{i \in J} \sum_{j \in M} (W_{ij} + P_{ij} + T_{ij}) \qquad \text{Eq. (3.1)}$$

where J: Set of jobs those need to be processed

M: Set of machines in which jobs need to be processed

W_{ij}: Waiting time of job i waiting to be processed in machine j

P_{ij}: Processing time of job i in machine j

T_{ij}: Transfer time of job i to machine j

3.4 Concluding Remarks

In this section, we have first introduced two major types of supply chain configurations (collaborative and competitive) and discussed the differences on applying the proposed simulation based control for them. Then, we have presented an overview on the semiconductor industry, semiconductor manufacturing processes, and semiconductor supply chains, which have been selected as a case study to illustrate and demonstrate the proposed methodology in this dissertation. Next, we have discussed significance of two major problems of our interest (preventive maintenance scheduling and part routing scheduling problem) arising in the semiconductor supply chains. Finally, we have explained the supply chain performance metrics adopted in this work. In the next chapter, we discuss the details of the proposed system architecture in the greatest detail, which will be used to resolve the preventive maintenance and part routing scheduling problems discussed in this chapter.

CHAPTER 4

PROPOSED DDDAMS ARCHITECTURE AND COMPONENTS

4.1 Measurements in Sensor Network

Effective control of large scale and complex manufacturing systems necessitates all-embracing acquisition of information about major components pertaining to a manufacturing system, such as the environment, equipment, processes, and products. By acquiring considerable quantities of sensor data, a state of the system under consideration can be estimated. While receiving an update for a single sensor data is relatively straightforward and computationally undemanding, receiving updates for various parameters over hundreds of sensors generates large amount of data and realization of these updates becomes computationally intensive. Furthermore, each sensor has a sampling period based on the dynamics of the physical system. Therefore, if concurrent access to data which can be obtained from different sensors having diverse timing constraints is needed, it turns out to require continuous real-time monitoring, which is even more demanding computationally. To lighten the computational burden, while at the same time to lighten the burden in corresponding modeling and analysis, dynamic fidelities are proposed in this dissertation research. The notion of dynamic fidelity allows for monitoring specific parts of the manufacturing system in detail while monitoring the rest in an aggregated manner. Various sensor types together with their numbers, measurement units, and range of possible values are shown in Table 4.1.

Table 4.1: Details of sensor types/performance metrics used in this study

Sensor Type/ Performance Metric	Description	Range	Unit	Abbreviation
Temp	Temperature	[60-85]	Fahrenheit	^{o}F
Press	Pressure	[9800-10100]	Pressure per square inch	Psi
Snd	Sound	[0-140]	Desibel	dB
AirQuality	Air quality	[0-500]	Air quality index	AQI
Hmd	Humidity	[0-30]	Gram per cubic meter	g/m^{3}
Vib	Vibration	[0-10]	Desibel	dB
PR	Production rate	[0-1]	-	-
RawMatSt	Raw material status	[0-100]	Unit	-
CT	Mean cycle time	[130-150]	Minute	Min

Most of the sensor types that are used in the system set-up of this research as well as their associated measurement units are commonly known, whereas *AirQuality* and *Snd* might be an exception. The unit of measurement which is used to scale the data obtained from the *AirQuality* sensors is called Air Quality Index (AQI). AQI is a standardized indicator of the air quality in a given location. It measures mainly ozone and particulates, but may also include sulfur dioxide, and nitrogen dioxide. Various agencies around the world measure such indices, though definitions may change between places. AQI structure used by the United States Environmental Protection Agency (EPA) is as given in Table 4.2. Sound sensors are also known as acoustic sensors. Although there are other measurement units such as *Hz* (1/*frequency*) and *dB*, the most widely used unit is the unit that works on a basis of quantifying the sound levels relative to a predetermined '0 *dB*' reference.

Table 4.2: AQI structure used by the United States Environmental Protection Agency

Range for Air Quality Index	Definition of Range
0-50	Good

51-100	Moderate
101-150	Unhealthy
151-200	Unhealthy
201-300	Very unhealthy
301-500	Hazardous

4.2 Overview of DDDAMS Architecture and Embedded Algorithms

The goal of the dynamic data driven adaptive multi-scale system (DDDAMS) developed in this dissertation research is to achieve the effective synchronization of time and information between the real-system and its simulated counterpart (which is used to perform timely evaluation of alternative control policies). This synchronization then leads the simulation to run with the most up-to-date and necessary data while utilizing computational resources efficiently. In this study, to enable such synchronization, DDDAM-simulations are implemented as real-time simulations. The interactions between DDDAMS (decision evaluator and task generator for a real system) and the real system is depicted in Figure 4.1, where they involve dynamic switching between multiple fidelities for efficient usage of computational resources and synchronized monitoring in a sensor network within a pre-determined time interval (see δt in Figure 4.1). At the beginning of each interval (δt), new measurements are requested from the necessary machines to be updated within the very next measurement interval. To this end, DDDAMS is realized by changing the number of data sources and data acquisition frequencies among the supply chain echelons and among various departments (and equipment pertaining to the departments) within each echelon. Here, a considerable amount of computational resource is consumed by simulation due to its execution as well

as sensor data update to reach such an accurately synchronized system. Therefore, DDDAM- simulations should dynamically adjust their levels of fidelities to assure synchronization of important information (without missing any significant measurement) under limited availabilities of computational resources. It is noted that a level of fidelity affects both the simulation execution time as well as the time taken to collect required sensory updates.

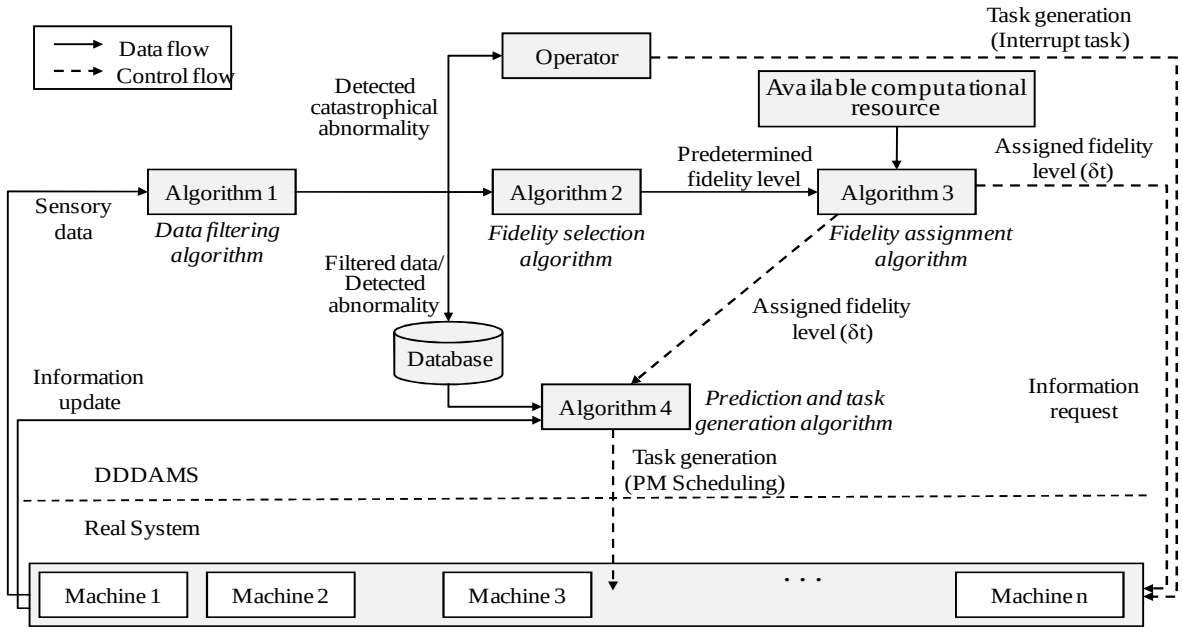


Figure 4.1: Overview and embedded algorithms of DDDAM-Simulation

As part of the framework proposed in this dissertation, four algorithms have been developed and embedded into the simulation to enable its DDDAMS capability (see Figure 4.1). Here, the purpose of Algorithm 1 is to filter noise and detect any abnormalities that may appear in the status of the system based on the measurement of

the current sensory data (such as temperature and pressure). The purpose of Algorithm 2 is to select the proper fidelity level of each component in the system via Bayesian inferencing. The purpose of Algorithm 3 is to opt for the available fidelity level of each component by taking the system level computational resource constraint into account. This algorithm obtains a matrix which encapsulates the proper (desirable) fidelity level of each component discussed in Algorithm 2 as an input as well as the available computational resource capacity from the grid computing service, and returns a new matrix which contains the assigned fidelity level for each component in the system that was evaluated at the current time as an output. The purpose of Algorithm 4 is to obtain optimal control tasks based on prediction (via fast-mode simulation or meta-models such as regression models) of future system performance. For the considered preventive maintenance scheduling, Algorithm 4 provides recommendations on when the next maintenance operation should take place. For the operations scheduling, it generates control tasks based on the optimal sequence of operations. Detailed explanations for each of these algorithms are provided in the following sections.

4.3 Algorithm 1 – Data Filtering and Abnormality Detection Algorithm

The purpose of Algorithm 1 is to detect an abnormal status of the system based on the measurement of the current sensory data (such as temperature and pressure) as well as the performance data (such as the cell throughput rate). The generic strength of the proposed method for this algorithm lies in its ability to spot abnormal data by using the combination of the X chart for Individuals (Shewhart control chart, Xie et al., 2002 and

Cassady et al., 2000) and the Moving Range chart (Wheeler and Chambers, 1992). This combination is often called an X and R_m or X_mR Chart (see Figure 4.2). The algorithm establishes control limits by constantly calculating a moving average of the recent measurements that are taken from each data source at each time interval, and is able to adapt to changing system conditions. The sample size of each measurement ($i.e.,N_s$) is determined as part of the operation of Algorithm 2, which is detailed in Section 4.4.

The XmR Chart was proven to be a successful technique for monitoring process measurements for those situations where successive measurements can be assumed to be independently distributed. Although several 'real-life' examples showing violation of the independence assumption can be discussed, in the first stage of the study the sensor data is assumed to be either totally independent or correlated up to some small extent which the Moving Range chart is able to handle. This rather strong assumption has been relaxed when we revised Algorithms 1 and 2 into one single algorithm during the second stage of this study (see Section 4.5). The autocorrelation of the current measurement data used in this study tends to be low since system status dynamically changes due to the effect of various factors and does not show any trend such as repetitive patterns or non-random behavior. Yet, as mentioned, this algorithm's duty is placed on a particle filtering algorithm in later sections in order to handle serially-correlated and/or auto-correlated data.

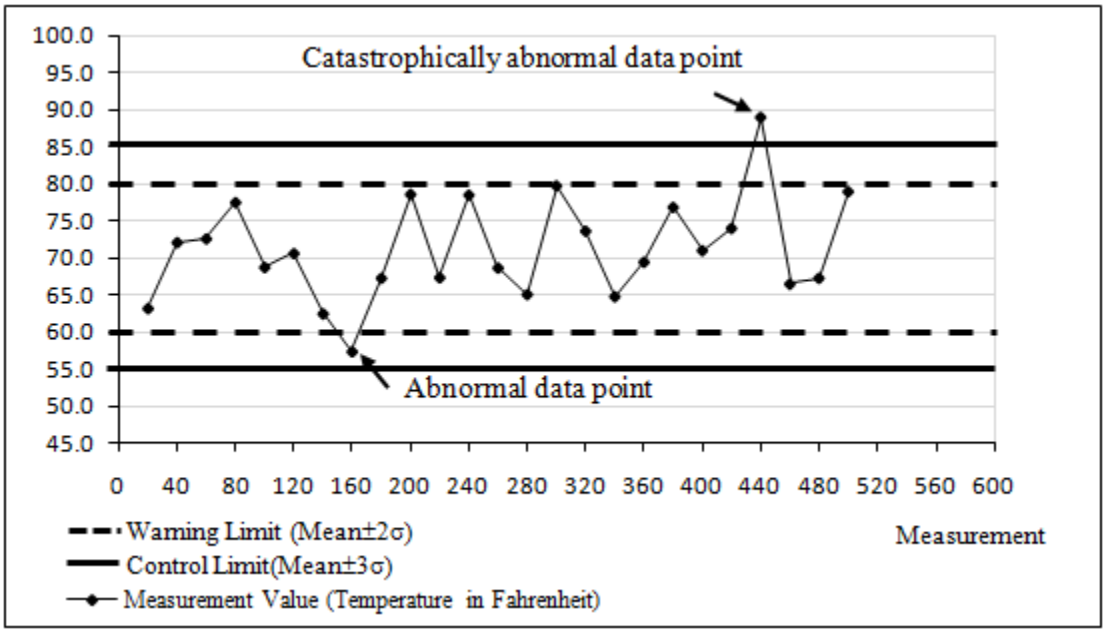


Figure 4.2: Operation of Algorithm 1

Algorithm 1 detects two types of abnormalities, namely abnormality, and catastrophic abnormality. If a single measurement in the newly updated dataset goes beyond the warning limits but is within the control limits, it is called abnormal. In this case, the abnormality condition is notified to Algorithm 2, which then takes an action to investigate the situation further (see Section 4.4 for details). If the measurement goes beyond the control limits, it is flagged as catastrophically abnormal and when this happens, Algorithm 1 alerts an operator/manager of an incident requiring immediate attention. This can be considered as an interrupt task as opposed to the regular tasks generated by Algorithm 4 (see Section 4.5 for details). In our work, the control limits used to assess incoming data are not calculated until at least one hundred data points are received (*i.e.*, initial sample size is set to $N_s = 100$). Therefore, a short training period is

required to establish proper control limits. During the training period, only basic checks against out-of-range values are performed on the input data.

In this work, Algorithm 1 has been implemented in Visual Basic (VB) Application and plugged into the Arena® simulation model using a Visual Basic Application (VBA) block. The Algorithm 1 module is called with a timestamp and latest data points, and returns filtered data value. It also stores past data points in the database, which is constantly updated at the rate of which new system measurements are taken. In the preventive maintenance and part routing application considered in this work, during data collection, three text files (as a form of database) keep data from each sensor on each machine. They are (1) parameter_history_file.txt – which stores the latest valid timestamps and data points of size N_s; (2) parameter_last_valid.txt – which stores the very last valid timestamp and data point; (3) parameter_consecutive_bad_data.txt – which counts the consecutive bad data points received. They are located in the system hard drive, are accessible by any other part of the system, and are updated each time Algorithm 1 is called. These text files are generic to any given sensor for any given application.

4.3.1 $\bar{X}$ Chart for Individuals and Limits

$\bar{X}$ charts for individuals are used for testing stability of the mean operation by calculating averages at regular intervals in time from samples of size N_s which are taken at each time step. In $\bar{X}$ charts for individuals, a center line is determined from either a target or specification value to monitor whether we are “on-specification”. Reference

data set is average of sample averages for data set collected when a process was operating normally (see Eq. (4.1)).

$$\bar{\bar{X}} = \frac{\sum_{j=1}^{N_s} \bar{X}_j}{N_s} \qquad \text{Eq. (4.1)}$$

Control limits for $\bar{X}$ charts for individuals are determined using the average range during normal operation, which is a reflection of usual process variability. Steps to compute these limits according to Shewhart's proposed general model for control charts are as follows:

$$\bar{R} = |X_{max} - X_{min}| \quad \text{for range } (R) \qquad \text{Eq. (4.2)}$$

$$UCL = \mu + \frac{3\sigma}{\sqrt{N_s}} \cong \bar{\bar{X}} + \frac{3\bar{R}}{d_2\sqrt{N_s}} \quad \text{for upper control limit } (UCL) \qquad \text{Eq. (4.3)}$$

$$CL = \mu \cong \bar{\bar{X}} \quad \text{for center line } (CL) \qquad \text{Eq. (4.4)}$$

$$LCL = \mu - \frac{3\sigma}{\sqrt{N_s}} \cong \bar{\bar{X}} - \frac{3\bar{R}}{d_2\sqrt{N_s}} \quad \text{for lower control limit } (LCL) \qquad \text{Eq. (4.5)}$$

4.3.2 Abnormality Detection Rules

Control limits are used to determine if the sensor data is under statistical control (*i.e.*, is having consistent output). The Western Electric Company (WECO) rules, which are summarized below, are employed as part of the decision rules for detecting "out-of-control" or non-random conditions on the control charts.

Rule 1: If one data point lays more than three standard deviations from center line, then signal “abnormal”.

Rule 2: If two out of three consecutive data points lie on the same side of the center line and are more than two standard deviations away from center line, then signal “abnormal”.

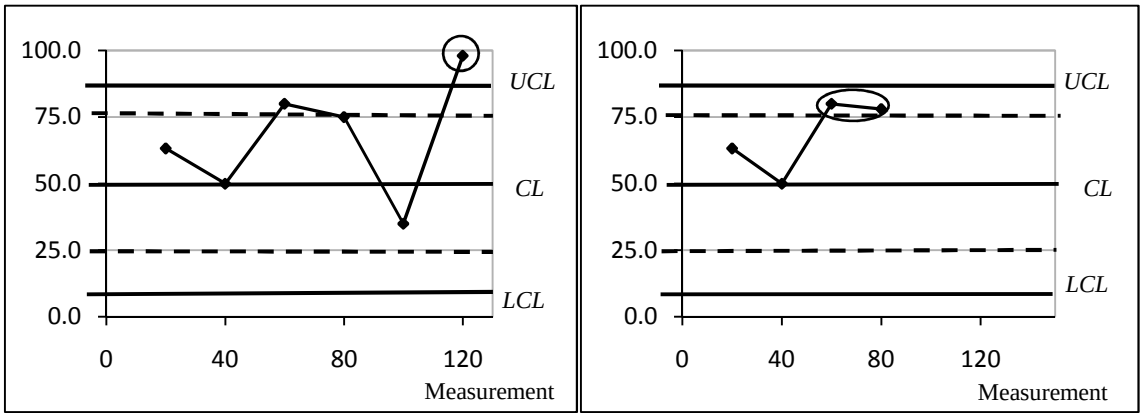


Figure 4.3: Graphical representation for WECO Rules 1 and 2

Rule 3: If four out of five consecutive data points lie on the same side of the center line and are more than one standard deviation away from center line, then signal “abnormal”.

Rule 4: If eight consecutive data points lie on the same side of the center line, then signal “abnormal”.

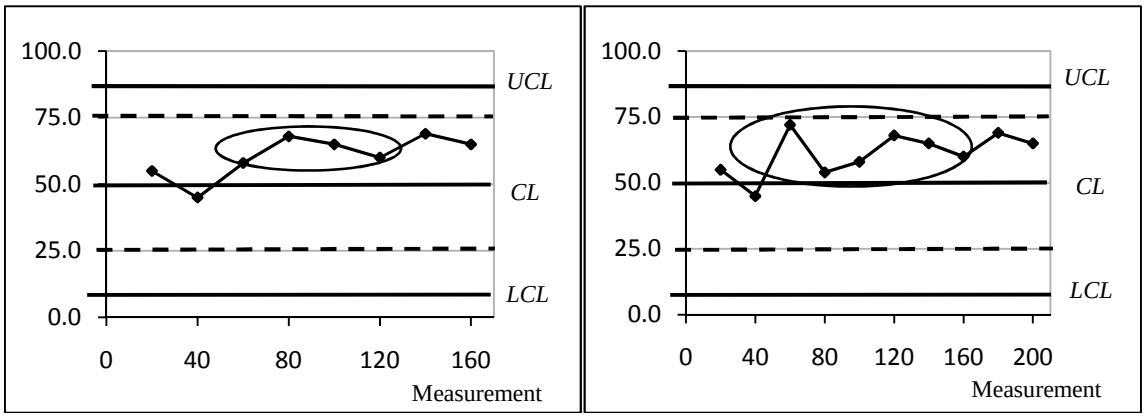


Figure 4.4: Graphical representation for WECO Rules 3 and 4

4.3.3 Moving Range Charts

In this study, moving range charts are employed to obtain a measure of dispersion for $\bar{X}$ chart limits and monitor consistency of variation in the process. The step by step computation of this chart is shown below in Eqs. (4.6) through (4.9). Each time a new data point becomes available, it is checked whether it is within the upper and lower limits. If it is not, it is detected as abnormal.

$$MR = |X_j - X_{j-1}| \text{ for moving range (MR)} \quad \text{Eq. (4.6)}$$

$$AMR = \overline{MR} = \frac{1}{N_s - 1}\sum_{j=2}^{N_s} |X_j - X_{j-1}| \text{ for average moving range (AMR)} \quad \text{Eq. (4.7)}$$

$$UL = \mu + 3\sigma = \bar{\bar{X}} + \frac{3\overline{MR}}{1.128} \text{ for upper limit (UL)} \quad \text{Eq. (4.8)}$$

$$LL = \mu - 3\sigma = \bar{\bar{X}} - \frac{3\overline{MR}}{1.128} \text{ for lower limit (LL)} \quad \text{Eq.(4.9)}$$

4.4 Algorithm 2 - Fidelity Selection Algorithm

Among the algorithms developed as part of this dissertation, the fidelity selection algorithm intends to determine a proper fidelity level of a simulation model (and the entire DDDAMS system) and plays a crucial role in the DDDAMS architecture for several reasons. First, it enables the simulation to adapt to changing system conditions considering the timely available computational resources. Second, it determines how much information is necessary to appropriately monitor and control the system while the conditions evolve over time. Depending on a specified fidelity, the simulation retrieves corresponding data from various types of sensors placed throughout the shop floor and its partners at corresponding frequencies. Third, as it is a main decision-making module in the DDDAMS architecture, its accuracy affects performance of the overall system (governed by the DDDAMS-based planning and control structure) most significantly. Fourth, a successive and rigorous computation of such decision-making algorithms, which are developed for large-scale, dynamic and complex systems, consumes considerable amount of computational resources, where its efficiency and parallelization become of primary importance for realization of the proposed architecture in a real-world distributed computing setting. During the initial development stage of the DDDAMS framework, the fidelity selection algorithm was realized via Bayesian Belief Networks (BBN). However, during the later stages it has been re-developed by incorporating particle filters along with newly proposed resampling rules. The reasoning behind the development and revision of both algorithms are detailed in the following subsections (see especially Sections 4.4.1 and 4.4.2).

4.4.1 Stage 1: Fidelity Selection via Bayesian Belief Networks

As mentioned before, the goal of Algorithm 2 is to select a proper fidelity level of each component in the system via a Bayesian Belief Network (Jensen, 2001; Pourret et al., 2008). It takes the filtered data as input from Algorithm 1 and outputs a matrix, which captures the proper fidelity level of each component (*e.g.*, machines). For the preventive maintenance and part routing selection application; considering a specific machine, if the status of a parameter that comes from Algorithm 1 is abnormal, Algorithm 2 determines the most likely interactions, which might have caused this result and accordingly selects the parameters (sensor data) which are needed to be measured in more detail.

A Bayesian Belief Network (BBN) has been employed for several reasons. First, the BBN is an increasingly used model to embody the above mentioned cause/effect interactions via a directed acyclic graph whose nodes represent variables and arcs represent statistical dependence relations (conditional distributions) among local probability distributions and these variables, given the values of their parents. Second, BBN provides valuable analysis even if some information is uncertain. Third, via the usage of BBN, the possible parental causes for an observed event (result) can be traced, which results in an increasingly intelligent BBN algorithm.

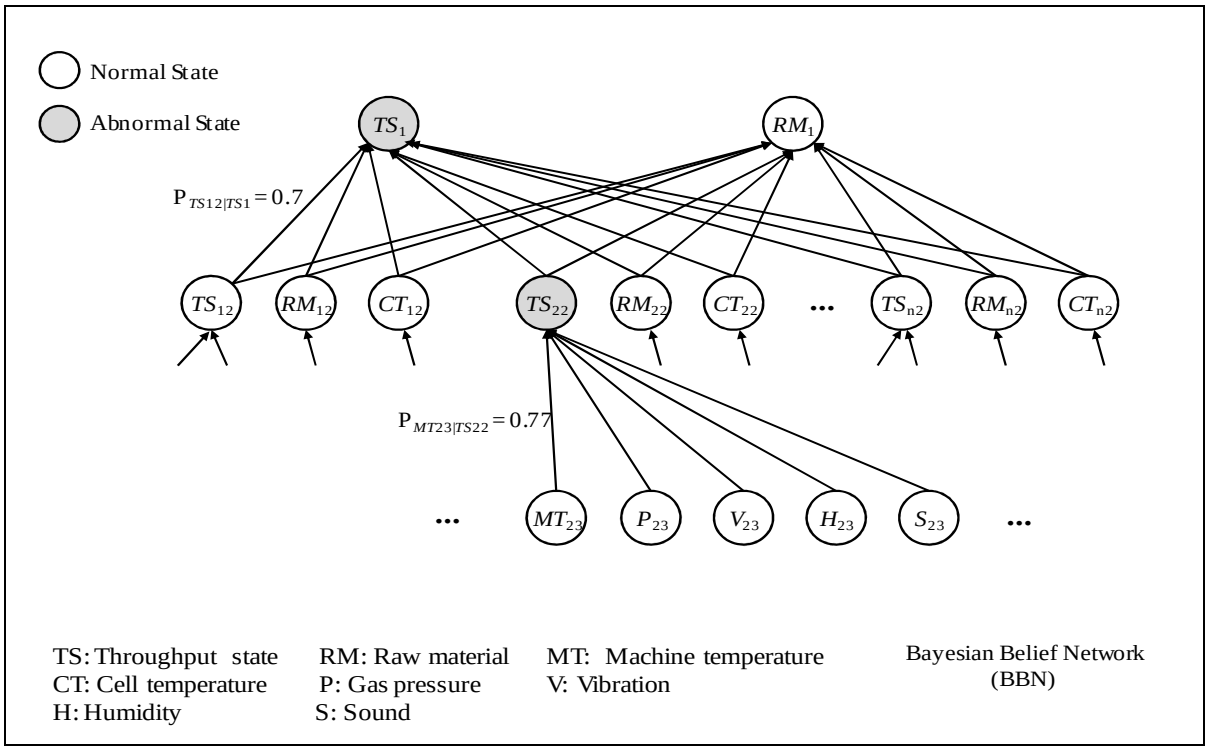


Figure 4.5: BBN for fidelity selection

In a BBN, the absence of an arc between two variables indicates conditional independence, that is, there are no situations in which the probability of one of the variables depends directly upon the state of the other. In a BBN, all variables that are important in modeling of the system should be included. Causal knowledge and prior knowledge should be used to guide the connections made in the graph and to specify the conditional distributions, respectively. Causal knowledge in this context means linking variables in the model in such a way that arcs lead from causes to effects.

In a general probability model (*e.g.* BBN), a random variable can take on a set of different values that are mutually exclusive and exhaustive. While probability theory as a whole can handle both discrete and continuous random variables, in this study only

discrete random variables with a limited number of possible values are considered due to excessive data required for constructing a BBN.

Before an inference is performed, the conditional probability for every node given its parents should be known. In the first stage of our research, all the considered nodes are discrete (as mentioned in the previously) and a conditional distribution between a node and its parents is modeled as an empirical distribution.

In a BBN, nodes can represent any kind of variable such as a measured parameter, a latent variable or a hypothesis (for any generic application). In our work, nodes represent the dynamically updated sensor data for each machine. Construction of the BBN for a specific system should be performed by a domain expert who is capable of determining the relationship and its degree between the parent and child nodes. Even though the BBN used in a running application can be modified both in terms of the relationships between the nodes and the degrees of these relationships depending on the offline analysis of results and expertise opinion, the BBN considered in our work is static in the sense that its structure does not change dynamically. The BBN and sensors used for each fidelity level in our model are depicted in Figure 4.6 and Table 4.3, respectively. In Fidelity 1, data coming from six sensors are used. In Fidelity 2, six additional sensors are activated and the number of sensors used increases to 12; and in Fidelity 3, a total of 15 sensors are used to determine the current system status by computing the mean time between failures and current cycle times of machines. While three specific levels of fidelity are considered in this work, the number of levels can be numerous in a generic sense.

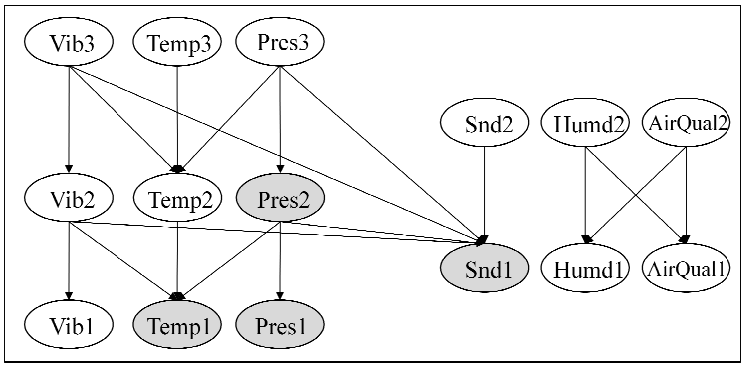


Figure 4.6: BBN for fidelity selection

Table 4.3: Sensors used for each fidelity level

Sensor name	Description	Fidelity
Temp1	Temperature sensor 1	1
Vib1	Vibration sensor 1	1
Pres1	Pressure sensor 1	1
Snd1	Sound sensor 1	1
Humd1	Humidity sensor 1	1
AirQual1	Air quality sensor 1	1
Temp2	Temperature sensor 2	2
Vib2	Vibration sensor 2	2
Pres2	Pressure sensor 2	2
Snd2	Sound sensor 2	2
Humd2	Humidity sensor 2	2
AirQual2	Air quality sensor 2	2
Temp3	Temperature sensor 3	3
Vib3	Vibration sensor 3	3
Pres3	Pressure sensor 3	3

Table 4.4 depicts the hierarchical structure of a BBN for the conditional probabilities of a specific parameter (sound). In Table 4.4, the initial values are based on the historical data and, as new measurements are taken, these values are updated. For a specific machine, if the status of a parameter that comes from Algorithm 1 changes from normal to abnormal, the probability of the abnormal state of the parent node is altered

based on Bayes' theorem and vice versa. By doing so, conditional probabilities related to the parent/child nodes are updated and the most likely interactions which might have caused abnormalities are determined. If a conditional probability is beyond its upper threshold value, then the fidelity level is increased by one, which causes the model to be more detailed. Similarly, if a conditional probability happens to be lower than its lower threshold value, then the fidelity level is decreased by one, and the model is switched to a more aggregated one. Selection of both the upper and lower threshold values for each fidelity level should be made by the experts and system analysts. While this selection can change dynamically as mentioned earlier, in our specific case study, the upper and lower threshold probability values were set to 0.4 and 0.1, respectively, in an arbitrary manner for all levels of fidelity.

Table 4.4: Hierarchical structure for the conditional probabilities of "sound"

Parent Node(s)					Snd1		
Snd2	Pres3	Pres2	Vib3	Vib2	Normal	Abnormal	bar charts
Normal	Normal	Normal	Normal	Normal	0.99	0.01	
				Abnormal	0.7	0.3	
			Abnormal	Normal	0.8	0.2	
				Abnormal	0.7	0.3	
		Abnormal	Normal	Normal	0.7	0.3	
				Abnormal	0.6	0.4	
			Abnormal	Normal	0.6	0.4	
				Abnormal	0.6	0.4	
	Abnormal	Normal	Normal	Normal	0.8	0.2	
				Abnormal	0.7	0.3	
			Abnormal	Normal	0.7	0.3	
				Abnormal	0.7	0.3	
		Abnormal	Normal	Normal	0.8	0.2	
				Abnormal	0.7	0.3	
			Abnormal	Normal	0.7	0.3	
				Abnormal	0.7	0.3	
Abnormal	Normal	Normal	Normal	Normal	0.7	0.3	
				Abnormal	0.6	0.4	
			Abnormal	Normal	0.6	0.4	
				Abnormal	0.6	0.4	
		Abnormal	Normal	Normal	0.5	0.5	
				Abnormal	0.4	0.6	
			Abnormal	Normal	0.4	0.6	
				Abnormal	0.4	0.6	
	Abnormal	Normal	Normal	Normal	0.6	0.4	
				Abnormal	0.5	0.5	
			Abnormal	Normal	0.4	0.6	
				Abnormal	0.4	0.6	
		Abnormal	Normal	Normal	0.5	0.5	
				Abnormal	0.4	0.6	
			Abnormal	Normal	0.4	0.6	
				Abnormal	0.4	0.6	

In our implementation for PM, data for fidelity level 1 is collected from six sensors namely, *Temp1, Vib1, Pres1, Snd1, Humd1, AirQual1* (see Figure 4.6). If any of the sensors has a probability greater than 0.4 (abnormal), then the node is decided to be critical and hence needs more detailed observation. If the mentioned sensor is not included in the current fidelity level, then the fidelity level is increased. Let us consider an example derived from Figure 4.6 where we consider a single machine whose initial fidelity level is 1. In this example we assume specific probability values for the sake of illustration. Here, let us assume the conditional probability that "*Pres2* is abnormal when

the sensor data for *Temp1*, *Pres1*, *Snd1* are normal" is 0.01, *i.e.*, *p*(*Pres2*=Abnormal | *Temp1* = Normal, *Pres1* = Normal, *Snd1* = Normal) = 0.01. However, the conditional probability increases to 0.2 when the sensor data for *Pres1* is abnormal while *Temp1* and *Snd1* are normal, *i.e.*, *p* (*Pres2*=Abnormal | *Temp1* = Normal, *Pres1* = Abnormal, *Snd1* = Normal) = 0.20. In this case, the fidelity level for this machine does not change because 0.2 is within the upper (0.4) and lower (0.1) threshold probability values. The conditional probability further increases to 0.78 when the sensor data for *Pres1* and *Snd1* are abnormal while *Temp1* are normal, *i.e.*, *p*(*Pres2*=Abnormal | *Temp1* = Normal, *Pres1* = Abnormal, *Snd1* = Abnormal) = 0.78. In this case, the fidelity level for this machine increases to 2 because the conditional probability of 0.78 is greater than our upper threshold value (0.4) indicating that an increase in the fidelity level is necessary. Similar conditioning applies to switches between fidelity levels 2 and 3 (see Figure 4.7 for a complete BBN developed in this work). As such, a BBN and rule-based system can be developed for any application by modifying the variables and parameters appropriately. This algorithm is hence extensible with certain modifications to any generic application of DDDAMS.

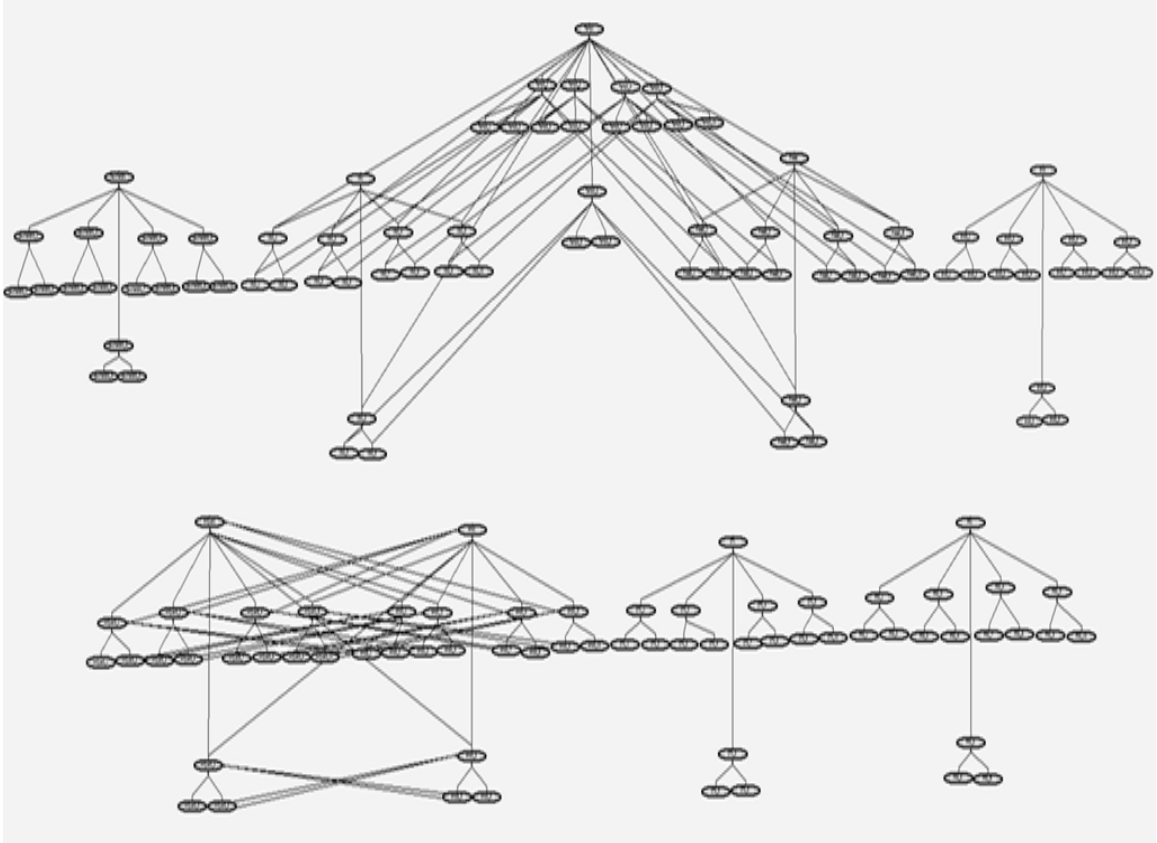


Figure 4.7: Complete BBN developed for fidelity selection in DDDAMS framework

As mentioned before, in the first stage of our research, all nodes involve discrete random variables and a conditional distribution between a node and its parents is modeled as an empirical distribution. Using empirical distribution is advantageous as it is calculated directly from the actual data and is easy to manipulate. However, there are also some critical issues related with it. First, the data used might not be representative enough to capture a system behavior in the event of insufficient training. Second, sensitivity analysis becomes considerably difficult. Third, results cannot be generalized to other systems. Fourth, modeling dependent data becomes difficult. Last, problems occur while capturing rare but important data.

One way to resolve the above-mentioned issues and at the same time to handle both discrete as well as continuous variables is to employ continuous-discrete filtering methods (continuous parameters measured at discrete instances of time). The advantage of continuous-discrete filtering model formulations (*i.e.*, Markov chains, particle filters, differential equations etc.) over a discrete time model formulation (*i.e.*, difference equations) is that the case of non-uniform sampling (*i.e.*, varying sampling interval) is naturally included in the model. Non-uniform sampling arises in practice, for example, when processing data from multiple sensors that are not synchronized. This is common, for example, in multiple target tracking applications. The continuous-discrete formulation is also more realistic than a pure continuous time model, because sensor measurements are often processed with digital computer which only allows processing of discrete time measurements.

In the continuous-discrete time models explained above, problem can be stated in a state space form as follows. A transition equation describes the prior distribution of a hidden Markov process $\{\mathbf{x}_k, k \in N\}$, so-called hidden state process, and an observation equation describes the likelihood of the observations $\{\mathbf{y}_k, k \in N\}$, k being a discrete time index. Within a Bayesian framework, all relevant information about $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$, given observations up to and including time k can be obtained from the posterior distribution $p(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k | \mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k)$. In many applications we are interested in estimating recursively in time this distribution, and particularly one of its marginal's, so-called filtering distribution $p(\mathbf{x}_k | \mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k)$. Given the filtering distribution, one can then routinely proceed to filtered point estimates such as the posterior mode or mean of the

state. This problem is known as the Bayesian filtering problem or the optimal filtering problem. Practical applications include target tracking, blind deconvolution of digital communications channels and digital enhancement of speech and audio signals. Except for a few special cases, including linear Gaussian state space models (Kalman filter) and hidden finite-state space Markov chains, it is impossible to evaluate these distributions analytically. From the mid 1960's, a great deal of attention has been devoted to approximating these filtering distributions; see for example Jazwinski (1970). More recent research has focused on Markov chain integration methods for Bayesian filtering, which have the great advantage of not being subject to the assumption of linearity or Gaussianity in the model (Doucet et al., 2000). Our next attempt in resolving the aforementioned challenges in fidelity selection (see Chapter 4.4.2) as well as improving the performance of DDDAMS framework involves this line of research as detailed in the next subsection.

4.4.2 Stage 2: Fidelity Selection via Particle Filtering

In Section 4.4.1, we have proposed a skeleton of the fidelity selection algorithm using a BBN. As mentioned before, in BBN all the nodes are restricted to discrete variables, and conditional probabilities between nodes and their parents were represented via empirical distributions only where the use if previous data may not represent the entire spectrum of possible system behaviors, rare but important events may not be captured and training of each node may lead to access of massive datasets which in turn

presents quite a computational challenge in a short amount of time for large-scale systems.

In this study, to address the above-mentioned issues on the empirical distributions and to employ both discrete as well as continuous variables directly to formulate the dynamics of systems under uncertainties, we propose a generic, modified Sequential Monte Carlo algorithm (also known as particle filtering algorithm) leveraging Bayesian inference. It is a continuous-discrete filtering method (continuous parameters measured at discrete instances of time) that performs fidelity selection in a rigorous but efficient manner. The specific goal of this algorithm is three-fold: 1) to extend the structure of DDDAM-Simulations in such a way to involve numerous fidelities (as opposed to a fixed number (3) as in Section 4.4.1), 2) to be able to predict a system status accurately using selective, massive sensor data coming from a large-scale, dynamic manufacturing environment and hence, to trace back the possible parental causes for an observed event in the most effective way, and 3) to perform a parallel Bayesian computation for different portions of a network in order to monitor specific parts of the manufacturing system in detail while monitoring the rest in an aggregated manner.

The goal of this algorithm is to develop an enhanced (integrated) algorithm, which will replace Algorithms 1 and 2 of our earlier framework (see Figure 4.1). In this proposed algorithm, Algorithm 1 of the original framework is not explicitly necessary as the proposed particle filtering method is able to follow the system status on time. Algorithm 1 of the original framework could be kept to filter the measurement data from noise; however, it would contribute to additional computational burden. On the other

hand, Algorithm 2 proposed in this work is able to handle noisy data via its Bayesian filtering capability. While the Algorithm 2 is built to enable highest computational efficiency, it is still based on the assumption that resources are always available for data retrieval and process whenever there occurs a need to obtain more measurements in higher frequencies. Hence, Algorithm 3 (which will be discussed in Chapter 4.5) is still required to deal with this assumption under the revised DDDAMS framework. From the control perspective, the goal of Algorithm 4 (which will be discussed in Chapter 4.6) of the original framework remains unchanged to derive necessary control tasks for the system.

4.4.2.1 Definition of Fidelities and States

In this study, a fidelity is defined as a measure of accuracy of a model (*i.e.*, simulation) when compared to a real system. As the fidelity increases, the similarity (in terms of physical and functional aspects) between the simulation and the real system increases. This is enabled via more frequent data updates. While frequent data updates in simulation is advantageous in terms of accuracy of simulation results, it results in higher computational resource usage and response (processing) time. Particularly, when the simulation is used as part of a real-time controller for a large-scale, supply chain (considered in this dissertation), timely collection of data becomes critical, but harder due to the sensors spread throughout the distributed elements and high processing requirements for massive information loads. In such cases, an appropriate fidelity should be selected for each subset of the system for the simulation efficacy.

In this study, we intend to obtain an optimal or near optimal fidelity of the DDDAM-Simulation for each cell in the shop by effectively using the measurement data. The measurement data used in this study are of two types: sensory data or performance metric data (which involves additional processing based on the sensory data). Table 4.4 depicts various sensor types and performance metrics (the last three rows of Table 4.4 in a separate box) considered in this study, together with their numbers, measurement units, and range of possible values. While the sensory data shows the instantaneous change in the system component, the performance metric data shows the cumulative effect of the successive changes in the system state or sensory data. The sensory data is only used when the model fidelity increases as a result of a need of a more detailed information regarding the system behavior. The performance metric data, on the other hand, is collected from system components (*e.g.*, machines) and used regardless of the model fidelity. However, the way the performance metric data is used changes depending on the model fidelity. For instance, in an aggregated fidelity (lower fidelity), we still collect CT (mean cycle time) data from machines. However, this data collection is in a random manner and we only use an average of them. Therefore, we do not necessarily know from where exactly the data comes from (*e.g.*, which machine). Here each data point can be visualized as a single particle capturing some data regarding the system. On the other hand, the sensory data is only requested in higher fidelities where their origin is known exactly. In lower fidelities, the sensory data is not requested in order to save from computational resources.

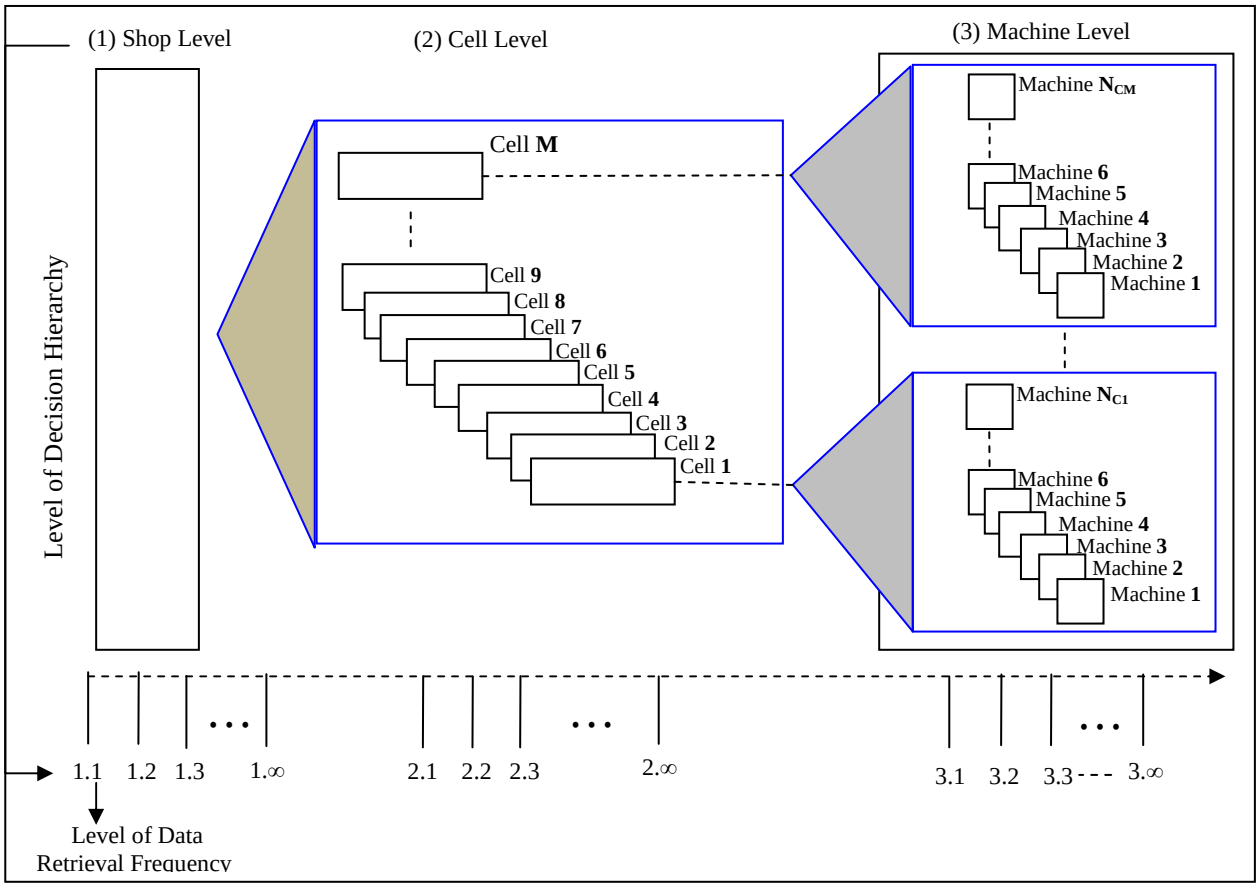


Figure 4.8: Definition of simulation fidelity with levels of decision hierarchy and data retrieval frequency

In this work, the fidelity of a DDDAM-simulation is represented by two numbers separated by a dot (*e.g.*, 2.3) (see Figure 4.8). The first number indicates the level of decision hierarchy. At each level, types of the measurements those need to be collected from various portions of the facility (*i.e.*, shops, cells or machines) are determined. For example, "1" denotes the shop level, meaning data coming from each performance metric is collected randomly from machines located throughout the entire shop. While the data still have a tag regarding which machine it is associated with, simulation is not necessarily interested in this information as at this level all the machines in the shop are

assumed to operate under similar conditions (*i.e.*, same room temperature). If it is "2" for a cell in the shop, the selected fidelity of the simulation is in the cell level for this specific cell, but aggregated for the rest (still in Fidelity 1). For a cell with simulation fidelity "2", data coming from each performance metric is collected randomly from machines located only in this cell. Similarly, if it is "3" for a cell in the shop, the selected fidelity of the simulation is in the machine level for this specific cell, but can be different for different portions of the shop (either Fidelity 1, 2 or 3). For a cell with simulation fidelity "3", data coming from each sensor type as well as performance metric is separately collected for each machine located in this cell. Once these levels of decision hierarchies are determined, levels of data retrieval frequency are determined by the second number. Although levels for the first selection are predetermined, levels for the second selection can change from "1" to infinite, meaning one single data or infinite amount of data can be collected during the simulation execution from each machine (see Figure 4.8).

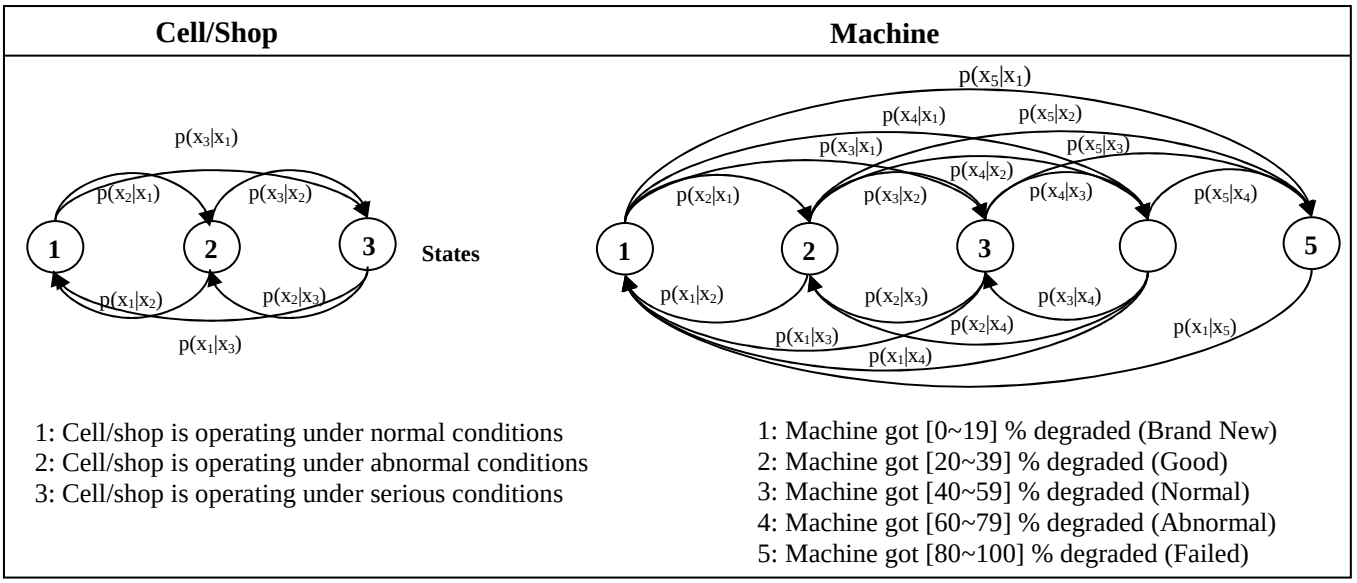
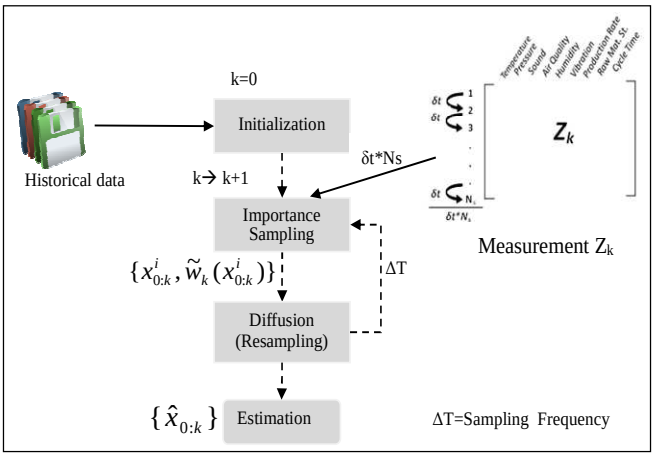


Figure 4.9: Markov chain property defined in terms of aggregated states for each machine, cell and shop

Once appropriate (optimal or near optimal) fidelities are chosen for each and every cell in the shop, measurements are performed accordingly. These measurements are then used to determine the very next fidelities, which in turn affect very next measurements in the considered DDDAMS framework. This process repeats during the simulation run. Given new measurements, simulation uses them to derive new Bayesian inferences through the particle filter algorithm developed in this study. In this work, Bayesian inferences are derived to reflect the system status in terms of machine status, cell conditions and/or shop statuses based on the sensory data. Figure 4.9 depicts the five states for each machine and the three states for each cell and shop as well as transitions among them. In the considered system, each machine can be only at one of the five states and similarly, each cell/shop can be only at one of the three states. Once the system status is determined as such at the end of each processing of Algorithm 2, Algorithm 4 is invoked in order to re-schedule (change the current schedule) preventive maintenance and generate operational tasks.

4.4.2.2 Adaptive Sequential Fidelity Selection Algorithm



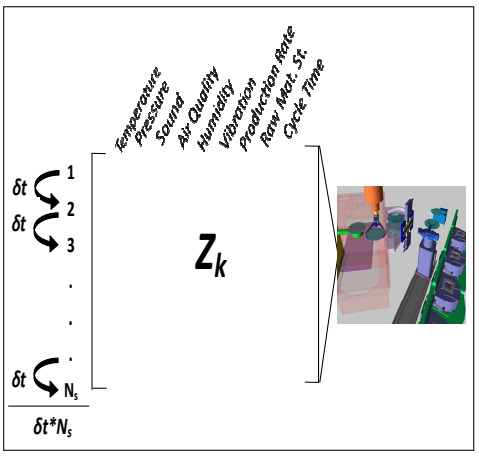


Figure 4.10: Operations of particle filter Figure 4.11: Acquiring measurement at time *k*

Figure 4.10 depicts an overview of the adaptive backward sequential fidelity selection algorithm (core of Algorithm 2 mentioned in the previous section) developed in this work. At the beginning, the historical data is used to obtain enough number of samples to generate a prior distribution of the variables of our interest. Later, the algorithm is tuned in a sequential manner as more sensory data arrives. Aggregated state definitions for the machine level, cell level, and shop level particle filters are shown in Figure 4.9. In this work, each machine, cell, and shop has its dedicated particle filter. The switch between these filters is dependent on how much accuracy is needed in what frequency during the abnormality detection process. There are no jumps allowed between hierarchical levels of fidelities (*i.e.*, a model has to go from fidelity 1.X to 2.X and then 3.X; immediate jumps from 1.X to 3.X are not allowed). The posterior distribution sought here is the $p(\mathbf{x}|\mathbf{z})$, where **x** stands for the state variables reflecting the machine's status in terms of how close it is to a failure, cell's status in terms of normality/abnormality, and shop's status in terms of the same. In addition, **z** reflects the

measurement data involving both sensory data (*i.e.*, temperature, sound, vibration, air quality, humidity, pressure) and performance metric data (*i.e.*, production rate, raw material status and cycle time) for machine level fidelities (*i.e.*, fidelity 3.X's) and only performance metric data for cell level and shop level fidelities (*i.e.*, fidelities 2.X's and 1.X's).

Table 4.5: Specifications of various fidelities included in DDDAMS

	Fidelity 1.X	***Fidelity 2.X***	***Fidelity 3.X***	N_s
Sampling Horizon (T)	T_s=300 time steps	T_c=200 time steps	T_m=100 time steps	--
Sampling Frequency (ΔT)	ΔT_s=30 time steps	ΔT_c=20 time steps	ΔT_m=10 time steps	--
Time for retrieving single measurement update (δt)	$3 < \delta t < 6$ time steps	$2 < \delta t < 4$ time steps	$1 < \delta t < 2$ time steps	--
Data type collected	Performance metric	Performance metric	Sensory	--
when RMSE < 0.1	1.1	2.1	3.1	50
when 0.1≤ RMSE < 0.2	1.2	2.2	3.2	60
when 0.2≤ RMSE < 0.3	1.3	2.3	3.3	70
when 0.3≤ RMSE < 0.4	1.4	2.4	3.4	80
when 0.4≤ RMSE < 0.5	1.5	2.5	3.5	90
when 0.5 ≤ RMSE	1.6	2.6	3.6	100
RMSE: Root Mean Square Error of State Estimation				

The sample size of a particle set (N_s) and time for retrieving single measurement update (δt) are determined by the previous level of data retrieval frequency. As the level of data retrieval frequency increases, N_s increases whereas δt decreases in order to obtain more accurate results in terms of the current system status. Here, regardless of the computation time of the algorithm, measurement takes '$\delta t. N_s$' time units to complete (see Figure 4.11). Computational resources are used in three separate places. First, they

are used to retrieve the measurement data from the real time machinery to the fidelity selection algorithm (Algorithm 2) (hence the real time simulation). Second, they are used for the prediction step of the particle filtering algorithm in order to derive the estimated states of the machines. Lastly, they are used for the update step of the algorithm where the current model is restored with the usage of new data in order to have better predictions in the future. Therefore, although increased N_s allows us to better estimate the posterior, it is costly in terms of time and computational resource usage. As a result, N_s should be kept as minimum as possible while obtaining a desired accuracy in estimating posterior. The time to re-iterate the algorithm from iteration k to $k+1$ (sampling frequency ΔT) is determined by the previous level of decision hierarchy. The specifications of the fidelity levels included in our DDDAMS work are summarized in Table 4.5, and details of the algorithm are discussed below.

In realizing any particle filter algorithm, careful selection of a proposal (importance) density is crucial as it determines the number of particles generated and the computational expense needed for each of the particles. In importance sampling, a heavy-tailed density is preferred for the proposal distribution. From a Bayesian perspective, the proposal distribution $q(\mathbf{x}|\mathbf{z})$ is assumed to approximate the posterior $p(\mathbf{x}|\mathbf{z})$ and $q(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$. Although it does not take measurements into account, for its ease of implementation, we accept the prior distribution as the proposal density in this study (*i.e.*, $q(.) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$). A sequence of steps of the algorithm is described below.

Step 1: Initialization

The initialization step is held assuming that there are adequate historical measurements obtained for each of the sensor types as well as performance measures in order to generate a prior distribution.

- Set $k = 0$, where k stands for the iteration number. Since initialization occurs only once while setting the fidelity selection algorithm, k becomes equal to zero.
- For $i = 1, \dots, N_s$, sample $\mathbf{x}_0^i \sim p(\mathbf{x}_0)$ where N_s denotes the sample size and $\mathbf{x}_0^i$ denotes each individual sample drawn at iteration 0.
- Set $k = 1$

Step 2: Importance Sampling

- For $i = 1, \dots, N_s$, sample $\tilde{\mathbf{x}}_k^i \sim q(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{z}_k)$ where $q(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{z}_k)$ is the proposal density and $\tilde{\mathbf{x}}_{0:k}^i = (\mathbf{x}_{0:k-1}^i, \mathbf{x}_k^i)$
- For $i = 1, \dots, N_s$, evaluate the importance weights $\mathbf{w}_k^i = \mathbf{w}_{k-1}^i \frac{p(\mathbf{z}_k | \mathbf{x}_k^i) p(\mathbf{x}_k^i | \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i | \mathbf{x}_{0:k-1}^i, \mathbf{z}_{1:k})}$
- Normalize importance weights $\tilde{\mathbf{w}}_k^i = \mathbf{w}_k^i / \sum_{j=1}^{N_s} \mathbf{w}_k^j$ where $\mathbf{z}_{1:k}$ are the measurements from time 1 to k

Step 3: Diffusion (Resampling)

This step is to avoid potential degeneracy problem, where only one particle has a normalized weight of "1" whereas all the rest has almost zero weight. In this step, the particles with low weights are eliminated whereas more particles in more probable

regions are selected. The decision of resampling can be made based on different reasoning (rules). In this work, we have developed new stratified sampling rules which are proved to be exact optimal in terms of minimized variance and minimized bias in order to improve the performance of this step. Details of these new resampling rules are provided in the next subsection. Here, if the effective number of particles is less than a given threshold $\widetilde{N}_{eff} = \frac{1}{\sum_{j=1}^{N_s}(\tilde{\mathbf{w}}_k^i)^2} < N_{threshold}$, then we perform resampling. Resampling steps of the algorithms are shown below and a corresponding example is provided in Figure 4.12.

- Resample N_s particles randomly with replacement $(\tilde{\mathbf{x}}_{0:k}^i i = 1, \dots, N_s)$ from the current particle set $(\tilde{\mathbf{x}}_{0:k}^i i = 1, \dots, N_s)$ with probabilities proportional to their normalized importance weights, $\tilde{\mathbf{w}}_k^i$
- For each particle $(\mathbf{x}_{0:k}^i i = 1, \dots, N_s)$, set $\tilde{\mathbf{w}}_k^i = 1/N_s$

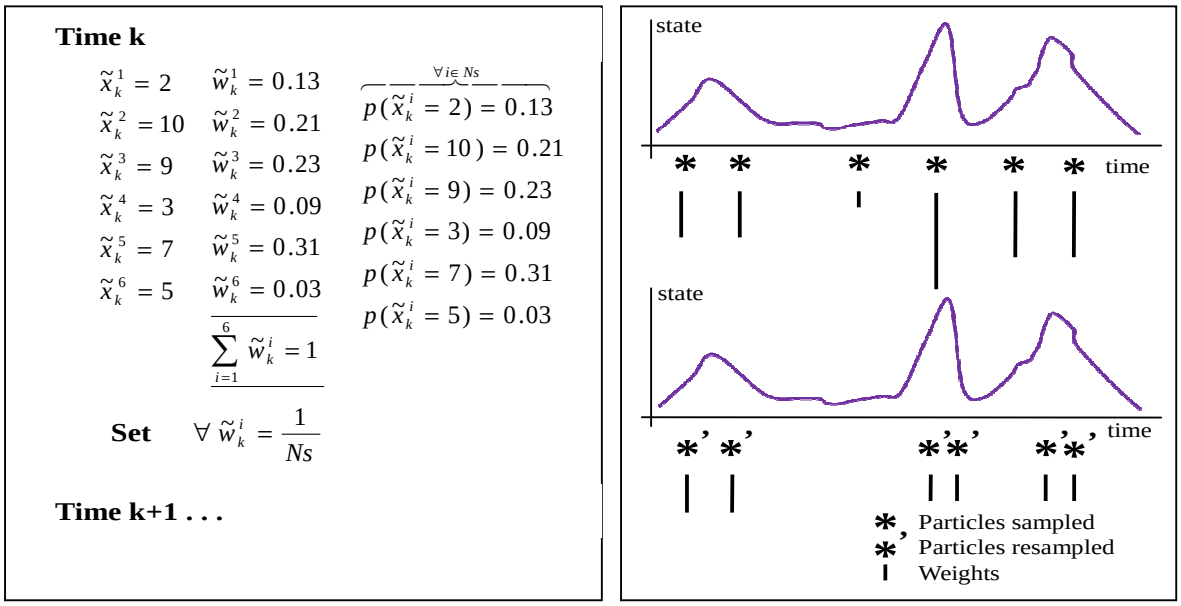


Figure 4.12: Exemplary diffusion (resampling) process

Step 4: Set $k \rightarrow k+1$

Proceed to the Step 2 as the next measurement arrives.

Step 5: Estimate posterior

A discrete weighted approximation to the true posterior is computed as: $p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} \mathbf{w}_k^i \, \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i)$, where δ are the Dirac peaks.

In this work, the proposed algorithm has been implemented in MATLAB 7.6.0 (R2008a), where particle filters are embodied in objects. There are separated particle filters built for each level of decision hierarchy (*i.e.*, Fidelity 1.X for shop, 2.X for each cell and Fidelity 3.X for each machine). The developed particle filter for Fidelity 1.X, 2.X and 3.X determines the optimal fidelity level at the shop level, cell level and machine level, respectively. When a particle filter object is called from the main simulation, it runs until it reaches a decision. In order to reach a decision, the algorithm filter part has to determine an optimum level of data retrieval frequency first given a desired accuracy level. It should be noted here that the sampling horizons (*i.e.*, T) and sampling frequencies (*i.e.*, ΔT) are different for each particle filter developed as the used sensory data as well as how fast they are needed are different. The particle filter for Fidelity 1.X samples data from the past 300 time steps, where each sampling is 30 time steps apart and time for retrieving single measurement update (δt) changes from 6 time steps to as low as 3 time steps. In addition, the data is only sampled from the specified performance

data. However, the particle filter for Fidelity 3.X samples data from the past 100 time steps, where each sampling is 10 time steps apart and time for retrieving single measurement update (δt) changes from 2 time steps to as low as 1 time steps. Besides, the data is only sampled from the specified sensory data.

4.4.2.3 Improved Resampling Techniques

As mentioned in Chapter 1, in the decision making process of coherent planning and control of supply chains, effective monitoring and hence obtaining the latest information reflecting current supply chain status and capabilities becomes critically important, while not disrupting its ongoing operations. However, supply chains generate massive datasets at each measurement point due to their large-scale, dynamic and complex nature, where both the collection and the analysis become quite challenging and computationally intensive. Even though this situation holds true for the strategic and tactical levels, it becomes even more obvious at the operational level where the number of parameters as well as the frequency of update for each parameter grow significantly. In order to enable timely monitoring, simulation-based analysis, and control of these supply chains at the operational level in an economical and effective way, Celik and Son (2010) proposed a data-driven adaptive simulation scheme incorporating Bayesian inferencing by means of particle filters. As mentioned in Chapter 2.2.7.3, Particle filtering defines a class of simulation-based estimation techniques, which have been used to solve various types of sequential Bayesian inference problems that are encountered in wide range of areas such as econometrics (Casarin and Sartore, 2008; Flury and

Shephard, 2008), signal and image processing (Brasnett et al., 2007; Xu and Li, 2007), robotics (Schulz and Burgard, 2001), and recently supply chain management (Celik and Son, 2010). In this study, the particle filtering algorithm intends to estimate the actual fidelity status of the simulations of supply chain members, which is further used in preventive maintenance and part routing scheduling problems in the supply chain control.

As mentioned earlier in this section, the idea behind the particle filtering resides in effective sampling from a sequence of probability distributions, and the algorithm is structurally composed of two major steps including importance sampling and resampling. Resampling, by definition is drawing repeated samples from subsets of available data based on a given criteria. It plays a critical role in the performance of the particle filter as it may resolve the potential issues of weight degeneration where after a few iterations, all but one particle will have negligible weight (Ristic et al., 2004) and waste of computational resources by replicating particles in proportion to their weights. In this work, we enhance the efficiency of the generic particle filtering algorithms by presenting improvements in their resampling techniques, namely variance-based resampling and bias-based resampling efficiency rules, respectively. Then, we revisit a half width-based resampling rule for benchmarking purposes. Here, all these three resampling rules arise from three distinct statistical standpoints. The proposed rules are first derived theoretically and their performances are benchmarked against the performances of the resampling rule developed by Kong et al. (1994) and half width-based resampling rule revisited in this work in terms of their resampling qualities and computational efficiencies using a simulation study.

4.4.2.3.1 Proposed Stratified Resampling Rules for Particle Filtering Algorithms

In this section, our focus is to develop efficient resampling rules to be used as part of Steps 2 and 3 of the algorithm given in Chapter 4.4.2.2. Following the same notation that is provided before in Chapter 4.4.2.2, we consider a random vector $\mathbf{x}_k = [\mathrm{x}_k^1, \mathrm{x}_k^2, \mathrm{x}_k^3, \dots, \mathrm{x}_k^i],\ i = 1, \dots, N_s$ with new samples of size N_s for each iteration k. However, throughout this section, by hiding the index k, we use only the random vector $\mathbf{x} = [\mathrm{x}^1, \mathrm{x}^2, \mathrm{x}^3, \dots, \mathrm{x}^i]$ to represent the sample of any iteration k to facilitate explanation. Here, random variable $\mathbf{x}$ is assumed to have the probability density function $p(\mathbf{x})$ (or probability mass function) and a function $h(\mathbf{x})$ (which is described below). In Step 3 of the particle filtering algorithm, our goal is to estimate the expected value of a function of $\mathbf{x}$ (e.g., μ), where $\mu = E_p[h(\mathbf{x})] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x}$, using the Monte Carlo approximation provided in Eq. (4.10). Here, $\hat{\mu}_1$ is an unbiased estimate of μ (see Appendix A for the proof of unbiasedness of $\hat{\mu}_1$).

$$\hat{\mu}_1 = \frac{1}{N_s} \sum_{i=1}^{N_s} h(\mathbf{x}^i) \qquad \text{Eq. (4.10)}$$

where $\mathrm{x}^1, \mathrm{x}^2, \mathrm{x}^3, \dots, \mathrm{x}^{N_s}$ are independent samples drawn from $p(\mathbf{x})$.

Importance sampling (Kong, 1992; Ristic 2004) suggests estimating properties of a desired distribution (e.g., the expected value in our case), while having samples generated from a different alternative distribution rather than the original distribution of interest when the original distribution is not known or very hard to generate samples

from. In this work, we represent that alternative distribution as $f(\mathbf{x})$. Using this alternative distribution, we can denote μ as given in Eq. (4.11), and its unbiased Monte Carlo estimate, $\hat{\mu}_2$, as given in Eq. (4.12) (see Appendix A for the proof of unbiasedness of $\hat{\mu}_2$).

$$\mu = \int \frac{h(\mathbf{x})p(\mathbf{x})}{f(\mathbf{x})} f(\mathbf{x})d\mathbf{x} = \int h(\mathbf{x})w(\mathbf{x})f(\mathbf{x})d\mathbf{x} = E_f[h(\mathbf{x})w(\mathbf{x})] \qquad \text{Eq. (4.11)}$$

$$\hat{\mu}_2 = \frac{\sum_1^{N_S} h(\mathbf{x}^i)w(\mathbf{x}^i)}{N_S} \qquad \text{Eq. (4.12)}$$

$$\overline{w} = \frac{1}{N_s}\sum_1^{N_s} w(\mathbf{x}^i) \quad \text{and} \quad w'(\mathbf{x}^i) = \frac{w(\mathbf{x}^i)}{\overline{w}} \qquad \text{Eq. (4.13)}$$

where $w(\mathbf{x}) = p(\mathbf{x})/f(\mathbf{x})$, and $w(\mathbf{x}^i)$ and $w'(\mathbf{x}^i)$ are the importance sampling weight of sample *i* and its normalized version, respectively.

The reason behind the introduction of term $w'(\mathbf{x}^i)$ is the fact that while $E_f[w(\mathbf{x})] = 1$, the sample mean $\overline{w}$ might not necessarily be equal to $E_f[w(\mathbf{x})]$. Therefore, the term $w'(\mathbf{x}^i)$ is introduced by having a sample mean of one by its normalized definition, to resolve this bias issue, and force $\overline{w}$ to be an unbiased estimate of $E_f[w(\mathbf{x})]$. Our estimate of μ can be represented as in Eq. (4.14).

$$\hat{\mu}_3 = \frac{\sum_1^{N_S} h(\mathbf{x}^i)w'(\mathbf{x}^i)}{N_S} = \frac{\frac{1}{N_S}\sum_1^{N_S} h(\mathbf{x}^i)w(\mathbf{x}^i)}{\frac{1}{N_S}\sum_1^{N_S} w(\mathbf{x}^i)} = \frac{\bar{z}}{\overline{w}} \quad \text{where} \quad z = h(\mathbf{x})w(\mathbf{x}) \qquad \text{Eq. (4.14)}$$

In this work, we have developed an improved stratified resampling rule based on the efficiency of $\hat{\mu}_3$ using the ratio of $\text{Var}_f[\hat{\mu}_3]/\text{Var}_p[\hat{\mu}_1]$ as a measure of relative efficiency between sampling from $p(\mathbf{x})$ and $f(\mathbf{x})$. In the prior literature (Kitagawa, 1996; Kong, 1992), in several cases, the estimation of the ratio $\text{Var}_f[\hat{\mu}_3]/\text{Var}_p[\hat{\mu}_1]$ is made via the Delta method which is a generalized combination form of central limit theorem and Slutsky's theorem. In some other particular cases, the approximation of the aforementioned ratio highly relies on the first order Taylor series expansion assuming that the higher order terms are negligible. In this study, we relax this quite strong assumption given the fact that the current sampling accuracy has a significant impact on determining the future sampling steps. In particular, we use the second order Taylor series expansion of ratio of variables in order to obtain a closed form solution that can be practically useful and efficient when employed in the particle filtering algorithms.

4.4.2.3.2 Closed Form Representation of the Variance of Ratio of Variables using Theorem of Taylor Series Expansion

In this section, we provide the proof to obtain a closed form formula of the variance of ratio of variables using the theorem of Taylor series expansion. The widely known Taylor series expansion of any function g of two variables (i.e., $g(x, y)$) can be found in earlier derivations shown by Guterman and Nitecki (2006), and Randall (2006). In this work, however, we are interested in the application of the Taylor expansion in a variance function of ratio of variables where we make use of the second order expansion in particular while the derivation of closed form formula is provided for any order n.

Let us consider a function of two variables $g(x,y)$, whose total change of the function can be due to changes either in variable x or variable y. Therefore, if dx and dy are assumed to be constant, the direction of the mentioned change in the (x,y) plane can be shown by Eq. (4.15).

$$dg = dx\frac{\partial g}{\partial x} + dy\frac{\partial g}{\partial y} \qquad \text{Eq. (4.15)}$$

Next, the chain of equations which derives the second total differential of the function $g(x,y)$ by taking the differential of the first total differential of $g(x,y)$, is shown in Eqs. (4.16).

$$d^2g = dx\frac{\partial(dg)}{\partial x} + dy\frac{\partial(dg)}{\partial y}; \quad d^2g = dx\frac{\partial\left(dx\frac{\partial g}{\partial x}+dy\frac{\partial g}{\partial y}\right)}{\partial x} + dy\frac{\partial\left(dx\frac{\partial g}{\partial x}+dy\frac{\partial g}{\partial y}\right)}{\partial y} \qquad \text{Eq. (4.16)}$$

$$d^2g = dx\frac{\partial\left(dx\frac{\partial g}{\partial x}+dy\frac{\partial g}{\partial y}\right)}{\partial x} + dy\frac{\partial\left(dx\frac{\partial g}{\partial x}+dy\frac{\partial g}{\partial y}\right)}{\partial y}; d^2g = (dx)^2\frac{\partial^2 g}{\partial x^2} + 2dxdy\frac{\partial^2 g}{\partial x\partial y} + (dy)^2\frac{\partial^2 g}{\partial y^2}$$

$$d^2g = \left(dx\frac{\partial g}{\partial x} + dy\frac{\partial g}{\partial y}\right)^2 g(x,y)$$

Here, the generalization of the derivation given in Eq. (4.16) to the total differential of order n yields to Eq. (4.17).

$$d^ng = \left(dx\frac{\partial}{\partial x} + dy\frac{\partial}{\partial y}\right)^n g(x,y) \qquad \text{Eq. (4.17)}$$

By using the closed form shown in Eq. (4.17), the Taylor series expansion of $g(x,y)$ for the neighborhood of the point (a,b) can be written as in Eq. (4.18).

$$g(x,y) = g(a,b) + \left[(x-a)\frac{\partial}{\partial x} + (y-b)\frac{\partial}{\partial y}\right]g + \frac{1}{2!}\left[(x-a)\frac{\partial}{\partial x} + (y-b)\frac{\partial}{\partial y}\right]^2 g + \cdots \quad \text{Eq. (4.18)}$$

At this point, by using the closed form formula shown in Eq. (4.18) for the second order Taylor series expansion, our function of ratio of variables $g(x,y) = y/x$ can be approximated via the series of Eqs. (4.19)-(4.22) around the neighborhood of the point (a,b).

$$g(x,y) = y/x \quad \text{Eq. (4.19)}$$

$$dg = dx\frac{\partial g}{\partial x} + dy\frac{\partial g}{\partial y} = \left[-\frac{y}{x^2}(x-a) + \frac{1}{x}(y-b)\right] \quad \text{Eq. (4.20)}$$

where $dx\frac{\partial g}{\partial x} = -\frac{y}{x^2}(x-a)$ and $dy\frac{\partial g}{\partial y} = \frac{1}{x}(y-b)$

$$d^2g = dx\frac{\partial(dg)}{\partial x} + dy\frac{\partial(dg)}{\partial y} = \left[(x-a)\left(\frac{-2ay}{x^3} + \frac{b}{x^2}\right) + (y-b)\left(\frac{a}{x^2}\right)\right] \quad \text{Eq. (4.21)}$$

where $\frac{\partial(dg)}{\partial x} = \left(-\frac{2ay}{x^3} + \frac{b}{x^2}\right)$ and $\frac{\partial(dg)}{\partial y} = \left(\frac{a}{x^2}\right)$

$$g(x,y) = \frac{b}{a} + \left[-\frac{y}{x^2}(x-a) + \frac{1}{x}(y-b)\right] + \frac{1}{2!}\left[(x-a)\left(\frac{-2ay}{x^3} + \frac{b}{x^2}\right) + (y-b)\left(\frac{a}{x^2}\right)\right] \quad \text{Eq. (4.22)}$$

Next, using the closed form formula derived in Eq. (4.22), we first estimate the variance of $g(x,y) = y/x$ using the second order Taylor approximation as shown in Eqs. (4.23)-(4.24) considering $a = E[x]$ and $b = E[y]$.

$$g(x,y) \approx \frac{b}{a} + [-\frac{b}{a^2}(x-a) + \frac{1}{a}(y-b)] + \frac{1}{2!}[(x-a)\left(\frac{-2ab}{a^3} + \frac{b}{a^2}\right) + (y-b)\left(\frac{a}{a^2}\right)] \quad \text{Eq. (4.23)}$$

$$= \frac{b}{a} + [-\frac{3b}{2a^2}(x-a) + \frac{3}{2a}(y-b)]$$

$$Var\big(g(x,y)\big) \approx Var\left(\frac{b}{a} + \left[-\frac{3b}{2a^2}(x-a) + \frac{3}{2a}(y-b)\right]\right) \quad \text{Eq. (4.24)}$$

$$= Var\left(\frac{b}{a}\right) + Var(-\frac{3b}{2a^2}(x-a) + \frac{3}{2a}(y-b))$$

$$= 0 + \frac{9b^2}{4a^4}Var(x) + \frac{9}{4a^2}Var(y) + 2\left(-\frac{3b}{2a^2}\right)\left(\frac{3}{2a}\right)Cov(x,y)$$

$$= \frac{9b^2}{4a^4}Var(x) + \frac{9}{4a^2}Var(y) + \frac{9b}{2a^3}Cov(x,y)$$

Utilizing the estimate of the variance of the function $g(x,y) = y/x$ shown in Eq. (4.24), we then derive the variance estimate of $g(x,y) = y/x$ as shown in Eq. (4.25) where N_s is the sample size.

$$Var\big(g(x,y)\big) \approx \frac{9b^2}{4a^4}\frac{Var(x)}{N_s} + \frac{9}{4a^2}\frac{Var(y)}{N_s} + -\frac{9b}{2a^3}\frac{Cov(x,y)}{N_s} \quad \text{Eq. (4.25)}$$

$$= \frac{9}{4a^2N_s}(\frac{b^2}{a^2}Var(x) + Var(y) - \frac{2b}{a}Cov(x,y)$$

4.4.2.3.3 Resampling Rule 1: Variance-based Relative Sampling Efficiency Rule

In the resampling step, the goal is to eliminate the particles with low weights while distributing more particles in more probable regions. The decision of resampling can be made based on different reasoning (rules). In our first contribution of this study,

we develop a stratified resampling rule, which is optimum in terms of minimized variance, by following the footsteps of a well-known stratified resampling rule, which was first proposed by Kong et al., (1994). In this section, by using the Taylor series approximation for variance of the ratio estimates, we develop an improved stratified resampling rule which focuses on the ratio of $Var_f[\hat{\mu}_3]/Var_p[\hat{\mu}_1]$ as a measure of relative efficiency between sampling from $p(\mathbf{x})$ and $f(\mathbf{x})$. To this end, following the explanation given in Eqs.(4.10)-(4.15), we obtain a closed formula relationship between $Var_f[\hat{\mu}_3]$ and $Var_p[\hat{\mu}_1]$ as derived in Eqs. (4.26)-(4.29).

$$Var_f[\hat{\mu}_3] = Var_f(\tfrac{\bar{z}}{\bar{w}}) \approx \tfrac{9}{4a^2N_s}(\mu^2 Var_f(\mathrm{w}) + Var_f(\mathrm{z}) - 2\mu Cov_f(w,z)) \quad \text{Eq. (4.26)}$$

$$= \tfrac{9}{4N_s}(\mu^2 Var_f(\mathrm{w}) + Var_f(\mathrm{z}) - 2\mu Cov_f(w,z))$$

where $a = \mathrm{E}_f[w] = 1$.

$$Cov_f(w,z) = \mathrm{E}_f[wz] - \mathrm{E}_f[w]\mathrm{E}_f[z] = \mathrm{E}_f[hw^2] - \mu = \mathrm{E}_p[hw] - \mu \quad \text{Eq. (4.27)}$$

$$= Cov_p(w,h) + \mu\mathrm{E}_p[w] - \mu$$

where $h = h(x)$.

$$Var_f(z) = E_f[w^2h^2] - {E_f}^2[wh] = E_p[wh^2] - \mu^2 \quad \text{Eq. (4.28)}$$

$$E_p[wh^2] \approx E_p[w]{E_p}^2[h] + \tfrac{1}{2}Var_p(h)\big(2E_p[w]\big) + Cov_p(w,h)\big(2E_p[h]\big) \quad \text{Eq. (4.29)}$$

$$= \mu^2 E_p[w] + Var_p(h)E_p[w] + 2\mu Cov_p(w,h)$$

Then, in our estimate of the variance $Var_f[\hat{\mu}_3]$, the resultant equation becomes as in Eq. (21).

$$Var_f[\hat{\mu}_3] \approx \frac{9}{4N_s}(Var_p(h)E_p[w] + \mu^2\big(1 + Var_f(\mathrm{w}) - E_p[w]\big) \qquad \text{Eq. (4.30)}$$

$$= \frac{9}{4}(Var_p(\hat{\mu}_3)\big(1 + Var_f(\mathrm{w})\big)$$

where $\frac{Var_p(h)}{N_s} = Var_p(\hat{\mu}_1)]$, and $E_p[w] = E_f[w^2] = Var_f(w) + 1$.

Finally, the proposed estimate of the ratio of variances $Var_f[\hat{\mu}_3]$, and $Var_p[\hat{\mu}_1]$ reduces to the following.

$$\frac{Var_f[\hat{\mu}_3]}{Var_p[\hat{\mu}_1]} \approx \frac{9}{4}\big(1 + Var_f(\mathrm{w})\big) \qquad \text{Eq. (4.31)}$$

Similar to the stratified resampling rule developed by Kong et al., (1994), the proposed improved stratified resampling rule does not involve $h(x)$. This makes the proposed estimation particularly useful as a measure of the relative efficiency of importance sampling. During importance sampling (e.g., within Sequential Monte Carlo algorithms), the effective sample size in the resampling stage is defined as below with $Var_f(w)$ being estimated by the sample variance of the standardized weights.

$$\frac{N_S}{\frac{Var_f[\hat{\mu}_3]}{Var_p[\hat{\mu}_1]}} \approx \frac{N_S}{\frac{9}{4}\left(1+Var_f(\mathrm{w})\right)} = \frac{4N_S}{9\left(1+Var_f(\mathrm{w})\right)} \quad \text{Eq. (4.32)}$$

4.4.2.3.4 Resampling Rule 2: Bias-based Relative Sampling Efficiency Rule

From the second order Taylor series expansion of $g(x,y)$ for a point in the neighborhood of the point (a,b), we obtain the approximation given in Eq. (4.33).

$$g(x,y) = g(a,b) + \left[(x-a)\frac{\partial}{\partial x} + (y-b)\frac{\partial}{\partial y}\right]g + \frac{1}{2!}\left[(x-a)\frac{\partial}{\partial x} + (y-b)\frac{\partial}{\partial y}\right]^2 g + \cdots \quad \text{Eq. (4.33)}$$

Given our function, $g(z,w) = z.w$, we determine the first and second order derivates as shown below.

$$\frac{\partial g(z,w)}{\partial z} = w, \quad \frac{\partial g(z,w)}{\partial w} = z, \frac{\partial^2 g(z,w)}{\partial z^2} = 0, \frac{\partial^2 g(z,w)}{\partial w^2} = 0, \frac{\partial^2 g(z,w)}{\partial w \partial z} = 1 \quad \text{Eq. (4.34)}$$

Furthermore, the approximation of our function, $g(z,w) = z.w$, can be denoted as follows.

$$g(z,w) \approx g(a,b) + b(z-a) + a(w-b) + \frac{1}{2}[2(z-a)(w-b)] \quad \text{Eq. (4.35)}$$

$$= a.b + b(z-a) + a(w-b) + (z-a)(w-b)$$

Now, if we take the expectation of this function knowing that $a = E_g[z]$ and $b = E_g[w]$, we obtain the general open form for the covariance.

$$E_g[zw] = E_g[ab + b(z-a) + a(w-b) + (z-a)(w-b)] \quad \text{Eq. (4.36)}$$

$$= ab + bE_g[z-a] + aE_g[w-b] + E_g[(z-a)(w-b)]$$

$$= ab + bE_g[z] - ab + aE_g[w] - ab + E_g[(z-a)(w-b)]$$

$$= E_g[z]E_g[w] + E_g[w]E_g[z] - E_g[z]E_g[w] + E_g[z]E_g[w] - E_g[z]E_g[w] + E_g[(z-a)(w-b)]$$

$$= E_g[z]E_g[w] + E_g[(z-a)(w-b)]$$

$$= E_g[z]E_g[w] + Cov_g(z,w)$$

Now, we switch to the notation used to determine our resampling rule as part of importance sampling. As mentioned earlier in Eq. (4.14), and repeated below is our variable of interest as the ratio of mean estimators.

$$\hat{\mu}_3 = \frac{\sum_1^{N_S} h(\mathbf{x}^i)w'(\mathbf{x}^i)}{N_S} = \frac{\frac{1}{N_S}\sum_1^{N_S} h(\mathbf{x}^i)w(\mathbf{x}^i)}{\frac{1}{N_S}\sum_1^{N_S} w(\mathbf{x}^i)} = \frac{\bar{z}}{\bar{w}} \quad \text{where} \quad z = h(\mathbf{x})w(\mathbf{x}) \quad \text{Eq. (4.37)}$$

The bias of the ratio estimator above can be determined as follows using the proofs from Koop (1951) and Koop (1976), respectively. It should be noted here that we used the letter f for our particular function of interest rather than g, which was used earlier to represent the more general derivations.

$$Bias(z,w) = E_f[z(\frac{1}{w} - \frac{1}{E_f[w]})] \quad \text{Eq. (4.38)}$$

$$E_f[z(\frac{1}{w} - \frac{1}{E_f[w]})] = -\frac{1}{E_f[w]} Cov_f(\frac{z}{w}, w) \quad \text{Eq. (4.39)}$$

Now, since $E_f[w] = 1$, the equation becomes as given in Eq. (4.40).

$$E_f[z(\frac{1}{w} - \frac{1}{E_f[w]})] = -Cov_f(\frac{z}{w}, w) \quad \text{Eq. (4.40)}$$

Provided $z = hw$, and $E_f[w] = 1$, our approximation of bias becomes as follows.

$$-Cov_f\left(\frac{z}{w}, w\right) = -E_f\left[\frac{z}{w}w\right] + E_f\left[\frac{z}{w}\right]E_f[w] = -E_f[hw] + E_f\left[\frac{hw}{w}\right] = -E_f[hw] + E_f[h]E_f[w] \quad \text{Eq. (4.41)}$$
$$= -Cov_f(h, w)$$

Therefore, the square of the bias, which is usually considered as part of the mean square error becomes as the following.

$$[Bias(z, w)]^2 = [Cov_f(h, w)]^2 \quad \text{Eq. (4.42)}$$

Finally, given a sample where w can be evaluated up to a constant, and $Cov_f(h, w)$ can be estimated by the sample covariance between the standardized weights and the distribution h, the bias based rule of effective sample size to be considered in the resampling stage of importance sampling becomes as the following.

$$\frac{N_s}{[Cov_f(h,w)]^2} \qquad \text{Eq. (4.43)}$$

4.4.2.3.5 Resampling Rule 3: Half Width-based Sampling Efficiency Rule

Assuming that the initial sample size during the importance sampling is N_s', $(1-\alpha)$ is the confidence level on μ where $0 < \alpha < 1$, σ_h is the desired half width, $t_{n-1,1-\alpha/2}$ is the value of a t-score in a two-tailed test with α-value, $n-1$ degrees of freedom using students t-distribution, and s is our current sample standard deviation, the probability of obtained values being in between the upper and lower control limits is given in Eq. (4.44).

$$P\left(-t_{n-1,1-\alpha/2}\frac{s}{\sqrt{N_s}} \leq \mu - \bar{x} \leq t_{n-1,1-\alpha/2}\frac{s}{\sqrt{N_s}}\right) = 1-\alpha \qquad \text{Eq. (4.44)}$$

The half width σ_h, is then defined as $\sigma_h = t_{n-1,1-\alpha/2}\, s/\sqrt{N_s}$ when we replace $z_{1-\alpha/2}$ with $t_{n-1,1-\alpha/2}$ based on the fact that a *t*-distribution with infinitely-many degrees of freedom is a normal distribution. Now if we solve the half width formula given above for N_s, we obtain Eq. (4.45).

$$N_s = t^2_{n-1,1-\alpha/2}\frac{s^2}{\sigma_h^2} \qquad \text{Eq. (4.45)}$$

To this end, if $N_s < N_s'$, then no further action is required since we can estimate the mean μ with a $(1-\alpha)$ confidence level using the current sample size. Otherwise if

$N_s > N_s'$, then we will need to increase the sample size by $N_s - N_s'$ samples in order to reach the effective sample size. By the definition of half width, $N_s \to \infty$ with probability 1 as $\sigma_h \to 0$. Assuming that the measurement errors are independent and identically distributed, if $\sigma_h > 0$, then $P(N_s < \infty) = 1$ and $lim_{\sigma_h \to 0} P(\mu + \sigma_h \leq \bar{x} \leq \mu + \sigma_h) \geq 1 - \alpha$ (for details of the proof, see Appendix A). Hence, the confidence interval formed is asymptotically valid as σ_h reaches to zero.

4.4.2.4 Almost Sure Convergence of Particle Filters

Almost sure convergence of particle filters has been focal point to many studies in the literature while it is not for this dissertation. However, in this section we provide the proof of asymptotical convergence of particle filters based on general stratified resampling rules regardless of its speed of convergence from Crisan and Doucet (2002). Here, we consider $w(x_t^i)$ which is the weight measure associated with the set of particles obtained at the end of the resampling step in the particle filter described earlier in this section. If we let (E, d) be a metric space and on this space, let $(a_t)_{t=1}^{\infty}$ and $(b_t)_{t=1}^{\infty}$be two sequences of continuous functions $a_t, b_t : E \to E$; and k_t and k be two other sequences of functions defines as $k_t \triangleq a_t(b_t)$ and $k_{1:t} \triangleq k_t(k_{t-1} \dots (k_1))$ After the resampling step (μ is the initial distribution of the function of interest),

$$w(\mathrm{x}_t^{N_s}) = c^{N_s}\left(a_t\left(c^{N_s}\left(b_t w\left(\mathrm{x}_{t-1|t-1}^{N_s}\right)\right)\right)\right) = k_{1:t}^{N_s}\left(w\left(\mathrm{x}_{t-1|t-1}^{N_s}\right)\right) \qquad \text{Eq. (4.46)}$$

$$w(\mathrm{x}_t^{N_s}) = k_{1:t}^{N_s}(c^{N_s}(\mu)) = k_{1:t}^{N_s}(\mu^{N_s})$$

where $\mu^{N_s} = c^{N_s}(\mu)$ and $w\left(\tilde{\mathrm{x}}_{t|t-1}^{N_s}\right)$=$c^{N_s}(b_t(w(\mathrm{x}_{t-1|t-1}^{N_s})))$.

The theorem of almost sure convergence and its proof is derived by Crisan and Doucet as follows. The analysis on the detailed asymptotical properties of particle filters can be conducted as part of the future work of this study where we can include the speed of convergence as well.

Theorem: Assuming that the transition kernel K is Feller and that the likelihood function is bounded, continuous, and strictly positive, then $\lim_{N_s\to\infty} w\left(\mathrm{x}_t^{N_s}\right) = \lim_{N_s\to\infty} w(\mathrm{x}_t)$ almost surely.

Proof: Since $\lim_{N_s\to\infty} \mu^{N_s} = \mu$ (from XX), then $\lim_{N_s\to\infty} w\left(\mathrm{x}_t^{N_s}\right) = \lim_{N_s\to\infty} k_{1:t}^{N_s}(\mu^{N_s}) = k_{1:t}(\mu) = w$. The proof is complete.

4.4.2.5 Parallelization Scheme

As mentioned in Chapter 4.4.2, for the current experimentation, three separate particle filters have been built for each of three different fidelities (Fidelities 1.X, 2.X and 3.X). Although these three objects can be called from each other in the main model, they are all located at one central computer. When the proposed fidelity selection algorithm is embodied in a DDDAM-simulation for a real distributed, large-scale system, this fidelity selection algorithm can be further split in order to allow process parallelization using the distributed computing resources. This parallelization schema can be implemented in two

ways. First, a separate particle filter can be built for each level of data retrieval frequency instead of the level of decision hierarchy. Then these filters can be called from the main model when the situation indicates that it is necessary. Since each filter has a different sampling frequency and sample size, the necessary ones can be selected based on the need and distributed resource availabilities. However, in this case, some of the sampling regions (a shop, a specific cell or a specific machine) for sensor data can overlap. For instance, particle filter for Fidelity 2.4 and 2.5 can try to access the same data of the same cell or machine, which may lead to either over-emphasizing some specific signals or repetitive consideration of the same information. The second and more concise way of parallelization requires keeping the particle filters as they are for each level of decision hierarchy whereas splitting the data regions among each fidelity of the data retrieval frequency. For instance, Fidelity 2.4 samples data from machines 1, 2, 3 whereas Fidelity 2.5 samples data from machines 4, 5, 6, all of which are located in the same cell (*e.g.* Cell #3). This way, no overlap occurs in terms of the sensory data used. In addition, some of the computational resources such as the ones closely located to specific machines can be dedicated to them. Therefore, no additional control system for resource allocation would be needed. While our work in this dissertation focuses on the realization of the actual sequential Monte Carlo-based fidelity selection algorithm (particle filter for each fidelity), the future work will include parallelization of the proposed algorithm based on the implementation scheme discussed in this section.

4.5 Algorithm 3 - Fidelity Assignment Algorithm

The purpose of Algorithm 3 is to opt for the available fidelity level of each component by taking the system level computational resource constraint into account. This algorithm decides the proper fidelity level of each component based on the desired fidelity obtained from Algorithm 2 and the currently available computational resource capacity obtained from the grid computing service. Therefore, the desired fidelity cannot be satisfied if the available resource is not sufficient. This algorithm is based on the well-known Knapsack problem (see Eq. (4.46)), where the available resources and resource requirements for each component are measured in units of computational power usage (*Ghz*hr*).

$$\min \ \sum_{i \in S} C_i (1 - x_i) \qquad \text{Eq. (4.46)}$$

$$\text{s.t.} \ \sum_{i \in S} R_i x_i \leq TR$$

where the corresponding parameters are

S : Set of components that require resource

C_i : Penalty cost if the resource requirement for component *i* is not satisfied

TR : Available resources (in units of Ghz*hr)

R_i : Resource requirements for component *i* (in units of Ghz*hr)

and the decision variable is

x_i : Binary variable indicating resource assignment for component i (1 if resource assigned, 0 otherwise)

Here, R_i is the function of the desired fidelity level obtained from Algorithm 2, and is defined for each component *i*. In this work, the range space (values) of this function was predefined via offline measurements before it was handed to the grid environment.

As the objective function denotes, the problem intends to minimize the penalty cost resulting from not fulfilling the computational resource needs of each component (each machine in the PM application). Moreover, in this case, we are unable to increase the computational resource on hand. Therefore, in order to minimize the total cost, we first assign the resources to the machines having greater penalty costs compared to others. To solve this problem, a greedy algorithm (widely known heuristic method for the knapsack problem) by Martello and Toth (1990) is used because it is efficient in terms of the execution time. Their algorithm sorts the items in a decreasing order of their penalty costs (C_i) and inserts all fitting items into the knapsack. The time complexity is defined as the sum of $O(n)$ and $O(n\log n)$ for this initial sorting. This method guarantees at least half of the optimum result. But, since the penalty cost C_i for each machine does not change frequently, the sorting process does not need to be performed in every step. So in this case (the greedy algorithm), the time complexity becomes $O(n)$.

In fact, the above mentioned penalty cost can be defined in various ways depending on the ultimate performance goal of the overall system and the relative contribution of each component toward this goal. Having higher penalty costs for the bottleneck machines allows the system to give a higher priority to those machines in terms of the measurement frequency and the amount of data to be collected in each

measurement. Thus, the system can adjust to highly efficient operating levels rapidly against the dynamic working conditions. For these reasons, in this work, higher penalty costs are assigned for the machines that have originated the bottleneck problem. To do so, the expected utilization values of the machines are estimated by the Arena® model as part of Algorithm 3, and then increased penalty costs are assigned in the order of increased machine utilization values.

As part of the future research, we will consider various methods of allocation of the computing resources for each component. For example, Chen et al. (1997) presents a gradient approach to optimally allocate the computing budget for selection of the best among the simulated designs. Their method is contingent upon the selection of the incremental computing budget where large increments may result in waste of resources, and in contrast very small increments may cause the allocation problem to be re-solved many times. Another related study has been conducted by Chen et al. (2008), where they developed an easy-to-implement heuristic sequential allocation procedure. This approach also reveals increased relative efficiency for larger problems and shares similar goals with our efforts in Algorithm 3 such as maximizing the probability of correctly selecting the designs (*i.e.*, machines in our case) and enhancing computational efficiency for simulation optimization. Therefore, we will extend our algorithm considering concept of optimal computing budget allocation in ranking and selection as part of our future research.

4.6 Algorithm 4 – Prediction and Task Generation Algorithm

The purpose of Algorithm 4 is online prediction and task generation where its aim is twofold. It provides a real system with 1) near optimal preventive maintenance (PM) scheduling recommendations for reliability of machines and 2) near optimal part routing (PR) recommendations for operational efficiency of jobs (parts) to be processed. This algorithm is comprised of three main steps, where Step I predicts mean time between failures (MTBF), Step II updates PM schedules based on the information obtained from Step I, and Step III updates PR schedules based on the updated PM schedules (see Figure 4.14). Details of each of these steps are explained in this subsection.

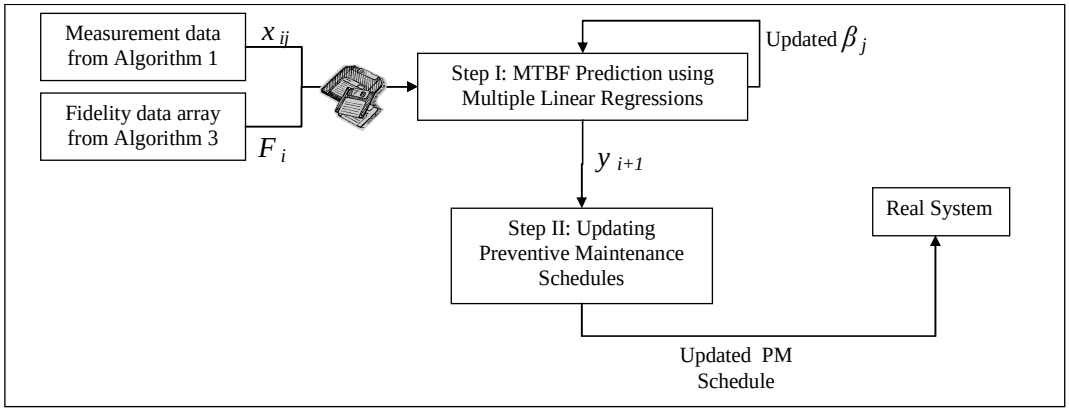


Figure 4.13: Operation of Algorithm 4

Step I: MTBF Prediction using Multiple Linear Regressions

Mean time between failures is defined as the mean (average) time between failures of a system, and is often attributed to the "useful life" of the device. Device in our study refers to machines in a shop floor. Calculations of MTBF assume that a system is "renewed", *i.e.* fixed, after each failure, and then returned to service immediately after repair. MTBF is a measure of reliability of a product. In this work, MTBF is computed

in units of days for easy elaboration. The higher the MTBF, the more reliable the product is. It should be noted that MTBF in this study stands for PM interval as well since maintenance is scheduled before an actual break down occurs.

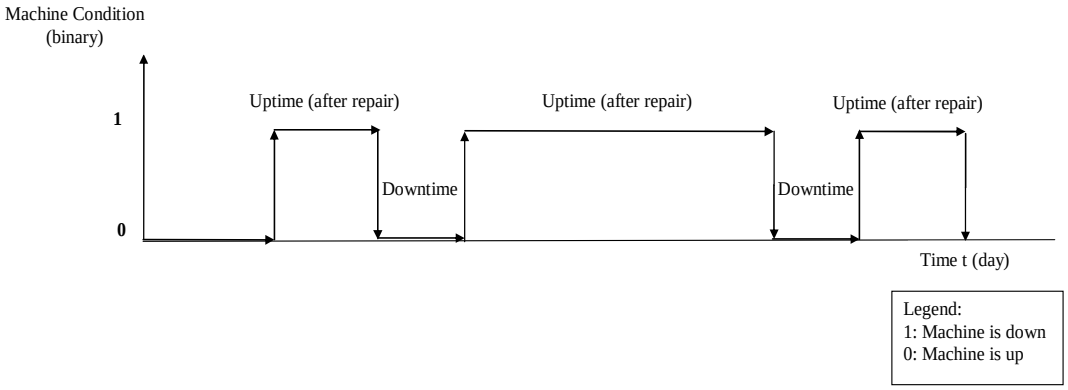


Figure 4.14: Mean time between failures (Source: http://mtbf.polimore.com)

$$MTBF = \frac{\sum(downtime - uptime)}{number\ of\ failures} \qquad \text{Eq. (4.47)}$$

Algorithm 4 in DDDAMS uses a formula (see Eq. (4.48)) to calculate the PM interval from a linear combination of the sensor inputs with variable weights assigned to each input. In Eq. (4.48), ε is the error term. Multiple linear regressions are setup with p coefficients, the regression intercept β_0 and n data points (sample size) for each machine under consideration. Parameter value of Y_{i+1} is then estimated for near optimal PM interval (MTBF) for period $i + 1$.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & & x_{2p} \\ \vdots & & & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \quad \text{Eq. (4.48)}$$

$$Y_i = \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j + \varepsilon_i \quad \text{Eq. (4.49)}$$

Eq. (4.49) depicts a vector-matrix notation version of Eq. (4.48), where the corresponding parameters are:

Y_i : PM interval (MTBF) for period i

X_{ij} : Sensor j measurement for maintenance period i

β_j : Weight assigned to sensor j measurement at period i

ε : Error term

The estimated values of the parameters β_j in every period i are given in Eq. (4.50).

$$\hat{\beta} = (X^T X)^{-1}(X^T Y) \quad \text{Eq. (4.50)}$$

The residuals (error terms), representing the difference between the observations and the model's predictions, are required to analyze the regression and are given by Eq. (4.51).

$$\hat{\varepsilon} = Y - X\hat{\beta} \quad \text{Eq. (4.51)}$$

Step I accepts two inputs: 1) the mean values for each sensor parameter during the specific period for the computation of updated weights, and 2) actually realized preventive maintenance interval by the maintenance team to tune up the model for better future results. The algorithm uses multiple linear regressions to fit the formula to the given data set, and returns the weights of the terms in the PM scheduling formula that provide the best fit. Initially, the formula uses weights derived as best estimates from historical data. As the simulation evolves, more data measurements and PM feedback is used to produce a formula that provides predictions of the best possible PM interval for the current maintenance period. Using multiple linear regressions, the weights in the desired formula can be estimated by solving the Eq. (4.50), where X is a matrix of the sensor measurements, Y is the near optimal interval estimates, and β is the vector of formula weights. Once these weights are known, they can be used along with current sensor measurement data to predict the near optimal PM interval for the next period. The predictions using this method of estimation are reliant on the accuracy of the feedback (actually realized preventive maintenance interval) coming from the maintenance personnel.

Step II: Updating Preventive Maintenance Schedules

In this step, the result obtained in Step I (the estimated MTBF of PM interval) for period $i+1$ is added up to the time since the last machine maintenance, in order to provide recommendations on when the next maintenance operation should take place while updating preventive maintenance schedule of each machine. The intention of

updating the preventive maintenance schedules is to save as much as possible from the maintenance expenses and at the same time to increase machine availability. Increased machine availability becomes crucial especially for the bottleneck machines. In this case, as the machine availability increases, production cycle time decreases.

Step III: Updating Part Routing Schedules using Fast Mode Simulation

Following the update of PM schedules of machines, part routing problem is assessed to even decrease the cycle time. The updated PM schedule forces the PR schedule to prohibit any potential part transfer to specific machines which are under maintenance in the current time period or will be under maintenance soon.

Step III of Algorithm 4 is implemented via Arena® fast mode simulation of the real system. The updated preventive maintenance schedules obtained in Step II is combined with previous part routing schedules in order to update part routing schedules. The resultant updated part routing schedules are then sent to real system as a task to be executed. Real system keeps executing this schedule until the next update where time to next update is determined by the current fidelity of the DDDAMS model.

In a die fFacility, 3 different types of jobs can be processed at a time. The process plans of these jobs are known and shown in Table 3.2. In this system, there is 1 dedicated cell for each operation which is comprised of 2 machines. In total, there are 5 work cells, namely, diffusion, etch, metals, photo and probe and 1 inspection cell. Once the customer submits a demand to the system, system knows its process plan but does not know what route (machines) parts should follow. The sequence of each machine to be

visited during the process of a job is called as part routing and determined dynamically using fast simulation of the actual system.

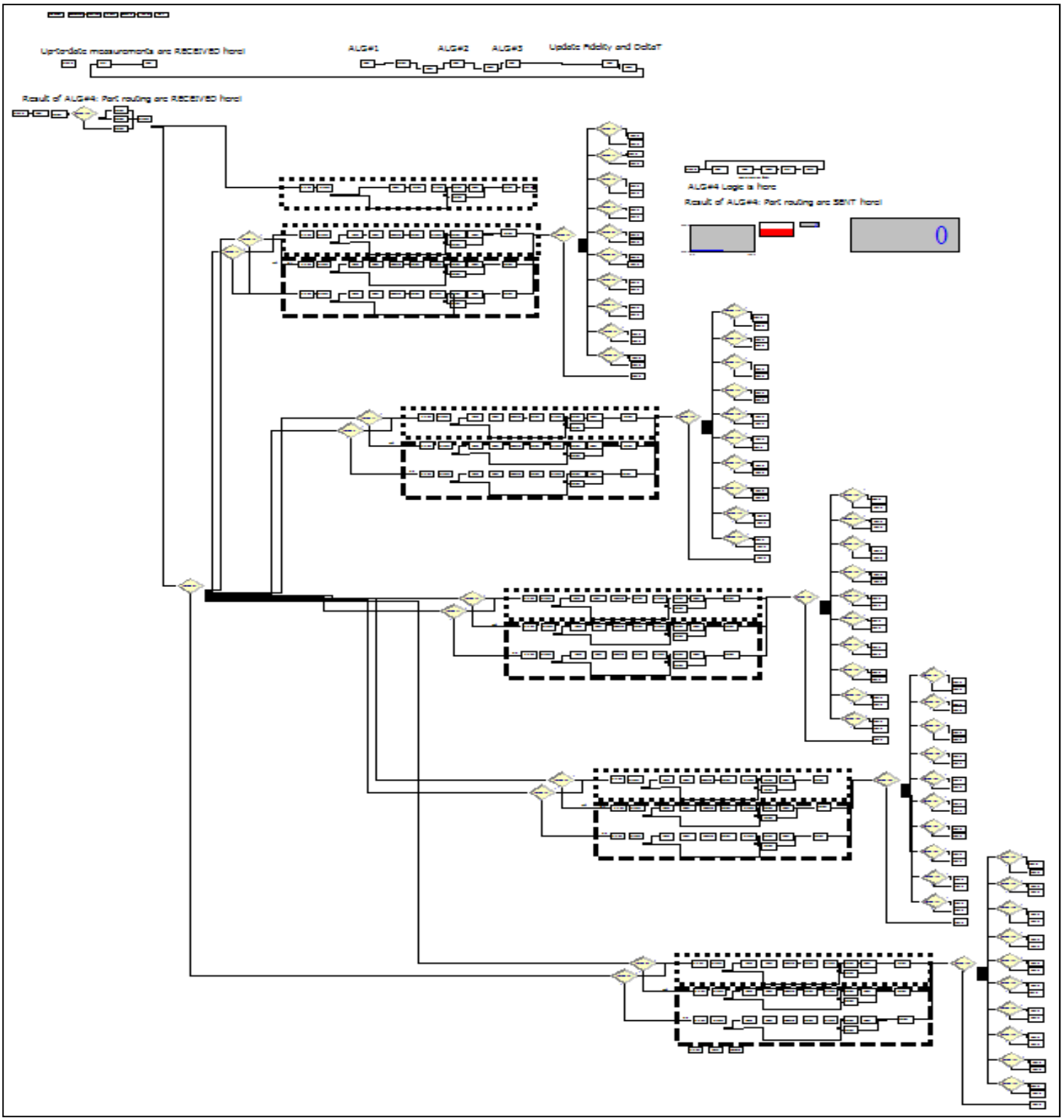


Figure 4.15: Updating part routing schedules using fast mode simulation

The fast mode simulation evaluates alternative part routings, and the best one is obtained via using OptQuest®. OptQuest® is based on scatter search, tabu search, and neural network, and is available as an add-on optimizer in most of the major discrete event simulation software. In this work, OptQuest® in Arena® is used to find optimal part routings based on the simulation-based evaluations.

In our case, a decision is to determine specific machines to be used in each cell for the upcoming orders in order to minimize mean cycle time. To this end, OptQuest® runs another fast-running simulation (as opposed to the DDDAM real-time simulation) of the system. After finding the near optimum solution, it records the machines to be used in a .txt file from which the DDDAMS model can read in. The DDDAMS model then, updates the part routing for each up-coming order by assigning the machines to perform the specific processes specified in the process plan (see Table 3.2). The frequency of the update of the part routing here depends on the dynamicity of the system (order arriving behaviors; system fluctuations).

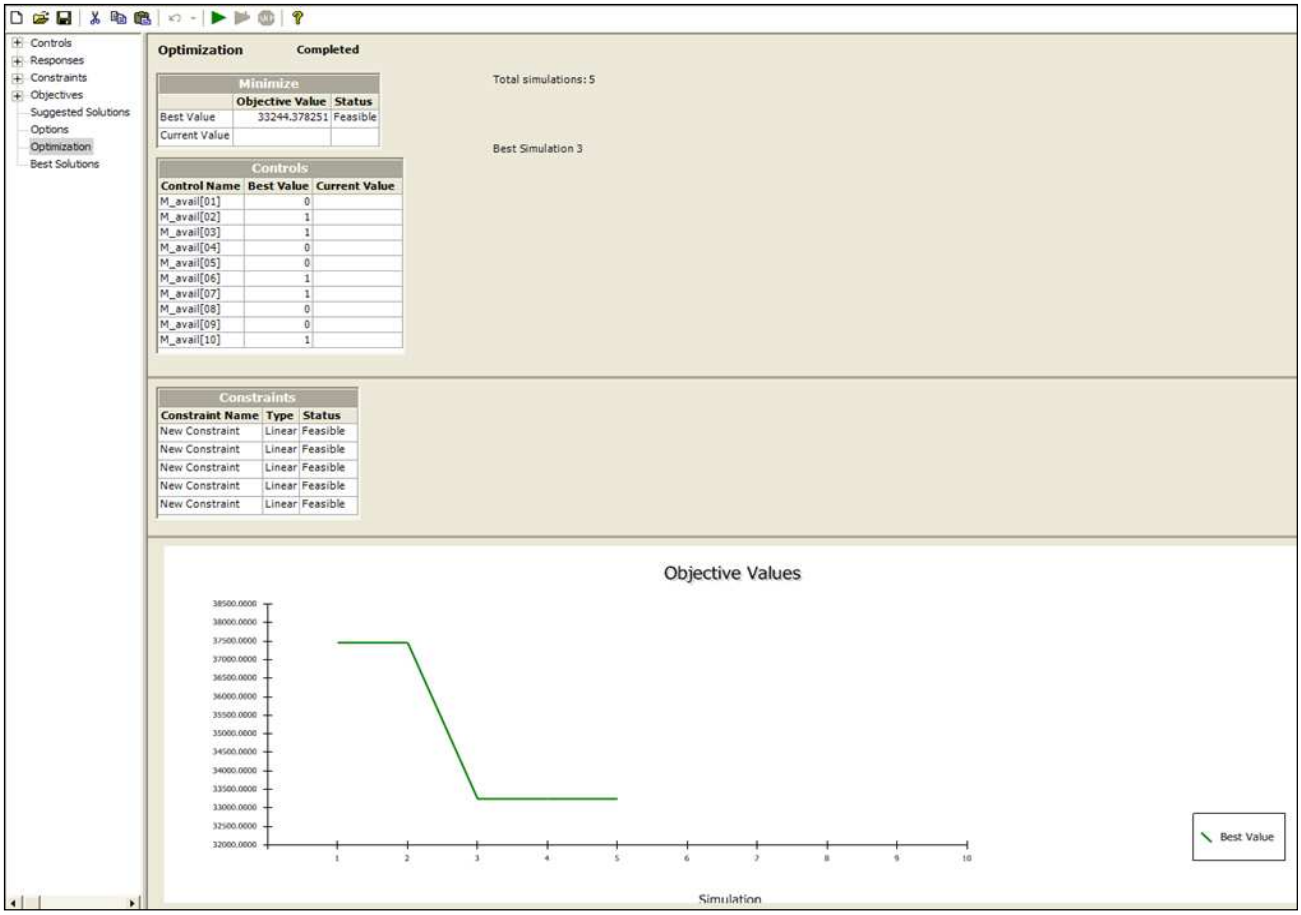


Figure 4.16: Determining machines to be used to minimize mean cycle time by OptQuest®

4.7 Concluding Remarks

In this chapter, we have described a novel framework to enable dynamic data driven adaptive simulations for monitoring and control of a large scale and complex supply chain. Incorporation of up-to-date data to the decision-evaluation and control system is significant to capture the dynamically changing system status. In this study, the measurement process is driven by Algorithms 1, 2, and 3, which determine the most appropriate fidelity level (the level of detail of simulation as well as control system) as

well as the level of data that needs to be incorporated given the available computational resources. While these three algorithms work in synch with each other in order to assign the current fidelity of simulation model, Algorithm 2 involving Bayesian inferencing via particle filtering is a major module as it determines how much information is necessary to appropriately monitor and control the system while the conditions evolve over time. In this work, we have enhanced the efficiency of the generic particle filtering algorithms via improvements made in the resampling techniques, namely variance-based resampling and bias-based resampling efficiency rules, respectively. Then, we have revisited a half width-based resampling rule for the benchmarking purposes. Here, the proposed rules are first derived theoretically and their performances are benchmarked against the performances of the resampling rule developed by Kong et al. (1994) and half width-based resampling rule revisited in this work in terms of their resampling qualities and computational efficiencies using a simulation study. In this section, we have explained the details of measurement process together with the decision mechanism of the proposed methodology, which involves four separate algorithms focusing on different venues of this research. In the following section, we discuss the realization of the proposed framework in the virtual computing setting involving grid computing and Web Services.

CHAPTER 5

REALIZATION OF DDDAMS IN VIRTUAL SETTING

In this section, we first explain the details of the real time DDDAM-Simulation developed as the backbone of this study. Second, the grid computing framework which is employed to provide parallel computing setting for the proposed DDDAMS system. Lastly, the issues regarding the time synchronization algorithms and communications among DDDAMS components are discussed.

5.1 DDDAM-Simulation Model in Arena®

This section discusses modeling details for the DDDAM-Simulation which has been developed for the preventive maintenance and part routing applications. In this work, Arena® 12.0 is used to build the DDDAM-Simulation as well as to mimic a real system (a test shop floor). While Arena® 12.0 is used for the illustration and demonstration purposes, the methodologies and system architecture proposed in this dissertation are independent of the simulation software. The real-time feature and messaging capability of Arena® allow it to effectively interact with other software and computational components (e.g. fast mode simulations; real-time simulation mimicking a real system). Also, they will allow for the future expansion of the current prototype model to include sensors and hardware components for real-time shop floor control.

The DDDAM-Simulation is split into two distinct modules: the algorithms sub-model (ASM) which runs in real-time and the process sub-model (PSM) which is called

by the ASM and runs in a fast mode. The ASM is further subdivided into each of the four algorithms (see Chapter 4). The ASM is a loop that reads in sensory data from the real system, where the update frequency is determined by the fidelity of each component of the real system. The PSM is used to model the different levels of fidelities of each component in the simulation while essentially mimicking the output of the real system. In PSM, each process is simulated based on the selected fidelity from the ASM. If the ASM concludes that the system is operating under normal conditions assuming that for instance the current fidelity level of decision hierarchy is 1, this would have two affects. First, the amount and type of data pulled from the real system become minimal and second, the level of detail of the fast mode simulation is set to the most aggregated level. Here, we lose some degree of information intentionally as we do not want to utilize our computational resources unnecessarily. On the other hand, if the algorithms in the ASM conclude that the system's conditions have changed assuming that for instance the fidelity level of decision hierarchy is currently 2 and should be 3, the amount and type of data pulled from the real system becomes maximal and the level of detail of the fast mode simulation is set to the most detailed level.

The ASM of the DDDAM-Simulation dynamically adjusts its level of fidelity based on the sensory data updates received from the real system and available computational power while running in real time to control the actual system. Hence, the actual data that drives the real time simulation comes from the real system. Once the model fidelity is determined by the algorithms embedded in to the ASM, this real-time simulation activates a fast mode simulation of a particular level of fidelity (hierarchical

level of fidelities 1, 2, or 3 in the current case) to predict/evaluate the future behavior of the system. Therefore, during the run of this fast mode simulation, the fidelity (and generated sample path) does not change. This fast mode simulation mimicking a network ("Job Shop") of machines with queues in front of them is the PSM of the DDDAM-Simulation model. It is noted that the PSM module can be implemented via a monolithic simulation model (which has been employed by our current approach) or multiple simulation models that are federated in the distributed computing environment. The model incorporates routing rules which are distinct for each part type. Each part holds an index (or step) number and a part type as attributes, and the simulation model uses them to route parts to appropriate machines. Movements of parts between machines are performed manually by committed personnel using carts. Intra-machine movements are handled by dedicated robots. Once parts arrive in the machine queue, a dedicated robot selects parts waiting in the queue for processing according to rule of first come first serve (FIFO). While lot prioritization has not been considered in the current simulation model, it is a possible extension of the model.

Processes in PSM can be modeled via different modeling techniques, such as different statistical distributions, differential equations, regression models, neural networks, or process simulators for each fidelity considering the modeling accuracy and computing power. For instance, if we need an aggregated insight of a process parameter, we may sample a value from an appropriate distribution. In contrast, in a higher fidelity, if we want to obtain more accurate estimates and are willing to pay for its computational expense, we may employ neural networks or detailed process simulators. Equation (5.1)

depicts a generic formula (function *f*) to mimic a process, and the Integrated DEFinition (IDEF0) functional diagram in Figure 5.1 depicts inputs, outputs and controls for the diffusion operation in a given fidelity. Among many output parameters, cycle times (in PSM) and MTBF (via Algorithm 4 in ASM) are considered in this work. f can have different inputs and outputs for different fidelities (*e.g.*, shop level, cell level, or process level). Equations (5.2) and (5.3) depict f in fidelity 1 and fidelity 2 (hierarchical level of fidelity), respectively. The predicted machine cycle time is used to keep the loops in the real time DDDAM-Simulation in sync with the actual machine cycle times in the real system. The cycle time of a machine is affected by various factors in the shop floor. For instance, if the temperature of a machine becomes abnormally high or low, its functioning diverges from its ideal behavior causing an increase in the machine cycle time. In Equation (5.2) $NormalCycleTime$ and $GoalTemp$ denote the ideal machine cycle time and temperature under normal conditions, respectively. The term $\left(\frac{|Temp_1 - GoalTemp|}{GoalTemp}\right) a$ denotes the normalized affect of deviation from the optimum temperature multiplied with an effect (proportionality) constant a. Effect constants represent how much their corresponding terms affect the $ActualCycleTime$, and they are either given by the system experts or estimated through experiments. When there is no deviation in any of the sensor types, then $ActualCycleTime$ is estimated as being equal to $NormalCycleTime$. Also, in Equation (5.2), since the sensor data comes from multiple sensors (two for each sensor type in this case), the average of these values are considered in the approximating functions. In this study, the effect constants (a through g) used in these functions have been developed based on the best of our knowledge. As a

future research, we plan to conduct extensive experiments to develop a more accurate function f (coefficients for each factor) in various fidelities. In addition, we will also investigate the accuracy and corresponding computing requirements for each level of fidelity; in this dissertation, the computing requirements for models with different fidelity have been reasonably assumed.

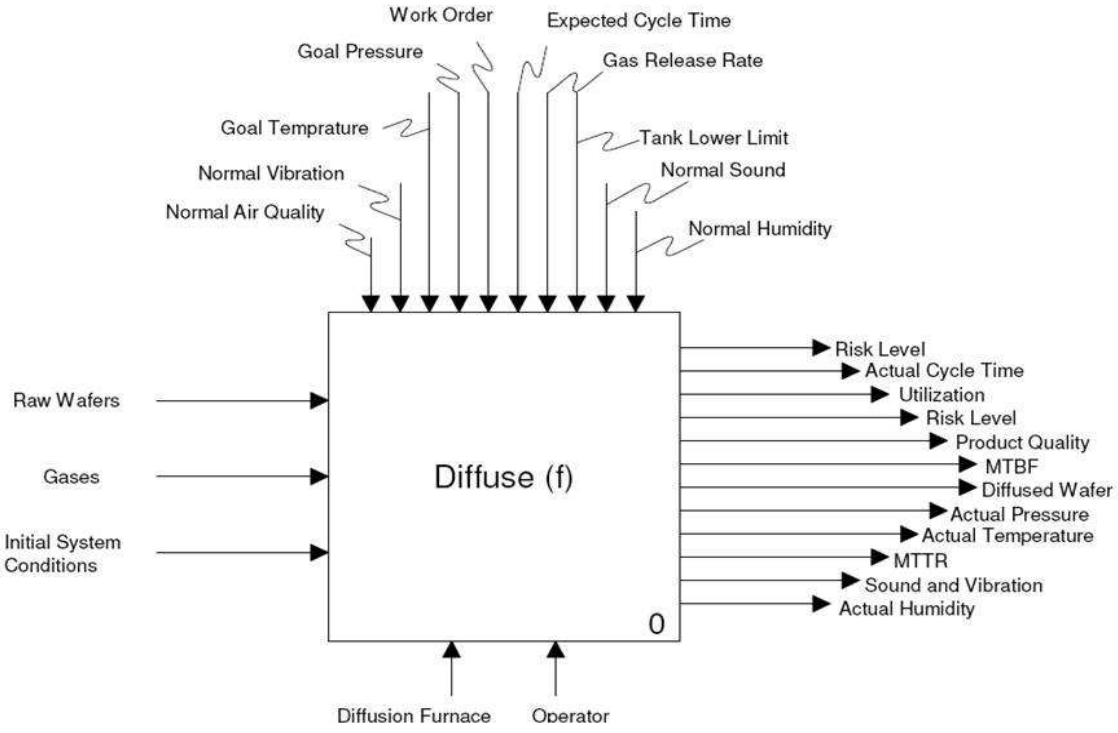


Figure 5.1: Function for output prediction – IDEF0 diagram

$$Output_i \approx f\left(Input_1, Input_2, \dots, Input_j, Control_1, Control_2, \dots, Control_k\right) \quad \text{Eq. (5.1)}$$

where i, j, and k are the number of outputs, inputs and control respectively

$$ActualCycleTime \approx NormalCycleTime + \frac{|Temp_1 - GoalTemp|}{GoalTemp}.a + \frac{|Pres_1 - GoalPres|}{GoalPres}.b$$
$$+\frac{|Vib_1 - GoalVib|}{GoalVib}.c + \frac{|Snd_1 - GoalSnd|}{GoalSnd}.d + \frac{|AirQual_1 - GoalAirQual|}{GoalAirQual}.e + \frac{|Hmd_1 - GoalHmd|}{GoalHmd}.g \quad \text{Eq. (5.2)}$$

$$ActualCycleTime \approx NormalCycleTime + \frac{|AvgTemp - GoalTemp|}{GoalTemp}.a + \frac{|AvgPres - GoalPres|}{GoalPres}.b$$
$$+\frac{|AvgVib - GoalVib|}{GoalVib}.c + \frac{|AvgSnd - GoalSnd|}{GoalSnd}.d + \frac{|AvgQual - GoalAirQual|}{GoalAirQual}.e + \frac{|AvgHmd - GoalHmd|}{GoalHmd}.g \quad \text{Eq. (5.3)}$$

where a, b, c, d, d, and g are the number of outputs, inputs and control respectively

The ASM uses a timing entity (agent) to execute each algorithm once in a cycle of length δt. Once δt has elapsed, the agent repeats its operation with different parameters for a different component (machine). The interactions between the real system and DDDAS simulation is modeled in the ASM loop. The communication is implemented through the web services explained in Chapter 5.4. The algorithms in the ASM loop are executed in sequence: Algorithm 1 populates the sensor machine-id array, which is used by Algorithms 2 and 3. These in turn populate the fidelity machine-id array and also machine-id array. The δt machine-id array is essentially an array that stores the δt value for each machine in the system. As discussed earlier, the δt value indicates the next point of data collection for a machine based on the fidelity that it is running at. Algorithm 4 then uses these arrays to update the preventive maintenance schedule (in the generic case this will be any suitable control action). Algorithm 4 also requests the data from the real system's data sources for the next loop of the ASM. Once the new data is received, the loop is re-executed after a δt time period. Figure 5.2 is a

snapshot of the DDDAMS system monitoring tool (that we have developed), which displays the sensory data, requested fidelity for each machine (result of Algorithm 2), assigned fidelity for each machine (result of Algorithm 3), and MTBF (output of Algorithm 4).

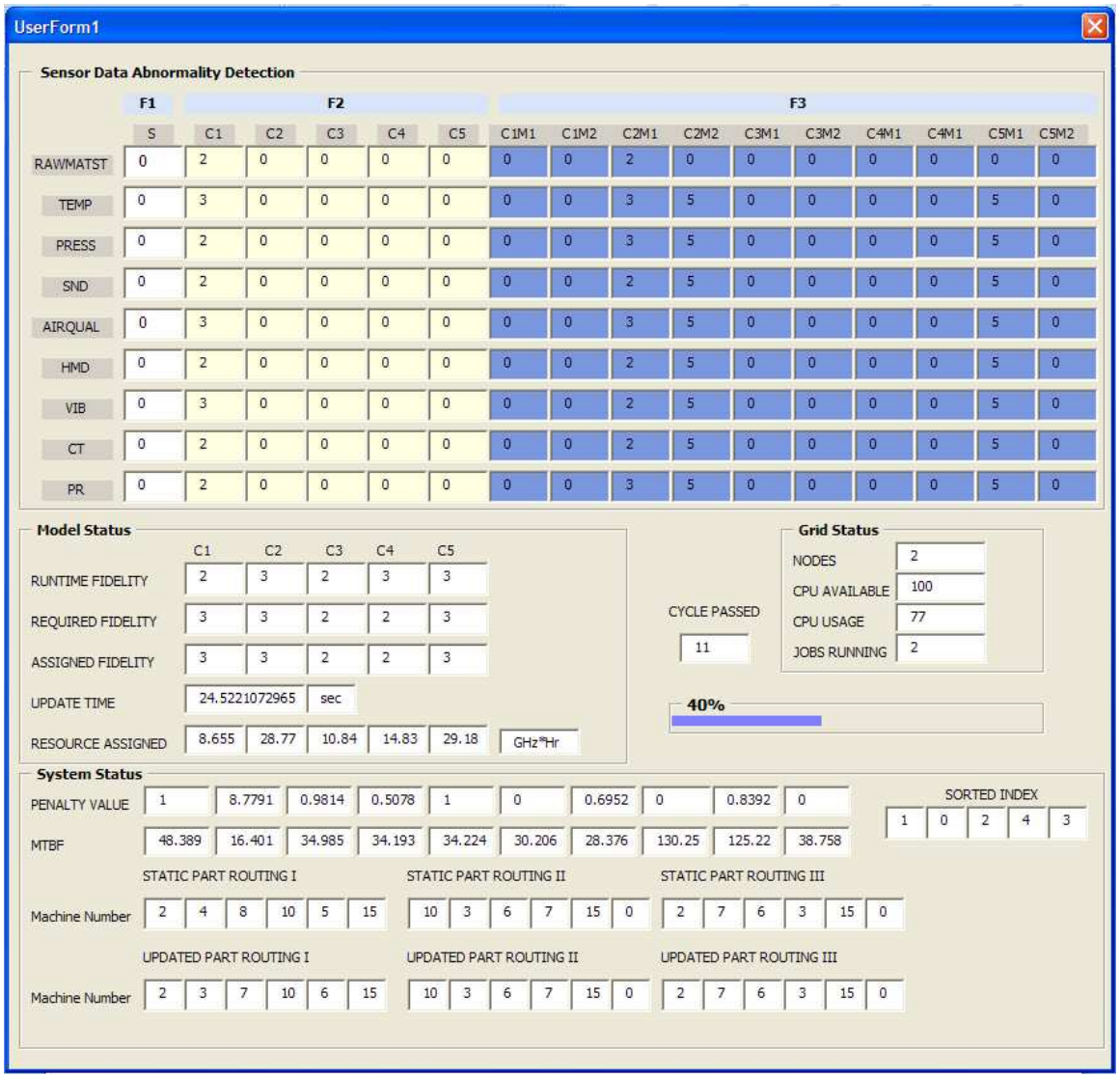


Figure 5.2: An instance of system status and results of the algorithms in DDDAMS model

As mentioned earlier, due to the unavailability of an actual shop floor for the testing, a real-time simulation model (which will be replaced with a real shop floor in the future) was used to mimic the real system. The real-time simulation model has been built in a way that allows it to be directly connected to real sensors in hardware components in a shop floor and still run with fair ease. An extension of the current DDDAMS system would be to incorporate this hardware structure in place of the real system simulation which will be quite challenging for two main reasons. First, developing a simulation model for shop floor control in an appropriate level of detail is a complex and costly task, which requires dedicated expertise. Second, both the time and communication synchronizations in-between the real shop floor and its executing simulation are challenging. However, real-time shop floor control can be facilitated by using a real time -RT feature of the Arena® simulation package, where the simulation model interacts directly with a shop floor execution system by sending and receiving messages. Son et al. (2003) presents more details about simulation-driven shop floor control based on the research conducted at The University of Arizona, The Pennsylvania State University, and Texas A&M University. This future research prospect was taken into account in every aspect of the proposed study including development of the web service, message based communication structure, so that added federates (e.g. hardware components) can be fully incorporated into the simulation.

5.2 Distributed Simulations over Grid Environment

Grid Computing is the automated sharing and coordination of the collective processing power of many widely scattered, robust computers (Cottenier and Elrad, 2004), and is used in our research for four reasons. First, it allows for loose coupling between both heterogeneous and geographically dispersed grids (computational resources participating in Grid Computing). Many previous works have demonstrated how grid computing assists achieving loose coupling (*e.g.*, Cottenier and Elrad (2004) have demonstrated it using the Globus Toolkit). Second, connecting a new grid or disconnecting an existing one is considerably easier compared to that of other ways of distributed computing such as Cluster Computing. A main reason is that resources are managed by a centralized resource manager in Cluster Computing whereas in Grid Computing, every node is a self-governing entity, having its own resource manager and behaving independently (Walter, 2004). Third, it helps us build an adaptive infrastructure especially when the computational availability changes dynamically in the environment. Ongoing researches and studies propose different techniques to provide adaptive infrastructure in grid computing environment. Forth, a distributed computational grid provides us with the ability of monitoring and utilizing the computational resources as necessary (Lee et al., 2003; Takeuchi et al., 2004).

To set up a grid computing framework, we have employed Alchemi Grid toolkit (an open source software) for several reasons. First, distributed simulations in this study are built in Arena® simulation package which is a native Microsoft (MS) Windows application. However, most commonly used grid toolkits as well as their components are

UNIX based, therefore not suitable for running Arena® simulations. Alchemi, on the other hand, is a MS Windows (.NET) based application. Second, it uses Web Services technology for interoperability with other grid middleware. Third, as opposed to many other toolkits offering many services such as authorization, resource discovery, monitoring, and management, which lead to an inefficient Grid framework involving a complex setup and overhead, Alchemi offers a smaller set of major services with a light weighted and efficient setup. Here, computers on the grid may be part of the local network or spread out over the Internet thus enabling setup of geographically distributed grids. This platform is comprised of one or more Manager and Executor nodes. The Manager (controller) is responsible for resource discovery, job scheduling, dispatching and monitoring. The Executor is a worker agent that runs on every grid node. The user of the grid submits tasks to the Manager through a job submitting interface.

In this study, a distributed computational grid has been established and used to facilitate monitoring and utilizing the computational resources as necessary in a distributed computing environment (federation of the partitions) for Algorithm 3 and Algorithm 4. As we have mentioned earlier, this research involves analysis of large number of machines in a supply chain system. When all of the machine's optimal fidelity level is selected to be 1 (the most aggregated model) by Algorithm 2, the DDDAM-Simulation needs to collect sensory data for each machine within a δt time interval as shown earlier in Chapter 4. Collecting, processing, and analyzing data from the sensors in a dynamic, multi-institutional environment demand considerable computing resources. As the scope of the model in the supply chain increases or the time interval δt decreases,

the required computational resources increases. Then, Algorithm 3 determines and assigns an adequate fidelity level for each of the machines in the system by monitoring the resource availability via the Grid Computing Manager (which is responsible for resource discovery, job scheduling, dispatching and monitoring). Later, once the fidelity level is assigned in the DDDAM-Simulation and data is collected from the real system, Algorithm 4 uses the Grid Computing to fire up and run the necessary fast mode simulations in the computers which are declared to be a worker agent that runs on every grid node. The user of the grid submits tasks to the Manager through a job submitting interface.

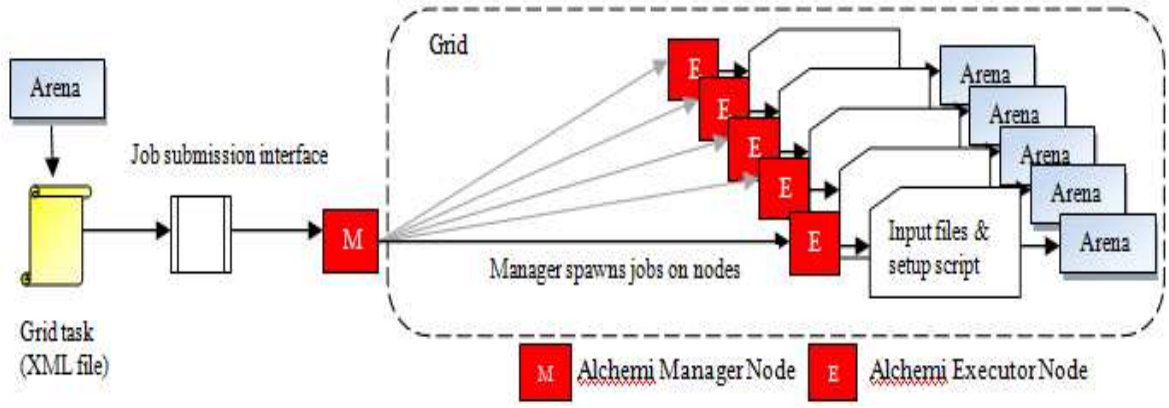


Figure 5.3: System setup with Arena® and Alchemi grid

In this study, grid interface is used from inside the Arena® model to submit tasks (Algorithm 4). The task is defined in an XML file conforming to the XML schema used by Alchemi. It lists all the input, executable and output files required for a simulation

model to be run on the grid. As part of a task, an executable that automates invoking Arena® on the grid node, an Arena® model, Arena® interoperability components and other initial setup scripts which are required for automation, are submitted. This job is then processed by the Manager through the job submission interface. The Executor then gathers all the files related to the task and invokes the automation executable which spawns the simulation model using Arena® interoperability components. In this study, we constructed an Alchemi based grid with five nodes having identical technical specifications, three of which are used for the initial experiments of this study (see Figure 5.3). The number of nodes included in Grid may be increased as the scale of the experiment is enlarged in the future. Figure 5.3 depicts the system setup with Arena® and Alchemi grid.

5.3 Communication using Web Services and Time Synchronization

For the execution of RT simulation, there is a need of communication between heterogeneous and distributed system simulations. This communication in this dissertation is facilitated by the communication server that has been developed in Lee et al. (2008) using web services technology (state-of-the-art distributed computing technology) that overcomes barriers of standard communication via the usage of W3C (http://www.w3c.org) standard protocols including XML, WSDL, and SOAP. The essential function of the communication server is to facilitate time and data management. It provides a communication backbone which complements the computation backbone provided by grid-based computing. Web Services provide an elegant mechanism for

communicating among distributed components without keeping much state information, as is required in conventional means such as socket based interaction.

When the simulations take their place on the grid and start running, the communications between these partitions should be synchronized to minimize any delayed information between these partitions and have their simulation clocks as close as possible for statistical accuracy while they move on with their own simulated part of the system. Delayed information in this study is considered as delay of notification of any external interaction event between partitions.

As discussed earlier in Chapter 2, synchronization between the partitions in a distributed environment can be enabled in three ways, including conservative, optimistic, and epoch. In this study, Epoch synchronization scheme with time intervals (Δt) is employed, where appropriate time intervals are determined based on the off-line simulation analyses. During the execution of epoch based synchronization, the next epoch (Δt) is fired at unison when all activities in all the partitions are completed for the previous epoch time interval. When the next epoch is fired, the necessary communications as well as previously scheduled entity routing steps take place. In the partitioned model, the routing of entities to their next partition after they are processed in a partition is enabled using a predefined map. This predefined map includes the information of which station belongs to which partition. Therefore, when an entity is to be routed to a next station, the predefined map is referred to distinguish whether it is an intra-partition routing (when the destination station is within the current partition) or an inter-partition routing (when the destination station is within the other partitions). While

intra-partition entity routings are done simply by sending the entity to its destination station within the same model, inter-partition entity routings are done via the Web Services and are accompanied by a communication message to inform the next partition regarding the attributes of incoming entity.

5.4 Sources of Computational Requirements in Simulation Execution

As discussed in Section 4.5, execution of Algorithm 3 needs to have computational requirements of simulation with varying levels of details and the current computational availabilities. The same sources of computational requirements become also of critical importance when we discuss the partitioning methods in extended DDDAMS framework in Chapter 7. In order to investigate various sources of computational requirements during simulation execution, we collected running (execution) times before and after individual and groups of blocks in the Arena® model. For better accuracy, we collected the average values for a set of entities instead of individual entity times. This was done by reading the internal "Timer" variable at different points and using "Tally" block for averaging the collected data.

As mentioned in Section 5.1, the PSM module can be implemented via a monolithic simulation model or multiple simulation models that are federated in the distributed computing environment. For the case of distributed simulation, it is observed that considerable running time is spent either for the synchronization and other interaction messaging between various partitions or for the internal computation of process modules. The simulation running time spent for synchronization and other

interaction messaging can be determined by measuring and comparing the running times of the original model and the partitioned model where the models do not include high intensity computational modules (*e.g.*, conveyors, material handlers, processes involving differential equations). Here, the simulation running times are comprised of their internal computation, times spent for synchronization and times spent other types of interaction messaging. However, in the case where simulation models are comprised of high intensity computational modules in a greater scale, the time spent for communication becomes relatively less significant compared to that for internal computation of these modules.

Another source of computational resource usage is the course of data retrieval. During its run time, depending on the specified fidelity, the simulation retrieves data from various types of sensors placed throughout the shop floor at various frequencies. Once a current status of the sole shop floor and whole supply chain is observed on the desired detail level, analysis of data and generation of tasks is carried out via algorithms embedded in DDDAM-Simulation using VBA blocks.

Data retrieval in this setup is substantially challenging for a number of reasons. First, system under consideration is wide-scoped and highly dynamic where data needs to be acquired in a multi-institutional environment at a considerably fast pace. A viable solution needs to provide a readily available pool of computing resources, which is built to meet the total computational demand. Second, these resources need to be provided in a transparent, distributed manner among requestors depending on the priority of requirement. If the necessary computational power cannot be met from available

resources at the local site, a request is sent to the manager of the resource pool for permission to use collaborators' excess resources. All of these requirements can be met using a distributed computational grid which provides computational resources that can be monitored and utilized as necessary.

5.5 Concluding Remarks

This chapter discussed modeling details for the DDDAM-Simulation. While in this work, Arena® 12.0 has been used to build the DDDAM-Simulation for the illustration and demonstration purposes, the methodologies and system architecture proposed in this dissertation are independent of the simulation software, and can be applicable to other simulation packages.

Next, the sources of computational resource usage have been pointed. These sources include the computation of internal blocks and efforts needed for synchronization as well as communications. In order to facilitate the usage of distributed computing resources we have set up a grid computing framework using Alchemi Grid toolkit (an open source native Microsoft (MS) Windows application). Furthermore, the communications between distributed simulations using Web Services have also been discussed.

CHAPTER 6

EXPERIMENTS AND RESULTS

In this chapter, the benefits of the proposed DDDAMS framework are demonstrated for the considered supply chain (see Chapter 3), where preventive maintenance and part routing schedules are dynamically obtained based on the real-time sensor data while saving computational power usage. To this end, the dynamic selection of fidelity in the simulation model is first demonstrated. Second, the results obtained regarding the performances of the fidelity selection algorithm (Algorithm 2) and the prediction and task generation algorithm (Algorithm 4) are provided. In addition, the performance of the fidelity selection algorithm is compared when different resampling rules (those developed as part of this dissertation and others available in the literature) are employed. Third, the performance of the system under the monitoring and control of the proposed DDDAMS framework (in terms of production cycle time and machine utilization) is compared against that obtained from the most aggregated (computationally light) and the most detailed (computationally heavy) models of the same system. As mentioned in Chapter 4, in the proposed DDDAMS system, machine maintenance is scheduled only when the sensors indicate a potential failure. As a result, we can avoid unexpected machine breakdowns, unnecessary maintenance, and downtime for problem identification, which allow us to achieve improved machine utilization and mean cycle time.

The DDDAM-Simulation model aims to simulate a semiconductor die manufacturing fab with demand information received from its upstream wafer manufacturing fab and throughput rate information sent to its downstream assembly and packaging fab (see Chapter 3 for details of the semiconductor manufacturing supply chain). The considered semiconductor die manufacturing fab is comprised of ten cells each having ten machines (a total of 100 machines in this echelon). In this semiconductor die manufacturing fab, six different types of die families are produced, where die production may need re-entrant flow of processes up to 20 times (see Table 3.2). As such, the considered semiconductor die manufacturing fab as well as its corresponding simulation model are quite complex. Therefore, it is crucial to make the selection of the most appropriate fidelity and to save computational resources while not loosing model accuracy. The DDDAM-Simulation in this work has been built using Arena® 12 software package, and the developed fidelity selection, fidelity assignment and task generation algorithms enabling its DDDAM capability have been embedded into the simulation using Visual Basic for Application (VBA) modules. A screenshot of the constructed DDDAM-Simulation is shown in Figure 6.1.

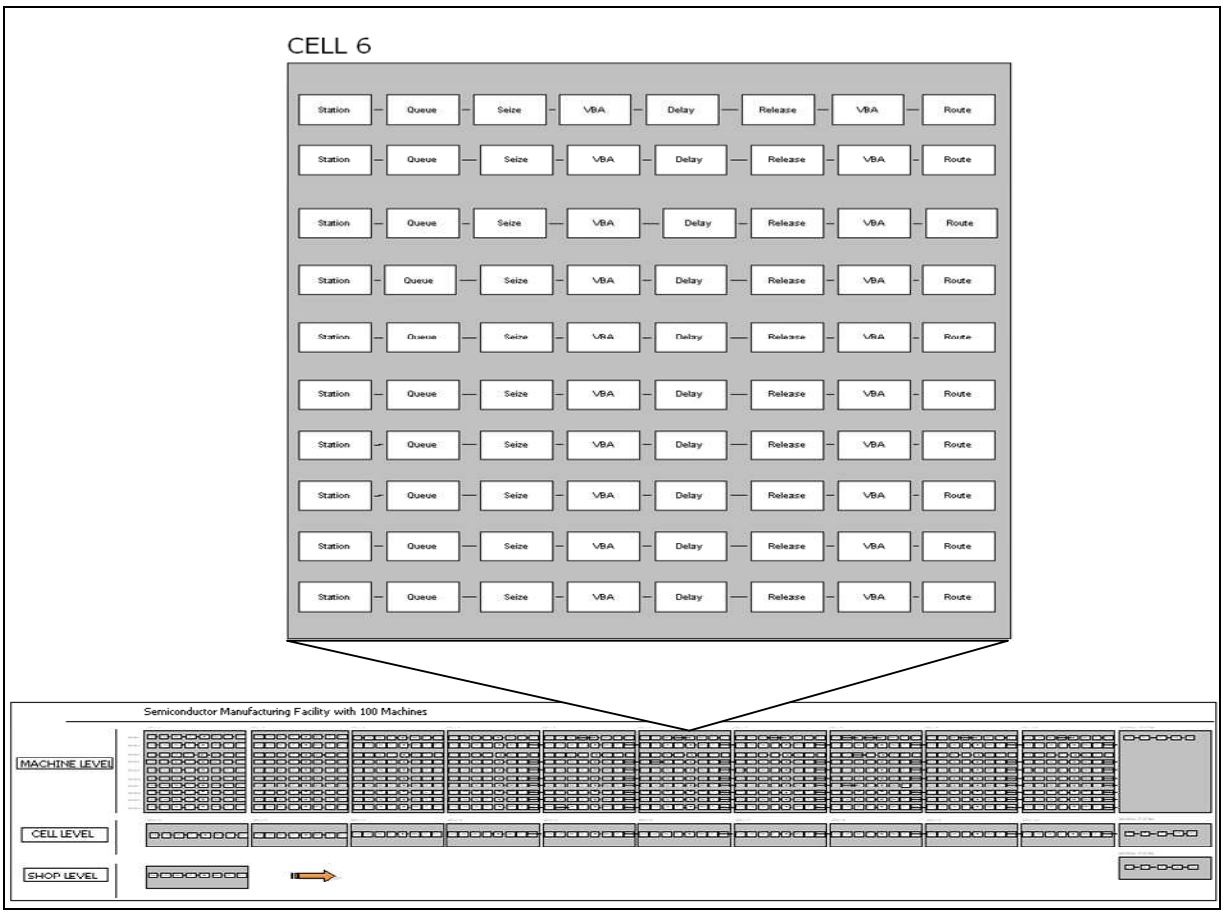


Figure 6.1: Partial snapshot of a DDDAMS simulation model (PSM) of semiconductor die manufacturing fab with 100 machines and re-entrant process flow

6.1 Demonstration of Dynamic Fidelity Selection

In this work, a system status is determined by a collection of statuses of the machines in terms of their mean time between failures (MTBF). This estimate of MTBF of the machines can be made in an aggregated manner, where all machines belonging to same cell can be estimated to have the same MTBF, or in a disaggregated manner, where the MTBF of each single machine can be estimated separately. This aggregation vs. disaggregation depends on the level of model fidelity as explained in Chapter 4. Figure

6.2 depicts dynamic changes of fidelity (in terms of hierarchy) in each cell of the semiconductor die manufacturing fab. These results are obtained from five replications, where each replication lasts for 6 months (180 days) of operation of the fab.

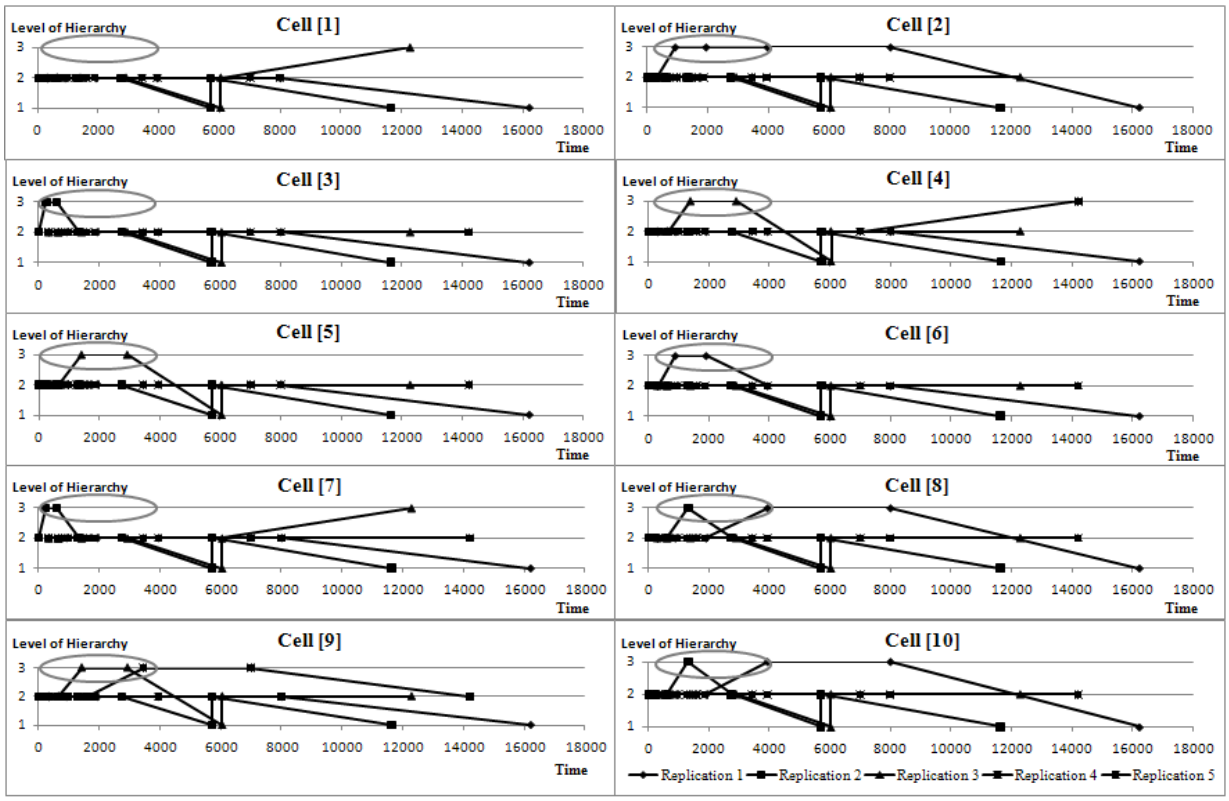


Figure 6.2: Dynamic change of fidelity (level of hierarchy) in different cells of a semiconductor die manufacturing fab model

When an abnormality is detected, the model does not necessarily have to simulate the entire system (all ten cells with a total of 100 machines) in a most detailed manner by increasing the fidelity level of the simulation model over all the cells. Instead, we can simulate a specific portion of the manufacturing environment in detail (via increased fidelity levels) while aggregating the rest (by decreasing the fidelity level) based on the

results of the algorithms. The circles in Figure 6.2 represent this notion, where some cells are simulated at Level 3 in terms of their level hierarchy, while the others are kept in Levels 1 and 2 during the same period of the same replication leading to computational savings. Overall, the dynamic change of fidelity over various cells can be seen from the same figure. For instance, during the first replication, Cell #10's fidelity level has started with a hierarchical level of 2, and around time 4000 increased to 3 (which means algorithms have detected some abnormality at this time in this cell and requested to go one level deeper). After the abnormality is resolved around time 8000, the fidelity level decreased to Levels 2 and 1 subsequently.

Figure 6.3 depicts the dynamic fidelity change of three sample machines in the die fab during execution of the DDDAS-Simulation for one replication which lasts 500 hours in addition to 95 hours of warm-up period. This figure is provided in order to be considered in line with Figure 6.4 which depicts dynamic changes of the computational resource (CR) availability in the grid environment and its usage by the DDDAM-Simulation for these sample machines as well as the cell that they all belong to. As a collaborative supply chain system is of our concern in this study, each fab competes for the available CR that is collected in one pool and managed by the resource manager component of the Alchemi toolkit.

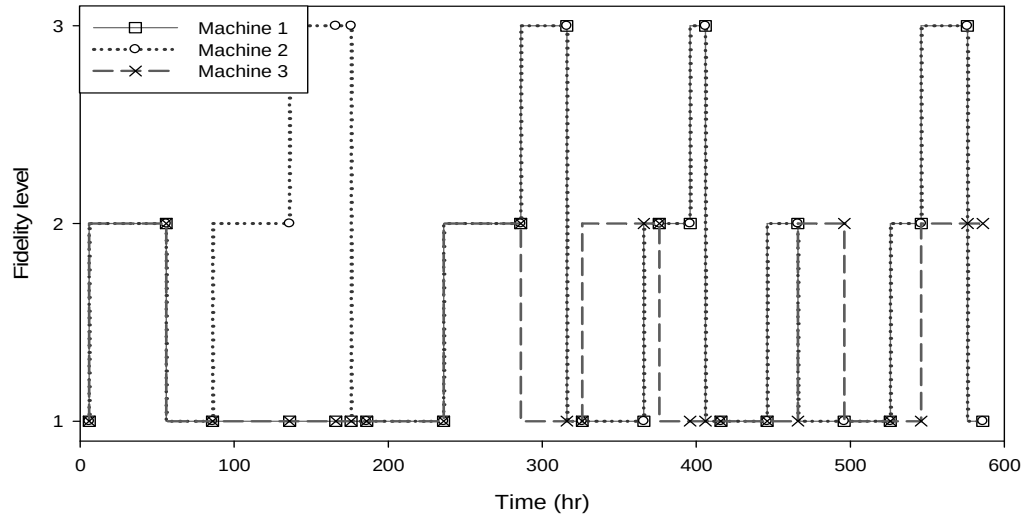


Figure 6.3: Dynamic change of fidelity level on each machine of die fab

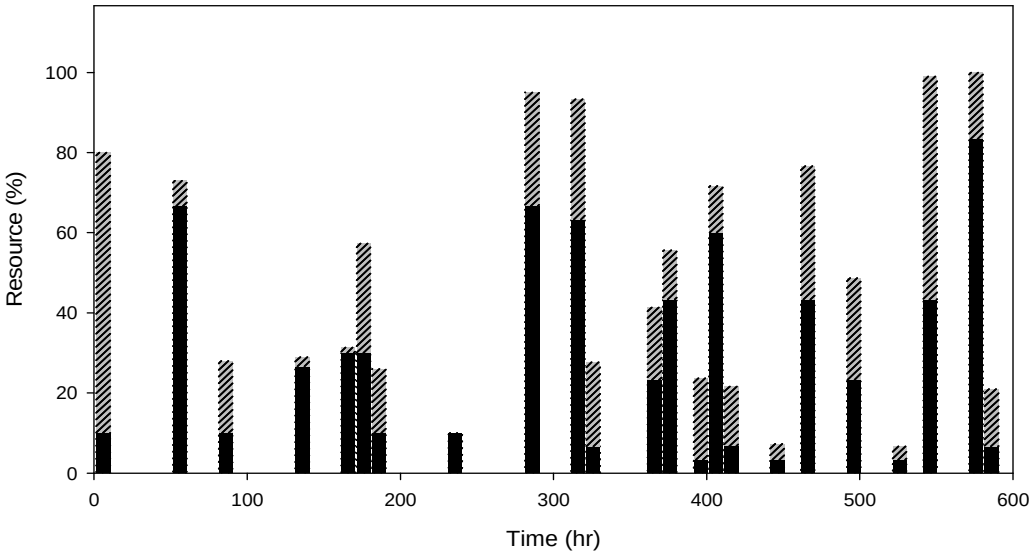


Figure 6.4: Dynamic change of CR availability and its usage in die fab

6.2 Latency in Data Collection

In addition to the computational burden, another challenge faced in models with higher fidelities is latency in data collection. Latency, denoted by δ in this dissertation, is

the time taken for updated information to be collected from the requested portions of the manufacturing environment (simulated real system) and sent to the DDDAM-simulation (or in other words time to update sensor data and schedules). Even though the sampling frequency is purposely decreased in this study (via the Algorithm 2 that is built using particle filtering technique), the latency in data collection increases as the level of fidelity becomes higher (assuming that greater amount of data is needed more frequently when an abnormality occurs). Figure 6.5 shows an exemplary case of this latency (as update time in vertical axis), where the simulation first starts with the most aggregated level of fidelity (Level 1) and increases to Levels 2 and 3 as the update number increases. As the model fidelity increases, the confidence interval of the latency increases since more data is requested from different places of the real system, and it takes the real system to obtain all these data and send it back to the DDDAM-simulation. Here, the latency data is collected using a built-in Matlab function via current system time between where communications are real and computations are virtual as the real system is emulated by a detailed simulation model.

In Figure 6.5, there is a major jump in the update time when the 7^{th} update request arrives to the real system; this is due to the fact that majority of the cells request data from multiple machines more frequently (*i.e.*, hierarchical level of fidelity is 3 for most of the cells) than that of the 8^{th} update. During the 8^{th} update, majority of the cells request data at a relatively low pace (*i.e.*, hierarchical level of fidelity is 3 for most of the cells) when compared to that of the 7^{th} update. Because of this increase in the confidence

interval of the latency, DDDAM-simulation intends to keep its fidelity as aggregated as possible, and increases its fidelity only when it becomes necessary.

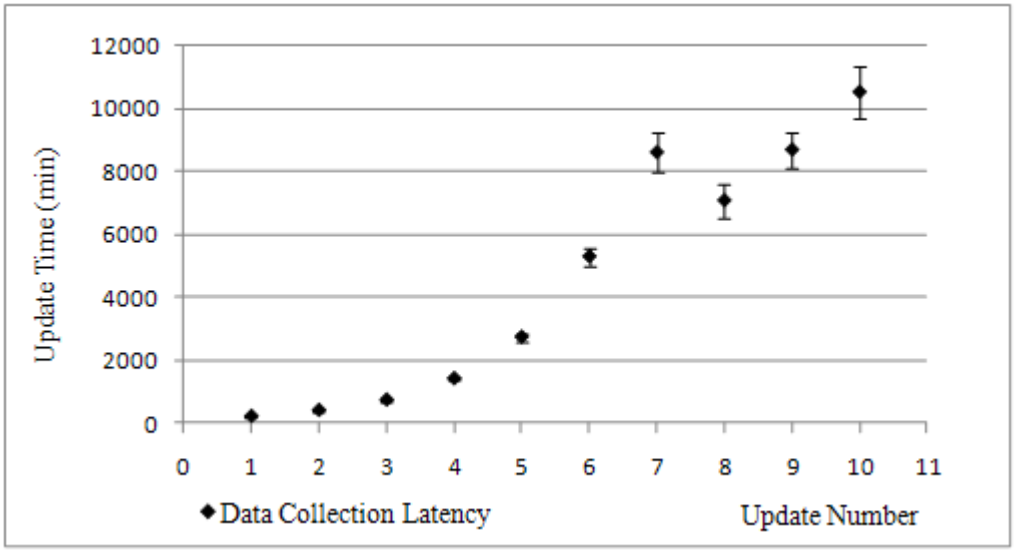


Figure 6.5: Update time (δ) in data collection with varying fidelities

In this dissertation work, another set of experiments have been conducted in order to be able to provide more details with the latency in the data collection process. In this additional set, the real-time DDDAMS model for the die fab has been run in synchronization with a simulation mimicking the real system for five replications each of which is 500 hours length. During these replications, the update time of sensor data and schedules is recorded when the most aggregated fidelity, fidelity being equal to 1 for each cell, is detected. Corresponding observations are summarized in Table 6.1. The confidence interval for the δ of the most aggregated fidelity case is computed by first finding the half width, h whose formula is given in Eq. (6.1). In Eq. (6.1), n is the replication number, $(1-\alpha)$ is the confidence level, s is the sample standard deviation,

and $t_{n-1,1-\alpha/2}$ is the t-value with $n-1$ degrees of freedom. In this case, $n=5$ as we have five replications, α is selected to be 0.05 and s is computed to be 1.6179. Hence, $t_{4,0.975}=2.776$ and $h=2.776\frac{1.6179}{\sqrt{5}}$ =2.009. Overall, mean of the δ observations is $\bar{x}=3.7809$. A 95% confidence interval of δ, time to update sensor data and schedules when the model Fidelity is 1 for each cell, is $\bar{x}\pm h=3.7809\pm 2.009$. In the same way, a 95% confidence interval of δ, time to update sensor data and schedules when the model Fidelity is either 2 or 3 for each cell, is computed. In this case, $\bar{x}$ =22.4781, n =5, α =0.05, s =11.6369 and $t_{4,0.975}$ =2.776. Hence, 95% confidence interval of δ is found to be $\bar{x}\pm h$ =22.4781±14.4468.

Table 6.1: δ, time to update sensor data and schedules (model fidelity is 1 for each cell)

Observation\Replication	Replication 1	Replication 2	Replication 3	Replication 4	Replication 5
Observation 1	2.3185	3.2351	6.0148	5.0102	1.8214
Observation 2	5.1923	2.1968	5.8461	3.7622	3.5588
Observation 3	2.1101	4.1891	6.8690	4.6953	3.4298
Observation 4	----------	----------	1.4731	----------	2.5535
Mean	3.2070	3.2070	5.0508	4.4892	2.8409
Standard Deviation	1.7225	0.9964	2.4267	0.6490	0.8132

$$h=t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}} \qquad \text{Eq. (6.1)}$$

The above comparison of the confidence intervals shows that as fidelity increases, the mean time to update sensor data and schedules noticeably increases. As discussed earlier, this is due to the fact as fidelity increases, the detail and information included in the model as well as their process times increase. As δ increases, the computational

resource usage associated with it increases as well. Our goal in this study is to keep the model as aggregated as possible to obtain the fastest, yet accurate results while saving from computational resource usage. Figure 6.6 depicts an overall plot of δ over dynamic fidelity changes by time (one single replication with 500 hours) for five specific cells of the semiconductor die manufacturing fab.

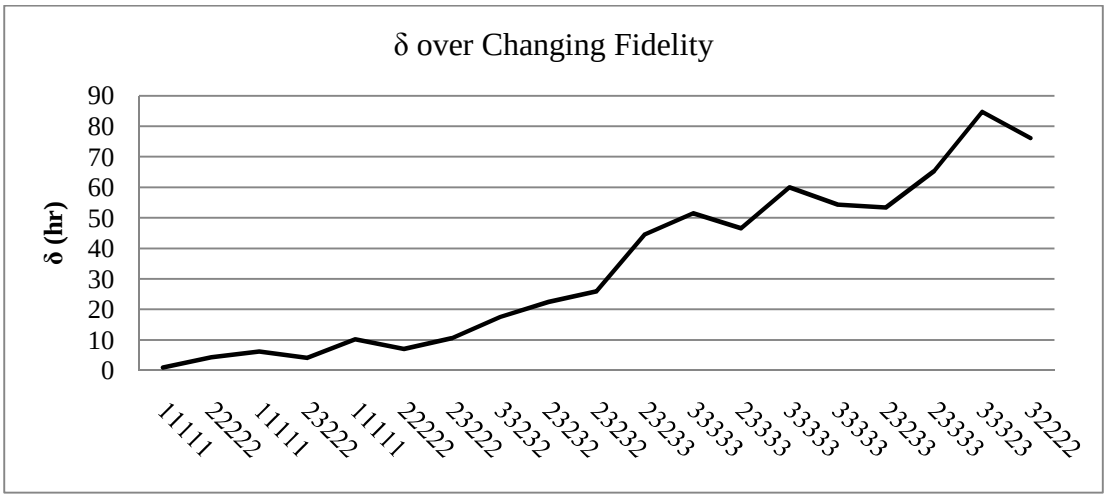


Figure 6.6: δ changes over dynamic fidelities

6.3 Performance of Proposed Fidelity Selection Algorithm with Synthetic Experiments

The goal of this section is to demonstrate the effectiveness of the proposed particle filter algorithm (fidelity selection algorithm) in reflecting the status of the considered supply chain system. First, performance of the developed particle filter algorithm is tested against various nonlinear (and non-Gaussian) systems whose posterior status information capturing their dynamicity is known. This synthetic stage of our

experiment is conducted involving fidelity selection algorithm only before it is combined with the DDDAM-system. Then, the developed fidelity selection algorithm is integrated into the DDDAM simulation, and the entire DDDAM simulation is tested with a real-time simulation mimicking a real system. While the former part of the experiment is discussed in this section, the latter part of the experiment is discussed in Chapter 6.6.

In the synthetic stage of our experiment, we intend to demonstrate effectiveness of the proposed fidelity selection algorithm in estimating the behavior of a system. As mentioned previously, the behavior (collection of statuses) of a system is represented by a posterior distribution. Therefore, the algorithm seeks to determine the posterior distribution based on the measurements that are fed into the algorithm. In this experiment, in order to represent our synthetic system's behavior, we have composed three sets of nonlinear state space equations (the state transition function and the observation function), which are shown in Eqs. (6.2), (6.3), and (6.4), respectively. Here it should be noted that the dimensionality of the vectors x_k in Eqs. (6.2), (6.3) and (6.4) are not necessarily the same. In our experiments, Kong (1994) resampling rule, which is exact optimal in terms of minimizing the variance, is adapted by setting the resampling threshold value to 0.6*N_s (number of particles). While the threshold value can be set to a constant as well as a dynamically changing number (such as ours), by defining a threshold value based on the current number of samples, we enable the filter adapt by sampling only as much as necessary given the sample weights. If a very large number is selected for this value, the resampling process repeats indefinitely, and cannot be ended. In contrast, if the same value is set to a very small number, the particles may cause a

situation called degeneracy where the re-sampling process cannot be initiated since the conditions for initiation cannot be met with a potential of filter leading to less accurate results. In addition, 0.6*N_s is determined as our own threshold value based on the offline analysis that is conducted as part of the experiments. In the following equations, a and b are constants that are arbitrarily selected as 0.005 and 0.5, respectively and the process noises u_k's are generated from the Gaussian function.

$$x_{k+T=}f_x(x_k, u_k) = 2 + \frac{4a}{1+a^2} + \sin(a\pi k) + bx_k + u_k \qquad \text{Eq. (6.2)}$$

$$z_{k+T=}f_z(x_k, v_k) = bx_k + v_k$$

$$x_{k+T=}f_x(x_k, u_k) = \exp(2.5)ak + 2b^2x_k + u_k \qquad \text{Eq. (6.3)}$$

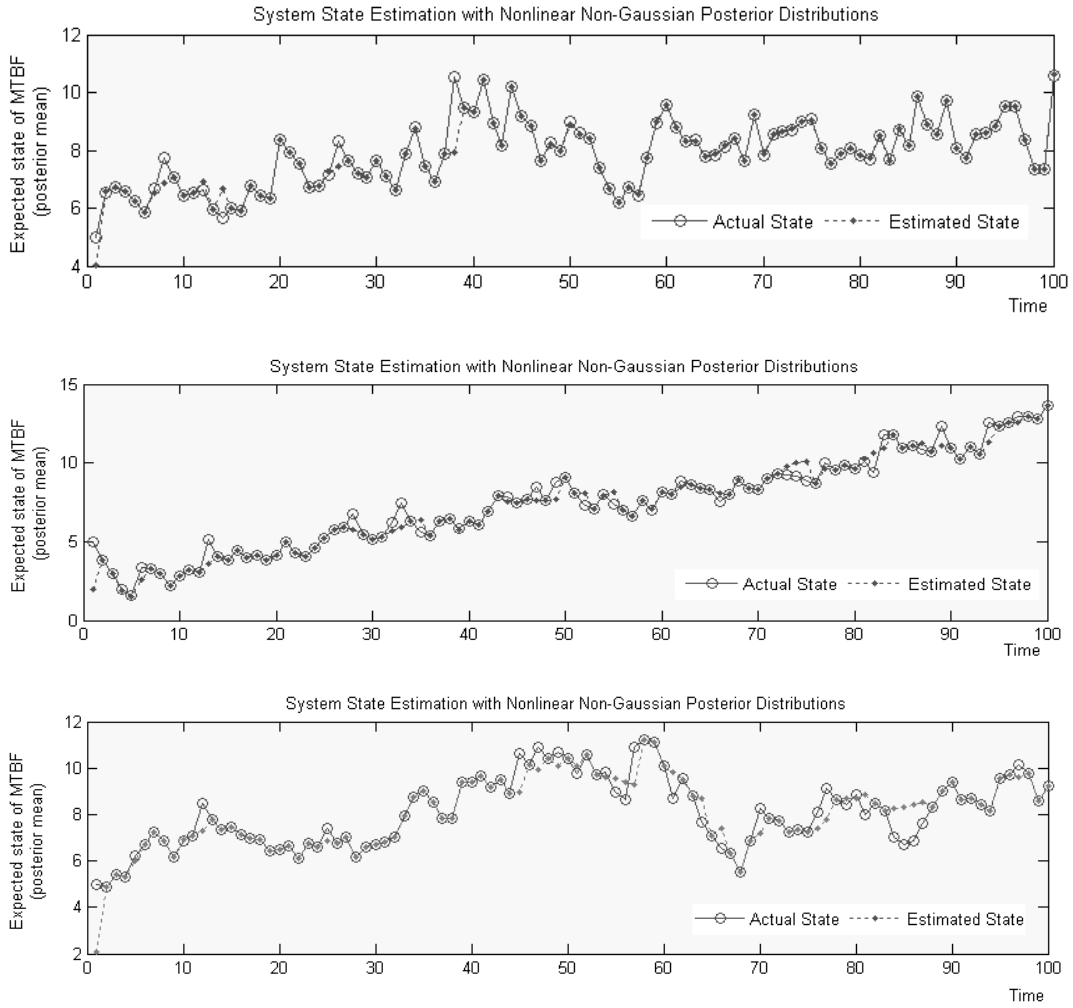
$$z_{k=}f_z(x_k, v_k) = bx_k^2 + 3 + v_k$$

$$x_{k+T=}f_x(x_k, u_k) = \frac{3a^2}{1+a} + 10Beta(3,2)x_k + u_k \qquad \text{Eq. (6.4)}$$

$$z_{k=}f_z(x_k, v_k) = bx_k^2 + 2 + v_k$$

Figures 6.7 (a), (b), and (c) depict the results of state estimation obtained from the fidelity selection algorithm (screen shots of single replication for each case) when using the Eqs. (6.2), (6.3) and (6.4), respectively. Figure 6.7 (d) depicts a discretized version of the Eq. (6.2), where its corresponding discretized states are defined in Figure 4.9. In Figure 6.7 (a-d), the lines with circles represent the true states obtained from the original

functions whereas the line with dots represent the state estimates (posterior mean) generated from the sequential Monte Carlo based fidelity selection algorithm developed in this study. Figure 6.7 reveals that the estimates obtained from the algorithm closely follow the true states of the system. The algorithm generates these close estimates by automatically adapting the sample size and the sampling frequency to the need of the system as shown in Table 4.5.



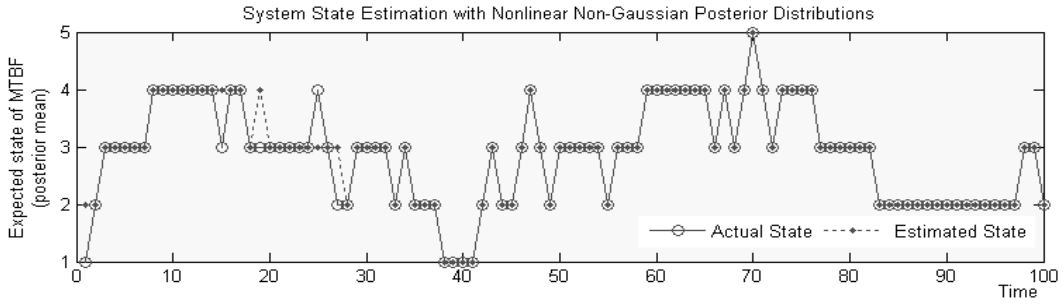


Figure 6.7 (a-d): Estimation results obtained from the sequential Monte Carlo based fidelity selection algorithm for cases in Eqs. (6.2), (6.3), (6.4) and a discretized version of the Eq. (6.2), where its corresponding dicretized states are defined in Figure 4.9, respectively.

In order to test the consistency of the results, the same set of experiments have been repeated for 50 independent replications each lasting 100 time steps (hours in this case). In the experiment for the case with Eq. (6.2), the expected value of root mean square error (RMSE-mean) and its corresponding variance (RMSE-var) were found to be 0.370 and 0.007, respectively. Similarly, the RMSE-mean and RMSE-var were found to be 0.203 and 0.0009 for the case with Eq. (6.3); 0.255 and 0.006 for the case with Eq. (6.4); and 0.336 and 0.016 for the experiment involving the discretized form of the Eq.(6.2), respectively.

An additional set of experiments have been conducted with the same functions (*i.e.*, cases in Eqs. (6.2), (6.3) and (6.4)), where the length of each replication is increased and the number of total replications is reduced compared to that of the previous set of experiments. Here, each run is composed of 10 independent replications, each lasting 1000 time steps; rather than 50 independent replications, each lasting 100 time steps when compared to the previous set. During this set of experiments; for the case with Eq. (6.2), the expected value of root mean square error (RMSE-mean) and its corresponding

variance (RMSE-var) were found to be 0.469 and 0.0806, respectively. Similarly, the RMSE-mean and RMSE-var were found to be 0.389 and 0.0083 for the case with Eq. (6.3); 0.361 and 0.0135 for the case with Eq. (6.4); and 0.458 and 0.1598 for the experiment involving the discretized form of the Eq. (6.2), respectively. As can be realized from these results where the maximum RMSE of state estimation and the average RMSE were found to be around 0.469 and 0.1598, respectively (when replicated 10 times), sequential Monte Carlo-based fidelity selection leads to accurate results while saving computational resources and time.

6.4 Performance of Proposed Resampling Rules

In Chapter 4.4.2.3, three different resampling rules have been discussed from three different statistical perspectives, where two new resampling rules (variance-based and bias-based relative sampling efficiency rules) were proposed, and one was revisited (half-width based sampling efficiency rule). Theoretical derivations have been also provided for the proposed rules as well as the revisited one. In this section, these three algorithms are compared against each other as well as against a resampling rule proposed by Kong et al., (1994) which we refer as “Kong rule” in terms of their resampling qualities (mean and variance of their root mean square errors) and computational effort (computation time). For this benchmarking, large scale supply chain simulations which have been built to represent a semiconductor manufacturing supply chain (with three echelons including wafer manufacturing fab, semiconductor manufacturing fab, and assembly and packaging fab) have been used. The status parameters sought in these

simulations are “mean time between failures (MTBF)” for each machine in the considered shop floor (member) of this supply chain, where each shop floor includes at least hundred machines. These machines perform various operations such as diffusion, etch, photo, metals, and probe processes. As part of the DDDAM-framework, by knowing the current system status (mean time between failures in this case), the final goal of the analysis is then to evaluate various preventive maintenance and part routing scheduling policies under various scenarios. All of the results shown in this section are obtained out of 30 replications of the same simulation with different seed values.

Figures 6.8 and 6.9 show the results obtained for the resampling qualities in terms of mean and variance root mean square errors (RMSE) of their estimates. The proposed bias-based relative efficiency sampling rule is shown to be the least effective among all of the compared algorithms since the mean RMSE values recorded for this resampling rule is greater than that of all the others. Half-width based sampling efficiency rule, while performing slightly better than the proposed bias-based relative sampling efficiency rule, is outperformed by both the proposed variance-based relative sampling efficiency rule as well as the Kong rule. Even though, mean RMSE decreases as the sample size increases, regardless of the used resampling rule, the proposed variance-based relative sampling efficiency rule produces the smallest RMSE on average. Regarding variance of the recorded RMSE values, Kong rule produces the largest variance in its results up to a sample size of 1000, and after this sample size, recorded variances of RMSE values decrease dramatically, and even becomes smaller than that of both half-width and bias-based resampling rules. The proposed variance-based resampling rule on the other hand,

depicts the smallest amounts of variances in RMSE in most of the cases. However, when the sample size is 200, half-width based resampling rule generates slightly lesser amount of variance in RMSE than that of variance-based resampling rule; and when the sample size is 1400, proposed bias-based resampling rule produces slightly lesser variance in RMSE when compared to the results of the same.

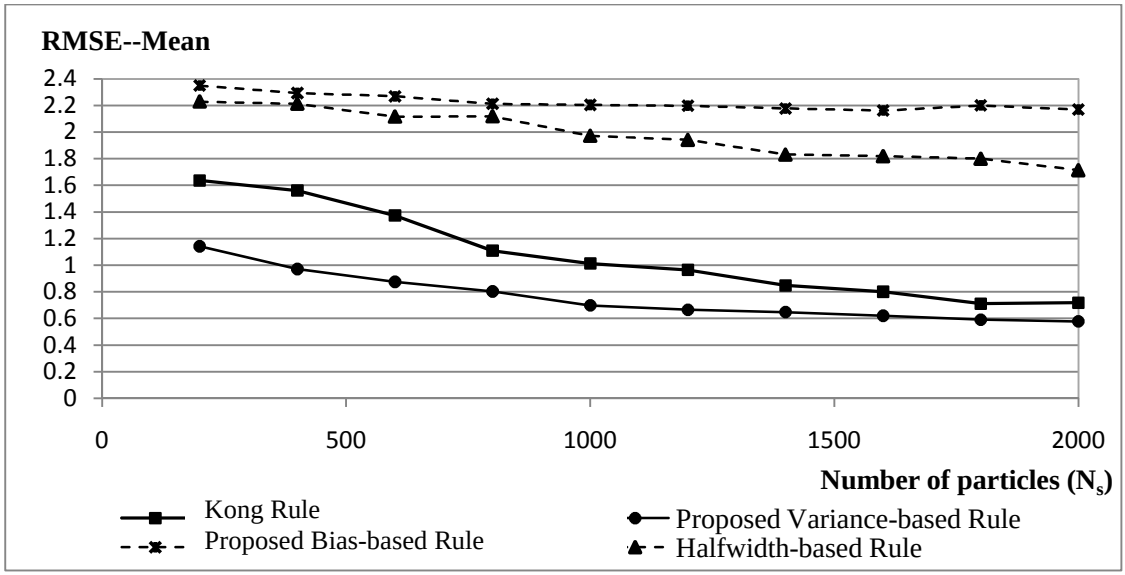


Figure 6.8: Mean of RMSE values for state estimates as a function of the number of particles for the supply chain simulation model

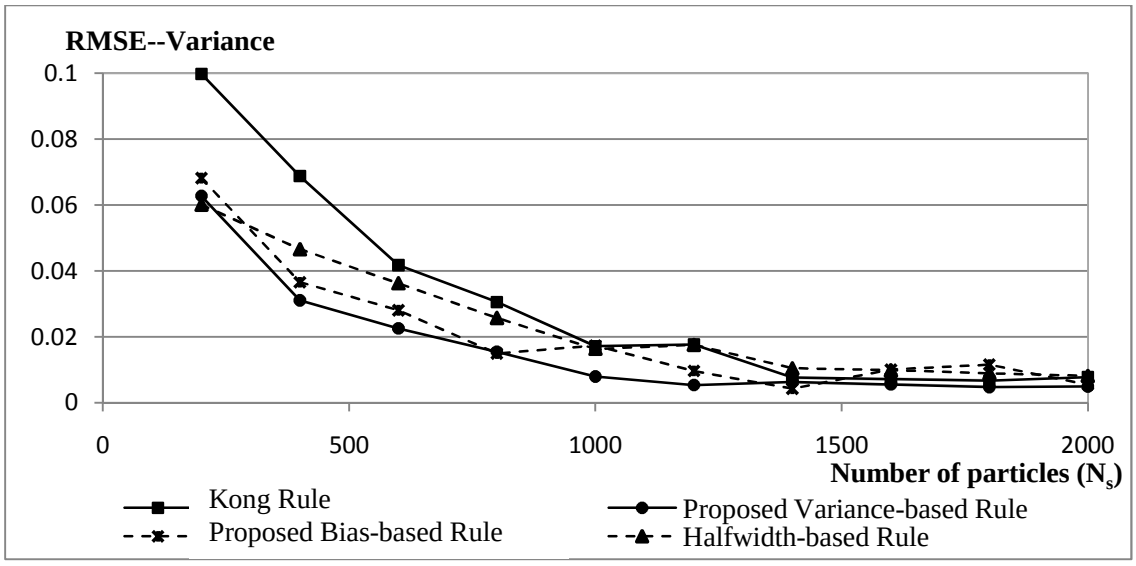


Figure 6.9: Variance of RMSE values for state estimates as a function of the number of particles for the supply chain simulation model

Based on the above results, the proposed bias-based relative sampling efficiency rule is not recommended for the cases where reasonable accuracy is necessary with a particle set size of less than 1000 due to computational resource unavailability. When the particle set size can be ranged between 1000 and 1500, the proposed bias based relative sampling efficiency rule can be selected over half-width based resampling rule as the former one results in smaller variances against very close mean values. On the other hand, the use of Kong rule might be risky as well when the sample set size is limited with 1000 particles, since the variance of the estimates are the highest among all the methods compared in this study. Overall, based on the estimation results plotted in this work, the proposed variance-based relative sampling rule is recommended to be employed in particle filtering algorithms, when the sample set size is upper bounded with 1000 particles, and the best possible accuracies are targeted with limited computational

resources. When this limitation is relaxed up to 2000 particles, the difference between the Kong rule and the proposed variance-based relative sampling rule becomes minimum, hence both algorithms are equally recommended. It should be noted here that, the results obtained in this work is based on the simulation considered in this work, and the performance of the proposed variance-based relative sampling efficiency rule may involve even more significant computational savings while preserving accuracy in the system estimates when it is applied to the real-world supply chain control situation, that is greater in both scale and complexity. In addition, while the performance of the bias-based relative sampling rule is shown to be weak when compared to other methods, the performance of the methods combining these proposed rules may still outperform the individual rules. Future venues of this study will focus on the performance of these combined methods. Also, the performance of the proposed resampling rules should be analyzed focusing on the context of the system that is being analyzed. For instance, the variance-based relative sampling rule can follow up the system status better with less number of particles when the system originally has high variance and low bias. Similarly, bias-based relative sampling rule may perform better when the system originally has low variance and high bias.

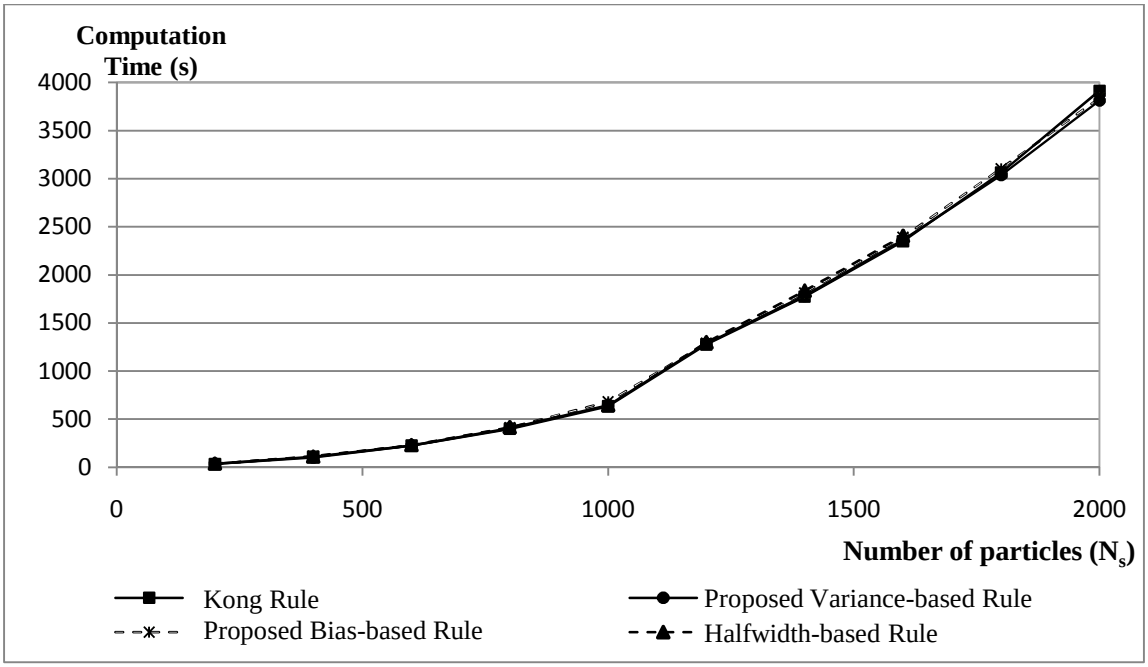


Figure 6.10: Computational effort required for different resampling rules as a function of the number of particles

As part of the experiment in this work, the computational effort required for the algorithms are investigated by measuring the time required to perform resampling of weight sequences. Figure 6.10 shows the measured times each resampling algorithm requires. As can be seen from the same figure, the computational effort necessary for the implementation of these algorithms are directly related to the particle set size and not related to the selection of the resampling method since all of the rules discussed in this study have computational complexities of $\mathcal{O}(N_s^2)$, where N_s is the particle set size as mentioned earlier.

6.5 Performance of Prediction and Task Generation Algorithm

The goal of this section is to describe the analysis procedure developed for Algorithm 4, the prediction and task generation algorithm. Issues such as correlation of data and obtaining real execution time of preventive maintenance with the facilities are specified in Chapter 8. In this section, the validity of the multiple linear regressions models for forecasting "maintenance scheduling times" is discussed for each machine that exists in our current exemplary system (see Chapter 3). The data to the linear regressions model is fed by the fast mode portion of the DDDAM-Simulations which are initiated using the assigned fidelity information by Algorithm 3, the task assignment algorithm.

The validity of the multiple linear model can be checked using various ways including the usage of 1) confidence interval for each of the parameters, $\widehat{\beta}_j$, 2) Pearson's coefficient of regression, 3) P-value and 4) F-statistics. In the first line of attack, which is not employed in the current regression model developed, if the confidence interval of any parameter, $\widehat{\beta}_j$, includes 0, then the parameter can be removed from the model. After that, new regression analysis are performed excluding that parameter and this loop is repeated until there is no parameter left to be removed. The remaining ways of checking model validity in terms of statistical analysis of data fitting, which are also utilized in this study, are explained in detail below.

- Residuals represent the difference between the observations and the model's predictions, are denoted by $\hat{\varepsilon}$ and computed as in Eq. (6.5).

$$\hat{\varepsilon} = Y - X\hat{\beta} \qquad \text{Eq. (6.5)}$$

- Standard deviation for the model is denoted by $\hat{\sigma}$ and computed as in Eq. (6.6) where the variance in the errors is Chi-square distributed with $\frac{(n-p-1)\hat{\sigma}^2}{\sigma^2} \approx \chi^2_{n-p-1}$.

$$\hat{\sigma} = \sqrt{\frac{\hat{\vec{\varepsilon}}^T\hat{\vec{\varepsilon}}}{n-p-1}} \qquad \text{Eq. (6.6)}$$

- Interval estimates for parameters are computed as in Eq. (6.7). In Eq. (6.7), where the 100(1 − α) % confidence interval for the parameter $\hat{\beta}_i$ is computed, t follows the Student's t -distribution with $n-p-1$degrees of freedom and $(X^TX)^{-1}_{ii}$denotes the value located in the i^{th} row and i^{th} column of the matrix.

$$\hat{\beta}_i \pm t_{\alpha/2,n-p-1}\hat{\sigma}\sqrt{(X^TX)^{-1}_{ii}} \qquad \text{Eq. (6.7)}$$

- Sum of squared residuals is denoted by SSR and computed as in Eq. (6.8), where $\bar{y} = \frac{1}{n}\sum y_i$ and $\vec{u}$ is an n by 1 unit vector (*i.e.* each element is 1). Note that the terms y^Tu and u^Ty are both equivalent to $\sum y_i$, and so the term $\frac{1}{n}(\vec{y}^T\vec{u}\vec{u}^T\vec{y})$ is equivalent to $\frac{1}{n}\sum {y_i}^2$. Error (or explained) sum of squares is denoted by ESS and computed as in Eq. (6.9). Total sum of squares is denoted by TSS and computed as in Eq. (6.10).

$$SSR = \sum(\hat{y}_{i-}\bar{y})^2 = \hat{\beta}^TX^T\vec{y} - \frac{1}{n}(\vec{y}^T\vec{u}\vec{u}^T\vec{y}) \qquad \text{Eq. (6.8)}$$

$$ESS = \sum(y_{i-}\hat{y})^2 = \vec{y}^T\vec{u} - \beta^TX^T\vec{y} \qquad \text{Eq. (6.9)}$$

$$TSS = SSR + ESS = \sum(y_{i-}\bar{y})^2 = \vec{y}^T\vec{u} - \frac{1}{n}(\vec{y}^T\vec{u}\vec{u}^T\vec{y}) \qquad \text{Eq. (6.10)}$$

- Pearson's coefficient of regression is denoted by R^2 and is computed as in Eq. (6.11). It is a measure of strength of regression. The higher the value, the better the regression is. This coefficient gives what fraction of the observed behavior can be explained by the given variables.

$$R^2 = \frac{SSR}{TSS} = 1 - \frac{ESS}{TSS} \qquad \text{Eq. (6.11)}$$

- Variance inflation factor represent the increase in variance caused by correlation between the explanatory variables and is denoted by VIF. The VIF for the j^{th} explanatory variable is computed as in Eq. (6.12). If one or more variables in a regression have large VIF's, the regression is said to be collinear. Colinearity is caused by one or more variables being almost linear combinations of the others and it results in imprecise estimation of regression coefficients.

$$VIF = \frac{1}{1-R_j^2} = \frac{1}{\frac{RSS_j}{TSS_j}} = \frac{TSS_j}{RSS_j} \qquad \text{Eq. (6.12)}$$

- F-Statistics: is a measure of goodness-of-fit and computed as in Eq. (6.13) where p is the degree of freedom of the model and $n - p - 1$ is the degree of freedom of error in the model.

$$F = \frac{MSM(mean\ square\ model)}{MSE\ (mean\ square\ error)} = \frac{\frac{\sum(\hat{y}_i - \bar{y})^2}{p}}{\frac{\sum(y_i - \hat{y})^2}{n-p-1}} \qquad \text{Eq. (6.13)}$$

Analysis of variance (ANOVA) calculations including the statistical fitting parameters explained above of the regression models are displayed in an analysis of

variance table (Table 6.2), for each machine in the system under consideration. Because the purpose of the proposed methodology is dynamic task generation in real-time environment, ANOVA calculations of the regression models shown below for each machine are obtained from one cycle of a single replication simulation. Corresponding p -values of the constructed regression models are remarkably small for all of the models, proving that the models are valid for the given system configuration. In addition, VIF value of each explanatory variable is considerable low representing that these variables are not collinear.

Table 6.2: Evaluation of MTBF prediction for a machine

Regression Analysis of Y (MTBF)

Regression equation: Y = 389.22 + 0.17 X1 + 0.29 X2 -0.04 X3 + 0.26 X4 + 0.01 X5 + 0.25 X6 + 4.06 X7 -0.04 X8 + 0.29 X9

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F	P-value
Regression	19548.14763	9	2172.016404	4.7027	0.0000
Error	41568.36495	90	461.8707217		
Total	61116.51258	99			

S	21.4912	Determinant	0.123366081
R-sq	71.99%	DW	1.31
R-Sq(adj)	71.99%		

Parameter Estimates

Predictor	Coef Est	Std Error	t value	P-value	VIFs
Constant	389.2241	285.1587	1.3649	0.1757	
X1	0.1652	0.1318	1.2531	0.2134	1.1304
X2	0.2932	0.1710	1.7151	0.0898	2.4508
X3	-0.0401	0.0285	-1.4048	0.1635	1.8935
X4	0.2612	0.0933	2.7996	0.0063	1.4593
X5	0.0052	0.0282	0.1843	0.8542	1.6611
X6	0.2482	0.2929	0.8477	0.3989	1.4457
X7	4.0640	1.1342	3.5831	0.0006	1.3613
X8	-0.0361	0.0432	-0.8344	0.4063	1.8176
X9	0.2852	0.1293	2.2059	0.0299	1.3053

100 observations were used in the analysis.

Computational time: 0.5 seconds.

6.6 Overall System Performance based on DDDAMS-framework

The goal of this section is to demonstrate the overall system performance of the proposed DDDAMS-framework. We divide the obtained results to two, where the first results are obtained from the DDDAMS system employing a Bayesian Belief Network (BBN) for the fidelity selection algorithm, and the second results are obtained from the DDDAMS system employing particle filtering for the fidelity selection algorithm. Both results are explained in terms of machine utilization and cycle time in the considered manufacturing supply chain system.

6.6.1 System Performance based on DDDAMS with BBN for Fidelity Selection

Figures 6.11 through 6.15 show the comparison of the resultant utilizations of each machine within the considered semiconductor die fab when simulated with the proposed DDDAMS system employing a BBN for fidelity selection (when Algorithms 1 and 2 are built separately during the initial stage of this work) versus when simulated in the most detailed manner. For each case, five replications have been performed to obtain the results. The obtained utilization values for each machine is quite close, and in some cases the machine utilizations are even better when PM schedules and part routing schedules are made by the DDDAMS-simulation. This may be due to the fact that the DDDAMS-simulation aims to postpone the PM maintenance as late as possible until the machine breaks down. One slight difference between the utilizations obtained as a result

of DDDAMS-simulation versus the detailed simulation is that, the minimum utilization point may be slightly less in DDDAMS case than that of the detailed simulation case. For instance, for Machine 2, the minimum utilization recorded is around 0.1 for the DDDAMS case whereas the same is recorded around 0.3 for the detailed simulation case. Hence, we can conclude that by running the detailed simulation as controller of the shop floor with having large amount of computational resources dedicated, the facilities may enable machines to have slightly higher minimum utilization values. However, in case of using the DDDAMS-simulation instead, they enable very similar performance with small sacrifice in the minimum utilization point with considerable less computational efforts.

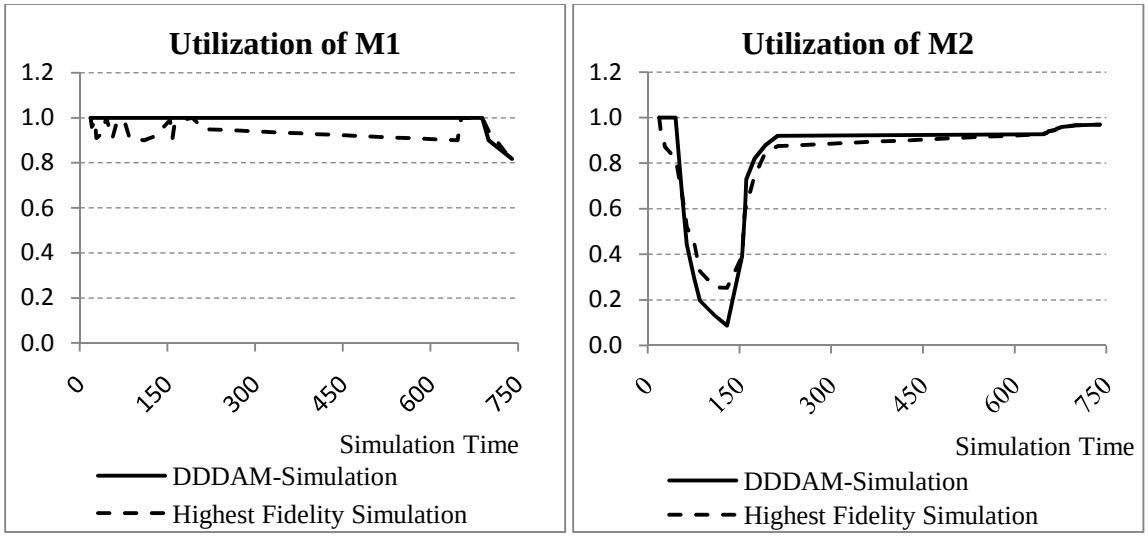


Figure 6.11: Plot of utilization of machines 1 and 2 in proposed DDDAM-Simulation vs. in high fidelity simulation

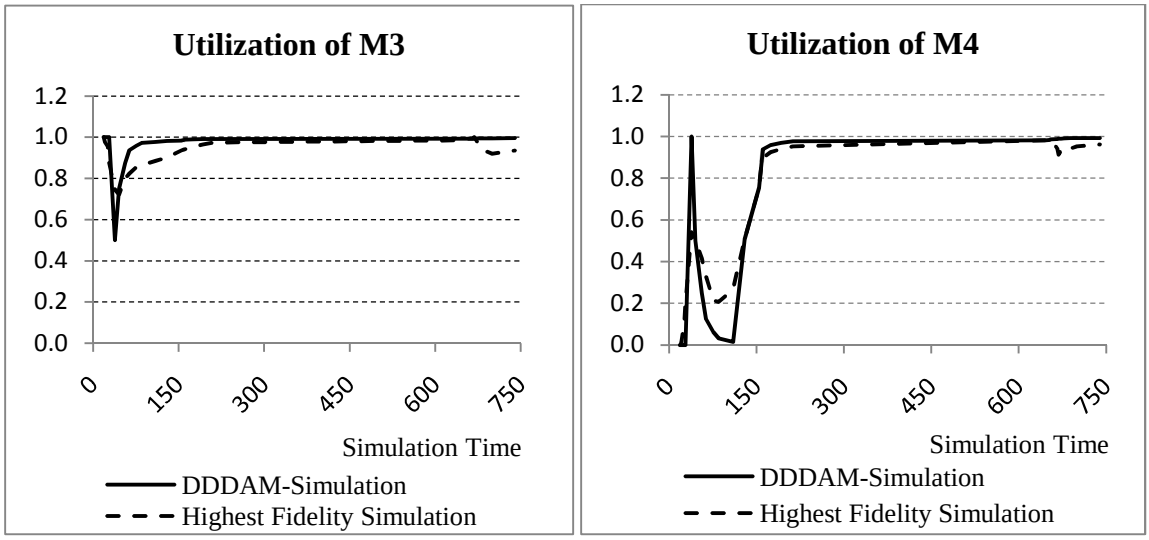


Figure 6.12: Plot of utilization of machines 3 and 4 in proposed DDDAM-Simulation vs. in high fidelity simulation

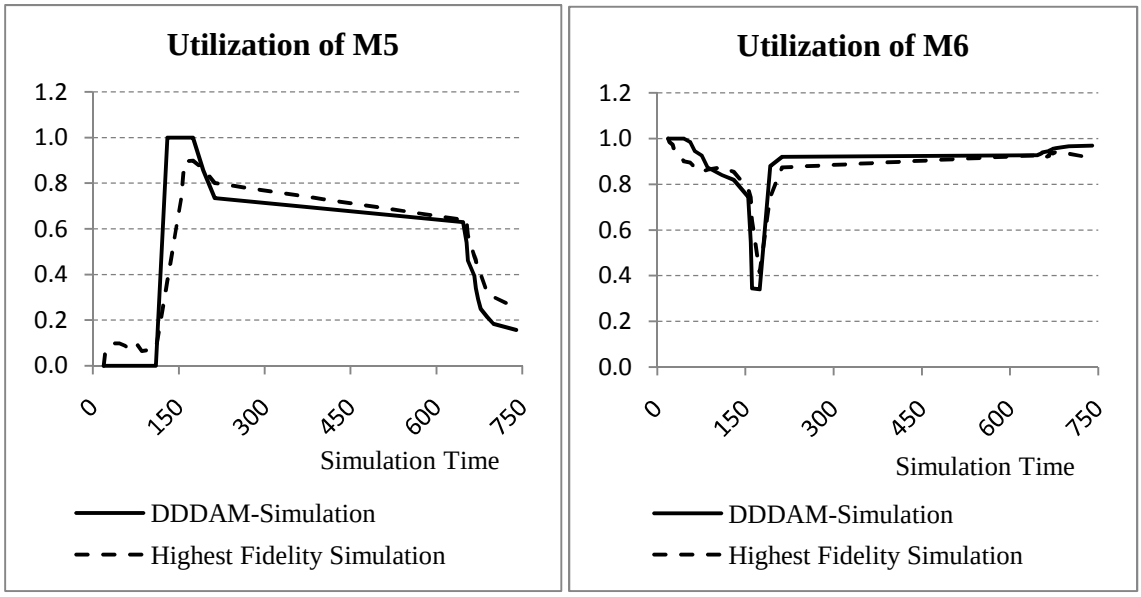


Figure 6.13: Plot of utilization of machines 5 and 6 in proposed DDDAM-Simulation vs. in high fidelity simulation

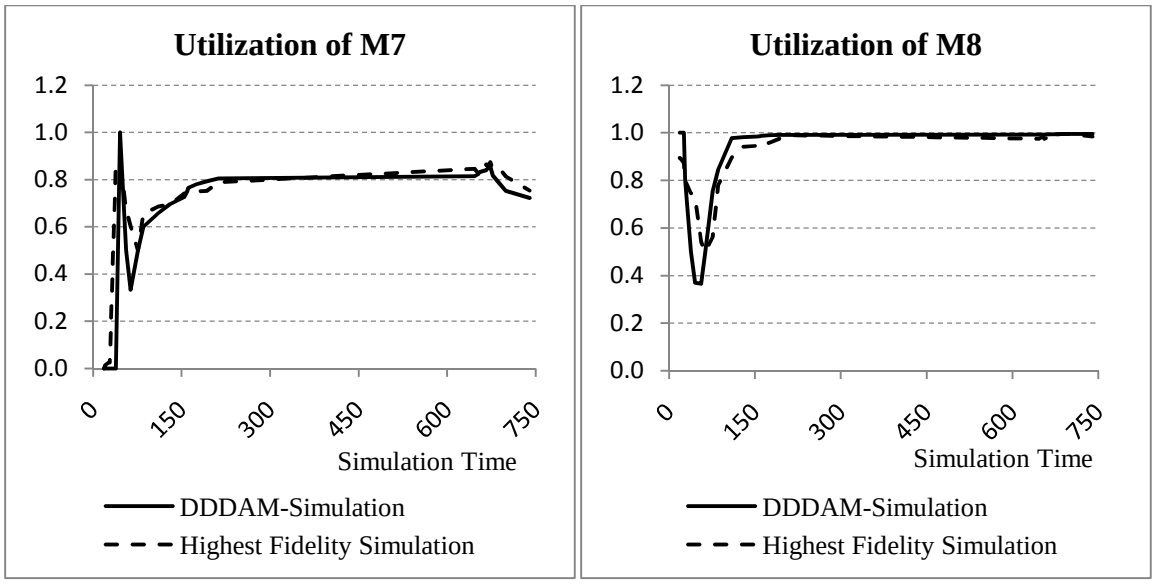


Figure 6.14: Plot of utilization of machines 7 and 8 in proposed DDDAM-Simulation vs. in high fidelity simulation

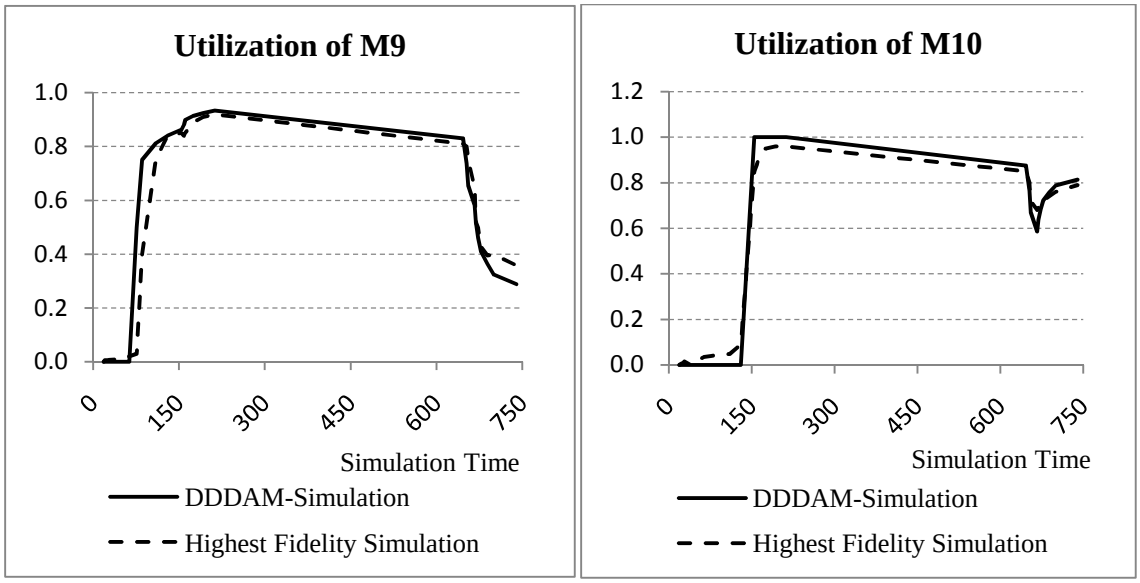


Figure 6.15: Plot of utilization of machines 9 and 10 in proposed DDDAM-Simulation vs. in high fidelity simulation

Another set of experiments have been conducted to test the performance of the DDDAMS in terms of the cycle time. The DDDAMS models for the supply chain

members have been executed in sync with the real system (implemented in real-time simulation) for five truncated replications, each of which is 500 hours long in addition to a warm up period of 95 hours. The models are initially at an empty-and-idle state, where all resources are idle and no entity exists in the system. The transient phase is captured in the determined warm up period and is removed from each replication when the statistical results are drawn. During these replications, the mean cycle time of production is recorded for each of the supply chain echelon. Table 6.3 depicts performance statistics for the proposed DDDAMS system compared against those of a high fidelity simulation (involving high computational power but high accuracy) and an aggregated simulation (involving low computational power but low accuracy), respectively. During the experiment, care has been taken to use same seed values for random number generation in real system. Table 6.3 shows 95% confidence intervals for mean cycle times of production within the mentioned simulations. Over the entire supply chain, the mean cycle time was reduced by about 30% due to the usage of PM schedules generated by the proposed DDDAMS system compared to that obtained from the aggregated model with moderate CR usage. Furthermore, the results (PM schedules) obtained from the proposed DDDAMS system are very close (9.5% difference) to those obtained from the detailed model with high CR usage. Cycle times are measured between start of production in the wafer manufacturing fab (echelon 1) and shipment from the assembly and packaging fab (echelon 3). This reduction in cycle time translates to significant monetary savings.

Table 6.3: Mean cycle time (CT) comparison of the proposed DDDAMS system

Echelon	Echelon 1	Echelon 2	Echelon 3	Average
DDDAS-Simulation w. Dynamic Fidelity	67.000 ± 5.193	36.439 ± 3.032	43.494 ± 2.487	48.978 ± 6.746
Highest Fidelity Simulation	64.000 ± 4.301	33.710 ± 0.937	37.568 ± 1.485	45.093 ± 6.471
Lowest Fidelity Simulation	69.400 ± 3.354	69.400 ± 3.354	69.400 ± 3.354	69.400 ± 3.105
% Increase in CT wrt. Highest Fidelity Simulation	4.688 ± 3.468	8.095 ± 5.207	15.773 ± 4.546	**9.519 ± 5.939**
% Decrease in CT wrt. Lowest Fidelity Simulation	3.458 ± 2.625	47.494 ± 3.927	37.328 ± 5.080	**29.427 ± 6.681**

6.6.2 System Performance based on DDDAMS with Particle Filtering for Fidelity Selection

In this section, the fidelity selection algorithm proposed during the second stage of this dissertation using particle filtering technique, which has been successfully validated against various nonlinear (and non-Gaussian) systems whose posterior status information is known (see Chapter 4.4.2), is plugged into the DDDAM-system in order to determine the required level of simulation model fidelity (when Algorithms 1 and 2 are combined via particle filtering technique during the second stage of this work). The performance results obtained from the DDDAM-simulation involving the proposed fidelity selection algorithm for a real system are compared with the results obtained from the most aggregated simulation of the same system (where the model fidelity--level of decision hierarchy is set to Level 1) and the most detailed simulation of the same system (where the model fidelity--level of decision hierarchy is set to Level 3) (see Figures 6.16 and 6.17). All simulations were run 30 replications, where each replication lasted for six months (180 days) of operation. Figure 6.16 depicts an average waiting time of

production batches in different cells of the considered semiconductor die manufacturing fab, where these batches may belong to any of the six product families. As shown in Figure 6.16, the average waiting time obtained from the DDDAM-simulation with the fidelity selection algorithm is less than that of the aggregated simulation for all the cells. Decrease in the average waiting time is more than 20 percent for all the cells except Cell #6, which may be the bottleneck of the production. Furthermore, the overall average waiting time (average over all cells) obtained from the DDDAM-simulation with the fidelity selection algorithm is slightly higher than that of the detailed simulation (with an increase around 7.5 percent). As can be seen from Figure 6.17, in some cases (Cell #3, 8 and 9), the proposed DDDAM-simulation resulted in reduction in average waiting time compared to the results of the detailed simulation. This is believed due to the effect of the changing levels of the data retrieval frequency based on the need of each specific cell.

Similarly, Figure 6.16 reveals that the average utilization of machines obtained from the DDDAM-simulation with the fidelity selection algorithm is higher than that of the aggregated simulation for all the cells. Increase in the average utilization of machines is more than 15 percent for all the cells except again Cell # 6. This increase in the average utilization may go up to around 50 percent in some cells (*i.e.*, Cell # 8). Furthermore, the average utilization of machines obtained from the DDDAM-simulation with the fidelity selection algorithm is slightly less than that of the detailed simulation for all the cells. Average decrease in the average utilization of machines is less than 8 percent over all cells. Contrary to what is expected though, in some specific cells (Cell #3, and 8), the proposed the DDDAM-simulation with the fidelity selection algorithm

resulted in slightly higher levels of utilization which may be due to the dynamic change in the levels of data retrieval frequency.

Based on the results summarized in Figures 6.16 and 6.17, it can be concluded that the DDDAM-simulation with the proposed fidelity selection algorithm is able to represent the actual manufacturing environment accurately while saving from computations when compared to that of detailed simulations. Dynamic routing plans and preventive maintenance schedules based on this DDDAM-simulation enables the real system operate more efficiently with decreased waiting time and better resource utilization.

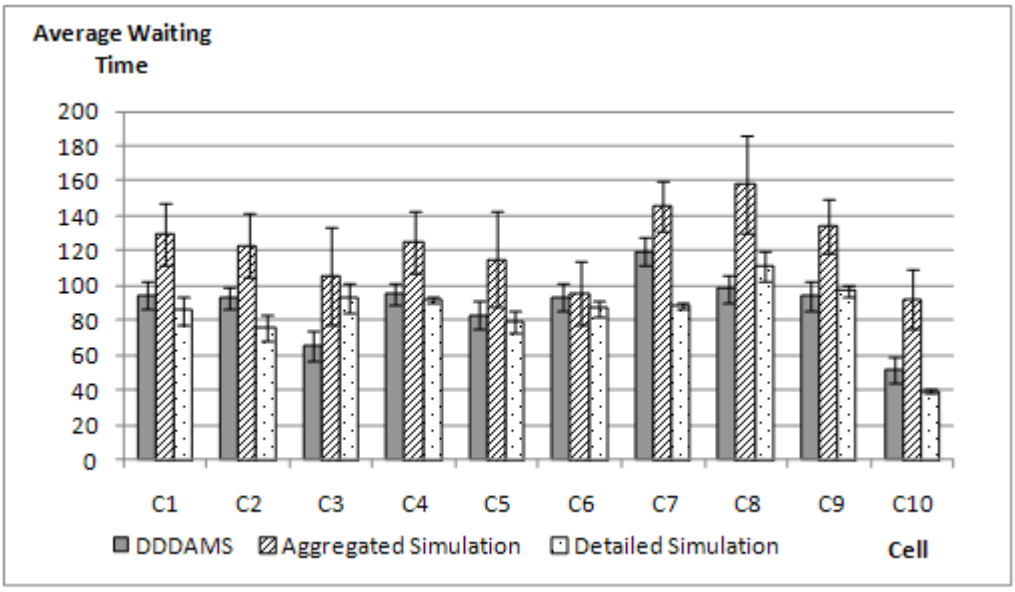


Figure 6.16: Performance of the proposed algorithm for average waiting time

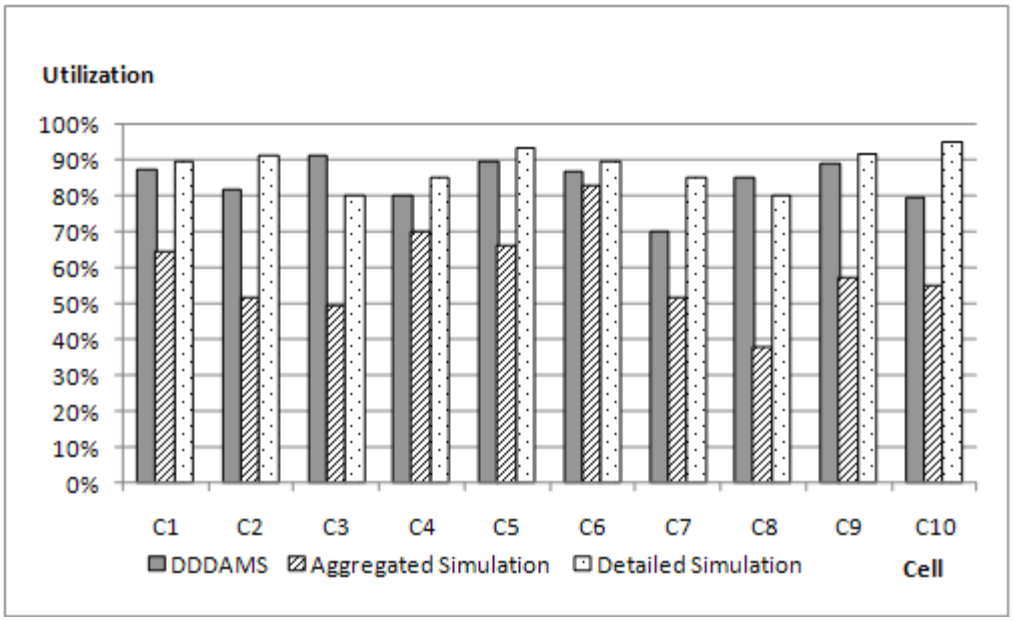


Figure 6.17: Performance of the proposed algorithm for utilization

6.7 Concluding Remarks

In this chapter, we have successfully demonstrated the benefits of the DDDAMS-framework developed in this dissertation based on our considered semiconductor die manufacturing supply chain system. We have started the chapter by demonstrating the dynamic fidelity change in the DDDAM-Simulations and discussing the latency associated with the data collection as a result of this change in fidelity. Then, we have demonstrated the performance results obtained for the proposed fidelity selection algorithm as well as the developed resampling rules using synthetic experiments before it is being plugged into the DDDAMS-framework. Next, we have demonstrated the overall system performance based on DDDAMS-framework with the aforementioned fidelity selection algorithms (both fidelity selection via BBN and fidelity selection via particle filtering).

In the next chapter, we discuss the extended DDDAMS framework involving simulation partitioning methodology, which is developed to save from simulation running times of fast mode simulations (needed in Algorithm 4 of the proposed DDDAMS system).

CHAPTER 7

EXTENDED DDDAMS FRAMEWORK: SIMULATION PARTITIONING

Simulating large-scale systems in fast mode usually entails exhaustive computational powers and lengthy execution times. This situation holds true for the fast mode simulations which we run as part of Algorithm 4 (prediction and task generation algorithm) of the DDDAMS framework when we want to test the candidate preventive maintenance and part routing schedules before selecting the best possible one. In this chapter, we propose a simulation partitioning methodology (Celik et al., 2010) as a major extension of our DDDAMS framework, which is designed to be used during the execution of fast model simulation models associated with Algorithm 4. The validity of this methodology has been tested via a separate manufacturing simulation apart from the original DDDAMS simulations and future venues of this work involves testing the validity and practicality of the proposed method using DDDAMS simulations in an online manner.

A more specific goal of this method is to reduce execution time of large-scale simulations without sacrificing their accuracy by partitioning a monolithic model into multiple pieces automatically and executing them in a distributed computing environment. While this partitioning allows us to distribute required computational power to multiple computers, it creates a new challenge of synchronizing the partitioned models. In this work, a partitioning methodology based on a modified Prim's algorithm is proposed to minimize the overall simulation execution time considering 1) internal

computation in each of the partitioned models and 2) time synchronization between them. In addition, we seek to find the most advantageous number of partitioned models from the monolithic model by evaluating the tradeoff between reduced computations vs. increased time synchronization requirements. In this work, epoch based synchronization is employed to synchronize logical times of the partitioned simulations, where an appropriate time interval is determined based on the off-line simulation analyses. The Achemi computational grid framework is specifically employed for execution of the simulations partitioned by the proposed methodology to facilitate the transition from offline experiments to the DDDAMS framework. The experimental results reveal that the proposed approach reduces simulation execution time significantly while maintaining the accuracy as compared with the monolithic simulation execution approach.

7.1 Simulation Partitioning Problem

Modern society depends upon many interacting large-scale dynamic systems, such as manufacturing, supply chain, defense, transportation, homeland security, and healthcare (Wysk et al., 2004). Although some of their constituent systems are amenable to mathematical and analytical analyses, the ensembles are not, because of scale, randomness, and complex dynamic response. As discussed earlier, among various simulation techniques, discrete-event simulation has become more widely used because it can take randomness into account, address aggregate as well as very detailed models, and enable a reusable experimentation set-up for analysis of such complex systems which is not possible or economical to be held in a real setting. While employing discrete-event

simulations during the design stage of complex systems has become a common practice, high execution time and computational power usage become challenges when they are used to support short-term decisions (*e.g.* operations or maintenance scheduling).

An obvious, alternative approach is to execute a gigantic simulation in multiple computers in a distributed computing environment. However, it creates three new challenges. First, in order to execute a large scale simulation in a distributed manner, the component simulation models must be built in a divided manner during the development stage; or, if a large scale simulation already exists, it should be partitioned into smaller pieces (focus of this research). Here, it is quite challenging to devise a method to partition a monolithic model into multiple pieces to reduce its execution time without sacrificing their accuracy (Vardanega and Maziero, 2001; and Kim et. al., 1998). Second, simulation clocks of those partitions should be synchronized during the execution. Synchronization of different partitions has always been a challenging task and extensive research works have been performed in this area. The conservative synchronization (one of the major approaches) requires the partitioned models to ask for synchronization request every time an event occurs. Therefore, no information is lost when this type of synchronization is used. However, conservative synchronization approaches can introduce excessive overhead in execution, and result in little parallelism, which can eliminate the speedup promised by distributed simulation (Xu, 2006). Optimistic synchronization (another major approach) on the other hand suffers from a possibility of roll-back, which can cause computationally intense operations (Jefferson, 1985; Fujimoto, 2000). Third, as communications among the partitions during the

execution may have a high impact on the performance of a distributed simulation, developing a reliable and efficient computational framework is another challenge (David et al., 1992). It is noted that as the number of partitions increases, the frequency of communications required by the partitions increases as well. Therefore, while it is beneficial to increase the number of partitions in terms of reduction of execution requirement for each individual simulation, it is at the same time disadvantageous in term of time synchronization and communication requirements. Therefore, an appropriate number of partitions must be determined considering both requirements.

In the rest of this chapter, we first assess and analyze the compromise between reduced computations and increased communication requirements in detail in partitioning of large scale simulations. Based on the analysis, we then propose a methodology to enable a near optimal partitioning of large-scale simulations into multiple pieces. Our methodology is novel for several reasons. First, while most of the previous studies have focused on simulation partitioning using a hierarchical topology of the existing models, we propose a new approach based on an efficient, polynomial-time heuristic algorithm (using a combination of modified Prim's algorithm and *k*-way partitioning). Second, to the best of our knowledge, this study is the first attempt to develop a partitioning methodology considering epoch time synchronization and grid computing (see Chapter 2 and 7.2 for more details). Third, we devise a generic method (and software converters), which generates a graphical representation of a simulation model automatically. By generic, we mean the method can be applicable to any kind of discrete event simulation

models. Lastly, a distinctive method is developed, which allows us to automatically obtain traffic flow information of arcs in the graph mentioned above.

Part A of the proposed method is founded on a graph based algorithm. Here, we first map a considered simulation model to an underlying graph. To this end, the physical and logical processes in a simulation model are expressed as nodes (N), and the connections between them as arcs (A) in a graph, $G(N, A)$. These processes are either the ones which add events into the future event list (*e.g.*, Delay, Route, or Create processes in Arena®) or the ones which do not add any event into the future event calendar yet still change the system status (*e.g.*, Seize, Queue, and Release processes in Arena®). Each arc in the graph corresponds with an interaction (transferring of information or entities) between two nodes. The number of interactions between each pair of nodes in the graph represents the "traffic" on the corresponding arc. And, each node is associated with the computation time for the corresponding process in the simulation model. Here, cost of an arc in the graph is defined as a function of both the traffic on the arc as well as computation times of two nodes connected by the arc. While an arc can be associated with only two nodes (*i.e.*, a beginning node and an end node) a node might be associated with multiple arcs. Considering this fact, the proposed cost function is devised without involving the computation times of the nodes multiple times (see Eq. (7.2)). In the proposed approach, a single replication of the monolithic simulation is run for a certain length (which is much shorter than the original replication length) first to identify the underlying graph. Then, the constructed graph is divided into k partitions according to our proposed algorithm modifying Prim's algorithm (Cormen et al., 2001). In this

algorithm, the goal is to minimize the overall simulation execution time via balancing distributed computational requirements, minimizing communications among the partitioned models ("minimum traffic on arcs"), and minimizing the "maximum computation time in nodes".

Part B of our proposed methodology focuses on selecting the most suitable number of partitions which minimizes the total execution time according to the results obtained from Part A. Upon identifying the most appropriate number of partitions, partitioned models are distributed across a computational grid. In this dissertation, partitioned models are manually constructed according to the identified partitioning policy; however, the process will be automated using an automatic simulation model generation methodology (Son and Wysk, 2001) in the future. In this dissertation, both Parts A and B of the proposed method are illustrated and demonstrated using a generic job-shop simulation.

7.2 Proposed Partitioning Problem Formulation and Methodology

7.2.1 Formulation of Partitioning Problem

As mentioned earlier, our main objective in this chapter is to minimize the total simulation execution time without compromising the accuracy of simulation results. Formally, given a monolithic simulation model and m available computers, we intend to find a number $k \in [1,2,\dots,m]$ of partitions, whose corresponding simulation execution time is shortest. To this end, we need to divide an original model into k partitions (see Figure 7.1).

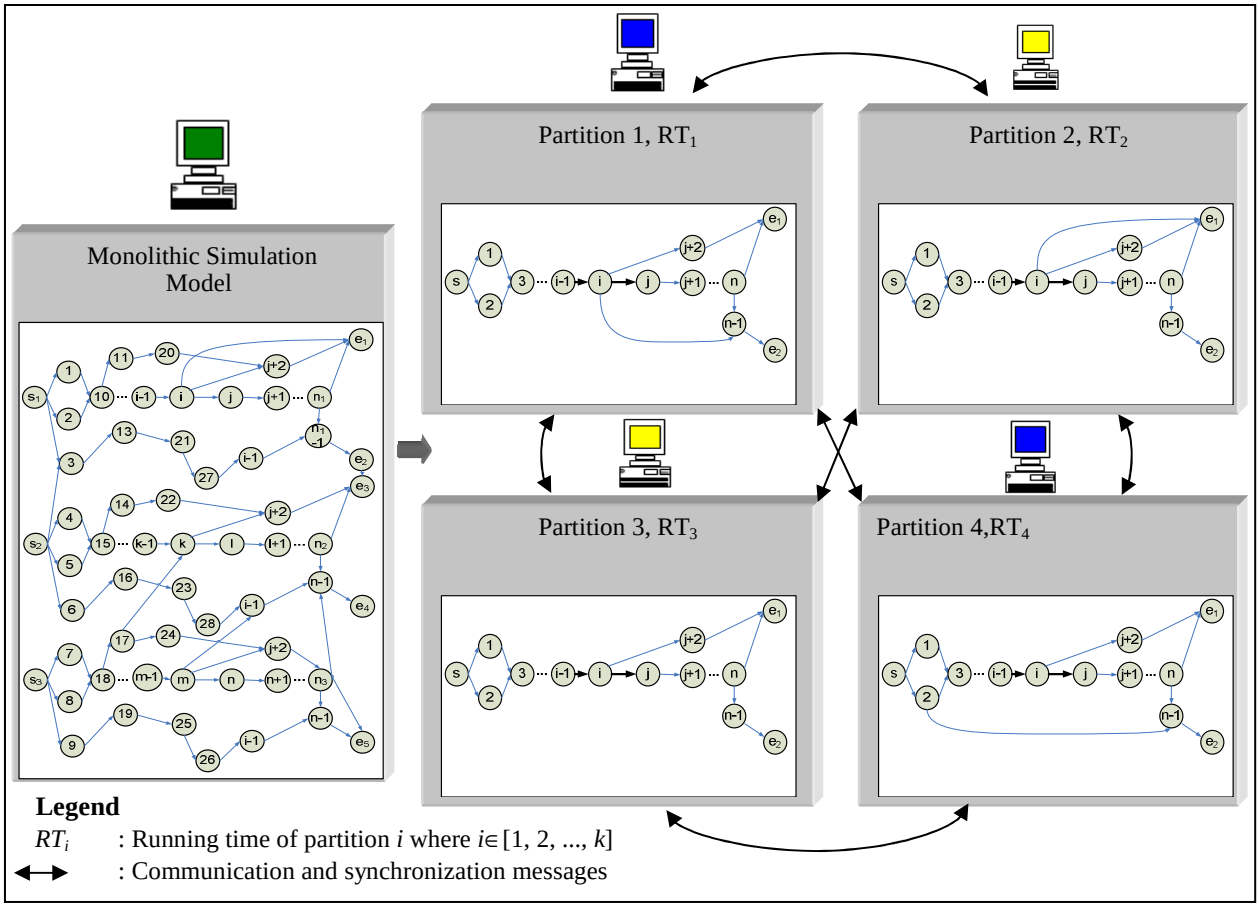


Figure 7.1: Monolithic and partitioned simulation models with their underlying graphs

After dividing the original model into k partitions, these k simulation partitions start running at the same time. The overall simulation completion time depends on the completion time of the last partition. Consequently, our goal (objective function) is to minimize the maximum of all individual simulation completion times (see Eq. (7.1)), which are summations of 1) run time in each partition $p \in [1,2,\dots,k]$ (RT_p) and 2) delay time (computational waste) in each partition p (DT_p) caused by synchronization with other partitions. Details of Eq. (7.1) are explained later in this chapter.

$$MinMax_{p=1}^{k}(RT_p + DT_p) \qquad \text{Eq. (7.1)}$$

Therefore in our work, we intend to find k, which minimizes the objective function in Eq. (7.1). However, in order to find such k, we need to find the best partitioning (involving the least total running time) among all possible combinations of partitioning for a given *k.* Thus, we face two different problems here: 1) given a number of partitions (k), how to partition the monolithic model so that the "total running time" is minimized (Part A of our methodology; see Chapter 7.2.2.2.1) and 2) given m different computers, how many of them should be used to obtain the best result (*i.e.* finding k whose corresponding "total running time" is minimum, where $k \in [1,2,\dots,m]$ (Part B of our methodology; see Chapter 7.2.2.2.2).

Figure 7.2 depicts the framework of proposed partitioning algorithm including both Parts A and B, where we first extract a graph representation (see Figure 7.1) of the original simulation model, and then seek for the best partitioning configuration. Each node in the graph represents a building block in the process-oriented simulation model. Each arc represents a feasible path for entity or information movement between the blocks. In the original simulation model, all of these movements as well as logical processes residing in nodes result in internal events (some of which are placed in the future event calendar and executed at future simulation times, and others are executed at the current simulation time). However, once the partitioning occurs, some of those movements result in external (interaction) events if the destination node belongs to

another partition. As the number of external events increases, the simulation runtime of the partitioned models increases as well, due to increased delay times.

In our proposed method, partitions are created in a way that the number of interaction events between subsequent partitioned models as well as the total simulation run time is minimized. This partitioning method is a variant of k-way partitioning, a well known NP-hard graph theoretic method (Andreev and Räcke, 2004). The computation time required to create the partitions contributes to the overall simulation time, and thus needs to be reduced as well. Therefore, we employ a polynomial-time heuristic to create these partitions. This heuristic is also further modified to enable load-balancing among the partitions. This algorithm is applicable to the family of graphs with costs on arcs only. Therefore, in this study the very first step is to transform the underlying graph generated for the original monolithic model into a graph with cost on arcs only. In other words, all the costs representing the computational load on nodes should be translated to costs on arcs.

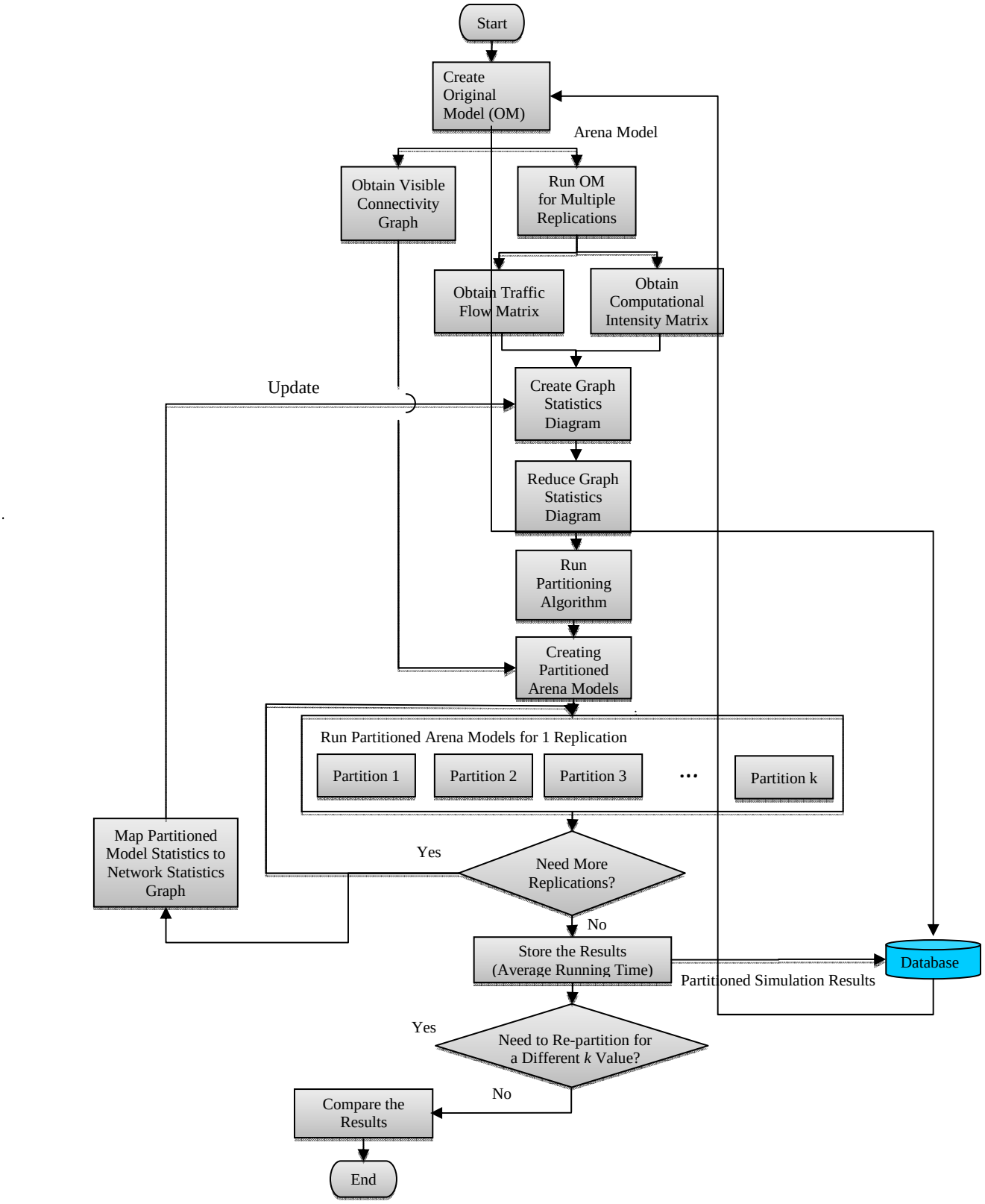

Start
Create Original Model (OM)
Arena Model
Obtain Visible Connectivity Graph
Run OM for Multiple Replications
Obtain Traffic Flow Matrix
Obtain Computational Intensity Matrix
Update
Create Graph Statistics Diagram
Reduce Graph Statistics Diagram
Run Partitioning Algorithm
Creating Partitioned Arena Models
Run Partitioned Arena Models for 1 Replication
Partition 1
Partition 2
Partition 3
...
Partition k
Map Partitioned Model Statistics to Network Statistics Graph
Yes
Need More Replications?
No
Store the Results (Average Running Time)
Database
Partitioned Simulation Results
Yes
Need to Re-partition for a Different k Value?
No
Compare the Results
End

Figure 7.2: Overview of the proposed partitioning approach

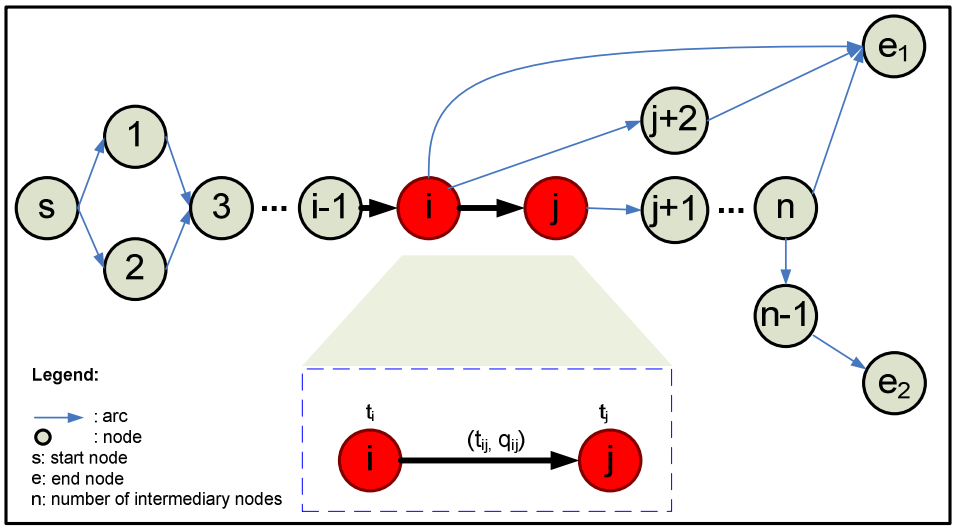


Figure 7.3: Computational time and traffic flow representation on an underlying graph of simulation model

$$c_{ij} = \begin{cases} q_{ij}(t_i + t_{ij}) & if\ i\ is\ a\ start\ node \\ q_{ij}(t_j + t_{ij}) & if\ j\ is\ a\ start\ node \\ q_{ij}\left(\frac{t_i + t_j}{2} + t_{ij}\right) & otherwise \end{cases} \quad \text{Eq. (7.2)}$$

In our partitioning algorithm, cost of each arc is first computed according to Eq. (7.2). Then, by cutting the arcs involving lowest costs, partitions are created. Here, a cost for an arc (i, j), c_{ij} is defined as a function of t_i, t_j, t_{ij} and q_{ij} (see Figure 7.3 and Eq. (7. 2)), where t_i is the computation time needed for execution of a single logical process at node i, t_j is the computation time needed for execution of a single logical process at node j, t_{ij} is the computation time needed for movement of an entity from node i to node j, and q_{ij} is traffic flow volume on the arc between nodes i and j (*e.g.*, 50

entities per time unit have traveled from node i to node j). If an arc is connected to a start node, the entire computational burden of that start node should be added to the cost of that arc (outgoing arc) since it does not have any incoming arcs. Similarly, if an arc is connected to an end node, the entire computational burden of that end node should be added to the cost of that arc (incoming arc) since it does not have any outgoing arcs. Otherwise, the total computation cost of each node is divided into two as an entity enters a particular node from exactly one entry arc and leaves the node from exactly one exit arc regardless of the total number of arcs associated with that node.

Running time of each partition p (RT_p) depends on 1) running time of internal events in partition p which is define by the summation of all arc costs within the partition p (first term in Eq. (7.3)) and 2) running time that partition p spends communicating with other partitions (second term in Eq. (7.3)). In Eq. (7.3), t_c denotes time to perform one communication (running time for one single communication to be completed), s_{ij} denotes the number of communications between nodes i and j where node i belongs to partition p and node j does not, and I_p is a set of index number of nodes belonging to partition p.

$$RT_p = \sum_{(i,j)\in I_p} c_{ij} + \sum_{i\in I_p, j\notin I_p} t_c s_{ij} \qquad \text{Eq. (7.3)}$$

Eq. (7.4) depicts total delay time in partition p (DT_p) caused by synchronization with other partitions, where $D_{(p,l)}$ is delay time in partition p caused by waiting for

synchronization messages from all the other partitions l. This delay time depends on various factors including the method of synchronizations discussed in Chapter 7.2.3.

$$DT_p = \sum_{l\in[1,\ldots,k],r\neq p} D_{(p,l)} \quad \text{Eq. (7.4)}$$

7.2.2 Proposed Partitioning Methodology

7.2.2.1 Definitions, Modeling Assumptions, and Modifications

In this section, we discuss definitions, assumptions, and modifications made in our research in order to obtain the traffic and computational usage information from an Arena® simulation model and to implement our proposed partitioning algorithm. Although Arena® simulation package is used in this study for the illustration purposes, our proposed methodology is generic, therefore is applicable to other process oriented discrete event simulation packages (*e.g.* ProModel, AutoMod).

The following definitions have been made to illustrate the proposed partitioning method:

- Original Model (OM): A single, gigantic simulation model before partitioning.
- Visible Connectivity Graph: A graph representation of the blocks and their connections in the Original Model.
- Server: A “server” is defined as a series of building blocks which can be isolated from the remaining blocks by cutting only two connections. In other words, these building blocks are connected to only a single starting block and a single ending

block (*e.g.*, starting with a "Station" block and ending with a "Route" block). As an example, blocks shown in Figure 4 define a server in an Arena® model.

- Traffic Flow Matrix: A matrix *Q*, where each element is a traffic flow on an arc between nodes and .
- Computational Intensity Matrix: A matrix *T*, where each element is a computation time involved in the movement of an entity from block to block .
- Graph Statistics Diagram: A graph *C*, where each element is a function of traffic flow and computational intensity (see Eq. (7.2)).
- Partitioned Model: A division of the Original Model into partitions, or federates.

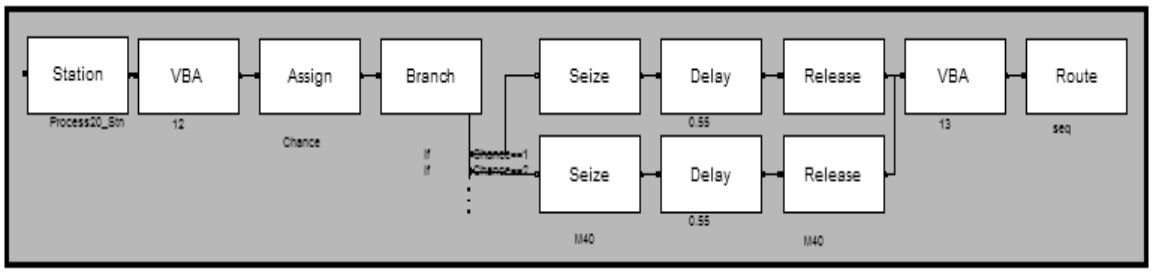


Figure 7.4: A group of highly related blocks in Arena® which is referred to as a *server* in this study

The following list discusses assumptions made and facts considered throughout the development of the proposed methodology:

- It is assumed that simulation models are created using modules available in the "Blocks" and "Elements" templates of Arena®. There are two motivations behind this assumption. First, "Blocks" and "Elements" are two most generic templates, using which various systems can be modeled in a desired level of detail. While the

proposed methodology is illustrated for a job shop manufacturing system, it is believed that key aspects of the proposed methodology can be directly applicable to other systems. Second, building blocks pertaining to all the high-level templates such as “Basic Processes” and “Advanced Basic Processes” available in Arena® are composed of building blocks pertaining to the “Blocks” and “Elements” templates. Therefore, our proposed approach can be extended to the simulation models developed using other templates in Arena®. As other process oriented discrete event simulation packages support those building blocks analogous to the “Blocks” and “Elements” templates in Arena®, the proposed method is applicable to other packages as well.

- While an OM may be composed of numerous blocks and arcs connecting the blocks, we do not need to tally statistics on all the blocks. Instead, we combine groups of highly related blocks in the OM into “servers” to generate a reduced graph. Here, while we deal with a reduced graph, the intensity of traffic on the remaining arcs does not change and any traffic related information is not lost. For instance, we can represent all blocks in a work cell by a single server in a reduced graph.
- It is noted that a greatest amount of computational usage occurs when an entity (or job) is transferred from one server to another. This is because it involves material handling activities, which are usually modeled in a continuous fashion (*e.g.* conveyors or automatically-guided-vehicles). Another major source of computational resource usage is the representation of processes within a server, especially when their behavior is modeled in a highest granularity to depict very small changes in their

status during simulation. Various approaches can be used to represent the processes, such as recursive algorithms, process emulators, and differential equations.

The following modifications have been made to facilitate implementation of the proposed methodology:

- All the blocks in an Arena® model are both labeled and tagged in a way that they are easily tractable (*i.e.* "B1", "B2"…).
- For each server in the model, two Visual Basic Application (VBA) blocks are inserted to tally the computational usage between servers. The first one is inserted right before the first block of the server, and the second one is inserted right after the last block of the server.
- A VBA block with an associated real-time delay is added at the end of each server to mimic the computational intensity caused by the inter-cell movements of jobs in a realistic setting. In reality, these inter-cell movements of jobs can be performed by dedicated robots, conveyors and automatic guided vehicles.

7.2.2.2 Algorithm and Partitioning Details

This section discusses details of the proposed partitioning framework together with the embedded partitioning algorithm which is depicted in Figure 7.2.

7.2.2.2.1 Part A: Partitioning a monolithic simulation into *k* pieces

Step 1: Create an Original Model (OM).

Step 2: Obtain the Visible Connectivity Graph. The OM has an underlying graph structure, which is automatically extracted from the source code of the model in this work. It is noted that in Arena® simulation package, the source code is written in SIMAN language.

Step 3: Run OM for multiple replications for a certain replication length (which is considerably shorter than the entire replication length). Then, we save sought statistics (*i.e.*, mean cycle time, machine utilization, average number of parts waiting in the queue, and throughput rate) of this shortened replication into a database. The number of replications is to be defined based on the desired accuracy (half-width of the confidence interval) of the simulation.

Step 4: Collect graph information via Traffic Flow Matrix and Computational Intensity Matrix.

- Obtain Traffic Flow Matrix: In order to obtain the traffic information between different servers, each entity's movement is monitored. To do so, all the blocks that a specific entity passes through are detected to determine the sequence of blocks for each entity. Once these servers are determined, "the sequences of blocks" of entities are modified to reflect "the sequences of servers". Afterwards, the number of interactions between servers is counted to obtain the overall traffic information.
- Obtain Computational Intensity Matrix: In order to obtain the computation time between the servers, the start time and end time of each transfer movement is

recorded and the difference is calculated. Computational intensity matrix depicts the average of these computed difference values for all the entities going through the same route.

Step 5: Create Graph Statistics Diagram and reduce it by defining the servers (see Chapter 7.2.2). The Graph Statistics Diagram is a function of the Traffic Flow Matrix and Computational Matrix (see Eq. (7.2)).

Step 6: Execute the proposed partitioning algorithm to create k federates. The proposed partitioning algorithm is developed to minimize synchronization requirements as well as to balance the computational load among partitioned federates. The steps for this algorithm are as follows (see Figures 7.5 and 7.6):

- Begin with the Graph Statistics Diagram (Figure 7.5(a)).
- Employ modified Prim's algorithm to build a maximum spanning tree which connects arcs in a non-increasing order of cost rather than in a non-decreasing order of cost (Figure 7.5(b)).
- Cut tree into k-order forest by cutting arcs. The cheapest arcs are first selected; if cutting an arc results in a federate which is smaller than the minimum size (maximum $(\frac{k}{n-2}, 2)$, then the next cheapest arc is selected. This process is repeated for a total of $k-1$ arcs cut. (Figures 7.6(a) and 7.6(b)).
- The Graph Statistics Diagram has now been partitioned into k federates. Map these federates in the Graph Statistics Diagram to the nodes in the Original Model.

Step 7: Run Partitioned Model (multiple federates) for one replication.

Step 8: Check the replication number. If simulation has been run for the required number of replications, proceed to Step 9. Otherwise, continue to Step 7.

Step 9: Compare simulation results of the Partitioned Models with the simulation results of the Original Model.

Step 10: End.

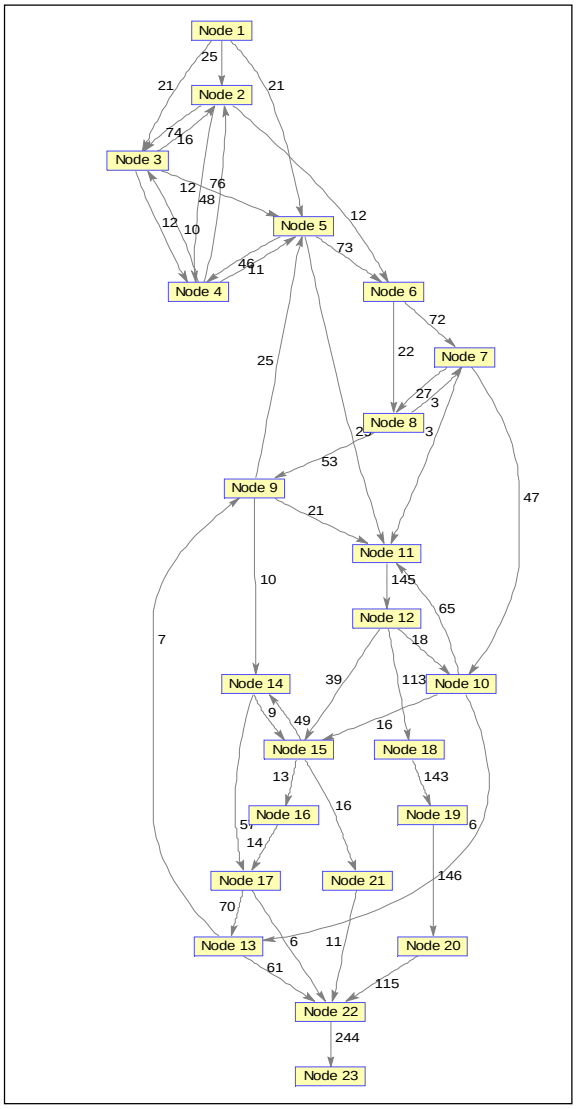


Figure 7.5(a): Graph Statistics Diagram

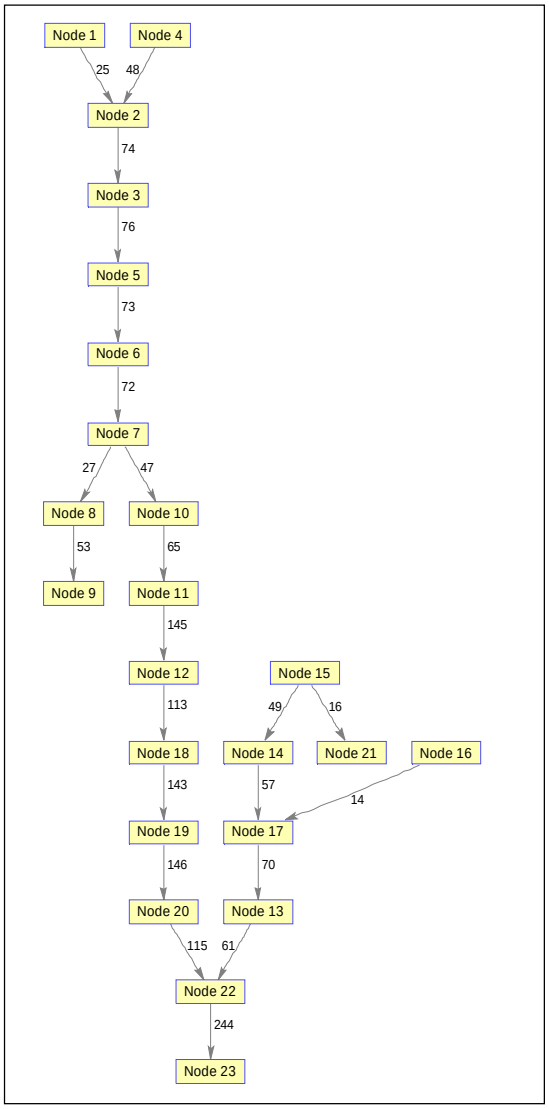


Figure 7.5(b): Maximum Spanning Tree

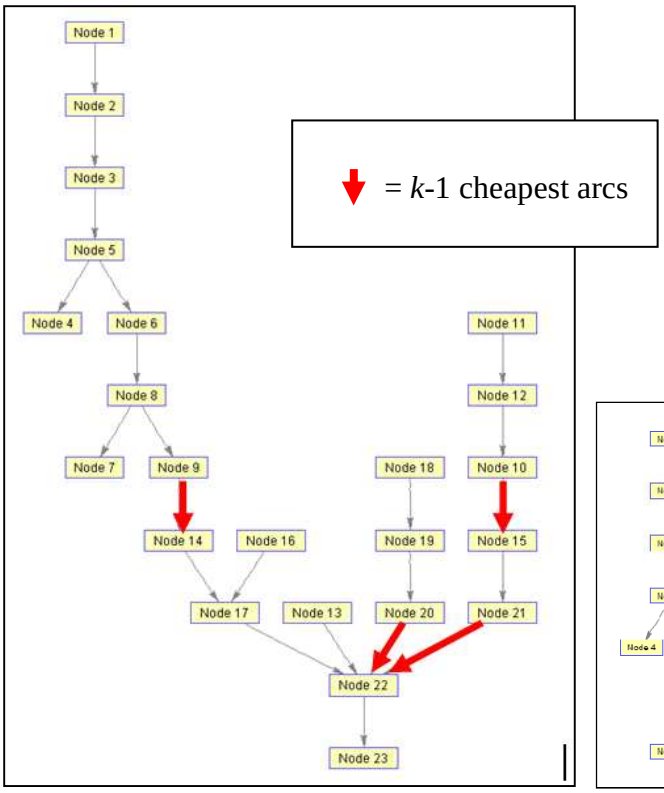


Figure 7.6(a): Cut $\boldsymbol{k-1}$ arcs

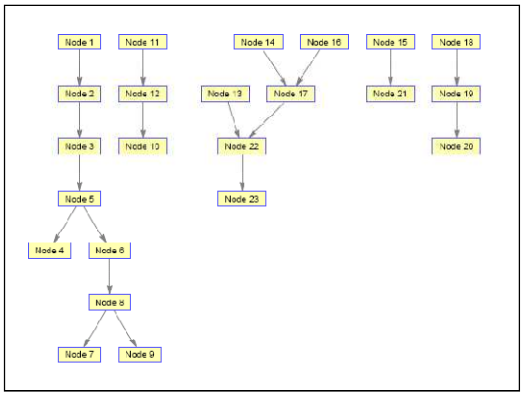


Figure 7.6(b): $\boldsymbol{k}$ partitions

7.2.2.2.2 Part B: Finding the most advantageous number of partitions

The aim of Part B of our proposed simulation partitioning methodology is to find out the most advantageous number of partitions by comparing the results obtained from Part A for various feasible numbers of partitions (k values).

Step 1: Run the algorithm explained in Part A for k partitions, where the initial k is 2. This simulation is to be run for multiple replications, where the number of replications is to be defined based on the desired accuracy (half-width of the confidence interval) of the simulation. At the end of the last replication, store the average running time.

Step 2: If k is less than m (the number of available computation resources in the network), increment k by 1 and repeat Step 1. Otherwise, go to Step 3.

Step 3: Compare the average running times obtained from Step 1 for all the k values.

Step 4: The k, which results in the smallest average running time, denotes the most advantageous number of partitions.

Step 5: End.

7.2.3 Computational Infrastructure and Grid Computing Framework

Among many available ways to facilitate distributed computing (federation of the partitions), Grid Computing is used in our research for four reasons. For the aspect of communication (other than Grid Computing operations) between the partitioned simulations, we use a Web Services based communication infrastructure (Lee et al., 2008). Web Services provide an elegant mechanism for communicating among distributed components without keeping much state information, as is required in conventional means such as socket based interaction. Details can be found in Chapter 5.3 and 5.4 for Grid Computing and Web Services technologies used in this dissertation.

7.2.4 Case Study: Manufacturing Enterprise Simulation

This section discusses a generic prototype job-shop (see Figure 7.8) used to illustrate the proposed approach. The prototype job shop is composed of 20 work cells performing various processes, where each cell contains two identical machines. In

addition, there is an inspection cell which consists of two inspection stations. Four different jobs are processed in this system, with each job having its own routing sequence (note that all the jobs need to be inspected before completion) (see Table 7.3). The jobs and their routing sequences in Table 7.3 have been arbitrarily selected in this study. The flow of jobs through the system is driven by a push system. When a job is sent to a cell according to its routing, it is assigned to one of the machines based on a balanced-workload machine selection rule. This machine selection rule receives the machine utilizations from each machine and directs the very next job to the machine having less utilization. This way, the workload between the machines within a cell is balanced. There is a dedicated queue, which is called "internal queue" for each machine in our system. Once a job is assigned to a particular machine, it takes its place in the corresponding internal queue of that machine and it cannot be re-directed to another internal queue. The processing time of each job at each work cell is described by a statistical distribution function embedded in the simulation model. It is assumed that pre-emption is not allowed and processing times in various work cells are statistically independent. After the job processing is completed in a work cell, it is sent to another cell based on its routing sequence. The inter-cell movements can be performed in various ways such as conveyors, robots or manual transportation. In this study, a dedicated component (*i.e.*, VBA block) is employed in each server to represent the computational load caused by the corresponding inter-cell movements from the current server to the very next server based on the pre-defined sequence for each particular job. This is enabled via a function embedded into this VBA block to generate corresponding real-

time delay. The elapsed time of computation in this component depends on the type of the movement (*i.e.*, conveyor, robot or manual transportation) and the distance occurred.

Table 7.1: Process routing sequence ($\boldsymbol{R_{ik}}$) for job $\boldsymbol{i}$, step $\boldsymbol{k}$, "$\boldsymbol{I}$"= Inspection

i \ *k*	1	2	3	4	5	6	7	8	8	10	11	12	13	14	15	16	17	18
1	1	3	2	3	4	5	7	6	10	11	9	12	8	13	14	15	16	I
2	4	3	1	2	4	5	6	9	10	11	17	18	19	I				
3	2	4	3	1	5	7	8	10	11	9	14	20	I					
4	1	2	1	2	4	5	6	7	8	4	10	11	14	13	16	12	I	

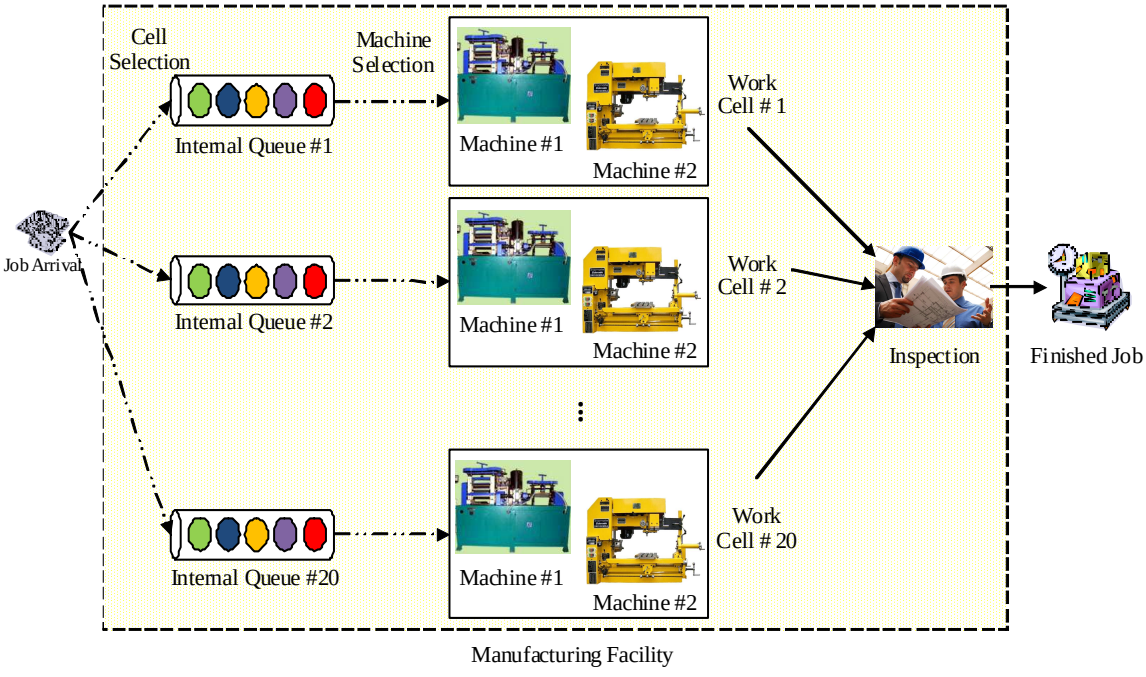


Figure 7.7: Job shop configuration (case study used in this study)

As discussed in Chapter 7.1, running time of each partition depends on running time of internal events and delay time caused by synchronization requirements with other partitions. Here, the delay time is directly relevant with interaction events communicated among the simulation partitions, which are domain specific. In this section, various events occurring in the considered manufacturing system are discussed beginning with the ones mostly related to the manufacturing enterprise (supply chain) and followed by those for the individual member facility (focus of this dissertation).

A supply chain event is “any individual outcome (or non-outcome) of a supply chain cycle, (sub) process, activity, or task” (Alvarenga and Schoenthaler, 2003). Some events have significant consequences and therefore must be monitored closely, while others are of lesser importance. The critical problem lies in extracting the significant events and responding to them in real-time. Doing so requires an ability to monitor them proactively, simulate them to help decision-making, and use them to control and measure business processes. In general, events in an organization can be of two types: 1) internal interaction events and 2) external interaction events (see Table 7.2).

Table 7.2: Interaction events in a manufacturing system

Event Type	Example
Internal Interaction Events	End of a task
	Beginning of a task
	Event “stock partially available” as a result of the “check availability” task
	Event "out of stock" as a result of the “check availability” task
External Interaction Events	New order arrival
	Inbound shipment delay

	Import policy change

Internal interaction events are the events happening within a facility unit. External interaction events, on the other hand, are the ones related to other supply chain partners or an external environment. Although the frequency of external interaction events is in general considerably low compared to that of internal interaction events, the impact of external event may be as significant as that of the internal events. Because our current focus is limited to single facility simulations, only internal interaction events are taken into consideration (*i.e.*, the end of a task and the beginning of a task). As part of the future work, external interaction events will also be considered for the entire supply chain simulations. Once the partitioning of simulation into multiple federates takes place, some of the internal events in the Original Model are transformed to external events because they contain the information that should be shared with other partitions.

7.3 Experimental Setup and Results

In this work, we have conducted an experiment to demonstrate our proposed partitioning algorithm and computational infrastructure and to evaluate their efficacy while not embedding the proposed partitioning algorithm into the greater DDDAMS framework. Details of the experiment setup as well as its results are discussed below in two main steps.

Step 1: A gigantic monolithic simulation model (*i.e.*, Original Model) is built to represent the considered manufacturing system (see Chapter 7.2.4). Then using our proposed partitioning algorithm, partitions as well as their underlying graph representations are

determined. The best number of partitions can be found by repeating the experiment for different number of partitions based on computation resources availability. The best result obtained among all experiments determines the close to optimal number of partitions. In this experiment, our Original Model is partitioned into two, three and four federates. The simulation models corresponding to each partition (*i.e.*, Partitioned Model) are created in Arena® simulation package. To be able to compare the results of above mentioned models, we define two metrics as described below:

- Simulation Accuracy: To measure the accuracy of partitioned simulation, we compare the mean cycle times between the Original Model and the Partitioned Model (see Eq. (7.5)).

$$A = 1 - |1 - \frac{\overline{CT}_{OM}}{\overline{CT}_{PM}}| \qquad \text{Eq. (7.5)}$$

In Eq. (7.5), A is accuracy, $\overline{CT}_{OM}$ is mean cycle time obtained from Original Model, and $\overline{CT}_{PM}$ is the mean cycle time obtained from Partitioned Model which is always greater than or equal to $\overline{CT}_{OM}$ due to the extra time which may be added to the cycle time for synchronization purposes. Here, A is a unitless measure where its values range from zero to one. The greater the A, the better the simulation accuracy is. Under the ideal conditions, A is equal to one. The minimum requirement for decent simulation accuracy is set to 0.95 in this study.

- Partitioning Performance: This is the ratio of the running time of the Original Model to that of the Partitioned Model (see Eq. (7.6)).

$$P = \frac{RT_{OM}}{RT_{PM}} \qquad \text{Eq. (7.6)}$$

In Eq. (7.6), P is Partitioning Performance, $RT_{OM}$is simulation running time of Original Model, and $RT_{PM}$is the simulation running time of Partitioned Model. Here, P is a nonnegative unitless ratio where the greater the P, the better the partitioning performance is. When P is equal to one, it denotes that the Original Model and the Partitioned Model involve the same running times.

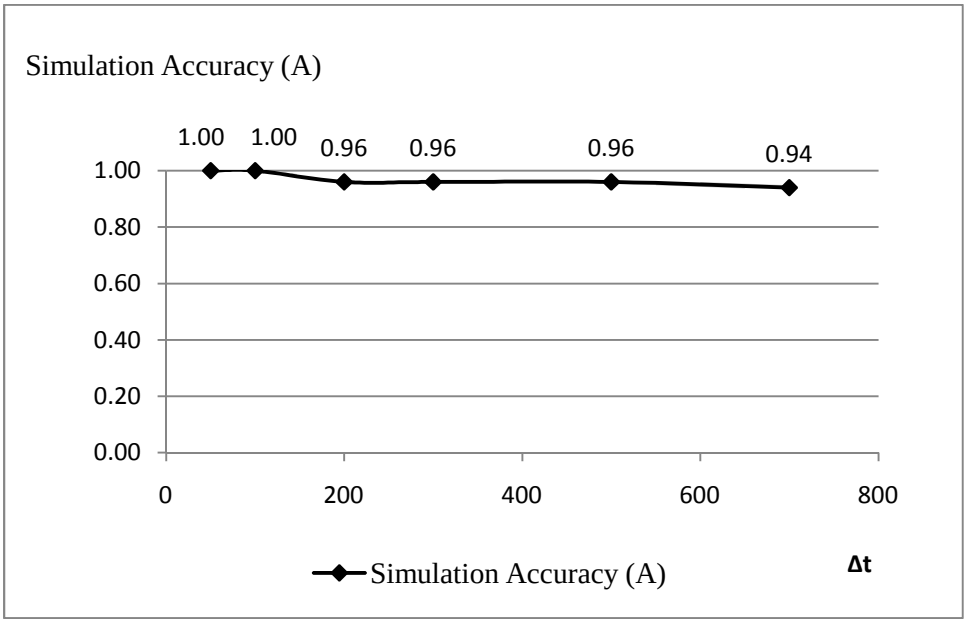


Figure 7.8: Accuracy of partitioned simulations for varying Δt values

Step 2: The Original Model as well as the Partitioned Model is run to completion for various simulation lengths under changing simulation conditions. The results in terms of the metrics explained above obtained from these experiments are then summarized. We evaluated the impacts of two different parameters on the simulation accuracy and partitioning performance of existing models: 1) time interval used for synchronization ($\Delta \boldsymbol{t}$), and 2) simulated inter-server transfer computation time which is a type of delay to represent computational intensity. In both Partitioned Model and Original Model, an identical seed value is used for random number generation to ensure identical creation of entity types.

In order to decrease computational burden without sacrificing the desired simulation accuracy of the Partitioned Model, we need to find the most appropriate Δt value. Here, as $\Delta \boldsymbol{t}$ decreases (*i.e.*, goes to zero), the simulation accuracy increases (*i.e.*, reaches to one). However, as Δt decreases, the computational intensity of the Partitioned Model increases as well leading the resultant executing times of these simulations to be larger. Figure 7.8 depicts the simulation accuracy of partitioned simulations for varying $\Delta \boldsymbol{t}$ values where other conditions are fixed. An examplary calculation of each instance (data point) in Figure 7.8 is explained in Table 7.3. Table 7.3 depicts the comparison of mean cycle times (Simulation Accuracy) obtained from Original Model versus Partitioned Model for seven replications, where each of them has simulation duration of 40 minutes for a fixed value of Δt (500 miliseconds) and a fixed value of inter-server tranfer computation time (700 miliseconds). The same calculation is repeated to find

each data point in Figure 7.8. The maximum Δt value which meets our minimum accuracy requirement (A =0.95) is found to be 500 miliseconds. When Δt value is equal to 500 miliseconds, we obtain a Simulation Accuracy result of 0.96.

Table 7.3: Comparison of mean cycle times from Original Model vs. Partitioned Model

Replication Number	Partitioned Model (CT_{PM})	Original Model (CT_{OM})	Simulation Accuracy
1	10.86	11.21	0.97
2	12.32	12.02	0.98
3	10.18	11.21	0.90
4	10.92	11.25	0.97
5	11.01	11.97	0.91
6	11.10	10.92	0.98
7	12.14	11.96	0.99
	Average Simulation Accuracy		0.96

Second, after finding the maximum Δt (500 miliseconds) enabling the minimum required Simulation Accuracy (A=0.95), we compared the performances of the Partitioned Models involving two, three and four partitions (built based on the proposed partitioning methodology) to those of the partitions built based on another partitioning method (Konas, 1994; and Soule, 1992). In the considered natural partitioning method, the simulation partitions are built according to the geographic locations of the production units in the original model. Therefore, the production units which are closely located belong to the same partition. Figures 7.9(a), (b) and (c) depict the comparison of the simulation runtimes between partitions built by the proposed partitioning methodology vs. partitions built by the natural partitioning method, where *k* is 2, 3, and 4 for (a), (b),

and (c), respectively. In all the cases, the partitions built by proposed partitioning methodology (Part A of our methodology) outperforms the partitions built by the reference partitioning method. For instance, with an inter server transfer computation time of 700 milliseconds, the Partitioned Models involving two, three and four partitions ran 16%, 19% and 28% faster than the models built by the reference partitioning method, respectively. In addition, the performances of the Partitioned Models involving two, three and four partitions are compared against that of the Original Model with increasing values of simulated inter-server transfer computation time (see Figure 7.10(d)). Among the Partitioned Models, the Partitioned Model with two partitions (k=2) runs considerable faster than the Partioned Models with higher partitions (Part B of our methodology). With the simulated inter-server computation time less than 185 milliseconds (0.185 sec), the Partitioned Models (with two, three and four partitions) showed no considerable advantage in running time. At values greater than 185 miliseconds (the case with two partitions) and 235 miliseconds (the case with three partitions), however, the Partitioned Models run significantly faster than the Original Model. The Partitioned Model with four partitions outperforms the Original Model at values greater than 405 miliseconds although it still runs slower than the Partitioned Models with two and three partitions. As depicted in Figures 7.10(a), 10(b), 10(c), and 10(d), partitioning becomes more beneficial as the inter-server computational intensity increases. For example, with a delay of 700 milliseconds, the Partitioned Models with two and three partitions ran 43% and 64% faster than the Original Model, respectively (see Figure 7.9(d)). It should be noted that in a real world setting, where inter-server computation is more intense than what's

considered in our experiment due to highly congested material handling activities, more significant reduction of the simulation execution time can be obtained via the proposed partitioning.

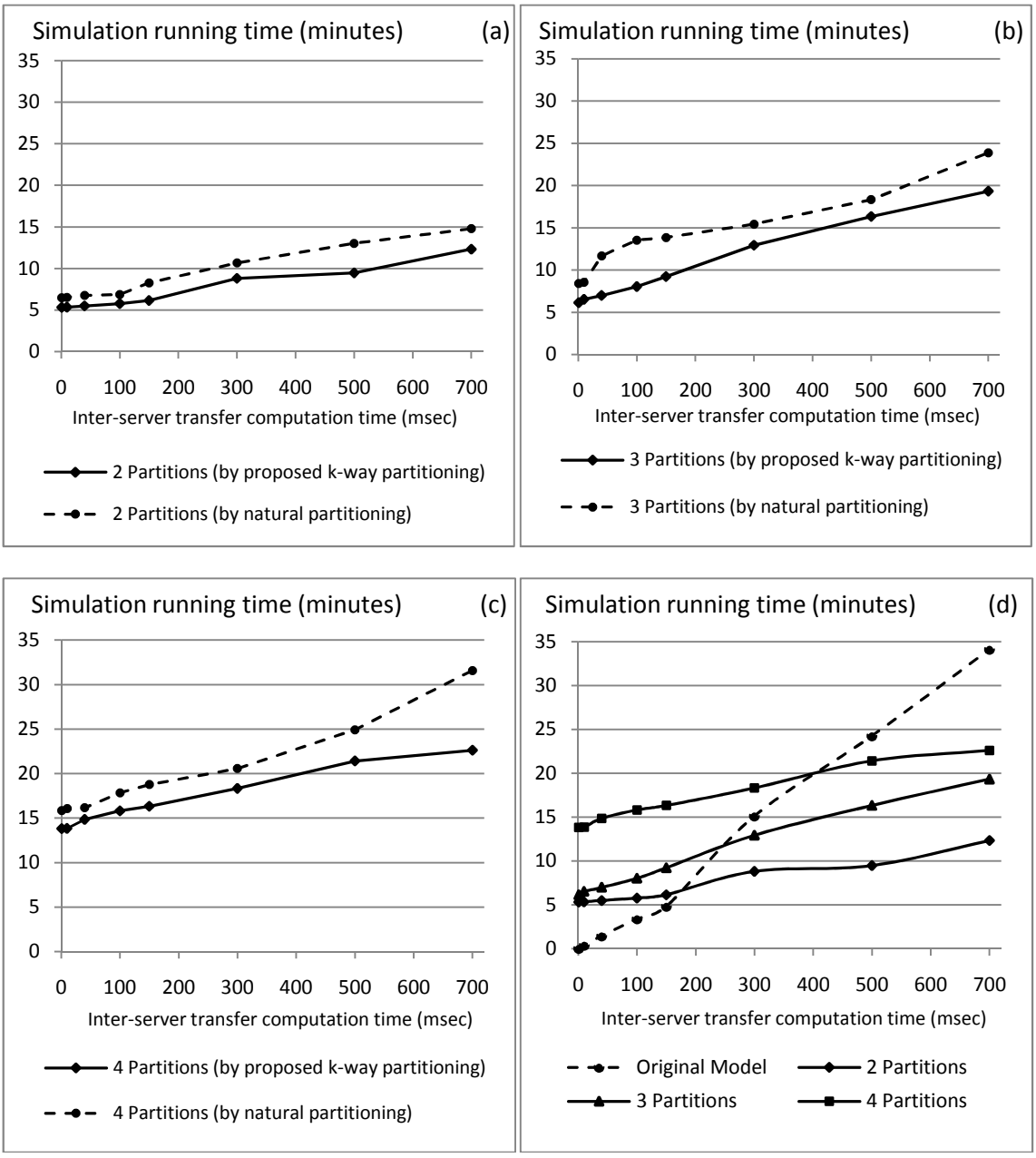


Figure 7.9: Simulation runtime comparisons between the proposed approach and a reference approach (a, b, c); and the Original Model (d)

7.4 Concluding Remarks

In this chapter, a novel simulation partitioning methodology was proposed as an extension to DDDAMS framework to enable a reduction in total simulation execution time by partitioning large-scale simulations into multiple federates which can run in different computers in a synchronized manner. The criticality of the compromise between reduced internal computation of each partition and increased synchronization efforts between the partitions were assessed to allow for realization of the most efficient partitions. We also presented an approach to find the most advantageous number of partitioned simulations by comparing partitioning performances of the partitioned models. The synchronization of the logical times of the created partitioned were enabled using a novel method based on Epoch synchronization where appropriate time intervals were determined based on the off-line simulation analyses. The same computational grid framework which is currently used in implementation of DDDAMS was employed for execution of the partitioned simulations created by the proposed methodology. The proposed partitioning framework involved a graph based partitioning algorithm which was used to partition the underlying graph of the monolithic model (*i.e.*, Original Model) into multiple graphs to determine the basic structure of the Partitioned Model. The partitioning was performed for a various number of partitions and their performances were compared against that of the models built based on another partitioning method available in the literature. The experimental results revealed that the Partitioned Model

built by proposed partitioning methodology outperformed the model built by the reference partitioning method in all the cases. The experimental results also revealed that the proposed partitioning approach reduced simulation execution time significantly while maintaining accuracy as opposed to the monolithic simulation approach as the inter-server computational intensity increased. For example, with an inter-server transfer computation time of 700 milliseconds, the Partitioned Model ran 64% faster than the Original Model. In a real world setting, where inter-server computation is more intense than what's considered in our study due to highly congested material handling activities, even further reduction of the simulation execution time can be obtained via the proposed partitioning. The next stage of this research will concern itself with automatically generating the partitioned models based on the proposed partitioning strategy and demonstrating the proposed method under the proposed DDDAMS framework.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this chapter, the main contributions of this dissertation and future research works are discussed.

8.1 Contributions of Dissertation

A major list of contributions in this dissertation is discussed below:

- A comprehensive DDDAMS system architecture has been proposed involving state-of-the-art information and computational technologies, including real-time simulation, grid computing, web services technology, sensor network, and database.
- To enable adaptive fidelity switching of the DDDAM-simulation against available computational resources and sensory updates from the real system, four algorithms have been developed, including a data filtering and abnormality detection algorithm (Algorithm 1), a preliminary fidelity selection algorithm (Algorithm 2), a fidelity assignment algorithm (Algorithm 3), and a prediction and task generation algorithm (Algorithm 4).
- Algorithm 1 has been developed to filter noise and detect any abnormalities that may appear in the status of the system based on the measurement of the current sensory data (such as temperature and pressure). The generic strength of the proposed method for

this algorithm lies in its ability to spot abnormal data by using the combination of the X chart for Individuals and the Moving Range chart.

- Algorithm 2 (fidelity selection algorithm) has been initially proposed and developed using the Bayesian Belief Network. Later, it has been enhanced using Sequential Monte Carlo sampling technique (also known as particle filtering) in order to make efficient inferences to determine the sources of abnormality in the system (shop floor in this research) given large datasets. The proposed method is built such a way to reveal cost-efficient inferences for determining the sources of abnormality in the system while saving from computational burden of the overall simulation system. The proposed algorithm has been first benchmarked against synthetic functions, and then further extended for preventive maintenance and part routing scheduling problems in a semiconductor supply chain. The experimental results have revealed that the developed sequential Monte Carlo-based fidelity selection algorithm was able to catch the system status quite closely.

- Two resampling rules have been proposed in order to be used as part of the particle filtering algorithms such as our Algorithm 2 of DDDAMS framework from different perspectives. These two rules, the variance and bias-based relative efficiency rules, are the improved version of the rules existing in the literature. The proposed rules have been first derived theoretically and then benchmarked against the half-width based resampling rule which is revisited in this dissertation as well as widely known Kong rule using a supply chain simulation in terms of their resampling qualities and computational efficiencies. Results obtained from our simulation experiment have revealed that the

proposed variance based resampling rule outperforms all the other three algorithms including the proposed bias-based resampling rule in terms of the recorded mean RMSE values. When estimating the states of the large scale, dynamic and complex systems such as supply chains, the utilization of this rule is recommended compared to the other three rules discussed in this work as part of particle filtering algorithms. This selection will enable more accurate results via less number of particle set sizes, which in turn prevents the simulations from wasting computational efforts. The proposed variance-based relative sampling rule may result in even greater savings in terms of computational resources when applied to the real-world complex scenarios, where both the state estimation and the measurement are more time-consuming and computationally challenging. On the other hand, the performance of the proposed bias-based relative sampling rule was proven to be weak when compared to the other methods. Appropriate selection of the resampling rule is also critical in achieving desired accuracy levels in estimations when the number of samples those can be drawn is limited with the nature of the process.

- Algorithm 3 (fidelity assignment algorithm) involving mathematical programming has been developed to opt for the available fidelity level of each component by taking the system level computational resource constraint into account. This algorithm obtains a matrix which encapsulates the proper (desirable) fidelity level of each component discussed in Algorithm 2 as an input as well as the available computational resource capacity from the grid computing service, and returns a new matrix which holds the assigned fidelity level for each component in the system that was

evaluated at the current time point. This algorithm is based on the well-known Knapsack problem. The estimate of time that is spent for communication to retrieve the information of available resources (*i.e.,* from the time point when grid manager sends the status request to each one of the computer resources until the time point when the update is received) was not considered when formulating the Knapsack problem. However, the estimations on the virtual connection speed and the communications time among the nodes in the grid can be included as part of future research. These estimations can be based on an appropriate distribution which is based on the historical connection data. This area may also lead another great area of research which focuses on connection reliability and its affect on DDDAMS performance.

- Algorithm 4 (prediction and task generation algorithm) has been developed either involving multi-linear regressions or distributed discrete-event simulation models running on fast mode (evaluating different decision rules such as part selection rules and machine selection rules). It provides a real system with 1) near optimal preventive maintenance (PM) schedule and 2) near optimal part routing (PR) recommendations for operational efficiency of jobs (parts) to be processed.

- As part of extended DDDAMS framework, a novel simulation partitioning methodology was proposed to enable a reduction in total simulation execution time by partitioning large-scale simulations into multiple federates which can run in different computers in a synchronized manner. The criticality of the compromise between reduced internal computation of each partition and increased synchronization efforts between the partitions was assessed to enable the most efficient partitions.

- In addition, an approach to find the most advantageous number of partitioned simulations was proposed by comparing partitioning performances of the partitioned models. The synchronization of the logical times of the created partitioned were enabled using a novel method based on Epoch synchronization (Rathore at al., 2005) where appropriate time intervals were determined based on the off-line simulation analyses. The proposed partitioning framework involved a graph based partitioning algorithm which was used to partition the underlying graph of the monolithic model (i.e., Original Model) into multiple graphs to determine the basic structure of the Partitioned Model. The partitioning was performed for a various number of partitions and their performances were compared against that of the models built based on another partitioning method available in the literature. The experimental results revealed that the Partitioned Model built by proposed partitioning methodology outperformed the model built by the reference partitioning method in all the cases. The experimental results also revealed that the proposed partitioning approach reduced simulation execution time significantly while maintaining accuracy as opposed to the monolithic simulation approach as the inter-server computational intensity increased. For example, with an inter-server transfer computation time of 700 milliseconds, the Partitioned Model ran 64% faster than the Original Model. In a real world setting, where inter-server computation is more intense than what's considered in our study due to highly congested material handling activities, even further reduction of the simulation execution time can be obtained via the proposed partitioning.

- The proposed DDDAMS system has been applied to minimize mean cycle times in a supply chain system by optimizing the PM schedules and part routing schedules of individual echelons. The processes in a three-echelon collaborative, semiconductor supply chain system have been considered to demonstrate performance of the proposed DDDAMS system.

- The automatic fidelity switching in each echelon and nearly optimized resource allocations have been successfully demonstrated. The DDDAM-system with the proposed fidelity selection algorithm has achieved 20 percent reduction in the average waiting time and 15 percent increase in the average machine utilization compared with the aggregated simulation of the considered supply chain. Similarly, the DDDAM system with the proposed fidelity selection algorithm has shown slightly higher values in the average waiting time and less than eight percent decrease in the average machine utilization compared with the detailed simulation of the considered supply chain. It can be concluded from these promising results that sequential Monte Carlo-based fidelity selection leads to decreased waiting time and better resource utilization while saving computational resources and time when integrated into the DDDAM-simulation.

- In his work, the results are obtained from the DDDAM-system when it is connected to a simulated real system (in the highest detail possible) in a realistic virtual setting. When the same DDDAMS system is connected to actual supply chain system with physical resources, there may occur errors in data collection in various layers. These layers include the datasets which are used as part of determining the simulation fidelities as well as the simulation itself, the algorithms responsible for automated

decision-making, the real-time and fast mode simulations as they are conceptual and highly viable to developer perceptions, and the grid framework. The quantification of errors at the various stages as explained above and the sensitivity analysis related to these errors are left as part of the future studies.

- While we focused on the PM for a collaborative supply chain, the generality (and necessary modifications) of the proposed DDDAMS system for the competitive supply chain has been discussed as well.
- It is noted that while particular software packages are used in this dissertation work for illustration and demonstration purposes, the proposed methodology is generic and can be realized using any simulation software package.

8.2 Future Research

Extensions are possible in the methodological, technological, and applications aspects of the research described in this dissertation. We itemize the future venues of this work as shown below:

- Technologically, the effect of integrating different high-speed sensor networks into the DDDAMS system on the performance can be studied.
- Other future research concerns scalability of the proposed system for larger systems such as distributed electricity grid, production scheduling, and transportation management, which involves three major challenges. First, building and managing highly detailed models (highest fidelity models in this case) to represent a real system in an accurate and efficient way is challenging, especially when the simulation is aimed to

support short-term decisions. Second, given the enormous amount of dynamically-changing data that exist in the system, and considering the fact that even a medium size facility is comprised of hundreds of machines and a medium size of a supply chain is comprised of several echelons, a highly efficient sensor network should be employed throughout the system for timely data update. Third, running several DDDAMS simulations (each for a facility or component of the system) in geographically different locations leads to even more complex synchronization problems (both in time and information) to be handled. Yet, realization of the DDDAMS system in such a greater scale will enable significant amounts of savings in computational power usage while allowing for significantly improved global performance.

- A parallelization framework has been conceptually discussed on how further reductions and distribution of the computational burden of algorithms can be enabled while maintaining the accuracy of parameter estimates. Future work of this research primarily concerns itself with the efficient realization of the sequential Monte Carlo-based fidelity selection algorithm in a parallel computing setting under the DDDAMS framework. The synchronization issues regarding both the distributed DDDAMS system and distributed computational resources are also among the challenges to face.

- Regarding the developed resampling rules for particle filtering, the performance of the proposed variance-based and bias-based relative sampling rules were analyzed and compared to the other methods on an individual basis. However, the performance of the methods combining two or more of the aforementioned rules may still outperform the individual rules. Our future work in this aspect will focus on the performance of these

blended resampling schemes. As all four of the discussed algorithms have complexities of $\mathcal{O}(N_s^2)$, they are equivalent when selection of the resampling is based solely on the computational effort.

- Performances of the proposed resampling rules are also worth exploring when employed for the particle filtering algorithms that involve sampling techniques different from the importance sampling technique used in this work such as Gibbs sampling (Gentle, 2002), Metropolis-Hastings sampling (MacEachern et al., 1999), rejection sampling, adaptive importance sampling (Oehlert, 1998), and smoothed importance sampling (Bolviken et al., 2001). Rejection sampling method in which samples are drawn from the filtering density without evaluating the integration is an example technique worth discussing. Here, the future works can be even extended to develop a measure of efficiency of the sampling from importance sampling and rejection sampling and point out the cases in where each of them could be especially useful while adapting the resampling rules discussed in this dissertation.

- The effect of information sharing on the performance of the supply chain is to be understood thoroughly and should be included in the decision making process with a right fidelity at a right time. The role of information sharing is to be researched (answers to questions like: what information needs to be shared? How much information is to be shared? How to use the information to optimize the goals of the supply chain? Is the meaning of the information understood the same throughout the supply chain? need to be sought after).

- The analysis and experiments in this dissertation have been performed with the objective of minimizing mean cycle time within each facility. The consequence of independent (possibly conflicting) objectives of different members of the supply chain on the overall system performance is another area of interest. The tradeoffs between local and global optima can be explored. The number of echelons to be represented in any model of the supply chain for effective analysis is also to be studied.
- The next stage of our research will concern itself with improving various aspects of the proposed partitioning methodology and its demonstration. Enabling a fully automated partitioning in an online fashion in a distributed simulation environment such that an immense simulation can be partitioned into multiple federates at any time during its run (without even completing one single replication) is another line of research extension. In addition, these partitioned simulations can be folded back to one single gigantic simulation depending on diversity and disparity of the computational resources available on the grid. Finally, we are interested in demonstrating the proposed method under the proposed DDDAMS framework.

8.3 Concluding Remarks

Knowledge gained from this dissertation is threefold; methodological, theoretical and practical. Numerous valuable lessons have been accomplished at all levels. My knowledge in the areas of large scale systems, complex supply chains, simulation-based control, dynamics in supply chain, modeling of dynamic large scale systems, fidelity of models and dynamic decision making has been significantly enhanced by the help of this

dissertation. This dissertation also has helped me advance my learning in the use of simulation techniques such as distributed simulation and simulation based-control and design of experiments to analyze a system. Substantial experience has been gained in terms of general research methodologies and report writing.

It is worthwhile to note that the Finagle's Law of Dynamic Negatives which is stated as "Anything that can go wrong, will—at the worst possible moment" and a Anonymous Quote which is stated as "Energy accomplishes more than genius" held true during several phases of the dissertation.

APPENDIX A

Proof to show $\hat{\mu}_1$ is the unbiased estimate of μ

Theorem: $\hat{\mu}_1$ is the unbiased estimate of μ.

Proof:

$$E_p[\hat{\mu}_1] = E_p[\frac{1}{N_s}\sum_{i=1}^{N_s} h(\mathbf{x}^{\mathrm{i}})] = \frac{E_p[\sum_{i=1}^{N_s} h(\mathbf{x}^{\mathrm{i}})]}{N_s} = \frac{N_s E_p[h(\mathbf{x})]}{N_s} = \mu$$

Since $E_p[\hat{\mu}_1] = \mu,\ \hat{\mu}_1$ is an unbiased estimate of μ. The proof is complete.

Proof to show $\hat{\mu}_2$ is the unbiased estimate of μ

Theorem: Proof to show $\hat{\mu}_2$ is the unbiased estimate of μ.

Proof:

$$E_f[\hat{\mu}_2] = E_f\left[\sum_{1}^{N_s} h(\mathbf{x}^i)\, w(\mathbf{x}^i)\right] \Big/ N_s = \frac{E_f[\sum_{i=1}^{N_s} h(\mathbf{x}^{\mathrm{i}}) w(\mathbf{x}^i)]}{N_s} = \frac{N_s E_f[h(\mathbf{x})w(\mathbf{x})]}{N_s} = E_p[h(\mathbf{x})] = \mu$$

Since $E_f[\hat{\mu}_2] = \mu,\ \hat{\mu}_2$ is an unbiased estimate of μ. The proof is complete.

Asymptotic properties of σ_h

Theorem: Assuming that the measurement errors are independent and identically distributed, if $\sigma_h > 0$, then $P(N_s < \infty) = 1$ and $lim_{\sigma_h \to 0} P(\mu + \sigma_h \leq \bar{x} \leq \mu + \sigma_h) \geq 1 - \alpha$.

Proof: The proof of this theorem is analogous the one given in Anscombe (1952) for, and Bayraksan (2005) in developing accelerated sequential procedure for assessing solution quality of stochastic programs.

Given $\sigma_h > 0$,

$$P(N_s = \infty) = P\left(\bigcap_{k=1}^{\infty}\{(N_s = k) > (N_s = k-1)\}\right)$$

$$= P\left(\bigcap_{k=1}^{\infty}\{t^2_{n-1,1-\alpha/2}\frac{s^2}{{\sigma_h}^2} > (N_s = k-1)\}\right)$$

This has a positive probability only if $s^2 \to \infty$ as $k \to \infty$ which is a contradiction.

REFERENCES

ALVARENGA, C., and SCHOENTHALER, R., 2003, A new take on supply chain event management. *Supply Chain Management Review*, March/April, 29-35.

ANDRADOTTIR, S. and BIER, V.M., 2000, Applying Bayesian ideas in simulation, *Simulation Practice and Theory*, **8**(3-4) 253-280.

ANGELOVA, D., and MIHAYLOVA, L., 2006, Joint Target Tracking and Classification with Particle Filtering and Mixture Kalman Filtering Using Kinematic Radar Information, *Digital Signal Processing, Elsevier Science*, 16 (2), 180-204.

ANDREEV, K., and RÄCKE, H., 2004, Balanced graph partitioning. *Proceedings of the 16th Annual Association for Computing Machinery Symposium on Parallelism in Algorithms and Architectures*, 120 – 124.

ARULAMPALAM, S., MASKELL, S., GORDON, N., and CLAPP, T., 2002, A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking, *IEEE Transactions on Signal Processing*, **50** (2), 174-188.

AYANI, R. and RAJAEI, H., 1992, Parallel simulation using conservative time windows, *In Proceedings of the 1992 Winter Simulation Conference*.

BAHMANI, O. and BROWN, F., 2004, Kalman filter approach to estimate the demand for international reserves, *Applied Economics*, **36**(15), 1655-1668.

BEAMON, B.M., 1998, Supply chain design and analysis: Models and methods, *International Journal of Production Economics*, **55**, 281-294.

BEAMON, B.M., 1999, Measuring supply chain performance. *International Journal of Operations and Production Management*, **19**(3/4), 275–292.

BEAMON, B.M. and CHEN, V.C.P., 2001, Performance analysis of conjoined supply chains, *International Journal of Production Research*, **39**, 3195-3218.

BHASKARAN, S., 1998, Simulation analysis of a manufacturing supply chain, Decision *Sciences*, **29**(3), 633-657.

BISWAS, S. and NARAHARI, Y., 2004, Object oriented modeling and decision support for supply chains, *European Journal of Operational Research*, **153**(3), 704-726.

BOKHARI, S.H., 1988, Partitioning problems in distributed parallel, pipelined, and computing, *IEEE Transactions on Computers*, **31**(1), 48-58.

BOLVIKEN, E., ACKLAM, P. J., CHRISTOPHERSEN, N., STORDAL, J. M, 2001, Monte Carlo filters for nonlinear state estimation, *Automatica*, **37**, 177**–**183, 2001.

BOUKERCHE, A., 2002, An adaptive partitioning algorithm for distributed discrete event simulation systems, *Journal of Parallel and Distributed Computing*, **62**, 1454–1475.

BRASNETT. P. MIHAYLOVA, L., BULL, D., and CANAGARAJAH, N., 2007, Sequential Monte Carlo Tracking by Fusing Multiple Cues in Video Sequences, *Image and Vision Computing, Elsevier Science*, 28 (1), 1217-1227.

CAMM, J.D., CHORMAN, T.E., DILL, F.A., EVANS, J.R., SWEENEY, D.J., and WEGRYN, G.W., 1997, Blending OR/MS, judgment, and GIS: Restructuring P&G's supply chain, *Interfaces*, **27**(1), 128-142.

CARNAHAN, J.C. and REYNOLDS, P.F., 2006, Requirements for DDDAS flexible point support, *In Proceedings of the 2006 Winter Simulation Conference*, 2101-2108.

CASARIN, R., and SARTORE, D., 2008, Matrix-State Particle Filter for Wishart Stochastic Volatility Processes, Discussion Paper (No: 0816), Dipartimento di Scienze Economiche, Università degli Studi di Brescia, Italy.

CASSADY, C. R., BOWDEN, R. O., LIEW, L., and POHL, E. A., 2000, Combining preventive maintenance and statistical process control: A preliminary investigation, *IIE Transactions*, **32**, 471–478.

CELIK, N., LEE, S., VASUDEVAN, K., and SON, Y., 2010, DDDAS-based multi-fidelity simulation framework for supply chain systems, *IIE Transactions on Operations Engineering*, **42**(5), 325–341.

CELIK, N., MAZHARI, E., KAZEMI, O., CANBY, J., SARFARE, P., ALOTAIBI, M., and SON, Y., Automatic partitioning of large scale simulation in grid computing for run time reduction, *International Journal of Operations Research and Information Systems*, 1(2), 2010, 64-90.

CELIK, N., and SON, Y., April 2010, Sequential Monte Carlo-based Fidelity Selection in Dynamic-Data-Driven Adaptive Multi-Scale Simulations, *International Journal of Production Research*, under review.

CHATFIELD, D.C., HAYYA, J.C., and HARRISON, T.P., 2007, A multi-formalism architecture for agent-based, order-centric supply chain simulation, *Simulation Modeling Practice and Theory*, **15**, 153-174.

CHEN, H. C., CHEN, C.H., DAI, L., and YUCESAN, E., 1997, New development of optimal computing budget allocation for discrete event simulation, *In Proceedings of the 1997 Winter Simulation Conference*, 334-341.

CHEN, F., DREZNER, Z., RYAN, J., and SIMCHI-LEVI, D., 2000, Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting, lead times, and information, *Management Science*, **46**(3), 436–443.

CHEN, C.H., HE, D., FU, M., and LEE L.H., 2008, Efficient simulation budget allocation for selecting an optimal subset, *INFORMS Journal on Computing*, **20** (4), 579-595.

CHEUNG, Y.W., 1993, Exchange rate risk premiums, *Journal of International Money and Finance*, **12**, 182-194.

CHICK, S.E., 2006, Bayesian ideas and discrete event simulation: Why, what and how? *In Proceedings of the 2006 Winter Simulation Conference*.

CHOI, J., REALFF, M.J., and LEE, J.H., 2006, Approximate dynamic programming: Application to process supply chain management, *Aiche Journal*, **52**(7), 2473-2485.

COHEN, M.A. and LEE, H.L., 1988, Strategic analysis of integrated production distribution systems: Models and methods, *Operations Research*, **6**(2), 216-228.

COMMANDEUR, J.J.F. and JAN KOOPMAN, S., 2007, An introduction to state space time series analysis, Oxford University Press.

CORMEN, T., LEISERSON, C., RIVEST, R., and STEIN, C., 2001, Introduction to algorithms. Second Edition, MIT Press and McGraw-Hill.

COTTENIER, T., and ELRAD, T., 2004, Layers of collaboration aspects for pervasive computing, In *Proceedings of the Workshop on Building Software for Pervasive Computing at the 19th Annual Conference on Object-Oriented Programming,*

Systems, Languages, and Applications (OOPSLA'04), Vancouver, Canada, October 2004.

CRISAN, D. and DOUCET, A., 2002, A survey of convergence results on particle filtering methods for practitioners, IEEE Transactions on Signal Processing, 50(3), 736-746.

DAREMA, F. (2004) Dynamic data driven applications system: A new paradigm for application simulations and measurements, *International Conference on Computational Science*, 662-669.

DOUGLAS, C.C., SHANNON, C.E., EFENDIEV, Y., EWING, R., GINTING, V., LAZAROV, R., COLE, M.J., JONES, G., JOHNSON, C.R., and SIMPSON, J., 2004, A note on data driven contaminant simulation, *Lecture Notes in Computer Science*, 701-708.

DAVID, V., FRABOUL, CH., ROUSSELOT, J.. and SIRON, P., 1992, Partitioning and mapping communication graphs on a modular reconfigurable parallel architecture. *Lecture Notes in Computer Science, 634*, 43-48.

DISNEY, S.M., LAMBRECHT, M., TOWILL, D.R., and VAN DE VELDE W., 2008, The value of coordination in a two-echelon supply chain, *IIE Transactions*, **40**(3), 341-355.

DOUCET, A., GODSILL, S., and ANDRIEU, C., 2000, On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing, **10**, 197-208.

DRAKE, G. and SMITH, J., 1996, Simulation system for real time planning scheduling and control, *In Proceedings of the 1996 Winter Simulation Conference*.

DREYER, H.C., ALFNES, E., STRANDHAGEN, J.O., and THOMASSEN, M.K., 2009, Global supply chain control systems: A conceptual framework for the global control centre, *Production Planning and Control*, **20**(2), 147-157.

FLURY, T. and SHEPHARD, N., 2008, Bayesian inference based only on simulated likelihood: Particle filter analysis of dynamic economic models, Report (No: 413), Department of Economics, University of Oxford, UK.

FORGET, P., D'AMOURS, S., and FRAYRET, J.M., 2008, Multi-behavior agent model for planning in supply chains: An application to the lumber industry, *Robotics and Computer-Integrated Manufacturing*, **24**(5), 664-679.

FUJIMOTO, R., 2000, Parallel and distributed simulation systems (New York: John Wiley & Sons).

GANESHAN, R. and HARRISON, T. P., 1993, An introduction to supply chain management. Available online via <http://silmaril.smeal.psu.edu/misc/ supply_chain_intro.html> [accessed May 2, 2010].

GAUR, V., GILONI, A., and SESHADRI, S., 2005, Information sharing in a supply chain under ARMA demand, *Management Science*, **51**(6), 961–969.

GEHRIG, T., NICKEL, K., EKENEL, H.K., KLEE, U., and MCDONOUGH, J., 2005, Kalman filters for audio-video source localization, *2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 16-19, New Paltz, NY.

GEOFFRION, A.M. and GRAVES, G.W., 1984, Multicommodity distribution systems design by Bender's decomposition, *Management Science,* 20(5), 822-824.

GENTLE, J. E., 2002, Random Number Generation and Monte Carlo, 2nd ed., Berlin: Springer-Verlag.

GODSILL, S.J. and RAYNER, P J.W., 1998, Robust reconstruction and analysis of autoregressive signals in impulsive noise using the Gibbs sampler, *IEEE Transactions on Speech and Audio Processing*, **6**(4), 352-372.

GORDON, N.J., HARRISON, D.J., and SMITH, A.F.M., 1993, Novel-approach to nonlinear non-gaussian Bayesian state estimation, IEEE Proceedings of Radar and Signal Processing, **140**(2), 107-113.

HARLAND, C.H., 1996, Supply chain management: Relationships, chains and networks, *British Journal of Management*, **7**(1), S63–S80.

HENNET, J.C. and ARDA, Y., 2007, Supply chain coordination: A game-theory approach, *Engineering Applications of Artificial Intelligence*, **21**(3) pp. 399-405.

IANNONE, R., MIRANDA, S. and RIEMMA, S., 2007, Supply chain distributed simulation: An efficient architecture for multi-model synchronization, *Simulation Modelling Practice and Theory*, **15**(3), 221-236.

ITO, T. and ABADI, S.M.M.J., 2002, Agent-based material handling and inventory planning in warehouse, *Journal of Intelligent Manufacturing*, **13**(3), 201-210.

JAZWINSKI, A. H., 1970, Stochastic processes and filtering theory, New York : Academic Press.

JEFFERSON, D., 1985, Virtual time, *Association for Computing Machinery Transactions on Programming Languages and Systems*, **7**, 404–425.

JENSEN, V. F., 2001, Bayesian networks and decision graphs, Springer.

JULIER, S., JEFFERY K., and HUGH, F., 1995, A new approach for filtering nonlinear systems, *In Proceedings of the 1995 American Control Conference*, 1628-1632.

KANDEPU, R., FOSS B., and IMSLAND L., 2008, Applying the unscented Kalman filter for nonlinear state estimation, *Journal of Process Control*, **18**(7-8), 753-768.

KERBACHE, L. and SMITH, J.M., 2004, Queueing networks and the topological design of supply chain systems, *International Journal of Production Economics*, **91**(3), 251-272.

KIM, K.H., KIM, T.G., and PARK, K.H., 1998, Hierarchical partitioning algorithm for optimistic distributed simulation of DEVS models, *Journal of Systems Architecture*, **44**, 433-455.

KITAGAWA, G., 1996, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics*, **5**, 1-25.

KLINGENER, J.K., 1996, Programming combined discrete-continuous simulation models for performance. *In Proceedings of the 1996 Winter Simulation Conference*, 833-839.

KONAS, P., 1994, Parallel architectural simulations on shared-memory multiprocessors. Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.

KONG, A., LIU, J.S., and WONG, W.H., 1994, Sequential imputations and Bayesian missing data problems, *Journal of American Statistical Association*, 89, 278-288.

KOOP, J. C., 1951, A note on the bias of the ratio estimate, *Bulletin of the* International *Statistical Institute*., 21(2), 141-146.

KOOP, J., 1962, On upper limits to the difference in bias between two ratio estimates, *Metrika*, (5) 1, 145-149.

KPMG REPORT, 2009, The Road to Recovery in the Global Semiconductor Industry.

KUHL, F., WEATHERLY, R., and DAHMANN, J., 1999, Creating computer simulations: An introduction to the high level architecture (New Jersey: Prentice-Hall).

LABARTHE, O., ESPINASSE, B., FERRARINI, A., and MONTREUIL, B., 2007, Toward a methodological framework for agent-based modeling and simulation of supply chains in a mass customization context, *Simulation Modeling Practice and Theory*, **15**, 113-136.

LAMBERT, D.M., COOPER, M.C., and PUGH, J. D., 1998, Supply chain management: implementation issues and research opportunities, *International Journal of Logistics Management*, **9**(2), 1-19.

LANCIONI, R.A., SMITH, M.F., and OLIVA, T.A., 2000, The role of the internet in supply chain management, *Industrial Marketing Management*, **29** (1), 45–56.

LEE, H.L. and BILLINGTON, C., 1993, Materials management in decentralized supply chain, *Operations Research*, **41**, 835-847.

LEE, H.L., PADMANABHAN, V., and WHANG, S., 1997, Information distortion in a supply chain: The bullwhip effect, *Management Science*, **43**(4), 546–558.

LEE, Y.H., CHO, M.K., and KIM, Y.B., 2002, A discrete-continuous combined modelling approach for supply chain simulation, *Simulation*, **78**(5), 321-329.

LEE, D., DONGARRA, J., and RAMAKRISHNA, R.S., 2003, visPerf: Monitoring tool for grid computing. Computational Science — ICCS 2003. Berlin: 1st. Springer.

LEE, S., ZHAO, X., SHENDARKAR, A., VASUDEVAN, K., and SON, Y., 2008, Fully dynamic epoch (fde) time synchronization method for distributed supply chain simulation. *International Journal of Computer Applications in Technology, 31*(3-4), 249-262.

LINSKER, R., 2008, Neural network learning of optimal Kalman prediction and control, *Neural Networks*, **21**(9), 1328-1343.

LIU, J.S., 1996, Metropolized independent sampling with comparison to rejection sampling and importance sampling, *Statistics and Computing*, **6**, 113-119.

LIU, J. and NICOL, D., 2001, Learning not to share, *In Proceedings of the 15th Workshop on Parallel and Distributed Simulation*, 22-31.

LIU, X. and CHIEN, A., 2003, Traffic-based load balance for scalable network emulation, *In Proceedings of the Supercomputing Conference*, 40.

LONGO, F. and MIRABELLI, G., 2008, An advanced supply chain management tool based on modeling and simulation, *Computers and Industrial Engineering*, **54**(3), 570-588.

MACEACHERN, S. N., CLYDE, M., and LIU, J. S., 1999, Sequential importance sampling for nonparametric Bayes models: The next generation, *Canadian J. Statist*., **27**, 251–267.

MADHUSUDAN, T. and SON, Y.J., 2005, A simulation-based approach for dynamic process management at web service platforms, *Computers and Industrial Engineering*, **49**(2), 287-317.

MANDEL, J., CHEN, M., FRANCA, L.P., JOHNS, C., PUHALSKII, A., COEN, J.L., DOUGLAS, C.C., KREMENS, R., VODACEK, A., and ZHAO, W., 2004, A note on dynamic data driven wildfire modeling, *Lecture Notes in Computer Science*, 725-731.

MARKOV, A.A., 1971, Extension of the limit theorems of probability theory to a sum of variables connected in a chain, reprinted in Appendix B of: R. Howard. Dynamic Probabilistic Systems, volume 1: Markov Chains (John Wiley and Sons).

MARTELLO, S. AND TOTH, P., 1990, *Knapsack problems: Algorithms and computer implementations*, John Wiley & Sons, Chichester, UK.

MARTINEZ-ESPLA, J.J., MARTINEZ-MARIN, T., and LOPEZ-SANCHEZ, J.M., 2007, Introduction of a grid-based filter approach for InSAR phase filtering and unwrapping, *Geoscience and Remote Sensing Symposium*, 4497-4500.

MCGINNIS, L., SHENG, X., and KE, W., 2005, Evaluate simulation design alternatives for large scale manufacturing systems, *IEEE International Symposium on Semiconductor Manufacturing*, 344- 347.

MEYN, S.P. and TWEEDIE, R.L., 2009, Markov Chains and Stochastic Stability, 2nd edition, Cambridge University Press (London: Springer-Verlag).

MIDDLETON, R.H. and GOODWIN, G.C., 1988, Adaptive control of time-varying linear systems, *IEEE Transactions on Automatic Control*, **33**, 150–155.

MIRHASSANI, S.A., LUCAS, C., MITRA, G., MESSINA, E., and POOJARI, C.A., 2000, Computational solution of capacity planning models under uncertainty, *Parallel Computing*, **26**(5), 511-538 .

MUHLICH, M., 2003, Particle filters an overview, JWGoethe-Universitat Frankfurt, Filter-Workshop, Bucuresti 2003.

NAMEKAWA, M., SATOH, A., MORI, H., YIKAI, K., and NAKANISHI, T., 1999, Clock synchronization algorithm for parallel road-traffic simulation system in a wide area, *Mathematics and Computers in Simulation*, **48**, 351-359.

NICOL, D.M., 1993, The cost of conservative synchronization in parallel discrete event simulations, *Journal of the Association for Computing Machinery*, **40**(2), 704-715.

NICULESCU, S.I., 2001, Delay effects on stability: A robust approach, (Springer, Berlin).

OEHLERT, G. W., 1998, Faster adaptive importance sampling in low dimensions, *J. Comput. Graph. Statist.*, **7**, 158**–**174.

OMARI, T., HOSSEINI, S., and VAIRAVAN, K., 2004, Traveling token for dynamic load balancing. *In Proceedings of the Network Computing and Applications, Third IEEE International Symposium*, 329-332.

OUYANG, Y.F., 2007, The effect of information sharing on supply chain stability and the bullwhip effect, *European Journal of Operations Research*, **182**(3), 1107-1121.

OZBEK, L. and OZLALE, U., 2005, Employing the extended Kalman filter in measuring the output gap, *Journal of Economic Dynamics and Control*, **29**(9), 1611-1622.

PATRIKALAKIS, N.M., MCCARTHY, J.J., ROBINSON, A.R., SCHMIDT, H., EVANGELINOS, C., HALEY, P.J., LALIS, S., LERMUSIAUX, P.F.J., TIAN, R., LESLIE, W.G., and CHO, W., 2004, Towards a dynamic data driven system for rapid adaptive interdisciplinary ocean forecasting, invited paper in dynamic data-driven application systems, Darema, F., editor. Kluwer Academic Publishers, Amsterdam, 2004, *to appear*.

PARNAS, D.L., 1979, Designing software for ease of extension and contraction, *IEEE Transactions on Software Engineering*, **5**, 128-138.

PETROVIC, D., 2001, Simulation of supply chain behaviour and performance in an uncertain environment, *International Journal of Production Economics*, **71**(1-3), 429-438.

POURRET, O., NAIM, P., and MARCOT, B., 2008, Bayesian networks: A practical guide to applications (UK: Wiley).

RABELO, L., HELAL, M., JONES, A., MIN, J., SON, Y., and DESHMUKH, A., 2003, A hybrid approach to manufacturing enterprise simulation, *In Proceedings of the 2003 Winter Simulation Conference*, 1125- 1133.

RATHORE, A., BALARAMAN, B., ZHAO, X., VENKATESWARAN, J., SON, Y., and WYSK, R., 2005, Development and benchmarking of an epoch time synchronization method for distributed simulation, *Journal of Manufacturing Systems*, **24**(2), 69-78.

RIDDICK, F., and MCLEAN, C., 2000, The IMS mission architecture for distributed manufacturing simulation, *In Proceedings of the 2000 Winter Simulation Conference*.

RIDGEWAY, G. and MADIGAN, D., 2003, A sequential Monte Carlo method for Bayesian analysis of massive datasets, *Data Mining and Knowledge Discovery*, **7**(3), 301-319.

RISTIC, B., ARULAMPALAM, M., and GORDON, A., 2004, Beyond Kalman Filters: Particle Filters for Target Tracking, Artech House.

RUPP, T.M. and RISTIC, M., 2000, Fine planning for supply chains in semiconductor manufacture, *Journal of Materials Processing Technology*, **107**(1-3), 390-397.

SAMADDAR, S., NARGUNDKAR, S.A., and DALEYA, M., 2006, Inter-organizational information sharing: The role of supply network configuration and partner goal congruence, *European Journal of Operations Research*, **174**(2), 744-76.5.

SCHULZ, D., and BURGARD W., 2001, Probabilistic state estimation of dynamic objects with a moving mobile robot, *Robotics and Autonomous Systems*, 34 (2-3).

SCHWARTZ, J.D., WANG, W.L., and RIVERA, D.E., 2006, Simulation-based optimization of process control policies for inventory management in supply chains, *Automatica*, **42**(8), 1311-1320.

SHAPIRO, J., 2000, Modelling supply chain (Duxbury Press).

SMITH, J.S., WYSK, R.A., STURROK, D.T., RAMASWAMY, S.E., SMITH, G.D., and JOSHI, S.B., 1994, Discrete-event simulation for shop floor control, *In Proceedings of the 1994 Winter Simulation Conference*, 962-969.

SON, Y.J. and WYSK, R.A., 2001, Automatic simulation model generation for simulation-based, real-time shop floor control, *Computers in Industry*, **45**(3), 291-308.

SON, Y.J., JOSHI, S.B., WYSK, R.A., and SMITH, J.S., 2002, Simulation-based shop floor control, *Journal of Manufacturing Systems*, **21**(5), 380-394.

SON, Y., WYSK, R., and JONES, A., 2003, Simulation-based shop floor control: formal model, model generation and control interface, *IIE Transactions*, **35** (5), 29-48.

SOULE, L.P., 1992, Parallel logic simulation: an evaluation of centralized-time and distributed-time algorithms. Ph.D. Thesis, Stanford University.

SPENS, K. and BASK, A., 2002, Developing a framework for supply chain management, *International Journal of Logistics Management*, **13**(1), 73–88.

STEVENSON, M., 1994, The store to end all stores, *Canadian Business Review*, **67**(5), 20–26.

SWAMINATHAN, J. M., SMITH, S. F., and SADEH, N. M., 1995, Modelling the dynamics of supply chains, The Robotics Institute, Carnegie Mellon University, Pittsburg, PA.

TAKEUCHI, H., KONDO, T., KOYAMA, Y., NAKAJIMA, J., ICHIKAWA, R., SEKIDO, M., KAWAI, E., OSAKI, H., KIMURA, M., and KUBOKI, H., 2004, A grid computing vlbi system for real-time monitoring of the earth orientation parameters. *Copernicus online system & Information system*.

TAYLOR, S., SUDRA, R., JANAHAN, T., TAN, G., and LADBROOK, J., 2002, GRIDS-SCF: An infrastructure for distributed supply chain simulation, *Simulation*, **78**(5), 312-320.

TOWILL, D.R., 1991, Supply chain dynamics, *International Journal of Computer Integrated Manufacturing*, **4** (4), 197-208.

TOWILL, D.R. and Del Vecchio, A., 1994, The application of filter theory to the study of supply chain dynamics, *Production Planning and Control*, **5**(1), 82-96.

TSIAKIS, P., SHAH, N., and PANTELIDES, C.C., 2001, Design of multi-echelon supply chain networks under demand uncertainty. *Industrial & Engineering Chemistry Research*, **40**(16), 3585-3604.

VAHDAT, A., WALSH, K., MAHADEVAN, P., KOSTIC, D., CHASE, J., and BECKER, D., 2002, Scalability and accuracy in a large-scale network emulator, *In Proceedings of the 5th Symposium on Operating Systems Design and Implementation*.

VAN DE VEN, A. and FERRY, D.L., 1980, Measuring and assessing organizations (New York: Wiley Series).

VAN DER ZEE, D.J., 2003, Modeling control in manufacturing simulation, *In Proceedings of the 2003 Winter Simulation Conference*, 791–798.

VAN DER ZEE, D.J. and VAN DER VORST, J.G.A.J, 2005, A modeling framework for supply chain simulation: Opportunities for improved decision making, *Decision Sciences*, **36**(1), 65-95.

VARDANEGA, F., and MAZIERO, C., 2001, A generic rollback manager for optimistic HLA simulations, *Transactions of the Society for Computer Simulation International*, **18**(2), 110-115.

VENKATESWARAN, J., and SON, Y., 2004, Design and development of a prototype distributed simulation for evaluation of supply chains, *International Journal of Industrial Engineering*, **11**(2), 151-160.

VENKATESWARAN, J. and SON, Y., 2004, Impact of modeling approximations in supply chain analysis, *International Journal of Production Research*, **42**(15), 2971 - 2992.

VENKATESWARAN, J. and SON, Y., 2005, Hybrid system dynamic: Discrete-event simulation based architecture for hierarchical production planning, *International Journal of Production Research*, **43**(20), 4397 - 4429.

VO, B.N., SINGH, S., and DOUCET, A., 2005, Sequential Monte Carlo methods for multi-target filtering with random finite sets, *IEEE Transactions on Aerospace and Electronic Systems*, **41**(4), 1224-1245.

WAGNER, B.J., 1995, Sampling design methods for groundwater modeling under uncertainty, *Water Resources Research*, **31**(10), 2581-2591.

WALTER, K.D., 2004, SSV embedded systems. *Embedded Grid Computing,* Unpublished.

WAN, E. and VAN DER MERWE, R., 2001, The unscented Kalman filter, In Kalman Filtering and Neural Networks, Wiley Publishing.

WANG X., TURNER S.J., LOW, M.Y.H., and Gan, B.P., 2005, Optimistic synchronization in hla-based distributed simulation, *Simulation*, **81**, 279-293.

WEISSMAN, J. and GRIMSHAW, A., 1994, Network partitioning of data parallel computations, Technical Report: CS-94-17, 149-156.

WENG, S., KUO C., and TU, S., 2006, Video object tracking using adaptive Kalman filter, *Journal of Visual Communication and Image Representation*, **17**(6), 1190-1208.

WEST, M. and HARRISON, J., 1997, Bayesian forecasting and dynamic models, in Springer Series in Statistics, 2nd ed. New York: Springer-Verlag.

WHEELER, D. J. and CHAMBERS, D. S., 1992, *Understanding statistical process control*, 2nd edition, SPC Press, Knoxville, TN.

WIKNER, J., TOWILL, D.R., and NAIM, M., 1991, Smoothing supply chain dynamics. *International Journal of Production Economics*, **22**, 231-248.

WU, S.D., and WYSK, R.A., 1989, An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing, *International Journal of Production Research*, **27**(9): 1603--1623.

WYSK, R., HALL, D., PEGDEN, D., SON, Y., MCGINNIS, L., and ZHOU, C., 2004, ITR: Collaborative research: as times go on-adaptive and scalable time synchronization mechanism for federations of distributed simulations. *Unpublished Report*.

XIA, Z. and LIN, K.C., 2003, Distributed combined discrete-continuous simulation for multiple MAVs motion analysis, *Simulation Series*, **35**(1), 162-170.

XIA, H., DING, Y., and MALLICK, B., 2008, Bayesian hierarchical model for integrating multi-resolution metrology data, submitted to *Technometrics*.

XIE, M., GOH, T. N., and RANJAN, P., 2002, Some effective control chart procedures for reliability monitoring, *Reliability Engineering and System Safety*, **77**, 143–150.

XU, S, 2006, Optimistic-conservative synchronization in distributed factory simulation. *In Proceedings of the 2006 Winter Simulation Conference (WSC 2006),* 1069 -1074.

XU, X. and LI, B., 2007, Adaptive Rao-Blackwellised particle filter and its evaluation for tracking in surveillance, *IEEE Transactions on Image Processing*, 16 (3), 838-849.

ZHANG, Y. and IOANNOU, P.A., 1996, Adaptive control of linear time-varying systems, *In IEEE 35th Conference on Decision and Control*, 837–842, Kobe-Japan.

ZHOU, H. and BENTON, W.C., 2007, Supply chain practice and information sharing, *Journal of Operations Management*, **25**(6), 1348-1365.