5-2010

MAXIMIZING THROUGHPUT USING DYNAMIC RESOURCE ALLOCATION AND DISCRETE EVENT SIMULATION

Gregory Gunn
Clemson University, ggunn@clemson.edu

MAXIMIZING THROUGHPUT USING DYNAMIC RESOURCE ALLOCATION AND DISCRETE EVENT SIMULATION

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Engineering

by
Greg Gunn
May 2010

Accepted by:
Maria Mayorga, Committee Chair
Kevin Taaffe
Georges Fadel

ABSTRACT

This research studies a serial two stage production system with two flexible servers which can be dynamically assigned to either station. This is modeled using discrete event simulation and more specifically the Arena software package by Rockwell. The goal is to determine dynamic allocation policies based upon the inventory level at each station to maximize the throughput of finished goods out of the system. This model adds to previous work by including actual switching time. The effect of the pre-emptive resume assumption is gauged, and the effectiveness of the OptQuest optimization package is also tested. Studies are conducted to determine the throughput of the system using easily implementable heuristics including when workers are together and separate. Additionally, the effect of buffer allocation and buffer sizing are studied, and it is shown that buffer allocation is not sensitive to changes in buffer ratio as long as there is buffer space available at each station while adding buffer space has a diminishing rate of return.

DEDICATION

I would like to dedicate this thesis to my parents who have stood behind me and supported me every step of the way. I truly could not have done this without them.

ACKNOWLEDGMENTS

This paper would not have been possible without the diligent guidance of my advisor Dr. Maria Mayorga. Her mentorship and patience through the years have made research more meaningful and enjoyable.

TABLE OF CONTENTS

Table of Contents (Continued)

LIST OF TABLES

LIST OF FIGURES

List of Figures (Continued)

CHAPTER ONE

SECTION 1.1: INTRODUCTION AND LITERATURE REVIEW

The popularity of lean manufacturing and Just-in-Time production practices in today's manufacturing and service environment has led operations managers to increase flexibility and reduce inventories. With these goals in mind many different industries have instituted the practice of cross training workers such that they are capable of completing many different tasks within the company. These cross trained workers provide operations managers with the unique challenge of allocating these flexible resources in a way that provides the most value to their company.

This research deals directly with a system containing two stations operating in a serial fashion and two flexible servers which can be allocated to either station. The servers are able to be dynamically allocated to either station based upon the inventory level at the stations in the system. The servers are able to work in collaboration with each other on a single piece of WIP and the service time is considered to be additive. The goal is to create an implementable dynamic allocation policy that will maximize the throughput of the system.

The issue of dynamic resource allocation is not unique to production processes. A significant amount of research has been done on the topic of dynamic resource allocation in reference to computing and parallel servers. Often times it is important to allocate the available resources in order to get the maximum computing power available with the resources at hand. One example of this is in Andradottir et al. (2003) who study the capacity of systems where the processing progress is indicated by the associated class of that job; they formulate a linear program to determine the upper bound of the capacity of a specific system. Slightly more recently Lee and Lee (2004) use evolutionary algorithms to maximize the throughput of a system with a predetermined amount of resources. However, in their formulation they constrain the problem such that all resources must be allocated to the tasks, and they include no switching penalty. This does not allow for any idle time of a resource. Ahn et al. (2004) show that the optimal policies for two parallel servers capable of serving two separate job classes and arrivals is characterized by one of three possible structures depending on two inequality conditions. They also completely characterized the optimal policy in the case with no arrivals into the system, also known as a queue clearing problem. Focusing on a slightly different problem Palmer (2005) uses dynamic programming to solve the case of several job classes arriving in the system and joining different queues with several servers all capable of doing all job types. The holding cost of jobs being in the system is minimized and an optimal policy is obtained. Simple heuristics are simulated and compared with the optimal solution. In Batta et al. (2007) the minimum staffing cost for a service center handling different types of services is found using integer programming. They include switching costs, penalties for unanswered calls as well as suggesting relevant heuristics.

Research has been conducted concerning tandem systems most of which is focused on minimizing the holding cost of jobs in the system. Rosberg (1982) developed an optimal policy as a function of the current state of the system, with the goal of minimizing costs while considering holding cost of a job and cost of switching a server from one station to another. In this case, as in other papers the concept of cost is used as a measure of throughput. A class of resource problems exist called queue clearing problems. Queue clearing problems focus on allocating resources in such a way so as to minimize the cost to get all the parts already in an already full system out when there are no new arrivals. One example of this type of problem is found in Farrar (1996) which explores the case of a tandem system with no new arrivals, two static servers, and an additional flexible server that may be used at either station or turned off. Further studies were performed in Ahn et al. (1999); in this paper they model a queue clearing system and classify when it proves optimal to put both servers at one station or the other. More recently Scheifermayr and Weichbold (2005) provide a complete solution structure for a two station tandem queue clearing system when the objective is to minimize the holding costs.

There exists some research which uses the objective of maximizing throughput of a tandem system. Van Oyen et al. (2001) show that what they coin an *expedite policy*, of having all workers follow a single part through the system, minimizes the cycle time for all parts and therefore maximizes the throughput. They also assume that work effort can be continuously monitored and changed. Andradottir et al. (2001) model a tandem system with the objective of maximizing the throughput of a system with two flexible servers using Markov Decision Processes. They show that if the service time depends either only on the server or only on the station that any non-idling policy is near optimal. Additionally they show that if the service time depends on both the server and the station then the optimal policy is to assign each server to the station which they are more proficient at and switch only to avoid blocking or starving. Andradottir et al. (2004) later extend their work to provide an optimal structure for tandem systems with more than two servers. They also give several heuristics which approach optimal long term average throughput. In a similar but alternate situation Yi and Argon (2008) extend the scope of the system to determine an optimal resource allocation policy for assembly type systems with several feeder stations and flexible servers. They must distribute resources between the assembly station as well as the feeder stations that supply the assembly station. They give several heuristics and demonstrate that they achieve near optimal throughput. Arumugam et al. (2009) use dynamic programming to provide an optimal solution structure to a tandem work system with the objective of maximizing throughput. Without taking into account the cost of switching a server from one station to the other it is shown that when workers collaborate it is always optimal for them to be together. Under the assumption of non-collaborative workers they show that it is sometimes optimal for the workers to be separated and present the structure for the optimal solution. Mayorga et al. (2009) continue their work in tandem systems with flexible servers by adding switching costs in a later work. In addition they also take into account holding cost. Optimal solutions are found using dynamic programming and optimal solution structures presented. Additionally they present heuristics and show that they may be used to achieve near optimal results.

One of the motivators for this research was to relax many of the assumptions made by previous research. Previous research has focused on maximizing profit when switching was viewed as a cost to the system. This research includes the actual time that it takes to get from one station to the other. In Arumugam et al. (2009) the servers operate under the assumption of preemptive resume. Under this assumption a worker is permitted to, at any point in time, stop working on the part they are currently working on and move to another station. They are then allowed resume work on that part at a later time from exactly where they left off. This paper removes this assumption and forces the worker to work on a part until completion at a certain station. That is a worker is only allowed to move once they have finished their operations at either Station 1 or at Station 2. With this in mind discrete event simulation was selected as the tool of choice. Using simulation, the throughput of a system can be measured while including the time it takes for a server to move from one station to another as opposed to the switching cost which was used in other research. Another benefit of simulation is that it allows some assumptions to be relaxed. One of the main assumptions that can be relaxed is the assumption of exponentially distributed arrival and service times. The exponential distribution is used because of its memory-less property which makes it possible to model using Markov decision processes. Relaxing this assumption may help to make the model of the actual process more realistic in certain situations.

Simulation has been shown to be a useful in several different areas of resource control with regard to manufacturing. Erickson et al. (1987) discuss the use of simulation to test the efficacy of PLC logic. PLC stands for programmable logic controller which control assembly lines and are ubiquitous in modern manufacturing plants. The authors conclude that simulation can be useful for testing after the system is set up and also before a system is made to test how it can work. They also describe the use of real time simulation to aid in resource scheduling and management. Drake and Smith (1996) provide a framework detailing how to incorporate simulation into real-time decision making. They describe the different components necessary to make the simulation output useable for multiple users and develop a framework for multi-user cooperation. Bischak (1996) uses simulation to show that a U-shaped production system with flexible workers can surpass the throughput of a system with static workers. This information is then extended to varying the throughput by varying the number of workers all in an effort to increase flexibility without the use of buffers and therefore reducing inventory. Chun and Mak (1996) use simulation to aid in solving the real-time problem of the number of checkout counter agents to make available. They use flight schedule data, airline data, and many other factors to simulate how many servers are needed to deliver the desired service level. This system was used with success in the Hong Kong Kai Tak airport.

In this paper we would like to use optimization practices in association with discrete event simulation to find a near optimal solution. One of the chief reasons to use simulation is that many times the situations to be simulated are not readily solvable with traditional methods. In this paper the optimization is done chiefly through the use of the commercially available software package OptQuest which pairs with Arena. Optimization through simulation is a newer field which is making strides with the recent advances of computing power and the growing popularity of discrete event simulation. Optimization with simulation presents many challenges which must be handled differently than other optimization problems. Glover et al. (1999) detail many of the challenges facing optimization when paired with simulation. They move on to several different optimality search methods and how they are incorporated into OptQuest, and then compare OptQuest to other methods of optimization. Kleijen and Wan (2006) use simulation to solve the (s,S) inventory problem. In order to obtain optimal results they test several different meta-heuristics including OptQuest and brute force grid computing as well as others. They conclude that OptQuest is just as effective as other methods though probably not as efficient. They also mention that OptQuest is much easier to set up than the other methods mentioned. Rogers (2008) discusses some of the uses of OptQuest. He applies it to several different scenarios and highlights some of the strong points and weaknesses of the program.

This research adds to previous research because it has the objective of maximizing throughput while taking into account the time that it takes for a server to switch from one station to the other. The complexity of the complete policy makes it a necessity to develop heuristics which are easily implemented. Several heuristics found in the literature are tested to see if they deliver good results in this model. Additionally several other heuristics are developed.

SECTION 1.2: MODEL DESCRIPTION

Many methods and metrics have been developed over the years to determine how well a production process is performing. One of the most important measures of how well a system is doing is the throughput of the system, or how many parts can be produced by the system in a given amount of time. The next natural need of any system is to keep costs down to a minimum. Effective manufacturing systems and policies are those which produce the most finished goods while using the least amount of workers and resources. An interesting question therefore is how to maximize the throughput of a system using a pre-determined staffing level.

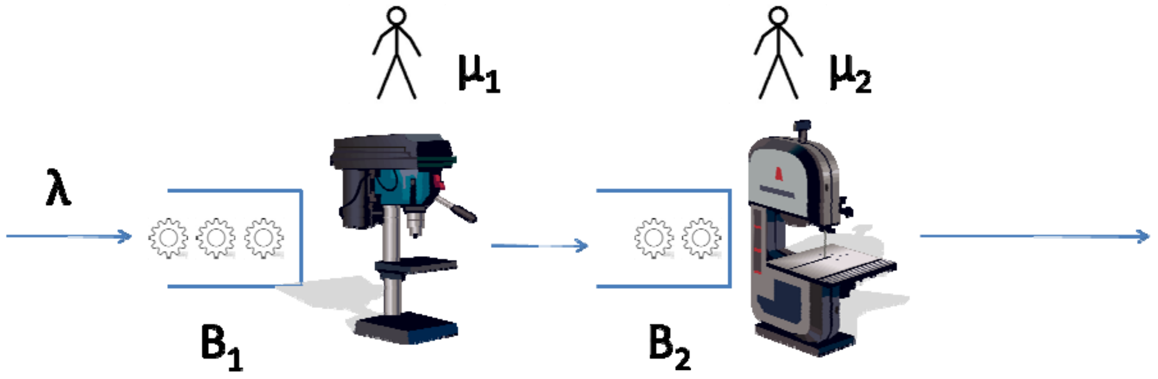


Figure Error! No text of specified style in document.**1.1: System Overview**

This research considers a serial system consisting of two stations with buffers in front of each station, and overview of which can be seen in Figure 1.1. Arrivals enter the system at a rate λ and proceed either to the first station or to the first buffer if there is work being done in the first station. The interarrival time is assumed to be exponential. The buffer in front of Station 1 is of a size B_1. A part balks from the system if the buffer in front of the first station contains B_1 parts. After service is complete at the first station the job proceeds to the second station. The buffer in front of the second station is of a size B_2. If the buffer in front of the second station contains B_2 parts the first station is said to be blocked and no more parts may be processed there until the current job is allowed to enter the second station. Upon completion at the second station the part is considered a finished good and exits the system.

There are two flexible servers, flexible here meaning that either server may work at either station, with associated exponentially distributed service times with rates μ_1 and μ_2, respectively. Workers are said to work *collaboratively* meaning that if both servers are at the same station their service rates are additive. For example if there were two servers at the same station with service rates of 3 parts per hour and 2 parts per hour, then the resulting rate while they were both working there would be 5 parts per hour.

Servers are moved from station to station as dictated by the allocation policy. The allocation policy can be represented by a matrix which indicates where servers should be as a function of the inventory in the system an example of which can be seen in Figure 1.2. There are four separate policies that can be used indicated by the integers 1, 2, 3, and 4. Policy 1 indicates that both servers are working at Station 1. Policy 2 indicates that the faster of the two servers is at Station 1 and the slower of the two servers is at Station 2. Policy 3 is the exact opposite of policy 2; it means that the faster of the two servers is located at Station 2. Policy 4 indicates that both servers are located in Station 2. To illustrate how such a policy should be interpreted the allocation policy represented in Figure 1.2 means that if there are 2 pieces of Work in Progress (WIP) located at Station 1 and 4 pieces of WIP located at Station 2 then the policy would be to have the slower server at Station 2 and the faster of the two servers at Station 1. It should also be noted that if the buffer is of size 6 there is still room for one part to be processed in addition to the WIP in the buffer. Therefore the example below is illustrative for the case when $B_1 = B_2=6$.

		Station 2							
		0	1	2	3	4	5	6	7
	0	1	4	4	4	4	4	4	4
	1	1	1	1	4	4	4	4	4
Station 1	2	1	1	1	1	2	3	4	4
	3	1	1	1	1	1	2	3	4
	4	1	1	1	1	1	2	3	4
	5	1	1	1	1	1	2	3	4
	6	1	1	1	1	1	2	1	4
	7	1	1	1	1	1	1	1	4

Figure 1.2: Example Allocation policy

This model takes actual switching time into account, thus, when a server switches from one station to another, there is a delay between the time that he leaves one station and is able to begin work at the other station, denoted as switching time (ST). This delay could be due to travel and/or required set-up time. We assume the switching time is deterministic. Most queuing models of production systems use a preemptive resume assumption (work on a part can be preempted and resumed at a later time exactly where it was left off). This assumption may not be realistic in practice. There are a few rules however that are used to help make the model more realistic to the manufacturing environment. They are listed below:

1. *Arriving workers do not preempt work in process*. An illustrative example: At time t=10 the workers are at separate stations, and the allocation policy dictates that the worker at Station 2 should join the worker at Station 1. If the switching time is 3 seconds and the time for the worker at Station 1 to finish a part is 5 seconds then at time t=10 the worker at Station 1 will start work on a part alone. At time t=13 the worker from Station 2 will arrive at Station 1. Even though they are said to work collaboratively the second worker may not join the first worker until the current part is finished. Therefore at time t=15 the worker originally at Station 1 will finish work on the current part. At this point the collaborative effort may begin at a rate equal to the sum of each worker's rate.
2. *A worker may not be redirected during transit*. That is, if the allocation policy changes while a worker is in transit, he will continue on his path and not be aware of such changes until he arrives at his destination. Therefore it is possible for a worker to leave a station and upon arrival at the new station be directed to immediately return to the previous station.
3. *Pre-emption is not allowed*. Workers must finish their task before switching, even if the allocation policy changes while they are processing a part.

This research helps to answer the question of how to utilize the two workers in this system in such a way as to maximize the throughput. The dynamic assignment of resources to work stations is an effective and intuitive tool for managers to use in order to maximize the throughput of their system.

Discrete Event Simulation (DES)was used to model the system described above. More specifically Rockwell Software's Arena Simulation Package was used in conjunction with OptQuest for Arena as a potential method of optimization. Additionally it should be noted that all the computations done in this research were performed on a 3.00 GHz Pentium 4 Processor with 512MB of RAM at 2.99GHz.

DES was chosen because the benefits of using this method outweighed the negative aspects in relation to our research. One nice feature of using Discrete Event Simulation is that it is not distribution dependent. In other models discussed in Section 1.1 it is necessary to limit the distributions to be exponential in order to make the analytical solution tractable. In this research we relax this assumption, while in the cases presented we do assume exponential interarrival and service times, the switching time is allowed to be deterministic. Other assumptions found in the literature, such as every resource must be used at all times, are not present in this model.

Discrete Event Simulation does have some downsides however. One of the major drawbacks is that it is impossible to prove that any given answer is optimal. As a potential counter measure to this OptQuest is tested because it is able to search through the solution space much more efficiently than brute force enumeration; however it could still be that there are better answers that it does not find. Additionally it is much more computationally expensive than analytical methods such as Markov Decision Processes.

SECTION 1.3: HEURISTIC DESCRIPTIONS

After designing the model the next step was to find several practical heuristics for dynamically allocating servers which could be easily implemented by manufacturing firms. One issue with the optimal policies found by analytical methods in the literature is that they may not be easy to interpret and act upon in everyday manufacturing situations. With this is mind several other heuristics were tested and compared using simulation. This research will test the effectiveness of several different heuristics which are described below including: MDP Approximation (MA), Base Stock, Base Percentage(BP), More Parts 1(MP1), More Parts 2(MP2), Expedite, No Blocking, Base Separate Policy (BSP), and OptQuest. It is of interest to this research whether different heuristics perform better in different circumstances. This is important to know for use as a prescriptive solution in different manufacturing environments. We describe each heuristic in detail below.

MDP Approximation (MA)

A similar model to the one presented in this manuscript was analyzed as a MDP by Arumugam et al. (2009), with several key differences: While their objective is also to maximize throughput, they use a preemptive-resume assumption and zero switching time. An optimal solution to their system is provided. The optimal policy to Arumugam et. al (2009), depends on the inventory level at each station and follows a switching curve. Furthermore, it always assigns both servers to the same station. Since we have changed several of the assumptions central to their model the optimal solution to their system can be applied as a heuristic to our model with the same parameters except for the switching time. We will refer to the optimal allocation policy as the MDP Approximation (MA) heuristic throughout the rest of this paper.

<u>Base Stock, Base Percentage, and Expedite</u>

One heuristic is the so called "Base Stock" heuristic, which is characterized by assigning a base stock level to one of the stations. In this heuristic both servers always operate together. Both servers will remain at the "Base" station until a certain inventory level is built up at the other station. When this inventory level is attained both servers switch to the other station until the inventory level is below the base stock inventory level at which time the servers return to "Base" station. For ease of use we will use the notation Base(x, y) to refer to the base stock model where workers stay at station x until there are y parts in inventory at the other station. It should be noted that a number of different policies under this heuristic are possible. The number of different policies is equal to the size of the buffer in the second station. The expedite policy championed by Van Oyen (2001) is covered in this heuristic by Base(1,1). When the base stock level is set to one the effect is for the workers to follow a part through the system. Because the number of possible policies for the Base Stock heuristic varied depending on the buffer size the Base Percentage (BP) heuristic was created. The BP heuristic works in the same manner as the Base Stock heuristic except that it switches when the non-"Base" station is y% full. Therefore the heuristic BP(1,50) would indicate that both servers stayed at Station 1 until Station 2 was 50% full. In the case where 50% of a buffer results in a fraction we have rounded up to the nearest whole number.

More Parts (MP1) and (MP2)

Another heuristic which is used is to always assign both servers to the station with more parts in it. This type of policy is of interest because it is very easy to follow in the manufacturing environment. We generated two heuristics from this idea: More Parts 1(MP1) and More Parts 2(MP2) so named because in MP1 when there is an equal number of parts in each station the workers are both in Station 1. Similarly in the heuristic MP2 ties are broken with both workers going to Station 2.

Base Separate (BSP)

The heuristic suggested by Mayorga et al.(2009) of having one server stationary and moving the other is also tested. We have dubbed this heuristic the Base Separate Policy(BSP). In this heuristic both workers are kept together at one station until the inventory at the other reaches a certain point and one server then switches stations. There are 4 different versions of this heuristic to be tested. Both Station 1 and Station 2 may be used as the "Base" station and either Server 1 or Server 2 may be moved to the other station periodically. For ease of implementation this policy is formulated similarly to the Base Percentage heuristic and uses similar notation, BSP(x, y, z). Where the servers are both at station x until that station is y% full at which time server z switches.

No Blocking

The No Blocking heuristic switches only to avoid blocking or starving. In this way the servers would have no unnecessary idle time and also reduce the number of time a server switched from station to station. The servers would remain at a station until it is no longer possible to operate at that station until action is taken at the other station. For example if both servers are at Station 1 they would remain there until the station became blocked because no work was being processed at Station 2

OptQuest

OptQuest uses several different algorithms in conjunction to effectively search the solution space including Tabu Search and Neural Networks. In this case the solution space is all possible resource allocation policies. Because there are four possibilities for each inventory level, if both buffers contained space for six parts there would be 3.4×10^{38} unique allocation policies possible. Unfortunately OptQuest's exact algorithm is proprietary and unavailable to tweak, however it is widely used for its ability to seamlessly integrate with Arena.

CHAPTER 2: METHOD

SECTION 2.1: DISCUSSION OF PARAMETERS

Before diving deeper into the results it is important to take note of the different parameters which influence system performance. The first of these factors is ρ which is a measure of the traffic intensity. It is essentially a measure of how close to capacity the system is operating and is defined as:

$$\rho = \frac{\lambda}{\frac{\mu_1 + \mu_2}{2}}$$

Switch time(ST) is the time it takes for a server to get from one station to another and begin working. This was measured as a proportion of the average service time in a given scenario and represented by the variable Switching Ratio (SR). The equation definition below uses ST for switching time and SR represents the desired switching ratio.

$$\text{SwitchingTime} = (\text{SwitchingRatio})\left(\frac{1}{\frac{\mu_1+\mu_2}{2}}\right)$$

For example, if SR is 0.5 then the switching time will be equal to half of the average service completion time.

Additionally different service ratios were tested to see the effect of having one server faster than the other. Without loss of generality, we assume Server 2 is faster than Server 1.

$$\text{Service Ratio} = \frac{\mu_2}{\mu_1}$$

The effect of the Buffer Ratio parameter was also explored. The buffer ratio is the ratio of the size of the first buffer to the size of the second buffer. The equation can be seen below.

$$\text{Buffer Ratio} = \frac{B_1}{B_2}$$

The parameters mentioned above were used to test multiple heuristics across a broad spectrum of scenarios in order quantify which heuristics performed better and when.

SECTION 2.2: RESEARCH QUESTIONS

So far we have extensively discussed how this system works. However we have not yet touched on what questions we hope to answer about this system. There are many questions about this system whose answers would be of value in industry. We have developed a few which we believe to be the most important and attempted to answer those.

The first question is: *How does a system without the assumption of pre-emptive resume compare to a system with pre-emptive resume?* Much of the research which has been conducted in the past has included the assumption of pre-emptive resume. It is of interest to determine whether a system which removes this assumption achieves significantly different throughput.

The second question is: *How does non-zero switching time affect the throughput of the system?* Previous systems approximated switching time as a cost to the system. We wanted to quantify how adding switching time to the system would affect the throughput of the system.

The third question is: *How do different cooperative heuristics perform with the addition of switching time?* Arumugam et.al (2009) showed that when there is no cost to switch servers from one station to the other that it is always optimal for the servers to be together. With this in mind we developed several heuristics which always keep both servers at the same station. We will refer to the MP1, MP2, No Blocking, and Expedite heuristics as *cooperative heuristics.* One of the main reasons for using heuristics is the difficulties associated with implementing many of the optimal allocation policies found in other literature. We wanted to see how heuristics which could be easily implemented in industry fared both with no switching time and when switching time was added to the system.

The fourth question is: *How do Base Stock Policies Perform?* After crafting several easily implemented cooperative heuristics we became interested in the performance of the Base Stock heuristic described in Section 1.3. This kind of policy is intriguing because it is extremely adaptable to industrial settings.

The fifth question is: *How do policies with one stationary server perform?* This question was inspired by the Mayorga et al. (2009) paper. In that paper a heuristic was used which held one server stationary and left the other free to move based on an allocation policy. We extended this idea to make such a policy more implementable and wished to see how it performed under a variety of circumstances, especially depending on how much faster the fast server is in relation to the slower server.

The final question is: *How should buffer spaces be allocated given a finite number of spaces?* For example, if the system only has room for 12 buffer spaces how many of those twelve should be placed at the first and second stations respectively? We wanted to determine whether the bulk of buffer spaces should be positioned at the first station, second station, or be allocated equally. This question is important because in many manufacturing environments space is at a premium, thus being able to achieve a higher level of productivity within a given confines would represent a large advantage.

CHAPTER 3: RESULTS AND CONCLUSIONS

SECTION 3.1: PRELIMINARY RESULTS

The first research question in Section 2.2 asked how a system without the assumption of preemptive resume compared to a system with preemptive resume. A design of experiments with 3 factors each at 3 levels was conducted to answer this question. A full list of the experiments and parameters is available in Figure A-1 in Appendix A. This design of experiment did not include switching time in order to more directly answer the motivating question. Thus, for this set of experiments when the allocation policy dictated that servers should switch station they were able to do so instantly.

Recall that when preemption is allowed and there is no switching time the MA heuristic is optimal as per Arumugam et al. (2009). The simulation was then run across all scenarios in the DOE with MA heuristic. The results of the simulation, in which preemption is not allowed, were compared to the results when preemption is allowed which can be computed analytically (Arumugam et al., 2009) when switching time is zero. A full table of the results is located in Figure A-2 in Appendix A. The results are compared in Figure 3.1 seen below.

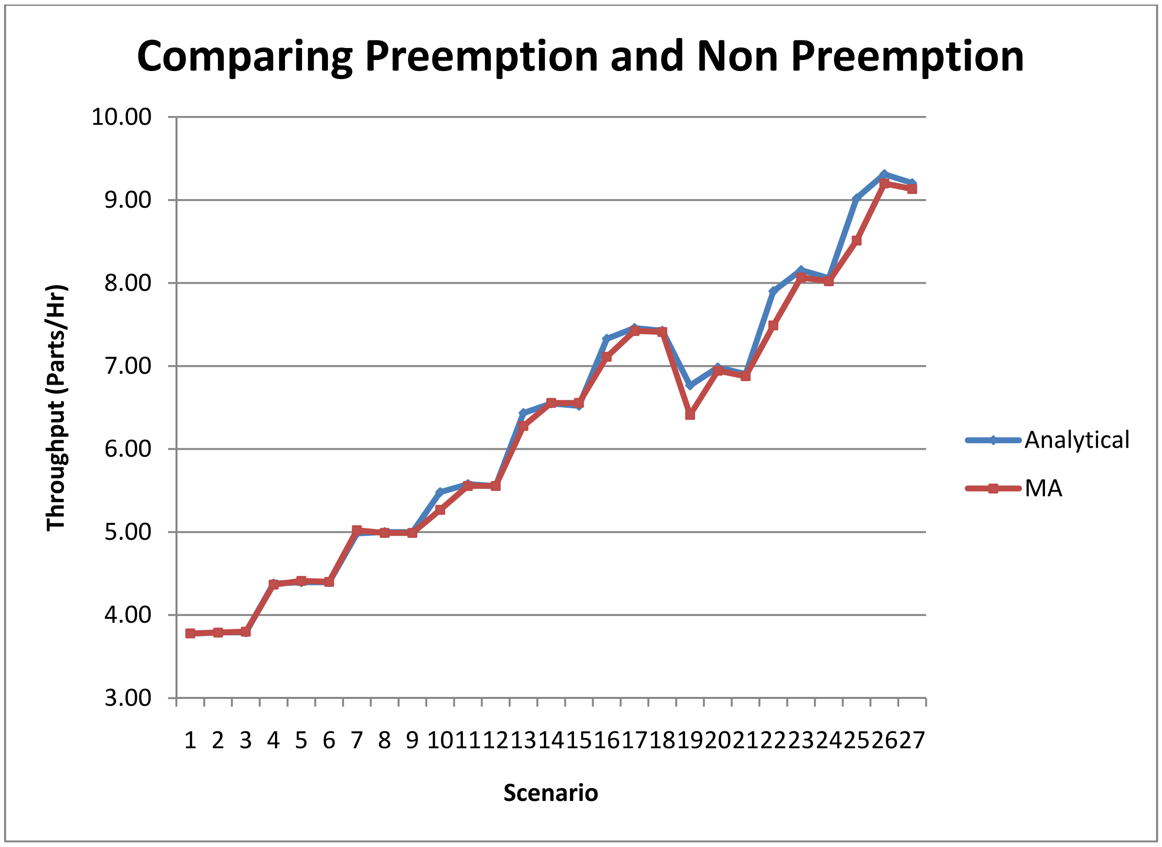


Figure 3.1: Comparing Systems with and without the preemptive resume assumption when switching time is zero

Figure 3.1 suggests that there is not a significant difference between the long run throughput determined analytically by Arumugam et al. (2009) and throughput of the simulation with the MA heuristic which removes the preemption assumption. As confirmation a student's t-test was performed and reported a p-value of .42 against the hypothesis that the difference in the samples was zero. This result confirmed our initial finding that there is not a significant difference in throughput between two systems where the only difference is the assumption of preemptive resume.

The OptQuest, MP1, No Blocking, and Expedite allocation policies were also tested using these same parameters to gain an initial understanding of how different allocation heuristics performed. The full results of these tests can be seen in Figure A-2 in Appendix A. The results were somewhat surprising and are graphically represented in Figure 3.2.

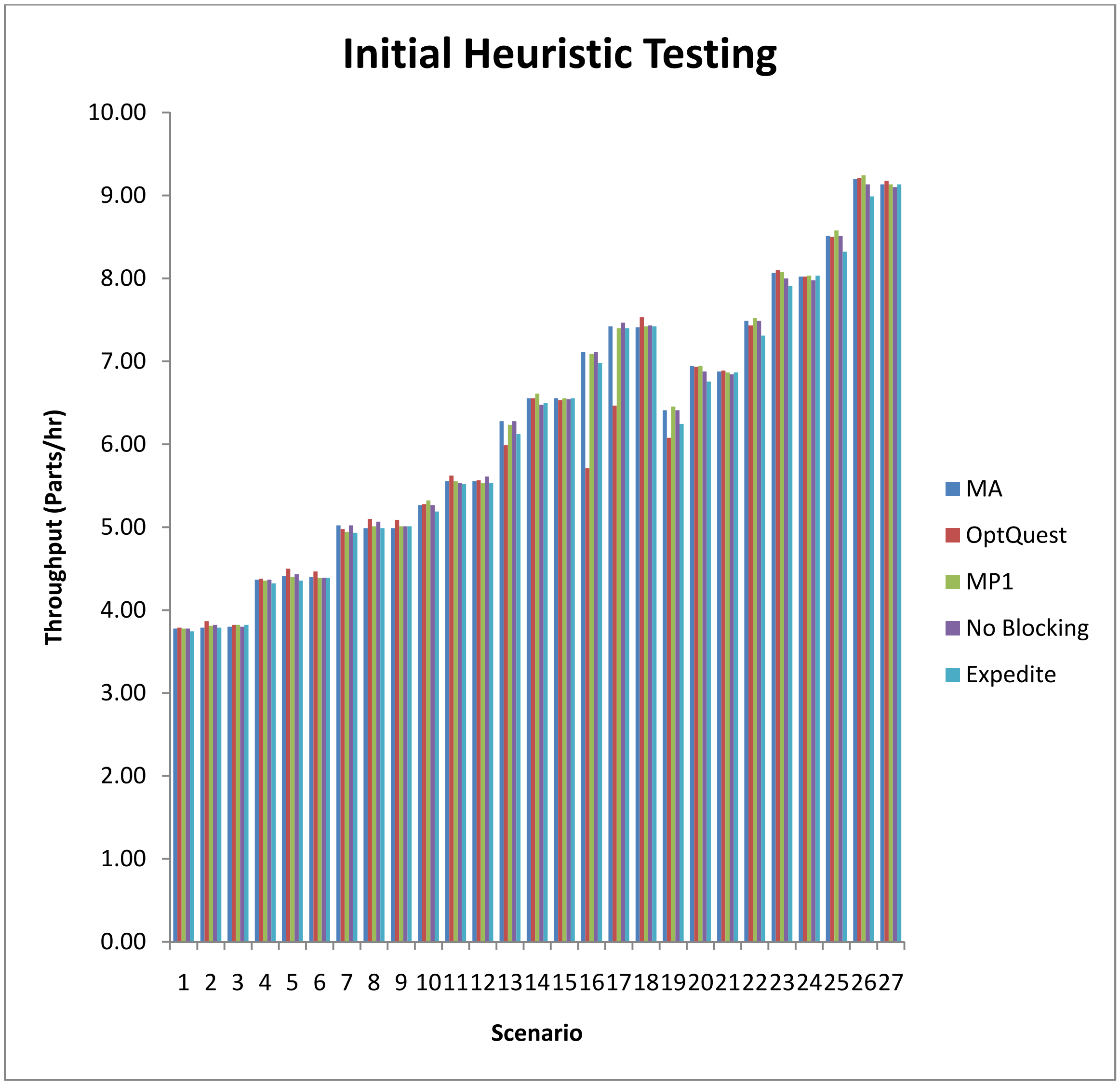


Figure 3.2: Comparison of Different Heuristics on Initial Testing

A summary of the results is located in Table 3.1. In Table 3.1 the average Rate was found by averaging the rate of throughput across all 27 tests. The average percentage difference from the MA heuristic was computed by finding the percentage difference for all 27 scenarios tested and then averaging them.

Table 3.1: Summary of Design of Experiments

	Avg .Rate (Parts/hr)	Avg. % Improvement from MA	% of tests worse than MA
MA	6.22	-	-
Expedite	6.15	-.97%	70%
OptQuest	6.13	-1.08%	33%
MP1	6.23	.1%	33%
No Blocking	6.21	-.02%	37%

The Expedite policy proposed by Van Oyen (2001) did not fare well, and on average produced nearly 1% less parts per hour than the MA policy. Additionally 70% of the tests run resulted in a lower throughput.

The MP1 and No Blocking heuristics performed slightly better than the MA policy. The MP1 heuristic performed better on average as well as performing better in two thirds of all scenarios tested. The No Blocking Heuristic performed slightly worse overall, however it performed worse than the MA heuristic in only 37% of the cases tested.

On average OptQuest performed worse than the MA heuristic. However Optquest did outperform the MA heuristic on two thirds of the scenarios run. A large reason for the poor overall performance of Optquest was due to two scenarios in which the throughput was nearly 20% less than that of the MA heuristic. Excluding those outliers the OptQuest policy was .98% better than the MA policy. As mentioned in Section 2.1 there is no certainty that OptQuest will find an optimal solution to any given scenario. This is illustrated by the extremely low throughput of the 2 outlying scenarios. Additionally when using OptQuest the user must decide on criteria to end the search for the best solution. We stopped the search after 4000 policies were tested because the improvement in solutions was rapid initially and reached a plateau. After reaching the plateau many solutions were found which provided equal throughput. This resulted in a 3.5 hour runtime required to compute a solution. This large runtime led to the conclusion that OptQuest does not seem to be a viable long term option. The final and most important issue with OptQuest is that the solutions which are reported are not necessarily any more easily implementable than the MA heuristic provided by Arumugam et al. (2009). Because there may be many solutions with the same throughput the user must sift through all of the solutions found to determine which one is the most implementable. This is not a feasible solution for implementation in the field. The difference between two example policies is demonstrated in Figure 3.3. The MA heuristic is actually much easier to implement than the OptQuest heuristic. After taking into account time considerations as well as implementation issues a decision was made not include OptQuest in further studies.

	Station 2							
Station 1	0	1	2	3	4	5	6	7
0	1	4	4	4	4	4	4	4
1	1	1	1	1	1	4	4	4
2	1	1	1	1	1	1	4	4
3	1	1	1	1	1	1	1	4
4	1	1	1	1	1	1	1	4
5	1	1	1	1	1	1	1	4
6	1	1	1	1	1	1	1	4
7	1	1	1	1	1	1	1	4

MA Heuristic

	Station 2							
Station 1	0	1	2	3	4	5	6	7
0	1	4	4	4	4	4	4	4
1	1	1	1	1	4	1	4	1
2	1	4	4	1	1	1	1	4
3	1	1	1	1	1	4	4	1
4	1	4	4	1	4	1	4	1
5	1	4	4	4	4	1	4	4
6	1	4	4	1	1	4	4	4
7	1	4	1	4	1	4	4	4

OptQuest

Figure 3.3: Comparison of Optquest and MA policies

This initial study answered the research question of how preemption affected a system with no switching time. Pre-emption was determined not to have a large effect when switching time was not present. Another result of this initial study is the decision to not use OptQuest as a heuristic from this point forward. OptQuest was determined to be too computationally expensive for the results it achieved and thus was eliminated from the list of potential heuristics which should be used. These results led us to design a much larger test for the heuristics tested in this study in addition to others mentioned in Section 1.3.

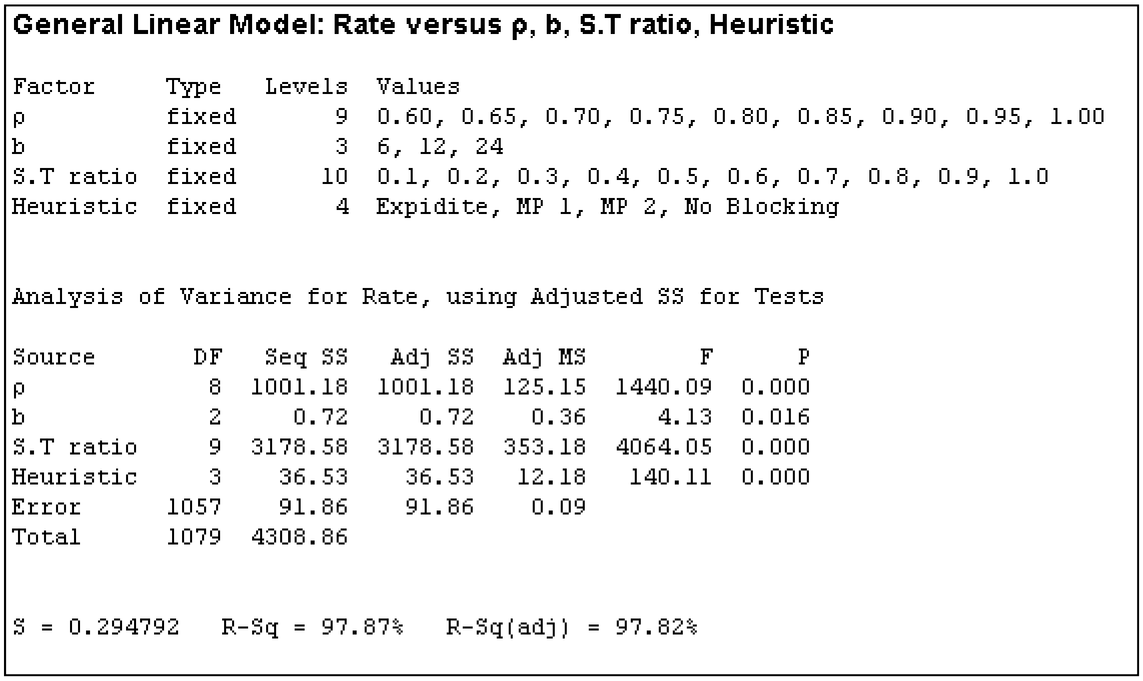
SECTION 3.2: COOPERATIVE HEURISTICS

The MA and OptQuest heuristics present a major challenge for implementation in an industrial setting. Difficulty of implementation in conjunction with the findings of the preliminary study led to the creation of a second study intended to answer the second and third research questions in Section 2.2. The second research question posed in Section 2.2 inquired how the addition of switching time affected the throughput. Thus a large scale set of scenarios was constructed which added the element of switching time to the system

The third research question in Section 2.2 wishes to determine how cooperative heuristics perform in a system with switching time. Arumugam et. al. (2009) showed that when there are no switching costs it is always optimal for both workers to be together. Based on that finding the MP1, MP2, Expedite, and No Blocking Heuristics were devised. We have dubbed these heuristics *cooperative* heuristics, a full description of each heuristic is presented in Section 1.3. These cooperative heuristics were then tested under many different scenarios to determine if there was a significant difference in the performance of each heuristic.

For this set of scenarios parameters ρ, Switching Rato, B_1, and B_2 were varied in this test in the ranges $\rho = \{0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00\}$, Switching Ratio $= \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00\}$, and B_1=B_2= $\{6, 12, 24\}$. The buffer sizes for this set of scenarios were assumed to always be equal. The heuristics tested were the MP1, MP2, No Blocking, and Expedite heuristics. Full results can be seen in the supplemental file.

A general linear model (GLM) was constructed from the set of scenarios outlined above to determine which parameters most affected the throughput. A table of the results from the GLM can be seen in Figure 3.4.



General Linear Model: Rate versus ρ, b, S.T ratio, Heuristic

```
Factor      Type   Levels  Values
ρ           fixed       9  0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00
b           fixed       3  6, 12, 24
S.T ratio   fixed      10  0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Heuristic   fixed       4  Expidite, MP 1, MP 2, No Blocking


Analysis of Variance for Rate, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS        F      P
ρ              8  1001.18  1001.18   125.15  1440.09  0.000
b              2     0.72     0.72     0.36     4.13  0.016
S.T ratio      9  3178.58  3178.58   353.18  4064.05  0.000
Heuristic      3    36.53    36.53    12.18   140.11  0.000
Error       1057    91.86    91.86     0.09
Total       1079  4308.86


S = 0.294792   R-Sq = 97.87%   R-Sq(adj) = 97.82%
```

Figure 3.4: General Linear Model for Cooperative Heuristics

Interestingly all the factors were found to have a significant impact on throughput at a 95% significance level. Figure 3.4 shows switching time has the largest impact on throughput. Similarly ρ or traffic intensity has a very large impact. However it is also shown that b = B_1=B_2 and which heuristic was used were also significant factors, though not as significant as switching time and traffic intensity. As expected, throughput decreased as switching time increased. This result can be seen graphically in Figure 3.5. Likewise as traffic intensity increased the throughput was adversely affected as can be seen in Figure 3.6.

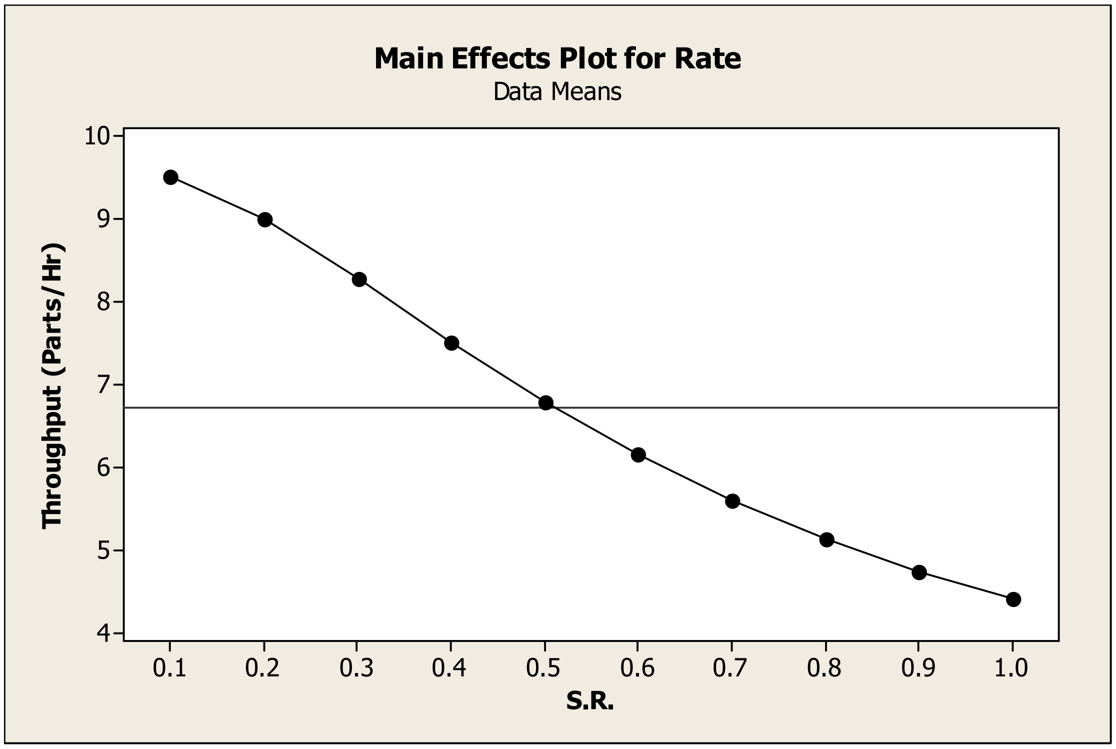


Figure 3.5: Effect of Switching Time on Throughput

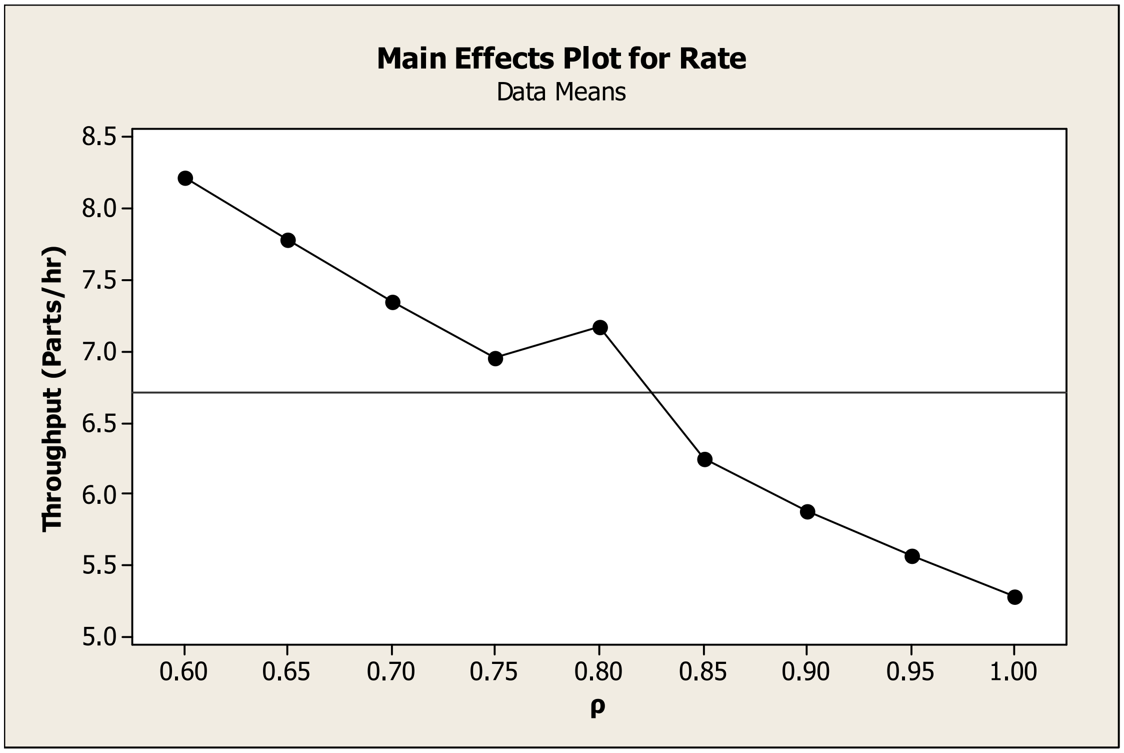


Figure 3.6: Effect of Traffic Intensity on Throughput

The effect of buffer size can be seen in Figure 3.7, and the effect of different heuristics can be seen in Figure 3.8. It appears that increasing buffer size has a diminishing rate of return. We will explore this hypothesis more in Section 3.5.

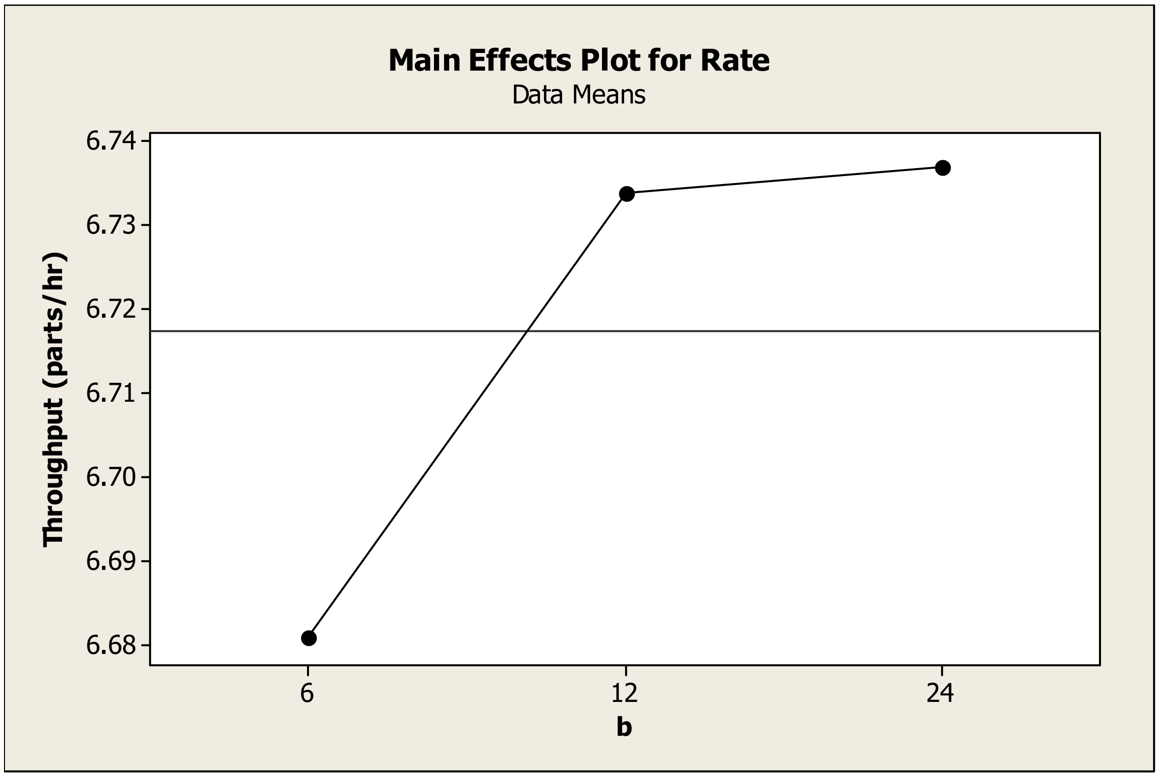


Figure 3.7: Effect of Buffer Size on Throughput

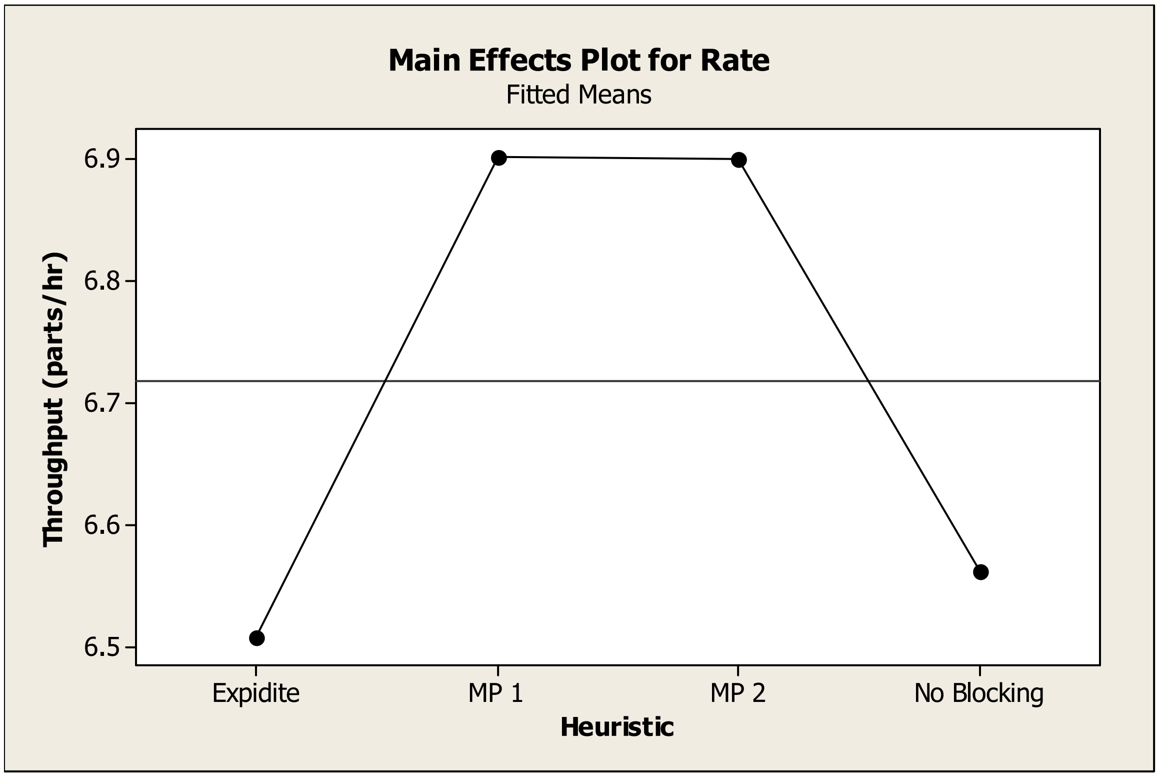
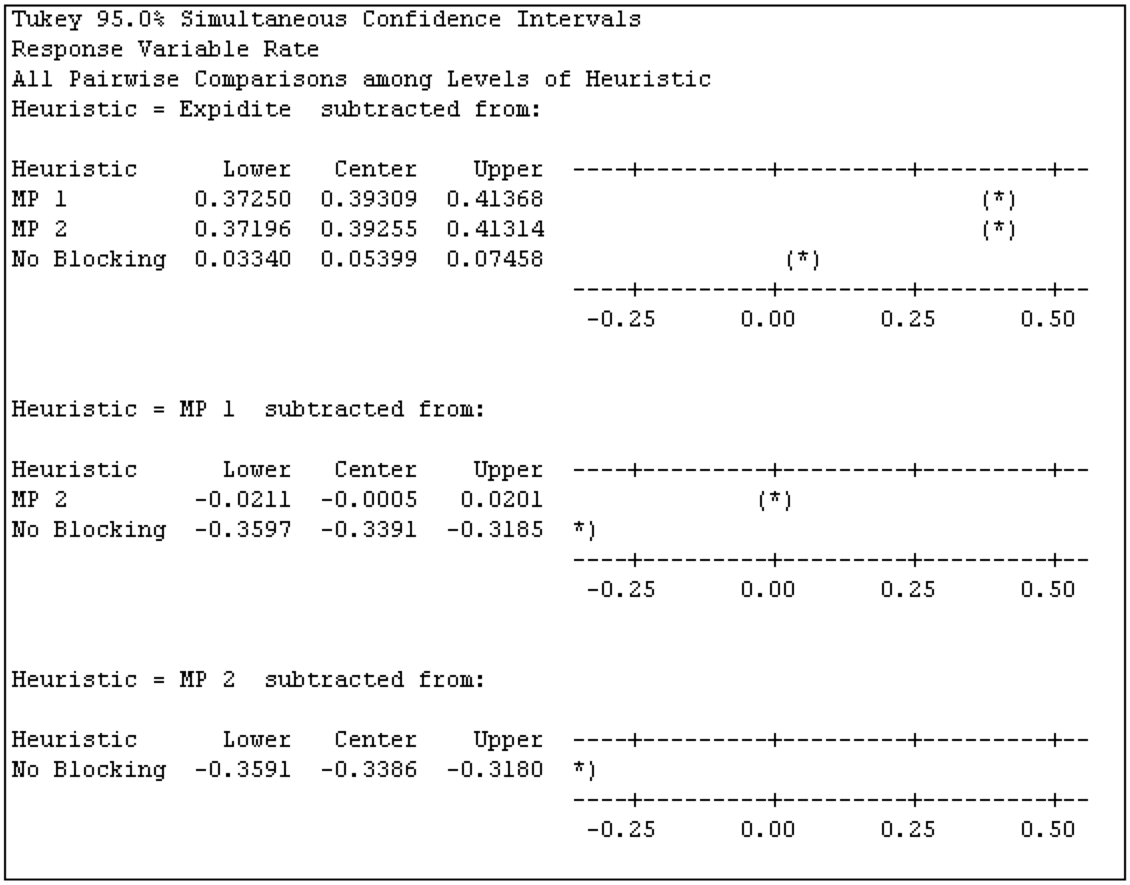


Figure 3.8: Effect of Heuristic on Throughput

As can be seen in the Figure 3.6, the MP1 and MP2 heuristics seem to fare much better than the No Blocking and Expedite heuristics. To substantiate these claims a Tukey's Multiple Comparisons Test was performed at a level of 95% confidence and is located in Figure 3.9.

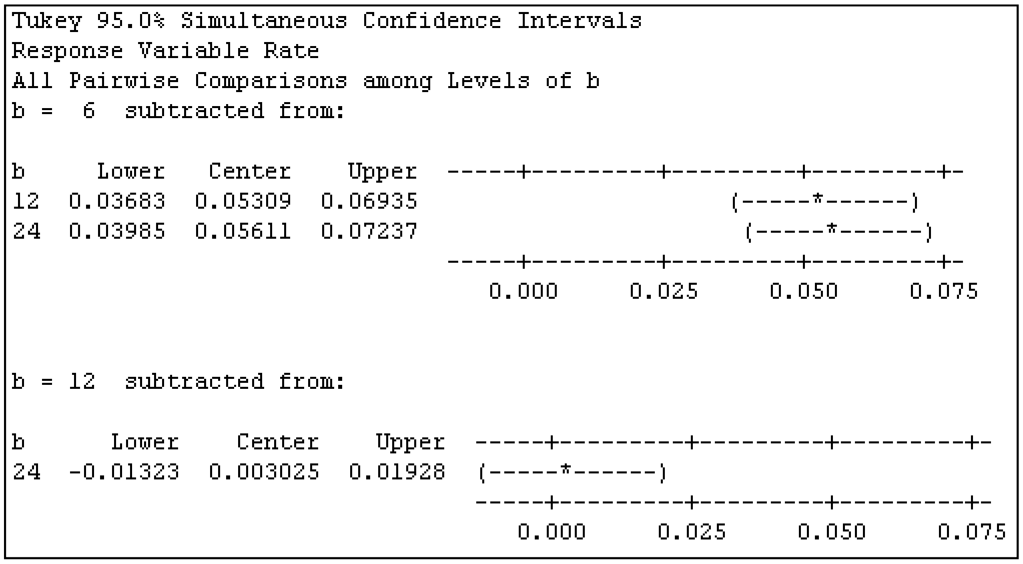


```
Tukey 95.0% Simultaneous Confidence Intervals
Response Variable Rate
All Pairwise Comparisons among Levels of Heuristic
Heuristic = Expidite  subtracted from:

Heuristic      Lower   Center    Upper  ----+---------+---------+---------+--
MP 1         0.37250  0.39309  0.41368                              (*)
MP 2         0.37196  0.39255  0.41314                              (*)
No Blocking  0.03340  0.05399  0.07458                (*)
                                        ----+---------+---------+---------+--
                                         -0.25      0.00      0.25      0.50


Heuristic = MP 1  subtracted from:

Heuristic      Lower   Center    Upper  ----+---------+---------+---------+--
MP 2         -0.0211  -0.0005   0.0201              (*)
No Blocking  -0.3597  -0.3391  -0.3185  *)
                                        ----+---------+---------+---------+--
                                         -0.25      0.00      0.25      0.50


Heuristic = MP 2  subtracted from:

Heuristic      Lower   Center    Upper  ----+---------+---------+---------+--
No Blocking  -0.3591  -0.3386  -0.3180  *)
                                        ----+---------+---------+---------+--
                                         -0.25      0.00      0.25      0.50
```

Figure 3.9: Tukey's Multiple Comparison Test on Collaborative Heuristics

In a Tukey's Multiple Comparison Test 95% confidence intervals are constructed around the difference in means of two heuristics. If a confidence interval contains 0 there is not a significant difference in the means of the two heuristics. If the lower bound of the confidence interval is positive the heuristic being subtracted is significantly worse than the heuristic to which it is being compared. Conversely if upper bound of the confidence interval is negative then the heuristic being subtracted is significantly better than the heuristic to which it is being compared. Thus Figure 3.9 shows that the MP1 and MP2 heuristics are not significantly different from one another, but are significantly better than the Expedite and No Blocking Heuristics. Additionally the No Blocking heuristic performs significantly better than the Expedite policy.

A Tukey's Multiple Comparison Test was also performed for the effect of buffer size and can be seen in Figure 3.10.



```
Tukey 95.0% Simultaneous Confidence Intervals
Response Variable Rate
All Pairwise Comparisons among Levels of b
b =  6  subtracted from:

b     Lower   Center    Upper  -----+---------+---------+---------+-
12  0.03683  0.05309  0.06935                      (-----*------)
24  0.03985  0.05611  0.07237                       (-----*------)
                               -----+---------+---------+---------+-
                                  0.000     0.025     0.050     0.075


b = 12  subtracted from:

b      Lower    Center    Upper  -----+---------+---------+---------+-
24  -0.01323  0.003025  0.01928  (-----*------)
                                 -----+---------+---------+---------+-
                                    0.000     0.025     0.050     0.075
```

Figure 3.10: Tukey's Multiple Comparison Test on Buffer Size

Buffer sizes of 6 performed worse than the larger buffer sizes of 12 and 24. However there was not a significant difference in the performance of buffer sizes 12 and 24. This leads us to believe that there is a diminishing rate of returns on buffer sizing.

Recall from the model description in Section 2.1 that a part balks from the system if Station 1 is full upon arrival. When buffer sizes are small there is a much greater chance that B_1 will be full upon arrival and the part will balk from the system resulting in a lower throughput. As buffer size increases the chance of balking decreases which results in a higher throughput. However the probability of balking can only decrease to a certain point and the benefit of larger buffers is reduced.

The final interesting result from this set of scenarios is that the allocation policy chosen has the greatest effect for moderate switching times. This can be seen graphically in Figure 3.11 below. If there were no interaction between switching time and heuristic the lines would be parallel. However they are closer together when switching time is small and when switching time is large but further apart at points in between.

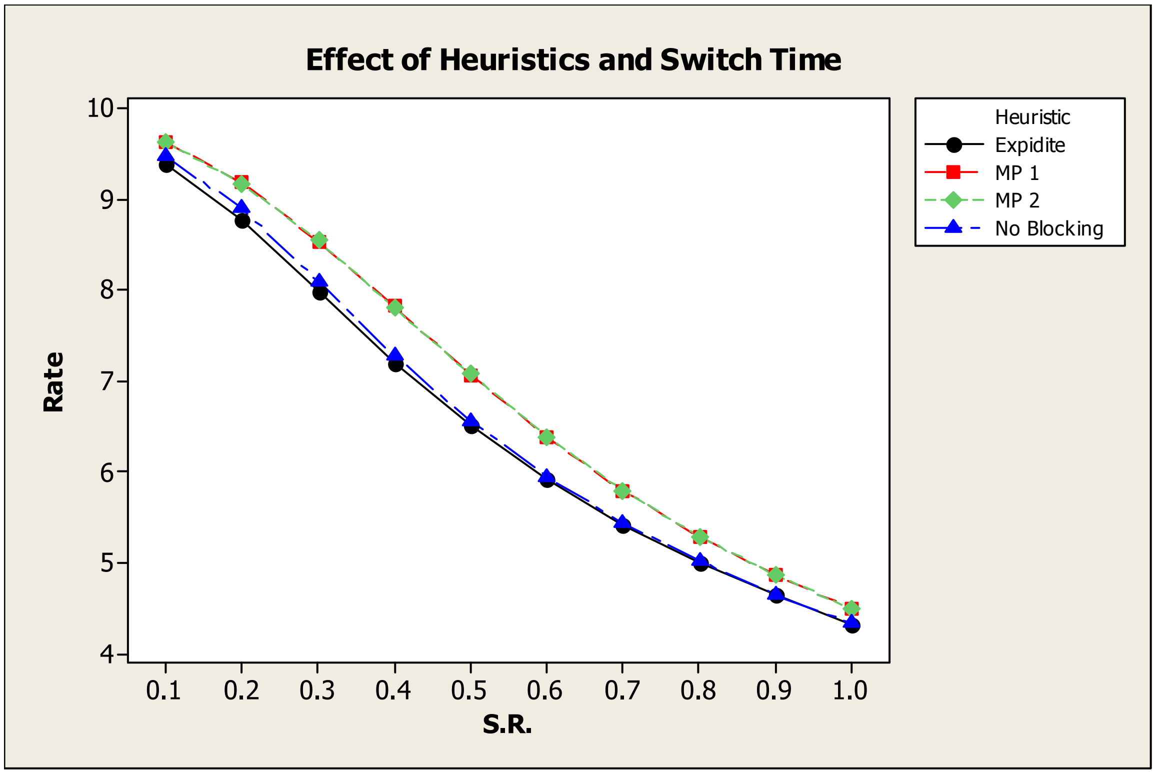


Figure 3.11: Interaction of Switching Time and Heuristics

This leads to the conclusion that if switching time is very small then the value of using different cooperative allocation policies is minimal. Conversely when the switching time is large in comparison to the service time the benefit of changing allocation policies is mitigated, and improvement efforts should be focused on reducing the switching time.

The research question of how switching time affects throughput was answered in this section. Throughput of the system decreases as switching time in increased. The research question regarding how cooperative heuristics performed under different scenarios is also addressed. The MP1 and MP2 heuristic perform significantly better than the No Blocking Heuristic and the Expedite heuristic, but are not significantly different than each other. Additionally the effect of buffer size has a diminishing rate of return as the size of the buffer increases. Finally, it was shown that the effect of which cooperative heuristic is used is greatest when switching times are moderate. This information is valuable for use in industrial settings to know under what conditions resource allocation policies should be closely re-evaluated.

SECTION 3.3: BASE STOCK HEURISTIC

The success of other easily implementable heuristics led us to the fourth research question in Section 2.2 which addresses the performance of a Base Stock heuristic. The Base Stock heuristic is of interest because of its ease of implementation in an industrial setting. As mentioned in Section 1.3 the base stock heuristic places both servers at one station until a specified number of parts are at the other station at which point both servers then switch stations. The notation for this is Base(x, y) where a worker stays at station x until there are y parts at the other station.

To test the effectiveness of the Base Stock heuristic a study similar to the one presented in Section 3.2 was used. Parameters ρ, Switching Rato, B_1, and B_2 were varied in this test in the ranges ρ = {0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00}, Switching Ratio = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00}, and B_1=B_2= {6, 12, 24}. Additionally the number of Base Stock heuristics which can be run on a particular scenario depends on the size of the buffer. For example in the case of B_1=B_2= 6 there are 12 possible heuristics, six when the servers are placed at Station 1 and six more when the servers are based at Station 2.

The Base Stock Heuristic is implemented by running all possible Base(x,y) combinations for each scenario and choosing the best one. The resulting policy is called the Base Stock Heuristic. This method requires large amount of computation because every possible Base(x, y) policy must be evaluated and compared. The Base Stock Heuristic was computed and compared to the Cooperative heuristics discussed in

Section 3.2 and can be seen in Figure 3.12. A full table of results of can be found in the supplemental file.

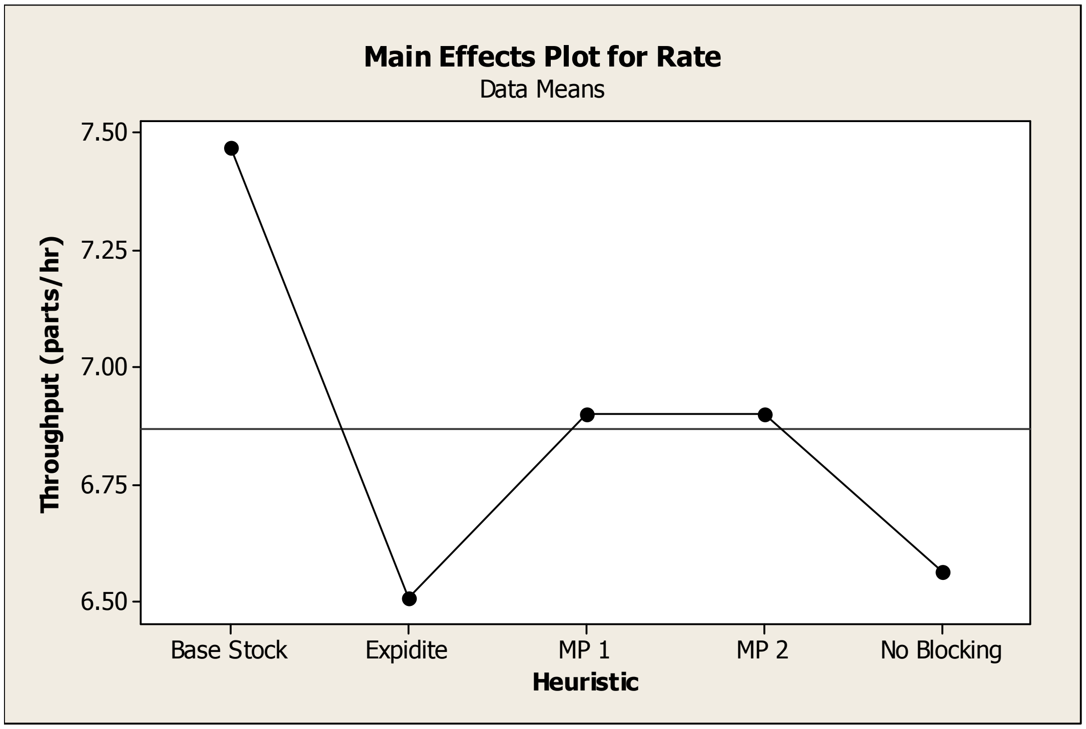
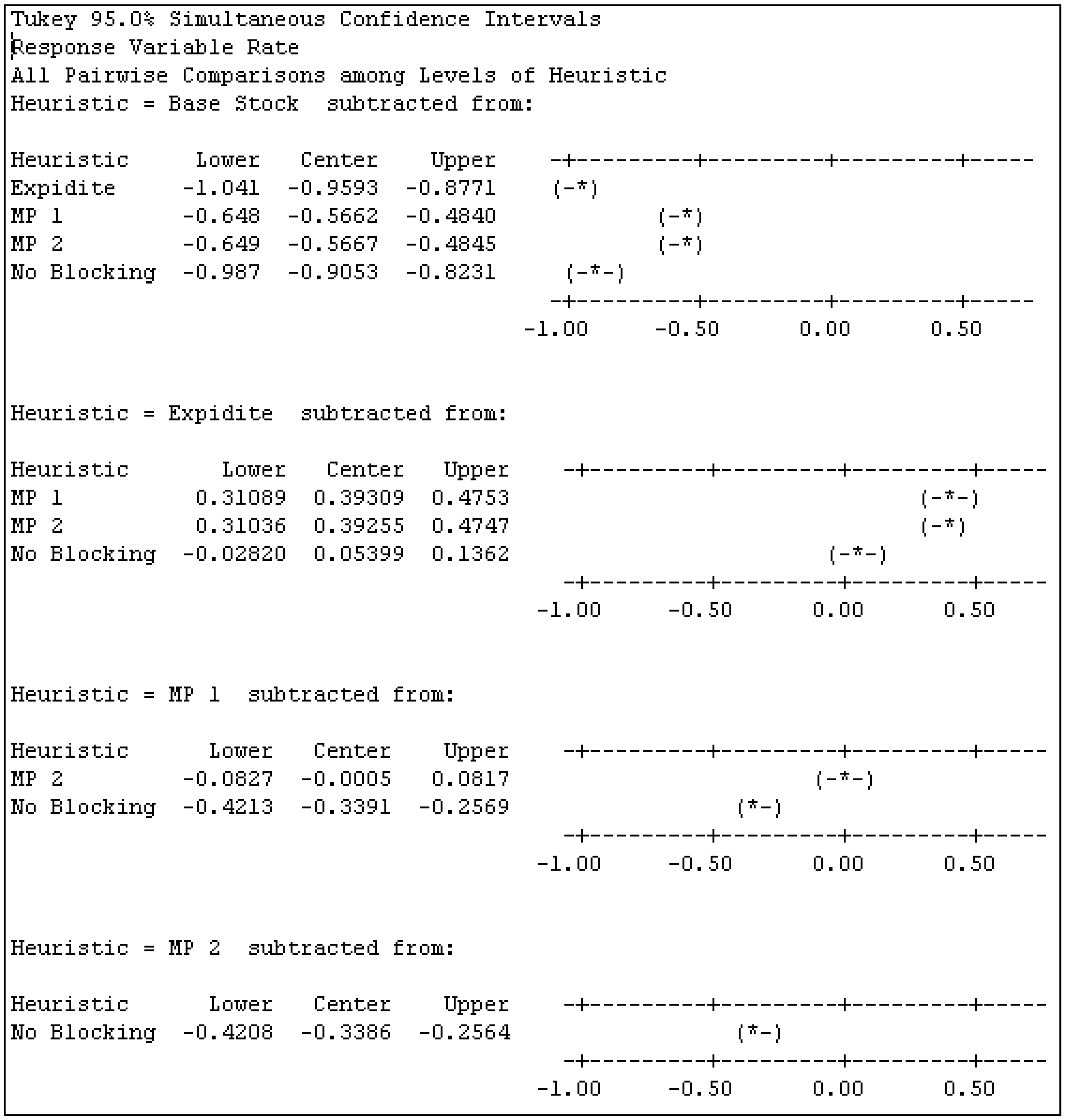


Figure 3.12: Effect of Heuristic on Throughput with Base Stock Heuristic

Figure 3.12 suggests that the Base Stock Heuristic performs better than the Cooperative heuristics discussed in Section 3.2. While the Expedite and No Blocking heuristics were concluded to perform worse than the MP1 and MP2 heuristics they are included to give a sense of scale to all the heuristics discussed. A Tukey's Multiple Comparison Test was run on the different heuristics to verify the conclusion that the Base Stock heuristic performs better than the Cooperative heuristics and can be seen in Figure 3.13.



```
Tukey 95.0% Simultaneous Confidence Intervals
Response Variable Rate
All Pairwise Comparisons among Levels of Heuristic
Heuristic = Base Stock  subtracted from:

Heuristic     Lower   Center    Upper    -+---------+---------+---------+-----
Expidite     -1.041  -0.9593  -0.8771    (-*)
MP 1         -0.648  -0.5662  -0.4840            (-*)
MP 2         -0.649  -0.5667  -0.4845            (-*)
No Blocking  -0.987  -0.9053  -0.8231     (-*-)
                                         -+---------+---------+---------+-----
                                       -1.00     -0.50      0.00      0.50


Heuristic = Expidite  subtracted from:

Heuristic       Lower   Center   Upper    -+---------+---------+---------+-----
MP 1          0.31089  0.39309  0.4753                               (-*-)
MP 2          0.31036  0.39255  0.4747                               (-*)
No Blocking  -0.02820  0.05399  0.1362                        (-*-)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50


Heuristic = MP 1  subtracted from:

Heuristic      Lower   Center    Upper    -+---------+---------+---------+-----
MP 2         -0.0827  -0.0005   0.0817                       (-*-)
No Blocking  -0.4213  -0.3391  -0.2569                 (*-)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50


Heuristic = MP 2  subtracted from:

Heuristic      Lower   Center    Upper    -+---------+---------+---------+-----
No Blocking  -0.4208  -0.3386  -0.2564                 (*-)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50
```

Figure 3.13: Tukey's Multiple Comparison Test for Base Stock Heuristic

Tukey's Multiple Comparison Test confirms that the Base Stock heuristic performs better than the Cooperative heuristics. A key result of Section 3.2 was that the effect of cooperative heuristics was greatest at moderate switching times. Figure 3.14 shows the interaction between heuristics and switching time.

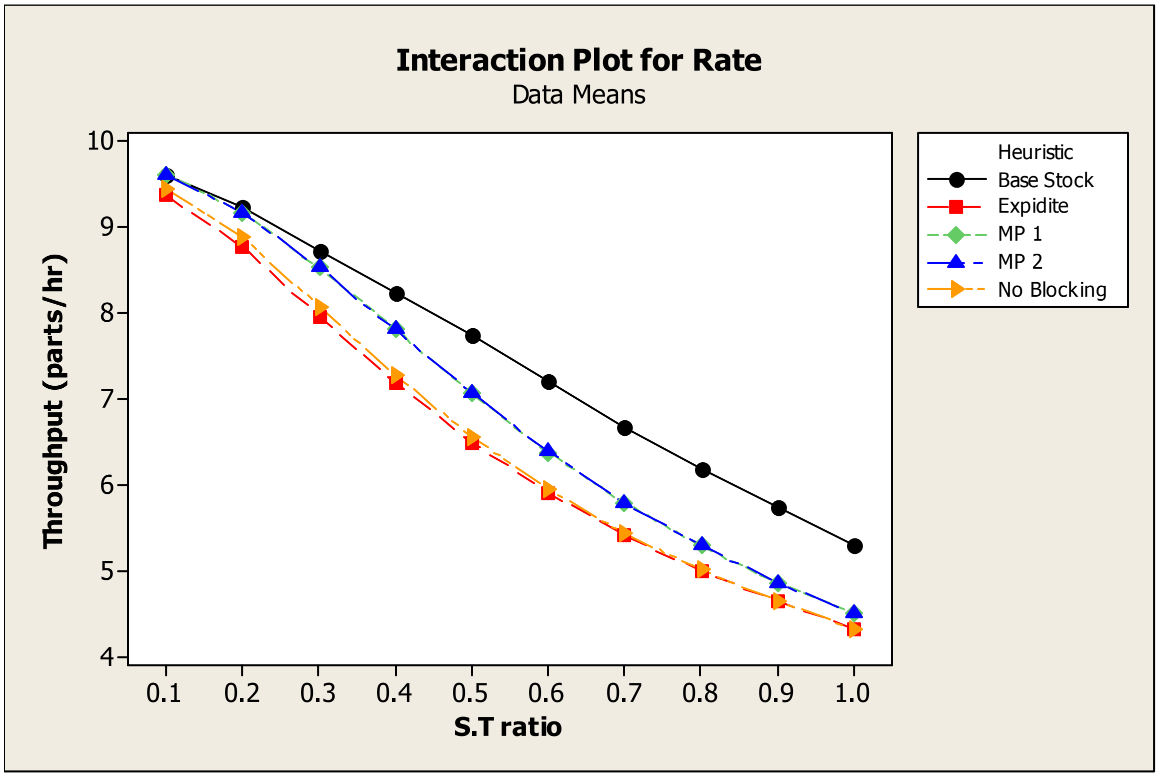


Figure 3.14: Interaction of Base Stock Heuristic and Switching Time

Figure 3.14 shows that the benefit of using the Base Stock Heuristic does not diminish as switching time increases in the same manner as the Cooperative heuristics. This indicates that the Base Stock Heuristic is more robust in relation to switching time.

One issue associated with the Base Stock heuristic is that is requires a large amount of computation to implement. The varying number of Base(x, y) policies depending on Buffer size presents a large obstacle in implementation. Thus the Base Percentage (BP) heuristic was created. The notation for this policy is BP(x, z) where both servers stay at Station x until the other station is z% filled. A policy such as this enables a more robust usage of the Base Stock heuristic because the same policy can be used across many different buffer sizes.

The results from testing the Base Stock heuristic showed a BP(2,85) heuristic was used in a majority of cases for the Base Stock Heuristic. However in the interest of completeness it was decided to test Base Percentage Policies: BP(1,E), BP(1,50), BP(1,85), BP(2,E), BP(2,50), and BP(2,85). The argument E in this case indicates an Expedite policy. Therefore in the BP(2,E) case both servers stay at Station 2 until there is at least one part in Station 1. These 6 new policies are compared to the Base Stock heuristic in Figure 3.15.

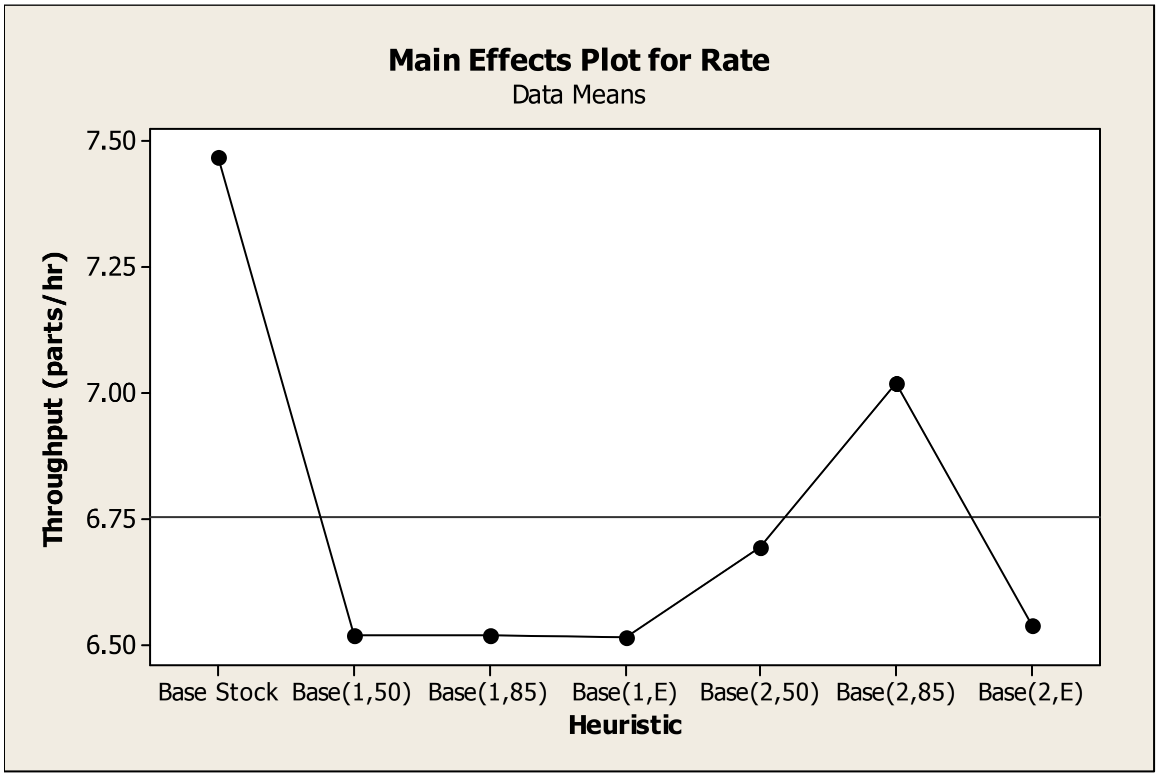
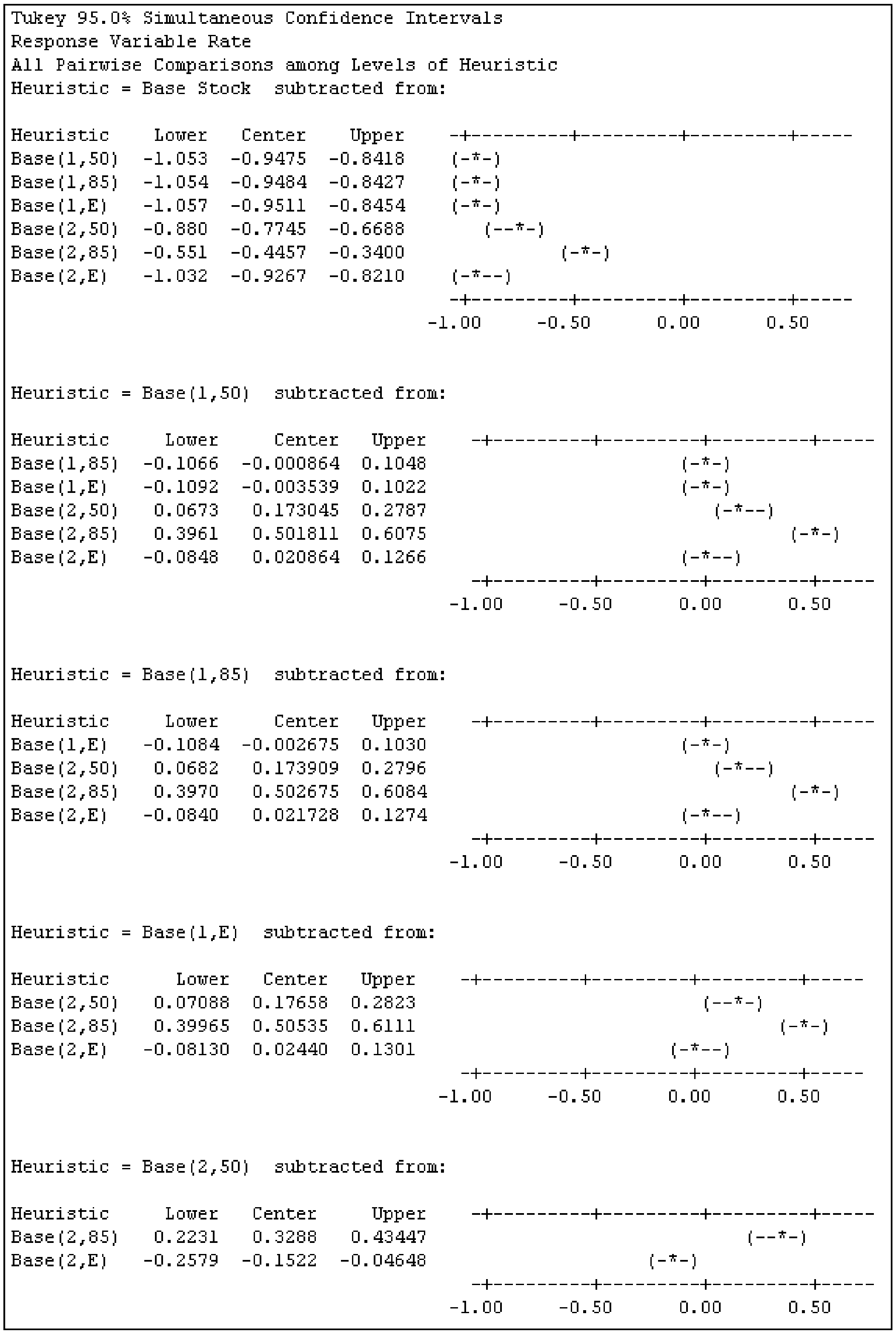


Figure 3.15: Comparing Base Stock an Base Percentage Heuristics

Figure 3.13 indicates that the Base Stock Heuristic performs better than any of the Base Percentage Heuristics, and Tukey's Multiple Comparison Test was again performed to assess the validity of this claim. Figure 3.16 shows the results.



```
Tukey 95.0% Simultaneous Confidence Intervals
Response Variable Rate
All Pairwise Comparisons among Levels of Heuristic
Heuristic = Base Stock  subtracted from:

Heuristic    Lower   Center    Upper    -+---------+---------+---------+-----
Base(1,50)  -1.053  -0.9475  -0.8418    (-*-)
Base(1,85)  -1.054  -0.9484  -0.8427    (-*-)
Base(1,E)   -1.057  -0.9511  -0.8454    (-*-)
Base(2,50)  -0.880  -0.7745  -0.6688       (--*-)
Base(2,85)  -0.551  -0.4457  -0.3400              (-*-)
Base(2,E)   -1.032  -0.9267  -0.8210    (-*--)
                                        -+---------+---------+---------+-----
                                      -1.00     -0.50      0.00      0.50


Heuristic = Base(1,50)  subtracted from:

Heuristic     Lower     Center   Upper    -+---------+---------+---------+-----
Base(1,85)  -0.1066  -0.000864  0.1048                       (-*-)
Base(1,E)   -0.1092  -0.003539  0.1022                       (-*-)
Base(2,50)   0.0673   0.173045  0.2787                          (-*--)
Base(2,85)   0.3961   0.501811  0.6075                                 (-*-)
Base(2,E)   -0.0848   0.020864  0.1266                       (-*--)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50


Heuristic = Base(1,85)  subtracted from:

Heuristic     Lower     Center   Upper    -+---------+---------+---------+-----
Base(1,E)   -0.1084  -0.002675  0.1030                       (-*-)
Base(2,50)   0.0682   0.173909  0.2796                          (-*--)
Base(2,85)   0.3970   0.502675  0.6084                                 (-*-)
Base(2,E)   -0.0840   0.021728  0.1274                       (-*--)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50


Heuristic = Base(1,E)  subtracted from:

Heuristic      Lower   Center   Upper    -+---------+---------+---------+-----
Base(2,50)   0.07088  0.17658  0.2823                          (--*-)
Base(2,85)   0.39965  0.50535  0.6111                                 (-*-)
Base(2,E)   -0.08130  0.02440  0.1301                       (-*--)
                                         -+---------+---------+---------+-----
                                       -1.00     -0.50      0.00      0.50


Heuristic = Base(2,50)  subtracted from:

Heuristic     Lower   Center     Upper    -+---------+---------+---------+-----
Base(2,85)   0.2231   0.3288   0.43447                             (--*-)
Base(2,E)   -0.2579  -0.1522  -0.04648                    (-*-)
                                          -+---------+---------+---------+-----
                                        -1.00     -0.50      0.00      0.50
```

Figure 3.16: Tukey's Multiple Comparison Test for Base Stock Heuristic and Base Percentage heuristics

Thus it can be said that overall the Base Stock Heuristic out-performs any of the Base Percentage Heuristics. However there was an additional finding. Figure 3.17 shows the effect of the different heuristics across switching times.

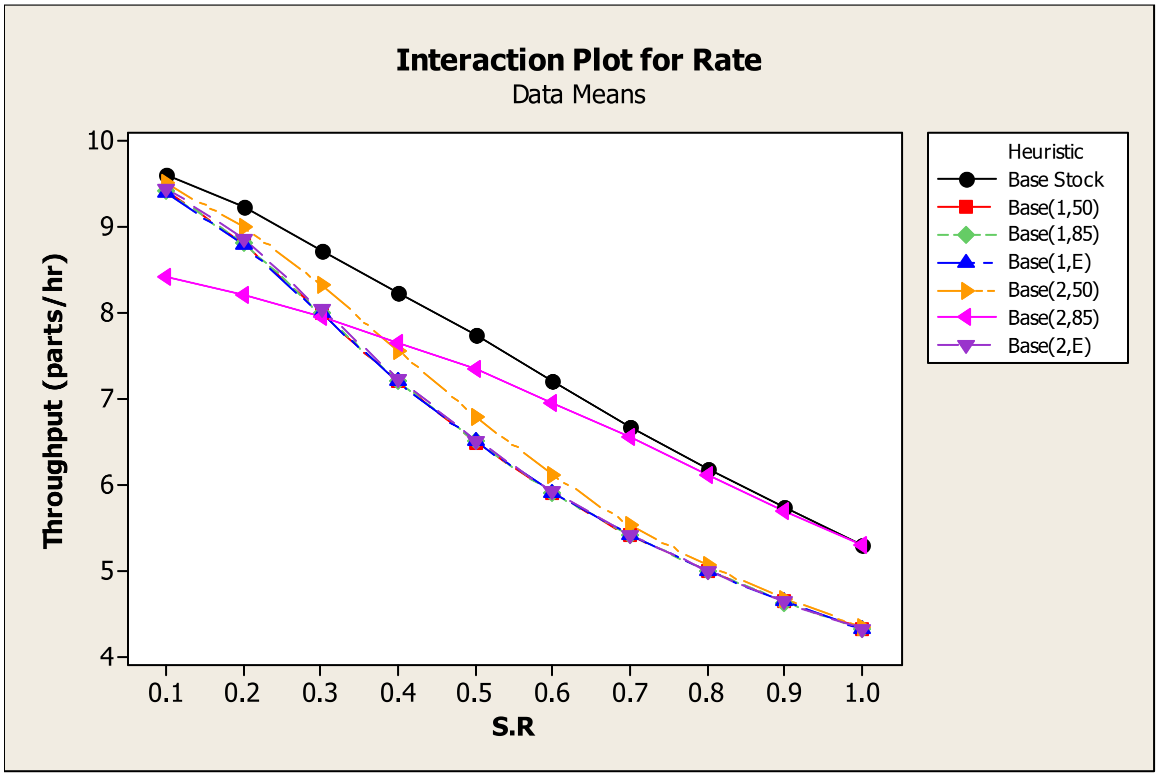


Figure 3.17: Comparing Base Stock and Base Percentage Policies to Switching Time

Figure 3.17 shows when switching time is low several of the Base Percentage heuristics perform nearly as well as the Base Stock policy. However as switching time increases the BP(2,85) policy out performs the other Base Percentage heuristics. In fact as switching time increases BP(2,85) approaches the Base Stock policy. This says that in an environment with high switching time a BP(2,85) policy may be substituted for the Base Stock heuristic without a large loss in performance. Additionally the computation time required for the BP(2,85) policy is much less than the computation time required for the Base Stock heuristic because not every combination must be evaluated. The BP(2,50) heuristic out performs the rest of the Base Percentage heuristics when the Switching Ratio is below 0.4. Using the BP(2,50) heuristic when there is a low Switching Ratio would enable the practitioner to save large amount of computational time and effort. However in the case of moderate switching time the benefit of the Base Stock heuristic may outweigh the cost of computational time and effort.

This study answered the research question of how Base Stock type policies perform. Overall a full Base Stock Heuristic is superior to Base Percentage Heuristics and Cooperative heuristics. However in scenarios with larger switching times the performance of the BP(2, 85) heuristic approaches that of the Base Stock Heuristic and requires less computation time.

SECTION 3.4: BASE SEPARATE HEURISTIC

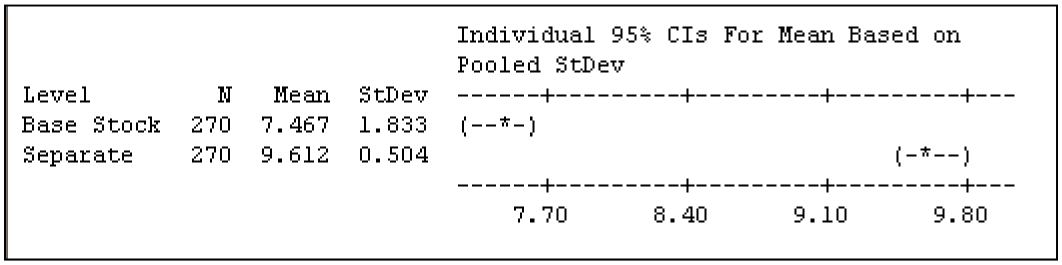
All the heuristics studied in previous sections kept both servers together at all times. Mayorga et al. (2009) showed in their work that when switching costs are included it is sometimes optimal for workers to separate. These findings and the performance of the Base Stock heuristic led us to ask the fifth research question in Section 2.2 which inquires about the performance of a Base Separate heuristic. The Base Separate heuristic is denoted by BSP(x, y, z) where both servers work at station x until it is y percent full at which time server z moves to the other station.

Recall the discussion of the parameter Service Ratio in Section 2.1. Service Ratio was not included in previous studies because both servers were always together, and due to the assumption of collaborative work the additive rate would be the same. In our study Server 2 is always the faster server. The initial study of the BSP heuristic is similar to the others performed, and the parameters ρ, Switching Ratio, B_1, and B_2 were varied in the ranges $\rho = \{0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00\}$, Switching Ratio = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00}, Service Ratio = {1.00, 2.00, 5.00, 10.00}, and B_1=B_2= {6, 12, 24}.

Much like the Base Stock Heuristic the BSP heuristic can result in an unwieldy number of heuristics to test if all possible combinations are tested. Therefore the BSP(1,E,1), BSP(1,E,2), BSP(1,50,1), BSP(1,50,2), BSP(1,85,1), BSP(1,85,2), BSP(2,E,1), BSP(2,E,2), BSP(2,50,1), BSP(2,50,2), BSP(2,85,1), and BSP(2,85,2) were calculated for every scenario because of those percentages' significance in previous tests.

The best heuristic in each scenario was then selected to calculate the Base Separate heuristic. The full results can be viewed in the supplemental file.

The BSP heuristic was compared to the Base Stock heuristic, and the Base Stock heuristic had a mean throughput of 7.46 parts/hour while the BSP heuristic had a mean of 9.612 parts/hour. Confidence intervals were constructed around the means for each heuristic with a 95% level of confidence, and can be seen in Figure 3.18.



```
                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level         N    Mean   StDev   ------+---------+---------+---------+---
Base Stock  270   7.467   1.833   (--*-)
Separate    270   9.612   0.504                                (-*--)
                                  ------+---------+---------+---------+---
                                      7.70      8.40      9.10      9.80
```

Figure 3.18: 95% confidence intervals for Base Stock and Base Separate Heuristics

Because there is no overlap in the confidence intervals and the BSP heuristic has a greater mean than the Base Stock Heuristic we can conclude that the BSP heuristic significantly outperforms the Base Stock heuristic overall. The next question to answer was how the BSP and Base Stock heuristics performed as switching time increases. This question is best answered by the chart seen in Figure 3.18. Figure 3.19 considers the BSP heuristic where the Service Ratio was 1.0 meaning both servers operated at the same speed.

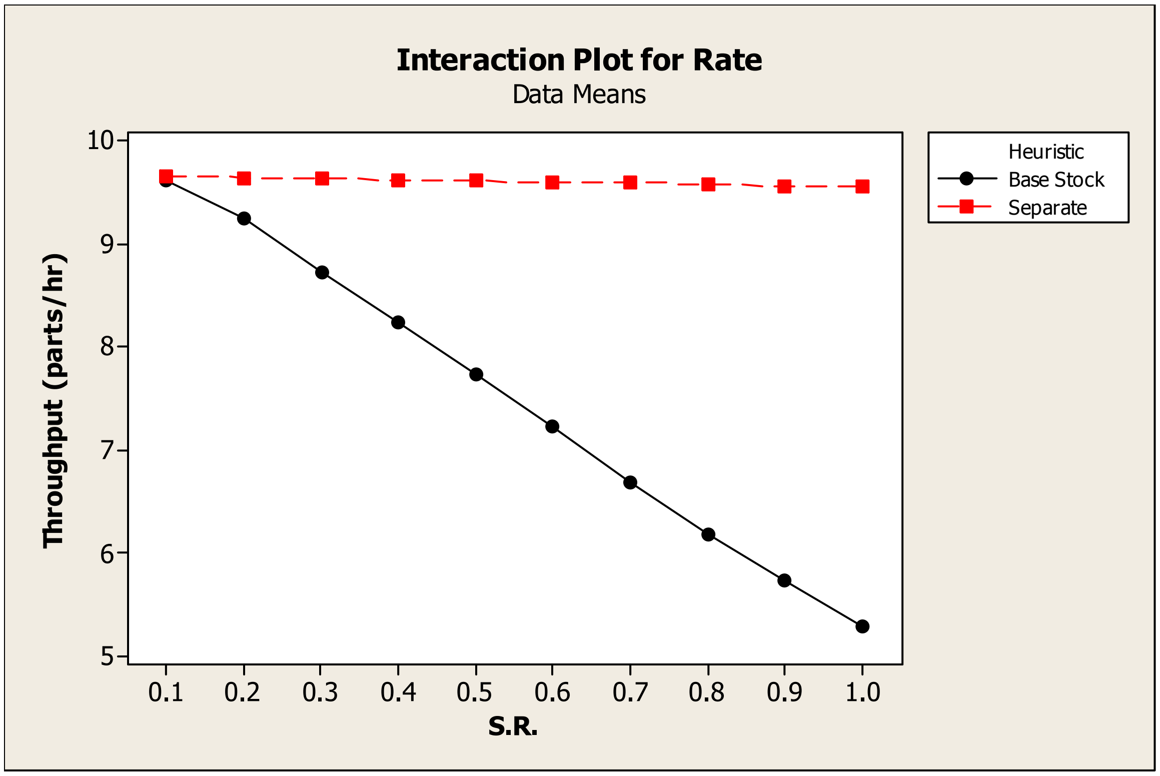


Figure 3.19: Comparing Base Stock and Base Separate heuristics with respect to Switching Ratio when service ratio =1

Figure 3.19 demonstrates that as switching time increases the benefit of using the BSP heuristic becomes greater. It is evident that the BSP heuristic is not nearly as sensitive to changes in switching time as the Base Stock heuristic. Because of this insensitivity the BSP heuristic should be used in scenarios where switching time is high. It should also be used in situations where switching time is highly variable. This mirrors the findings of Mayorga et al. (2009) who tested heuristics in which one server remained stationary. They found that as switching cost increased the advantage of using one stationary server increased. However the BSP heuristic is sensitive to the Service Ratio which is not a factor in for the Base Stock Heuristic. Therefore a test was run to determine the level of sensitivity to Service Ratio which is shown in Figure 3.20.

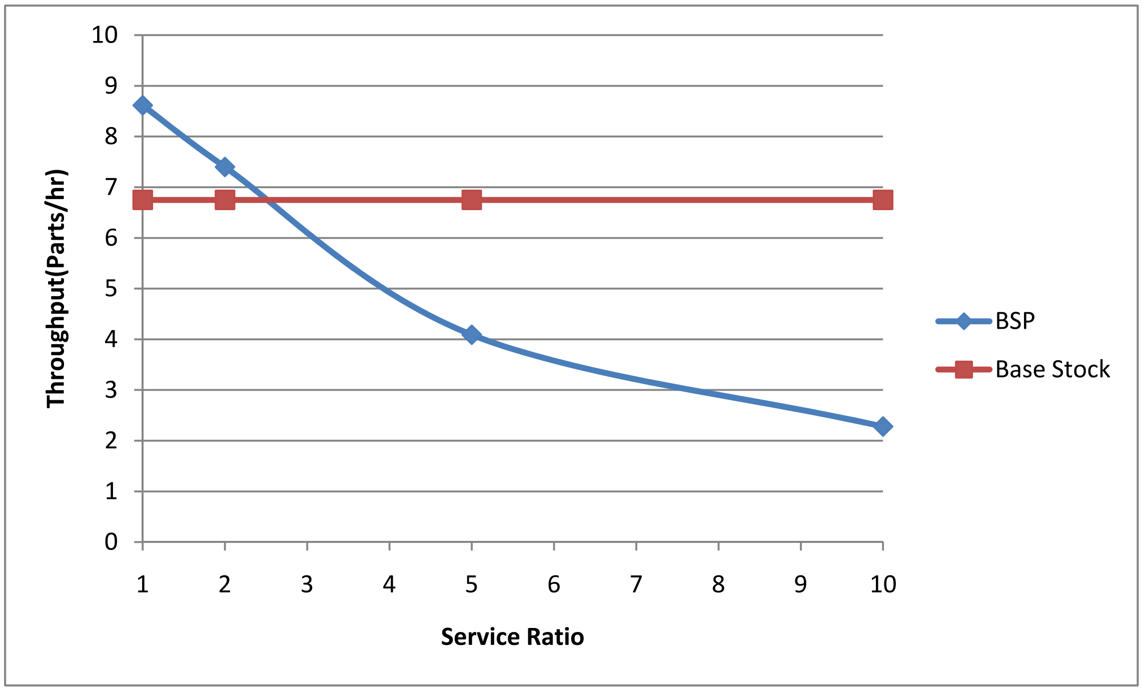


Figure 3.20: BSP heuristic sensitivity to Service Ratio

It can be seen in Figure 3.20 that the BSP heuristic is sensitive to changes in Service Ratio. The throughput ranges from nearly 10 parts per hour when the Service Ratio is 1 to just over 2 parts per hour when Service Ratio is 10. This result led to question of how the BSP heuristic compared to the Base Stock Heuristic when the Service Ratio was higher than 1. This comparison can be made it Figure 3.21.

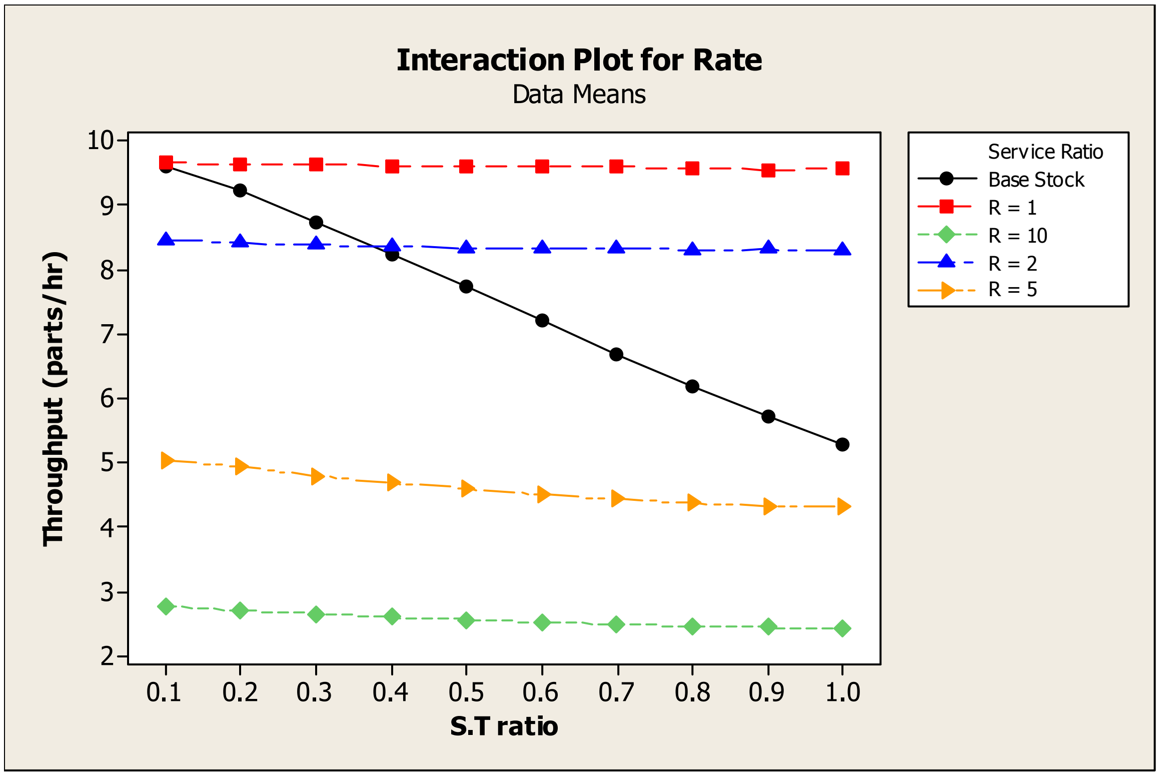


Figure 3.21 BSP Heuristic with different Service Ratios compared with Base Stock heuristic

Figure 3.21 shows that when Service ratio is closer to 1 the BSP policy significantly outperforms the Base Stock Policy. However as the Service Ratio increases the performance of the BSP heuristic degrades. The results of this comparison lead to mixed conclusions. While the BSP policy is much more stable with respect to switching time, it is much more sensitive to the changes in the service ratio. Therefore in situations where the Service Ratio is higher such as when one server is in training or new the Base Stock heuristic may be preferable to the BSP heuristic.

This study answers the question of how a Base Stock policy with one stationary server performs. The BSP heuristic is very stable with respect to switching time. Therefore in a system with servers operating at the same speed the BSP heuristic is preferable to the Base Stock Heuristic. However because the workers are allowed to separate the Service Ratio has a significant effect on the throughput of the system. Thus in an environment where the workers operate at different speeds the Base Stock Heuristic may perform better than a BSP heuristic. Therefore it may be advisable to use the Base Stock heuristic in situations such as training new workers who may not be able to work at the same speed as more experienced workers. The final observation regarding the BSP heuristic was that the BSP(2, E, 2) was the superior heuristic in nearly all scenarios when the Service Ratio was 1.0 or 2.0. However as the Service Ratio Increased to 5.0 and 10.0 the BSP(2, 85, 2) performed the best. This shows that, in general, it is superior to use a BSP heuristic which leaves both servers at Station 2 and switch the faster of the 2 servers. The Station 1 inventory level at which the server should switch however is dependent upon the Service Ratio.

SECTION 3.5: BUFFER ALLOCATION

After the most appropriate resource allocation policy is selected there are still other issues which confront manufacturers. Frequently manufacturing environments have space limitations when implementing manufacturing systems. Because of this it is of interest to determine how to allocate a finite amount of buffer space. This study answers the sixth research question in Section 2.2 regarding the allocation of a finite buffer. Additionally because of the costs associated with adding buffer space we have conducted a study to determine the benefit of adding additional buffer space.

This study made use of the knowledge gleaned from previous studies. It was desired to test the effect of Different Buffer Ratios (BR) using different heuristics. Recall from Section 2.1 that BR is the ratio of B_1 to B_2. A study was then constructed using scenario numbers 2, 6, 24, 70, and 90. These scenarios were selected because different heuristics performed best for each. For example scenarios 2 and 14 responded better to the Base Stock heuristics while scenarios 6 and 24 responded better to the MP1 heuristic. Refer to the supplemental file for details of each scenario. We tested BR levels of 0, 0.2, 0.5, 0.7, 1.5, 2.0, 5.0, and infinity (all buffer space allocated to B_1). Additionally total buffer sizes of 12, 24, and 48 were tested with the aforementioned BR levels. The results of this study are summarized in Figure 3.22.

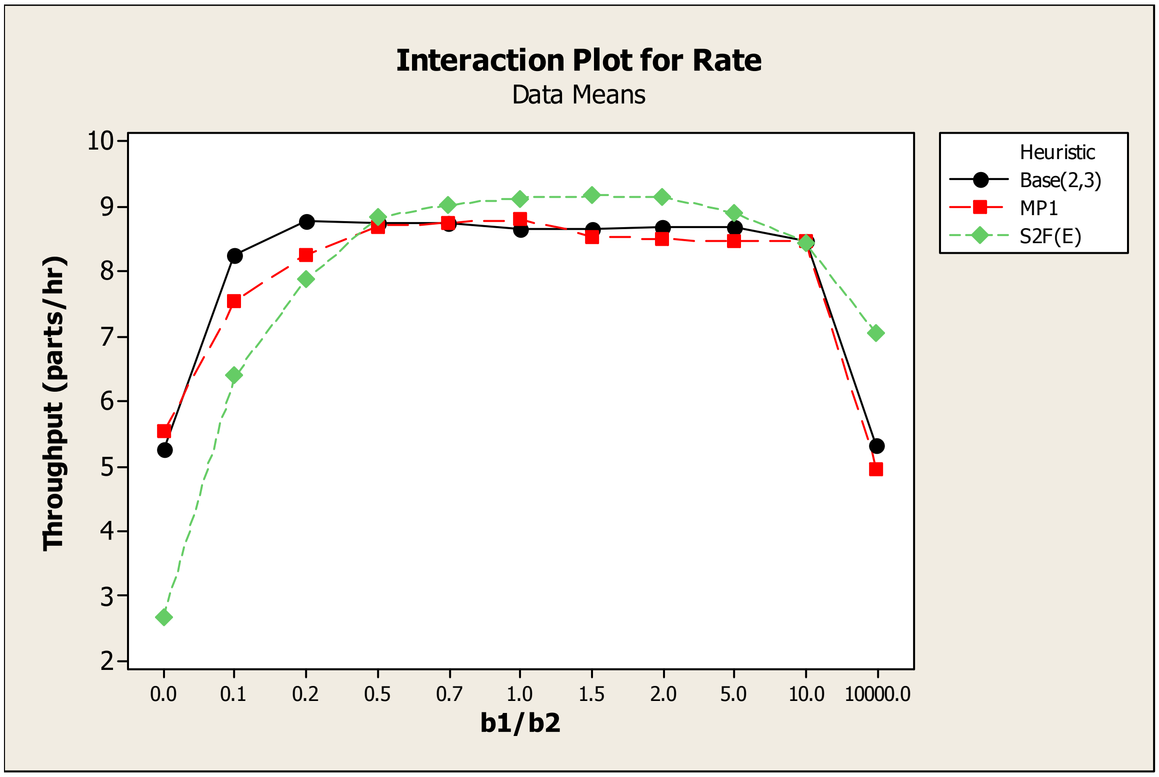
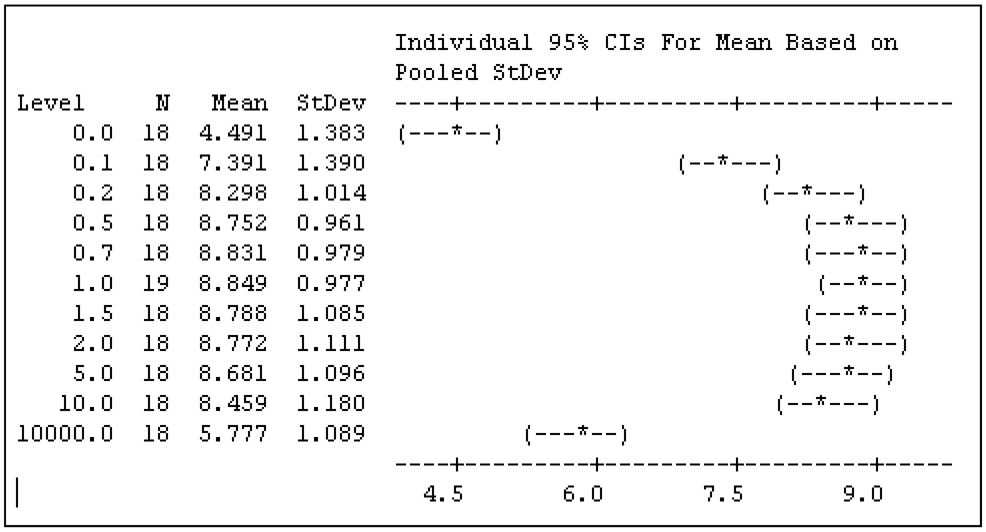


Figure 3.22: Comparison of Different Buffer Ratios with different Heuristics

Figure 3.22 shows that for most values of BR there is not a large difference in the throughput of the system. It should be noted that a value of 10,000 was used in the Figure 3.22 to denote infinity. There is a large difference when all the buffer space is allocated to one buffer or the other indicated by the precipitous drops in throughput at BR levels of zero and infinity. However this effect is relegated to BR values approaching zero and infinity. These conclusions were confirmed using 95% confidence intervals constructed around the tested BR levels which can be seen in Figure 3.23.



```
                                    Individual 95% CIs For Mean Based on
                                    Pooled StDev
Level      N    Mean   StDev   ----+---------+---------+---------+-----
    0.0   18   4.491   1.383   (---*--)
    0.1   18   7.391   1.390                       (--*---)
    0.2   18   8.298   1.014                             (--*---)
    0.5   18   8.752   0.961                                (--*---)
    0.7   18   8.831   0.979                                (---*--)
    1.0   19   8.849   0.977                                 (--*--)
    1.5   18   8.788   1.085                                (---*--)
    2.0   18   8.772   1.111                                (--*---)
    5.0   18   8.681   1.096                               (---*--)
   10.0   18   8.459   1.180                              (--*---)
10000.0   18   5.777   1.089            (---*--)
                               ----+---------+---------+---------+-----
                                  4.5       6.0       7.5       9.0
```

Figure 3.23: 95% confidence intervals for BR levels

Figure 3.23 shows that only BR levels of 0.0 and infinity are significantly different from the other levels tested. This leads to the conclusion that buffer strategies which place all available buffer space at one station perform poorly, but strategies which allocate at least some buffer space to both stations perform equally well.

With the knowledge that the buffer allocation was very robust we wanted to explore the benefits of adding buffer space when BR = 1. Thus a final experiment was performed to determine the effect of adding buffer space. The parameters ρ, Switching Ratio, Service Ratio, B_1, and B_2 were varied in the ranges ρ = {0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00}, Switching Ratio = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00}, and Service Ratio = {1.0, 2.0, 5.0, 10.0}, and B_1=B_2= {1, 2 … 30}. The BSP heuristic was used because it performed the best in previous studies. Figure 3.24 summarizes the results of this study.

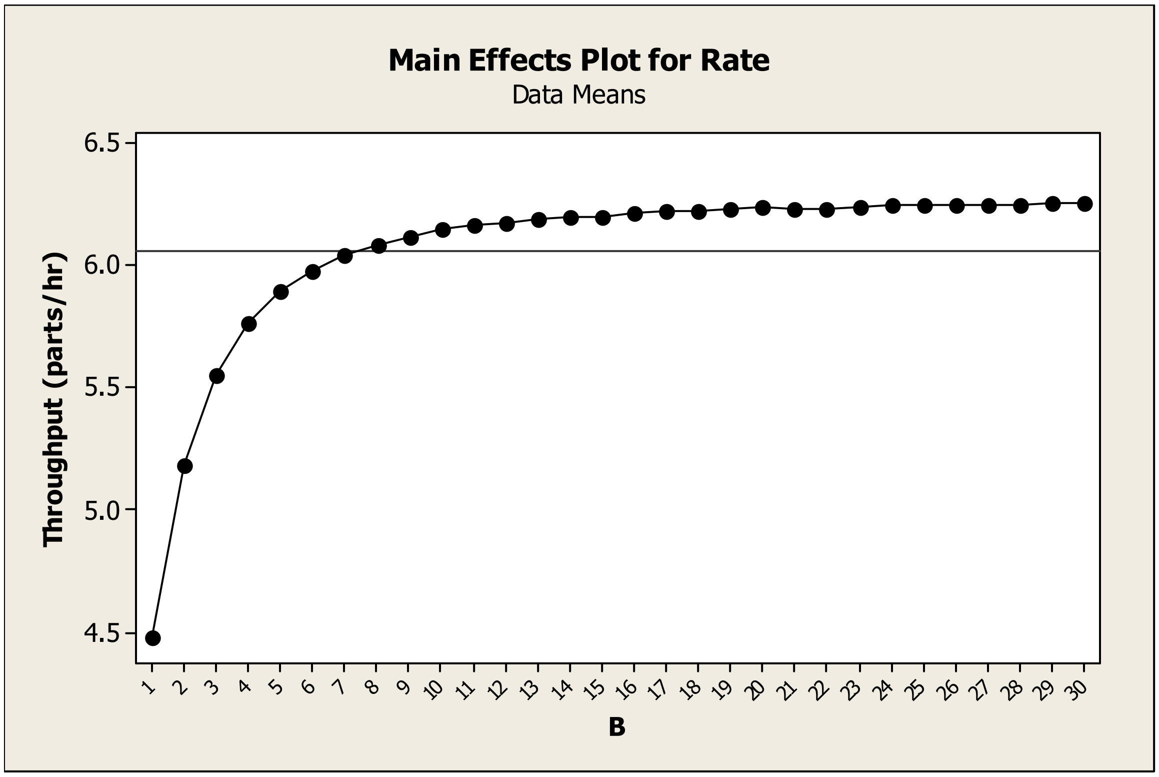


Figure 3.24: Effect of buffer size on throughput

Figure 3.24 shows that the value of adding buffer space diminishes as more buffer space is added. This tells managers that adding buffer space may be the best investment for the process up to a certain point, however eventually throughput is no longer significantly affected by adding additional buffer space. The use of the BSP heuristic added the element of Service Ratio to the testing. Therefore it was of interest to see what the effect of the Service Ratio was on the throughput. The results can be seen in Figure 3.25.

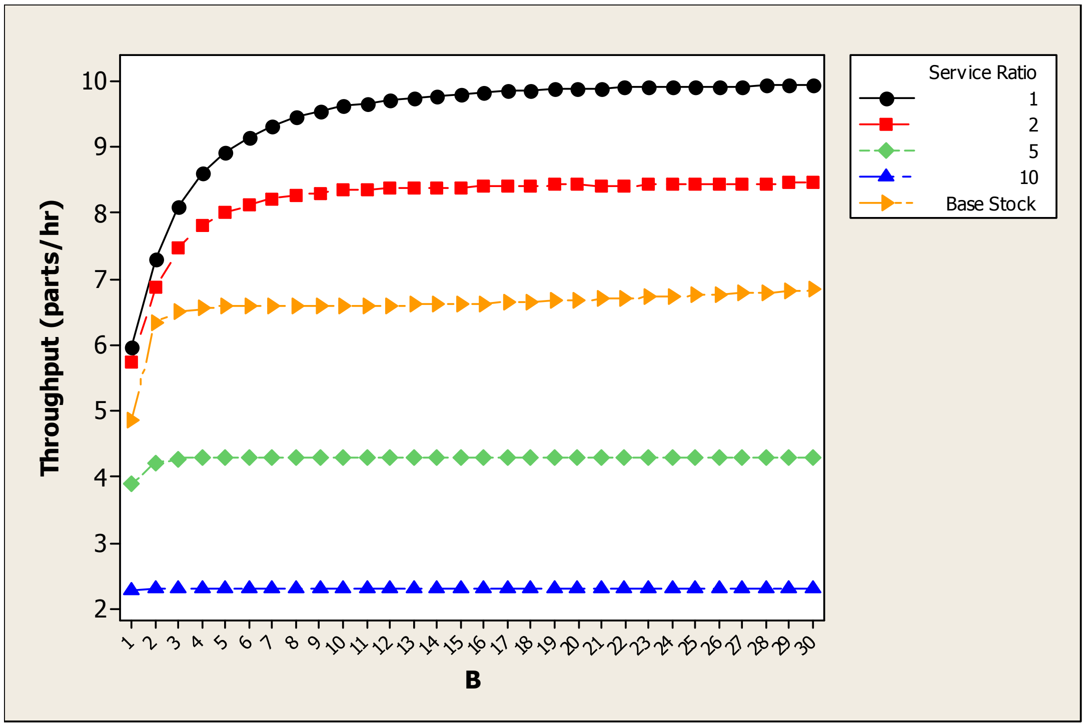


Figure 3.25: Comparison of buffer sizes with varying Service Ratios

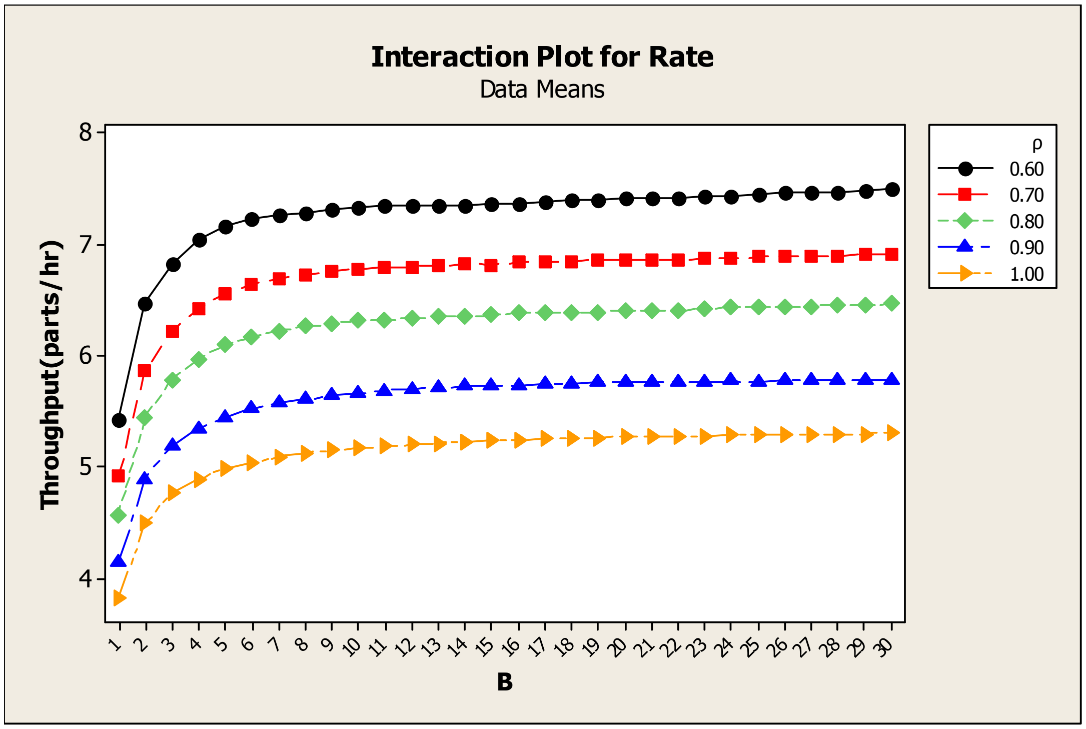


Figure 3.26: Interaction Plot for ρ and Buffer Size

Figure 3.25 shows that the effect of different Service Ratios on the throughput. It can be seen that the effect of buffer size is greatest when both servers operate at the same speed and therefore have a Service Ratio of 1.0. The effect is lessened significantly as the Service Ratio is increased. Therefore it makes more sense to invest money to decrease the difference in service speeds rather than adding buffer space. Additionally it shows that an imbalance in service speed cannot be offset by increasing buffer size. Figure 3.26 shows the diminishing rate of returns for several values of ρ. This shows that the effect of increasing buffer size is similar across several values of ρ.

This section answered the research question of how to allocate buffer sizes given a finite number of buffer spaces. It was shown that throughput drops dramatically as BR approaches zero and infinity, and is very stable for BR levels between. We have also shown that there is a steep diminishing rate of returns for adding buffer space to the system. Therefore adding buffer space is only beneficial if the current environment has very limited buffer space. Additionally it was determined that adding buffer size is ineffective when a large imbalance of server speeds is present. This also means that an imbalance of server speeds cannot be counteracted by simply adding buffer space.

Section 3.6: Conclusion

In this manuscript we have modeled a tandem production system with collaborative flexible servers with a goal of maximizing throughput when switching time is positive. We have focused on developing heuristics which can be easily implemented in manufacturing environments.

Preliminary testing was conducted to determine the effect of the pre-emptive resume assumption and to test the efficacy of the OptQuest heuristic. The pre-emptive resume assumption was determined not to have a large effect on the throughput of the system when there was no time associated with switching. After the preliminary testing it was decided that the OptQuest heuristic would no longer be used because it requires extensive computation time. In addition to the large computational time requirement the results from OptQuest were not significantly better than other heuristics which were tested. The final reason OptQuest was no longer used was because the resource allocation policies which it devised were not easily implementable in a manufacturing environment.

A large scale test which took switching time into consideration was then constructed to test several cooperative heuristics which always kept both workers together. The results of this test showed that switching time greatly impacts the achievable throughput of the system. Additionally it was shown that the MP1 (More Parts 1) and MP2 heuristics, which moves both servers to the station with a higher inventory level and break ties by moving both servers to Stations 1 & 2 respectively, performed significantly better than the No Blocking heuristic, which moves servers only to avoid blocking, and the Expedite heuristic which moves servers to follow a part through the system. The MP1 and MP2 were superior because they allow both buffers to be used a moderate amount. The No Blocking and Expedite policies result in a large build up in the Station 1 buffer causing incoming jobs to balk from the system. The final conclusion was that the heuristic chosen had the greatest effect when switching times were moderate. This is because when switching time is small there is enough time to recover from a poor allocation policy, and if switching time is high even an optimal allocation policy will not allow you to recover all the time spend switching.

After developing Cooperative heuristics the Base Stock heuristic, which assigns the worker to stay at one station until a base stock level is reached at the other station, was developed because of its ease of implementation. However the Base Stock heuristic requires a much larger computational effort to determine than the cooperative heuristics. This is compounded by the fact that the number of permutations necessary to compute the Base Stock heuristic changes when the buffer sizes are altered. It was found that the Base Stock heuristic was superior to the other cooperative heuristics overall. However the Base Stock heuristic requires a much larger time to determine than the other cooperative heuristics, and the Base Stock Policy differs when the buffer sizes are changed. To combat this we developed the Base Percentage (BP) heuristic was developed. In the BP heuristic both servers stay at a station until the other station is a certain percentage full at which time they both switch. Using percentages allows the heuristic to be valid for all buffer sizes. The Base Percentage heuristic does not perform as well overall as the full Base Stock heuristic, but it can require significantly less computation time. Additionally it was shown that the BP(2,85) heuristic which approaches the performance of the Base Stock Heuristic as switching time increases.

We also tested the effectiveness of a Base Separate Policy (BSP) heuristic. In a BSP heuristic the workers are both assigned to a "Base" station until a certain stock level is reached at the other station at which time one of the servers switches to the non-"Base" station. It was shown that the BSP heuristic was very stable with respect to switching time. Therefore when the workers are operating at the same speed the BSP heuristic is superior the Cooperative heuristics and Base Stock heuristic. However because the workers were allowed to separate the Service Ratio became an important factor. When the BSP heuristic is used, the throughput of the system significantly decreases as the Service Ratio increases. Additionally it was concluded that the BSP heuristic should be used with both workers based at Station 2 and switching the faster server at an inventory level which is dependent on the Service Ratio.

Finally the allocation of buffer space was explored. It was determined the throughput of a system was very stable with respect the Buffer Ratio (B_1/B_2). Throughput of the system only dropped significantly when the all buffers were allocated solely to one station. Additionally, it was shown that adding buffer space for a system where $B_1=B_2$ had a diminishing rate of return. When buffer sizes were very small adding additional buffer space made a large difference, but as the buffer size grew adding more buffer space made much less of an impact on the throughput of the system.

This paper adds to previous work because it takes actual switching time into account, where previous literature has dealt with switching costs. Additionally it tests the effectiveness of allocation resource policies which can be readily implemented in manufacturing environments. It was also shown that simulation optimization strategies may not be the best approach to this problem because of the large computational requirements and lack of implementable solutions.

Section 3.7: Future Research

This research has led to other questions which should be answered with future research.

1. *How does the system perform when variability is added to the switching time?* Up to this point switching time has been modeled as a deterministic value. How does it affect the throughput of the system if we assign a distribution to switching time?
2. *Can the conclusions reached by this work be extended to systems with more stations?* We have only modeled a system with two serial stations. It would be of interest to model systems with three or more stations.
3. *Can a large system of stations be improved by grouping workers and stations and optimizing these smaller cells?* Optimizing a large number of stations can be a very complicated task. Can we approximate a global optimum by optimizing smaller groups of cells within a larger system?
4. *How do optimal policies differ in systems with large warm up effects?* In larger systems there may be more significant warm-up effects present when the system begins. How do optimal policies in these situations differ from polices with no warm up effect?
5. *What happens when service rates of the workers depend on the stations?* For example, workers may be primarily trained at one station so that they work faster at that station than at another.

APPENDICES

Appendix A

Preliminary Design of Experiments

Experiment	ρ	Service Ratio	Buffer Ratio	λ	u1	u2	B1	B2
1	0.5	0.5	0.5	3.8	5	10	3	6
2	0.5	0.5	1	3.8	5	10	6	6
3	0.5	0.5	1.5	3.8	5	10	6	3
4	0.5	0.75	0.5	4.4	7.5	10	3	6
5	0.5	0.75	1	4.4	7.5	10	6	6
6	0.5	0.75	1.5	4.4	7.5	10	6	3
7	0.5	1	0.5	5.0	10	10	3	6
8	0.5	1	1	5.0	10	10	6	6
9	0.5	1	1.5	5.0	10	10	6	3
10	0.75	0.5	0.5	5.6	5	10	3	6
11	0.75	0.5	1	5.6	5	10	6	6
12	0.75	0.5	1.5	5.6	5	10	6	3
13	0.75	0.75	0.5	6.6	7.5	10	3	6
14	0.75	0.75	1	6.6	7.5	10	6	6
15	0.75	0.75	1.5	6.6	7.5	10	6	3
16	0.75	1	0.5	7.5	10	10	3	6
17	0.75	1	1	7.5	10	10	6	6
18	0.75	1	1.5	7.5	10	10	6	3
19	1	0.5	0.5	7.5	5	10	3	6
20	1	0.5	1	7.5	5	10	6	6
21	1	0.5	1.5	7.5	5	10	6	3
22	1	0.75	0.5	8.8	7.5	10	3	6
23	1	0.75	1	8.8	7.5	10	6	6
24	1	0.75	1.5	8.8	7.5	10	6	3
25	1	1	0.5	10.0	10	10	3	6
26	1	1	1	10.0	10	10	6	6
27	1	1	1.5	10.0	10	10	6	3

Figure A-1: Parameters used for Preliminary DOE

Appendix A

Preliminary Design of Experiments

Experiment	Arumugam	MA	Optquest	More Part	No Block	Expedite
1	3.78	3.78	3.79	3.78	3.78	3.74
2	3.79	3.79	3.87	3.81	3.82	3.79
3	3.79	3.80	3.82	3.82	3.80	3.82
4	4.38	4.37	4.38	4.36	4.37	4.32
5	4.39	4.41	4.50	4.40	4.43	4.36
6	4.39	4.40	4.47	4.39	4.39	4.39
7	4.98	5.02	4.98	4.94	5.02	4.93
8	5.00	4.99	5.10	5.01	5.07	4.99
9	5.00	4.99	5.09	5.01	5.01	5.01
10	5.48	5.27	5.28	5.32	5.27	5.19
11	5.58	5.56	5.62	5.56	5.53	5.52
12	5.55	5.56	5.57	5.53	5.61	5.53
13	6.43	6.28	5.99	6.23	6.28	6.12
14	6.55	6.56	6.56	6.61	6.48	6.50
15	6.52	6.56	6.53	6.56	6.54	6.56
16	7.33	7.11	5.71	7.09	7.11	6.98
17	7.46	7.42	6.47	7.40	7.47	7.40
18	7.43	7.41	7.53	7.42	7.43	7.42
19	6.77	6.41	6.08	6.46	6.41	6.24
20	6.98	6.94	6.93	6.94	6.88	6.76
21	6.90	6.88	6.89	6.87	6.84	6.87
22	7.90	7.49	7.43	7.52	7.49	7.31
23	8.16	8.07	8.10	8.08	8.00	7.91
24	8.06	8.02	8.02	8.03	7.98	8.03
25	9.02	8.51	8.50	8.58	8.51	8.32
26	9.31	9.20	9.21	9.24	9.13	8.99
27	9.20	9.13	9.18	9.13	9.10	9.13

Figure A-2: Preliminary DOE Results in Parts/hr

Appendix B

Description of Experiments in Supplemental Excel File

3.2-Cooperative Heuristics

In this set of scenarios λ is held constant while μ_1 and μ_2 are decreased to change ρ. The variables are contained in the range: $\rho = \{0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00\}$, Switching Ratio = $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00\}$, and $B_1=B_2= \{6, 12, 24\}$. The heuristics MP1, MP2, No Blocking and Expedite are tested over all possible combinations of these variables. Please refer to Section 1.3 for more information about the heuristics.

Sec 3.3-Base Stock

In this set of scenarios λ is held constant while μ_1 and μ_2 are decreased to change ρ. B_1 and B_2 are both the same size for all tests run. All rates are in parts per hour. The variables are contained in the ranges: $\rho = \{0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00\}$, Switching Ratio = $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00\}$, and $B_1=B_2= \{6, 12, 24\}$. The results from the Base Stock heuristic are listed here. This worksheet omits the results from all Base(x,y) combinations and simply reports the Base Stock heuristic.

Sec 3.4-BSP Heuristic

In this set of scenarios λ is held constant while the sum of μ_1 and μ_2 are decreased to change ρ. In this set of experiments μ_1 and μ_2 are also changed in relation to each other to reflect changes in the Service Ratio. The variables were varied in the ranges: $\rho = \{0.6,$

0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.00}, Switching Ratio = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.00}, Service Ratio = {1.00, 2.00, 5.00, 10.00}, and B_1=B_2= {6, 12, 24}. This worksheet omits all BSP(x, y, z) combinations and reports only the BSP results.

<u>Sec 3.5-Buffer Allocation</u>

This worksheet uses experiments 2, 6, 24, 70, and 90 from Section 3.2. BR levels of 0, 0.2, 0.5, 0.7, 1.5, 2.0, 5.0, and infinity (all buffer space allocated to B_1) are tested. In addition total buffer sizes of 12, 24, and 48 are used.

<u>Sec 3.5-Buffer Sizing</u>

This worksheet uses the same set of scenarios Contained in Sec 3.4-BSP Heuristic. Additional tests are run with the Base Stock Scenarios from Sec 3.3 Base Stock. This is because it is unnecessary to use different Service Ratios with the Base Stock Heuristic. Refer to Section 3.4 for more information. Buffer sized of B_1=B_2=B are used with values of B ranging from 1 to 30.

References

1. Andradottir, S., Ayhan, H., and Down, D., 2003, “Dynamic Server Allocation for Queueing Networks with Flexible Servers,” Informs, 51(6), 952-968.
2. Lee, Z., Lee, C., 2005, “A Hybrid Search Algorithm with Heuristics for Resource Allocation Problem,” Information Sciences, 173, 155-167
3. Ahn, H., Duenyas, I., Zhang, R., 2004, “Optimal Control of a Flexible Server,” Advances Applied Probability, 36, 139-170.
4. Palmer, J., Mitrani, I., 2005, “Optimal and Heuristic Policies for Dynamic Server Allocation,” Journal of Parallel and Distributed Computing, 2005, 65, 1204-1211
5. Batta, R., Berman, O., Wang, Q., 2007, “Balancing Staffing and Switching Costs in a Service Center with Flexible Servers,” European Journal of Operations Research, 177, 924-938.
6. Rosberg, Z., Varaiya, P., Walrand, J., 1982, “Optimal Control of Service in Tandem Queues,” IEEE Transactions on Automatic Control, AC-27, 600-610
7. Farrar, T.M., 1993, “Optimal Use of an Extra Server in a Two Station Tandem Queuing Network,” IEEE Transactions on Automatic Control, 38(8), 1296-1299.
8. Ahn, H., Duenyas, I., Zhang, R., 1999, “Optimal Stochastic Scheduling of a Two-Stage Tandem Queue with Parallel Servers ,” Advances Applied Probability, 31(4), 1095-1117.
9. Schiefermayr, K., Weichbold, J., 2005, “A Complete Solution for the Optimal Stochastic Scheduling of a Two-Stage Tandem Queue,” Journal of Applied Probability, 42(3), 778-796
10. Arumugam, R., Mayorga, M., Taaffe, K., 2009, “Inventory Based Allocation Policies for Flexible Servers in Serial Systems,” Ann Oper. Res., 172, 1-23
11. Mayorga, M., Taaffe, K., Arumugam, R.. 2009, “Allocating Flexible Servers in Serial Systems with Switching Costs,” Ann Oper. Res., 231-242
12. Van Oyen, M., Gel, E., Hopp, W., 2001, “Performance Opportunity for Workforce Agility in Collaborative and Non-Collaborative Work Systems,” IIE Transactions, 33, 761-777.
13. Andradóttir, S., Ayhan, H., Down, D., 2001, “Server Assignment Policies for Maximizing the Steady-State Throughput of Finite Queueing,” Management Science, 44(10), 1421-1439.
14. Andradóttir, S., Ayhan, H., 2005, “Throughput Maximization for Tandem Lines with Two Stations and Flexible Servers,” Operations Research, 53(3), 516-531.
15. Tsai, Y., Argon, T., 2008, “Dynamic Server Assignment Policies for Assembly-type Queues with Flexible Servers,” Wiley Interscience (in press).
16. Thesen, A., Grant, H., Kelton, W.D., 1987, “Simulation, Animation, and Shop-Floor Control, Proc. of the Winter Simulation Conference, Dec 14-16, Atlanta, Georgia, 649-653.
17. Drake, G., Smith, J., 1996, “Simulation System for Real-Time Planning, Scheduling, and Control,” Proc. of the Winter Simulation Conference, December 8-11, Cornado, California, 1083-1090.
18. Bischak, D., 1996, “Performance of a Manufacturing Module with Moving Workers, “ IIE Transactions, 28, 723-733.
19. Chun, H., Wai Tak Mak, R., 1999, “Intelligent Resource Simulation for an Airport Check-In Counter Allocation System,” Ieee Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 29(3), 325-335.
20. Glover, F., 1999, “New Advances for Wedding Optimization and Simulation,” Proc. of the Winter Simulation Conference, Dec 5-8, Phoenix, Arizona, 255-260.
21. Kleijnen, J., Wan, J., 2007, “Optimization of Simulated Systems: OptQuest and Alternatives,” Simulation Modeling Practice and Theory, 15, 354-362.
22. Rogers, P., 2002, “Optimum Seeking Simulation in the Design and Control of Manufacturing Systems: Experience with OptQuest for Arena,” Proc. of the Winter Simulation Conference, December 8-11. San Diego, California, 1142-1150.