BOTTLENECK DETECTION AND MITIGATION IN SERIAL PRODUCTION SYSTEMS

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Abishek Ramesh

Certificate of Approval

_____	_____
John L. Evans Associate Professor Industrial and Systems Engineering	Emmett J. Lodree Assistant Professor Industrial and Systems Engineering
_____	_____
Jeffrey S. Smith, Chair Professor Industrial and Systems Engineering	Joe F. Pittman Interim Dean Graduate School

BOTTLENECK DETECTION AND MITIGATION IN SERIAL PRODUCTION SYSTEMS

Abishek Ramesh

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 15, 2006

BOTTLENECK DETECTION AND MITIGATION IN SERIAL PRODUCTION SYSTEMS

Abishek Ramesh

Permission is granted to Auburn University to make copies of this thesis at its discretion, upon request of individuals or institutions and at their expense. The author reserves all publication rights.

_____
Signature of Author

_____
Date of Graduation

THESIS ABSTRACT

BOTTLENECK DETECTION AND MITIGATION IN SERIAL PRODUCTION SYSTEMS

Abishek Ramesh

Master of Science, December 15, 2006
(B.S.M.E., Kumaraguru College of Technology, Bharathiyar University, 2004)

138 Typed Pages

Directed by Jeffrey S. Smith

A variety of analytical models have been proposed to model and analyze serial manufacturing systems. Most analytical models, however, make simplifying assumptions in order to remain mathematically tractable. These analytical formulations do not, however, model the underlying real-world system accurately. Discrete event simulation is one of the primary tools, which provides decision support by capturing the working of complex systems at level of detail and accuracy needed. This thesis analyzes the working of serial production lines characterized by capacitated buffers, stochastic processing times, unreliable machines, rework loops, maintenance and operator issues with the help of discrete event simulation to ascertain its throughput. However, in the past researchers have not been excited about the time it takes to build a complete simulation model. In an

effort to fast track the process of model development, a VBA project is undertaken which will dynamically generate a simulation model in Arena 7.01 from an Excel template. We also propose an algorithm which will automatically detect the bottleneck of a serial manufacturing system and provide recommendations to the analyst. These include reallocation of operators, addition of buffers or parallel resources with an objective of increasing the throughput of the system with due economic consideration. Simulation studies are undertaken on different serial manufacturing lines to illustrate the effectiveness of the techniques developed.

ACKNOWLEDGEMENTS

I would like to thank Dr. Jeffrey S. Smith for his invaluable advice and support through every step of this research and Dr. John L. Evans and Dr. Emmett J. Lodree for their kind consent to serve on the thesis committee. I would also like to thank my wonderful family and friends for their support. Finally, I would like to thank Skylab Gupta, who has been a friend and a mentor throughout my time at Auburn University.

Style manual or journal used Computers and Industrial Engineering.

Computer software used Microsoft Word 2003.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Process improvement programs and reengineering have taken a front seat due to a significant increase in competition amongst the giants in manufacturing industry (Snodgrass, 1994). These programs require an accurate estimate of performance metrics such as throughput capacity of a given layout to justify projects and make valid comparisons between various restructuring options which would cost millions of dollars to implement. The focus of this thesis is on the analysis of serial production lines which are characterized by capacitated buffers, stochastic processing times, unreliable machines, rework loops, maintenance and operator issues. This constitutes a complex manufacturing system for which we have adopted discrete event simulation as a tool to predict the performance metrics. As a part of this thesis, a VBA project was undertaken in an effort to automate the process of building a simulation model in Arena 7.01 from an Excel template. This will enable line managers with the working knowledge of simulation develop and modify models with minimal efforts and cut down on repetitive model building and modification time to a great extent. We have also developed an algorithm which will automatically detect the bottleneck in a serial production system and suggest appropriate changes to the analyst with an objective of increasing the

throughput of the system. These techniques are embedded in a ten step methodology presented in this thesis.

1.1 Serial Manufacturing Lines

Serial production lines are sequential arrangements of machines designed for a specific product. Products enter the system through the first station, get processed in a sequential order and leave the system as finished products (Cochran and Erol, 2001). The stations are independent of each other (for instance station x failing will have no bearing on when station $x+1$ is going to fail next) and have varying processing times. Buffers might be located at specific points and when a buffer in front of the bottleneck machine is full, the upstream stations are blocked and the downstream stations are starved (Blumenfeld, 1990).

Dallery and Greshwin (1992) classify features and properties of serial manufacturing lines. They are detailed as follows:

a) Synchronous / asynchronous: In synchronous systems all machines start and stop at the same instant and their operating times are assumed to be deterministic. The system automatically indexes at constant intervals. Events like repair and replenishment occur at discretized time intervals. Asynchronous systems have buffers between stations which make them independent of one another as long as the buffers are neither full nor empty. The station begins to work on a new unit as soon as the previous unit is completed.

b) Saturated / non-saturated: Models built on the assumption that the first machine is never starved and the last machine is never blocked are called saturated models.

Saturated models are used to predict the production rate of a system. The concept of having uncertain arrival and departure of parts constitute non-saturated models.

c) Blocking, Starving and decoupling: When a machine ceases to operate, the upstream machines can still pass on parts until the buffers upstream are full and the downstream machines will receive parts until buffers downstream are empty. When the upstream buffers are full, those machines are said to be blocked and when the downstream buffers are empty those machines are said to be starved. When there is no or very small buffering it causes the greatest coupling between machines and there exists least coupling when the buffer sizes are large.

d) Failures: Failures can be classified into time dependent and operation dependent failures. Failures in stations that follow a chronological frequency are classified under time dependent failures. When the failures occurring depend on the number of units processed by a station it is called an operation dependent failure.

e) Operating times and policy: Efforts are made to design the production lines such that the machines in the system have more or less the same production capacity. The reason being the throughput of a line depends on the production rate of the bottleneck machine and the investments made on other expensive machines in the line cannot be economically justified if its production rate is significantly higher. But in the real world scenarios due to the practical constraints this is not always achieved.

Serial production systems in this thesis are depicted as a sequential arrangement of identical/non identical stations which are capable of processing one part at a time or parts in batches. The stations might have inbuilt conveyors, multiple head processing systems and other complex configurations. The processing time of a product might vary

from one machine to the other and can be stochastic. Each station is subjected to failures and exhausts. When a station is in a state of failure/exhaust a technician is seized for repair/replenish operations and the priority with which technicians are seized can also be specified. The stations are connected to one another by means of a conveyor which can be either accumulating or non-accumulating. Capacitated buffers might be placed at various points in the line between stations and the buffer discipline can be FIFO/LIFO. The lines might include parallel "legs" to increase production. In this configuration, the main line is split into parallel lines and these parallel lines merge into the main line after some operations. Some stations might be dedicated to inspect the quality of products at various points in the production line. The production line might also have rework loops where components which don't meet the quality criteria are being worked on and some parts might be scraped (Li, 2005).

In this thesis the effects of capacitated buffers are studied from the perspective of only increasing the production rate and it doesn't take inventory costs into account. Also the products in the buffers are thought of as nonperishable items.

1.2 Discrete Event Simulation

"Discrete-event simulation consists of a collection of techniques that, when applied to the study of a discrete-event dynamical system, generates sequences called sample paths that characterize its behavior" (Fishman, 2001). The term simulation in this thesis refers to discrete-event simulation. Simulation has evolved into a powerful decision support tool for manufacturing industries which is dominated by dynamic and stochastic

variables. A serial production line can be viewed as a system that has resources (machines and operators) arranged in some predetermined order, which processes entities (units). Simulation helps the decision makers to gain insights into such systems to make improvements. By doing so, simulation increases the utilization and productivity of a system and the organization ends up making quality decisions.

Simulation helps answer the 'what if' questions posted by the management to decision makers. Some of the common questions posted are:

- Can target production be met?
- Is a layout change necessary?
- Should the material handling system be changed?
- What should be the size of buffers at various points in the line?
- Which is the bottleneck machine and what's its effect on the production rate?
- Should additional resources be added to the line?
- What is the effect of product mix on the line?

Simulation can provide answers to all of the above questions. With increased computational power, the cost of a simulation study is estimated to be less than 1% of the total amount spent in implementing a design or redesign (McLean, 2001). The general consensus is that simulation is not adopted by most of the manufacturing industries in spite of these advantages (McLean, 2001). This thesis highlights the importance of simulation in the field of manufacturing systems and provides a methodology which will enable managers with basic knowledge of simulation to make effective use of this tool.

1.3 Motivation and Overview of Thesis

This research is an extension of Mukkamala et al. (2003), Mukkamala (2003), Jadhav and Smith (2005) and Jadhav (2005) work. Mukkamala et al. (2003) and Mukkamala (2003) developed a domain specific template for the automated assembly of PCB. Their research focused in building PCM assembly template to analyze serial production PCB assembly lines where a typical machine has an input area, a processing area and an output area. The input area comprises of an input conveyor and an optional capacitated input buffer. Similarly an output buffer comprises of an output conveyor and an optional capacitated output buffer. Jadhav and Smith (2005) and Jadhav (2005) used the templates to build simulation models and predict the throughput of serial production systems. The simulation studies conducted were time consuming and the bottleneck resource detection was done by eyeballing the resource state graph.

In this thesis we have undertaken a VBA project to dynamically generate the simulation model of the underlying system using an Excel template. We have also developed an algorithm which will automatically detect the bottleneck resource and suggest appropriate changes to the analyst in an effort to mitigate the bottleneck resource, thereby improving the throughput of the system. These techniques which fast track the process of a simulation study, have been embedded in a ten step methodology.

The next chapter will review literature relevant to this thesis. Chapter 3 presents the problem statement and the methodology developed. Chapter 4 comprises of case studies which verify the proposed techniques. Chapter 5 contains the conclusion and discusses future research.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews the complexities and assumptions involved in developing analytical models to estimate the throughput of serial production systems present in the literature. This discussion is followed by contrasting the bottleneck detection methods present in the literature to ours. Buffer allocation which plays an important part in enhancing the performance of serial production systems is also discussed in this chapter. Finally, the ideas in literature to develop a simulation interface to expedite the process of model building are presented.

2.1 Analytical Modeling and Evaluation of Serial Production Systems

Cochran and Erol (2001) developed an analytical model for serial production lines to estimate throughput rates, scrap rates and outgoing quality levels by incorporating traffic rate equations. The serial production lines in this analysis are modeled as directed flow networks. The model explicitly differentiates between operation stations and inspection/repair stations. This model can be used for inspection configuration designs and formulation of optimal cost models.

Li (2005) evaluates the throughput of complex serial production system with parallel, rework and scrap operations using the overlapping decomposition technique. The idea behind this methodology is to divide the complex system into a number of serial production systems wherein the last machine of a serial line overlaps with the first machine of the next serial line. They adapt aggregation techniques and the throughput of a system is estimated when the procedure converges. This method is developed under the assumption that the processing rates of all the machines in the system are the same.

Choong and Greshwin (1986) develop a decomposition method to analyze capacitated transfer lines with capacitated buffers and random service times. It is based on a model developed by Greshwin (1983) that approximates a (K-1)-buffer system by K-1 single-buffer systems. Throughput and average buffer levels of the system are calculated by formulating an iterative search algorithm. This model has been verified with the help of several numerical examples and the authors conclude that this approach is viable only if the probability that a machine is starved and blocked at the same time is small. Machine *k* is said to be both blocked and starved, only when machine k_{i-1} is either under repair, starved or is processing a piece and machine k_{i+1} is either under repair, blocked or is processing a piece. The buffer in front of machine *k* is empty while the one in front of machine k_{i+1} is full. According to the author, this probability will be high if there is a huge variation in efficiencies (probability that a station is processing a work piece) and processing rates of the machines. Under such conditions the method may break down.

Burman (1995) investigates flow lines with unreliable machines and develops an analytical decomposition-type approximation technique to estimate throughput of such

systems. “Accelerated Dallery-David-Xie” (ADDX) algorithm was developed as a part of his thesis for solving the decomposition equations developed for flow lines. This analytical technique accounts for non-homogeneous flow, operation dependent failures, unreliable machines and failing buffers. The application of this technique is however limited to flow lines that have deterministic processing times with no rework operations.

Paik et al. (2002) used decomposition and aggregation principles to approximately estimate throughput for finite buffered closed loop production system with unreliable machines and exponentially distributed processing times. They also propose a simple algorithm that predicts the upper bounds of throughput for such systems. The effectiveness of the procedure is illustrated in their paper with the help of extensive computational experiments.

Tempelmeier and Burger (2001) analyze unbalanced serial production systems with finite buffers to estimate throughput of the system by employing an analytical approximation technique. This technique accounts for machine breakdown and defective part production. The time to failure was assumed to be exponentially distributed. In this procedure, M-station-lines are decomposed into M-1 two-station-lines and are analyzed with the help of $GI/G/1/N_{\max}$ queuing model. This model also accounts for simultaneous blocking and starving that Choong and Greshwin (1986) model lacked.

Chen et al. (2003) estimate the value and variance of throughput for serial production systems with unreliable machines using a sample path method. Their method conducts a sensitivity analysis on the mean and variation of throughput with respect to mean up time or mean down time to determine the amount of improvement that can be achieved in the system. It is assumed that a failed machine is immediately taken for

repair. The model assumes that all machines in the system have the same production rate with no inter-stage buffers. The idea of having inspection stations at various points in the line is also not considered in this study and hence we cannot use this technique for our projects.

2.2 Bottleneck Detection

A resource which impedes the performance of a system in the strongest manner is defined as the bottleneck resource (Chiang et al., 2001). In other words, it has the largest impact on reducing the throughput of a system. This section presents techniques available in the literature to detect the bottleneck resource of a system.

Utilization of a machine is defined as the ratio of arrival rate of parts to be processed to effective production rate (Hopp and Spearman, 2000). According to Hopp and Spearman (2000), the percentage of utilization is calculated for each machine in the system and the machine with the largest percentage of utilization is considered to be the bottleneck resource. The author does not discuss about the % of time spent by a machine in blocked state when they illustrate this technique with an example. We contrast this bottleneck detection technique with the one developed as a part of this thesis in Chapter 4.

Roser et al. (2001) categorize all possible states of a machine into two groups: active and inactive states. A machine is said to be in active state when the current state of the machine is aimed at improving the system throughput. Starving, blocking and waiting for services are classified under inactive states. In this technique, the duration of a

machine remaining in an active state without being interrupted by an inactive state is measured. The machine which has the longest active period is detected as the bottleneck. They illustrate this method by simulating a production line which has eight machines with three capacitated buffers between machines and contrast it with the conventional percentage utilization approach. They conclude that their method detected the bottleneck with a higher level of confidence than the conventional percentage utilization approach. This approach assumes that there is minimum or no operator interference in the system which may not be true for all systems. Another drawback of this approach is that by considering only the longest active period it implicitly assumes that the machines in the system are highly reliable. In this thesis a methodology for detecting the bottleneck is proposed which can do away without these assumptions.

Law and Kelton (2000) considers bottleneck machine to be one with the longest waiting time in queue for systems with unlimited buffer sizes. The waiting time in queue for a machine increases as the length of the queue increases by Little's law. But every machine in the real world will have a finite buffer size which sets an upper bound to the wait in queue. And also the machines might have different buffer sizes. These factors might compromise the accuracy of identifying the actual bottleneck machine (Roser et al., 2001).

Chiang et al. (2001) use an aggregation procedure to analyze the performance of Markovian lines with different cycle times and develop a method for cycle time bottleneck identification. This procedure is illustrated with the help of a case study and it is concluded that the probabilities of machine blockages and starvation play a critical role in bottleneck identification. Similar results are observed in this thesis.

Downtime Bottlenecks (DT-BN) were developed by Chiang et al. (2000) to identify bottlenecks based on probabilities of a machine being blocked and starved. The input parameters for the algorithm are: average uptime, downtime, frequency of blockages and starvations. This analytical approach can be applied to only Markovian lines.

Lawrence and Buss (1994) analyze the phenomenon of shifting production bottlenecks by identifying bottleneck machine as one which has the maximal queue length. They propose a bottleneck shiftiness method to tests policies like chasing short-run bottlenecks, increasing capacity at long-run bottlenecks and increasing capacity at non-bottleneck work centers. They conduct simulation studies and conclude that the policy of adding capacity at non-bottleneck work centers will reduce the bottleneck shiftiness to a great extent but the degree of system performance improvement is smaller than those when the other two policies are adopted. The method they adopt for identifying the bottleneck might fail to identify the true bottleneck machine when capacitated buffers are used. Also, they have not considered the costs of adding extra capacity to non-bottleneck work centers to minimize the bottleneck shifting.

Moss and Yu (1999) use Lawrence and Buss (1994) methods to access the factors which will have the greatest influence on bottleneck shiftiness by using multiple regression. Their study concludes that job arrival rate, processing time at bottleneck machine and size of shop, have the greatest impact on bottleneck shiftiness and that the managers could use these parameters to make better capacity decisions. The results are however limited to only FCFS queuing disciplines.

2.3 Buffer Allocation

Enginarlar et al. (2002), present an analytical approach to find the smallest amount of buffering required in a serial production line with unreliable machines necessary to meet the target production rate. To simplify the analytical model development they assume that the machine up times follow exponential distributions. In their analysis they found that a large amount of buffering is required for those machines which have high coefficient of variation of downtime. Their findings also state that the level of buffering doesn't explicitly depend on the average up time of a machine but its efficiency, which is the ratio of average up time to average up and down times of a machine. These findings are based on the assumption that all the machines in the system are identical. According to Enginarlar et al. (2002), these results can be extended to non-identical machines only by making a critical assumption that the efficiency of all the machines in the system are roughly the same. This assumption might not hold true for systems with highly variable processing times.

Yamashita and Altiok (1998) investigated the minimum total buffer allocation required to meet the throughput in production lines with phase-type processing times. They have developed a dynamic programming algorithm which allocates the minimum total buffer space and estimates the throughput of the system at every stage by employing decomposition technique. They illustrate their methodology with a numerical example and conclude that when variability of the processing times increases the throughput of the system decreases, provided the mean processing time and buffer configuration remain constant.

Shi and Men (2003) incorporate Tabu search heuristic into the Nested Partitions Framework and develop a hybrid algorithm in an effort to optimally allocate buffers in large production systems. The model accommodates for different processing rates between machines and failure of machines. The authors claim that their algorithm is robust and can reduce the search effort for buffer allocation problems to a great extent. An efficient algorithm to find the optimal allocation of a fixed stock of buffer capacity for serial production systems was developed by Harris and Powell (1999). A simplex search procedure is embedded in the algorithm to ascertain the search direction which is determined by the best and the worst of current candidate solutions. This algorithm also employs simulation to estimate the throughput for each candidate configuration. The drawback of this algorithm is that it assumes that there are no unreliable machines in the system.

Powell and Pyke (1998) study the issue of optimally deploying limited buffer capacity in short unbalanced assembly lines. They develop simple heuristic rules for unbalanced assembly lines with random processing times. Their study shows that mean and standard deviation of processing time distributions play a vital role in the selection of optimal location of the first buffer. Their heuristic was tested for assembly lines with 2, 3 and 4 stations and it was found to successfully select the optimal location of the first buffer more than 90% of the time. This heuristic doesn't take machines failures into account and it has not been tested for large complex systems.

Papadopoulos and Vidalis (2001) investigate the buffer allocation problem for unreliable, unbalanced short production lines consisting up to six machines. The model assumes that the service and repair times follow Erlang-K distribution and time to failure

follow exponential distribution. A good initial buffer vector is found by the algorithm developed by them which makes use of parameters like mean service, repair and failure rates. The algorithm employs a sectioning search method to search for optimal buffer allocations. Their algorithm was 97% successful in identifying the optimal buffer allocation for 373 experiments conducted. They provide no evidence in their paper to show that this algorithm is effective for production lines of any size.

Roser et al. (2003) develop a prediction model which uses a single simulation run to determine the effect of increased buffer capacity on the system performance. This method is applicable to both balanced (production rate of the machines are equal) and unbalanced production systems. It's a two step methodology wherein the first step is to determine the causes for starving and blocking of all the machines thereby evaluating different buffer location options to reduce the idle time. The next step is to evaluate the improvement in performance metrics due to increases in buffer sizes by analyzing the data from the simulation report and the previous step. The prediction model has been fully automated for easier handling. The model cannot handle systems with machine breakdowns.

2.4 Simulation Interface

A simulation environment was developed to simplify the process of building a simulation model for high volume electronics manufacturing systems by Farrington et al. (1995). The environment can be decomposed into three basic elements, which provide increasing level of modeling capability and it reduces effort involved to build a

simulation model. The elements are interconnected through data transfer links and feedback loops. The problem definer element using a graphical interface develops the initial definition of a system. Static analysis is conducted on the system by the static analyzer element. The code generator element generates the simulation code for simulation packages. The Excel template developed as a part of this thesis performs a similar task described in this paper.

Doss and Ulgen (2004) present the idea of building application specific models using various simulation software engines in an effort to reduce simulation modeling efforts and provide a continuous improvement tool for the companies. These ideas from the literature have taken the shape of the Excel template described in the next chapter and it is embedded in a ten step methodology proposed in this thesis in an effort to simplify repetitive model building and modification time.

2.5 Conclusions

The review in this chapter focused on analytical modeling methods for estimating the throughput in serial production systems, bottleneck detection techniques, buffer allocation policies and the idea of creating a simulation interface to simplify the process of model building. From the literature it is clear that the analytical models require simplifying assumptions in order to remain mathematically tractable. These analytical models do not, however, model the underlying real-world system accurately. Some of the common assumptions made in analytical modeling techniques can be listed as follows:

- Single part processing

- No inspection stations
- Specific distributions for processing time, and TTF
- No operator interference
- Specific queuing disciplines
- No product mix
- Negligible transportation times between stations

While, some analytical models address a part of these assumptions, it ignores the rest. The bottleneck detection techniques presented in this chapter also make assumptions like unlimited buffer sizes in front of stations, exponential processing time, minimum or no operator interference, and specific queuing disciplines. While these techniques might work for some lines, it might not correctly identify the bottleneck resource of any given serial production system. This thesis attempts to capture the behavior of serial production systems and there by detect the bottleneck resource and mitigate it, without making the assumptions stated above. We have also adapted the idea of creating a simulation interface proposed by Farrington et al. (1995) and Doss and Ulgen (2004) to reduce repetitive model building and modification time. So given a serial production system, the Excel template developed in this thesis will enable the analyst to generate the simulation model dynamically. The results from the model will be analyzed with the help of the algorithm proposed, which will detect the bottleneck and propose modifications to the system in an effort to curb the bottleneck. The next section discusses the problem statement followed by the techniques developed in this thesis.

CHAPTER 3

PROBLEM DEFINITION AND METHODOLOGY

3.1 Problem Definition

A technique to detect bottleneck resources in complex serial production systems with the help of discrete event simulation is proposed in this thesis. A procedure for mitigating the bottleneck of any given serial production line is also developed. This will provide decision support to the analyst in the form of addition of buffers, reallocation of operators, or addition of parallel bottleneck resources. The objectives of this thesis are to identify the line bottlenecks and to use an iterative procedure to maximize line throughput. Our bottleneck detection technique is then contrasted with a traditional bottleneck detection method with the help of case studies. The simulation model used for analyzing the serial production system is dynamically generated from a Microsoft Excel template in Arena 7.01 with the help of a VBA project. A static model for the system under consideration is also built using the Excel template. In the next section we define a typical station and its processing parameters.

3.2 Formalization of a Typical Station

A serial production line is characterized as K work stations arranged in series and each station is labeled as k where $k = 1$ to K. Let N be the number of products being

processed in the line and each product is marked as n where n = 1 to N. The product mix is denoted by M_n, where M_n is the proportion of production allocated to product n. Then,

$$\sum_{n=1}^{N} M_n = 1.00$$

The processing time per unit (P) for the n^{th} product at station k is denoted as $P_k(n)$. The attributes of a station are listed in Table 3.1.

Table 3.1: Processing Parameters

Attribute	Expression	Symbol
Processing time per unit	Stochastic	$P_k(n)$
Time to failure	Stochastic	L_k
Cycles to exhaust	Stochastic	C_k
Time to repair	Stochastic	R_k
Time to replenish	Stochastic	H_k

The static model built as a part of this thesis, uses mean values of these attributes. Some of the other notations used in this thesis are:

T is the total production time

F_k is the expected time lost due to repair

G_k is the expected time lost due to replenishment

B_k is the % of time spent by station k in blocked state

I_k is the Maximum units that can be produced in station k without considering failures and exhausts

$U_k = T - F_k$ is the available processing time in station k after the expected total repair time has been removed

$V_k = T - F_k - G_k$ is the available processing time in station *k* after the expected total repair and replenish times are removed

J_k is the maximum units that can be produced in station *k* by considering repair and replenishment

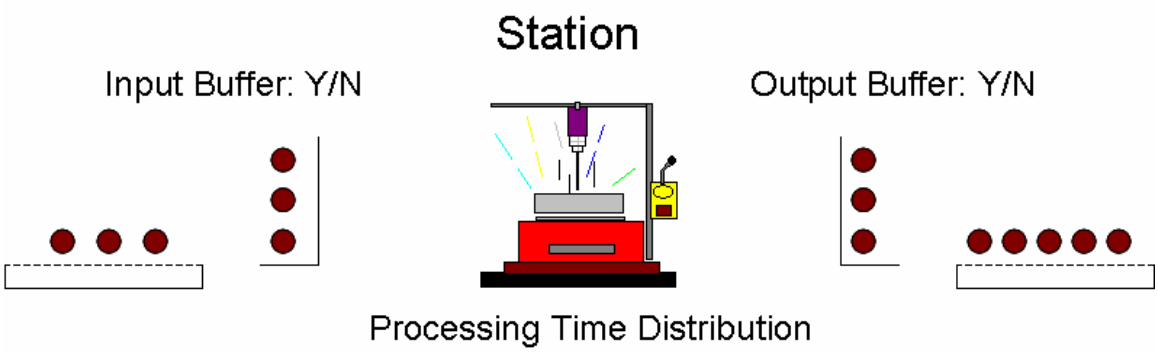

Figure 3.1: Typical Station

Figure 3.1 is a snap shot of a typical station in our serial production system. A station will exist in one of the seven states idle, busy, failed, repair, exhaust, replenish and blocked. A machine remains in the idle state when it is not processing a part and there are no parts available in the input area. As soon as it gets started to work on a unit, its state changes to busy. Production process can be interrupted due to machine failures and component part exhaustion. As soon as a machine fails, its state changes to failure and an operator is requested for service. The state of the machine changes from failure to repair when an operator arrives to repair the machine. Machine exhaustion occurs when the raw material in a machine is exhausted. The machine remains in the exhaust state until an operator arrives to replenish the exhausted parts, which will change the state of the machine to replenish. When a machine is finished processing a unit it passes the unit

on to the next machine. If it is unable to do so due to unavailable capacity in its output area then the machine is considered to be blocked.

Figure 3.2 illustrates the busy and idle states of a machine. Here Station A is working on a part and the Station B has exhausted the parts in the buffer in front of it. Station B is ready to work but there are no parts available for it to work on and it is in idle state (starved). Stations A and C are in busy state. Figure 3.3 illustrates the blocked, failed and repair states of a machine. In this case, Station B has finished working on its part but is not able to work on the next part because the buffer downstream is full and cannot accept any more parts. Station B is thus said to be blocked. Stations A and C have failed and both will be fixed by 'Operator X'. Here Station A is in the process of being fixed and is in repair state where as Station C is still waiting for 'Operator X' and is in failed state. Figure 3.4 portrays replenish and exhaust states. Its very similar to the previous case the only difference being the stations have exhausted their resource instead of failing and Station A, which is being served by 'Operator X' is in replenish state and Station C is in exhaust state waiting for service from 'Operator X'.

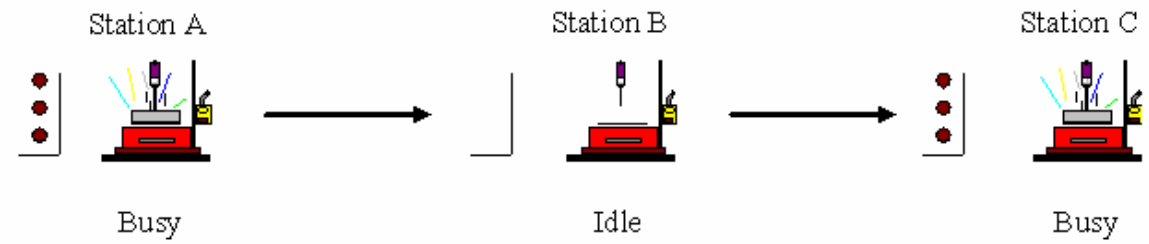

Figure 3.2: Busy and Idle States

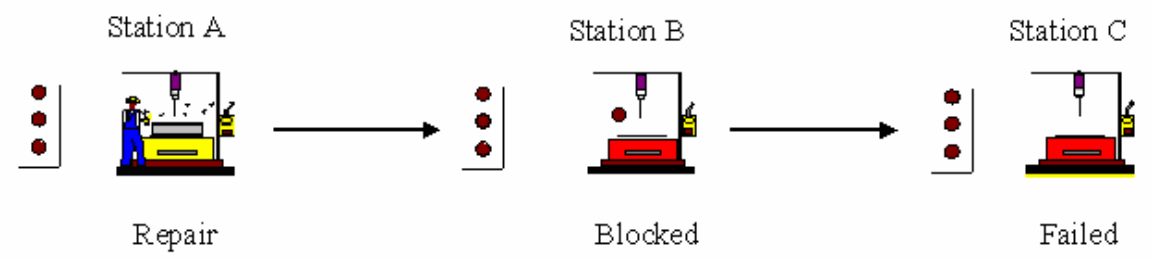

Figure 3.3: Blocked, Failed and Repair States

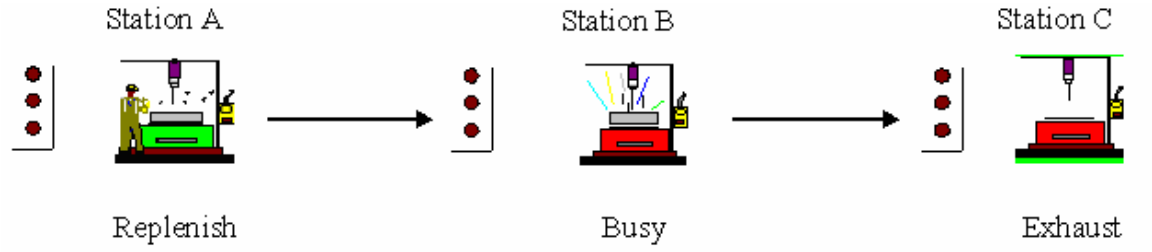

Figure 3.4: Replenish and Exhaust States

3.3 Methodology

3.3.1 Data Collection

Simulation models are computer programs that process input data to predict the output of the system by statistical sampling. Even when the underlying system is modeled accurately, if the input values plugged in are incorrect then the results will be misleading. A simulation model which is highly complex and stochastic in nature is difficult to validate. This is one of the primary reasons why the analysts need to have data that is representative of the actual system. According to Jadhav and Smith (2005) and Jadhav (2005), data can be extracted from the system either from historical databases if the system exists or similar systems if the system is nonexistent. The techniques developed in

this thesis are embedded in a ten step methodology proposed by Law and Kelton (2000). The flowchart of the methodology is shown in Figure 3.5.

The ten step methodology provides an iterative procedure to maximize the throughput of the line. If the desired throughput of the system is not met for a proposed scenario then the bottleneck of the system is detected and mitigated using the techniques suggested in Section 3.3.7. The configuration of the system is modified accordingly and the iterative procedure is carried out until the target production rate for the system is achieved.

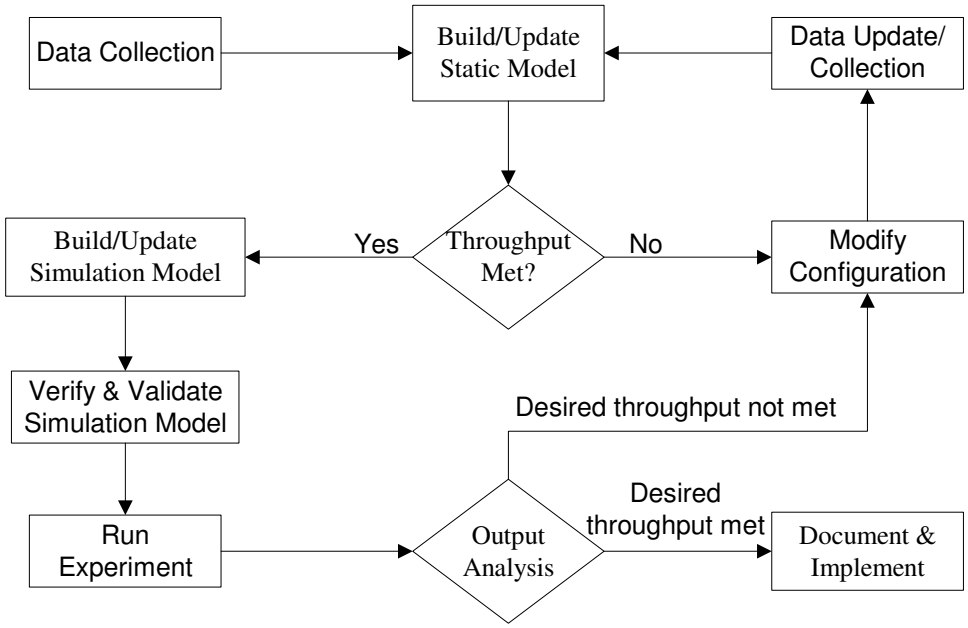

Figure 3.5: Methodology for Simulation Modeling and Analysis of Serial Production Systems

Robertson and Perera (2001) conducted a study and found that data collection in most simulation studies is unstructured, not automated, time consuming and hence forth comes at a high cost. The data thus extracted from the system may or may not be simulation friendly data or in other words the data would have to be processed further to

be used in a simulation study. Hence we have standardized the set of data that will be used in the simulation study to quicken this process.

The data required for a simulation study can be classified into i) Data for static model and ii) Data for simulation model. The simulation model will also make use of the data available for static model.

Data for static model are listed as follows:

i) Shift Details:

a. Available time per year
b. Changeover Time
c. Changeover Frequency

ii) Process Details:

a. Operator Issues
    i. Failure distribution
    ii. Time to Repair
b. Maintenance Issues
    i. Cycles to Exhaust
    ii. Time to Replenish

iii) Product Mix Details

Data for simulation model are listed as follows:

i) Machine Details:

a. Processing type: single or batch
b. Indexing time (if required)

c. Length of conveyor (if required)

ii) Operator Details:

a. Allocated set of stations

b. Priorities

iii) Inspection Details:

a. Percentage passing

b. Rework details

i. Rework resource

ii. Rework time

iii. Percentage scrapped

iv) Buffer Details:

a. Location and capacity of buffer

b. Queuing policy followed

v) Conveyor Details:

a. Length

b. Accumulating/non-accumulating

c. Speed

vi) Process flow of the system

Data pertaining to failures can be represented/collected in three ways namely, i) % of failures per shift, ii) Time To Failure, and iii) # of failures per shift. The static model developed in our study uses the # of failures per shift data.

3.3.2 Build/Update Static Model

Static model which is an analytical modeling technique is a formalization of Mean Value Analysis (Reiser and Lavenberg, 1980) where mean values for processing times, failure/exhaust, repair and replenish times are used to estimate production rates. Using the data collected, a static model is generated dynamically with the help of the Excel template which would predict the throughput of the system under consideration. In this section we will describe the processing steps involved in static model generation.

The processing steps involved in generation of static model are:

Step 0: Calculate mean processing time per unit $P_k = \sum_{n=1}^{N} \left[E\left[P_k(n) \right] \times M_n \right]$

Step 1: Throughput without considering failures and exhausts $I_k = \frac{T}{P_k}$

Step 2: Time lost due to repair $F_k = \frac{\{E[R_k] \times T\}}{\{E[L_k] + E[R_k]\}}$

Step 3: Calculate $U_k = T - F_k$

Step 4: Time lost due to replenishment $G_k = \frac{\{E[H_k] \times U_k\}}{\{E[H_k] + (E[C_k] \times P_k)\}}$

Step 5: Calculate $V_k = U_k - G_k$

Step 6: Throughput considering failures and exhausts $J_k = \frac{V_k}{P_k}$

This procedure is carried out for every station in the system. The throughput is essentially determined by the processing rate of the slowest machine (Jadhav and Smith, 2005 and Jadhav, 2005). The static model also dynamically generates the resource state graph which is a graphical representation of the % of time spent by each resource in the

seven states discussed earlier. For more information on resource state graph the reader is directed to refer Jadhav and Smith (2005) and Jadhav (2005). The expressions used for calculation of % of time spent by a resource in each of the states are listed below.

For resource *k*,

a. % of time spent in the Busy state $= \dfrac{\left[P_k \times \min_{\forall k=1,\ldots,K}(J_k)\right]}{T}$

b. % of time spent in the Replenishment state $= \dfrac{G_k}{T}$

c. % of time spent in the Repair state $= \dfrac{F_k}{T}$

d. % of time spent in the Failure/Exhaust state $= 0$ (Time in exhaust and failure states model complex operator interference issues; assumed zero in the static model)

e. % of time spent in the Blocked state $= 0$ (Time in blocked state indicates complex inter-process interactions; assumed zero in the static model)

f. % of time spent in the Idle state $= 100 -$ (Σ Other States)

The static model doesn't consider i) complex operator interference issues, ii) influence of capacitated buffers, iii) inter-process interactions, and iv) effects of rework and hence the values obtained are just estimates. Calculating the % of time spent by a resource in the blocked state in the static model is not straightforward and is beyond the scope of this thesis. It is assumed to be zero in our static model. The static model predicts the throughput for two standardized scenarios:

a. Scenario 1: Without Failures and Exhausts

b. Scenario 2: With Failures and Exhausts

3.3.3 Expected Throughput Met?

Once the static model is developed it is compared with existing systems or similar systems to check for correctness of the input data. Since static model doesn't consider complex operator interference issues and inter-process interactions, the expected throughput from the static model for the scenario 2 is considered to be overestimated. In other words, a simulation model built with this input data cannot be expected to have higher values for throughput. So if the domain experts feel that the throughput is below their requirement or expectation then the configuration of the system is modified by adopting procedures to detect and mitigate the bottleneck described in this thesis. The data is updated /collected and static model is rebuilt accordingly. This feedback loop which is made available by the static model reduces time by giving the user a heads-up (by providing a rough estimate of the throughput). This is particularly useful for systems which are non-existent.

3.3.4 Build/Update Simulation Model

The simulation model built will be an exact representation of the system under study. It will account for operator interference issues, effect of capacitated buffers in between stations, inter-process interactions like blocking and starving, and the effects of rework and scrapping of parts. The simulation model will be dynamically generated from

the Excel template. The processing steps involved in dynamically generating the Arena model are listed as follows:

Step 0: Shortlist the modules that will be used in the simulation study

Step 1: Identify the operands whose values are to be obtained from the Excel template

Step 2: Select the appropriate module for each station in the system from a pull down menu provided in the Excel template. Each module in the pull down menu has a unique feature.

For instance, if a station is meant to have a built-in conveyor and multiple head processing, then the module which provides this feature is selected. On the other hand if the model developer wants to add inspection stations at different points in the line, then a module which reroutes the parts based on a fixed probability is chosen.

Step 3: Enter the details required to build the simulation model in the Excel template

Step 4: Execute the VBA code which will generate the simulation model in Arena 7.01

Usually the first two steps are exercised only at the beginning of a simulation study (since the modules that will be used for a study will remain fairly constant). If a new module is to be used then the dataset is updated accordingly. Steps 2, 3 and 4 are to be executed each time a new model is built.

Before the user executes the code to build the simulation model, it has to be made sure that the template used in a study is automatically attached to the project bar when Arena 7.01 opens. This is done by opening Arena 7.01 Tools/Options/Settings/Auto Attach Panels and specifying the path where the .tpo file being used in modeling is located.

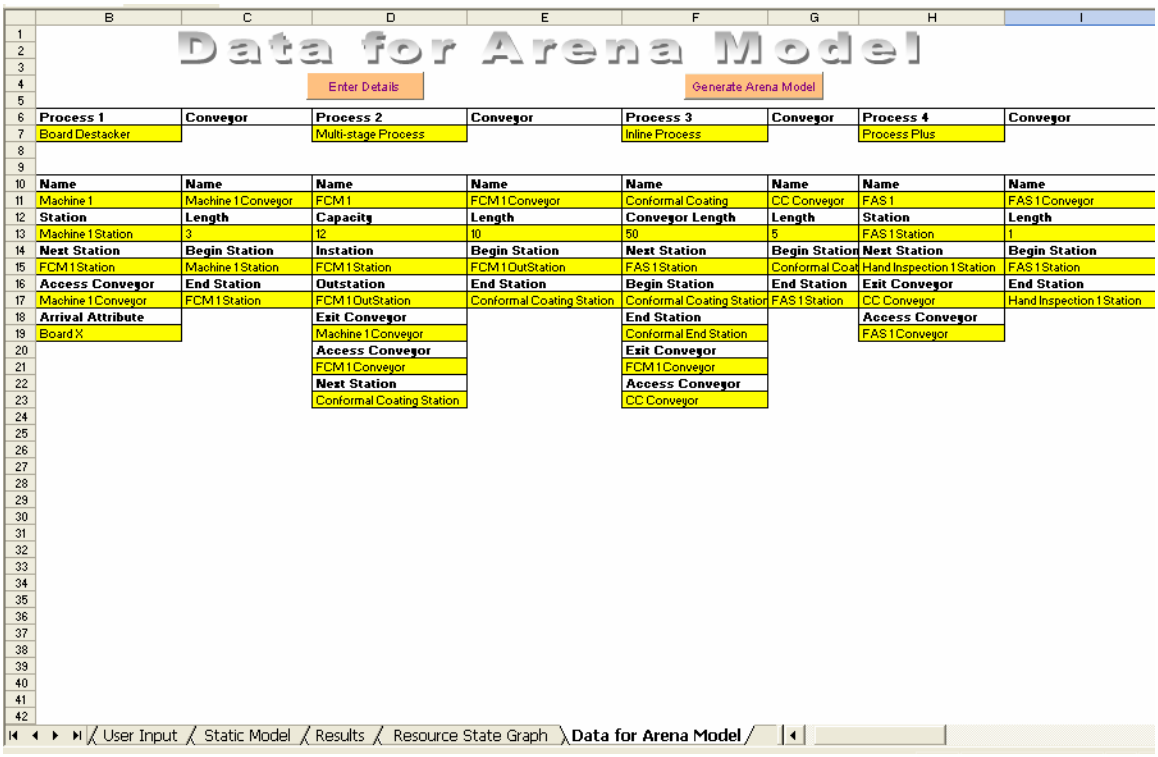

Figure 3.6: Excel Template from which Arena Model is generated

In this thesis, the simulation model which was dynamically generated makes use of the template developed by Mukkamala et al. (2003) and Mukkamala (2003). A conveyor is defaulted between each station. A snap shot the Excel template from which the Arena model is generated is shown in Figure 3.6.

3.3.5 Verify and Validate Simulation Model

The simulation model with base configuration (that is no failures and exhausts) is run and compared with scenario 1 of the static model. The simulation model is animated and process flow of the products is verified with the real system. The base simulation model will consider capacitated buffers and conveyors/transporters in between the resources. The static model assumes that there is no time lost when a product moves from

one resource to another. These factors will influence the results of the simulation model. But if the simulation model is an accurate representation of the real system, the throughput determined by the simulation model will not vary significantly from that of the static model. The managers are consulted for validation purposes.

3.3.6 Run Experiment

Stochastic variables like failure, repair, exhaust and replenishment rates, % of defective units, etc. are incorporated into the verified and validated simulation model at this point. These stochastic variables make the model dynamic. If the concerned firm has some predetermined strategies for the line then each of those ideas are incorporated in a separate simulation model. The number of replications is decided by making a trial run and the resulting 95% confidence interval of the half width of the throughput from the underlying production system (Jadhav and Smith, 2005 and Jadhav, 2005). Each model will usually have a different value for number of replications due to the randomness. The simulation model is run accordingly and the throughput of the line under study is determined. A resource state graph for the system is generated and the system is analyzed which is explained in the next section.

3.3.7 Output Analysis of the System

The simulation results thus obtained will reveal the performance level of each station with respect to the seven states we have defined. The results have to be analyzed and effective measures are to be undertaken to enhance the efficiency of the system. To

do this the analyst will have to identify the station which has the greatest impact on the performance of the whole system, which will be the bottleneck resource.

The busy state reflects upon the effective % of time a resource was engaged in production. Replenish and repair states reflect upon the inability of a station to manufacture products due to exhaustion and failure respectively. Exhaust and failure states quantify the operator interferences. Idle and blocked state of a station quantifies the interdependency (coupling) of the station on others. It can also be interpreted as the effective % of time that a resource can utilize for production if it were decoupled from other stations in the system. When a machine is in blocked or idle state it is ready to accept parts for processing but is unable to do so due to complex inter-process interactions. The machine which is least affected by the inter-process interactions will be the bottleneck resource. This transforms to the fact that the machine which has the least value for the % of time spent in idle and blocked states is the bottleneck resource. In this thesis we compare our bottleneck detection technique with the concept of 'Machine with highest utilization' being the bottleneck resource. In chapter 4 we present case studies where in our bottleneck detection technique works better. This value will be referred to as Bottleneck Index from here on.

The states of a machine are represented in the resource state graph which is generated from Arena with the help of VBA (Jadhav and Smith, 2005 and Jadhav, 2005). Three possible changes can be made in the system to counter the bottleneck problem. They are listed as follows:

a. Add buffers before or after the bottleneck process
b. Re-allocate the workforce to reduce the time spent in failure/exhaust

c. Consider parallel processing option for the bottleneck resource

The decision to select the options stated above is not straightforward and hence we have developed an algorithm which will be helpful for the analyst in this regard. The processing steps are as follows:

Step 0: If Exhaust/failure state for any station is >= Z %, consider reallocating the operator for those stations

Step 1: If Blocked state of the bottleneck is > 0 %, then add buffers right after the bottleneck resource

Step 2: If Idle state of the bottleneck is >= Y %, then add buffers appropriately for machines upstream and stop

Step 3: If Idle state of the bottleneck is > 0 %, add buffers right before the bottleneck resource

Step 4: If (Idle + Blocked) states of bottleneck resource are =< X %, add parallel resource and stop

The values that the parameters X, Y and Z will assume are decided by the analyst and the management, which is going to be subjective and it will depend on the objective of the analysis that is being conducted. Usually the values range from 1 to 40. The values of X, Y and Z parameters used in this thesis are assumed to be 5, 25 and 2 respectively. The interpretations of variables are:

1. Higher values of 'Z': Operator interchangeability/sharing is not easy or very few cross trained operators available. Another interpretation is that, highly skilled operators are required for certain processes.

2. Lower values of ‘Y’: Management is ready to add buffers around non-bottleneck resources to reduce the % of time spent by a bottleneck resource in idle state.
3. Higher values of ‘X’: Management is in favor of adding resources either due to the need for substantial increase in production or the cost of adding machines is relatively less expensive than adding buffers in the long run.

As a part of this research, another algorithm was developed to help the analyst decide on the changes to be made. This algorithm is an approximation technique which uses the % of time spent by the machines in blocked state upstream to that of the bottleneck resource and the blocking of the bottleneck resource due to the machines downstream. An index (referred to as Blocked Index) is calculated based on these values and the analyst is advised to either add buffers or add a parallel processing resource based on the value of this index. If the n^{th} machine is the bottleneck resource then Blocked Index is calculate as follows:

$$\text{Blocked Index} = [(\sum_{i=1}^{n-1} B_i - B_n) / (n-1)]$$

Mathematical interpretation to the computation of the Blocked Index is that the effects of machines downstream to bottleneck resource can be approximately represented by the % of time spent by the bottleneck resource in the blocked state. The effect of the bottleneck resource on the machines upstream can similarly be approximated to the blocked state of those machines. Hence the average effect of the bottleneck machine on resources upstream is given by the Blocked Index.

If the value of Blocked Index is higher than 40% (which is subjective), then the analyst will be directed to add a parallel resource to the bottleneck machine.

Pitfalls of this algorithm:

1. It overestimates the effect of the bottleneck machine on the resources upstream as it fails to account for the blocking due to other non-bottleneck resources
2. The starving of the bottleneck machine is not taken into account, which can be minimized by adding buffers, thereby increasing the throughput without adding a parallel resource
3. The failure/exhaust states are not considered and hence the operator allocation issue is not addressed

Due to the above stated inconveniences this algorithm was not used for decision making purposes. Modifying the configuration of the system accordingly is discussed in the next section.

3.3.8 Modify Configuration

After the analysis of the simulation models the proposed changes and the results are shared with the decision makers and their suggestions for improvements are also considered. If the decision makers want to conduct a cost analysis then factors like WIP, machine costs, Labor costs etc. are taken into account. After analyzing all the possible options a viable option is chosen and the data is updated /collected accordingly.

3.3.9 Data Update/Collection

The existing data is either updated or new data is collected to rebuild the static model. Data is usually updated in cases where the company makes the choice of adding a parallel resource to the line for which data is already available. New data is collected usually when a new resource is added to the system for which the processing time or

failure/exhaust rates are unknown or some process improvements on a resource is done where the cycle time or the failures/exhausts or both the parameters change. The static model and the simulation model are updated.

3.3.10 Documentation and Implementation

The process is terminated when the decision makers requirement/expectation are met. Static and simulation models for various configurations are documented for future reference and the proposed methodology is enforced.

CHAPTER 4

CASE STUDIES

The case studies conducted on four sets of production lines are presented in this chapter. We analyze ten station lines and fifty station lines with low and high variability in processing parameters. The production lines listed here can be thus be categorized into: i) Long line with low variability in processing parameters , ii) Long line with high variability in processing parameters, iii) Short line with low variability in processing parameters and iv) Short line with high variability in processing parameters. The methodology proposed in the previous chapter has been adapted in modeling and analysis of these lines. The processing parameters under considered are:

- Processing time
- Time to failure
- Cycles to exhaust
- Time to repair
- Time to replenish

4.1 Experimental Setup

In this thesis, we have conducted ten sets of experiments for each category of production line mentioned above. An experiment is setup by arbitrarily assigning mean

values to the processing parameters. The length of the line and coefficient of variation (C.V.) of processing parameters are decided upon by the category in which it falls. For example, a fifty machine line with a C.V. of around 10% for the processing parameters would fall under the category of long lines with low variability in processing parameters. The labeling of the experiments is shown in Table 4.1.

Table 4.1: Labeling of Experiments

Length of Production Line	C.V. between Process Parameters	
	~ 0.10	~ 0.30
10	SL	SH
50	LL	LH

Data points are generated for each processing parameter and each machine in the system is assigned with a set of processing expression. Distribution for each processing expression is selected arbitrarily and the system is dynamically generated from the Excel template. The system thus modeled is simulated and tested against the algorithm proposed in this thesis. This procedure can be explained better with the help of an example. Let us consider a short line with low variability in processing parameters. The mean values and C.V. between processes which were arbitrarily assigned are listed below in Table 4.2. The range of all the mean values used in this thesis for processing parameters is obtained from Jadhav and Smith (2005) and Jadhav (2005).

Table 4.2: SL-Configuration Example

Attribute	Mean Value	Units	C.V. between processes
Processing Time P_x	16	Sec	0.12
Time to Failure L_x	64.5	Min	0.10
Cycles to Exhaust C_x	30	Parts	0.13
Time to Repair R_x	280	Sec	0.11

Time to Replenish H_x	90	Sec	0.09

Normal distribution is selected to generate the data points and each machine is assigned a data set as shown in Table 4.3. These data points in the table are used as mean values for stations in the static model and a distribution is selected for each attribute in the simulation model.

Table 4.3: Data for the Stations

Resource	Attribute				
	Processing Time	Time to Failure	Cycles to Exhaust	Time to Repair	Time to Replenish
Machine A	16.72	84.75	25	313.01	97.07
Machine B	17.86	82.07	25	326.76	81.98
Machine C	16.62	65.12	31	270.59	87.65
Machine D	19.54	72.54	23	302.26	98.05
Machine E	17.13	68.58	33	287.75	83.23
Machine F	17.06	88.33	34	307.07	87.24
Machine G	16.25	74.72	28	282.19	98.58
Machine H	18.87	84.68	40	288.56	74.11
Machine I	18.63	65.3	27	298.04	94.51
Machine J	16.61	72.73	37	292	80.45

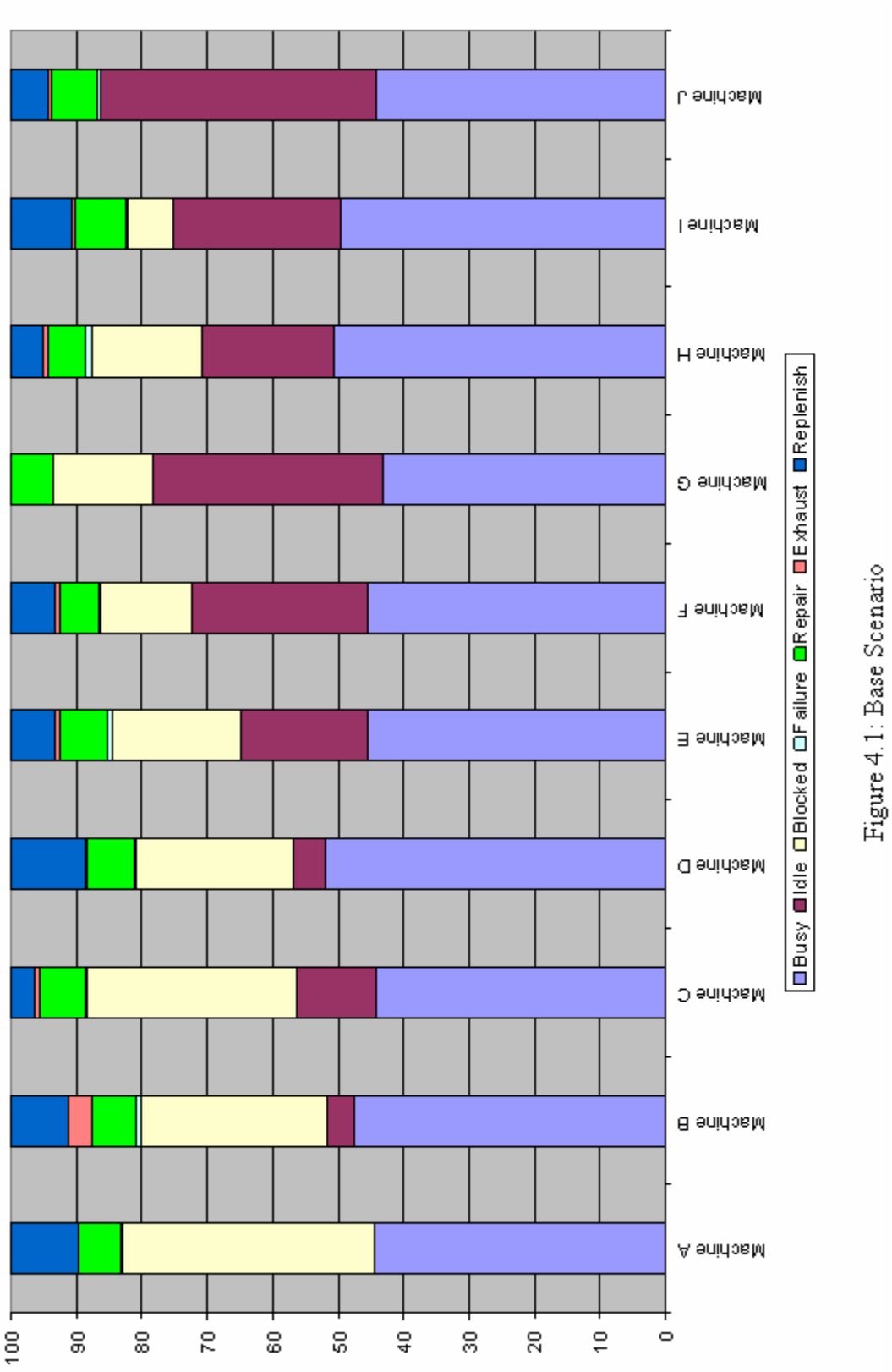

Figure 4.1: Base Scenario

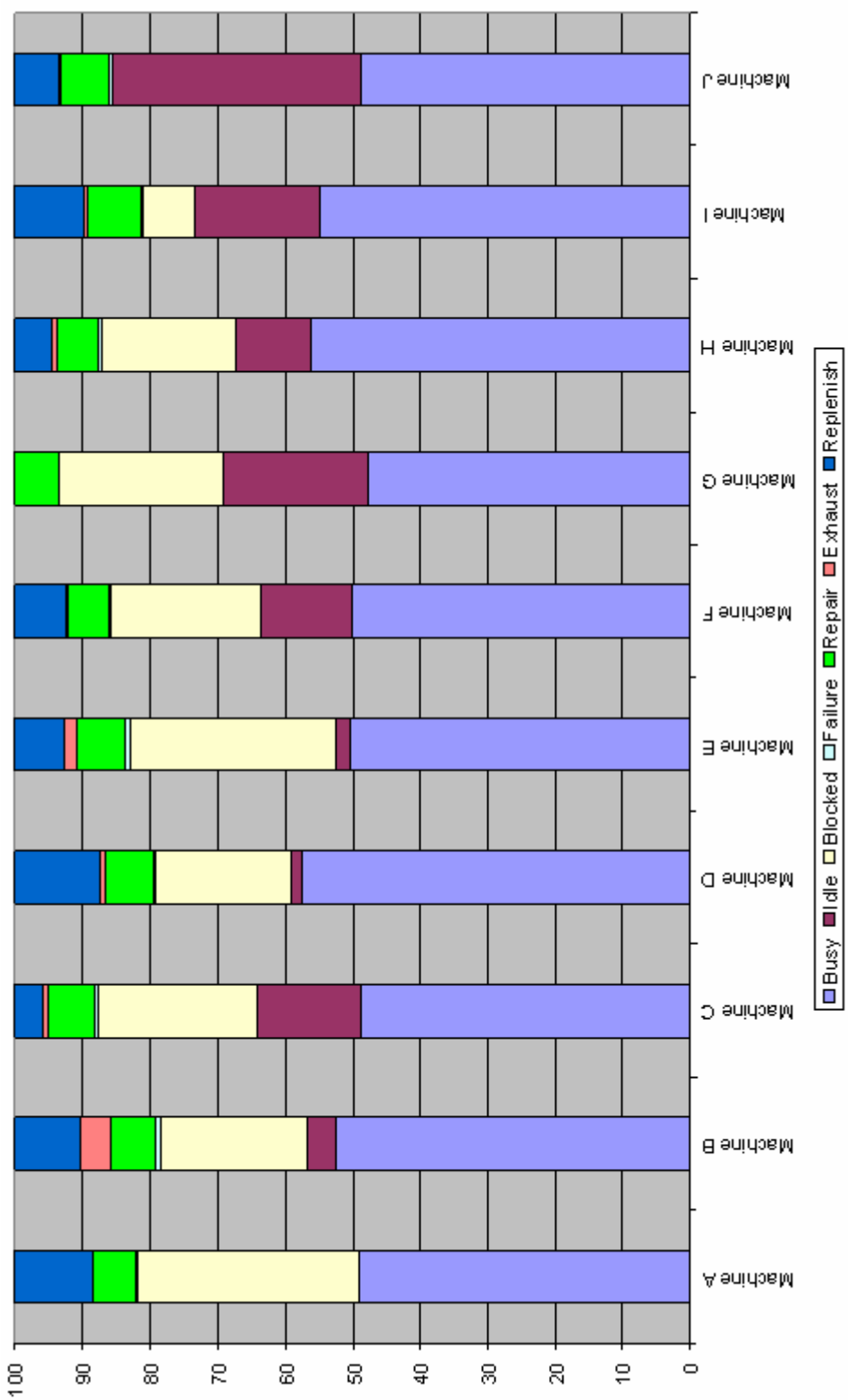

Figure 4.2: Improved System

Results:

Table 4.4: SL Example Results

Results		
Initial throughput: ***479,984***		
	Bottleneck Index Method	Highest Utilization Method
Bottleneck	Machine D	Machine D
Mitigation	Add 7 buffers in front of Machine D and 10 behind it	Add 7 buffers in front of Machine D and 10 behind it
Throughput	***530,581***	***530,581***

The resource state graphs of the base case scenario and the one after adding buffers to the bottleneck resource are shown in Figure 4.1 and Figure 4.2 respectively. The case studies are presented in the next section.

4.2 Analysis of LL-Configuration

Case 1:

Table 4.5: Case LL-1

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	20	Sec	0.12
Time to Failure L_k	53.25	Min	0.13
Cycles to Exhaust C_k	115	Parts	0.11
Time to Repair R_k	155	Sec	0.08
Time to Replenish H_k	155	Sec	0.09
Results			
Initial throughput: ***88,184***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine X	Machine F	
Mitigation	Add 10 buffers in front of Machine X and 15 behind it	Add 10 buffers in front of Machine F and 20 behind it	
Throughput	***104,335***	***100,734***	

Case 2:

Table 4.6: Case LL-2

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	12	Sec	0.10
Time to Failure L_k	71	Min	0.10
Cycles to Exhaust C_k	70	Parts	0.10
Time to Repair R_k	120	Sec	0.10
Time to Replenish H_k	140	Sec	0.10
Results			
Initial throughput: ***125,452***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AV	Machine AV	

Mitigation	Add 10 buffers in front of Machine AV and 7 behind it and 3 buffers in front of and behind Machine AT	Add 10 buffers in front of Machine AV and 7 behind it and 3 buffers in front of and behind Machine AT
Throughput	***143,623***	***143,623***

Case 3:

Table 4.7: Case LL-3

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	19	Sec	0.09
Time to Failure L_k	71	Min	0.13
Cycles to Exhaust C_k	135	Parts	0.11
Time to Repair R_k	170	Sec	0.13
Time to Replenish H_k	145	Sec	0.08
Results			
Initial throughput: ***85,106***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AI	Machine AI	
Mitigation	Add 10 buffers in front of Machine AI and 10 behind it, and an operator	Add 10 buffers in front of Machine AI and 10 behind it, and an operator	
Throughput	***109,253***	***109,253***	

Case 4:

Table 4.8: Case LL-4

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	6	Sec	0.11
Time to Failure L_k	56.8	Min	0.11
Cycles to Exhaust C_k	100	Parts	0.09
Time to Repair R_k	80	Sec	0.11

Time to Replenish H_k	60	Sec	0.09
Results			
Initial throughput: ***268,871***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine F	Machine F	
Mitigation	Add 8 buffers in front of Machine F and 20 behind it	Add 8 buffers in front of Machine F and 20 behind it	
Throughput	***319,006***	***319,006***	

Case 5:

Table 4.9: Case LL-5

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	10	Sec	0.13
Time to Failure L_k	85.2	Min	0.12
Cycles to Exhaust C_k	100	Parts	0.08
Time to Repair R_k	100	Sec	0.09
Time to Replenish H_k	90	Sec	0.11
Results			
Initial throughput: ***175,707***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine Z	Machine R	
Mitigation	Add 5 buffers in front of Machine Z and 12 behind it	Add 10 buffers in front of Machine R and 8 behind it	
Throughput	***200,748***	***192,407***	

Case 6:

Table 4.10: Case LL-6

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	14	Sec	0.11
Time to Failure L_k	60.85	Min	0.09

Cycles to Exhaust C_k	120	Parts	0.13
Time to Repair R_k	130	Sec	0.12
Time to Replenish H_k	100	Sec	0.09
Results			
Initial throughput: ***133,799***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine X	Machine AT	
Mitigation	Add 10 buffers behind Machine X	Add 5 buffers in front of Machine AT and 10 behind it	
Throughput	***155,529***	***152,734***	

Case 7:

Table 4.11: Case LL-7

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	21.5	Sec	0.10
Time to Failure L_k	56.8	Min	0.09
Cycles to Exhaust C_k	105	Parts	0.13
Time to Repair R_k	135	Sec	0.09
Time to Replenish H_k	135	Sec	0.10
Results			
Initial throughput: ***95,876***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AU	Machine AU	
Mitigation	Add 10 buffers in front of Machine AU and 5 behind it	Add 10 buffers in front of Machine AU and 5 behind it	
Throughput	***106,815***	***106,815***	

Case 8:

Table 4.12: Case LL-8

Configuration

Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	18	Sec	0.08
Time to Failure L_k	65.5	Min	0.11
Cycles to Exhaust C_k	130	Parts	0.09
Time to Repair R_k	120	Sec	0.13
Time to Replenish H_k	140	Sec	0.12
Results			
Initial throughput: ***114,569***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AQ	Machine AQ	
Mitigation	Add 10 buffers in front of Machine AQ and 5 behind it	Add 10 buffers in front of Machine AQ and 5 behind it	
Throughput	***127,498***	***127,498***	

Case 9:

Table 4.13: Case LL-9

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	8	Sec	0.13
Time to Failure L_k	94.6	Min	0.08
Cycles to Exhaust C_k	95	Parts	0.12
Time to Repair R_k	90	Sec	0.12
Time to Replenish H_k	80	Sec	0.11
Results			
Initial throughput: ***217,909***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AC	Machine AF	
Mitigation	Add 12 buffers in front of Machine AC and 5 behind it	Add 20 buffers in front of Machine AF and 3 behind it	
Throughput	***244,860***	***231,759***	

Case 10:

Table 4.14: Case LL-10

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	16	Sec	0.09
Time to Failure L_k	77.45	Min	0.08
Cycles to Exhaust C_k	105	Parts	0.12
Time to Repair R_k	145	Sec	0.11
Time to Replenish H_k	120	Sec	0.08
Results			
Initial throughput: ***118,892***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine V	Machine V	
Mitigation	Add 10 buffers in front of Machine V and 15 behind it	Add 10 buffers in front of Machine V and 15 behind it	
Throughput	***136,133***	***136,133***	

Ten sets of lines with LL configuration were studied in this section. It was found that the Bottleneck Index Method identified the same bottleneck as that of the Highest Utilized Machine Method on six occasions. The Bottleneck Index Method yielded better results in terms of throughput of the line for the remaining four cases. The results are presented in Table 4.15. The scenarios where both methods found the same bottleneck are bolded.

Table 4.15: Summary of LL-Configuration Results

Experiment #	Throughput		
	Base Scenario	Bottleneck Index Method	Highest Utilized Machine Method
Case 1	88,184	104,335	100,734
Case 2	**125,452**	**143,623**	**143,623**
Case 3	**85,106**	**109,253**	**109,253**
Case 4	**268,871**	**319,006**	**319,006**
Case 5	175,707	200,748	195,407

Case 6	133,799	155,529	152,734
Case 7	**95,876**	**106,815**	**106,815**
Case 8	**114,569**	**127,498**	**127,498**
Case 9	217,909	244,860	231,759
Case 10	**118,892**	**136,133**	**136,133**

4.3 Analysis of LH-Configuration

Case 1:

Table 4.16: Case LH-1

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	12	Sec	0.30
Time to Failure L_k	71	Min	0.30
Cycles to Exhaust C_k	130	Parts	0.30
Time to Repair R_k	120	Sec	0.30
Time to Replenish H_k	140	Sec	0.30
Results			
Initial throughput: ***102,850***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AV	Machine AV	
Mitigation	Add 15 buffers in front of Machine AV and 7 behind it	Add 15 buffers in front of Machine AV and 7 behind it	
Throughput	***111,725***	***111,725***	

Case 2:

Table 4.17: Case LH-2

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	10	Sec	0.33
Time to Failure L_k	85.2	Min	0.32
Cycles to Exhaust C_k	100	Parts	0.28

Time to Repair R_k	100	Sec	0.29
Time to Replenish H_k	90	Sec	0.33
Results			
Initial throughput: ***143,240***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine R	Machine R	
Mitigation	Add 5 buffers in front of Machine R and 10 behind it	Add 5 buffers in front of Machine R and 10 behind it	
Throughput	***149,833***	***149,833***	

Case 3:

Table 4.18: Case LH-3

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	14	Sec	0.31
Time to Failure L_k	60.85	Min	0.29
Cycles to Exhaust C_k	120	Parts	0.33
Time to Repair R_k	130	Sec	0.32
Time to Replenish H_k	100	Sec	0.29
Results			
Initial throughput: ***92,533***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine X	Machine AT	
Mitigation	Add 15 buffers in front of Machine X and 20 behind it	Add 25 buffers in front of Machine AT and 15 behind it and 4 buffers in front of and behind Machine AR	
Throughput	***111,221***	***107,092***	

Case 4:

Table 4.19: Case LH-4

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	16	Sec	0.29
Time to Failure L_k	77.45	Min	0.28
Cycles to Exhaust C_k	105	Parts	0.32
Time to Repair R_k	145	Sec	0.31
Time to Replenish H_k	120	Sec	0.28
Results			
Initial throughput: ***75,131***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine V	Machine V	
Mitigation	Add 5 buffers in front of Machine V and 15 behind it	Add 5 buffers in front of Machine V and 15 behind it	
Throughput	***87,154***	***87,154***	

Case 5:

Table 4.20: Case LH-5

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	18	Sec	0.28
Time to Failure L_k	65.5	Min	0.31
Cycles to Exhaust C_k	130	Parts	0.29
Time to Repair R_k	120	Sec	0.33
Time to Replenish H_k	140	Sec	0.32
Results			
Initial throughput: ***83,335***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AW	Machine J	
Mitigation	Add 20 buffers in front of Machine AW and 5 behind it	Add 10 buffers in front of Machine J and 20 behind it	
Throughput	***94,947***	***91,045***	

Case 6:

Table 4.21: Case LH-6

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	20	Sec	0.32
Time to Failure L_k	53.25	Min	0.33
Cycles to Exhaust C_k	115	Parts	0.31
Time to Repair R_k	155	Sec	0.28
Time to Replenish H_k	155	Sec	0.29
Results			
Initial throughput: ***63,416***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine G	Machine G	
Mitigation	Add 8 buffers in front of Machine G and 25 behind it, and an operator	Add 8 buffers in front of Machine G and 25 behind it, and an operator	
Throughput	***73,051***	***73,051***	

Case 7:

Table 4.22: Case LH-7

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	21.5	Sec	0.30
Time to Failure L_k	56.8	Min	0.29
Cycles to Exhaust C_k	105	Parts	0.33
Time to Repair R_k	135	Sec	0.29
Time to Replenish H_k	135	Sec	0.30
Results			
Initial throughput: ***57,588***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine V	Machine V	
Mitigation	Add 7 buffers in front of Machine V and 12 behind	Add 7 buffers in front of Machine V and 12 behind	

	it, and an operator	it, and an operator
Throughput	***64,561***	***64,561***

Case 8:

Table 4.23: Case LH-8

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	8	Sec	0.33
Time to Failure L_k	94.5	Min	0.28
Cycles to Exhaust C_k	95	Parts	0.32
Time to Repair R_k	90	Sec	0.32
Time to Replenish H_k	80	Sec	0.31
Results			
Initial throughput: ***155,645***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine W	Machine W	
Mitigation	Add 10 buffers in front of Machine W and 20 behind it	Add 10 buffers in front of Machine W and 20 behind it	
Throughput	***170,172***	***170,172***	

Case 9:

Table 4.24: Case LH-9

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	6	Sec	0.31
Time to Failure L_k	56.8	Min	0.31
Cycles to Exhaust C_k	200	Parts	0.29
Time to Repair R_k	80	Sec	0.31
Time to Replenish H_k	60	Sec	0.29
Results			

Initial throughput: ***204,178***		
	Bottleneck Index Method	Highest Utilization Method
Bottleneck	Machine T	Machine T
Mitigation	Add 5 buffers in front of Machine T and 20 behind it	Add 5 buffers in front of Machine T and 20 behind it
Throughput	***230,787***	***230,787***

Case 10:

Table 4.25: Case LH-10

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	19	Sec	0.29
Time to Failure L_k	71	Min	0.33
Cycles to Exhaust C_k	220	Parts	0.31
Time to Repair R_k	170	Sec	0.33
Time to Replenish H_k	145	Sec	0.28
Results			
Initial throughput: ***58,779***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine AO	Machine AO	
Mitigation	Add 20 buffers in front of Machine AO and 5 behind it and 2 buffers in front of and behind Machine AK	Add 20 buffers in front of Machine AO and 5 behind it and 2 buffers in front of and behind Machine AK	
Throughput	***66,163***	***66,163***	

Ten sets of lines with LH configuration were studied in this section. It was found that the Bottleneck Index Method identified the same bottleneck as that of the Highest Utilized Machine Method on eight occasions. The Bottleneck Index Method yielded better results in terms of throughput of the line for the remaining two cases. The results

are presented in Table 4.26. The scenarios where both methods found the same bottleneck are bolded.

Table 4.26: Summary of LH-Configuration Results

Experiment #	Throughput		
	Base Scenario	Bottleneck Index Method	Highest Utilized Machine Method
Case 1	**102,850**	**111,725**	**111,725**
Case 2	**143,240**	**149,833**	**149,833**
Case 3	92,533	111,221	107,092
Case 4	**75,131**	**87,154**	**87,154**
Case 5	83,335	94,944	91,045
Case 6	**63,416**	**73,051**	**73,051**
Case 7	**57,588**	**64,561**	**64,561**
Case 8	**155,645**	**170,172**	**170,172**
Case 9	**204,178**	**230,787**	**230,787**
Case 10	**58,779**	**66,663**	**66,663**

4.4 Analysis of SL-Configuration

Case 1:

Table 4.27: Case SL-1

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	6	Sec	0.08
Time to Failure L_k	53.25	Min	0.12
Cycles to Exhaust C_k	60	Parts	0.11
Time to Repair R_k	120	Sec	0.12
Time to Replenish H_k	35	Sec	0.08
Results			
Initial throughput: ***1,650,761***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine E	Machine D	
Mitigation	Add 5 buffers in front of Machine E and 5 behind it	Add 5 buffers in front of Machine D and 5 behind it	

Throughput	***1,834,244***	***1,736,535***

Case 2:

Table 4.28: Case SL-2

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	20	Sec	0.11
Time to Failure L_k	106.5	Min	0.09
Cycles to Exhaust C_k	30	Parts	0.08
Time to Repair R_k	300	Sec	0.09
Time to Replenish H_k	90	Sec	0.11
Results			
Initial throughput: ***450,382***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine E	Machine F	
Mitigation	Add 5 buffers in front of Machine E and 10 behind it, and an operator	Add 15 buffers in front of Machine F and 8 behind it, and an operator	
Throughput	***523,395***	***500,963***	

Case 3:

Table 4.29: Case SL-3

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	15	Sec	0.10
Time to Failure L_k	50	Min	0.10
Cycles to Exhaust C_k	50	Parts	0.10
Time to Repair R_k	20	Sec	0.10
Time to Replenish H_k	120	Sec	0.10
Results			
Initial throughput: ***984,053***			
	Bottleneck Index Method	Highest Utilization Method	

Bottleneck	Machine E	Machine E
Mitigation	Add Machine E	Add Machine E
Throughput	***1,004,148***	***1,004,148***

Case 4:

Table 4.30: Case SL-4

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	16	Sec	0.12
Time to Failure L_k	64.50	Min	0.10
Cycles to Exhaust C_k	30	Parts	0.13
Time to Repair R_k	280	Sec	0.11
Time to Replenish H_k	90	Sec	0.09
Results			
Initial throughput: ***479,984***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine D	Machine D	
Mitigation	Add 7 buffers in front of Machine D and 10 behind it	Add 7 buffers in front of Machine D and 10 behind it	
Throughput	***530,581***	***530,581***	

Case 5:

Table 4.31: Case SL-5

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	22	Sec	0.09
Time to Failure L_k	85	Min	0.13
Cycles to Exhaust C_k	70	Parts	0.11
Time to Repair R_k	140	Sec	0.08
Time to Replenish H_k	85	Sec	0.12
Results			

Initial throughput: ***625,096***		
	Bottleneck Index Method	Highest Utilization Method
Bottleneck	Machine I	Machine G
Mitigation	Add 5 buffers in front of Machine I and 3 behind it	Add 5 buffers in front of Machine G and 8 behind it
Throughput	***665,599***	***633,404***

Case 6:

Table 4.32: Case SL-6

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	25	Sec	0.08
Time to Failure L_k	60.8	Min	0.11
Cycles to Exhaust C_k	60	Parts	0.08
Time to Repair R_k	100	Sec	0.09
Time to Replenish H_k	60	Sec	0.11
Results			
Initial throughput: ***587,503***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine B	Machine B	
Mitigation	Add Machine B	Add Machine B	
Throughput	***588,447***	***588,447***	

Case 7:

Table 4.33: Case SL-7

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	18	Sec	0.13
Time to Failure L_k	106.5	Min	0.08
Cycles to Exhaust C_k	50	Parts	0.09
Time to Repair R_k	150	Sec	0.11

Time to Replenish H_k	100	Sec	0.13
Results			
Initial throughput: ***539,741***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine H	Machine E	
Mitigation	Add 7 buffers in front of Machine H and 10 behind it, and an operator is added to the line	Add 7 buffers in front of Machine E and 15 behind it, and an operator is added to the line	
Throughput	***587,335***	***564,214***	

Case 8:

Table 4.34: Case SL-8

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	12	Sec	0.10
Time to Failure L_k	60.85	Min	0.10
Cycles to Exhaust C_k	25	Parts	0.10
Time to Repair R_k	300	Sec	0.10
Time to Replenish H_k	70	Sec	0.10
Results			
Initial throughput: 584,591			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine D	Machine D	
Mitigation	Add 5 buffers in front of Machine D and 10 behind it	Add 5 buffers in front of Machine D and 10 behind it	
Throughput	664,215	664,215	

Case 9:

Table 4.35: Case SL-9

Configuration			
Attribute	Mean Value	Units	C.V. between processes

Processing Time P_k	10	Sec	0.09
Time to Failure L_k	85.5	Min	0.13
Cycles to Exhaust C_k	40	Parts	0.12
Time to Repair R_k	220	Sec	0.11
Time to Replenish H_k	45	Sec	0.12
Results			
Initial throughput: ***1,089,495***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine E	Machine D	
Mitigation	Add 3 buffers in front of Machine E and 5 behind it	Add 20 buffers behind Machine D	
Throughput	***1,180,721***	***1,140,280***	

Case 10:

Table 4.36: Case SL-10

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	8.5	Sec	0.13
Time to Failure L_k	71	Min	0.12
Cycles to Exhaust C_k	50	Parts	0.10
Time to Repair R_k	170	Sec	0.10
Time to Replenish H_k	50	Sec	0.11
Results			
Initial throughput: ***1,111,752***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine H	Machine C	
Mitigation	Add 10 buffers in front of Machine H and 10 behind it	Add 15 buffers in front of Machine C and 15 behind it	
Throughput	***1,219,555***	***1,141,881***	

Ten sets of lines with SL configuration were studied in this section. It was found that the Bottleneck Index Method identified the same bottleneck as that of the Highest

Utilized Machine Method on four occasions. The Bottleneck Index Method yielded better results in terms of throughput of the line for the remaining six cases. The results are presented in Table 4.37. The scenarios where both methods found the same bottleneck are bolded.

Table 4.37: Summary of SL-Configuration Results

Experiment #	Throughput		
	Base Scenario	Bottleneck Index Method	Highest Utilized Machine Method
Case 1	1,650,761	1,834,244	1,736,535
Case 2	450,382	523,395	500,963
Case 3	**984,053**	**1,004,148**	**1,004,148**
Case 4	**479,984**	**530,581**	**530,581**
Case 5	625,096	665,599	633,404
Case 6	**587,503**	**588,447**	**588,447**
Case 7	539,741	587,335	564,032
Case 8	**584,591**	**664,215**	**664,215**
Case 9	625,096	665,599	633,404
Case 10	1,111,752	1,219,555	1,141,881

4.5 Analysis of SH-Configuration

Case 1:

Table 4.38: Case SH-1

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	18	Sec	0.30
Time to Failure L_k	7	Min	0.30
Cycles to Exhaust C_k	150	Parts	0.30
Time to Repair R_k	150	Sec	0.30
Time to Replenish H_k	200	Sec	0.30
Results			
Initial throughput: ***477,761***			

	Bottleneck Index Method	Highest Utilization Method
Bottleneck	Machine C	Machine C
Mitigation	Add 10 buffers in front of Machine C and 15 behind it	Add 10 buffers in front of Machine C and 15 behind it
Throughput	***553,850***	***553,850***

Case 2:

Table 4.39: Case SH-2

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	16	Sec	0.33
Time to Failure L_k	106.5	Min	0.32
Cycles to Exhaust C_k	110	Parts	0.29
Time to Repair R_k	240	Sec	0.28
Time to Replenish H_k	180	Sec	0.33
Results			
Initial throughput: ***517,369***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine H	Machine H	
Mitigation	Add 7 buffers in front of Machine H and 3 behind it	Add 7 buffers in front of Machine H and 3 behind it	
Throughput	***566,800***	***566,800***	

Case 3:

Table 4.40: Case SH-3

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	10	Sec	0.31
Time to Failure L_k	86.5	Min	0.29
Cycles to Exhaust C_k	45	Parts	0.33
Time to Repair R_k	180	Sec	0.32
Time to Replenish H_k	100	Sec	0.29

Results		
Initial throughput: ***763,443***		
	Bottleneck Index Method	Highest Utilization Method
Bottleneck	Machine H	Machine E
Mitigation	Add 7 buffers in front of Machine H and 15 behind it	Add 10 buffers in front of Machine E and 18 behind it
Throughput	***857,098***	***795,458***

Case 4:

Table 4.41: Case SH-4

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	13.5	Sec	0.29
Time to Failure L_k	106.5	Min	0.28
Cycles to Exhaust C_k	70	Parts	0.32
Time to Repair R_k	120	Sec	0.32
Time to Replenish H_k	90	Sec	0.28
Results			
Initial throughput: ***878,667***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine H	Machine H	
Mitigation	Add 1 buffers in front of Machine H and 3 behind it	Add 1 buffers in front of Machine H and 3 behind it	
Throughput	***959,073***	***959,073***	

Case 5:

Table 4.42: Case SH-5

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	8	Sec	0.28
Time to Failure L_k	213	Min	0.31
Cycles to Exhaust C_k	30	Parts	0.29

Time to Repair R_k	46.5	Sec	0.33
Time to Replenish H_k	23.5	Sec	0.32
Results			
Initial throughput: ***1,565,133***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine E	Machine B	
Mitigation	Add Machine E	Add Machine B	
Throughput	***1,619,663***	***1,565,133***	

Case 6:

Table 4.43: Case SH-6

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	25	Sec	0.32
Time to Failure L_k	170	Min	0.33
Cycles to Exhaust C_k	130	Parts	0.31
Time to Repair R_k	80	Sec	0.28
Time to Replenish H_k	70	Sec	0.27
Results			
Initial throughput: ***427,741***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine C	Machine C	
Mitigation	Add Machine C	Add Machine C	
Throughput	***523,610***	***523,610***	

Case 7:

Table 4.44: Case SH-7

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	20	Sec	0.30
Time to Failure L_k	71	Min	0.29

Cycles to Exhaust C_k	90	Parts	0.33
Time to Repair R_k	220	Sec	0.29
Time to Replenish H_k	130	Sec	0.30
Results			
Initial throughput: ***564,081***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine E	Machine E	
Mitigation	Add 3 buffers in front of Machine E and 3 behind it	Add 3 buffers in front of Machine E and 3 behind it	
Throughput	***605,291***	***605,291***	

Case 8:

Table 4.45: Case SH-8

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	12	Sec	0.33
Time to Failure L_k	142	Min	0.28
Cycles to Exhaust C_k	100	Parts	0.32
Time to Repair R_k	65	Sec	0.32
Time to Replenish H_k	140	Sec	0.31
Results			
Initial throughput: ***856,211***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine C	Machine F	
Mitigation	Add 4 buffers in front of Machine C and 10 behind it	Add 15 buffers in front of Machine F and 5 behind it	
Throughput	***929,791***	***882,391***	

Case 9:

Table 4.46: Case SH-9

Configuration			
Attribute	Mean Value	Units	C.V. between processes

Processing Time P_k	15	Sec	0.31
Time to Failure L_k	94	Min	0.31
Cycles to Exhaust C_k	120	Parts	0.29
Time to Repair R_k	130	Sec	0.31
Time to Replenish H_k	100	Sec	0.29
Results			
Initial throughput: ***632,703***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine C	Machine C	
Mitigation	Add Machine C	Add Machine C	
Throughput	***705,078***	***705,078***	

Case 10:

Table 4.47: Case SH-10

Configuration			
Attribute	Mean Value	Units	C.V. between processes
Processing Time P_k	22	Sec	0.29
Time to Failure L_k	77.5	Min	0.33
Cycles to Exhaust C_k	880	Parts	0.31
Time to Repair R_k	160	Sec	0.33
Time to Replenish H_k	170	Sec	0.28
Results			
Initial throughput: ***466,609***			
	Bottleneck Index Method	Highest Utilization Method	
Bottleneck	Machine H	Machine H	
Mitigation	Add 3 buffers in front of Machine H and 6 behind it	Add 3 buffers in front of Machine H and 6 behind it	
Throughput	***501,315***	***501,315***	

Ten sets of lines with SH configuration were studied in this section. It was found that the Bottleneck Index Method identified the same bottleneck as that of the Highest Utilized Machine Method on seven occasions. The Bottleneck Index Method yielded

better results in terms of throughput of the line for the remaining three cases. The results are presented in Table 4.48. The scenarios where both methods found the same bottleneck are bolded. The resource state graphs of all case studies have been presented in the appendix.

Table 4.48: Summary of SH-Configuration Results

Experiment #	Throughput		
	Base Scenario	Bottleneck Index Method	Highest Utilized Machine Method
Case 1	**477,761**	**553,850**	**553,850**
Case 2	**517,369**	**566,800**	**566,800**
Case 3	763,443	857,098	795,458
Case 4	**878,667**	**959,073**	**959,073**
Case 5	1,565,133	1,619,663	1,565,133
Case 6	**427,741**	**523,610**	**523,610**
Case 7	**564,081**	**605,291**	**605,291**
Case 8	856,211	929,791	882,391
Case 9	**632,703**	**705,078**	**705,078**
Case 10	**466,609**	**501,315**	**501,315**

The simulation models for the case studies presented above were built with the help of the Excel template developed as a part of this thesis. This reduced the modeling efforts to a great extent; wherein the simulation model was built at a 'click of a button'. The algorithm that was developed to detect the bottleneck in any serial production system and give the simulation analyst a direction in an effort to increase throughput with the consideration of economics was tested and verified under different conditions.

It was found that out of the forty scenarios that were analyzed in this thesis, there were only twenty-five cases in which the highest utilized machine was actually the bottleneck resource (both Bottleneck Index and Highest Utilized Machine Methods detected the same bottleneck in these twenty-five cases). We use the mitigation procedure developed in this thesis for bottlenecks detected by Bottleneck Index and Highest

Utilized Machine Methods. For the fifteen cases in which the methods identified different bottlenecks, our technique of detecting the bottleneck and mitigating the same yielded better throughput. This difference in throughput between the lines after mitigation gives us evidence that our method of identifying the bottleneck of a serial production system is more accurate than the traditional method. We analyzed the resource state graphs of these fifteen cases to find out the conditions in which Bottleneck Index Method chose a different resource rather than choosing the one with the highest utilization. It is listed as follows:

- It occurs only when there are some machines in the line with very similar processing times as that of the highest utilized machine and
- In the absence of station breakdown (failure and exhaust) both the methods always pick the same bottleneck resource

From the fifteen cases in which the methods identified different bottlenecks, it was found that on an average there were around 2.93 resources in those systems whose average percentage of utilization was 2.27% lesser than that of the highest utilized machine. A mathematical explanation to why both methods pick the same bottleneck resource in the absence of breakdowns is that our method detects the bottleneck based on the time a resource spends in idle and blocked states. A machine can exist only in three states namely, busy, idle and blocked in the absence of breakdowns. The % of time a resource spends in busy state is dictated by its processing time. So mathematically, a machine with highest utilization in the absence of station breakdowns will spend the least % of time in idle and blocked states. But in the presence of machine breakdowns, factors like operator interference affect the % of time spent by a resource in idle and blocked

states and our method might choose a different resource other than the highest utilized machine to be the bottleneck resource.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This thesis dealt with the problem of dynamic generation of simulation models and analysis of serial production systems which is characterized by capacitated buffers, stochastic processing times, unreliable machines, rework loops, maintenance and operator issues. This is a significant issue in the field of manufacturing where the decision makers are often faced with the task of accessing the throughput of a production line, thereby allocating resources and buffers accordingly to meet the annual requirement of the plant. This process has to be carried out in relatively short time with no compromises in accuracy of the estimates.

In this thesis, a VBA project was undertaken in an effort to automate the process of simulation model building, which will enable decision makers with working knowledge of discrete event simulation to carryout case studies with minimal efforts. The result of this VBA project was an Excel template, where in the analyst could specify the processing parameters for each station and the simulation model of the line will be dynamically generated in Arena 7.01. This reduced the simulation modeling efforts to a great extent.

We have developed a technique (Bottleneck Index Method) to detect the bottleneck resource in any serial production system by effectively using the resource state statistics, which records the % of time spent by a station in the seven different states

discussed in this thesis. This technique of ours is compared to that of the traditional bottleneck detection method, where in the resource with the highest utilization is considered to be the bottleneck. We analyze ten station lines and fifty station lines with low and high variability in processing parameters. The production lines analyzed in this thesis can be thus be categorized into: i) Long line with low variability in processing parameters , ii) Long line with high variability in processing parameters, iii) Short line with low variability in processing parameters and iv) Short line with high variability in processing parameters. We have conducted ten sets of experiments for each category. As a part of this thesis, we have developed a procedure to curb the bottleneck of a system after the detection of the same. We use this mitigation procedure for bottlenecks detected by Bottleneck Index and Highest Machine Utilization Methods. Out of the forty cases that were analyzed, both Bottleneck Index and Highest Machine Utilization Methods detected the same bottleneck machine in twenty-five cases. For the rest of the cases (which is about 37.5%) it was shown that our technique of detecting the bottleneck and mitigating the same yielded better throughput. The ability to rightly identify the bottleneck is demonstrated in this thesis. It was also shown in the case studies that the throughput of a system could be improved appreciably by proper adaptation of the mitigation techniques suggested in this thesis. Any marginal increase in the throughput of a line could result in saving millions of dollars. Proper application of the proposed methodologies will help management achieve the desired production rate with minimal outlay.

Buffer allocation is an area which needs further investigation. Currently, the algorithm proposed in this thesis analyses output from a simulation model and may

suggest changes like adding buffers to a resource. But there is no quantification of the size of the buffer that is to be added. The analyst has to guess estimate this value by looking at the % of time that resource has spent on different states. If the buffers added are surplus or deficit then there is a chance that the study will undergo an extra iteration. Another algorithm which will quantify the amount of buffers to be added can be developed and it can be integrated into the techniques proposed in this thesis. This way time taken for the complete study can be further reduced.

Another potential extension to this research will be to make the simulation model accessible through the World Wide Web where in the user can build the model, run the same and analyze the system from a remote server. This can be accomplished with the help of XML. Standards like B2MML can be used in developing the schemas for XML; thereby the sender and receiver will have the same expectations of the data that is being shared.

REFERENCES

Blumenfeld, D. E. (1990). A simple formula for estimating throughput of serial production lines with variable processing times and limited buffer capacity, *Int. J. Prod. Res*., Vol. 28, No. 6, 1163-1182.

Burman, M.H. (1995). New results in flow line analysis, MIT, Thesis.

Chen, C.T., Yuan, J., and Lin, M.H. (2003).Transient Throughput Analysis using Sample Path Method for Serial Unreliable Work Centers, *International Journal of The Computer, The Internet and Management*, Vol. 11, No.1, 30 – 41.

Chiang, S.Y., Kuo, C.T., and Meerkov, S.M. (2000). DT-Bottlenecks in Serial Production Lines: Theory and Application, *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 5, 567- 579.

Chiang, S.Y., Kuo, C.T. and Meerkov, S.M. (2001). C-Bottlenecks in Serial Production Lines" Identification and Application, *Mathematical Problems in Engineering* Vol. 7, 543-578.

Choong, Y.F., and Gershwin, S.B. (1986). A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times, MIT Laboratory for Information and Decision Systems Report LIDS-P-1476. 1-36.

Cochran, J. K., Erol, R. (2001). Performance modelling of serial production lines with inspection/repair stations, *Int.J. Prod. Res*., Vol. 39, No. 8, 1707-1720

Dallery, Y. and Gershwin, S.B. (1992). Manufacturing flow line systems: a review of models and analytical results. *Queuing Systems*, 12, 3-94.

Doss, D.L. and Ulgen, O.M. (2004). A case for generic, custom-designed simulation applications for material handling and manufacturing industries. *Brooks Automation's Worldwide Automation Symposium*, 1-5.

Enginarlar, E., Li, J., Meerkov, S.M., and Zhang, R.Q. (2002). Buffer capacity for accommodating machine downtime in serial production lines*, Int. J. Prod. Res*., Vol. 40, No. 3, 601- 624.

Farrington, P.A., Rogers, J.S., Schroer, B.J., Swain, J.J., and Evans, J.J. (1995). Simulation environment for electronics manufacturing. *In* p*roceedings of 1995 Winter Simulation Conference*, 917-924.

Fishman, G. S. (2001). Discrete Event Simulation: Modeling, Programming and Analysis, Springer-Verlag, New York.

Gershwin, S. B. (1983). An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage Space and Blocking*,* MIT Laboratory for Information and Decision Systems Report LIDSP-P1309.

Harris, J.H., and Powell, S.G. (1999), An algorithm for optimal buffer placement in reliable serial lines, *IIE Transactions 31*, 287 – 302.

Hopp, W. J. and Spearman, M. L. (2000). Factory Physics. McGraw Hill.

Jadhav, P.D., Smith, J.S., 2005, Analyzing Printed Circuit Board Assembly Lines Using A PCB Assembly Template. In *Proceedings of the 2005 Winter Simulation*

Conference, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 1335-1342.

Jadhav, P.D. (2005), Simulation modeling and analysis of printed circuit board assembly lines, Auburn University, Thesis.

Law, A.M., and Kelton, W.D. (2000). Simulation Modeling and Analysis. McGraw Hill.

Lawrence, S.R., Buss, A.H. (1994). Shifting production bottlenecks: Causes, Cures and Conundrums, *Production and Operations Management*, Vol. 3 No. 1, 22-38.

Li, J. (2005). Overlapping Decomposition: A System-Theoretic Method for Modeling and Analysis of Complex Manufacturing Systems. *IEEE Transactions on Automation Science and Engineering* Vol. 2, No. 1, 40-53.

McLean, C., and Leong, S. (2001). The expanding role of simulation in future manufacturing, *Proceedings of the 2001 Winter Simulation Conference*, Eds. B.A. Peters, J.S. Smith, D.J. Medeiros and M.W. Rohrer. 1478 - 1486.

Moss, H.K. and Yu, W.B. (1999). Toward the estimation of bottleneck shiftiness in a manufacturing operation, *Production and Inventory Management Journal*; 40, 2; 53-58.

Mukkamala, P. S., Smith, J.S., Valenzuela, J. F., 2003, Designing Reusable Simulation Modules For Electronics Manufacturing Systems. In *Proceedings of the 2003 Winter Simulation Conference*, eds. S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice, 1281-1289.

Mukkamala, P. S. (2003), Design of a template in simulation for various machines in electronics assembly and it's implementation in the simulation package Arena, Auburn University, Thesis.

Paik, C.H., Kim, H.G. and Cho, H.S. (2002). Performance analysis for closed-loop production systems with unreliable machines and random processing times, *Computers & Industrial Engineering* 42, 207- 220.

Papadopoulos, H.T., Vidalis, M.I. (2001). A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines, Computers & Industrial Engineering 41, 261 – 277.

Powell, S.G., and Pyke, D.F. (1998), Buffering unbalanced assembly systems, *IIE Transactions 30*, 55 – 65.

Reiser, M., and Lavenberg, S.S. (1980). Mean-value analysis of closed multichain queueing networks. Journal of the ACM, 27 (2), 312 - 322

Roser, C., Nakano, M., Tanaka, M. (2001). A practical bottleneck detection method, *Proceedings of the 2001 Winter Simulation Conference,* B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds., 949-953.

Roser, C., Nakano, M., Tanaka, M., (2003). Buffer allocation model based on a single simulation, *Proceedings of the 2003 Winter Simulation Conference,* S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., 1238-1246.

Shi, L., and Men, S., (2003). Optimal buffer allocation in production lines, *IIE Transactions 35*, 1–10

Snodgrass, E., (1994). Beyond the Basics of Reengineering: Survival Tactics for the 90's. Quality Resources, White Plains, New York, USA.

Tempelmeier, H. and Burger, M. (2001), Performance evaluation of unbalanced flow lines with general distributed processing times, failures and imperfect production*, IIE Transactions* 33, 293 – 302.

Yamashita, H., and Alitok, T. (1998), Buffer capacity allocation for a desired throughput in production lines, *IIE Transactions 30*, 883 – 889.

APPENDICES

APPENDIX I

RESOURCE STATE GRAPHS FOR LL-CONFIGURATION

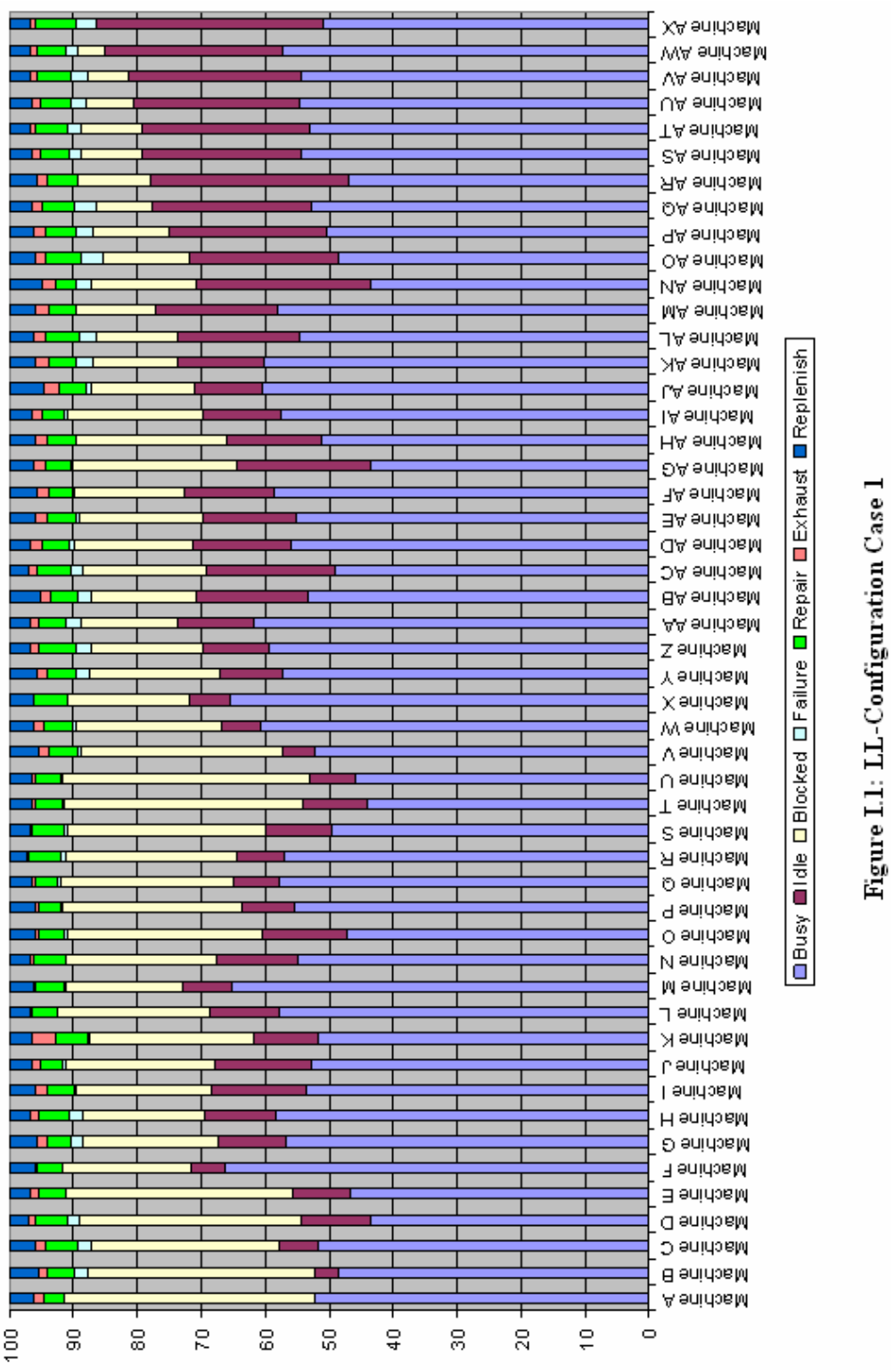

Figure I.1: LL-Configuration Case 1

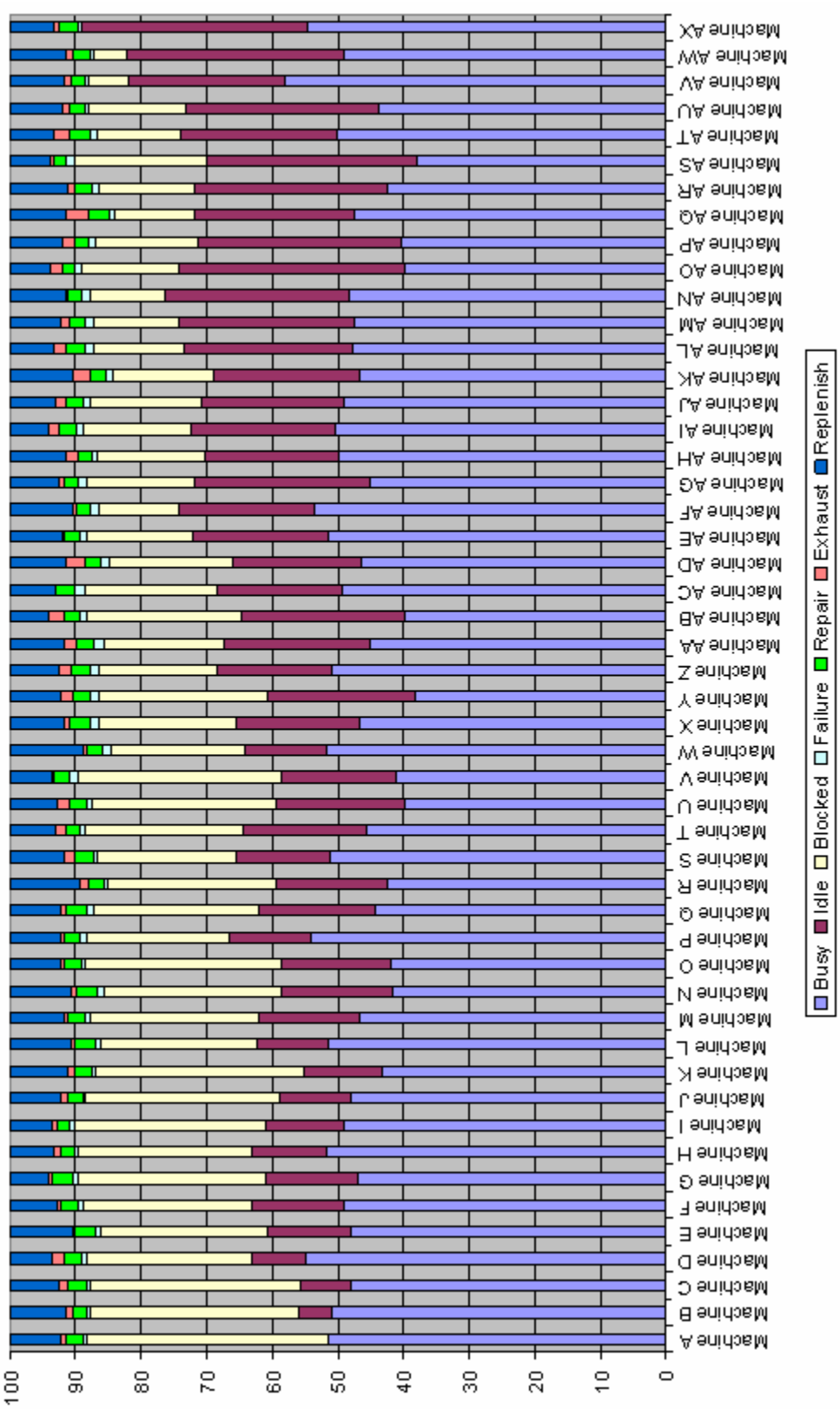

Figure I.2: LL-Configuration Case 2

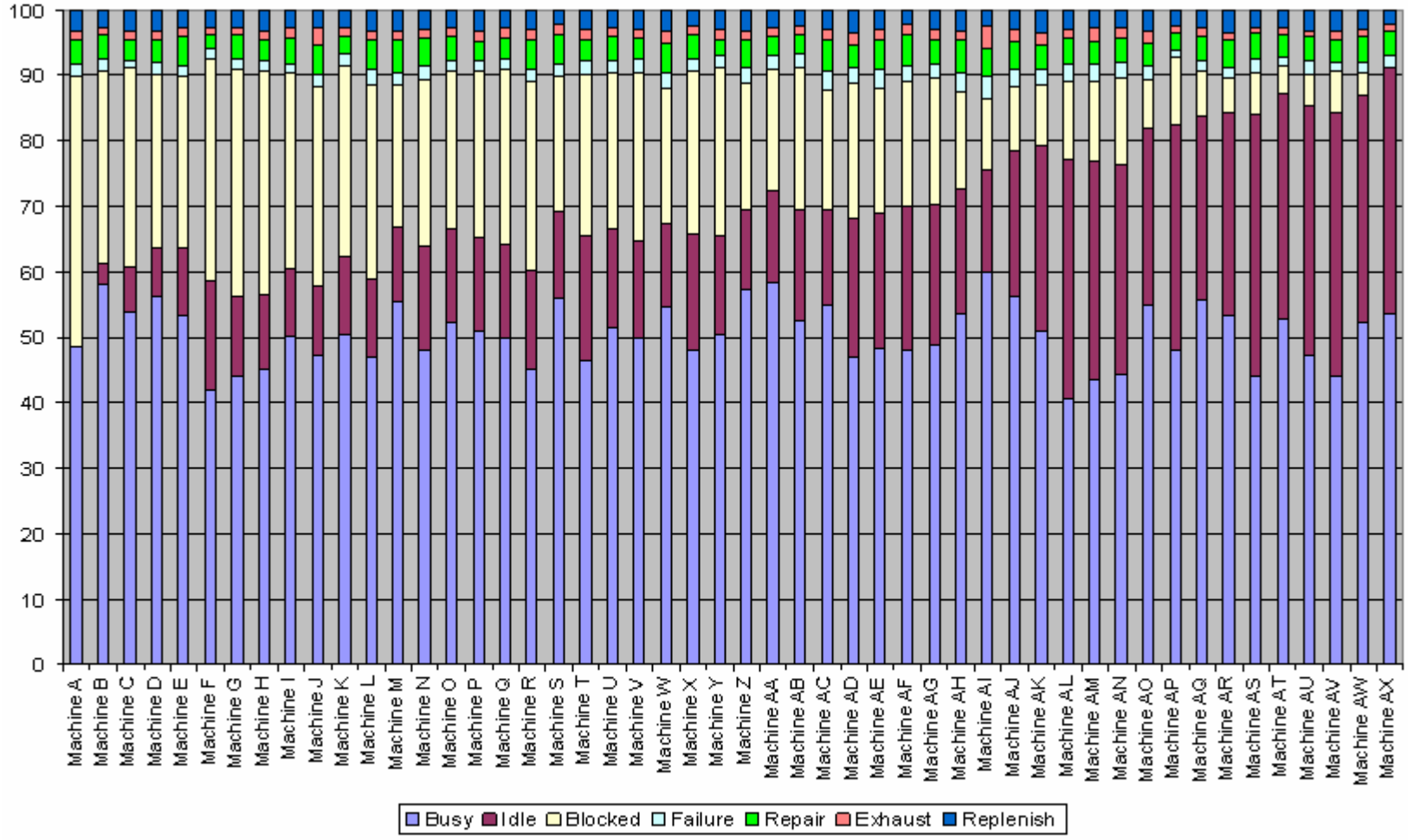

Figure I.3: LL-Configuration Case 3

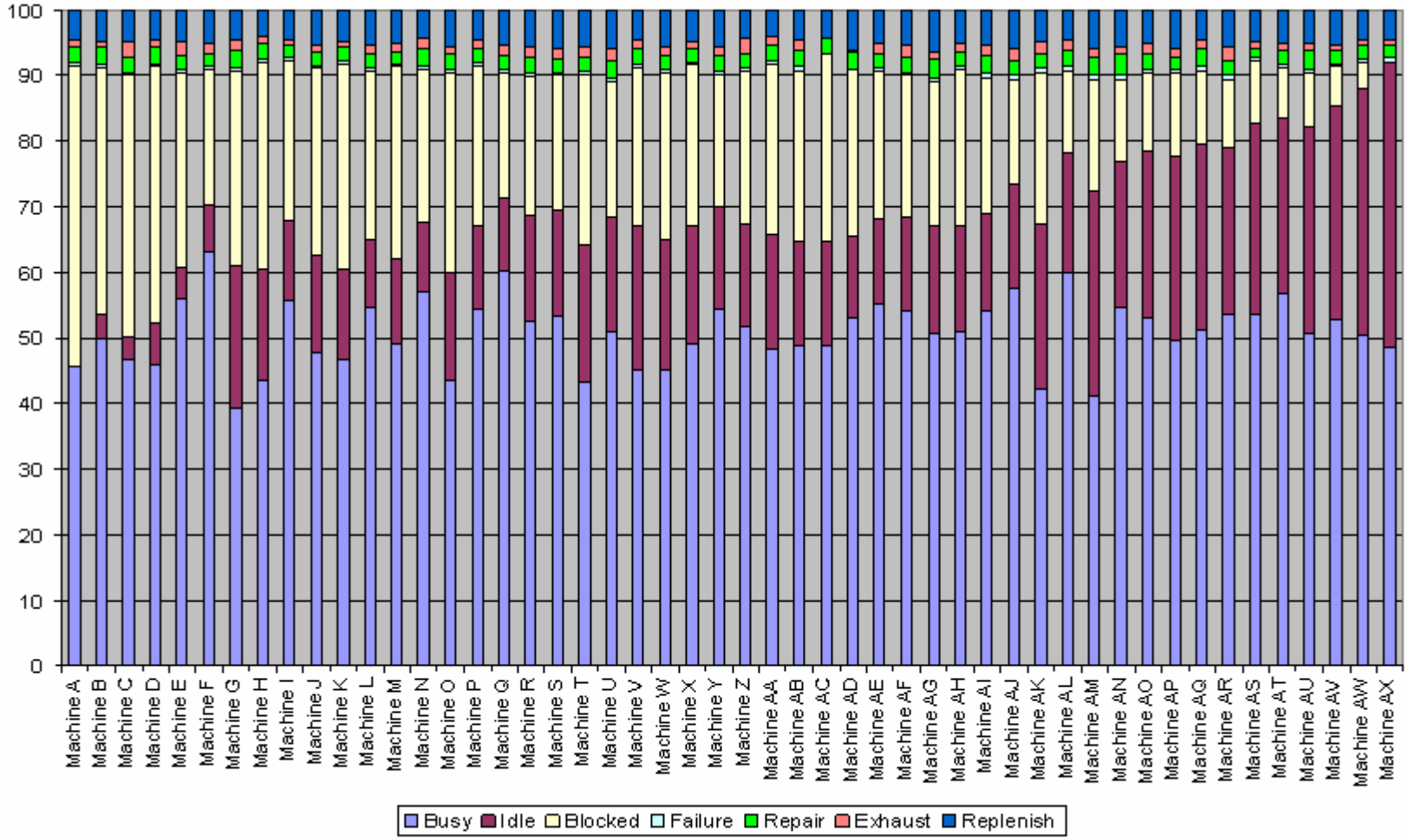

Figure I.4: LL-Configuration Case 4

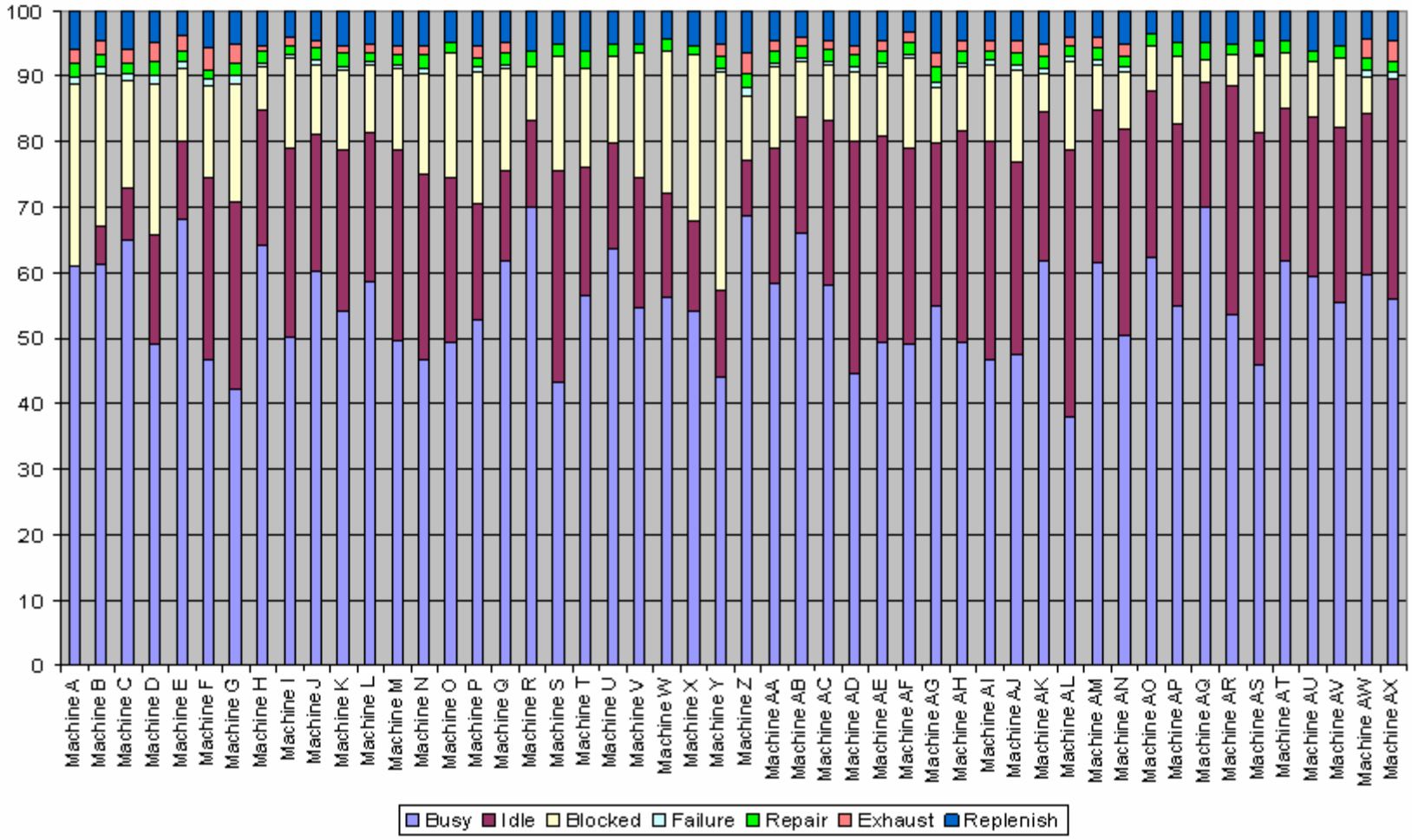

Figure I.5: LL-Configuration Case 5

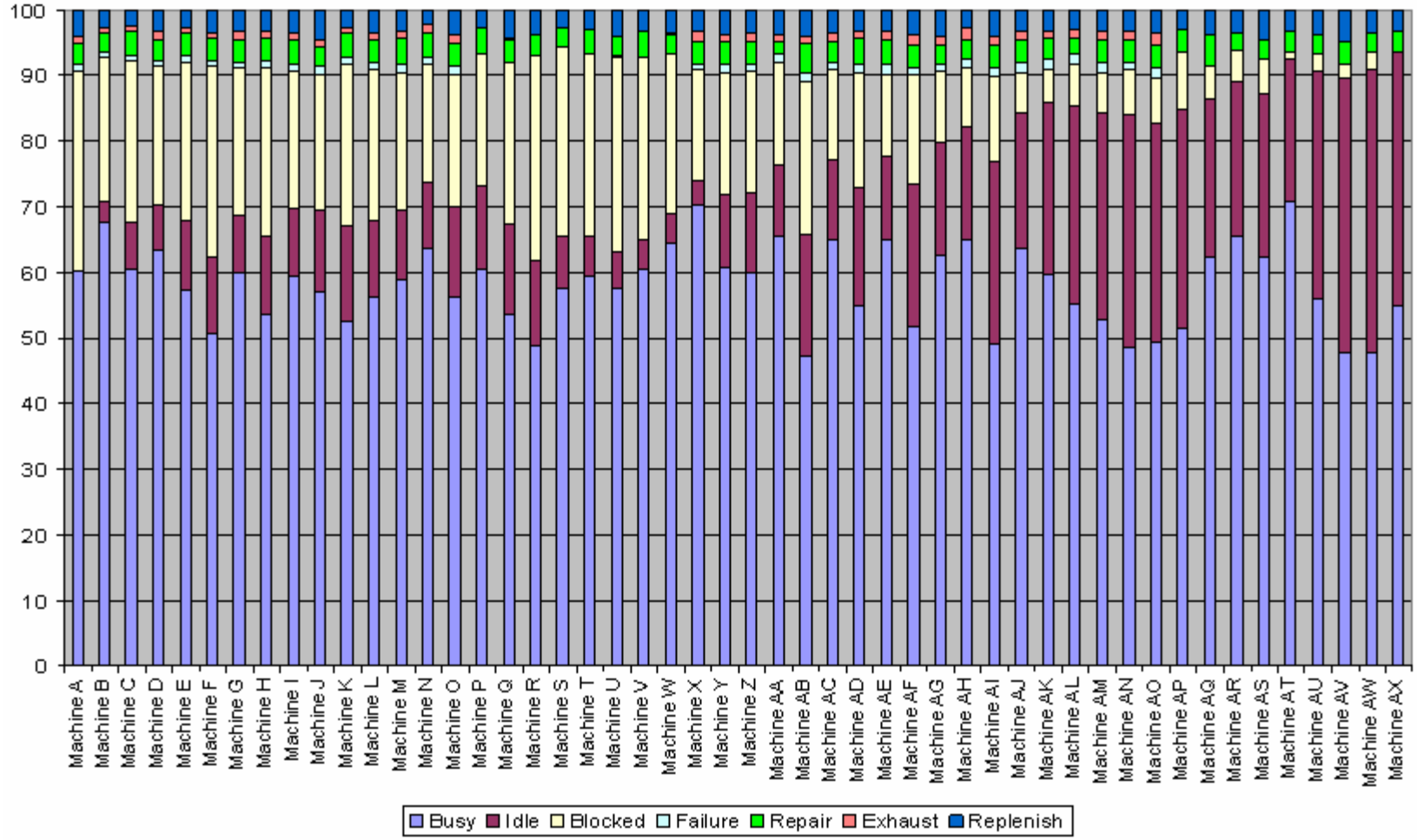

Figure I.6: LL-Configuration Case 6

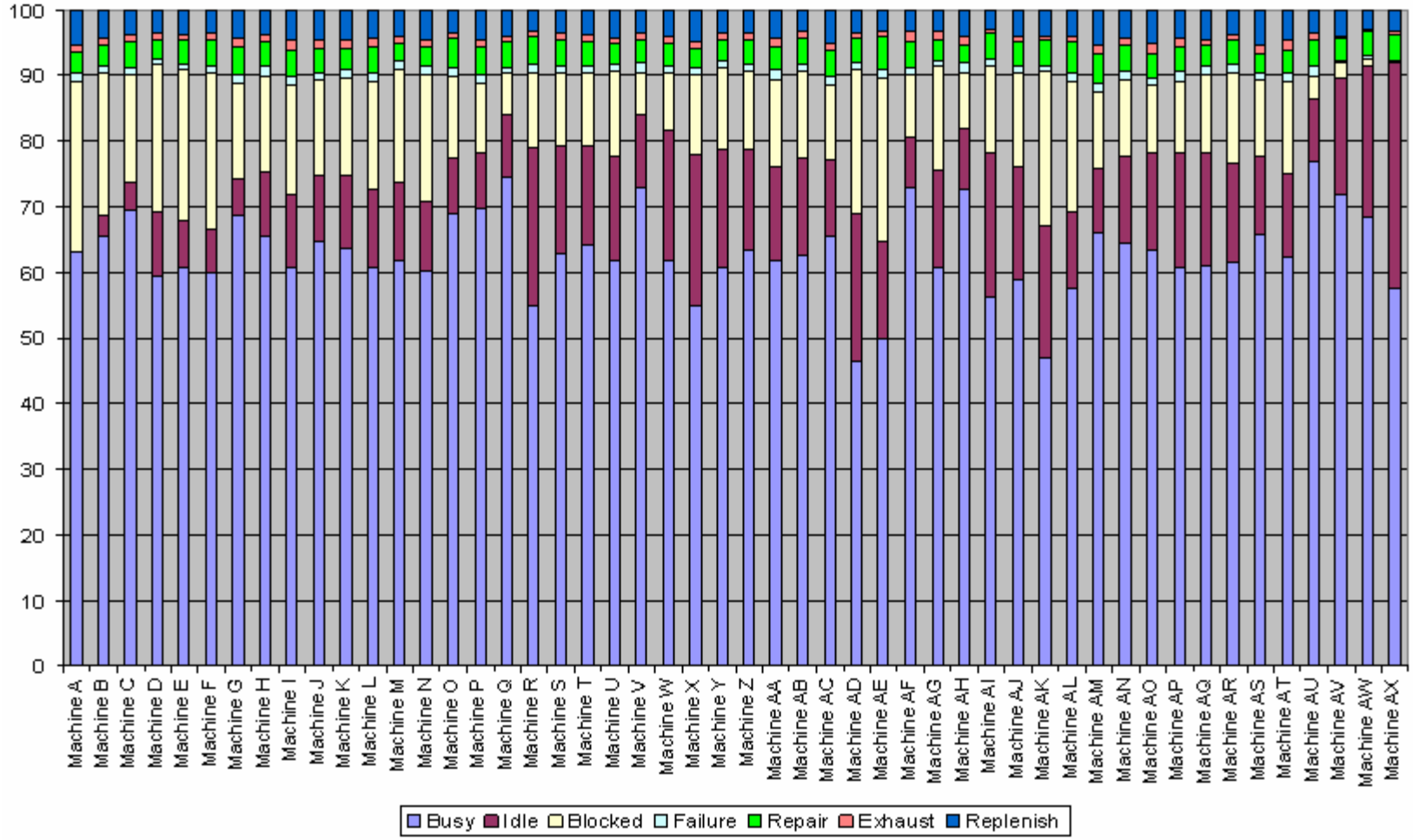

Figure I.7: LL-Configuration Case 7

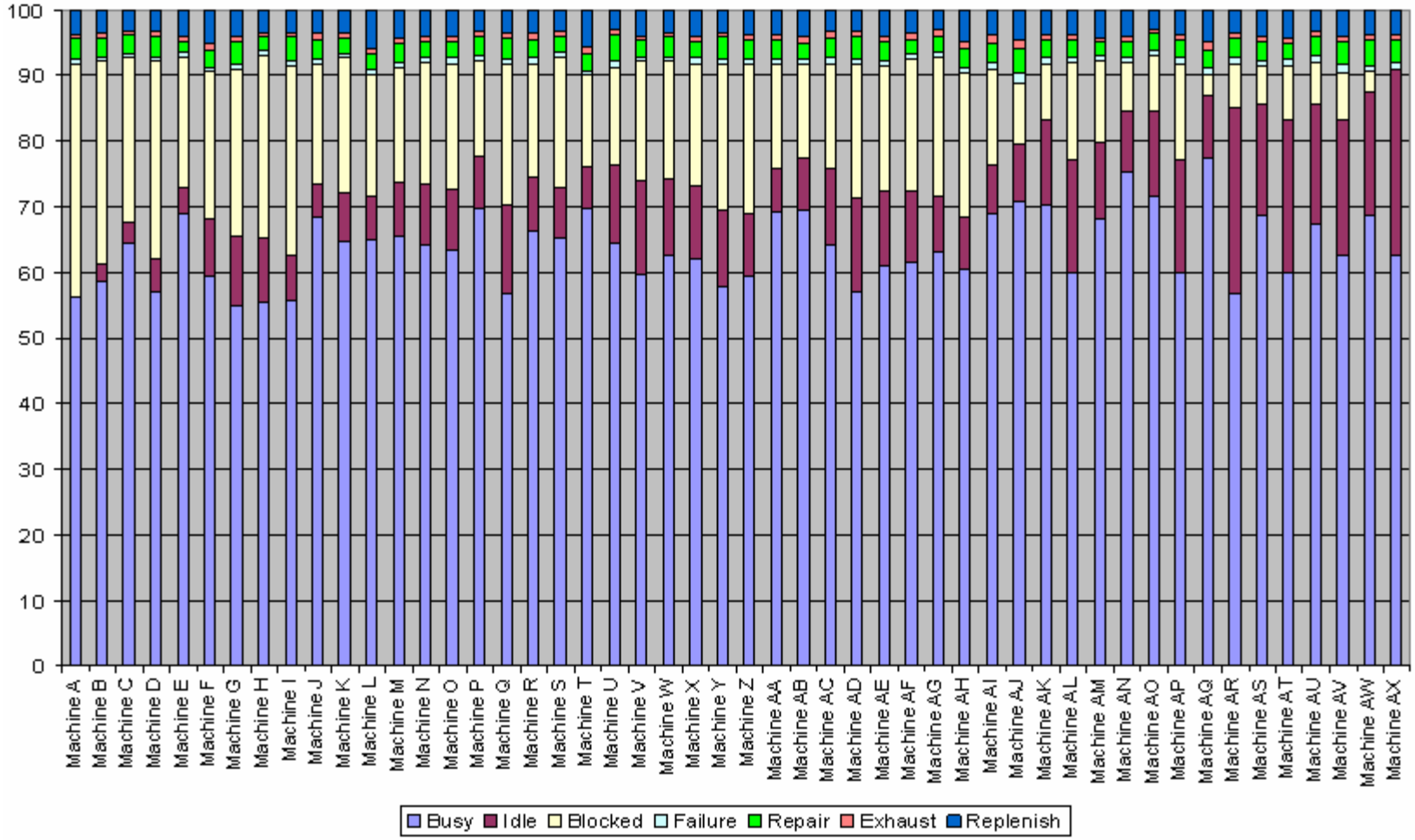

Figure I.8: LL-Configuration Case 8

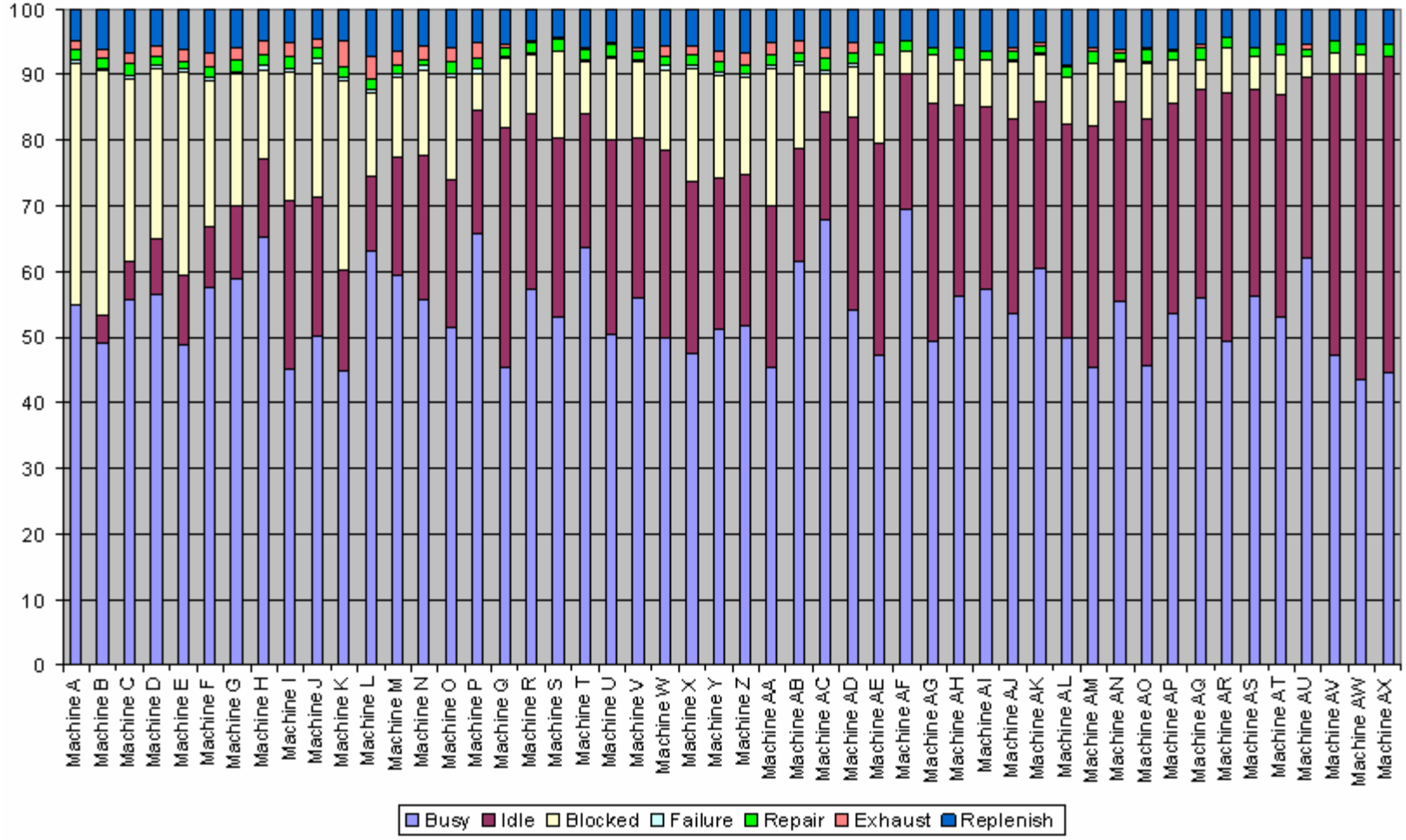

Figure I.9: LL-Configuration Case 9

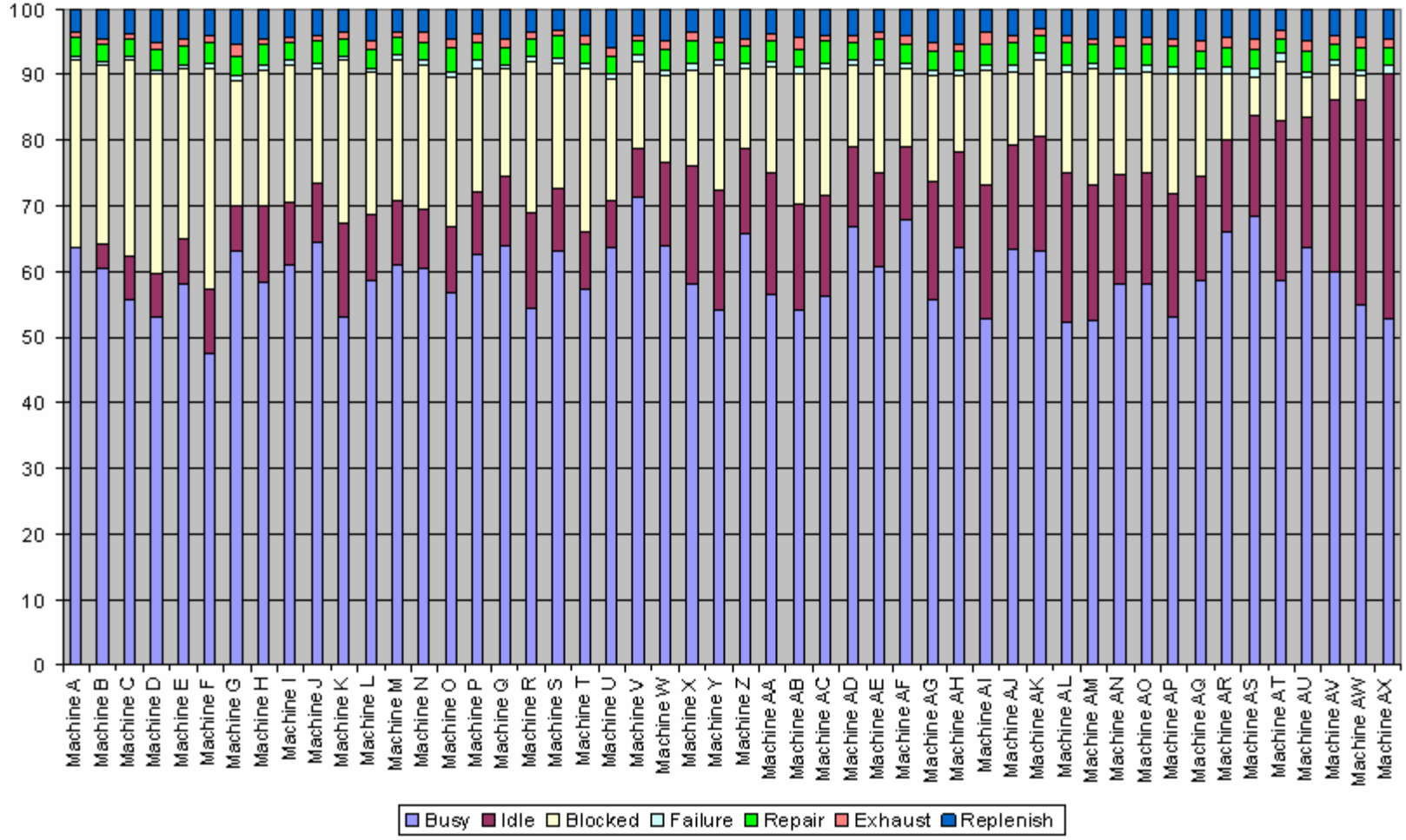

Figure I.10: LL-Configuration Case 10

APPENDIX II

RESOURCE STATE GRAPHS FOR LH-CONFIGURATION

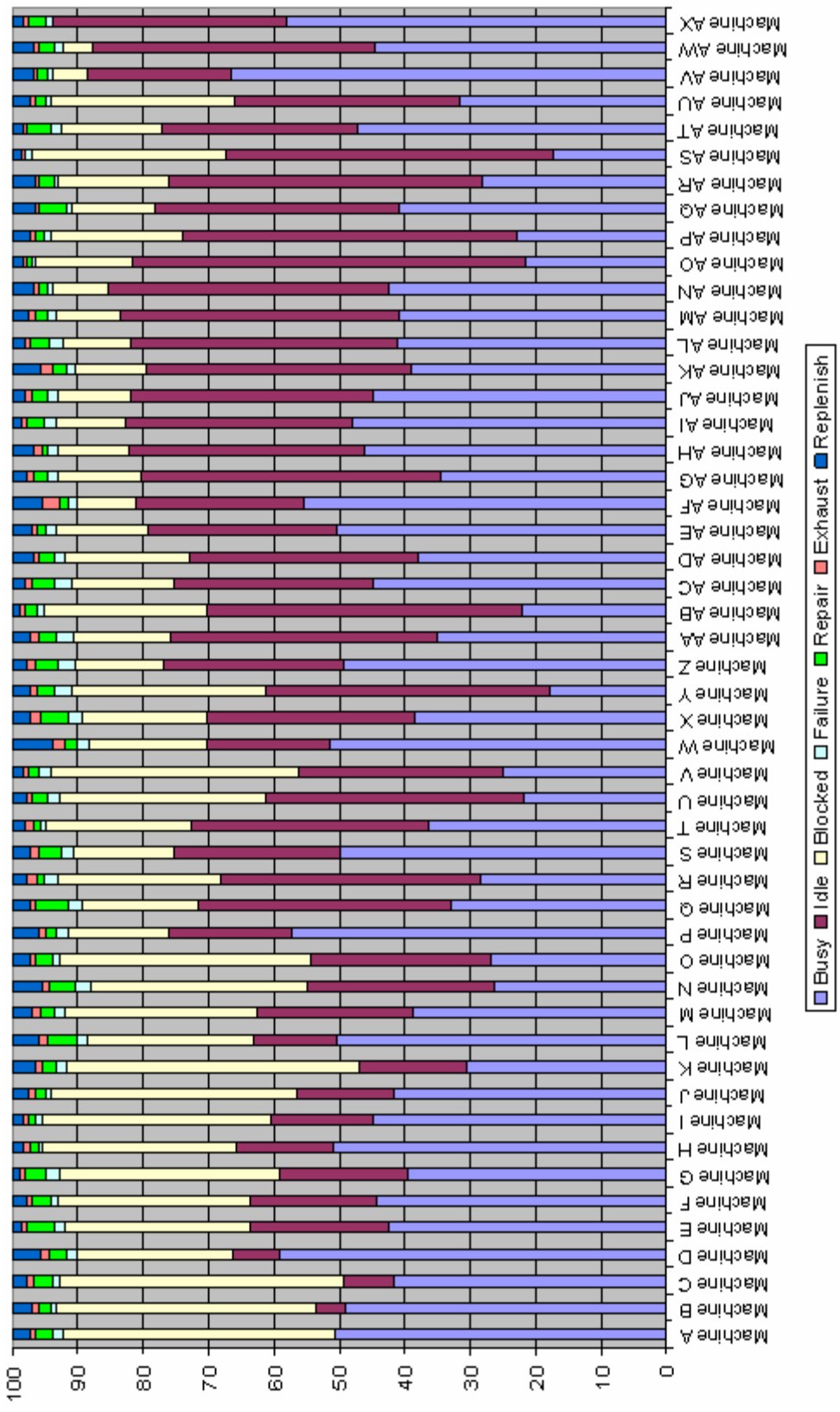

Figure II.1: LH-Configuration Case 1

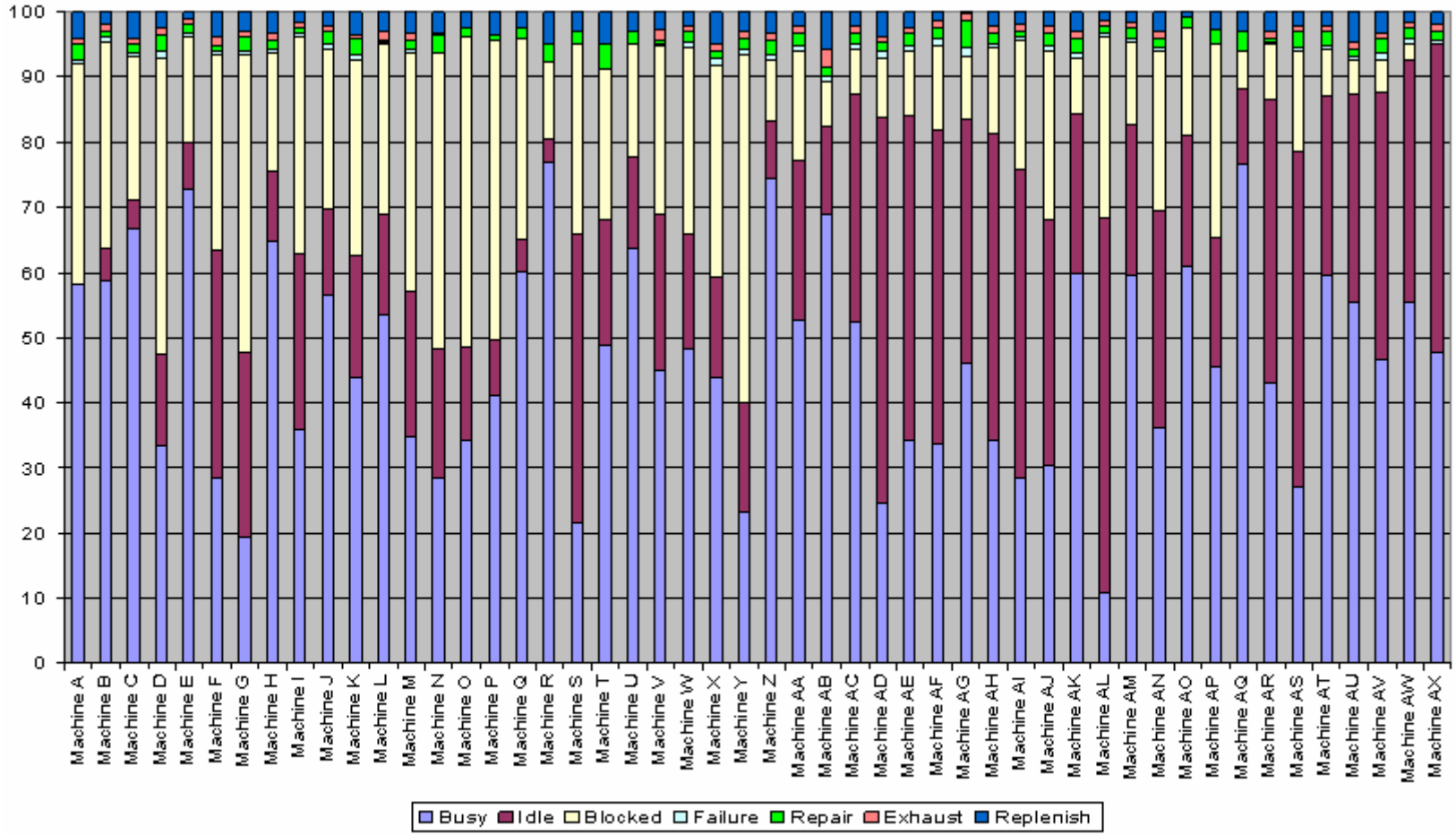

Figure II.2: LH-Configuration Case 2

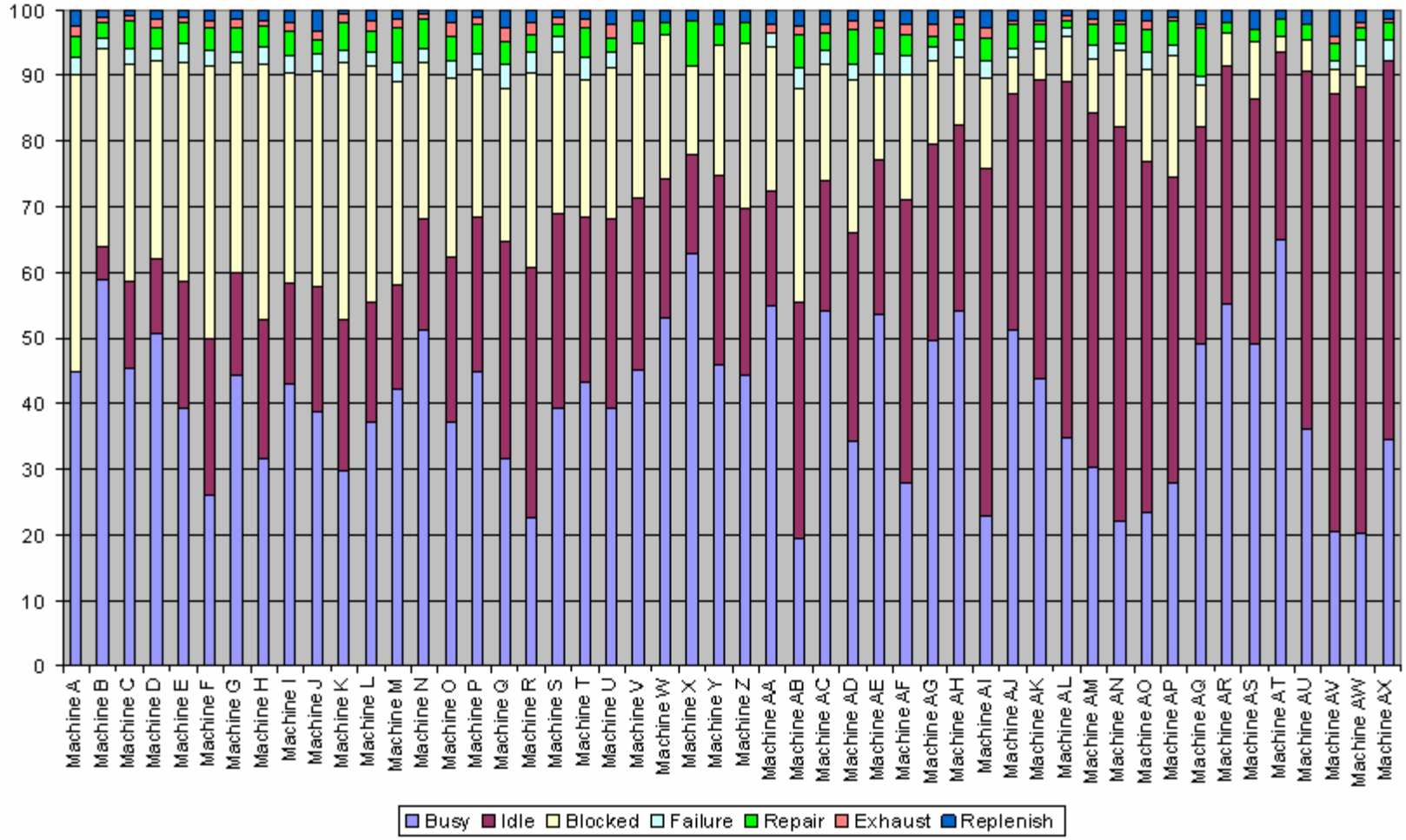

Figure II.3: LH-Configuration Case 3

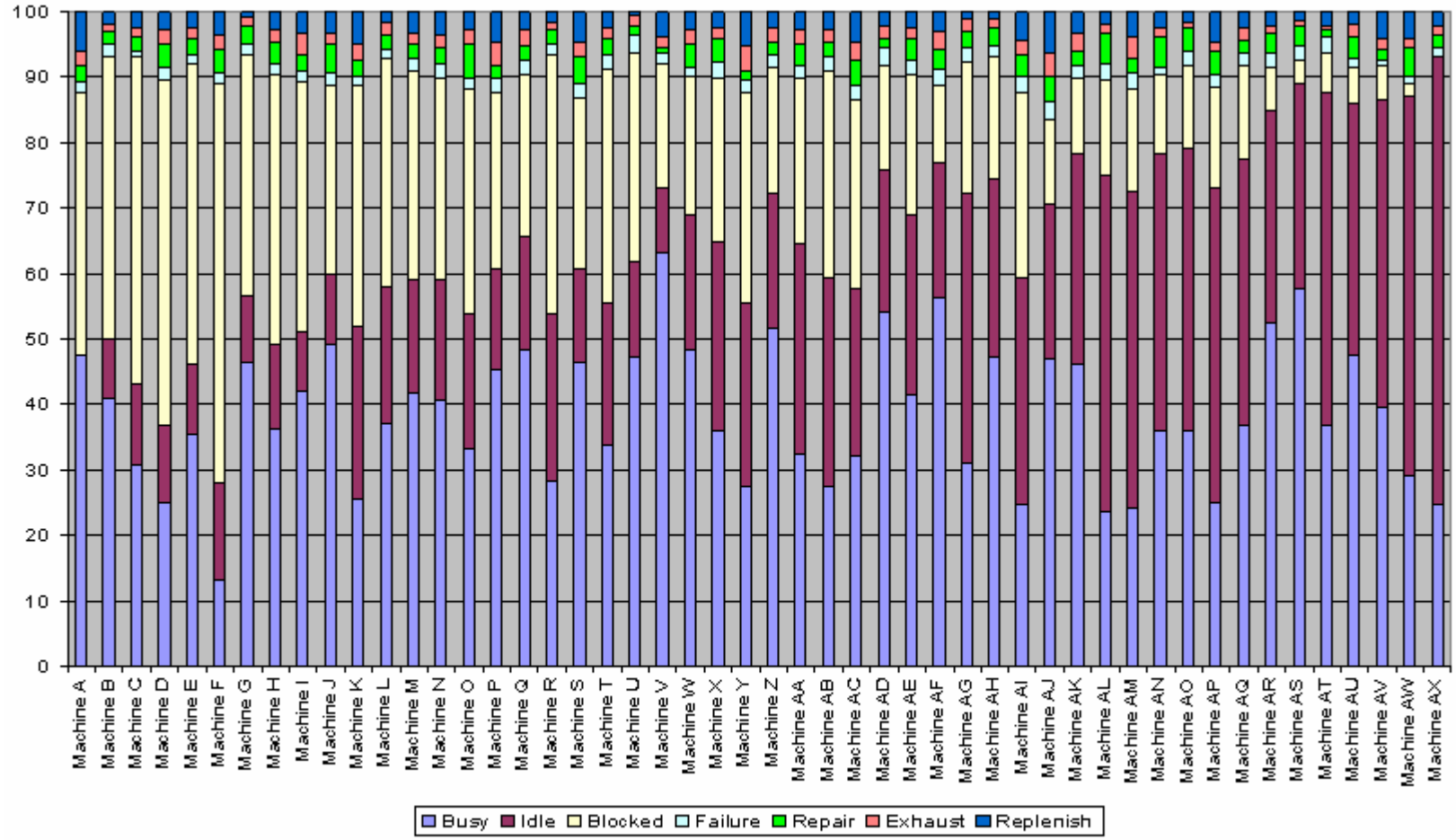

Figure II.4: LH-Configuration Case 4

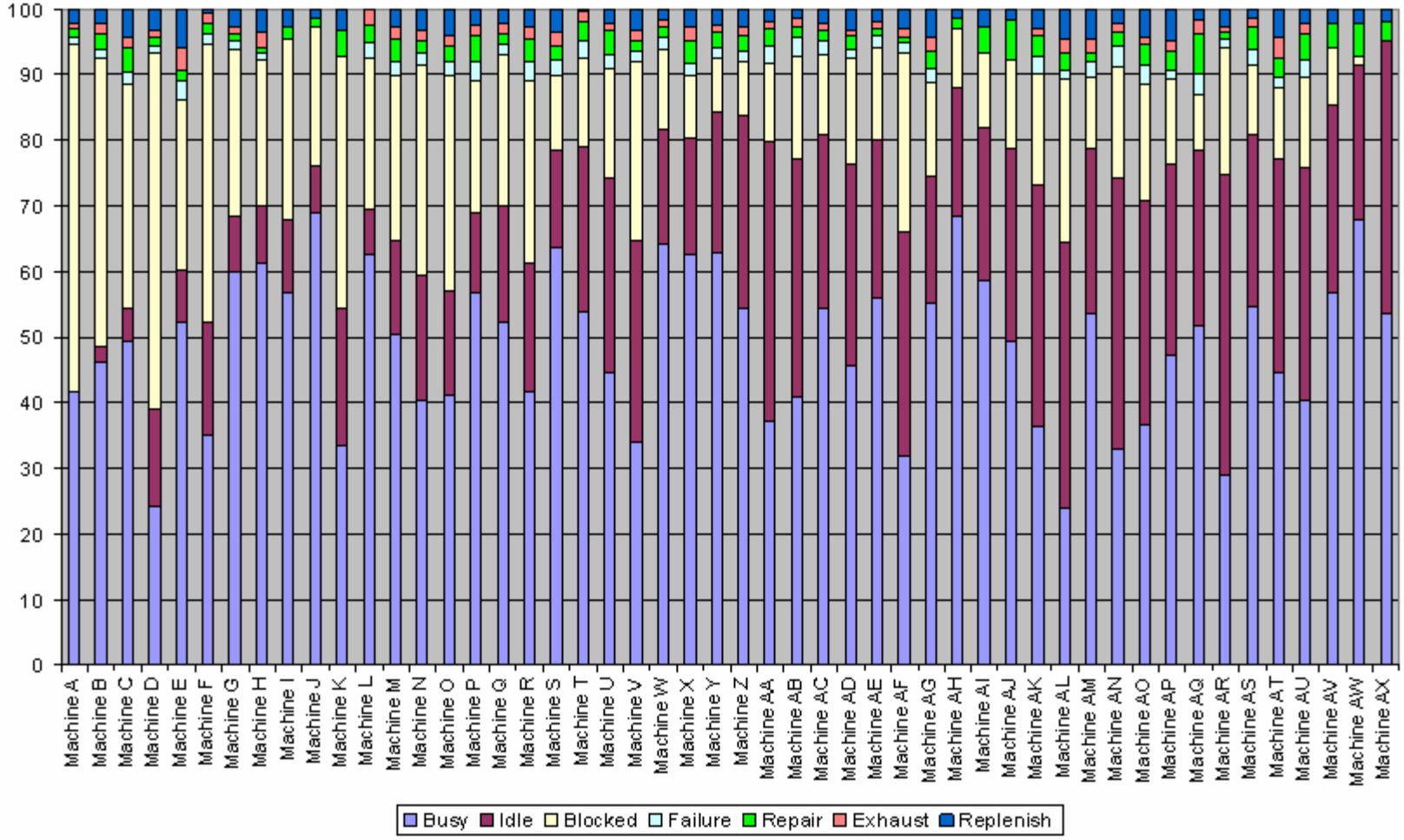

Figure II.5: LH-Configuration Case 5

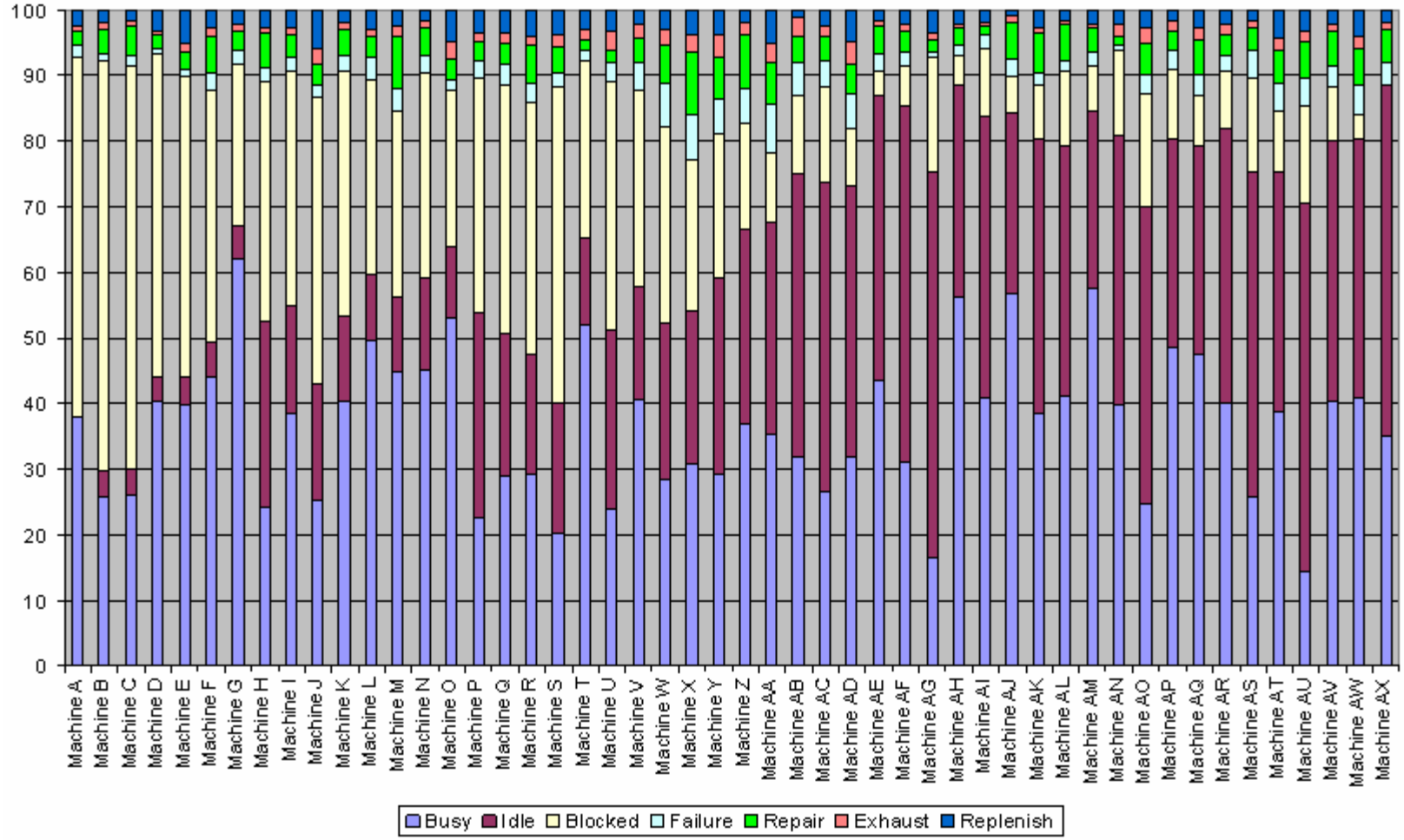

Figure II.6: LH-Configuration Case 6

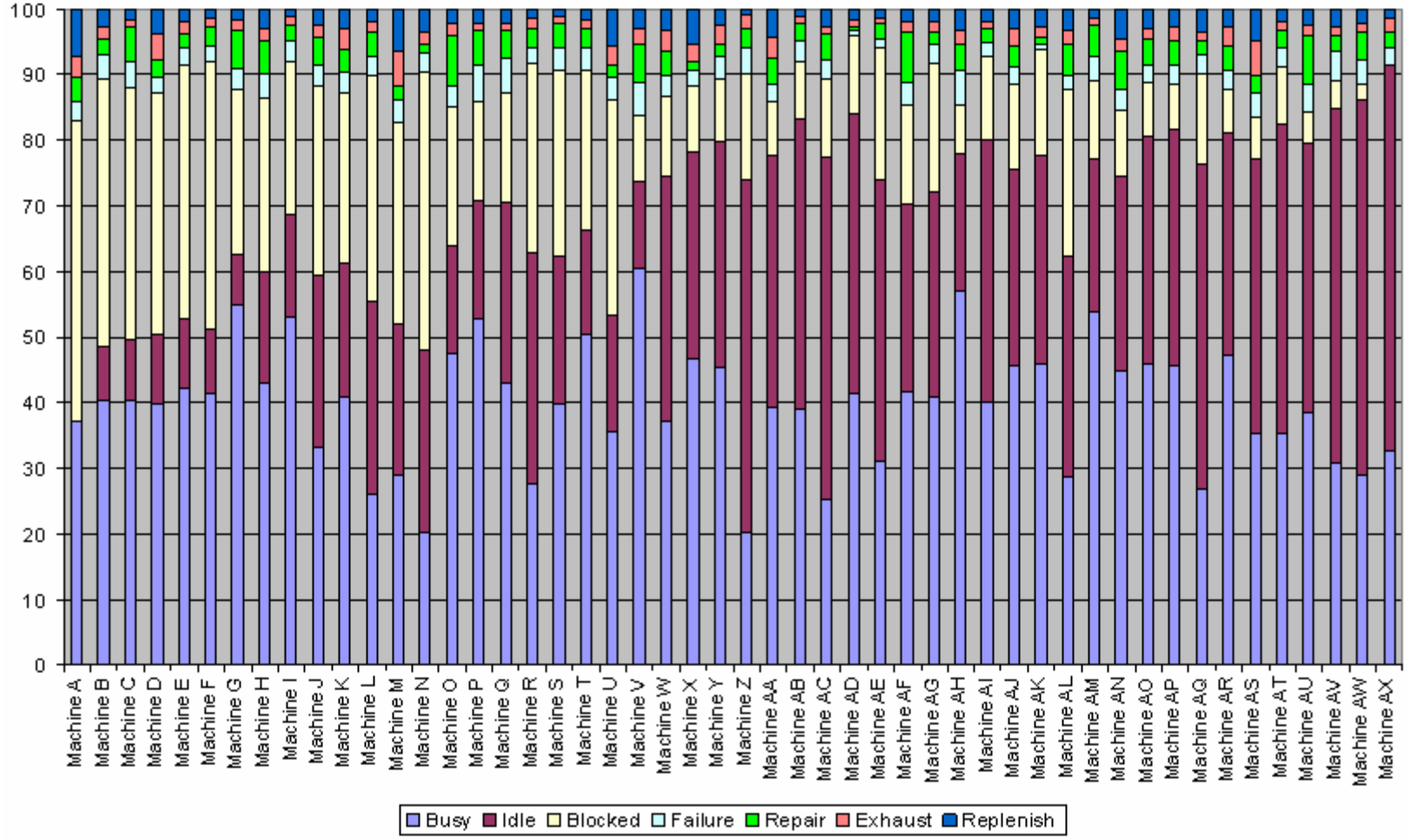

Figure II.7: LH-Configuration Case 7

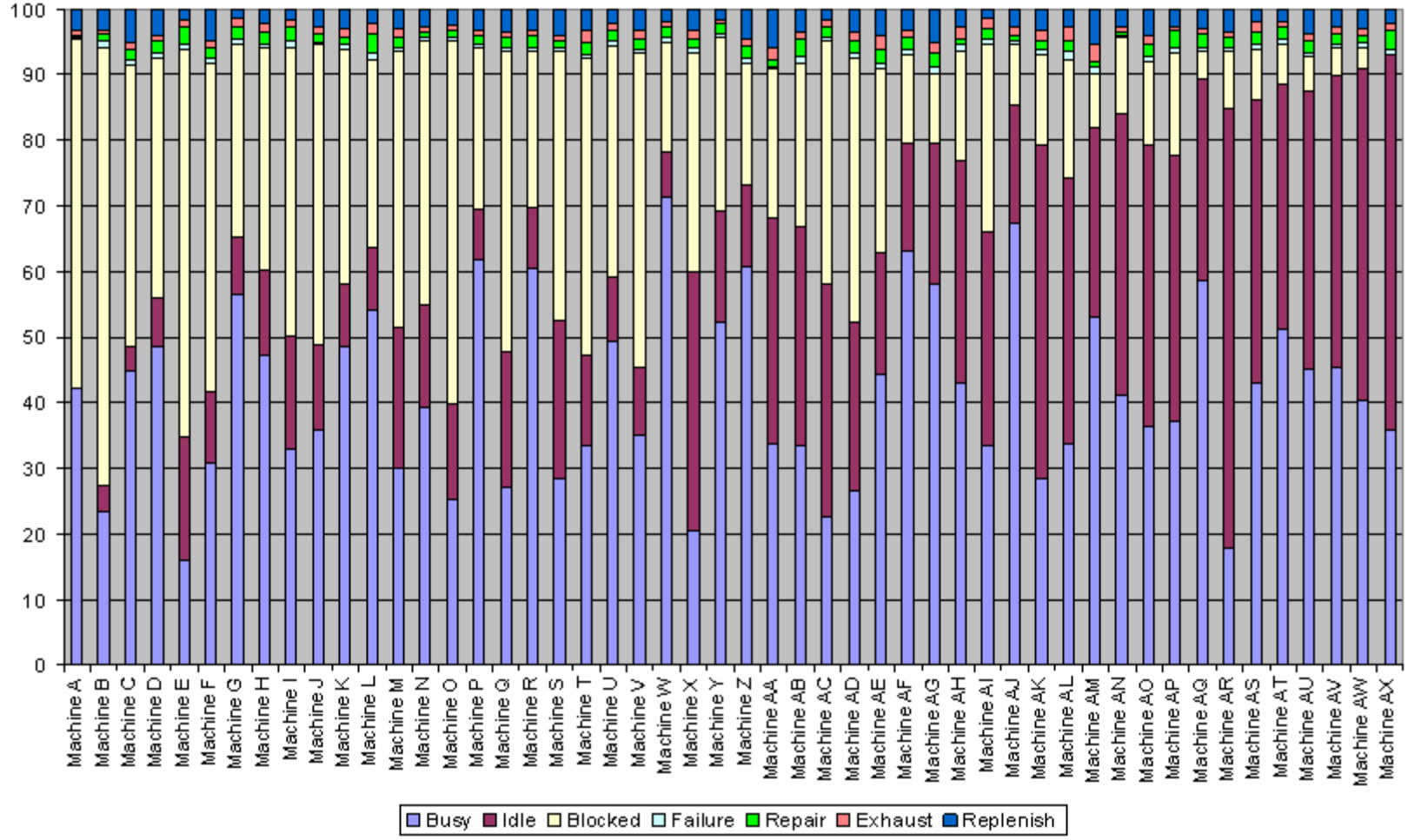

Figure II.8: LH-Configuration Case 8

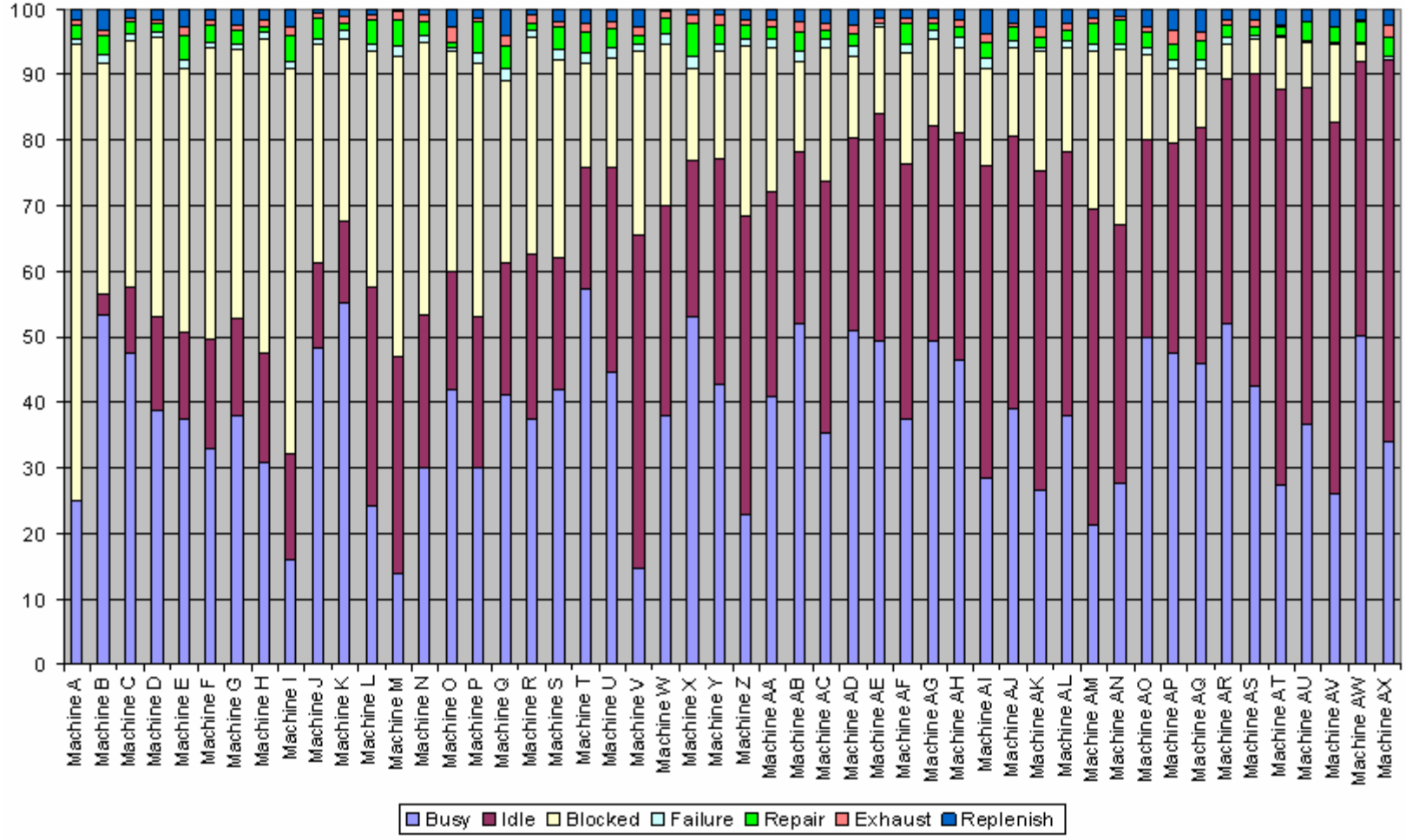

Figure II.9: LH-Configuration Case 9

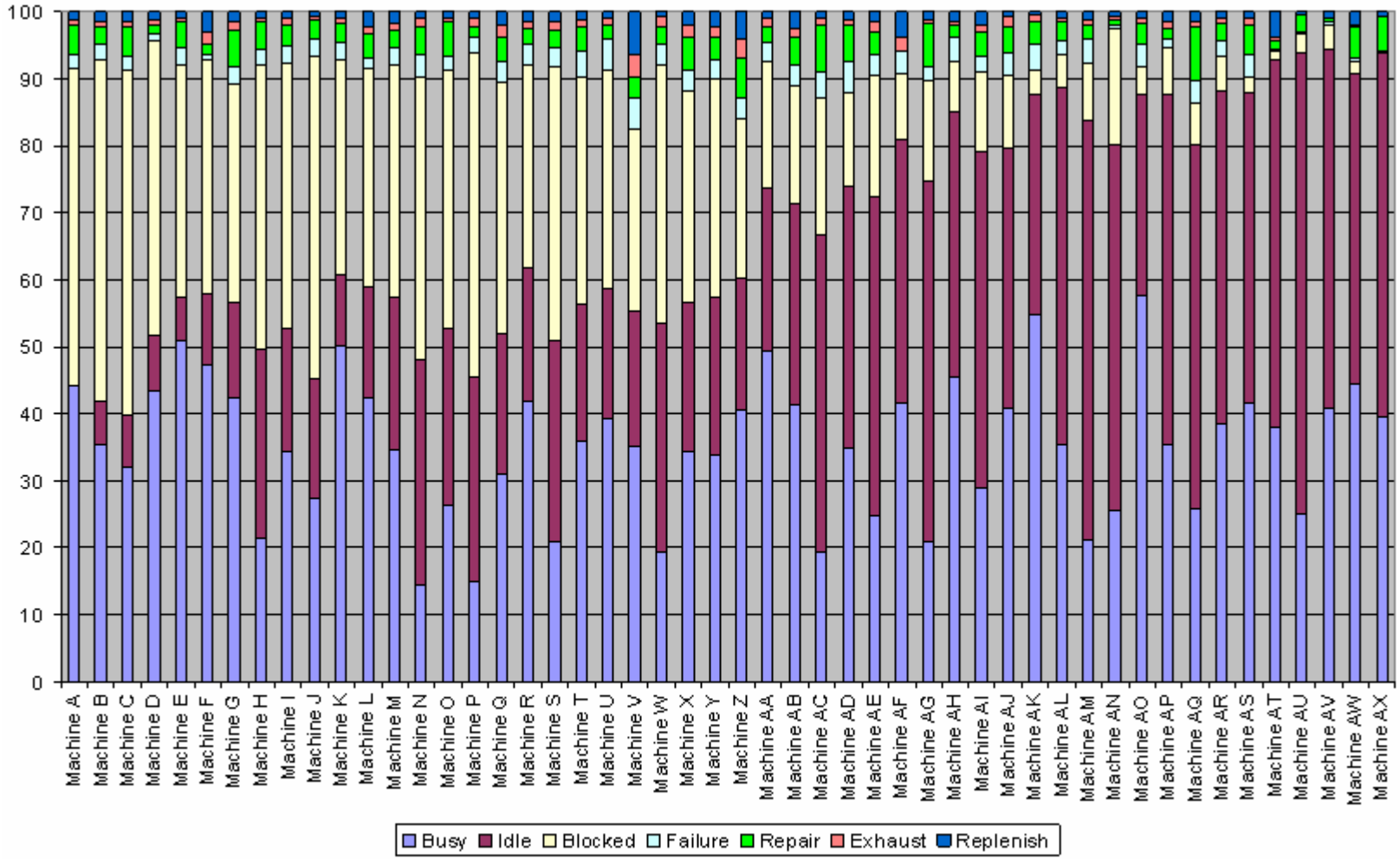

Figure II.10: LH-Configuration Case 10

APPENDIX III

RESOURCE STATE GRAPHS FOR SL-CONFIGURATION

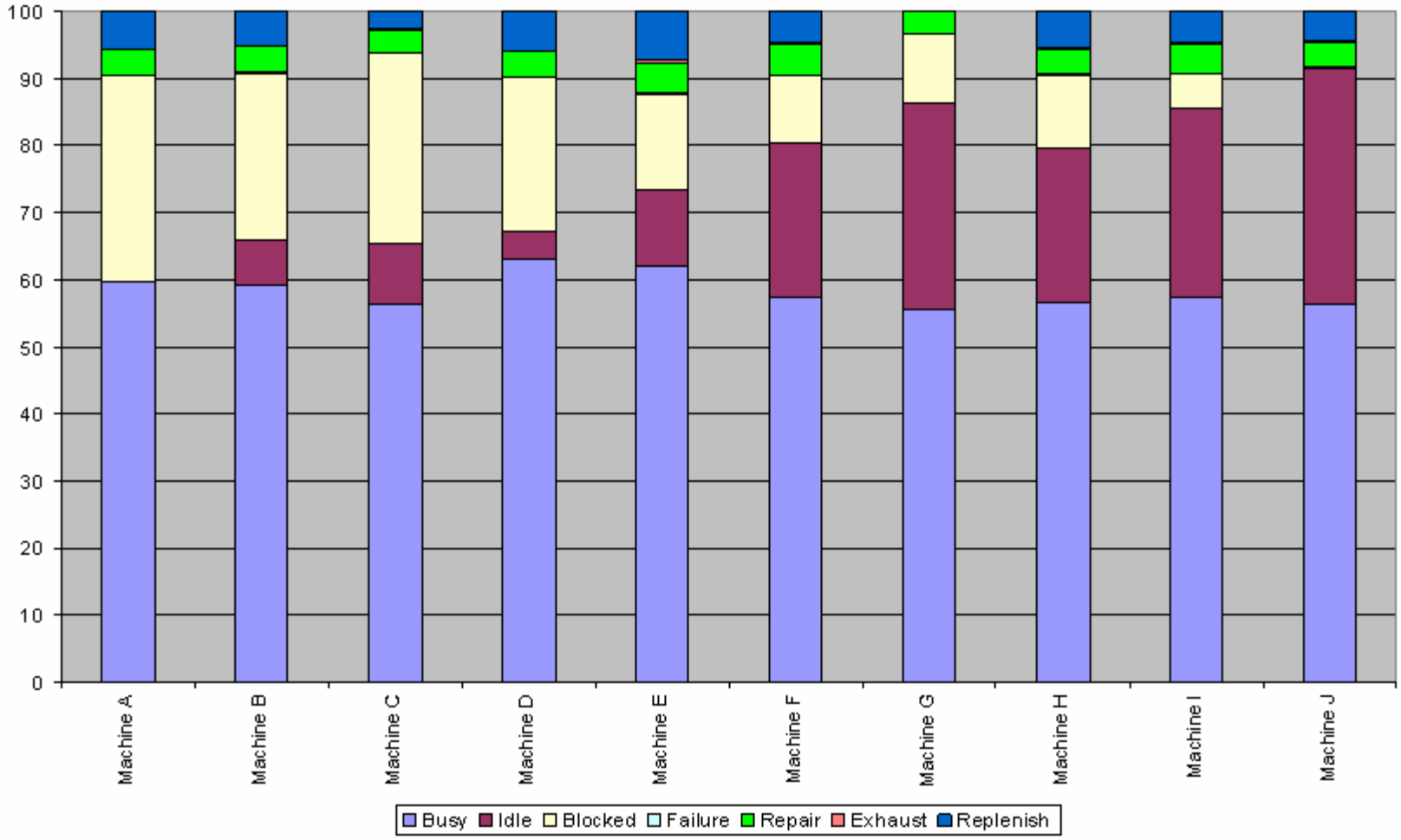

Figure III.1: SL-Configuration Case 1

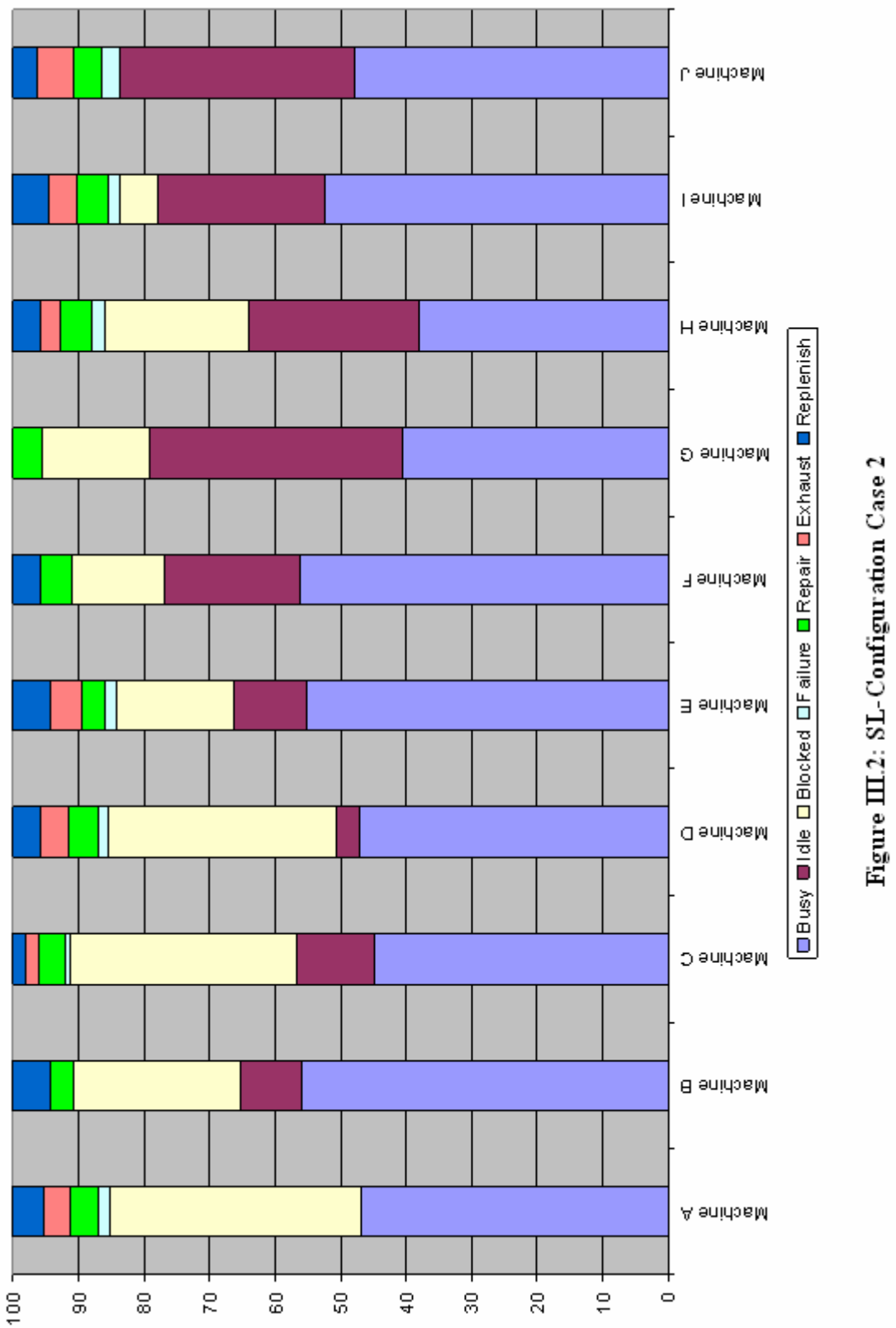

Figure III.2: SL-Configuration Case 2

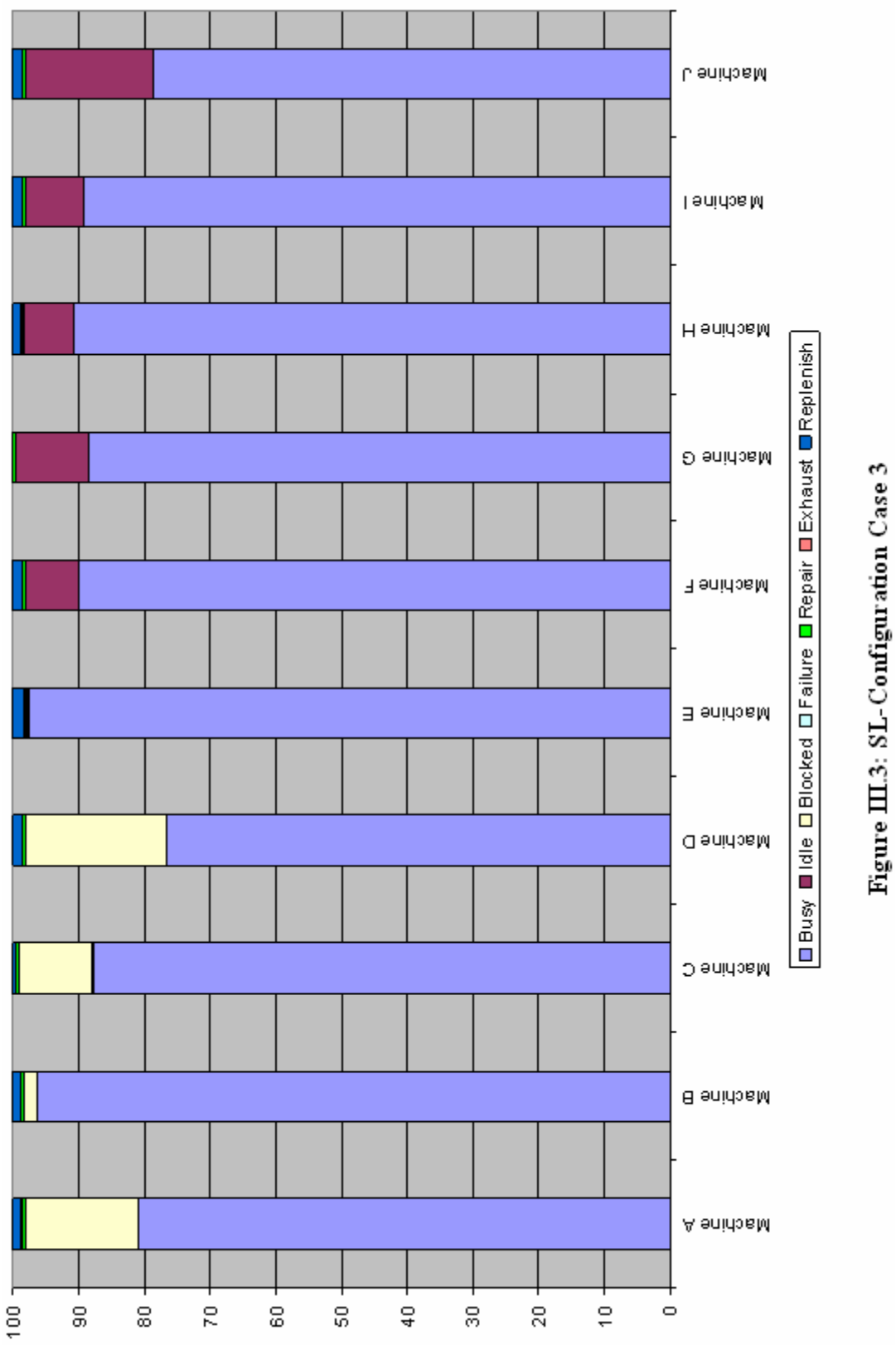

Figure III.3: SL-Configuration Case 3

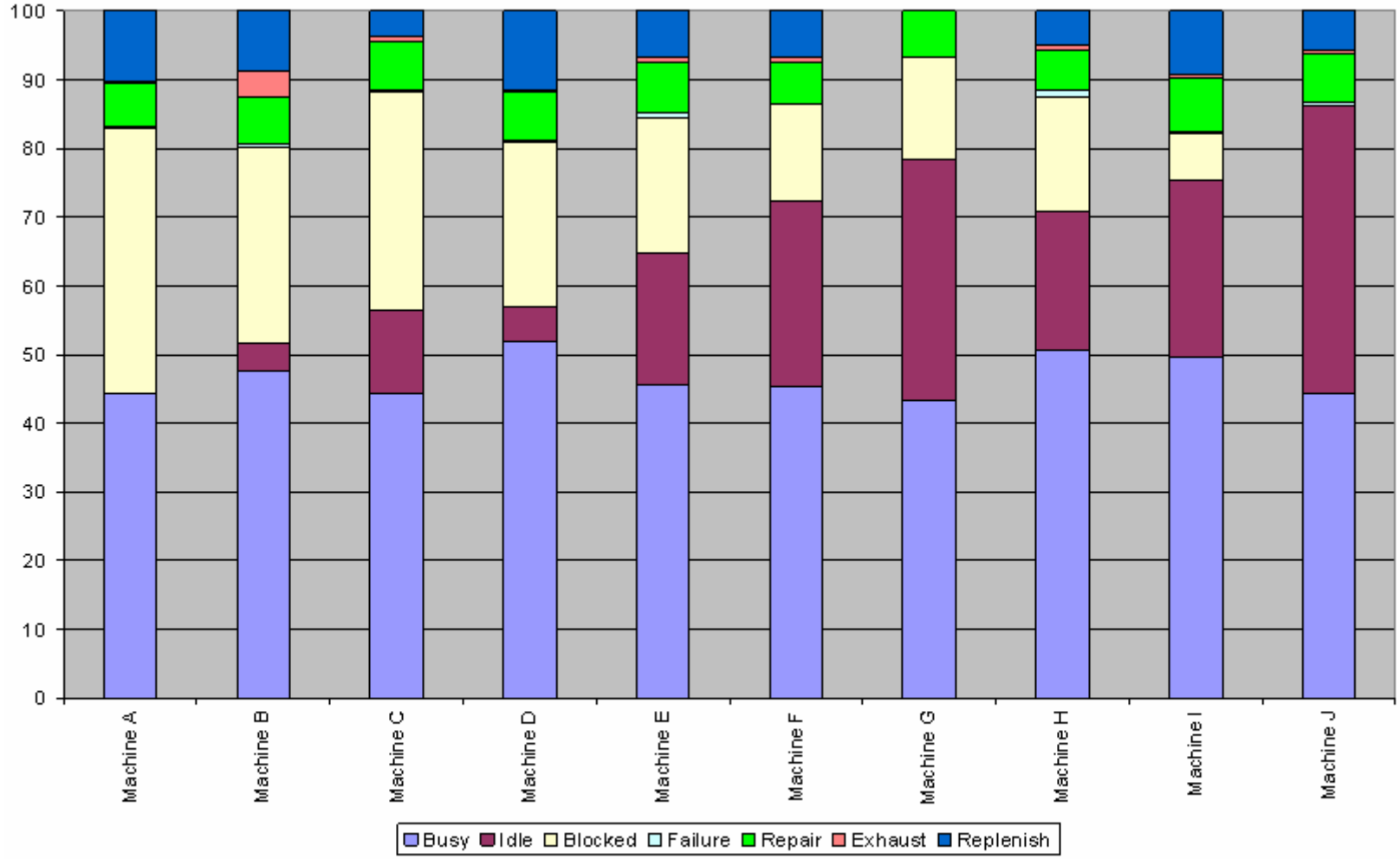

Figure III.4: SL-Configuration Case 4

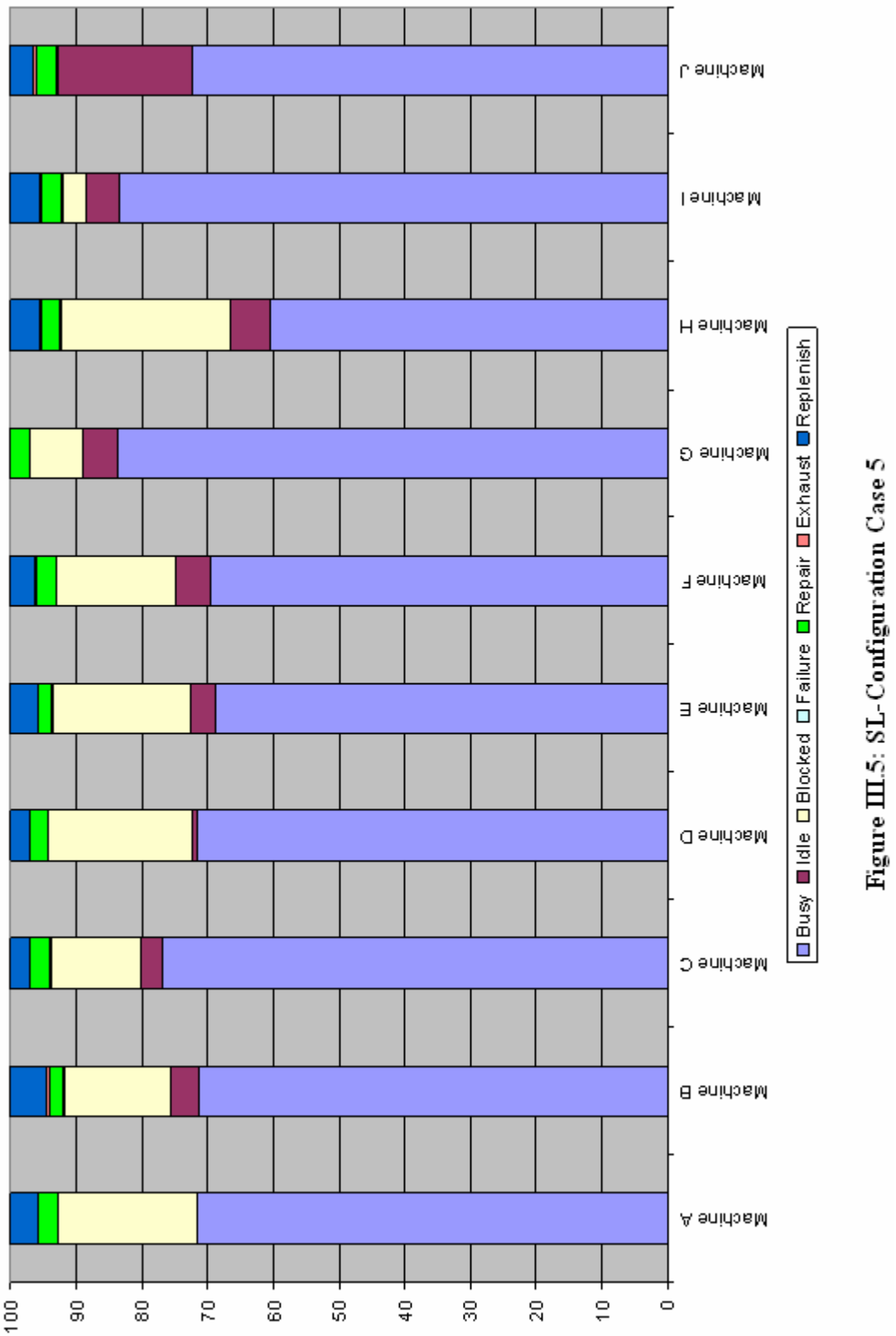

Figure III.5: SL-Configuration Case 5

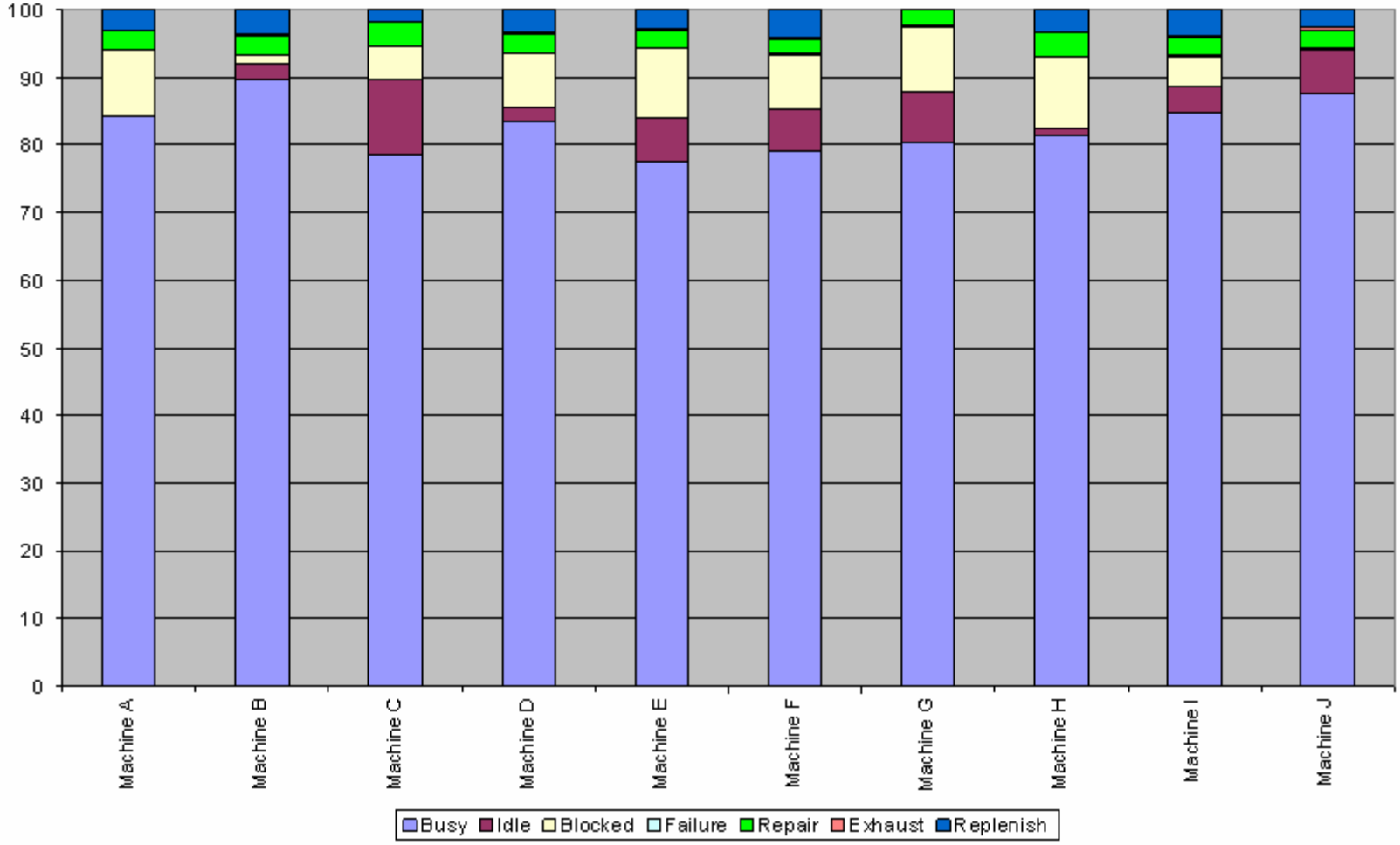

Figure III.6: SL-Configuration Case 6

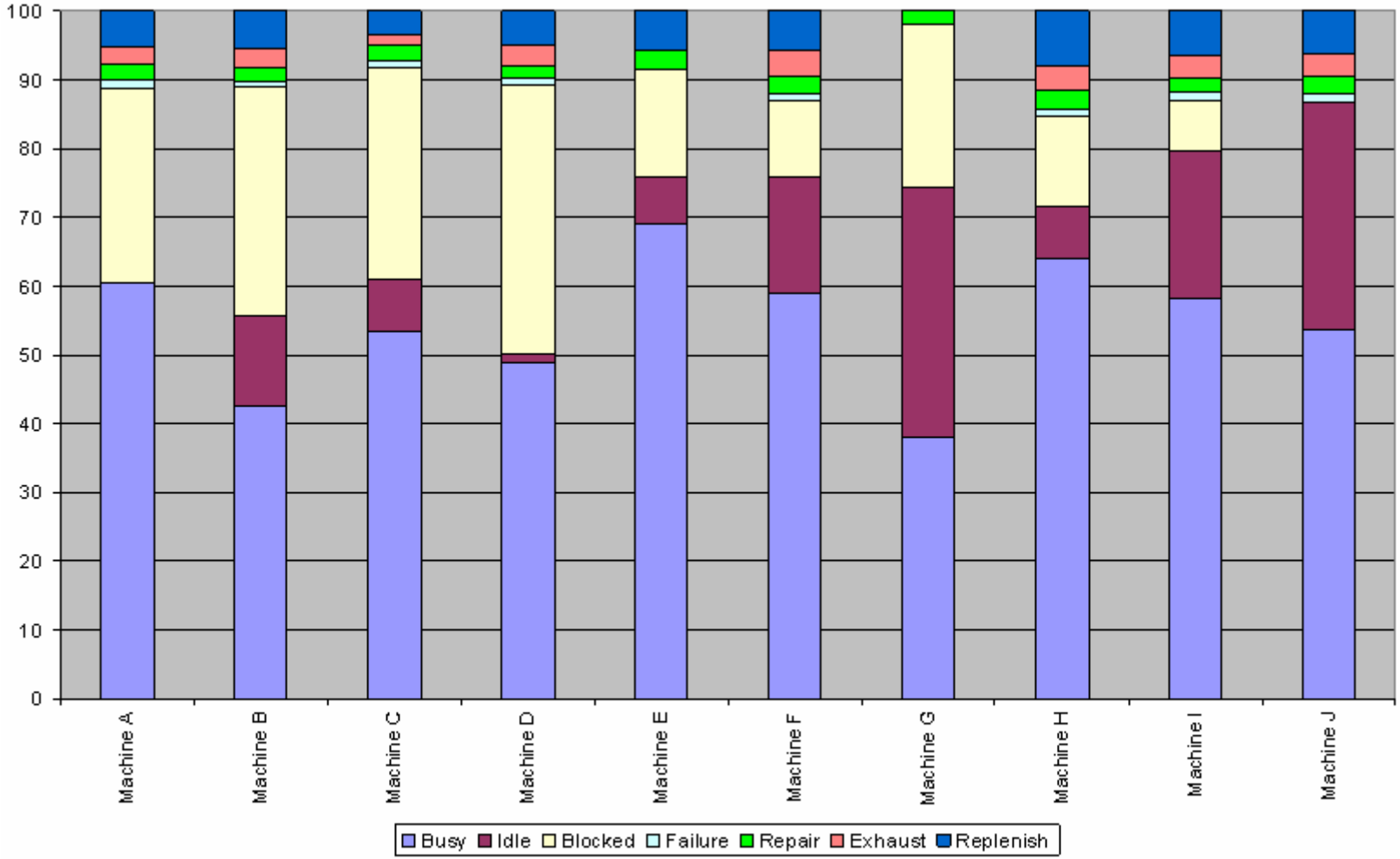

Figure III.7: SL-Configuration Case 7

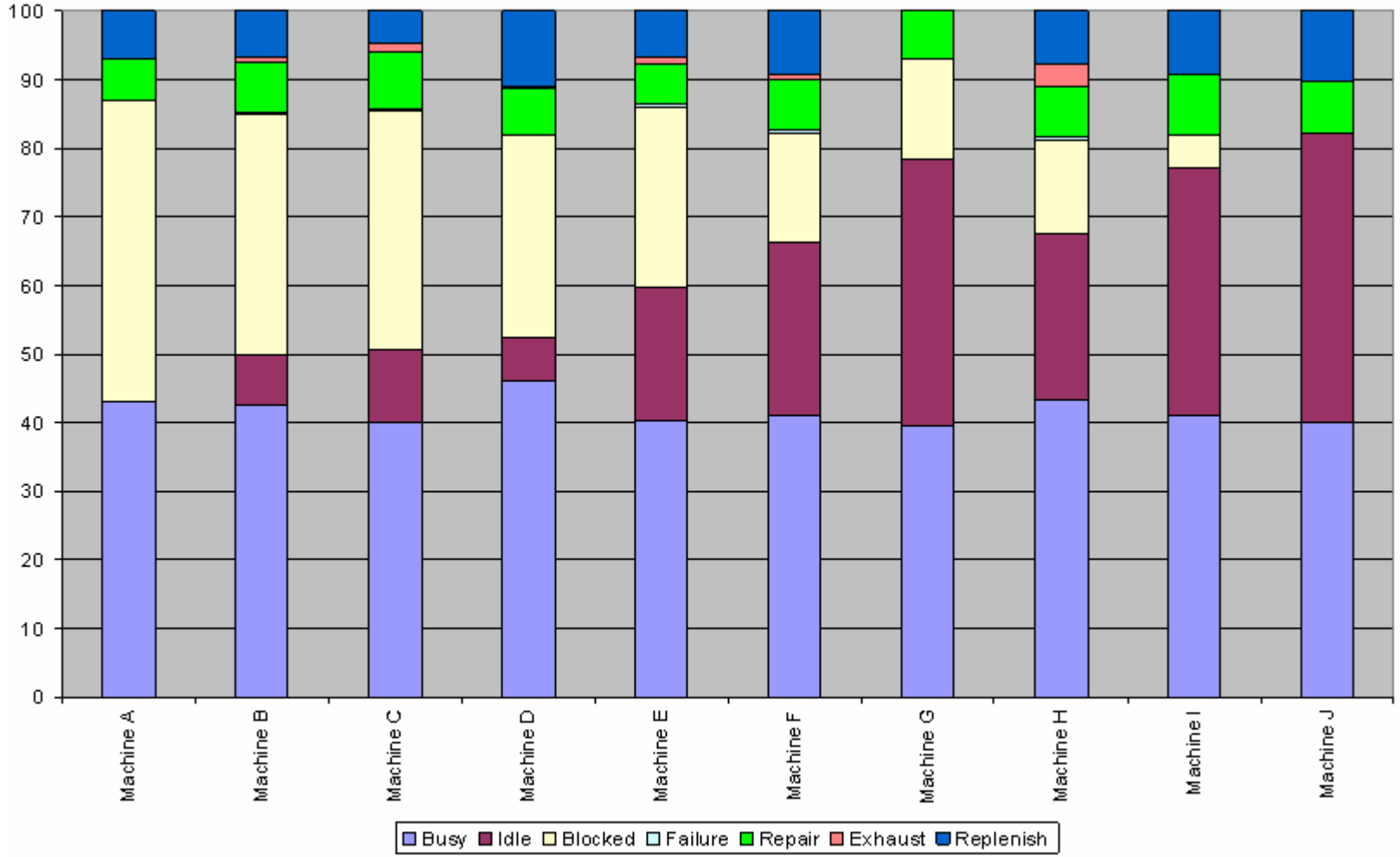

Figure III.8: SL-Configuration Case 8

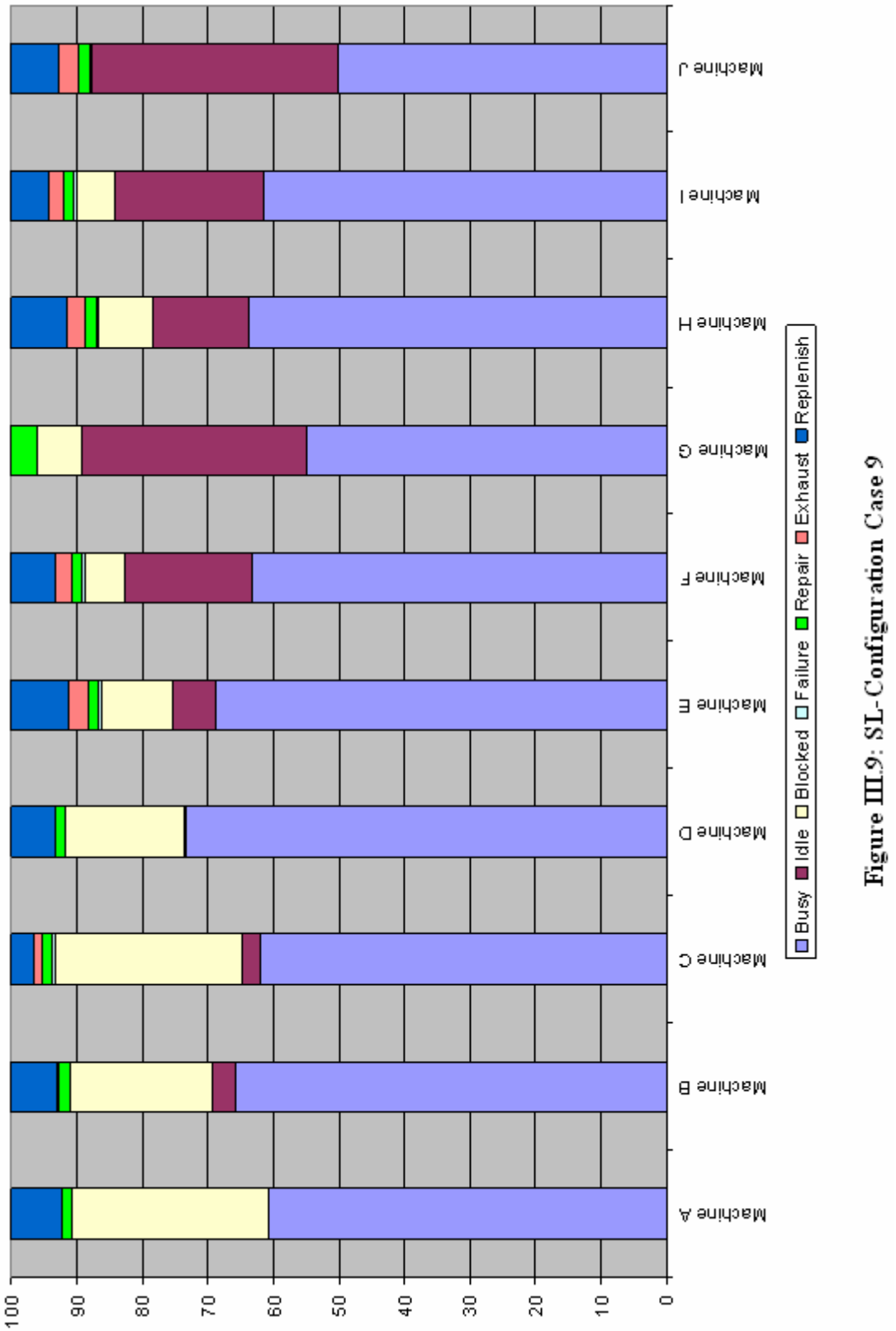

Figure III.9: SL-Configuration Case 9

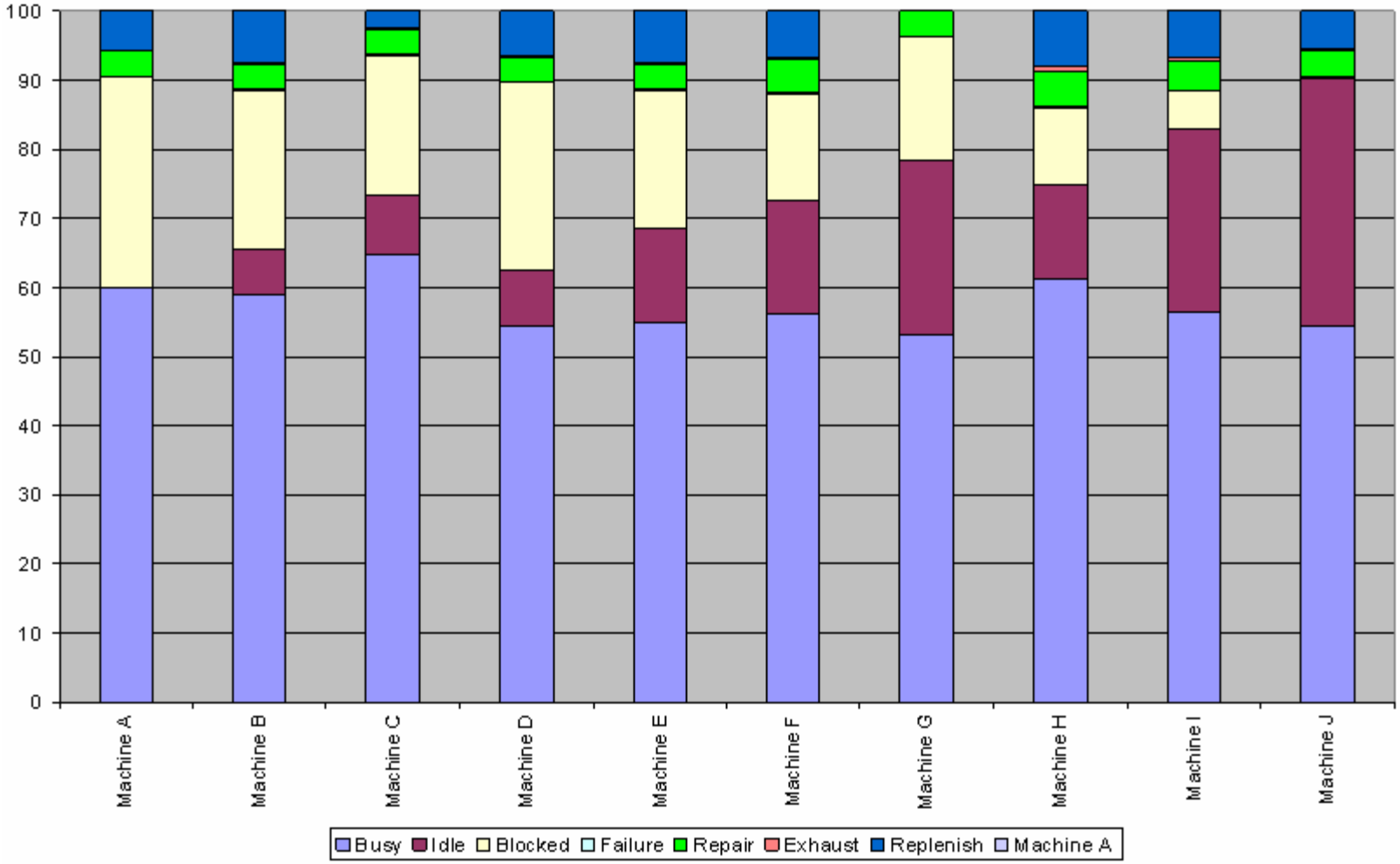

Figure III.10: SL-Configuration Case 10

APPENDIX IV

RESOURCE STATE GRAPHS FOR SH-CONFIGURATION

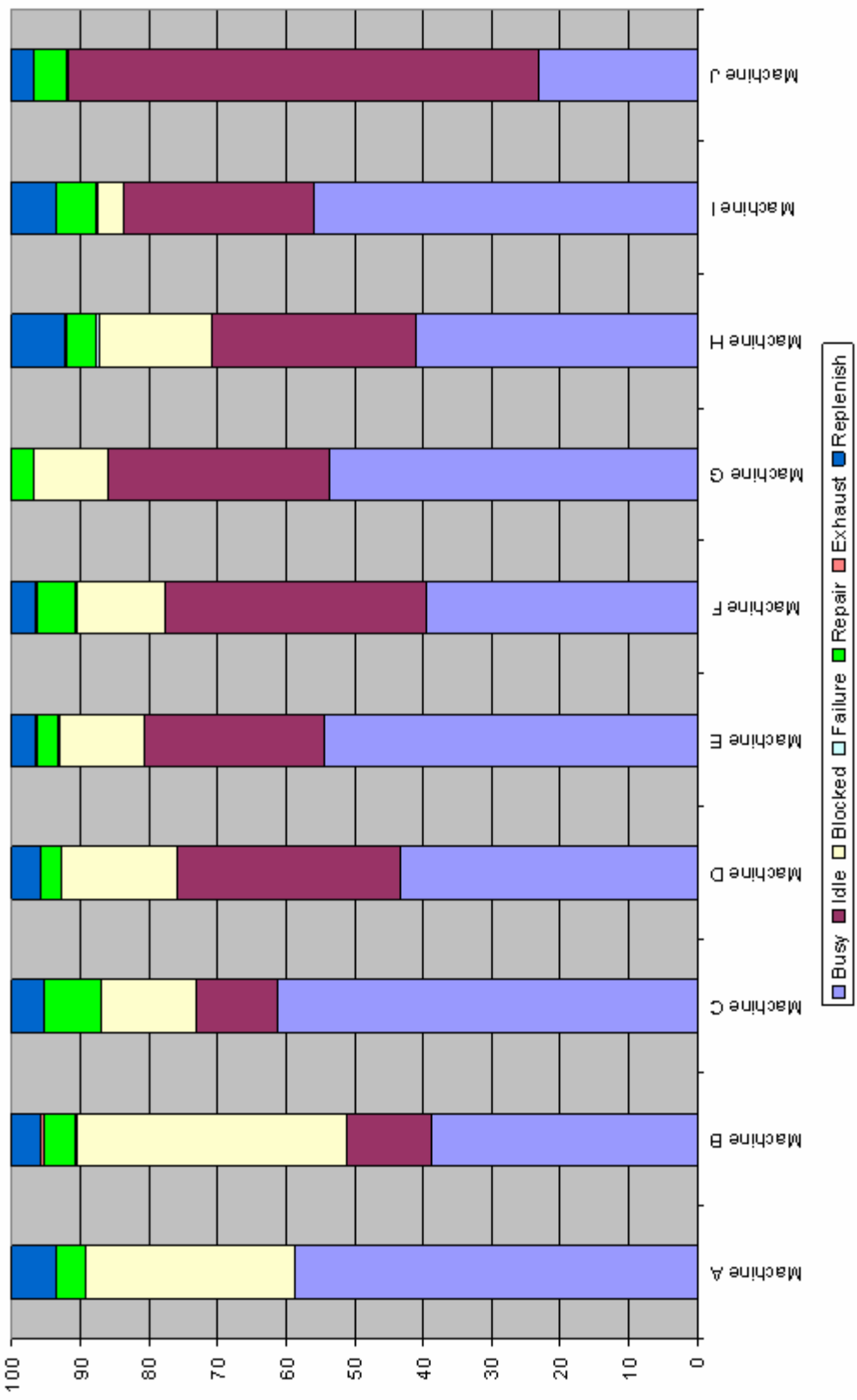

Figure IV.1: SH-Configuration Case 1

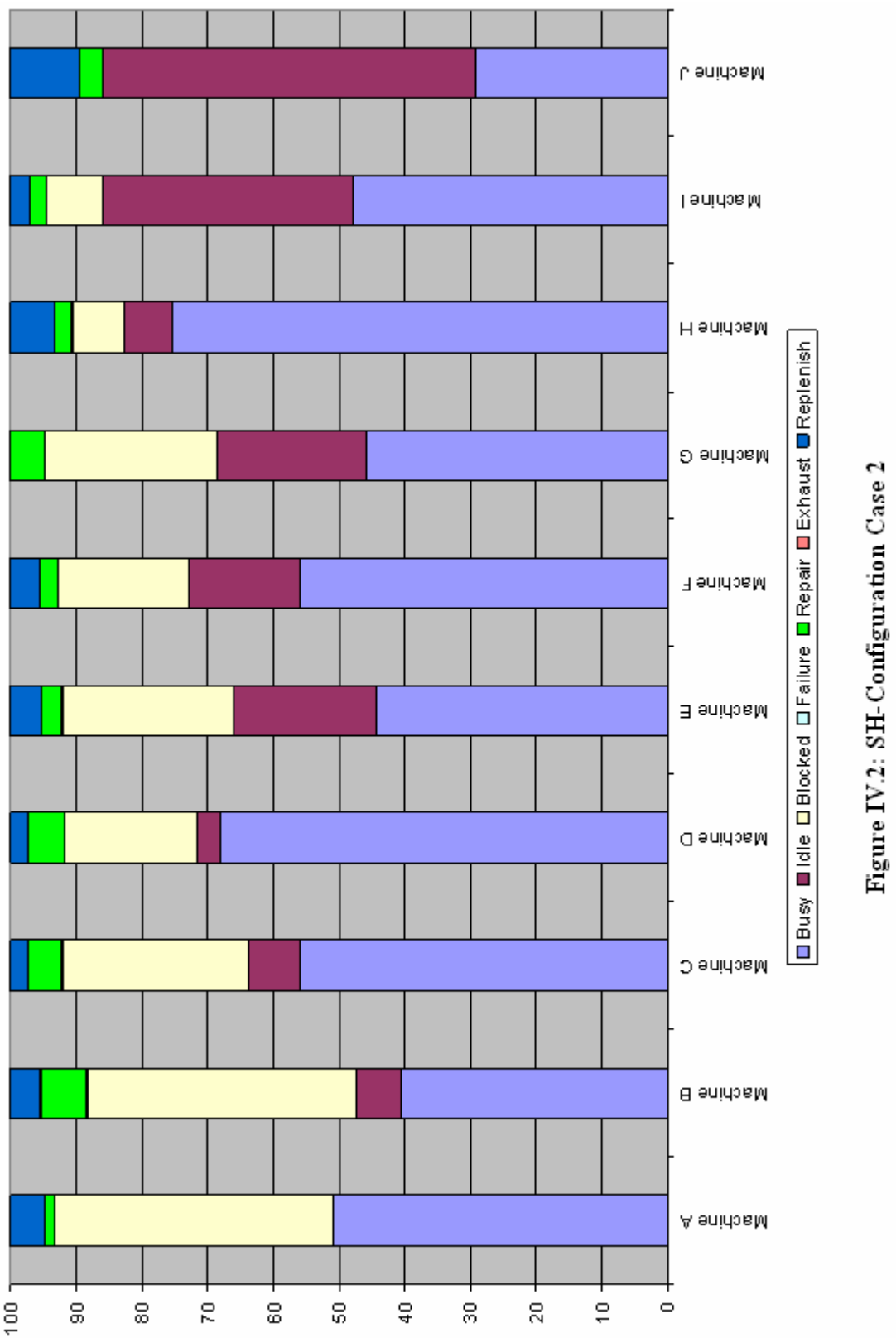

Figure IV.2: SH-Configuration Case 2

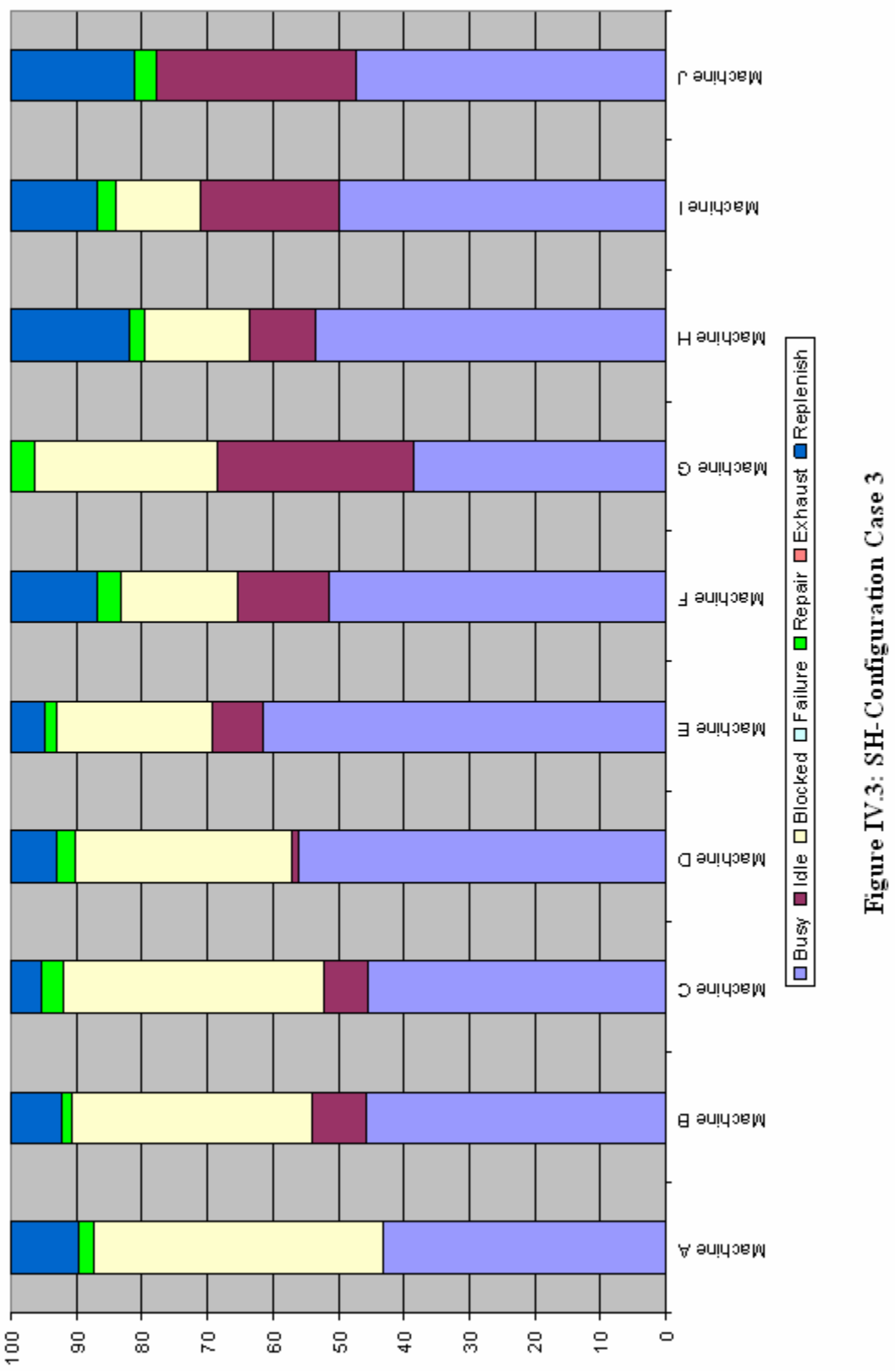

Figure IV.3: SH-Configuration Case 3

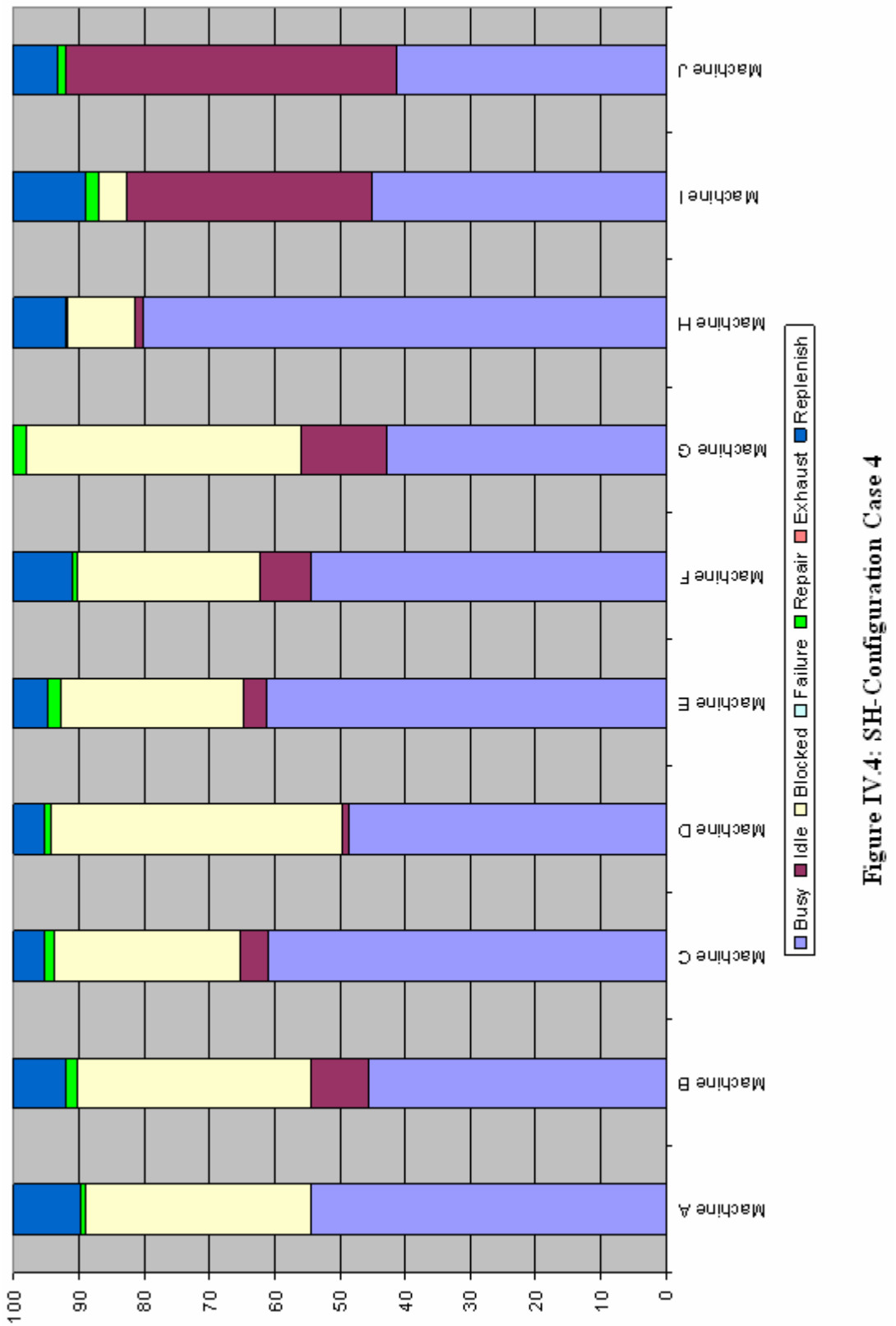

Figure IV.4: SH-Configuration Case 4

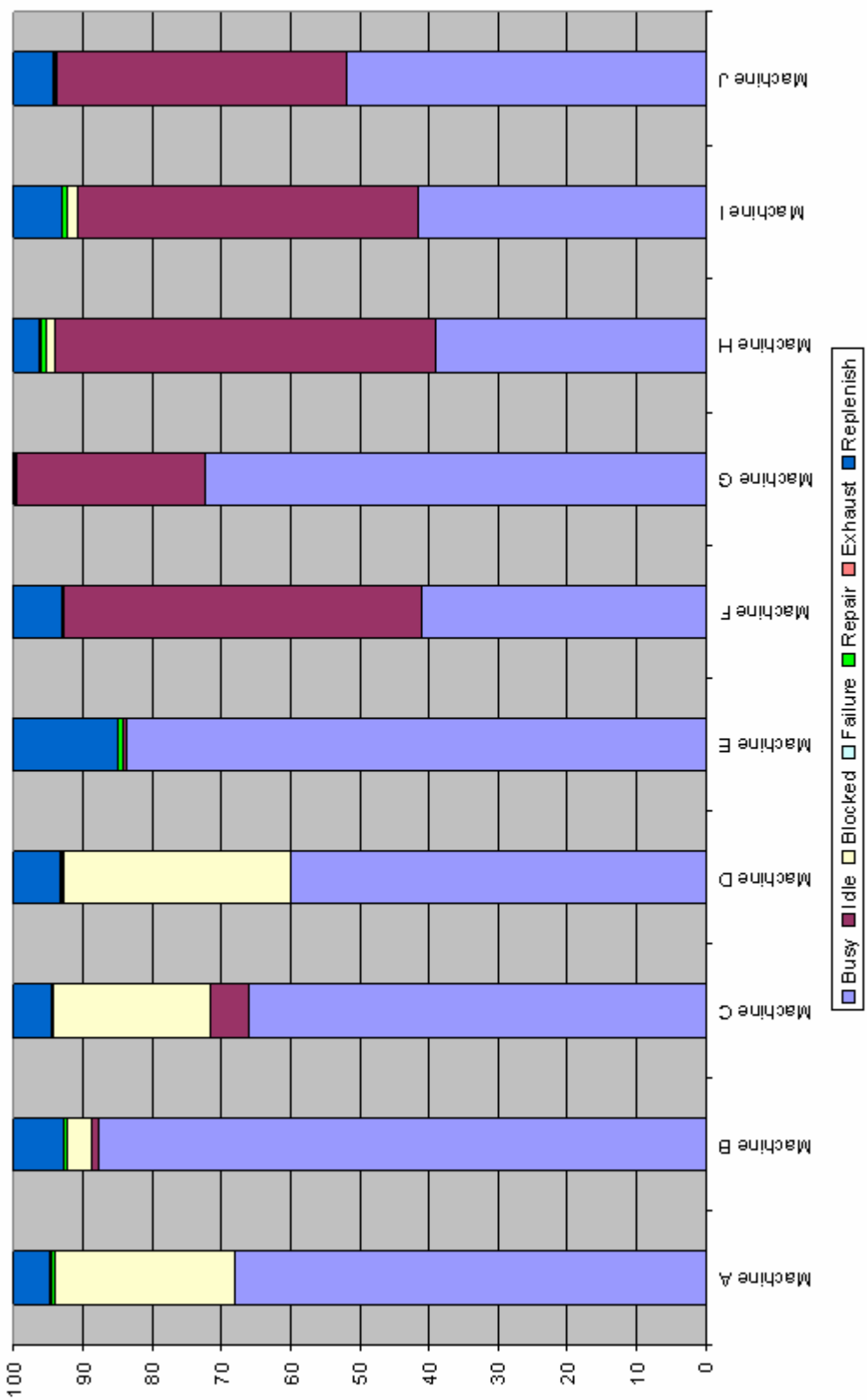

Figure IV.5: SH-Configuration Case 5

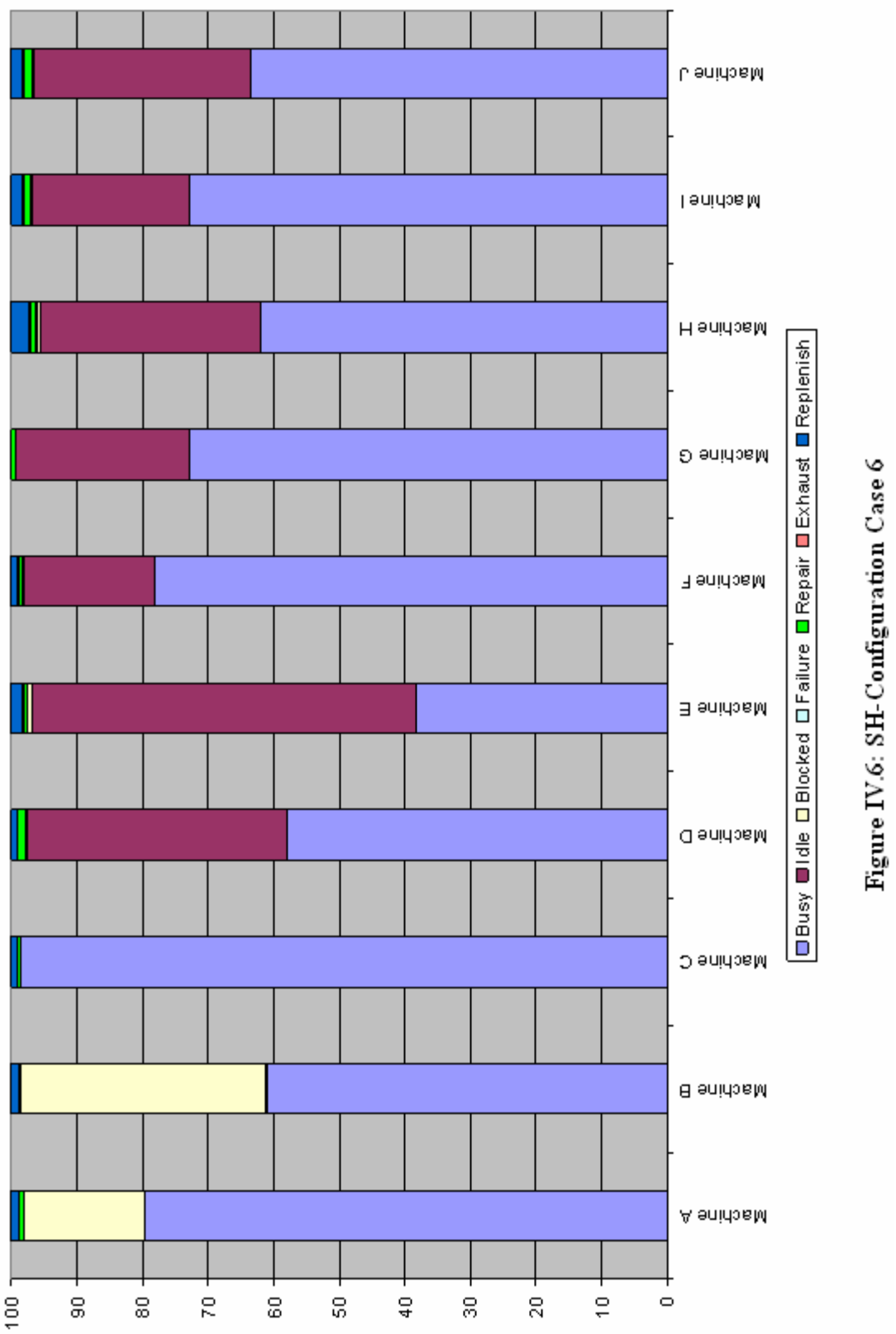

Figure IV.6: SH-Configuration Case 6

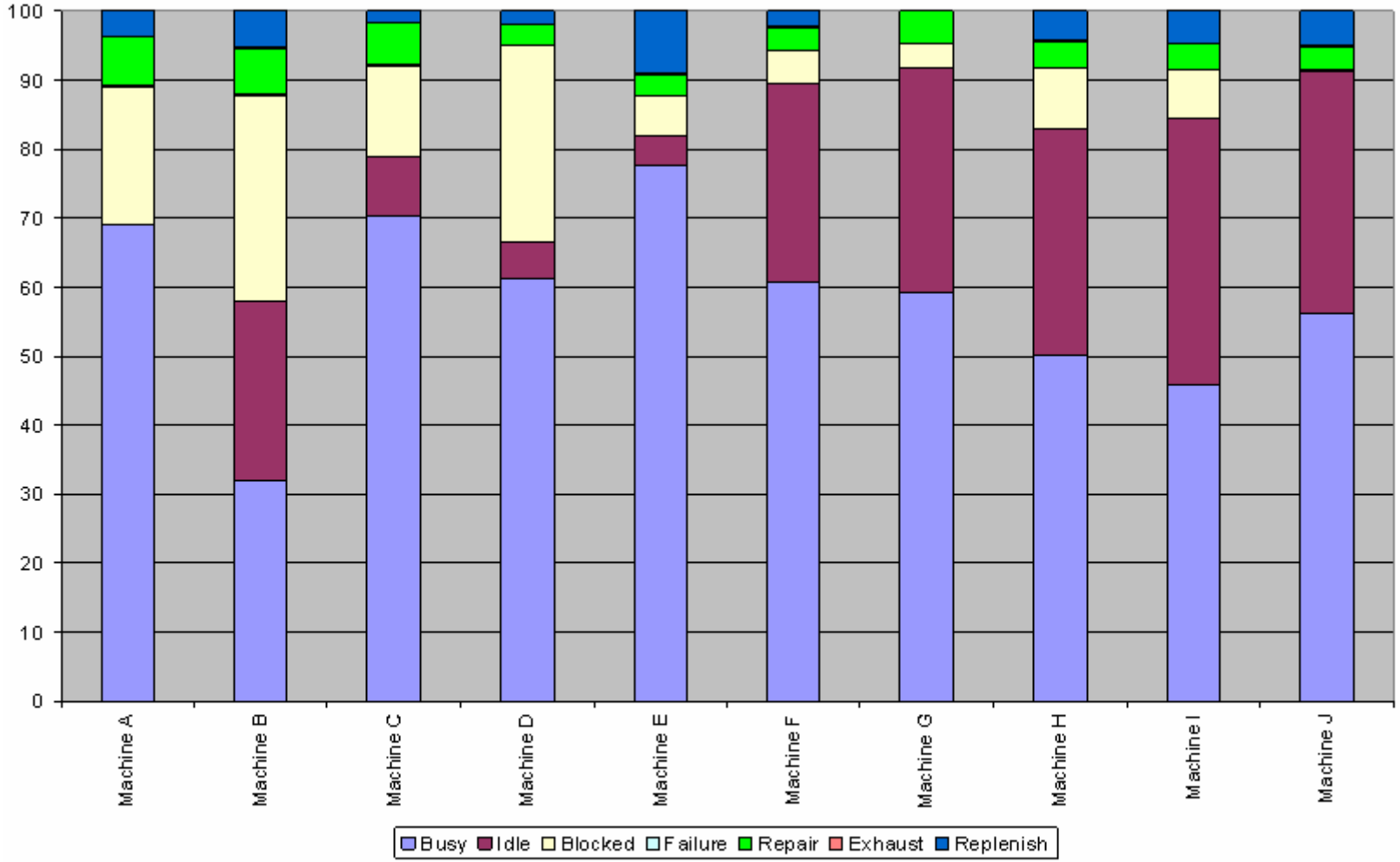

Figure IV.7: SH-Configuration Case 7

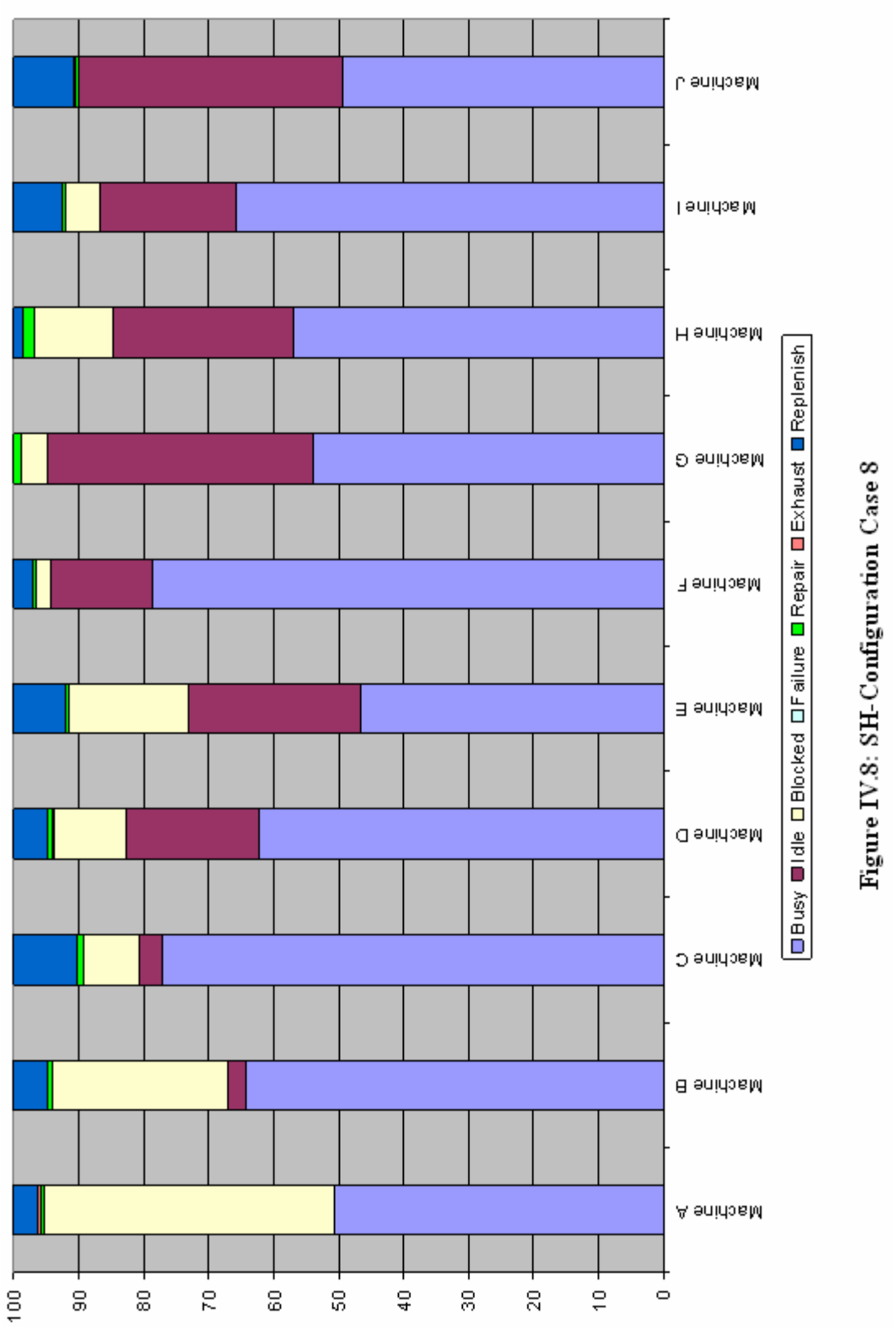

Figure IV.8: SH-Configuration Case 8

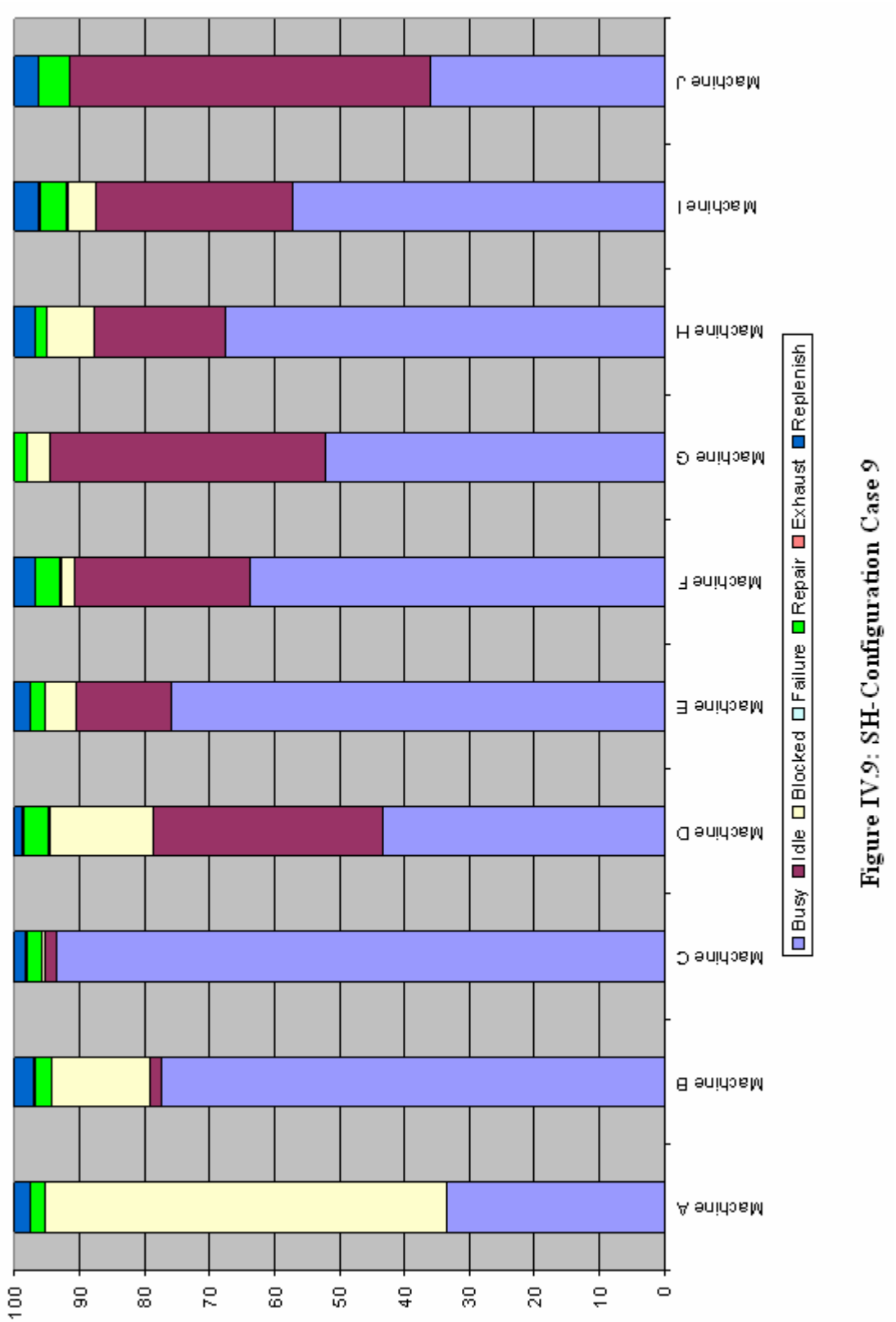

Figure IV.9: SH-Configuration Case 9

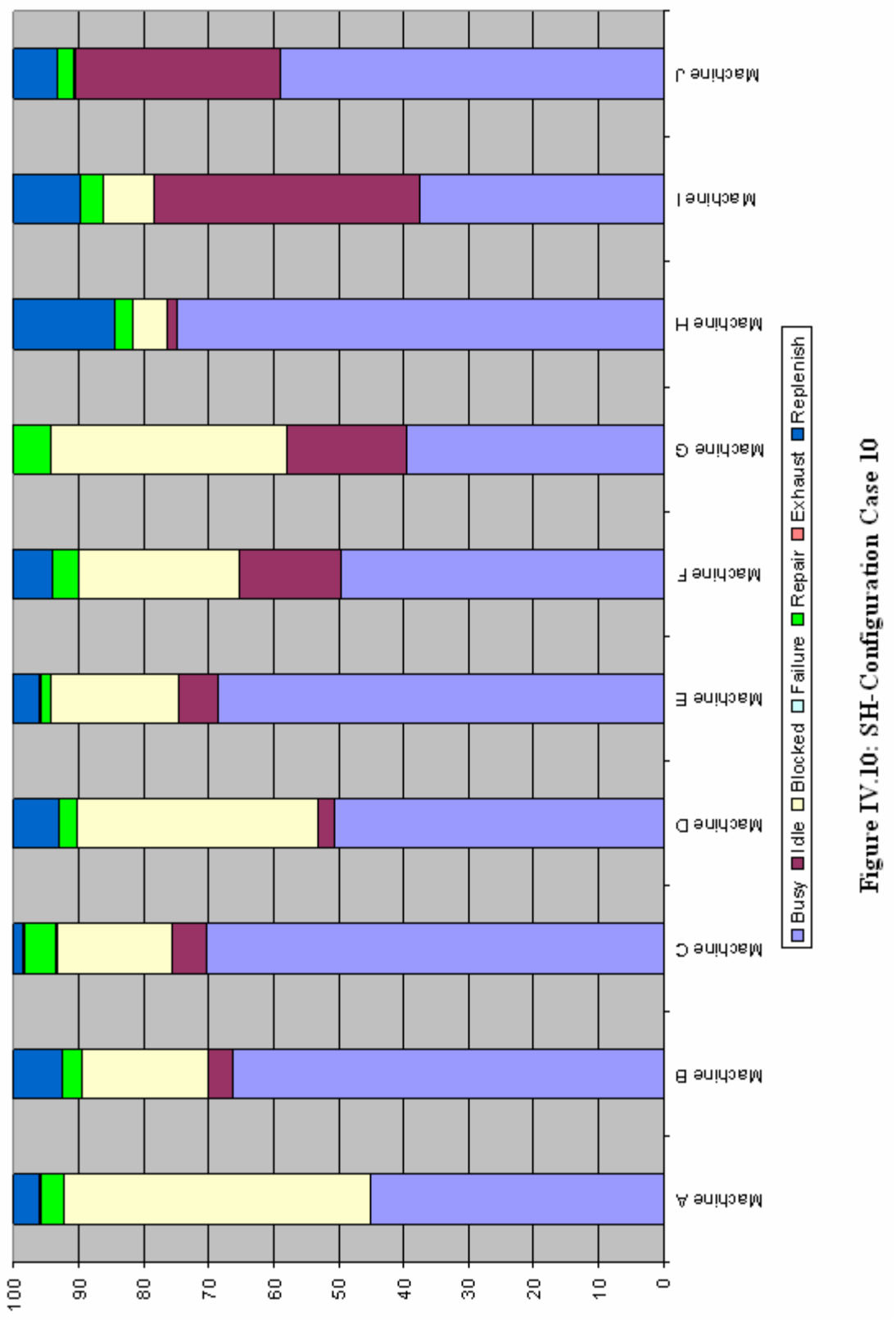

Figure IV.10: SH-Configuration Case 10