

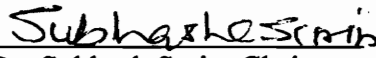
SEQUENCING POLICY FOR A CONWIP PRODUCTION SYSTEM

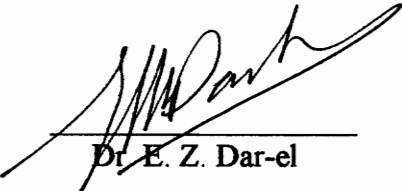
by


Michael P. Greco

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree
of
Master of Science
in
Industrial and Systems Engineering

APPROVED:


Dr. Subhash Sarin, Chairman


Dr. E. Z. Dar-el


Dr. George Ioannou


Dr. John Kobza

March, 1996

Blacksburg, Virginia

Keywords: CONWIP, Sequencing, Production, Queueing, Throughput

c. 2

LD
5655
V855
1996
G 743
c. 2

SEQUENCING POLICY FOR A CONWIP PRODUCTION SYSTEM

by

Michael P. Greco

Dr. Subhash Sarin, Chairman

Industrial & Systems Engineering

(ABSTRACT)

The optimization of the performance of a constant Work-in-Progress (CONWIP) production system through the sequencing of its backlog list is investigated. The performance measures considered are throughput, optimum WIP level (m^*), and flow time. Analysis of the effects of sequence dependent bottlenecks on system performance is provided. A procedure is presented to determine a lower bound for (m^*) given the product mix. A method that determines (m^*) given the sequence of jobs is provided.

A heuristic algorithm is provided for the purpose of determining a sequence to minimize (m^*). The algorithm attempts to sequence the jobs to achieve the “best fit” between consecutive jobs so that machine and job idle times are minimized. The algorithm is tested through computer implementation to reveal its proficiency.

ACKNOWLEDGEMENTS

The author would like to offer special thanks to his advisor, Dr. Subash Sarin, for his technical assistance, encouragement, and many helpful suggestions. Sincere appreciation is also extended to the other members of his graduate committee, Dr. E. Z. Dar-el, Dr. George Ioannou, and Dr. John Kobza for their interest and suggestions.

The author wishes to express his gratitude to his parents for their continual support, guidance, and for giving him the opportunity to pursue higher education. The author would also like to express his gratitude to Christine for her encouragement and support during this endeavor.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iii
LIST OF TABLES	iv
LIST OF FIGURES	ix
<u>Chapter</u>	<u>Page</u>
1.0 INTRODUCTION	1
1.1 Introduction to CONWIP Production System	1
1.2 Statement of Problem	8
1.3 Research Objective	10
1.4 Outline of Research	11
2.0 LITERARY REVIEW	12
2.1 CONWIP	13
2.2 Closed Queuing Networks	17
2.3 Flow Shop	20
2.4 Kanban	24
3.0 CONCEPTS AND CHARACTERISTICS OF CONWIP	29
3.1 One Product Case	29
3.2 Introduction to Multi-product Case	31
3.3 Transitions	34
3.4 Effects of Sequencing Policy	47
3.4.1 Two Product Equal Demand Case	49
3.4.2 Multi-Product Mix	53
3.5 Flow Times	56
3.6 m^* in the Multi-product Case	58
3.7 Effects of Grouping	62

4.0	LOWER BOUND DETERMINATION OF m^*	65
4.1	Lower Bound for m^*	65
4.3	Determining m^* in Multi-Product Case	87
5.0	HEURISTIC PROCEDURE TO SEQUENCE JOBS ON THE BACKLOG LIST	96
5.1	Introduction to the Proposed Heuristic Procedure	96
5.2	Results from Experimentation with Heuristic	104
6.0	CONCLUSIONS AND RECOMMENDATIONS	118
6.1	Summary and Conclusions	118
6.2	Recommendations	124
	List of References	126

LIST OF TABLES

<u>Table</u>	<u>Page</u>
(2.1-1) Summary of CONWIP Literature	13
(3.3-1) Transition Effects for Example (3.3-1)	37
(3.3-2) Transition Effects for Example (3.3-2)	38
(3.3-3) Transition Effects for Example (3.3-3)	39
(3.3-4) Transition Effects for Example (3.3-4)	40
(3.3-5) Transition Effects for Example (3.3-5)	41
(3.3-6) Non-Bottleneck Effects	43
(3.3-7) Transition Effects for Example (3.3.7)	45
(3.3-8) Transition Effects for Example (3.3.8)	45
(3.4.1-1) Throughput Values for Two Product Example	51
(4.1.1) Processing Times $t(i,j)$ for the Lower Bound Example	67
(4.1.2) Sum of Processing Times for Leaving and Entering Jobs	68
(4.1.3) Illustration of Lower Bound Procedure I, Step (1.1): determine the $tr(i,i')$ matrix values, which are the sum of processing time of job (i) after the bottleneck and the processing time of job (i') before the bottleneck.....	76
(4.1.4) Illustration of the Lower Bound Procedure I, Step (1.2): eliminate $tr(i, i')$ in descending order until only one element remains in any row or column and select that element designated as $N(1)$ at $tr(6,3)$	77
(4.1.5) Illustration of the Lower Bound Procedure I, Step (1.3): eliminate any remaining elements in the selected element's row or column.....	77
(4.1.7) Illustration of the Lower Bound Procedure I, Step (1.5): eliminate $tr(i, i')$ in descending order, select elements when necessary, eliminate any remaining elements in the selected element's row or column.....	77

<u>Table</u>	<u>Page</u>
(4.1.6)	Illustration of the Lower Bound Procedure I, Step (1.4): eliminate $tr(i, i')$ in descending order until a second element is selected $N(2)$ 78
(4.1.8)	Illustration of the Lower Bound Procedure II, Step (2.1): define the matrix $tr'(i, i') = \max [tr(i, i'), tr(i', i)]$ 80
(4.1.9)	Illustration of the Lower Bound Procedure II, Step (2.2): eliminate the pairs of $[tr'(i, i'), tr'(i', i)]$ in descending order 81
(4.1.10)	Illustration of the Lower Bound Procedure II, Step (2.2): determine that a feasible set no longer exists 81
(4.1.11)	Illustration of the Lower Bound Procedure II, Step (2.3): re-include the last pair that was eliminated and determine the set of $tr'(i, i')$ 82
(4.1.12)	Illustration of the Lower Bound Procedure III, Step (3.1) 83
(4.1.13)	Standard 4 x 4 $tr(i, i')$ Matrix 86
(4.2-1)	Example of Optimum One Product Values of $T(i,j)$ 88
(4.2-2)	Example of Non-Optimum One Product Values of $T(i,j)$ 89
(4.2-3)	Example of Non-Optimum Two Product $T(i,j)$ Values 92
(4.2-4)	Example of Optimum Two Product $T(i,j)$ Values 92
(4.2-5)	Generic (k) Product Table at Optimality 95
(5.1-1)	Processing times for example 5.1.1 99
(5.1-2)	Cost Matrix C_{ij} : The Sum of Absolute Residuals for the Predecessor and Successor Pair (i,j) 99
(5.1-3)	Row Differences Between the Second and First Minimums in the Row of the C_{ij} Matrix 100
(5.1-4)	Column Differences Between the Second and First Minimums in the Column of the C_{ij} Matrix 101
(5.1-5)	Updated Cost Matrix C_{ij} with the Selection of $C_{5,4}$ and Elimination of other Appropriate C_{ij} 101

<u>Table</u>	<u>Page</u>
(5.1-6) Row Differences Between the Second and First Minimums in the Row of the Updated C_{ij} Matrix	102
(5.1-7) Column Differences Between the Second and First Minimums in the Column of the Updated C_{ij} Matrix	102
(5.1-8) Updated Cost Matrix C_{ij} with Selection of $C_{2,3}$ and the elimination of the other appropriate C_{ij}	103
(5.1-9) Final Cost Matrix for example 5.1.1	103
(5.2-1a,b,c) Average Throughput and Optimal WIP levels for Low, Random, and High Cost Sequences over 30 Observations	105
(5.2-2a,b,c) Average Throughput and Optimal WIP levels for Low, Random, and High Cost Sequences over 30 Observations	106
(5.2-3a,b,c) Average Throughput and Optimal WIP levels for Low, Random, and High Cost Sequences over 30 Observations	107
(5.2-4a,b,c) Average Throughput and Optimal WIP levels for Low, Random, and High Cost Sequences over 30 Observations	108
(5.2-5a,b,c) Average Throughput and Optimal WIP levels for Low, Random, and High Cost Sequences over 30 Observations	109
(5.2-6a,b,c) Throughput vs. WIP Levels	114

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
3.3-1	Illustration of Bottleneck Movement during Transitions	34
4.1-1	Illustration of Entering and Leaving Job Pairs	66
5.2-1	Throughput vs. WIP Curves for High Variance	116
5.2-2	Throughput vs. WIP Curves for Medium Variance	116
5.2-3	Throughput vs. WIP Curves for Low Variance	117
5.2-4	Throughput vs. WIP Curves for Random Sequences	117
	with High, Medium and Low Variance	

1.0 INTRODUCTION

1.1 Introduction to CONWIP Production System

The origin of the idea of a constant work in process (CONWIP) production system stems from the desire to obtain the potential benefits of a pull production system while avoiding some of the limitations of the most prevalent pull system known as Kanban. Industrial experience and research studies indicate that Kanban is difficult to implement when certain conditions are encountered in the manufacturing environment. These conditions include the existence in the production facility of a large number of different parts, bottlenecks, variable processing times and long setup times. When these conditions are present, Kanban will possess excessive and costly WIP and it will not have the necessary smooth flow of jobs through the system for it to function properly.

The CONWIP system was originally presented as a single serial production line, (Spearman et al. (1990)), in which parts are pushed between serial workstations in standard containers. However, it was not really a "push" system by definition since it did not schedule the start times of the jobs. Cards which are not part number specific are attached to the containers at the beginning of the line. When the container is used at the end of the line, the card is sent back to the beginning of the line. When the first station is idle and there is a card present, the next job in the backlog list is assigned a card and the work is initiated at the first station. The part number and the system entry time are assigned to the card.

CONWIP, as originally presented, is a flow shop with a most notable additional parameter of card count. Flow shops have been thoroughly investigated by researchers in the sequencing and scheduling area. They are of high interest due to their utilization in a wide variety of manufacturing environments. A typical problem investigated in flow shop research considers the criterion of minimizing the makespan, which is the completion time of the last job. Since the scheduling of these problems is combinatorial in nature, they are not suited well for enumerative schemes. Heuristic techniques have resulted in providing good solutions to the makespan problem but they do not guarantee optimality.

Due to the continuous flow of products that is anticipated for CONWIP production, the makespan criterion is not of primary interest. The parameters to optimize a CONWIP line consist of throughput, card count, and mean flow time in its steady state. Results from flow shop analysis are predominately reported for the static case. Thus, they frequently address factors that are of less concern to CONWIP in its steady state such as start-up machine idle time and shut-down machine idle time. However, the desire to sequence CONWIP's backlog list closely parallels scheduling of flow shops.

The maintenance of the backlog list in CONWIP is accomplished by production control. To expedite, the backlog list can be reordered. However, under no circumstances can work be started if a card is not present. The queue discipline is "first come first served" and thus no passing between jobs is possible on the line.

The backlog list that CONWIP provides is utilized to determine the sequence of the products as they are released into the system. The effects of the backlog list on the

performance of the system is an area that hasn't been investigated extensively. The major focus of this research is to determine these effects and their repercussions to sequencing policy for the backlog list.

The backlog list, for example, can be exploited to sequence products in a manner that yields higher levels of throughput for the same number of cards in an environment with products possessing variable processing times and with individual bottlenecks at different stations. Also, the sequence may itself affect the number of cards that are necessary to obtain the maximum level of throughput. Thus, the multi-product problem can be considered as that of sequencing the backlog list to obtain maximum throughput as the primary criterion and minimizing the number of cards as a secondary criterion. Optimization of throughput and card count is imperative when demand exceeds capacity.

Thus, the performance measure of concern in this research is the throughput rate. The factors that determine bounds on the throughput rate are investigated. The research examines the types of sequencing policies that facilitate achieving the upper bound on throughput rate in a system.

Another issue of concern, as mentioned above, is to determine how to utilize sequencing policy to minimize the card count (WIP) necessary to achieve the optimum throughput rate. For the multi-product scenario, the optimum WIP level (m^*) is the minimum level that will produce optimum throughput. Minimizing (m^*) also minimizes mean cycle time, by Little's law.

Although Kanban has provided benefits for those who have been able to incorporate it successfully, it has the disadvantage of not being suited well to many manufacturing environments. CONWIP, on the other hand, can obtain the benefits of pull type systems in less pristine manufacturing environments. When a wide changing mix of products is required, Kanban becomes impractical due to its requirement for WIP of each product at each station. The repercussion of Kanban in this environment is high WIP inventory costs. CONWIP doesn't require maintaining WIP for each product and thus can overcome this drawback.

The wide range of products may also possess variable processing times and bottlenecks. These conditions also impede the performance of a Kanban system and thus warrant consideration of the CONWIP system as an alternative. In CONWIP, WIP will build at the bottleneck whereas in Kanban WIP tends to build upstream of the bottleneck. For this reason, CONWIP can maintain a higher utilization of the bottleneck resulting in higher levels of throughput, (Spearman et al. (1990)) . It is desired for Kanban in the JIT environment to maintain a smooth balanced flow of operations at all stages, (Monden (1983)). Variability in processing times is thus undesirable in Kanban. In contrast, CONWIP does not require non-variable balanced operation.

For the above reasons, it appears that CONWIP, in comparison with Kanban, is better suited for the less restrictive production environment, where a wide range of products with variable processing times and bottlenecks are present. Due to the recent development of the CONWIP system, this particular scenario has yet to be investigated.

In order to implement the CONWIP system properly in an industrial environment of this type, it would be highly beneficial to possess results for the expected effects on the system's behavior when these circumstances are present.

The advantages of pull systems, when contrasted with push systems, provide the motivation for utilizing a pull type system. To distinguish, in a push system, a job's start time is determined by subtracting its lead time from its due date, whereas in a pull system, the start of a job is triggered by the completion of another job and cannot be scheduled a priori. The operation of the two types of systems can be contrasted as follows: a push system controls the throughput and measures WIP while a pull system controls WIP and measures throughput.

A pull system has a strong competitive advantage due to a greater ease with which WIP can be controlled compared to throughput. In a push system, throughput must be controlled by specifying an input rate. To provide optimum throughput in a push system, the difficult task of first estimating the capacity of the line and then providing an equivalent input rate into the system must be accomplished. However, it may be difficult to do so as the available capacity depends a complex interaction of a number of factors. When input exceeds capacity, WIP will grow without bound. If input is below capacity, then throughput will equal the input rate. On the other hand, WIP can simply be controlled by cards, computers, or other means in a pull system.

Another extremely important ramification of the difference in the control mechanisms of the two types of systems is their robustness, i.e., the effect on results when

an error occurs in their control mechanisms. When a push system over estimates capacity, WIP and cycle times will grow larger until a corrective action is taken. Undesirable actions such as cancellation of orders and overtime would become necessary. In a pull system, WIP can be adjusted slightly based upon whether or not a sufficient amount of work is being maintained at the bottleneck of the system.

An undesirable phenomenon that frequently occurs while operating with a push system is the "Overtime Vicious Cycle" as described in Spearman et al. (1990). Initially, average capacity is estimated based upon availability of the bottleneck station. Capacity is booked by accepting orders, and a master production schedule (MPS) is established. Work is released by the MPS at an average rate equal to the average capacity that is expected. The bottleneck has the potential of becoming starved - a situation not accounted for in the initial capacity determination. Thus, actual capacity is less than originally estimated. While the bottleneck is starved, work continued to be released into the system. This causes WIP to increase since the work was released into the system at a greater rate than the actual capacity. The increase in WIP results in the increment of flow time which in turn causes orders to be late. Overtime must be incurred to alleviate the problem of late orders.

Operating with CONWIP prevents this phenomenon from occurring. As a result, it provides more reliable flow times. With the CONWIP system, it is much easier to foresee the potential for late orders. Overtime will still be necessary to meet the due dates of some orders, but the system doesn't deteriorate by building up excess WIP as a push

system normally does. CONWIP essentially provides a feedback mechanism by which the system prevents WIP from building. In addition, the feedback mechanism allows the availability of unanticipated capacity to be taken advantage of, assuming there is demand for that capacity.

An appropriate modeling representation of the two types of systems would involve a closed queuing network for a pull system and an open queuing network for a push system. These models can provide corroborating evidence of why a pull system outperforms a push system. It has been shown, for equivalent open and closed exponential queuing networks with the same throughput, that a closed system will possess lower WIP and lower average flow times, (Spearman and Zazanis (1988)). Of course, an exponential model isn't always an accurate representation of an actual system, but, with a wide variety of parts with variable processing times it approaches being an appropriate representation, (Spearman and Zazanis (1988)).

Additional results presented in the literature suggest that closed networks (the model for pull systems) have flow times with lower variance than open networks (the model for push systems) (Whitt (1984)). This result is a strong competitive advantage for pull systems in terms of setting lead times. For a fixed mean flow time and lead time, reducing variance in flow time will result in a reduced average finished goods inventory (FGI), (Spearman et al. (1990)). Also, intuitively, if the lead time is set properly, then a lower variance in flow times will result in a lesser number of late jobs.

1.2 Statement of Problem

The problem considered in this research is to determine an optimal sequence for releasing multiple products under the CONWIP production control strategy. The primary objective is to provide guidelines of a policy to sequence jobs on the backlog list to achieve optimum throughput. A secondary objective is to provide a methodology to generate the backlog sequence that requires minimal WIP to achieve the optimum level of throughput. Concurrently, minimizing WIP will minimize mean flow time which is, thus, a part of the secondary objective.

We make the following assumptions:

- (1) All requirements set forth in the description of the CONWIP system are assumed to hold. These include the following:
 - (a) known card count (m), the WIP level for the line that is not to be exceeded,
 - (b) known backlog list that is under the jurisdiction of production control,
 - (c) equal amount of work at the bottleneck station for all products,
 - (d) First-come-first-serve (FCFS) queue discipline at all stations with no passing,
- (2) Fixed deterministic processing times which include set-up times that are not schedule dependent,
- (3) Queues may form at any station,
- (4) The system initiates scheduling with the first (m) jobs of the backlog list in queue at the first station. All other queues are empty,

- (5) No machine may process more than one job at a time,
- (6) No two operations for the same job may be processed simultaneously,
- (7) Demand for all the products is deterministic and known for the determination of the backlog list,
- (8) Number of machines is known and fixed,
- (9) Demand exceeds capacity.

1.3 Research Objectives

The major objectives of this research are:

- (i) Study the sequence dependent bottleneck and determine its impact on throughput for a multi-product CONWIP system.
- (ii) Provide analysis for specific cases in order to obtain generalizations on the expected effects on system behavior, i.e., throughput, flow times, and optimal card count, of different strategies for the backlog list. Determine bounds on throughput and optimal card count, and establish guidelines for sequencing policies based upon the analysis.
- (iii) Provide a methodology to determine the optimal value of card count for a predetermined sequence.
- (iv) Provide an efficient methodology to sequence the backlog list with the primary objective of optimizing throughput and the secondary objective of minimizing card count and mean flow time.

1.4 Outline of Thesis

The outline of the thesis is as follows. In Chapter 2, a literature review of the CONWIP production system and the areas that are significantly related to it is presented. Results regarding the effects of sequencing policy upon the system performance measures of throughput, optimum WIP levels and cycle times are presented in Chapter 3. In Chapter 4, a methodology to provide a sequence independent lower bound on the optimum number of jobs in the system (m^*), given information regarding their processing times is presented. Also, a method is presented to determine m^* utilizing recursive equations. Chapter 5 then develops a heuristic algorithm to sequence jobs so that the optimum WIP level (m^*) is minimized. Results of experimentation with the algorithm are also presented in this chapter. Finally, Chapter 6 presents conclusions and recommendations for further research.

2.0 Literary Review

The objective of this chapter is to provide a presentation of literature contained in the CONWIP area as well as other areas significantly related it. These related areas are closed queuing networks, flow shops, and Kanban.

2.1 CONWIP

Precursors to the reasoning behind the CONWIP production system can be found in the results of Spearman and Zazanis (1988). They examine why pull systems outperform push systems. They submit three conjectures for proof. First, pull systems have less congestion than push systems. Second, pull systems are easier to control than push systems. Lastly, the benefits of pull systems are more attributable to maintaining constant WIP than the pull mechanism itself.

The congestion conjecture is tested by utilizing open and closed queuing network models to represent the two types of systems. For the control mechanism conjecture, arguments are made why controlling WIP for a pull system is easier than estimating capacity for a push system. Also, arguments are made concerning why the robustness of the control mechanism of a pull system is better than that of a push system. To prove the conjecture that benefits of pull are more attributable to maintaining Constant WIP than the pull mechanism itself, a CONWIP type system is tested against Kanban for the same WIP.

These results provided the motivation for the development of the CONWIP system by Spearman, Woodruff, and Hopp (1990). The paper includes push and pull system comparisons, a description of the operation of a single CONWIP production line, a simulation study confirming superiority of CONWIP over traditional push system, and suggestions for future areas of research.

Hopp and Spearman (1991) model a single product CONWIP production line with deterministic processing times and random outages (DPRO). They justify DPRO as a

realistic scenario for an industrial environment as opposed to an exponential system. They develop an Approximate Regeneration Model (ARM) incorporating renewal theory to estimate throughput. They utilize simulation studies to compare their model with the methodology of Mean Value Analysis (MVA), Reiser and Lavenberg (1980). Their results provide evidence their model is robust and superior to MVA.

Spearman (1991) presents an analytical model for a CONWIP production line with exponential processing times. The model incorporates three parameters of bottleneck rate, raw processing time, and a congestion coefficient. He demonstrates that observation of cycle time for a given WIP level can be utilized for predicting cycle times at other WIP levels. This is applied in their model by establishing propositions regarding the behavior of the WIP / Cycle time curve. He compares the model with simulation results and MVA over a range of conditions.

Duenyas and Hopp (1993) address an exponential system consisting of fabrication lines feeding an assembly station. The start of a new job occurs at the fabrication lines when an assembly has been completed. In this manner, the assembly will pace the fabrication lines. They derive an upper bound for throughput in the assembly system. The upper bound is then applied to obtain an approximation for throughput and queue sizes. Then the approximation is tested with simulation studies for a variety of conditions. They illustrate how their approximations can be utilized for capacity and WIP setting decisions. In addition, they provide qualitative observations concerning the behavior of the assembly system.

Duenyas and Hopp (1992) model a DPRO CONWIP assembly system. A closed form expression for the approximation of throughput for the assembly system is provided. The throughput approximation is then compared with simulation results over a range of cases. Their results show under what conditions the model will work well. They observe that throughput is an increasing function of processing rate but is not an increasing function for repair rates or failure times. In addition, they conclude that it is advantageous to locate a bottleneck in fabrication as opposed to assembly.

Duenyas, Hopp and Spearman (1993a) address quota setting for pull systems. Pull systems do not explicitly consider due dates and therefore other means are necessary to maintain due date integrity. They demonstrate how quotas are utilized when demand is below capacity. Models with fixed and variable costs of overtime, lost sales and backorders are analyzed. The profitability associated with reducing the variance of the production rate is established. Analysis of the tradeoffs associated with under capacity scheduling for four models is provided.

Duenyas, Hopp, and Spearman (1993b) extend the results for a single DPRO CONWIP line by approximating the variance of the throughput rate of this system. This result can be utilized in conjunction with the previously obtained approximation for the mean of the throughput rate in order to obtain the distribution for the throughput rate. The distribution can then be incorporated into an algorithm for the purpose of setting WIP and quotas for the line. They illustrate the robustness of their model for these purposes with empirical tests. They utilize a profit function that incorporates production quotas and overtime costs. Their results show their algorithm to be quite robust.

Table (2.1-1) Summary of CONWIP Literature

AUTHORS	SUBJECTS PRESENTED
Spearman and Zazanis 1988	Conjectures of less congestion and easier control of pull systems Superiority of Pull due to Constant WIP not pull mechanism
Spearman, Woodruff and Hopp 1990	Description of CONWIP production system and arguments of its superiority to other systems
Hopp and Spearman 1991	Estimation of throughput for single product CONWIP system with deterministic processing times and random exponential outages (DPRO)
Spearman 1991	Analytical model for exponential processing times relating mean cycle time or throughput as a function of WIP for closed production systems
Duenyas and Hopp 1992	Approximation for throughput for DPRO fabrication lines feeding DPRO assembly station and observations based upon conditions of system
Duenyas and Hopp 1993	Approximations for throughput and average WIP for exponential fabrication lines feeding an exponential assembly station operating under CONWIP
Duenyas, Hopp and Spearman 1993a	Models addressing quota setting. Analysis of consideration of conditions to allow for "Under Capacity Scheduling"
Duenyas, Hopp and Spearman 1993b	Approximation of mean and variance of throughput for DPRO CONWIP system and implementation of results for quota and WIP setting purposes

2.2 Closed Queuing Networks

Closed queuing networks (CQN) are excellent representations for the CONWIP system, (Spearman *et al.* (1990)). In contrast, attempts at modeling stochastic Kanban systems have encountered great difficulties. The knowledge contained in the CQN area is of great assistance in predicting the performance for the CONWIP system. Therefore, the significant research of CQN warrants presentation in this review.

Gordon and Newell (1967) consider the problem of obtaining the equilibrium distribution of customers in a CQN with exponentially distributed service times. The equilibrium equations for the joint probability distribution of customers are found with a separation of variables technique. The CQN is shown to be equivalent to an open network where the number of customers cannot exceed a certain level. Assumptotic analysis is given for systems where the service rates for stations are either uniform or non-uniform. It is shown for non-uniform systems that the distribution of customers is governed by the slowest service rate.

Baskett *et al.* (1975) derive the joint equilibrium distribution for queue sizes with different classes of customers for mixed networks of queues. The networks may be closed to some classes of customers and open to others. The model has four different service centers, different types of customers with different routings and customers may change types in the system. The model is motivated by the desire to represent computer systems. From their results, other measures of system performance can be obtained including cycle times, throughput rate, and utilization for service stations.

Reiser and Lavenberg (1980) developed a method termed Mean Value Analysis (MVA) due to problems previously encountered in determining the joint equilibrium distribution. They reveal that the joint distribution is more information than is needed for determining system performance. This eliminates the need for determining normalization constants and product terms found in previous methods. The MVA algorithm is based on the notion that an arriving customer sees the system in its steady state with itself removed. The algorithm incorporates iterations for computing mean cycle time, throughput and the average number of jobs seen at each station.

Significant effort can be found in the area of determining "monotonicity" properties for CQN. The property of monotonicity is used to describe a behavior of the system or individual stations of the system. Monotonicity for CQN is a term to describe a well-behaved parameter as the number of jobs in the system is changing in a certain direction.

Suri (1985) models an exponential (CQN). He shows that throughput is non decreasing as the number of jobs in the system is increased for a system with homogeneous service times, i.e., it possesses monotonicity. Also, the average number of jobs at each station increases as the number of jobs in the system increases.

Shanthikumar and Yao (1987) prove that the vector of queue lengths are stochastically increasing as the number of jobs in the system is increasing and thus monotonicity is applicable. They extend Suri's results to the case where all stations have

service rates that are non decreasing. Their results are applicable in the transient state, whereas Suri's results are for equilibrium.

Adan and Van der Wal (1989) consider Erlang and general service times for CQN. Monotonicity for throughput is proved in their research for these cases. They implement a sample path argument for the general service time case. They investigate both single and multiserver stations.

Stecke and Solberg (1984) show that unbalanced configurations and workloads outperform balanced configurations and workloads in CQN. The goal of their study is to obtain theoretical results to be incorporated in FMS design. They examine how to partition machines into groups and their findings indicate that fewer groupings with unbalanced configurations provide better production rates. In addition, unbalanced allocations of work to the unbalanced groups maximizes the expected production rate. Balancing the work load for the groups is optimal only when the groups are balanced.

2.3 Flow Shop

The problem being considered in this research is essentially optimization of a flow shop with a constant number of jobs in its steady state. In the CONWIP system work is still pushed between serial work stations. It is designated as a pull system due to the fact that a new job begins only upon completion of a previous job. It doesn't schedule jobs as a traditional push system does. Sequencing of CONWIP's backlog list parallels the desire to schedule a flow shop. The area of flow shops has been thoroughly researched during the past 40 years. A vast array of procedures have been investigated for flow shop scheduling. These procedures warrant investigation due to their potential adaptability to the scheduling of CONWIP through its backlog list.

The literature for flow shop scheduling has strongly concentrated on the objective of minimizing makespan. Constructive algorithms are one approach. Constructive algorithms are those algorithms that guarantee optimality with a single algorithmic pass. No constructive algorithm exists for systems with more than two machines. The combinatorial nature of scheduling flow shops is the reason why constructive algorithms for larger systems have not been obtained.

Johnson (1954) presents a constructive algorithm for the two machine problem and for special cases of the three machine problem. This algorithm schedules jobs beginning with jobs with processing times progressing from short to long. Then it schedules the jobs with processing times progressing from long to short. The algorithm's approach and

choice of minimizing makespan as the objective have had significant influence on future results in this area.

Procedures for scheduling flow shops have included branch and bound algorithms, mathematical programming algorithms, elimination algorithms and heuristic algorithms. The investigation of the flow shop literature will focus on heuristic algorithms, since the other methods are not expected to be applicable to this research.

Heuristic algorithms can be divided into functional heuristics and sequence building heuristics. A functional heuristic attempts to derive numeric values based upon a job's processing times at different stations and then arrange the values in a monotonic order to obtain a sequence. They generally incorporate the results of Johnson (1954) such as minimizing start-up machine idle time and shut-down machine idle time. Heuristics based upon these objectives appear biased towards the optimization of a transient cycle as opposed to a CONWIP system in its steady state cycle. Thus, the results of functional heuristics are not expected to be applicable to this research for the CONWIP system. Therefore, the presentation of functional heuristics will be brief.

Page (1961) first introduced the concepts of slope-indexes and priority functions for heuristics. He approached the problem as a sorting problem with techniques such as individual and group exchanging, merging and pairing.

Palmer's heuristic (1965) incorporated concepts from Page. He utilized slope indexes to determine sequence based upon giving priority to a job with the tendency of progressing from short to long processing times. The heuristic attempts to minimize start-

up machine idle time and shut-down machine idle time. The heuristic is computationally fast, but is not as successful as the number of machines increases.

Gupta (1971) incorporated results from Johnson, Palmer and Page in a functional heuristic. He shows the heuristic to provide lower makespans than Palmer's heuristic. Bonney and Gundry (1976) provide a functional heuristic with similarities to Palmer's and Gupta's heuristics. It utilizes start and end slope indexes calculated with linear regression. The algorithm was compared with the ones previously discussed and appeared to perform better for larger problems.

The other category of heuristics is sequence building. Campbell, Dudek and Smith (1970) develop a sequence building heuristic which is an extension of Johnson's algorithm. They partition the problem into two machine problems and solve the two machine problems using Johnson's algorithm.

Gupta (1972) developed three sequence building heuristics of MINIT, MICOT and MINIMAX. They incorporate Johnson's logic, combinatorial approaches and minimizing idle or completion time on the last station. Due to problems inherent in these heuristics, Gupta and Maykut (1973) attempt to resolve these difficulties by revising the previous heuristics with a heuristic decomposition method.

King and Spachis (1980) provided five heuristics based upon reducing machine idle time in the system. Stinson and Smith (1982) provided heuristics for reducing job idles time in addition to machine idle times. They test jobs as immediate predecessors to all

other jobs with the objective of determining how well a job fits in terms of machine and job idle time with all other jobs in the system.

Their heuristic consists of two stages. The first stage determines a cost matrix $C(i,j)$. This cost matrix is found for all possible pairs of predecessors (i) and successors (j) excluding the diagonal. To determine the elements in the cost matrix, residuals are summed from the values of $t(i, k) - t(j, k-1)$. Thus, lower costs are associated with closer processing times on machines k and k-1 for jobs i and j which reduces the likelihood of machine and job idle times.

The second stage of the algorithm generates a closed ring of successors and predecessors from the cost matrix with a traveling salesman approach. The approach consists of finding the difference between the smallest and second smallest element in each row and column, then choosing the smallest element that has the highest difference with the second smallest element in all rows or columns. The selection of this element eliminates all other elements in its row or column. This step is then repeated until the sequence is obtained.

Six different heuristics each which differently emphasizes positive and negative residuals were developed and tested. Their results showed negative residuals to be more deserving of stronger weighting, but due to the difference in objectives of this research their findings are not immediately considered to be applicable.

2.4 Kanban

CONWIP was created in order to provide a potential alternative to the Kanban production system. In order to shed light on some of the difficulties that are presented with Kanban and how CONWIP might alleviate them , the following literature was reviewed.

Kanban is a method of production that is incorporated to meet the needs of the manufacturing philosophy known as Just-in-Time (JIT). JIT consists of broad range of objectives encompassing issues such as reducing setup times, perfect quality, and worker involvement. An extensive simulation study by Krejewski et al. (1987) provides evidence that proper conditions for the manufacturing environment are a prerequisite for JIT to be appropriate for an organization. Many factors that affect the performance of JIT were investigated including product structures, demand and supplier influences, buffer mechanisms and facility design. Kanban is not a separate entity from JIT and therefore Kanban can not be utilized if JIT is not utilized.

Spearman and Zazanis (1988) site such factors as large numbers of different parts, long setup times, and unbalanced processing time requirements as being detrimental factors to the implementation of the Kanban system.

Simulation has been frequently utilized to analyze the behavior of Kanban due to the difficulty in obtaining analytical results. A comprehensive critique of the available results in simulation research of JIT systems incorporating Kanban is provided by Chu and Shih (1992). They find that although there has been a wide range of simulation results

presented, many concerns still remain. They express concern that conclusions have been formed without utilizing proper statistical and simulation methods. They conclude that many important questions for successful JIT and Kanban implementation are still unanswered.

Several studies have addressed the need for smooth and balanced operations in the Kanban system. This is highly relevant to this research since the performance of a multi-product CONWIP system with unbalanced and variable processing times is investigated. The necessity for smooth balanced operation in Kanban has been confirmed by the simulation studies addressing this issue.

Villeda et al. (1988) performed a simulation study of the effects of imbalance and variability for a model with three lines each with three stations feeding an assembly station. The study illustrated that systems with high variability may create a preference for unbalanced systems. The effects of imbalance for four scenarios of the three stations were investigated. The four scenarios were high-low-high "bowl phenomenon", low-medium-high, high-low-low, and high-medium-low.

The "bowl phenomenon" in push systems has been previously shown to improve production rates. However, results from this study determined that the high-low-high scenario did not improve the production rate relative to a balanced system even with high variability in the system. Also, for the cases of low-medium-high and high-low-low the production rates were not improved. The high-medium-low showed the ability to improve the production rate slightly.

For all unbalanced cases and the balanced case, it was shown that production rate decreased as variability in the processing times at the stations was increased with no variability at the final assembly station. When higher variability is present, it was observed that unbalanced systems have higher production rates than balanced. Two additional significant observations were made. The first is that as the variability in the final assembly station increases, the production rate for both balanced and unbalanced systems decreases. Second, improvement with unbalanced systems is more significant with higher variability at the final assembly station. The main conclusion is that imbalance can be beneficial when high variability exists in the system with high-medium-low imbalance generally being preferred to the others.

Simulation study by Gupta and Gupta (1989) produced findings that showed balanced operations and reduced variability are imperative for Kanban to operate effectively. For their model, variability in processing times reduced the production rate and increased shortages for a fixed WIP level.

Sarker and Harris (1988) also address the need for balance in the Kanban system. Their findings for an unbalanced system include that unbalance produces unequal utilizations, fluctuations in production rate, and blocking and starvation on the line. They conclude that balanced operations are necessary for Kanban to operate properly.

Results from Changchit and Kung (1988), Huang et al.(1983), and Krejewski et al.(1987) reinforce the conclusions from the studies previously mentioned. Therefore, the

consensus of the studies is to attempt to achieve balanced and smooth operations in the Kanban system as in Monden (1983).

Another important concern in implementing the Kanban system is determining the number of Kanbans that are needed. The number of Kanbans or amount of WIP in the Kanban system needs to be determined for each station in the system. In contrast, for CONWIP the much less difficult task of determining the WIP level for the entire line needs to be accomplished. Traditionally, the Japanese have approached the problem through experimentation in their facilities. Then as product mix changes the system is adjusted through further experimentation.

Wang and Wang (1990) incorporate a Markov process approach for determining the number of Kanbans. The model considers demand for products as the departure rate and the production rate as the input rate. The state space is the number of Kanbans in the system. The transition matrix and its limiting distribution are found. Then an expected cost consisting of holding and shortage costs for each value of the number of Kanbans is determined. The number of Kanbans with the lowest associated total cost is optimal. The methodologies for three different production configurations are provided.

The issue has also been mentioned in simulation studies by Changchit and Terrel (1988) and Ebrahimpour and Fathi (1984). Chu and Shih (1992) in their review of simulation results for Kanban systems conclude that the issue of determining the number of Kanbans or inventory level has yet to be adequately resolved.

In conclusion, the extremely important concern for determining the number of Kanbans or inventory level for the system is an issue that doesn't presently contain enough knowledge to be relied upon in an implementation for a Kanban system. Some of the limited results may be of assistance but it appears this difficult task must be carried out by an organization with a trial and error approach. Thus, a proper implementation of Kanban is anticipated to present difficulties that would not be incurred with implementing CONWIP as was alluded to in Spearman and Zazanis (1988).

3.0 CONCEPTS AND CHARACTERISTICS OF CONWIP

3.1 One Product Case

The simplest case for the CONWIP production system is the one involving a single product. For deterministic processing times, the system will go through an initial transient state and then quickly reach a steady state where flow times, time between departures, and throughput are constant for a fixed level of WIP (m).

The primary concern, when operating with a single product line, is to determine the optimum number of jobs m^* in the system. The value of m^* is the first integer value of (m) that produces the maximum level of throughput. Throughput increases with increment in m , until m reaches m^* , and then, it becomes constant for values of m above m^* . The WIP level of m^* yields 100% utilization of the bottleneck resource. The value of m^* in the single product case, where t_i is the processing time at station i and t_b is the processing time at the bottleneck station, can be given as follows: (Hopp and Spearman 1991)

$$(3.1.1) \quad m^* = \frac{\sum_{i=1}^N t_i}{t_b}$$

When m^* is a fractional value, it must be rounded up to the next integer value. The limit for m^* in the one product case ranges from 2 to N . For the lower limit of $m^* = 2$, the bottleneck will be extremely dominant or the number of stations will be small. When the bottleneck's processing time is at the same level as the other stations, the value of m^* approaches N , where N = number of stations. Thus, for a balanced system $m^* = N$.

Operating at m^* is critical for several reasons. First, when operating below m^* , throughput will be lower than that can be achieved. Second, when operating above m^* , the system will form excessive WIP at the bottleneck thereby increasing flow times.

The two most significant parameters to describe the operation of a CONWIP system are its throughput rate Θ and flow time or cycle time C . For the one product deterministic system operating with a WIP of m^* in steady state, these parameter values are,

$$(3.1.2) \quad C = m^* \cdot tb$$

$$(3.1.3) \quad \Theta = \frac{1}{tb}$$

These relationships illustrate that the throughput rate is inversely proportional to the bottleneck processing time. The flow times are directly proportional to the bottleneck time and the number of jobs in the system. As the WIP level is increased above m^* , the flow time increases as dictated by expression (3.1.2) while throughput remains constant.

Holding non-bottlenecks constant, the location of the bottleneck will not affect the behavior of the CONWIP system for the one product case. That is, throughput, flow times and m^* are not affected by the location of the bottleneck station.

3.2 Introduction to Multi-product Case

An important element of the CONWIP production system proposed by (Spearman et al. (1990)) is a backlog list. This backlog list is used to determine which job will next enter the system when a job has been completed. The backlog list provides an opportunity for a production control personnel to determine product sequencing.

In the one product deterministic case, a queue can only form at the bottleneck station of that product. This station bounds the throughput rate. In the multi-product case, sequencing among jobs can be utilized to create a queue at the combined bottleneck station (cbn), a station with the largest amount of processing time t_{cbn} for all the products. The throughput rate will now be bounded by the cbn. For this reason, the throughput rate can be improved by properly mixing the products in the sequence. A prerequisite for this phenomenon to occur is that all products should not possess bottlenecks at the same station.

Deterministic systems operating under CONWIP with repetitive sequences will first pass through a transient phase before they reach steady state. This steady state in the one product case is achieved when the flow times and throughput rate have become constants. In the multi-product case with repetitive sequences, a steady state is also reached when the average flow time and throughput rate become constants.

Assuming long runs of individual products, a one product average can be utilized to illustrate that mixed sequences can provide a higher upper bound on throughput than unmixed. This one product average for throughput would be equivalent to arranging the

backlog list of CONWIP with long runs of one product type followed by long runs of others. This would render the effects of the transition periods between products to be insignificant. The throughput rates bounded by (t_b) for single products not mixed in the system can be incorporated to determine the one product average. It can then be shown that the throughput rate that is obtainable with mixed sequences governed by the cbn is superior to unmixed sequences. We present this in section 3.4.

We examine in section 3.3 as to how a system behaves when a product changeover occurs. The average flow times and throughput rates during these transitions are investigated. Analysis is also provided in order to determine how certain factors affect the behavior of these transitions.

The effect on average flow times due to the mixing of products is discussed in section 3.5. The factors affecting m^* for the multi-product case are presented in section 3.6. The effects from dividing the products into groups are discussed in section 3.7.

The following notation will be used for the multi-product CONWIP system:

m = constant number of jobs in the system

m^* = the first value of m to produce optimum throughput in steady state

$t(i,j)$ = deterministic processing time including set-up time of job i at station j

N = the number of stations in the serial production line

n = total number of jobs

k = the total number of product types in the sequence

$C(i)$ = cycle time or flow time for job i

$C_{tr}(i)$ = average cycle time during a transition for job i

Θ = throughput rate

cbn - station with largest sum of processing time for all the jobs in the sequence

$tcbn$ = sum of processing time at the cbn for each product type in the sequence

$tcb(i)$ = processing time of job i at the cbn

$tb(i)$ = processing time of product i at its bottleneck station

$tq(j)$ = queuing time at station j

$tr(i, i')$ = return time to bottleneck for jobs i and i'

$d(i)$ = demand for product i

3.3 Transitions

When a new type of job begins to be released into the system following a previous job in its steady state, a transition phase occurs. This transition is characterized by a build up at the new product's bottleneck as the previous product's bottleneck is being depleted. The system's behavior during these transitions is affected by whether the bottleneck is moving upstream or downstream. Figure (3.3.1) illustrates these bottleneck movements.

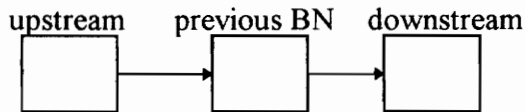


Figure (3.3.1) Illustration of Bottleneck Movement during Transitions

Transitions upstream can be expected to be characterized by first a fall in flow times and then a rise in flow times for the products. The fall in flow times normally occurs as the new product in the sequence is gradually waiting for less time in the queue of the previous product's bottleneck station downstream. In addition, the new bottleneck upstream has yet to be formed. The rise in flow time normally occurs as the new product begins to gradually wait longer in queue at its own bottleneck that is forming upstream. The cumulative effect for the upstream transition is that the average flow time during the transition is generally lower than the flow time for the new product in its steady state. Holding (m) constant, this phenomenon must increase throughput by Little's law.

For downstream transitions, the most significant factor is the processing time of the previous product's bottleneck relative to the processing time of the new product's bottleneck. If the new product's bottleneck processing time is larger, transitions downstream are generally characterized by a linear rise in flow times to the steady state level of the new product. Similarly, when the new product's bottleneck processing time is smaller, there will be a linear decline in the flow times to the steady state level of flow time for the new product.

The behavior of both types of transitions is independent of the number of consecutive products of the same type in the sequence provided the transition occurs after the previous product has reached steady state. For example, the transition phase from job A to job B and B to A will be identical whether the sequence is 10A/10B, 50A/50B, or 100A/100B provided the transition occurs after the previous product has reached steady state.

In addition to the direction in which the bottleneck is moving, other factors affect the behavior of the system during transitions. These factors will be illustrated in examples (3.3.1-3.3.8).

Example 3.3.1.

The following example is an illustration of how flow times are affected during transitions. The WIP level (m) is fixed at four for the example. The processing time matrix $t(i,j)$ for job i at station j is given below. For product 1, the bottleneck station is 2, and for product

2, the bottleneck station is 4. Table (3.3.1) summarizes the transition behavior for this example. The data was obtained by Gantt chart calculations done by hand. The data is initiated with product two in its steady state. The first column provides the product type 1 or 2. The second column gives the state of the job and the direction of bottleneck movement during the transition. The third column gives the cycle time of the job. The time spent in queue at the bottleneck stations two and four, designated as $tq(2)$ and $tq(4)$, is given in columns four and five.

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 6 & 3 & 3 \\ 3 & 3 & 3 & 9 \end{bmatrix}$$

The WIP level $m = 4$ must provide optimal throughput for both products from equation (3.1.1). Therefore, the steady state flow times for the two jobs being processed individually and the throughput rates can be determined by the Little's law as follows:

$$C(1) = \frac{m}{\Theta} = \frac{4}{\frac{1}{6}} = 24 \quad C(2) = \frac{m}{\Theta} = \frac{4}{\frac{1}{9}} = 36$$

The average flow times $C_{tr}(1)$ and $C_{tr}(2)$ during the transitions from the values in Table (3.3.1) are,

$$C_{tr}(1) = 21.0 \quad C_{tr}(2) = 30.0$$

Table (3.3-1) Transition Effects for Example (3.3.1)

Product	State	Flow Time	tq(2)	tq(4)
2	ss	36	0	18
2	ss	36	0	18
1	tr-up	30	0	15
1	tr-up	24	0	9
1	tr-up	18	0	3
1	tr-up	15	0	0
1	tr-up	18	3	0
1	tr-up	21	6	0
1	ss	24	9	0
1	ss	24	9	0
2	tr-down	27	9	0
2	tr-down	30	6	6
2	tr-down	33	3	12
2	ss	36	0	18

The steady state flow times for the jobs being processed individually are (24, 36) for jobs one and two respectively. For the example problem, the average flow times for both the jobs during transitions are reduced from the steady state values. Holding (m) constant and reducing flow times must improve throughput by Little's law. Therefore, the net effect of the transitions is beneficial. The length of the upstream transition is six jobs and the downstream transition is three jobs. This example demonstrates the previously described behavior of upstream and downstream transitions. The following examples are provided in order to illustrate how the transition phases are affected as the processing times values $t(i,j)$ and (m) are perturbed.

Example 3.3.2,

$$m = 4, \quad t(i, j) = \begin{bmatrix} 5 & 6 & 5 & 5 \\ 5 & 5 & 5 & 9 \end{bmatrix}$$

Table (3.3-2) Transition Effects for Example (3.3.2)

Product	State	Flow Time	tq(2)	tq(4)
2	ss	36	0	12
1	tr-up	32	0	11
1	tr-up	28	0	7
1	tr-up	24	0	3
1	tr-up	21	0	0
1	tr-up	22	1	0
1	tr-up	23	2	0
1	ss	24	3	0
2	tr-down	27	3	0
2	tr-down	30	2	4
2	tr-down	33	1	8
2	ss	36	0	12

The average flow times during the transitions are,

$$C_{tr}(1) = 25.0 \quad C_{tr}(2) = 30.0$$

In this example, the non-bottleneck's processing times have been increased from three to five. The lengths of the transitions remain three jobs for downstream transitions and six jobs for upstream transitions. The upstream transition increased its average cycle time during the transition from 21 to 25. This can be attributed to the increment in the minimum cycle time (processing time only) for job one from 15 to 21. The transition downstream illustrates a behavior identical to that of example (3.3.1). It maintains the same number of jobs in the transition, a constant slope of three, and the same average cycle time of 30 during the transition.

Example 3.3.3,

$$m = 3, \quad t(i, j) = \begin{bmatrix} 3 & 6 & 3 & 3 \\ 3 & 3 & 3 & 9 \end{bmatrix}$$

Table (3.3-3) Transition Effects for Example (3.3.3)

Product	State	Flow Time	tq(2)	tq(4)
2	ss	27	0	9
2	ss	27	0	9
1	tr-up	21	0	6
1	tr-up	15	0	0
1	tr-up	15	0	0
1	ss	18	3	0
1	ss	18	3	0
2	tr-down	21	3	0
2	tr-down	24	0	6
2	tr-down	27	0	9
2	ss	27	0	9

The average flow times during the transitions are,

$$C_{tr}(1) = 17.0 \quad C_{tr}(2) = 24.0$$

In this example, the WIP has been reduced by one from that of example (3.3.1). All other factors are the same. The average cycle times during the transitions decrease. The transition upstream is shortened from six to three jobs. The cycle time for job one reaches its minimum of 15 during the transition. The transition downstream illustrates similar behavior to that of example (3.3.1) . It maintains the same number of jobs in transition and a constant slope of three.

Example 3.3.4,

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 9 & 3 & 3 \\ 3 & 3 & 3 & 6 \end{bmatrix}$$

Table (3.3-4) Transition Effects for Example (3.3.4)

Product	State	Flow Time	tq(2)	tq(4)
2	ss	24	0	9
2	ss	24	0	9
1	tr-up	21	0	3
1	tr-up	21	3	0
1	tr-up	24	6	0
1	tr-up	27	9	0
1	tr-up	33	15	0
1	ss	36	18	0
1	ss	36	18	0
2	tr-down	33	18	0
2	tr-down	30	12	3
2	tr-down	27	6	6
2	ss	24	0	9

The average flow times during the transitions are,

$$C_{tr}(1) = 25.2 \quad C_{tr}(2) = 30.0$$

In this example, the bottleneck locations have been switched. The larger bottleneck is now upstream. The transition downstream is now characterized by a linear fall in flow times as opposed to a linear rise in flow times. The slope is now negative three. The upstream transition again exhibits a fall and then rise in flow times. However, the average flow time during the transition increases to 25.2.

Example 3.3.5,

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 9 & 3 & 3 \\ 3 & 3 & 3 & 9 \end{bmatrix} \quad C(1,2) = 36,$$

Table (3.3-5) Transition Effects for Example (3.3.5)

Product	State	Flow Time	tq(2)	tq(4)
2	ss	36	0	18
2	ss	36	0	18
1	tr-up	30	0	12
1	tr-up	24	0	6
1	tr-up	18	0	0
1	tr-up	18	0	0
1	tr-up	24	6	0
1	tr-up	30	12	0
1	ss	36	18	0
1	ss	36	18	0
2	tr-down	36	18	0
2	tr-down	36	12	6
2	tr-down	36	6	12
2	ss	36	0	18
2	ss	36	0	18

The average flow times during the transitions are,

$$C_{tr}(1) = 24.0 \quad C_{tr}(2) = 36.0$$

In this example, both jobs have a bottleneck with the same magnitude. This results in a downstream transition that maintains a constant cycle time during transition.

However, the time spent in the two queues changes during the transition. The upstream transition exhibits a fall and then rise in the flow times as has been seen in the previous examples.

Table(3.3-6) summarizes the effects that occur when non-bottleneck times are altered. The processing times of job 1 at the non-bottleneck stations were varied within the range of (0-6) while those of job 2 were varied over the range (0-9). It was observed that this affected the average transition times $C_{tr}(1)$ and $C_{tr}(2)$.

Changes made in $t(1, j \neq bn)$ resulted in changes of $C_{tr}(1)$ similar to those of $t(1, j \neq bn)$ for all non-bottlenecks, i.e., increments caused increases and decrements caused decreases. No changes occurred in $C_{tr}(2)$ when $t(1, j \neq bn)$ were increased. However, different behavior was observed for $C_{tr}(2)$ when $t(1, j \neq bn)$ was decreased. No change in $C_{tr}(2)$ occurred when $t(1, 1)$ was decreased, while decreases in $t(1, 3)$ and $t(1, 4)$ resulted in increases of $C_{tr}(2)$. Decreases in $t(1, 3)$ and $t(1, 4)$ caused an increment in $tq(2)$ thereby resulting in the increment of $C_{tr}(2)$. Whereas, a decrease in $t(1, 1)$ did not increase $tq(2)$ or $C_{tr}(2)$.

$C_{tr}(1)$ did not change for increment or decrement of $t(2, j \neq bn)$. No changes were observed for $C_{tr}(2)$ when $t(2, j \neq bn)$ was decreased. However, $C_{tr}(2)$ was affected due to increment in $t(2, j \neq bn)$ and each station displayed its own unique behavior. $C_{tr}(2)$ was not affected as long as $t(2, 1) \leq 6$. The increment in $t(2, 1)$ was canceled out by decrement in $tq(2)$. $C_{tr}(2)$ increased for increment of $t(2, 1) > 6$ due to a queue that forms at station 1. $C_{tr}(2)$ increased due to increment in $t(2, 2)$. It was observed that increments in $tq(2)$ were canceled out by decrements in $tq(4)$ and $tq(1)$ and $tq(3)$ did not change. However, $C_{tr}(2)$ must increase due to increment in $t(2, 2)$. $C_{tr}(2)$ was increased due to increment in $t(2, 3)$ due to the increment in $t(2, 3)$. Increments in $tq(3)$ were canceled out by decrements in $tq(4)$ while $tq(1)$ and $tq(2)$ did not change.

Example 3.3.6.

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 6 & 3 & 3 \\ 3 & 3 & 3 & 9 \end{bmatrix}$$

Table(3.3-6) Non-Bottleneck Effects

Processing Time	Change in $t(i, j)$	$C_{tr}(1)$	$C_{tr}(2)$
$t(1, 1)$	increase	increase	no change
$t(1, 1)$	decrease	decrease	no change
$t(1, 3)$	increase	increase	no change
$t(1, 3)$	decrease	decrease	increase
$t(1, 4)$	increase	increase	no change
$t(1, 4)$	decrease	decrease	increase
$t(2, 1)$	increase	no change	increase $t(2, 1) > 6$
$t(2, 1)$	decrease	no change	no change
$t(2, 2)$	increase	no change	increase
$t(2, 2)$	decrease	no change	no change
$t(2, 3)$	increase	no change	increase
$t(2, 3)$	decrease	no change	no change

In the following examples (3.3.7-3.3.8), one non-bottleneck processing time in the system is altered from its original time in example (3.3.1). The new values are denoted as C' , $tq(2)'$, $tq(4)'$ for the cycle time, time spent in queue at station two, and time spent in queue at station four respectively. The original values from example (3.3.1) are denoted as C , $tq(2)$, $tq(4)$. In example 3.3.7, when the non-bottleneck processing time $t(1,3)$ was increased by one, it was observed that the upstream's transitional cycle times increased by one for the last four jobs in the transition. However, for the first three jobs in the transition, the increase in processing time was canceled out by a decrease in $tq(4)$. The downstream transition was not affected in this example.

In example 3.3.8, when the non-bottleneck processing time $(2,2)$ was increased by one, the downstream transition times increased by one for all four transitional jobs. This resulted from an increase in processing time $(2,2)$ while the changes in $tq(2)$ and $tq(4)$ canceled each other out. The upstream transition was not affected. The transitions in examples 3.3.7&8 indicate that the cycle time for the last job in the transitions increased slightly above the steady state cycle time before reaching its steady state cycle time. This is an interesting behavior that was not present in examples (3.3.1-3.3.5) where non-bottlenecks were held constant.

Example 3.3.7.

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 6 & 4 & 3 \\ 3 & 3 & 3 & 9 \end{bmatrix}$$

Table (3.3-7) Transition Effects for Example (3.3.7)

Product	State	C'	tq(2)'	tq(4)'	C' - C	tq(2)' - tq(2)	tq(4)' - tq(4)
2	ss	36	0	18	0	0	0
2	ss	36	0	18	0	0	0
1	tr-up	30	0	14	0	0	-1
1	tr-up	24	0	8	0	0	-1
1	tr-up	18	0	2	0	0	-1
1	tr-up	16	0	0	+1	0	0
1	tr-up	19	3	0	+1	0	0
1	tr-up	22	6	0	+1	0	0
1	tr-up	25	9	0	+1	0	0
1	ss	24	8	0	0	-1	0

Example 3.3.8.

$$m = 4, \quad t(i, j) = \begin{bmatrix} 3 & 6 & 3 & 3 \\ 3 & 4 & 3 & 9 \end{bmatrix}$$

Table (3.3-8) Transition Effects for Example (3.3.8)

Product	State	C'	tq(2)'	tq(4)'	C' - C	tq(2)' - tq(2)	tq(4)' - tq(4)
1	ss	24	9	0	0	0	0
1	ss	24	9	0	0	0	0
2	tr-down	28	9	0	+1	0	0
2	tr-down	31	7	5	+1	+1	-1
2	tr-down	34	5	10	+1	+2	-2
2	tr-down	37	3	15	+1	+3	-3
2	ss	36	0	17	0	0	-1
2	ss	36	0	17	0	0	-1

Examples (3.3.1-3.3.5) illustrated different factors that affect the behavior of transitions. First, all non-bottleneck times were increased in example (3.3.2). This resulted in the average upstream transition time $C_{tr}(1)$ to increase. In example (3.3.3), the WIP is reduced by one. This resulted in decreasing both the average transition times. Also, the number of jobs in the upstream transition decreased from 6 to 3. In example (3.3-4), the larger bottleneck was switched with the smaller bottleneck. As a result, the downstream transition times decreased instead of increasing while the average cycle time for the upstream transition increased. In example (3.3.5), both bottlenecks were made equal. The downstream transition maintained a constant cycle time. Examples (3.3.6,7,8) illustrated effects from changes in the non-bottleneck processing times. The examples indicate that the location of non-bottlenecks is a factor that affects the behavior of transitional cycle times when non-bottleneck processing times are perturbed.

3.4 Effects of Sequencing Policy

Sequencing policy can have a significant impact on the performance of the CONWIP system when the bottleneck of the system is sequence dependent. Sequence dependent bottlenecks exist when all products in the system do not have the largest amount of processing time at the same station. To optimize throughput, idle time must be eliminated at the system's bottleneck. Thus, the sequencing policy must facilitate work being maintained at the system's bottleneck. First, the true bottleneck station needs to be identified. The true bottleneck, i.e, the combined bottleneck (cbn), possesses the largest amount of processing time (t_{cbn}) from all the products in the system as dictated by equation (3.4.1) below.

$$(3.4.1) \quad t_{cbn} = \max_j \left(\sum_{i=1}^k d(i) * t(i, j) \right)$$

The cbn station bounds throughput for the multi-product case. The cbn has a strong probability of being at one of the individual products' bottleneck stations. However, it could also exist at a station that is not a bottleneck station for any of the individual products.

The three cases that may exist are,

- (1) All individual products' bottlenecks are not at the cbn
- (2) Some individual products' bottlenecks are at the cbn
- (3) All individual products' bottlenecks are at the cbn

In sections (3.4.1,3.4.2) it will be mathematically shown how mixed sequences of products can provide better throughput than long runs of single products. Long runs of single products will render the transition effects insignificant. Therefore the transition effects are not included in the analysis. The average value of throughput for the unmixed case will be used for comparison with the mixed case to illustrate how improvement in throughput is achieved with mixed sequences. It will then be shown that a mixed sequence can not yield an upper bound on throughput rate lower than the upper bound for the unmixed sequence. However, the upper bound on throughput is independent of sequence when all products in the sequence have their bottlenecks at the cbn, i.e., Case (3). For this case, as long as the number of jobs in the system is large enough to maintain 100 % utilization of the cbn, the throughput will not change regardless of the sequence.

3.4.1 Two Product Equal Demand Case

In this section, the two product case with equal demand for both products is considered. For the two product equal demand case, the cbn station with processing time t_{cbn} from both products is determined by the following:

$$(3.4.1.1) \quad t_{cbn} = \max_j \sum_{i=1}^2 t(i, j)$$

If the bottlenecks for the products are at different stations:

$$(3.4.1.2) \quad t_{cbn} \leq tb(1) + tb(2)$$

Consequently, the following inequality must be satisfied:

$$(3.4.1.3) \quad \frac{2}{t_{cbn}} \geq \frac{2}{tb(1) + tb(2)}$$

The first term of (3.4.1.3) is the upper bound on the throughput level for a mixed sequence. The second is applicable to the unmixed sequence. Thus, the above expression confirms that mixing the two products can increase the upper bound on throughput with the exception of Case (3).

If the product changeover in the backlog sequence is chosen not to be frequent, bottlenecks will form at the individual products' bottlenecks. These bottlenecks may not be at the cbn. This will be a source of idle time at the cbn. Idle time at the cbn causes throughput to drop. Consider case (2), when one job's bottleneck station is at the cbn and the other job's bottleneck station is not at the cbn. When the sequence changes occur less frequently, the job with the bottleneck not at the cbn begins to form a queue. This will

force idle time at the cbn and thus a drop in throughput. In that case, the only possible option is to increase the number of jobs in the system (m) during the slower sequence changes to maintain throughput, but obviously this is an undesirable option.

If the cbn is not at either one of the product's bottlenecks, i.e., Case (1), then a slow sequence change will begin to produce bottlenecks at both of the individual product bottleneck stations. Since these bottlenecks are not at the cbn, they will force idle time at the cbn. This will decrease throughput rate. Thus, Case (1) doesn't exhibit different behavior than Case (2). In addition, repeating jobs for Case (1) is expected to degrade throughput more than that of Case (2) since both bottlenecks can cause idle time at the cbn.

Rapid changes in the job sequence force the bottleneck to the cbn and prevent bottleneck buildups away from the cbn. Thus, rapid sequence changes will greatly facilitate the optimization of throughput. This will be illustrated in example 3.4.1.1.

Example 3.4.1.1,

The following matrix of processing times $t(i,j)$ will be utilized to illustrate the advantages that are achieved with fast changes in sequencing. The number of jobs in the system is held constant at ($m=4$) for all the cases. Since the cbn is at station two and the individual products have bottlenecks away from the cbn at stations one and four, there is an opportunity to positively affect throughput with proper sequencing.

$$t(i,j) = \begin{bmatrix} 6 & 10 & 6 & 12 \\ 12 & 10 & 6 & 6 \end{bmatrix}$$

The table below provides the throughput rates for different repetitive sequencing policies. The throughput rates were determined by simulating the flow of jobs through the system. The results in the table reinforce the notion that repeating products in the sequence will be detrimental to the throughput rate when the specified conditions are present. The table illustrates how the throughput rate is converging to its one product average of $1/12 = .0833$ as sequence changes are slowed. This occurs as the transition phase is gradually having less of an effect on the system.

Table (3.4.1.1) Throughput Values for the Two Product Example under Different Sequencing Policies

Sequence	Throughput
1A/1B	.1000
2A/2B	.1000
3A/3B	.0967
4A/4B	.0930
5A/5B	.0909
6A/6B	.0896
7A/7B	.0886
8A/8B	.0879
9A/9B	.0874
10A/10B	.0869
20A/20B	.0851
50A/50B	.0840
100A/100B	.0837
200A/200B	.0835

Overall, the optimum sequence, in this example, improves throughput over other sequences by as much as 20%. The optimum throughput can be maintained with slower sequence changes only by increasing the number of jobs in the system above the present value of ($m = 4$). Increases in (m) are undesirable as they result in the increase in flow times. However, when sequences possess extended runs, i.e., 10A/10B, it becomes difficult to maintain the throughput level obtained by the quicker changes due to the extremely high levels of WIP.

In the example, the value of m^* will have to be increased to maintain throughput when throughput drops as occurred for the sequence of 3/3. Thus, assuming that there is a product with a bottleneck not at the cbn, less frequent changes in the sequence will require a larger m^* for achieving optimum throughput. This larger amount of WIP prevents the cbn from becoming idle.

However, having an increment in m^* is in contradiction to the basic premise of the CONWIP system of maintaining a lower WIP level. Furthermore, it will have the undesirable consequence of increasing the average flow time. Thus, the example confirms the conjecture that the quickest changes in sequencing are preferable for achieving maximum throughput with the lowest possible amount of WIP (m^*).

3.4.2 Multi-Product Mix

The potential to positively affect throughput can be further expanded to larger product mixes. To reiterate, improving the upper bound on throughput through sequencing requires that there be at least one product in the sequence with its bottleneck station not at the cbn. Thus, the upper bound of throughput is dependent on the sequence. The upper limit on the upper bound of throughput applies to mixed sequences. The upper limit on the upper bound of throughput for a (k) product mix with demands d(i) is as follows:

$$(3.4.2.1) \quad \Theta \leq \frac{\sum_{i=1}^k d(i)}{\sum_{i=1}^k d(i) * tcb(i)}$$

The lower limit on the upper bound for throughput is found by averaging the optimum throughput rates from long runs of individual products as follows:

$$(3.4.2.2) \quad \Theta \leq \frac{\sum_{i=1}^k d(i)}{\sum_{i=1}^k d(i) * tb(i)}$$

Also, the following inequality must hold:

$$(3.4.2.3) \quad tcb(i) \leq tb(i) \quad i = 1, \dots, k$$

If each product has its bottleneck at the cbn, i.e. $tcb(i) = tb(i)$, then the upper and lower limits on the upper bound of throughput will be equivalent. This can easily be verified by equations (3.4.2.1, 3.4.2.2).

For our purposes, jobs will now be grouped in the sequence such that the jobs included in the group are produced in ratios approximately equal to the ratios of their demands. The jobs included in the group will then be assumed to be one of a kind. For example, the three jobs A, B, and C with a 3:2:1 demand ratio will now be considered as one group of six jobs A,B,C,D,E, and F. The number of jobs in the product mix will increase from $k = 3$ to $k' = 6$ and all $d(i) = 1$. The upper limit on the upper bound which applies to a mixed sequence is now defined as follows:

$$(3.4.2.4) \quad \Theta \leq \frac{k'}{\sum_{i=1}^{k'} tcb(i)} = \frac{k'}{tcbn}$$

The lower limit on the upper bound which applies to the unmixed sequence is defined as follows:

$$(3.4.2.5) \quad \Theta \leq \frac{k'}{\sum_{i=1}^{k'} tb(i)}$$

The requirement to achieve the optimum throughput level found in equation (3.4.2.4) is to maintain 100% utilization at the cbn. To obtain 100% utilization of the cbn, an extension of the results of the two product case dictates that continuously changing products in the sequence will be most desirable. Products should be repeated in the sequence with the least possible frequency in order to most easily force WIP to build at the cbn. For example, for a five product mix, the following sequence type will most readily allow for maximum throughput levels to be obtained with the lowest m^* .

ABCDE ABCDE

The most significant advantage of repetitive sequences is that they will greatly simplify the task of sequencing the jobs. The optimization of a non-repetitive sequence will require significantly more effort than a repetitive sequence. Furthermore, a non-repetitive sequence has a much greater likelihood of forming bottlenecks away from the cbn which jeopardizes optimum throughput.

Only products with their own bottleneck station at the cbn should be allowed to be repeated in the sequence. However, repeating the products with a bottleneck at the cbn provides an opportunity for a new bottleneck to be created by the remaining jobs in the sequence. This will occur if the remaining products have a maximum amount of processing time that is not at the cbn station. To illustrate this, if product A has its bottleneck at the cbn and B,C, D, and E have a maximum amount of processing time away from the cbn, then the sequence AA..A BCDE...BCDE forces idle time at the cbn after the A's are completed and the BCDE sequence is in process. A drop in throughput will result.

Repeating jobs with bottlenecks away from the cbn is obviously detrimental. Also, repeating jobs with a bottleneck at the cbn may also jeopardize optimum throughput. Thus, the safest policy to obtain maximum throughput with the lowest WIP level is not to violate the sequence of type (ABCDEABCDE.....).

3.5 Flow times

Flow time or (cycle time) averages may be improved from the one product averages when products are mixed in the backlog sequence. This is evident from Little's law below. If two products are mixed and the throughput increases while m^* does not increase, then it is evident that average cycle time would have to be reduced. However, it cannot be assumed that m^* will not increase when the products are mixed.

$$(3.5.1) \quad \Theta = \frac{m^*}{\bar{C}} \quad \text{or}$$

$$(3.5.2) \quad \bar{C} = \frac{m^*}{\Theta}$$

An analysis to determine the total impact on average cycle times when products are mixed in larger groups requires knowledge of m^* for the sequence mix and m^* for the unmixed sequence. When sequencing larger groups, there will be a greater amount of variability in the system which creates a greater chance for bottlenecks to form away from the cbn. These queues will lead to increment in m^* thereby increasing the average cycle time.

To determine if a mixing sequence gives rise to an overall improvement in the average cycle time, it must be directly compared with the unmixed sequence. Expression (3.5.3) below gives the average cycle time for the unmixed sequence while expression (3.5.4) expresses the same for the mixed sequence.

$$(3.5.3) \quad \overline{C_u} = \frac{m^*(1) \cdot tb(1) + \dots + m^*(k') \cdot tb(k')}{k'}$$

Where $m^*(i)$ and $tb(i)$ are the optimal WIP level and bottleneck processing time for product (i) respectively.

$$(3.5.4) \quad \overline{C_m} = \frac{(m^*) \cdot tcbn}{k'}$$

The tendency for m^* to increase when mixing jobs will increase the average cycle time. The bottleneck processing time ($tcbn$) which must be less than or equal to the sum of the individual bottleneck processing times ($tb(i)$) will reduce the average cycle time when m^* does not increase. Therefore, which ever one of these factors dominate will determine whether or not the average cycle time increases or decreases.

3.6 m^* in the Multi-product Case

In contrast to the one product case where m^* can be determined directly using equation (3.1.1), m^* in the multi-product case cannot be found in this manner. In the multi-product case, m^* is dependent on a value which will be specified as the return time to the bottleneck $tr(i,i')$. This return time consists of the total amount of time from the event job (i) leaves the bottleneck until the time job (i') that it triggers to enter the system, reaches the bottleneck. The pairs of (i,i') will be maintained for every cycle of the jobs when the sequence is repetitive. The determination of these pairs will be discussed in section (4.1).

To reiterate, the optimum WIP level m^* must maintain 100% utilization of the cbn. This requires the system to maintain sufficient WIP (m) so that each pair (i, i') can return to the cbn before the cbn becomes idle. The return times $tr(i, i')$ will be affected by the formation of queues outside the cbn and by the type of job that enters upon the completion of another. Therefore, m^* will also be affected by these factors.

The time for a job to return to the bottleneck $tr(i, i')$ consists of three components as depicted in equation (3.6.1) below. The first term consists of the total amount of processing time in front of the bottleneck station for the entering job. The second term consists of the total amount of processing time after the bottleneck for the leaving job. The third term is the time spent in queue for the entering and leaving jobs.

$$(3.6.1) \quad tr(i,i') = \sum_{j=1}^{B-1} t(i',j) + \sum_{j=B+1}^N t(i,j) + \sum_{j=1}^N tq(j)$$

Next, equation (3.6.2) describes the relationship between the return times and m^* . It is assumed that the processing time for all products at the cbn, i.e. $tcb(i)$, is equivalent. The value of one in equation (3.6.2) is a result of the job itself that determines the return time. The match of the entering and leaving job (i, i') with the maximum return time of all matches is utilized in equation (3.6.2) to determine m^* .

$$(3.6.2) \quad m^* = 1 + \frac{\max(tr(i, i'))}{tcb(i)}$$

The first term of (1) in the equation (3.6.2) is for the job (i, i') itself. The second term is the amount of work that must be in the system so that the job (i, i') returns to the cbn before it becomes idle. The factors that affect (m^*) in the multi-product case can now be deciphered from equations (3.6.1) and (3.6.2). Similar to that in the one product case, when the cbn is dominant, relative to the other stations, it will reduce the value of m^* needed to optimize throughput. In other words, as the cbn processing time $tcb(i)$ in equation (3.6.2) is increased and non-bottlenecks are held constant, it will reduce m^* .

It is obvious from equation (3.6.2) that increasing the maximum return time $tr(i, i')$ will increase m^* in the multi-product case. This return time can be increased by increasing the non-bottleneck processing times. It can also be increased by queuing outside of the cbn which results from variance in the processing times. In addition, it is worth noting that the upper limit of m^* in the multi-product case can exceed the one product limit of the number of stations in the system.

Operating below m^* in the multi-product case will result in less than optimum throughput. This lower level of throughput can be attributed to the fact that the cbn resource doesn't have 100% utilization when operating below m^* . The sequencing policy when operating below m^* will also affect the throughput of the system. The analysis of this scenario is not of great interest here since the primary concern is to determine a sequencing policy when operating at m^* to maintain optimum throughput.

Operating above m^* has the effect of increasing queuing at the multi-product cbn for quick changes in sequencing. This larger bottleneck buildup at the cbn will delay the drop in throughput as sequencing changes are slowed. However, the detrimental effects of higher WIP and longer flow times will be incurred. Thus, this case is also of lesser interest since it is a high priority to operate with lower values of m^* .

Obtaining m^* from equations (3.6.1) and (3.6.2) would require determining the return times $tr(i,i')$ of all the jobs. The amount of time a job spends in queues outside of the cbn station affects the value of the return times and thus m^* . Another factor is the sequence in which the jobs enter the system upon the completion of other jobs. After determining the effects of queuing and the job sequence in which jobs enter the system, the return times $tr(i,i')$ can be found. Then, m^* can be obtained exhaustively from equation (3.6.2).

It should now be apparent that the exact sequence of the products, for example ABCDE vs. EDCBA, will affect the interaction of the products in the system and consequently the return times which determine m^* . The number of possible sequences of the jobs is $k!$. Each sequence will have its own distinct optimal WIP level m^* . The

optimum sequence will have the lowest value of m^* which is designated as m^{**} . The possible scenarios range from all sequences having the same m^* to where only one sequence will possess the lowest m^* .

3.7 Effects of Grouping

Previously, it was assumed that all jobs will be processed together in a single group. However, it may be desired to utilize more than one group due to the availability of multiple lines, dependent set-up times, too large a number of different products or different demand periods. The division of jobs into groups can have a significant effect on the performance of the system.

The primary objective in this research is to optimize throughput. The upper limit of throughput for one grouping of jobs is given by equation (3.4.2.1). When the group is subdivided, it will either maintain this upper limit or reduce it. Dividing the jobs can never increase this upper bound of throughput.

It is anticipated that the optimum throughput will become more difficult to maintain for a fixed WIP level as the size of the group is increased. To reiterate, m^* is determined by the maximum return time from section (3.6). Increasing the number of jobs in a grouping will increase the potential for a higher maximum return time. Thus, a larger grouping has the potential to increase m^* and average cycle time which is undesirable.

A single grouping has the advantage of being able to obtain the upper limit of throughput under any conditions. Smaller, divided groupings may have the advantage of lower optimal WIP levels and average cycle times while maintaining optimum throughput. To maintain optimum throughput, all groupings must consist of products with cbns at the station that is the cbn for all products. This will allow the highest level of throughput to be achieved while reducing the potential for increasing m^* and mean flow times.

The following analysis illustrates how throughput will be affected if the products are divided. Let A and B be the number of products in the two groupings. The variable t_{cbn} consists of the total amount of processing time at the cbn for one grouping. For the divided groups A and B, the variables $t_{cbn}(a)$ and $t_{cbn}(b)$ are the total amounts of processing time at their cbns, which are $cbn(a)$ and $cbn(b)$ respectively. The processing time for all jobs at the cbn for one grouping is the same. Assuming that groupings A and B have their cbns at the cbn for one grouping, the throughput for the undivided group can be expressed by the following,

$$(3.7.1) \quad \Theta = \frac{A + B}{t_{cbn}}$$

The average throughput level for the two divided groupings becomes,

$$(3.7.2) \quad \Theta' = \frac{A}{A + B} \left(\frac{A}{t_{cbn}(a)} \right) + \frac{B}{A + B} \left(\frac{B}{t_{cbn}(b)} \right)$$

If $cbn(a)$ and $cbn(b)$ are at the cbn for one grouping, then

$$(3.7.3,4) \quad t_{cbn}(a) = \left(\frac{A}{A + B} \right) t_{cbn} \quad t_{cbn}(b) = \left(\frac{B}{A + B} \right) t_{cbn}$$

From equations (3.7.1,2,3,4) the following must be true,

$$\Theta = \Theta'$$

Therefore, dividing the products into smaller groups did not reduce the throughput relative to the original undivided group. To reiterate, this can be attributed to the groups being divided in a manner that did not create a new cbn away from the original cbn.

However, it may not always be feasible to divide the groups in a manner that does not create new cbns away from the original cbn. If one of the divided groups possesses a cbn that is not at the original cbn then $t_{cb}(a)$ or $t_{cb}(b)$ must be increased. This must reduce throughput ($\Theta' < \Theta$) by (3.7.1,2).

Example (3.7.1),

$$t(i,j) = \begin{bmatrix} 15 & 10 & 5 \\ 15 & 10 & 5 \\ 3 & 10 & 5 \\ 3 & 10 & 5 \end{bmatrix}$$

The above matrix contains the processing times for job (i) at station (j). The cbn is station two with total processing time of 40 for the four jobs. The optimum throughput for the system when the jobs are all grouped together is 1/10. When jobs are divided into the groups of (1,2) and (3,4), the optimum throughput rate becomes,

$$\Theta = \frac{4}{30+20} = \frac{4}{50}$$

Thus, the optimum throughput rate has decreased by 20%. This is a result of the cbn for jobs (1,2) being at station one which is not the cbn for one grouping, i.e., station two. This cbn has total processing time of 15 + 15 = 30. The cbn for jobs (3,4) remained at station two with total processing time of 10 + 10 = 20. Next, consider dividing the jobs into the groups of (1,4) and (2,3). The optimum throughput rate will be,

$$\Theta = \frac{4}{20+20} = \frac{1}{10}$$

The optimum throughput rate did not decrease, since the cbns for both groups are at the cbn for one grouping, i.e., station two. Therefore, the example illustrates that the optimum throughput rate may or may not decrease depending upon which jobs are divided into which groups. Again, the determining factor is whether or not the cbns for the divided groups are at the cbn for one grouping.

4.0 LOWER BOUND DETERMINATION OF m^*

4.1 Lower Bound for m^*

The purpose for obtaining a lower bound for m^* is to be able to more quickly find m^* . All values of m below the lower bound for m^* can be disregarded as possible values for m^* . This increases the efficiency in finding m^* . In section 4.2, a recursive algorithm will be presented to determine if a given (m) is in fact m^* for a given sequence. The algorithm will utilize the lower bound as its starting point.

The concept of entering and leaving jobs and their potential to affect m^* was first introduced in section (3.6). To reiterate, a job is designated as a leaving job after it leaves the cbn and an entering job when it enters the system as a result of the leaving job. In a repetitive multi-product sequence, (m) and the number of jobs in the sequence (k) will determine which job enters the system when a certain job leaves. Assuming the sequence to be (1, 2, 3,... k , 1, 2, 3,... k ...), for job (i) leaving the system the corresponding job (i') entering the system is given by

$$(4.1.1) \quad i' = \text{Mod} \frac{\text{Mod} \frac{m}{k} + i}{k}$$

Equation (4.1.1) requires that if $\text{Mod} = 0$, it is replaced by k . An illustration of leaving and entering jobs is given in figure (4.1.1). For this example, $m = 3$ and $k = 5$. The (i, i') pairs are (1,4), (2,5), (3,1), (4,2) and (5,3). These pairs can be determined with equation (4.1.1).

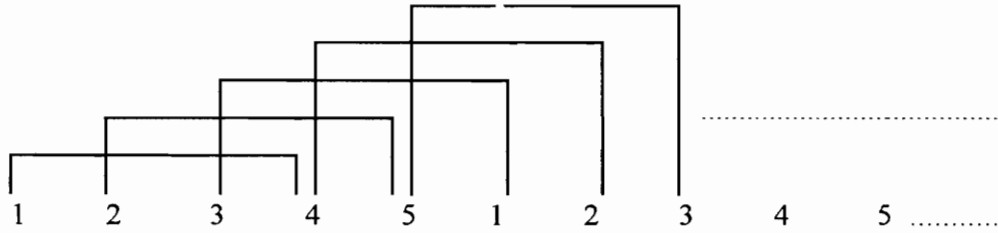


Figure (4.1.1) Illustration of Entering and Leaving Job Pairs

The relationship between entering and leaving jobs will exist during every cycle of the jobs due to the sequence being repetitive. Thus, the values of (m) and (k) determine which leaving job will always trigger the entrance of the identical job for every cycle of the sequence.

The knowledge of this characteristic can now be exploited in order to obtain a lower bound. A lower bound can be obtained from the set of pairs (i, i') by assuming that queuing does not occur outside of the cbn. Thus, the term that represents queuing in the return time equation (3.6.1) can be deleted to form equation (4.1.2) below. The return time from (4.1.2) provides the absolute minimum for the return time $tr(i, i')$.

$$(4.1.2) \quad tr(i, i') = \sum_{j=1}^{B-1} t(i', j) + \sum_{j=B+1}^N t(i, j)$$

The maximum return time $tr(i, i')$ from equation (4.1.2) for the set (i, i') can then be utilized in equation (4.1.3) to determine a lower bound for m^* for that set of pairs.

$$(4.1.3) \quad m^* \geq 1 + \frac{\max(tr(i, i'))}{tbn}$$

Two special cases occur depending upon the location of the bottleneck being at the first or the last station. If the bottleneck is at the first station then only leaving jobs will exist as per the definition of a leaving job. Similarly, if the bottleneck is at the last station then only entering jobs will exist. This eliminates the need for matching entering and leaving jobs. A lower bound of m^* is found using expression (4.1.3). With the assumption of no queuing in the system outside of the bottleneck, the return times $tr(i)$ for these two special cases will simply be the processing times of the products at stations other than the bottleneck. The maximum of these return times determines a lower bound for m^* from (4.1.3).

Example 4.1.1,

The sequence considered is (1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6,) with $k=6$. The WIP level is $m=4$. The processing times are contained in Table (4.1.1).

Table (4.1.1) Processing Times $t(i,j)$ for the Lower Bound Example

job \ station	station 1	station 2	station 3 *BN*	station 4	station 5
job 1	7.0	11.0	10.0	4.0	5.0
job 2	10.0	12.0	10.0	9.0	8.0
job 3	5.0	6.0	10.0	11.0	12.0
job 4	7.0	8.0	10.0	6.0	7.0
job 5	13.0	14.0	10.0	8.0	11.0
job 6	15.0	4.0	10.0	13.0	13.0

Table (4.1.2) Sum of Processing Times for Leaving and Entering Jobs

JOB #	ENTR (i)	LEAV (i)
1	18.0	9.0
2	22.0	17.0
3	11.0	23.0
4	15.0	13.0
5	27.0	19.0
6	19.0	26.0

In Table (4.1.2) above, the total processing time associated with the entering jobs (i) is in the first column and for the leaving jobs (i) in the second column. The entering jobs' processing time is the sum of the processing time on stations 1 and 2 before the bottleneck. The leaving jobs' processing time is the sum of the processing times from stations 4 and 5 after the bottleneck.

The following is the set of pairs of leaving and entering jobs (i,i') that are determined by expression (4.1.1).

$$(i,i') = \{(1, 5), (2, 6), (3, 1), (4, 2), (5, 3), (6, 4)\}$$

The return times $tr(i,i')$ can then be obtained from the values in table (4.1.2).

$$tr(1,5) = 9 + 27 = 36 \qquad tr(2,6) = 17 + 19 = 36$$

$$tr(3,1) = 23 + 18 = 41 \qquad tr(4,2) = 13 + 22 = 35$$

$$tr(5,3) = 19 + 11 = 30 \qquad tr(6,4) = 26 + 15 = 41$$

$$tr(i,i') = (36, 41, 30, 36, 35, 41)$$

$$\max tr(i,i') = 41$$

The lower bound can then be obtained by equation (4.1.3).

$$m^* \geq 1 + \frac{41}{10} \geq 6$$

There are three cases that exist that place their own unique restrictions on the (i, i') matches for certain values of (m) in relation to (k) .

- (1) (m) is not a multiple of (k) or odd multiple of $(k/2)$
- (2) (m) is an odd multiple of $(k/2)$
- (3) (m) is a multiple of (k)

For case (3) where (m) is a multiple of (k) , the leaving job (i) must be the same as the entering job (i') or $(i = i')$. For case (2), a leaving job (i) that is matched with an entering job (i') requires that leaving job (i') be matched with entering job (i) , i.e. $\{(i, i'), (i', i)\}$. The reverse of this restriction is true for Case (1) where the (i, i') match eliminates the match of (i', i) . Expression (4.1.1) can be utilized to confirm that these cases restrict the matches of leaving and entering jobs in the manners just described.

Example 4.1.2.

Sequence: $i = (1,2,3,4)$, $k=4$

$m = \text{multiple of } k: 4,8,12 \dots\dots$	$m = \text{odd multiple of } k/2: 2, 6, 10 \dots$
$i' = (1,2,3,4)$	$i' = (3,4,1,2)$
$(i,i') = (1,1), (2,2), (3,3), (4,4)$	$(i,i') = (1,3), (3,1), (2,4), (4,2)$

Given the set of entering and leaving jobs (i, i') , the pair with the maximum value for the return time $tr(i, i')$ determines a lower bound for that set as equations (4.1.2,3) dictate. Procedure (I), for case one, will be presented next to determine the (i, i') set that satisfies the objective of $\min(\max tr(i, i'))$ over all feasible sets of (i, i') for case one.

A constraint of the set (i, i') for all three cases is that each job (i) can only be matched with one other job (i') during every cycle and vice versa. This results from the sequence being repetitive and m being constant. During the process of elimination of the values in the $tr(i, i')$ matrix, this translates into the fact that any row or column must eventually contain exactly one element. The following is the problem formulation for Case (1), where $X_{ij} = 1$ if a $tr(i, i')$ element is in the optimal set:

Objective: $\min(\max(tr(i, i')))$

Subject to: $\sum_i X_{ij} = 1 \quad i = (1, \dots, k)$

$\sum_j X_{ij} = 1 \quad j = (1, \dots, k)$

$X_{ij} + X_{ji} \leq 1$

$X_{ij} = (0,1)$

In Procedure (I), the $tr(i, i')$ matrix is defined which will represent the return times to the cbn for each feasible pair of entering and leaving jobs (i, i') . The values in the matrix are then eliminated in a manner that results in a feasible set (i, i') that satisfies the objective of $\min(\max tr(i, i'))$. Then, this value of $tr(i, i')$ is utilized in equation (4.1.3) to obtain a lower bound of m^* .

Procedure (II) is then presented for case two, where (m) is an odd multiple of $(k/2)$. When (k) is odd, (m) cannot be a multiple of $(k/2)$ and therefore Procedure (II) can be deleted from the analysis. Procedure (II) follows similar steps to Procedure (I), but requires some additional considerations. Procedure (III) is then presented for case three, where (m) is a multiple of (k) .

Lower Bound Procedure One (For Case One Conditions):

(1.1) Define the matrix $tr(i, i')$ to be the sum of processing times on stations upstream of the bottleneck of job (i') designated as an entering job and the sum of processing times downstream of the bottleneck of job (i) designated as a leaving job. Exclude all matches of job (i) and job (i') when $i = i'$. In other words, the diagonal is voided.

(1.2) Eliminate the highest value in the matrix of $tr(i, i')$ values and denote it as $X(1)$. Continue eliminating the $tr(i, i')$ values from the matrix in descending order until only one element remains in a row or column. Ties can be broken arbitrarily. Denote this element as $N(1)$ and the deleted elements as the set $(X(1), \dots, X(n))$. The argument for $N(1)$'s selection is as follows:

If $N(1)$ is the last remaining element in its column, then the other elements in the column must be from $(X(1), \dots, X(n))$. From step 1.2, the elements were eliminated in descending order and thus must satisfy

$$N(1) < (X(1), \dots, X(n))$$

Any element from $(X(1), \dots, X(n))$ will increase the lower bound (by equation (4.1.3)) which violates the objective. Similarly, if $N(1)$ is the last remaining element in its row then all other elements in that row must be from $(X(1), \dots, X(n))$ and hence again $N(1)$ must be part of the optimum set.

(1.3a) Only one element can exist in each row and column in the final matrix containing the set of matches. Again, this is a consequence of holding m constant and utilizing repetitive sequencing. The above requirement results in the following step of the procedure. Eliminate the remaining elements in $N(1)$'s row or column. The argument for this step in the procedure is as follows:

Suppose $N(1)$ was initially the last remaining element in its column. Then the other remaining elements in its row must be eliminated. The choice of one of the removed elements in the row of $N(1)$ instead of $N(1)$ will eliminate the choice of $N(1)$ in its row. The choice of $N(1)$ eliminated as a choice in its row also eliminates its choice in its column. Since each column must have one choice, the remaining choice must come from the set $(X(1), \dots, X(n))$. The maximum would then now come from $(X(1), \dots, X(n))$ instead of being $N(1)$. Consequently, the maximum value would be increased from $N(1)$ to a member of $(X(1), \dots, X(n))$. This will increase the lower bound by equation (4.1.3) which violates the objective. Hence, if $N(1)$ is optimum in its column then it must also be optimum in its row and vice versa.

(1.3b) Another property of this particular case is that if $N(1)$ is chosen at $tr(i, i')$ then a $N(i)$ cannot exist at $tr(i', i)$. Thus, when $tr(i, i')$ is chosen $tr(i', i)$ must be eliminated if it has not been already eliminated.

(1.4) At this point in the procedure, the elements removed from the matrix are those in the set $(X(1), \dots, X(n))$ and those removed in the row or column containing $N(1)$. Again, begin eliminating the remaining $\text{tr}(i, i)$'s in descending order until only one element remains in a row or column. Denote this optimal element as $N(2)$ and the elements removed in this step as $(X(n+1), \dots, X(m))$. The argument for the selection of $N(2)$ is as follows:

Since they were removed in descending order, we have

$$(X(n+1), \dots, X(m)) < (X(1), \dots, X(n))$$

and

$$N(2) < (X(n+1), \dots, X(m))$$

Given $N(2)$ was the last remaining element in its column, the eliminated elements in its column can be from $(X(1), \dots, X(n))$, $(X(n+1), \dots, X(m))$ or elements removed by the choice of $N(1)$. For the first two cases,

$$N(2) < (X(1), \dots, X(n)) \text{ and } (X(n+1), \dots, X(m))$$

In the third case choosing an element that was eliminated by the choice of $N(1)$ will eliminate the choice of $N(1)$. Eliminating the choice of $N(1)$ will force a choice from $(X(1), \dots, X(n))$ from steps (1.2, 1.3).

Thus, the choice of $N(2)$ is optimal in its column. From step (1.3a), it must also be optimal in its row. Similarly, the argument can be made if the element was the last remaining element in its row then it will be optimal in its row and column.

- (1.5) Continue eliminating the highest remaining $tr(i,i')$ until only one element remains in a row or column to obtain all $N(i)$ s. The following $N(i)$ s can be shown to be optimal by reapplying the previous steps .
- (1.6) The maximum of the set $N(i)$ possesses the minimum of the maximum return times $tr(i,i')$ of all feasible sets. This return time is now implemented to find a lower bound for m^* using expression (4.1.3). This value is rounded up to the next highest integer that is not a multiple of the number of different jobs k or $k/2$.

Example (4.1.1) Continued.

Example (4.1.1) is continued to illustrate Lower Bound Procedure I. Table (4.1.3) below is for matrix $tr(i, i')$. The elements that are shaded are those which have been eliminated.

Table (4.1.3) Illustration of Lower Bound Procedure I, Step (1.1): determine the $tr(i, i')$ matrix values, which are the sum of processing time of job (i) after the bottleneck and the processing time of job (i') before the bottleneck

	31	20	24	36	28
35		28	32	44	36
41	45		38	50	42
31	35	24		40	32
37	41	30	34		38
44	48	37	41	53	

The methodology of the Lower Bound Procedure (I) is initiated in the following Table (4.1.4), elements are eliminated until element $tr(6,3)$ is the last element remaining in its row. This element is then selected in that row and the elements remaining in its column are eliminated in Table (4.1.5).

Table (4.1.4) Illustration of the Lower Bound Procedure I, Step (1.2): eliminate $tr(i, i')$ in descending order until only one element remains in any row or column and select that element designated as $N(1)$ at $tr(6,3)$

	31	20	24	36	28
35		28	32	44	36
41	45		38	50	42
31	35	24		40	32
37	41	30	34		38
44	48	*37*	41	53	

Table (4.1.5) Illustration of the Lower Bound Procedure I, Step (1.3): eliminate any remaining elements in the selected element's row or column

	31	20	24	36	28
35		28	32	44	36
41	45		38	50	42
31	35	24		40	32
37	41	30	34		38
44	48	*37*	41	53	

The process continues with the selection of $tr(3,4)$ in its column and row.

Table (4.1.6) Illustration of the Lower Bound Procedure I, Step (1.4): eliminate $tr(i, i')$ in descending order until a second element is selected $N(2)$ at $tr(3,4)$.

	31	20	24	36	28
35		28	32	44	36
41	45		*38*	50	42
31	35	24		40	32
37	41	30	34		38
44	48	*37*	41	53	

Again, continue eliminating and selecting elements until all $N(I)$ s are found.

Table (4.1.7) Illustration of the Lower Bound Procedure I, Step (1.5): eliminate $tr(i, i')$ in descending order, select elements when necessary, eliminate any remaining elements in the selected element's row or column

	31	20	24	*36*	28
35		28	32	44	36
41	45		*38*	50	42
31	*35*	24		40	32
37	41	30	34		*38*
44	48	*37*	41	53	

The set of entering and leaving jobs (i, i') that are selected are:

$$(1, 5), (2, 1), (3, 4), (4, 2), (5, 6) \text{ and } (6, 3)$$

The maximum return time for the set,

$$tr(1, 5) = 36 \quad tr(2, 1) = 35$$

$$tr(3, 4) = 38 \quad tr(4, 2) = 35$$

$$tr(5, 6) = 38 \quad tr(6, 3) = 37$$

$$\max tr(i, i') = \max (36, 35, 38, 35, 38, 37) = 38$$

From equation (4.1.3), a lower bound of m^* is as follows:

$$m_1^* \geq 1 + (38 / 10) = 4.8 = 5.0$$

Procedure (II) is presented next for the case of odd multiples of $(k/2)$. When (k) is odd, (m) cannot be a multiple of $(k/2)$ and therefore Procedure (II) can be deleted from the analysis. Procedure (II) follows steps similar to those of Procedure (I), but requires some additional considerations. The problem formulation for case (2) is as follows:

Objective: $\min(\max(\text{tr}(i,i)))$

Subject to: $\sum_i X_{ij} = 1 \quad i = (1, \dots, k)$

$\sum_j X_{ij} = 1 \quad j = (1, \dots, k)$

$X_{ij} + X_{ji} \neq 1$

$X_{ij} = (0,1)$

Lower Bound Procedure II (For Conditions of Case Two):

(2.1) For case two, if an element is selected at $\text{tr}(i,i)$ then it must also be selected at $\text{tr}(i,i)$. This can be proven given the conditions for this case with expression (4.1.1). Also, it follows that elimination of $\text{tr}(i,i)$ will eliminate $\text{tr}(i,i)$. The $\text{tr}(i,i)$ matrix will now be altered in the following way to facilitate determining the optimal set for this case.

$$\text{tr}(i,i) = \text{Max}(\text{tr}(i,i), \text{tr}(i,i))$$

(2.2) Eliminate the pairs of elements $\text{tr}(i,i)$ and $\text{tr}(i,i)$ in descending order. After each pair is eliminated, determine if a complete set of pairs is still feasible from the remaining pairs, i.e., the remaining set of pairs must contain at least one element in every row and column. If a set is still feasible, then eliminate the next highest pair and then test for feasibility again. When a complete set is no longer feasible, proceed to step (2.3).

(2.3) When a complete set is no longer feasible, re-include the last pair of $tr(i, i')$ that was eliminated. The pair must be part of the final set of pairs since eliminating it destroys feasibility. The pair will possess a maximum value relative to the remaining pairs in the matrix. Therefore, it must provide the maximum value from the set of pairs. Since the maximum value from the set of pairs determines the lower bound, the exact set of pairs need not be determined. It need only be confirmed that a feasible set of pairs exists, that includes the pair and pairs from the remaining elements. The procedure can now proceed to Step (2.4).

(2.4) Apply expression (4.1.3) with the new maximum return time. Find the lower bound for this case by rounding up to the next highest odd multiple of $(k/2)$.

Example 4.1.1 Continued,

The following is the matrix for Procedure II for example 4.1.1,

Table (4.1.8) Illustration of the Lower Bound Procedure II, Step (2.1): define the matrix $tr(i, i') = \max [tr(i, i'), tr(i', i)]$

	35	41	31	37	44
35		45	35	44	48
41	45		38	50	42
31	35	38		40	41
37	44	50	40		53
44	48	42	41	53	

Table (4.1.9) Illustration of the Lower Bound Procedure II, Step (2.2): eliminate the pairs of $[tr(i, i'), tr(i', i)]$ in descending order

	35	41	31	37	44
35		45	35	44	48
41	45		38	50	42
31	35	38		40	41
37	44	50	40		53
44	48	42	41	53	

After the pair of $tr(6,3)$ and $tr(3,6)$ is eliminated, a feasible set no longer exists. The last remaining element in its column $tr(4,6)$ must be selected and thus also $tr(6,4)$. This can be seen in table (4.1.9). Selecting this pair forces the selection of $tr(5,1)$ and $tr(1,5)$. The only elements that could be part of a feasible set with these elements would be $tr(2,3)$ and $tr(3,2)$. These elements were previously eliminated with values of 45. This can be seen in table (4.1.10) below. The lightly shaded elements in table (4.1.10) are elements that were eliminated by the selections of $tr(5,1)$, $tr(1,5)$, $tr(4,6)$ and $tr(6,4)$.

Table (4.1.10) Illustration of the Lower Bound Procedure II, Step (2.2): determine that a feasible set no longer exists

	35	41	31	* 37 *	44
35		45	35	44	48
41	45		38	50	42
31	35	38		40	* 41 *
* 37 *	44	50	40		53
44	48	42	* 41 *	53	

In the following table (4.1.11), the pair $tr'(3,6)$ and $tr'(6,3)$ with values equal to 42 is re-included in the set as dictated by step (2.3). Then, the optimal feasible set is found to contain the pairs (1,2), (2,1), (4,5), (5,4), (3,6), and (6,3).

Table (4.1.11) Illustration of the Lower Bound Procedure II, Step (2.3): re-include the last pair that was eliminated and determine the feasible set of $tr'(i, i')$

	* 35 *	41	31	37	44
* 35 *		45	35	44	48
41	45		38	50	*42*
31	35	38		*40*	41
37	44	50	*40*		53
44	48	*42*	41	53	

The lower bound when (m) is an odd multiple of $(k/2)$, i.e., $m = (3, 9, 15, 21, \dots)$ is determined below with equation (4.1.3). The value obtained from equation (4.1.3) must be rounded up to the next highest value of $k/2$.

$$m_2^* = 1 + 42/10 = 5.2 = 9$$

Procedure III for the case when (m) is a multiple of (k) will be presented next. It does not require analysis with a matrix since the leaving and entering jobs are the same. It requires determining the maximum amount of processing time off of the bottleneck of all products.

Lower Bound Procedure III (For Case Three):

(3.1) The final case is where (m) is a multiple of (k). This forces the entering and leaving jobs to be the same for a repetitive sequence. The maximum return time is found with the following where $i=i'$

$$tr = \max \left(\sum_{j=1}^{B-1} t(i, j) + \sum_{j=B+1}^N t(i, j) \right)$$

(3.2) The return time from step (3.1) will be utilized in expression (4.1.3) and the lower bound for m^* for this case will be rounded up to the next highest integer multiple of (k).

Finally, the lower bound can be found by taking the minimum lower bound from the steps (1.6, 2.4, 3.2).

Example 4.1.1 Continued.

For Procedure III when m is a multiple of k, i.e. $m = (6, 12, 18, 24, \dots)$ is found with the maximum value from the table below,

Table (4.1.12) Illustration of the Lower Bound Procedure III, Step (3.1)

JOB #	ENT + LEAV
1	27
2	39
3	34
4	28
5	46
6	45

Finding (m^*) when it is a multiple of (k),

$$m_3^* = 1 + 46/10 = 5.6 = 6$$

The lower bound for m^* is then determined by taking the minimum of the values found in the three procedures,

$$m^* \geq \min(5,9,6) \geq 5$$

The following analysis is based upon Table(4.1.13) and Figure(4.1.1). The analysis is given in order to provide insight into the relative importance of each of the three cases. It is expected that Procedures (II, III) will almost always yield a greater lower bound for m^* than Procedure (I) as occurred in example (4.1.1). This is due to the fact that the number of possible sets of matches of $tr(i, i')$ and the possible candidates for m^* lower bound are less for Procedures (II, III).

It can be seen from the analysis for the 4 job system that Procedure (I) possesses six possible sets, Procedure (II) three sets and Procedure (III) only one set. This alone leads to a greater expectation of the lower bound coming from Procedure (I). In addition, Procedure (I) has a 2:1 advantage in terms of the possible values for m^* lower bound. These advantages for Procedure (I) increase as the number of different jobs (k) increase.

However, when the value for the lower bound of m^* is in close proximity to a multiple of k or an odd multiple of $k/2$, Procedures (II,III) will possess a greater chance of providing a lesser lower bound for m^* than Procedure (I). This can occur due to the sets of matches for Procedures (I), (II) and (III) being mutually exclusive and the constraints on the values of m^* lower bound. Thus, the actual lower bound will most likely come from Procedure (I), but Procedure (II,III) are also necessary. This is especially true for Procedure (I) as (k) becomes large.

Table (4.1.13) Standard 4 x 4 tr(i, i') Matrix

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

Figure (4.1.1) The possible sets of tr(i,i') values and m* lower bound values for

Procedures (I, II, III) are listed below.

<u>Procedure I</u>	<u>Procedure II</u>	<u>Procedure III</u>
(2,1), (3,2), (4,3), (1,4)	(1,2), (2,1), (3,4), (4,3)	(1,1), (2,2), (3,3), (4,4)
(2,1), (4,2), (1,3), (3,4)	(1,3), (3,1), (2,4), (4,2)	
(3,1), (1,2), (4,3), (2,4)	(4,1), (1,4), (3,2), (2,3)	
(3,1), (4,2), (2,3), (1,4)		
(4,1), (3,2), (1,3), (2,4)		
(4,1), (1,2), (2,3), (3,4)		
<u>m* lower bound</u>	<u>m* lower bound</u>	<u>m* lower bound</u>
(1, 3, 5, 7,)	(2, 6, 10, 14,)	(4, 8, 12, 16,)

4.2 Determining m^* for Multi-Product Case

For the one product case, m^* is determined by expression (3.1.1). When multiple products are in the system, that expression is no longer valid. A different methodology is therefore needed for determining m^* in a multi-product system. Tables that contain the information of completion times for each job on each machine can be easily produced with recursive equations. These values will be designated as $T(i, j)$ for the completion time of job i at station j . The values in these tables will be implemented to determine m^* for the multi-product case. A steady state is always achieved with deterministic processing times and repetitive sequences in the CONWIP system. The value of m^* is found for a specified repetitive sequence in steady state by utilizing the completion time table values.

First, the behavior of the values in the table for the single product case will be analyzed. The bottleneck must maintain 100% utilization for throughput to be at its upper bound. Therefore, at optimal throughput, the completion time at the bottleneck must be increased by (tb) for each subsequent job. This will also force the completion times at the other stations to be incremented by (tb) for each subsequent job. Thus, when the table values $T(i, j)$ start to increase by the bottleneck time (tb) for each new row which belong to a new job, then throughput reaches its upper bound in steady state as in (4.2.1) below.

$$(4.2.1) \quad T(i + 1, j) - T(i, j) = tb$$

The following example (4.2.1) includes Tables (4.2.1,2) which represent WIP levels of three and four respectively. The values in the table were obtained through Gantt

chart calculations done by hand. The tables illustrate the difference in behavior of the $T(i,j)$ values when the throughput is optimal or non-optimal.

In Table (4.2.1) with $m = 4$, rows (1,2,3,4) describe the system while it is in a transient state. In each of the following rows (5, 6, 7, 8), the values are increased by eight which is the bottleneck time $t_b = 8$. This indicates that optimum throughput is reached.

In Table (4.2.2) with $m = 3$, idle time exists at the bottleneck which indicates a non-optimum throughput rate. Therefore, the optimum WIP level must be ($m^* = 4$).

Example 4.2.1,

$$t(i, j) = [6 \quad 8 \quad 6 \quad 6]$$

$$m^* = 4,$$

Table (4.2.1) Example of Optimum One Product Values of $T(i,j)$

Job #	M 1	M 2	M 3	M 4
1	6	14	20	26
2	12	22	28	34
3	18	30	36	42
4	24	38	44	50
5	32	46	52	58
6	40	54	60	66
7	48	62	68	74
8	56	70	76	82

$m = 3,$

Table (4.2.2) Example of Non-Optimum One Product Values of $T(i,j)$

<u>Job #</u>	<u>M 1</u>	<u>M 2</u>	<u>M 3</u>	<u>M 4</u>
1	6	14	20	26
2	12	22	28	34
3	18	30 *i*	36	42
4	32	40	46	52
5	40	48	54	60
6	48	56 *i*	62	68
7	58	66	72	78
8	66	74	80	86

i denotes idle time at the bottleneck station

The objective now is to generalize the above development to determine m^* in a multi-product system for a given sequence. Since the sequence is repetitive, the system will reach a steady state. In this steady state, the cbn will not become idle if throughput has reached its upper bound. When idle time does not exist at the cbn during steady state, it can be concluded that throughput is optimal. Therefore, when expression (4.2.2) below is not violated, throughput must be optimal. If throughput is optimal for the WIP level of m while it was non-optimal for the previous WIP level of $(m-1)$, then the current WIP level is optimal. That is, $m = m^*$.

$$(4.2.2) \quad T(i + k, j = cbn) - T(i, j = cbn) = tcbn$$

Thus, to determine the optimal m^* , its value is increased step by step from the lower bound determined in section (4.1). When expression (4.2.2) is satisfied for a given m , that (m) determines the optimal value of (m) , i.e., $m = m^*$. Otherwise, (m) will be increased by one and the optimal throughput will be tested for again. (m) will continue to be increased until the optimal throughput is reached. We illustrate this using example (4.2.2). The following tables (4.2.3,4) contains the completion times of jobs, that were determined with Gantt chart calculations done by hand.

Table (4.2.3) is for the two product system of the example with the WIP level $(m = 3)$. The table illustrates, that for $m = 3$, the optimum throughput is not achieved since (4.2.2) is not obeyed. Table (4.2.3) exhibits idle time occurring at the cbn in rows $(2, 4, 6, 8, \dots)$. The idle times alternate between two and eight in the table. The system will continue to possess these idle times at the cbn as the sequence is repeated. Equation

(4.2.2) is violated since the difference between row (i) and row (i-2) at station (2) alternates between 22 and 28 every two jobs. The difference should be 20 for optimum throughput. The other stations will also maintain the same differences of 22 and 28 as the (cbn) in every other row. This pattern in the $T(i,j)$ values will repeat forever. Thus, the system exhibits a non-optimal steady state. This guarantees that for $m = 3$ optimality will never be achieved.

Table (4.2.4) is for the two product example with ($m = 4$). The system first passes through a transient state in rows (1, 2, 3, 4). Rows (5, 7, 9) are for product one and are shifted by a time of twenty. This is the amount of processing time t_{cbn} of the cbn at station two. Also, rows (6, 8, 10) are shifted by twenty for product two. Optimum throughput is confirmed since expression (4.2.2) is not violated and the system is in steady state. Thus, $m = m^* = 4$ since the throughput level was not optimal for ($m-1 = 3$).

Example 4.2.2.

$$t(i, j) = \begin{bmatrix} 6 & 10 & 6 & 12 \\ 12 & 10 & 6 & 6 \end{bmatrix}$$

Table (4.2.3) Example of Non-Optimum Two Product T(i,j) Values

Job #	Product	M 1	M2	Idle Time M2	(i,2) - (i-2, 2)	M3	M 4
1	1	6	16	0	-	22	34
2	2	18	28	2	-	34	40
3	1	24	38	0	22	44	56
4	2	34/46	56	8	28	62	68
5	1	40/52	66	0	28	72	84
6	2	56/68	78	2	22	84	90
7	1	68/74	88	0	22	94	106
8	2	84/96	106	8	28	112	118
9	1	90/102	116	0	28	122	134
10	2	106/118	128	2	22	134	140

Table (4.2.4) Example of Optimum Two Product T(i,j) Values

Job #	Product	M 1	M 2	M 3	M 4
1	1	6	16	22	34
2	2	18	28	34	40
3	1	24	38	44	56
4	2	36	48	54	62
5	1	42	58	64	76
6	2	54	68	74	82
7	1	62	78	84	96
8	2	74	88	94	102
9	1	82	98	104	116
10	2	94	108	114	122

Equations (4.2.7,8) provide the completion times of the jobs at each station. The start time of a job at station (j) is the later of the previous job's completion time at station (j) or that job's completion time at station (j-1). The Mod function is implemented to efficiently provide the processing times $t(i,j)$ of each job in the repetitive sequence.

$$(4.2.7) \text{ for } i = (m+1, \dots, rk) \quad \text{** Jobs following job (m) at the first station **}$$

$$\text{for } j = 1$$

$$T(i, j) = \text{Max} \{ T(i-1, 1), T(i-m, N) \} + t(\text{Mod}(i/k), j)$$

$$(4.2.8) \text{ for } i = (m+1, \dots, rk) \quad \text{** Jobs following job (m) at the remaining}$$

$$\text{for } j = (2, \dots, N) \quad \text{stations **}$$

$$T(i, j) = \text{Max} \{ T(i-1, j), T(i, j-1) \} + t(\text{Mod}(i/k), j)$$

At optimality, the time at the cbn during the subsequent occurrence of a product (k) in the table of $T(i,j)$ values is increased by the total amount of processing time t_{cbn} at the cbn, as expression (4.2.2) dictates. Since the products leave the cbn in the same manner for each cycle, the products will interact away from the cbn in an identical manner for each cycle. It follows that the entire table should shift by the total time t_{cbn} at the cbn for each consecutive cycle of all (k) products as in expression (4.3.9) below.

$$(4.2.9) \quad T(i+k, j) - T(i, j) = t_{cbn}$$

The following Table (4.2.5) summarizes equations (4.2.3-4.2.8) for optimality conditions.

Table (4.2.5) Generic (k) Product Table at Optimality

ROW	M (1)	M (2)	M (j)
Cycle (C)				
i	T(i,1)	T(i,2)	T(i,j)
i+1	T(i+1,1)	T(i+1,2)	T(i+1,j)
:	:	:	:	:
:	:	:	:	:
i+k-1	T(i+k-1,1)	T(i+k-1,2)	T(i+k-1,j)
Cycle (C+1)				
i+k	T(i,1) + tcbn	T(i,2) +tcbn	T(i,j) + tcbn
i+k+1	T(i+1,1) +tcbn	T(i+1,2) + tcbn	T(i+1,j) +tcbn
:	:	:	:	:
:	:	:	:	:
i+2k-1	T((i+k-1,1) + tcbn	T(i+k-1,2) + tcbn	T(i+k-1,j) + tcbn

5.0 A HEURISTIC PROCEDURE TO SEQUENCE JOBS ON THE BACKLOG LIST

5.1 Introduction to the Proposed Heuristic Procedure

The problem on hand is to sequence the jobs on the backlog list to optimize throughput with a minimum amount of WIP (m^*). Previously, it has been shown in chapter 3 that the sequence in which the jobs are arranged affects throughput for certain conditions for bottlenecks. More significantly, it was illustrated that a sequence of the type (ABCDABCD.....) is a superior strategy to optimize throughput. However, the optimum WIP level (m^*) is different for different orders of jobs. For the four jobs A,B,C,D, examples of two sequences are: (ABCDABCD.....) and (DCBADDCBA.....). The m^* for these two sequences need not be the same. It is desired to determine the sequence that will minimize m^* .

The number of possible sequences for CONWIP or the static flowshop involving k different jobs is $(k!)$. As the number of jobs (k) is increased, obtaining the optimum sequence becomes computationally difficult. For the static flowshop, this has resulted in the development of heuristic procedures to obtain a good and not necessarily an optimal solution to the makespan problem. Accordingly, a heuristic approach is developed to determine the sequence of backlog for the CONWIP system to optimize throughput in steady state with a minimum value for (m^*).

A heuristic approach that has been previously incorporated for the static flowshops to minimize makespan is to consider how well a job fits with another job as a predecessor and successor pair. A superior fit is defined to be the one in which less job

blocking and machine idle time are expected. A cost for each predecessor (i) to another job (j), denoted as (C_{ij}) , is determined from the processing times of the jobs. These costs which reflect how well the pair fits are then implemented to determine the sequence.

Stinson and Smith (1982) utilized this approach. It will be investigated as to how well this type of approach meets the objectives of the CONWIP system.

Completion times for jobs are determined recursively by the following expression:

$$(5.1.1) \quad T(j, k) = \max\{ T(i, k), T(j, k-1) \} + t(j, k)$$

Where $T(j, k)$ is the completion time of job (j) at station (k) and $t(j, k)$ is the processing time of job (j) at station (k). Expression (5.1.1) dictates that the start of job (j) at station (k) is the later of the completion time of job (j) at the previous station or the completion time of the previous job at station (k).

The rationale behind the previously mentioned heuristic comes from the effects of the differences between $T(i, k)$ and $T(j, k-1)$. Queuing and job idle time will occur for job (j) at station (k) when $T(i, k) > T(j, k-1)$. Machine idle time will be incurred at station (k) when $T(i, k) < T(j, k-1)$ before job (j) arrives. The desired occurrence is not to incur machine or job idle times.

To achieve the elimination of the machine and job idle times, it would be necessary for $t(i, k) = t(j, k-1)$ for stations $k = (2, \dots, m)$. Obviously, it would be highly unlikely to meet these conditions. However, by pairing the jobs with closer values for the processing times ($t(i, k), t(j, k-1)$), it is anticipated that the flow of jobs through the system will be smoother. For the static flowshop, the desired effect from this would be a reduction in the makespan value as reported in Stinson and Smith (1982). In the

CONWIP system, it is expected that this smoother flow of products will result in lowering the value of m^* .

Stinson and Smith (1982) define the difference in the processing times $t(i,k)$ and $t(j,k-1)$ as a residual $r(ij,k)$.

$$(5.1.2) \quad r(ij,k) = t(i,k) - t(j,k-1) \quad k = (2, \dots, m)$$

This residual value is then utilized in six different heuristics to determine a cost of a predecessor and successor pair. The six heuristics include Sum of Absolute Residuals and Sum of Residuals Squared with and without negative residual carryover, Sum of Negative Residuals with negative residual carryover and Sum of Absolute Residuals with negatives weighted double. A negative residual carryover is utilized to account for a negative residual at stations downstream of where the initial negative residual occurred.

Stinson and Smith (1982) investigated weighting negative residuals more severely. They theorized that the machine idle time that is expected from a negative residual would be more detrimental to the objective of minimizing the makespan than job idle times. Their results demonstrate that the heuristic incorporating stronger weights on the negative residual is superior to the other heuristics. The different weighting of positive and negative residuals will also be examined in this research.

The following expression indicates Sum of Absolute Residuals as cost of a pair of jobs (i,j).

$$(5.1.3) \quad C_{ij} = \sum_{k=2}^m |r(ij,k)|$$

Example (5.1.1) Illustration of Heuristic Procedure

The following Table (5.1-1) contains the processing times for the heuristic procedure example.

Table (5.1-1) Processing times for example 5.1.1

JOB	M-1	M-2	M-3
1	10	14	10
2	10	12	4
3	10	6	12
4	10	8	8
5	10	10	8

From (5.1.3) the cost for job (1) followed by job (2) is determined as follows:

$$C_{12} = r(12,2) + r(12,3)$$

$$r(12,2) = \text{abs}(t(1,2) - t(2,1)) = 4$$

$$r(12,3) = \text{abs}(t(1,3) - t(2,2)) = 2$$

$$C_{12} = 4 + 2 = 6$$

The cost matrix corresponding to all the pairs of jobs is now summarized in Table 5.1-2. The diagonal elements are not valid since no job can follow itself.

Table (5.1-2) Cost Matrix C_{ij} : The Sum of Absolute Residuals for the Predessor and Successor Pair (i,j)

i / j	1	2	3	4	5
1	X	6	6	6	4
2	12	X	4	6	8
3	6	4	X	8	6
4	8	6	4	X	4
5	6	4	2	0	X

The low cost elements in the matrix are preferred since this indicates the likelihood of a better fit between the two jobs. Thus, the objective is to find the sequence that minimizes total cost. This is analogous to the classical traveling salesman problem. Heuristic methods have commonly been applied to solve the traveling salesman problem. Stinson and Smith (1982) utilized a heuristic procedure for this purpose which will also be implemented in this research.

The first step consists of finding the difference between the minimum and second minimum element in each row and column. The second step is to select the element that has the highest difference in all rows and columns. For example, in the first row the minimum element is $C_{15} = 4$ and the second minimum elements $C_{12} = 6$. Taking the difference yields a value of $6-4=2$ for the first row. The remaining row and column differences are provided below.

Table (5.1-3) Row Differences Between the Second and First Minimums in the Row of the C_{ij} Matrix

Row Difference

1	2
2	2
3	2
4	2
5	2

Table (5.1-4) Column Differences Between the Second and First Minimums in the Column of the C_{ij} Matrix

Column Difference

1	2
2	2
3	2
4	6
5	2

The selected element will be from column four which has the maximum difference of six between the lowest and second lowest elements of any row or column. The selected element is the pair (5,4) and the partial sequence of 5,4 is obtained. Then each remaining element in the selected pair's row and column is eliminated. The element (4, 5) is also eliminated to insure that the sequence will remain feasible.

Table (5.1-5) Updated Cost Matrix C_{ij} with the Selection of $C_{5,4}$ and Elimination of other Appropriate C_{ij}

i / j	1	2	3	4	5
1	X	6	6	X	4
2	12	X	4	X	8
3	6	4	X	X	6
4	8	6	4	X	X
5	X	X	X	**0**	X

Again, the maximum difference is found for all remaining rows and columns. It is determined that row two contains the maximum difference. The element to be selected will be element (2,3).

Table (5.1-6) Row Differences Between the Second and First Minimums in the Row of the Updated C_{ij} Matrix

Row Difference

1	2
2	4
3	2
4	2
5	X

Table (5.1-7) Column Differences Between the Second and First Minimums in the Column of the Updated C_{ij} Matrix

Column Difference

1	2
2	2
3	2
4	X
5	2

Table (5.1-8) Updated Cost Matrix C_{ij} with Selection of $C_{2,3}$ and the elimination of the other appropriate C_{ij}

i / j	1	2	3	4	5
1	X	6	X	X	4
2	X	X	**4**	X	X
3	6	X	X	X	6
4	8	6	X	X	X
5	X	X	X	**0**	X

This process is continued with the ties being broken arbitrarily. The final matrix is shown below in Table (5.1-9). The full sequence that results is (2,3,1,5,4,.....).

Table (5.1-9) Final Cost Matrix for example 5.1.1

i / j	1	2	3	4	5
1	X	X	X	X	**4**
2	X	X	**4**	X	X
3	**6**	X	X	X	6
4	X	**6**	X	X	X
5	X	X	X	**0**	X

5.2 Results from Experimentation with the Proposed Heuristic

The objective of the experimentation is to determine the effectiveness of the heuristic in achieving its objectives. The sequencing heuristics, lower bound procedure, and recursive equations for determining (m^*) were programmed in FORTRAN. The processing times of the jobs on different stations were randomly generated with uniform distributions, except for their times on the bottleneck station which were held constant for all jobs as dictated by the framework of the CONWIP system. For each set of experiments, the same bottleneck station always possessed the maximum combined amount of processing time of all stations.

The effects of varying the number of stations and jobs, distributions, bottleneck location and heuristics are examined through the experimentation. Three different types of sequences are incorporated in the experimentation. These are random, high, and low cost sequences. The low and high cost sequences are determined by the heuristic procedure. Three levels of variance (high, medium, low) are utilized to determine the effects of variability.

The three sequences are examined for two performance measures. The first is the throughput rate at the lower bound for optimal WIP, i.e, the lower bound determined in Chapter 4. These are designated as $\Theta_{1,2,3}$ in the following tables. Θ_1 corresponds to the low cost sequence, Θ_2 to the random sequence and Θ_3 to the high cost. This will provide an indication of how near the lower bound is to the optimal WIP levels for the three sequences. The second measure to be utilized is the optimal WIP level for all three sequences. m_1^* corresponds to the low cost sequence, m_2^* to the random cost, and m_3^* to the high cost sequence. The number of different products in the repetitive sequence is given in the first column of the following tables.

Table (5.2-1a,b,c)

Average Throughput and Optimal WIP Levels for Low, Random, and High Cost Sequences over 30 Observations

Stations = 5, Bottleneck Station = 3, Bottleneck Time = 10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0,15)

Mean = 7.5 Variance = 18.75

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0927	6.10	0.0916	6.60	0.0882	7.05	4.95
15	0.0926	6.60	0.0893	6.90	0.0872	7.40	4.80
20	0.0932	6.60	0.0910	7.10	0.0885	7.40	4.93
25	0.0910	6.70	0.0888	7.10	0.0867	7.60	4.75

Distribution of Processing Times at Non-bottleneck Stations: Uniform (3,12)

Mean = 7.5 Variance = 6.75

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0978	5.30	0.0969	5.15	0.0956	5.55	4.70
15	0.0981	5.40	0.0971	5.75	0.0963	5.85	4.85
20	0.0989	5.40	0.0982	5.85	0.0979	6.00	4.95
25	0.0986	5.77	0.0984	5.80	0.0979	5.97	4.97

Distribution of Processing Times at Non-bottleneck Stations: Uniform (5,10)

Mean = 7.5 Variance = 2.08

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0994	5.0	0.0994	5.0	0.0993	5.0	4.85
15	0.0995	5.0	0.0996	5.0	0.0995	5.0	4.90
20	0.0991	5.0	0.0991	5.0	0.0991	5.0	4.80
25	0.0997	5.0	0.0996	5.0	0.0996	5.0	4.90

Table (5.2-2a,b,c)

Average Throughput and Optimal WIP Levels for Low, Random, and High Cost Sequences over 30 Observations

Stations = 8, Bottleneck Station = 4, Bottleneck Time = 10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0,15)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0892	9.27	0.0860	9.77	0.0830	10.30	7.17
15	0.0883	9.73	0.0856	10.47	0.0839	11.30	7.27
20	0.0899	9.87	0.0870	11.00	0.0840	11.67	7.33
25	0.0899	9.97	0.0860	10.87	0.0829	11.73	7.20

Distribution of Processing Times at Non-bottleneck Stations: Uniform (3,12)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0961	7.87	0.0941	8.20	0.0930	8.37	6.97
15	0.0960	8.03	0.0941	8.30	0.0936	8.67	7.00
20	0.0952	8.30	0.0937	8.47	0.0924	8.80	7.00
25	0.0960	8.43	0.0943	8.53	0.0933	8.73	7.00

Distribution of Processing Times at Non-bottleneck Stations: Uniform (5,10)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0999	7.20	0.0996	7.40	0.0998	7.35	7.00
15	0.0997	7.40	0.0995	7.30	0.0996	7.55	7.00
20	0.0997	7.45	0.0995	7.65	0.0996	7.60	7.00
25	0.0998	7.45	0.0995	7.75	0.0995	7.65	7.00

Table (5.2-3a,b,c)

Average Throughput and Optimal WIP Levels for Low, Random, and High Cost Sequences over 30 Observations

Stations = 8,

Bottleneck Station = 4,

Bottleneck Time = 10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0,15)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0859	9.50	0.0853	10.00	0.0836	10.80	7.05
15	0.0884	9.75	0.0848	10.50	0.0841	10.85	7.20
20	0.0887	9.80	0.0847	11.10	0.0828	11.65	7.20
25	0.0882	10.30	0.0849	11.05	0.0831	11.85	7.25

Distribution of Processing Times at Non-bottleneck Stations: Uniform (3,12)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0957	8.10	0.0945	8.10	0.0931	8.40	7.00
15	0.0952	8.20	0.0939	8.50	0.0932	8.65	7.00
20	0.0950	8.45	0.0939	8.50	0.0923	8.85	6.95
25	0.0962	8.30	0.0942	8.50	0.0921	8.70	7.00

Distribution of Processing Times at Non-bottleneck Stations: Uniform (5,10)

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_2	m_2^*	θ_3	m_3^*	
10	0.0997	7.20	0.0995	7.55	0.0994	7.45	7.00
15	0.0999	7.15	0.0998	7.40	0.0999	7.20	7.00
20	0.0999	7.20	0.0995	7.60	0.0995	7.65	7.00
25	0.0996	7.50	0.0994	7.50	0.0995	7.80	7.00

Table (5.2-4a,b,c)

Average Throughput and Optimal WIP Levels for Low, Random, and High Cost Sequences over 30 Observations

Stations = 8, Bottleneck Station = 4, Bottleneck Time = 10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0,15)

	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		
#Product	Θ_1	m_1^*	Θ_2	m_2^*	Θ_3	m_3^*	m_{lb}^*
10	0.0878	9.40	0.0856	10.20	0.0841	10.50	7.25
15	0.0891	9.80	0.0854	10.60	0.0823	11.40	7.20
20	0.0895	9.85	0.0850	10.85	0.0829	11.55	7.20
25	0.0893	9.95	0.0856	10.85	0.0830	11.55	7.10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (3,12)

	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		
#Product	Θ_1	m_1^*	Θ_2	m_2^*	Θ_3	m_3^*	m_{lb}^*
10	0.0943	8.05	0.0943	8.05	0.0924	8.35	6.95
15	0.0955	8.15	0.0933	8.50	0.0920	8.65	6.95
20	0.0958	8.10	0.0947	8.35	0.0930	8.80	7.00
25	0.0962	8.20	0.0952	8.50	0.0935	8.80	7.00

Distribution of Processing Times at Non-bottleneck Stations: Uniform (5,10)

	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		
#Product	Θ_1	m_1^*	Θ_2	m_2^*	Θ_3	m_3^*	m_{lb}^*
10	0.0996	7.25	0.0995	7.30	0.0996	7.45	7.00
15	0.0995	7.20	0.0994	7.25	0.0994	7.50	7.00
20	0.0997	7.40	0.0996	7.45	0.0996	7.40	7.00
25	0.0998	7.40	0.0996	7.60	0.0996	7.60	7.00

Table (5.2-5a,b,c)

Average Throughput and Optimal WIP Levels for Low, Random, and High Cost Sequences over 30 Observations

Stations = 8,

Bottleneck Station = 1, 4, 8,

Bottleneck Time = 10

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0,15)

Bottleneck Station: 1

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_1	m_2^*	θ_1	m_3^*	
10	0.0973	9.40	0.0946	10.07	0.0930	10.43	8.47
15	0.0969	9.73	0.0946	10.23	0.0915	11.23	8.70
20	0.0979	9.93	0.0943	10.90	0.0915	11.83	8.93
25	0.0980	9.83	0.0947	10.90	0.0929	11.33	8.90

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0, 15)

Bottleneck Station: 4

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_1	m_2^*	θ_1	m_3^*	
10	0.0892	9.27	0.0860	9.77	0.0830	10.30	7.17
15	0.0883	9.73	0.0856	10.47	0.0839	11.30	7.27
20	0.0899	9.87	0.0870	11.00	0.0840	11.67	7.33
25	0.0899	9.97	0.0860	10.87	0.0829	11.73	7.20

Distribution of Processing Times at Non-bottleneck Stations: Uniform (0, 15)

Bottleneck Station: 8

#Product	Low Cost Sequence		Random Cost Seq.		High Cost Sequence		m_{lb}^*
	θ_1	m_1^*	θ_1	m_2^*	θ_1	m_3^*	
10	0.0973	9.15	0.0942	9.90	0.0915	10.53	8.43
15	0.0976	9.55	0.0947	10.4	0.0924	11.10	8.80
20	0.0988	9.60	0.0950	10.60	0.0923	11.35	9.0
25	0.0983	9.60	0.0950	10.85	0.0920	11.65	8.90

The data illustrated in Tables (5.2-1a,b,c) are for a system with five stations. The bottleneck is at station three. The heuristic utilized is the Sum of Absolute Residuals. The data is for an average obtained from thirty experiments. It was determined that this number of experiments were sufficient for the data to stabilize through observations of the data's behavior. The three tables each represent a different distribution.

The data illustrates the effectiveness of the heuristic in determining the sequence to meet the stated objectives. The low cost sequence provided lower average optimum WIP levels relative to the random sequence. In addition, the high cost sequence resulted in higher average optimum WIP levels relative to the random sequence.

The low cost sequence possessed higher throughput at the lower bound than the random sequence which outperformed the high cost sequences. It has been observed that sequences with higher throughput rates below the optimum WIP level have the tendency for lower optimum WIP levels. Therefore, the throughput measured at the lower bound also confirmed the effectiveness of the heuristic

It is substantiated in the data for the random sequence that higher variance distributions result in higher average optimum WIP levels. Also, higher variances increase optimum WIP levels for the high and low cost sequences. The data provides evidence that higher levels of variance result in a larger range of optimum WIP levels among all sequences. Therefore, the potential benefits of utilizing the heuristic are greatest when higher variance levels exist. Tables (5.2-1a,b,c) illustrate that optimum WIP levels are

most significantly improved by the heuristic in comparison with random sequences in systems with high variance. Table (5.2-1c) illustrates that the low variance distribution resulted in all three average optimum WIP levels for all amounts of products being the same.

An important concern that was examined is how the lower bound is affected by certain changes in the system. Higher variance did not increase the average lower bound in the distributions observed. Greater variance increases queuing in the system which can potentially increase the optimum WIP level. The higher variance distributions resulted in higher optimum WIP levels but the lower bound was not increased by the additional variance. Therefore, it can be concluded that the lower bound will be closest to the optimum WIP level when the system possesses low variance.

The effect of increasing the number of products in the sequence on the lower bound was not evident. The trend of the lower bound was neither increasing or decreasing as the number of jobs were increased for the three distributions. This may be attributed to different factors having a cancellation effect on each other. The positive effect of increasing the number of jobs is the availability of more potential matches in the lower bound procedure. The negative effect is the potential for greater amounts of processing time away from the bottleneck as the number of jobs in the sequence is increased.

The data in tables (5.2-2a,b,c) is for an eight station system. The same distributions and heuristics were utilized as in the five station system in tables (5.2-1a,b,c).

The data shows that increases in the number of stations increases the lower bound and optimal WIP levels when all other factors are held constant. The data illustrates the same behavior that was previously discussed for the data from the five station system in tables (5.2-1a,b,c). The larger number of stations increased the range of optimal WIP levels and therefore accentuates the benefits of utilizing the heuristic.

Another concern that was examined in the experimentation was whether the positive or negative residual was more significant. The positive residual tends to create job idle time while the negative residual tends to create machine idle time. The experimentation consisted of holding all other factors constant from tables (5.2-2a,b,c) and weighting the residuals differently. In tables (5.2-3a,b,c) the positive residuals were weighted double. In tables (5.2-4a,b,c) the negative residuals were weighted double.

The data in tables (5.2-3,4a,b,c) for weighting the positive or negative residual more strongly does not confirm any advantage or disadvantage when compared with the equally weighted residuals in tables (5.2-2a,b,c). Thus, it is concluded that both positive and negative residuals are equally important in the heuristic. This is in contrast to Stinson and Smith's results which illustrate that weighting negative residuals more significantly is advantageous to improving the performance for the makespan criteria.

The effect of moving the bottleneck while holding other factors constant was also investigated. The data for these experiments is contained in tables (5.2-5a,b,c). The experiments are for an eight station system with a high variance uniform distribution. The

Sum of Absolute residuals was utilized. The bottleneck station was moved between the first, last and middle stations.

The data in tables (5.2-5a,b,c) verifies that moving the bottleneck does not effect the optimal WIP levels for the three sequences. The bottleneck was found to increase the lower bound when moved to either of the ends of the line. This is an effect which is inherent to the lower bound procedure itself. Thus, the lower bound is closer to the optimal WIP value as the bottleneck is moved away from the center of the line.

Table (5.2-6a,b,c) Throughput vs. WIP Levels

Stations = 8, Bottleneck Station = 4,

Distribution: Uniform (0,15)

WIP	Theta-1	Theta-2	Theta-3
5	.0701	.0669	.0655
6	.0795	.0764	.0740
7	.0870	.0847	.0813
8	.0931	.0902	.0876
9	.0969	.0951	.0925
10	.0992	.0985	.0974
11	.10	.0999	.0994
12	.10	.10	.10

Distribution: Uniform (3,12)

WIP	Theta-1	Theta-2	Theta-3
5	.0755	.0740	.0726
6	.0884	.0853	.0839
7	.0964	.0941	.0926
8	.0998	.0994	.0983
9	.10	.10	.10

Distribution U(5,10)

WIP	Theta-1	Theta-2	Theta-3
5	.0781	.0776	.0771
6	.0916	.0908	.0904
7	.0998	.0998	.0996
8	.10	.10	.10

Figures 5.2-1 to 5.2-4 illustrate throughput vs. WIP curves. The data for the graphs is contained in Tables (5.2-6 a,b,c). The system consists of eight stations with the bottleneck at the fourth station. The number of products was twenty. The Sum of Absolute Residuals heuristic was utilized. The data is for averages of thirty experiments. Figure 5.2-1 is for high variance with a uniform distribution (0,15). Figure 5.2-2 is for medium variance with a uniform distribution (3,12). Figure 5.2-3 is for low variance with uniform distribution (5,10). Figure 5.2-4 is for random sequences with high, medium, and low variance.

Figures 5.2-1 to 5.2-3 confirm the ability of the heuristic to improve the throughput of the system over a range of WIP levels. The figures illustrate that the low cost sequences possess better throughput over a range of WIP and reach optimality quicker. Also, the high cost sequence possesses lower throughput when contrasted with the random sequence. The improvements in throughput were less significant for WIP levels near optimality as the curves converged. Again, the improvements in throughput are most significant with higher levels of variance in the system.

Figure 5.2-4 depicts random sequences with different variance levels. The figure illustrates that lower variance in the system will result in optimal throughput at lower levels of WIP. The low variance system reaches optimality at eight jobs, the medium variance at nine jobs and the high variance at eleven jobs. This can be attributed to high variance systems resulting in queuing outside of the system's bottleneck.

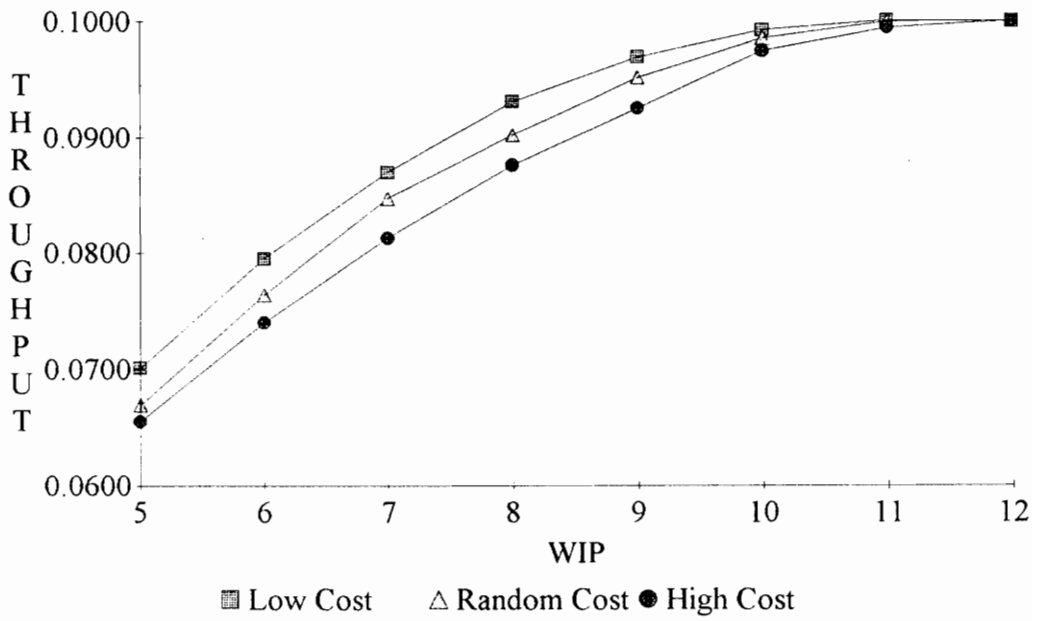


Figure (5.2-1) High Variance System

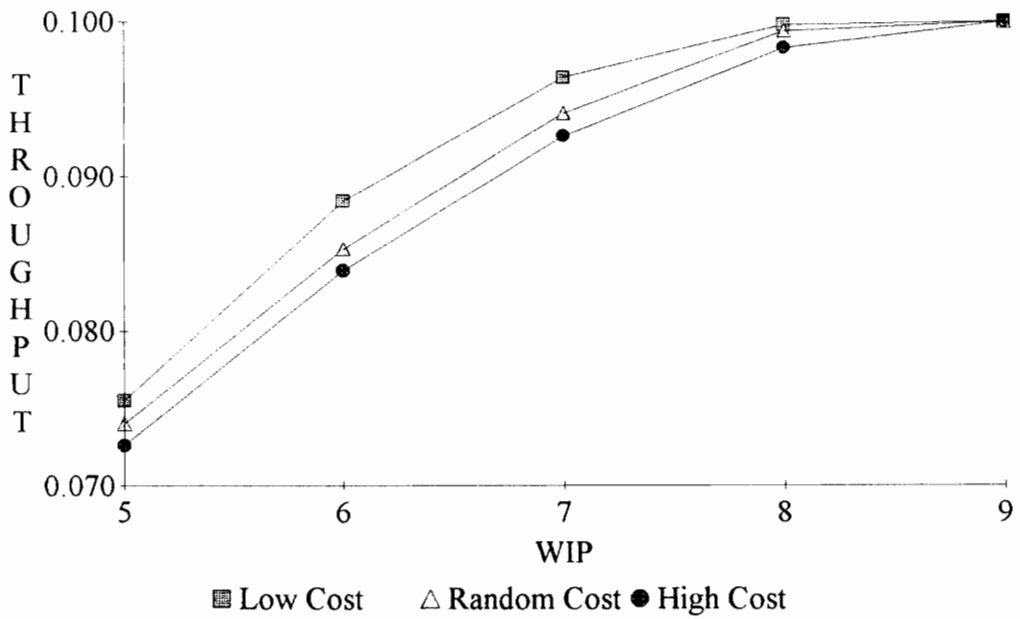


Figure (5.2-2) Medium Variance System

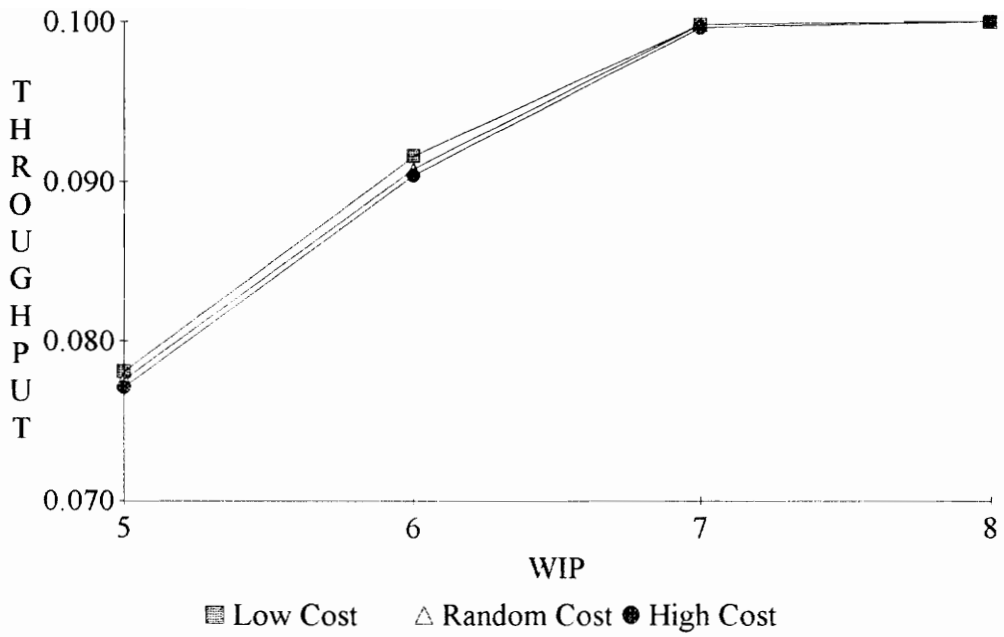


Figure (5.2-3) Low Variance System

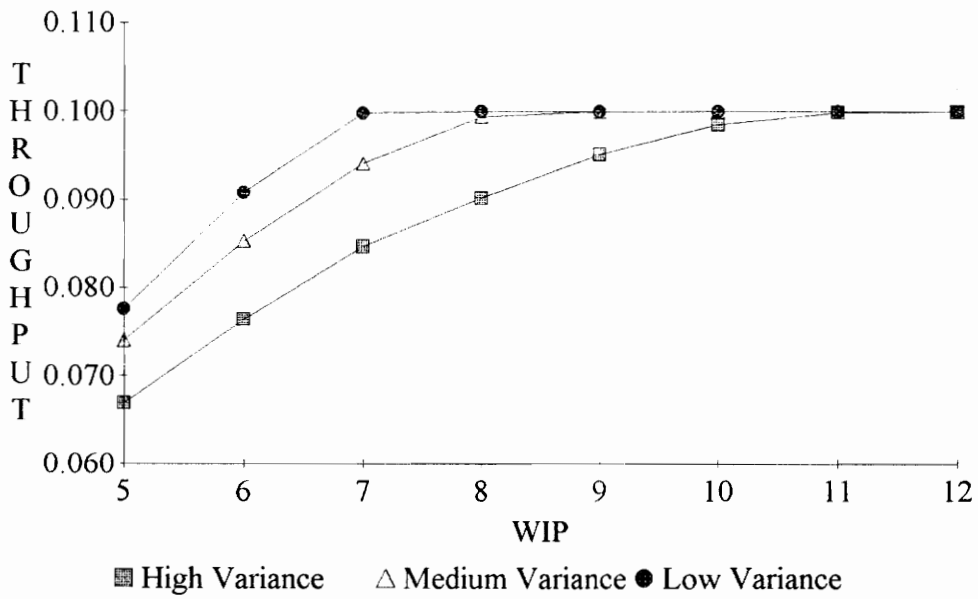


Figure (5.2-4) Random Sequences

6.0 Conclusions and Recommendations

6.1 Summary and Conclusions

The research in this thesis has focused on the effects of backlog sequencing on the performance of the pull type production system of CONWIP. The framework for the CONWIP system presented in its original paper was maintained throughout this research. The ordering of the CONWIP backlog list, which specifies the sequence, affects the performance of the system and therefore warrants research.

The performance of the CONWIP system in this research was measured in terms of the throughput level for a fixed level of WIP in deterministic steady state. Deterministic steady state is reached in a system that possesses deterministic processing times, constant WIP and a repetitive sequence. Throughput remains constant over each cycle of the sequence in deterministic steady state. The objective of this research was to determine the backlog sequence that achieves optimum throughput in deterministic steady state with a minimum WIP level.

The issues investigated in this research concerning the CONWIP production system are new. The CONWIP system, as described originally, was for a serial production line which is essentially a flow shop. The key difference of the CONWIP system from a flow shop is that the level of WIP is fixed in the CONWIP system. Flow shops have been thoroughly investigated for the problem of minimizing makespan. Both optimization and heuristic procedures have been proposed to sequence jobs to be processed in a flow shop. We use one existing heuristic approach to determine the required backlog sequence.

The effect of backlog sequence on system performance was studied. A two product system was initially analyzed. The findings can be summarized for two different cases. The first case considered the situation when both the products have the same station as their bottlenecks. This case proved to be of lesser interest because the type of sequence does not greatly affect the optimum WIP level. The second and more significant case is when two products have different stations as bottlenecks. In this case, a new combined bottleneck station is defined to be the one which possesses the largest sum of processing times of both the products. This combined bottleneck station was found to determine the upper bound of throughput in the system. To achieve the optimum throughput level, it is necessary to maintain 100% utilization of this combined bottleneck station.

For this case, at least one product will not have its bottleneck station at the combined bottleneck station. If this product is repeated in the sequence, it will eventually reach a steady state where the combined bottleneck possesses idle time. This results in reduced throughput. The ramification of this phenomenon to sequencing policy is that individual products must not be repeated to ensure optimum throughput. This establishes the optimum sequencing policy to be of the type (ABABAB.....).

This analysis was extended further to larger product mixes. Similar to the two product case, it was observed that the upper bound on throughput was determined by the combined bottleneck station corresponding to the greatest sum of processing times of all the products. It was then argued that for the multiple product case the optimum sequence

policy should be of the type (ABCD....ABCD....). The basis of the argument follows from the observations of the two product case. In addition, it was realized that the optimum WIP level m^* and the throughput level for a fixed WIP level are sequence dependent. The optimum WIP level m^* was determined by the value of the maximum return time to the bottleneck. The sequence affects these return times, thereby, affecting (m^*).

Further study of flow times indicated that the behavior of the flow times during transitions between products is affected by the direction in which the bottleneck moves. This gives rise to the potential of exploiting the transition phases in order to improve the performance of the system. Flow times obey Little's law. Average flow times are reduced during transition phases under certain conditions. By Little's law, throughput must increase when average flow times are reduced and WIP is held constant.

Recursive equations were determined which provide the completion times of the jobs at all stations in the system. The primary objective for determining the equations was to determine the optimum WIP level m^* for a specific sequence starting from a lower bound for m^* . The throughput was measured in steady state by allowing a significant number of cycles to occur prior to measuring the throughput. The equations were programmed in Fortran in order to readily provide the desired result.

A procedure to determine the lower bound for m^* was devised in order to increase the efficiency in determining the m^* . This procedure is based on the fact that the bottleneck must maintain 100% utilization to optimize throughput. The concept of entering and leaving jobs was developed. A repetitive sequence results in the same

matches of entering and leaving jobs during each cycle of jobs. Enough WIP needs to be in the system to maintain 100% utilization of the bottleneck during the combined time of the entering and leaving jobs. This time was designated as the return time for the entering and leaving job match.

The lower bound procedure initially requires determining a matrix of the return time values for all feasible matches. A methodology was then presented that provides a set of matches that yields a sequence independent lower bound. The procedure does not consider interaction of the products in the system. Thus, queuing may occur outside of the bottleneck station which increases the optimal WIP level above the value determined by the lower bound procedure.

The lower bound procedure was incorporated into the computer program. The behavior of the lower bound was then observed. The experimentation utilized uniform distributions for the processing times of the jobs throughout. It was observed that the lower bound tends to increase as the bottleneck is moved away from the center station. The behavior of the lower bound was not affected as the variance of the processing times and the number of jobs were varied.

A heuristic algorithm approach was utilized to achieve the objective of minimizing the optimum WIP level m^* . The algorithm was tested to determine its effectiveness in meeting its objective. The approach assigns costs based upon the relationship of $(t(i, j+1) - t(i+1, j))$ for a predecessor and successor pair of products. The term denoted as the "residual" (r_{ij}) is summed for all stations to determine the cost (C_{ij}).

A traveling salesman problem based approach is then incorporated to determine a low cost sequence. It is theorized that for a low cost sequence jobs will “fit” well together. This will improve the flow of jobs in the system by reducing the tendency for machine’s and job’s to have idle times. This results in a better performance of the system in terms of the stated objectives.

The algorithm was programmed and then tested. It is shown to provide sequences that perform significantly better when compared with random sequences. Three sequences were tested that included low cost, random cost, and high cost sequences. The high cost sequence was utilized to further reinforce the notion that the cost of the sequence is strongly related to its performance.

A factor that was observed to have the greatest effect on system performance is the variance of the processing times. The optimum WIP level was shown to increase as the variance was increased while keeping the other factors unchanged. Greater variance in processing times was shown to increase the range for throughput values at a fixed WIP level. Therefore, proper sequencing is most important for systems with higher processing time variances. The location of the bottleneck station did not seem to effect the optimum WIP level while holding all other factors constant.

Another factor that was examined is the weighting of the residual r_{ij} based upon its sign. A positive residual tends to cause job idle time. A negative residual tends to cause machine idle time. The results of weighting either of the residuals more than the other did

not improve the performance of the heuristic. A conclusion that can be drawn from this experimentation is that both the residuals have equal significance.

The complexity of the problem resulted in the development of the heuristic approach. The results of the research illustrate the effectiveness of this heuristic approach. Furthermore, it is anticipated that extensions of this type of approach could provide improved results.

6.2 Recommendations

This research has shown that backlog sequencing for the CONWIP production system affects the performance of the system. The heuristic approach provides results that verify its effectiveness and therefore warrant further study. Potential ideas for areas of future research include:

1. Examine the effects of sequencing for other types of production configurations, such as: job shop, assembly, etc.
2. Expand the algorithm to include additional procedures to perturb the initial sequence to facilitate obtaining an improved sequence from the initial sequence provided.
3. Examine new variations of heuristics that utilize concepts from this heuristic and previous flow shop heuristics.
4. Introduce stochasticity in areas such as: the processing times of products, demand for products, and machine breakdown.
5. Eliminate some of the assumptions made in the original description of the CONWIP system and this research, such as: no job passing, constant amount of work at the bottleneck, and unlimited queue space.
6. Incorporate the effects of sequence dependent set-up times into the criteria for the problem.

The potential advantages of the CONWIP production system when compared with other systems have been documented. It is necessary to examine the system's behavior under realistic conditions to further understand its potential benefits and to be able to predict its behavior for implementation. The potential areas for research with the CONWIP production system are quite significant. The positive results presented in this research and previous research warrant additional study of the CONWIP system.

REFERENCES

- [1] Adan, I., and Van der Wal, J., "Monotonicity of the Throughput of a Closed Queuing Network in the Number of Jobs", Operations Research, 1989, 953-957.
- [2] Baskett, F., Chandy, K. M., Muntz, R. R., Palacios, F. G., "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", Journal of Association of Computing Machinery, Vol. 22, No. 2, 1975, 248-260.
- [3] Bonney, M.C., and Gundry, S.W., "Solutions to the Constrained Flow Shop Sequencing Problem", Operations Research Quarterly, Vol.27, No.4, 1976, 869-883.
- [4] Campbell, H.G., Dudek, R.A., Smith, M.L., "A Heuristic Algorithm for the n-job, m-machine Sequencing Problem", Management Science, Vol.16, No.10, June 1970, 630-637.
- [5] Changchit, C., and Kung, H., "Effect of Learning in JIT Production System: a Simulation Experiment on Microcomputer", Computer and Industrial Engineering, Vol. 15, 1988, 172-178.
- [6] Changchit, C., and Terrel, M. P., "Issues in Just-in-Time Production Systems", International Industrial Engineering Conference Proceedings, 360-364.
- [7] Chu, C., and Shih, W., "Simulation Studies in JIT Production", International Journal of Production Research, Vol. 30, No. 11, 1992, 2573-2586.
- [8] Duenyas, I., and Hopp, W. J., "CONWIP Assembly with Deterministic Processing and Random Outages", IIE Transactions, Vol. 24, No. 4, September 1992, 97-109.
- [9] Duenyas, I., Hopp, W. J., and Spearman M. L., "Characterizing the Output Process of a CONWIP Line with Deterministic Processing and Random Outages", Management Science, Vol. 39, No. 8, August 1993, 975-988.
- [10] Duenyas, I., and Hopp, W. J., "Estimating the Throughput of an Exponential CONWIP Assembly System", Queueing Systems, Vol. 14, 1993, 135-157.
- [11] Ebrahimpour, M., and Fathi, B. M., "Dynamic Simulation of a Kanban Production Inventory System", International Journal of Operations and Production Management, Vol.5, 1984, 5-14.

- [12] Gordon, W. J., and Newell, G. F., "Closed Queuing Systems with Exponential Servers", Operations Research, Vol. 15, 1967, 254-265.
- [13] Gupta, J. N. D., "A Functional Heuristic Algorithm for the Flowshop Scheduling Problem", Operations Research Quarterly, Vol. 22, No. 1, March 1971, 39-47.
- [14] Gupta, J. N. D., "Heuristic Algorithms for Multi-stage Flowshop Scheduling Problem", AIIE Transactions, Vol. 4, No. 1, 1972, 11-19.
- [15] Gupta, Y., and Gupta, M., "A Systems Dynamics Model for a Multi-stage Multi-Line Dual-Card JIT-Kanban System", International Journal of Production Research, Vol.27, No.2, 1989, 309-352.
- [16] Gupta, J. N. D., and Maykut, A. R., "Flowshop Scheduling by Heuristic Decomposition", International Journal of Production Research, Vol. 11, No. 2, 1973, 105-111.
- [17] Hopp, W. J. and Spearman M.L., " Throughput of a CONWIP Manufacturing Line Subject to Failures", International Journal of Production Research, Vol. 29, No. 3, 1991, 635-655.
- [18] Hopp, W. J., and Spearman, M. L., and Duenyas, I., "Economic Production Quotas for Pull Manufacturing Systems", IIE Transactions, Vol. 25, No. 2, March 1993, 71-79.
- [19] Huang, P. Y., Rees, L. P., and Taylor, B. W., "A Simulation Study of the Japanese Just-in-Time Techniques for a Multi-line, Multi-stage Production System", Decision Sciences, Vol. 14, 1983, 326-344.
- [20] Johnson, S. M., "Optimal Two and Three Stage Production Schedules with Set-up Time Included", Naval Research Logistics Quarterly, Vol. 1, No. 1, March 1954, 61-68.
- [21] King, J. R., and Spachis, A. S., "Heuristics for Flowshop Scheduling", International Journal of Production Research, 18, 1980, 345-357.
- [22] Krajewski, L. J., King, B. E., Ritzman, L.P., and Wong, D.S., "Kanban, MRP, and Shaping the Manufacturing Environment", Management Science, Vol. 33, 1987, 39-57.
- [23] Monden, Y., Toyota Production System, Industrial Engineering and Management Press, 1983.

- [24] Page, E. S., "An Approach to Scheduling of Jobs on Machines", Journal of the Royal Statistical Society, Vol. 23, 1961, 484-492.
- [25] Palmer, D. S., "Sequencing Jobs Through a Multi-stage Process in the Minimum Total Time - A Quick Method", Operational Research Quarterly, 16, 1965, 101-107.
- [26] Reiser, M., and Lavenberg, S. S., "Mean Value Analysis of Closed Multichain Queuing Networks", Journal of Association of Computing Machinery, Vol. 27, No. 2, 1980, 313-322.
- [27] Sarker B. R., and Harris, R.D., "The Effect of Imbalance in a Just-in-Time Production System: A Simulation Study", International Journal of Production Research, Vol. 26, No. 1, 1988, 1-18.
- [28] Shanthikumar, J. G., and Yao, D. D., "Stochastic Monotonicity of the Queue Lengths in Closed Queueing Networks", Operations Research, Vol. 35, No. 4, 1987, 583-588.
- [29] Spearman, M. L., "An Analytic Congestion Model for Closed Production Systems with IFR Processing Times", Management Science, Vol. 37, No. 8, August 1991, 1015-1029.
- [30] Spearman, M. L., Woodruff, D. L., and Hopp, W. J., "CONWIP: A Pull Alternative to Kanban", International Journal of Production Research, Vol. 25, No. 5, 1990, 879-894.
- [31] Spearman, M. L., and Zazanis, M. A., "Push and Pull Production Systems: Issues and Comparisons", Technical Report 88-24, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1988.
- [32] Stecke, K. E., and Solberg, J. J., "Optimality of Unbalancing Both Workloads and Machine Group Sizes in Closed Queuing Networks of Multiserver Queues", Operations Research, 1984, 883-910.
- [33] Stinson, J. P., and Smith, A. W., "A Heuristic Programming Procedure for Sequencing the Static Flowshop", International Journal of Production Research, Vol. 20, No. 6, 1982, 753-764.
- [34] Suri, R., "A Concept of Monotonicity and its Characterization for Closed Queuing Networks", Operations Research, Vol. 33, 1985, 606-624.

- [35] Villeda, R., Dudek, R., and Smith, M. L., "Increasing the Production Rate of a Just-in Time Production System with Variable Operation Times", International Journal of Production Research, Vol. 26, No. 11, 1988, 1749-1768.
- [36] Wang, H., and Wang, B., "Determining the Number of Kanbans: a Steps Towards non_stock Production", International Journal of Production Research, Vol. 28, No. 11, 1990, 2101-2115.
- [37] Whitt, W., "Open and Closed Models for Networks of Queues", AT&T Bell Laboratory Technical Journal, 63, 1984, 1911-1978.

VITA

Michael Patrick Greco was born on March 17, 1968, in Ridgewood, New Jersey, USA. He graduated from Rutgers University with a Bachelor of Science degree in Electrical Engineering in August, 1990. He has been a student in the Manufacturing Systems Engineering option of the ISE department at Virginia Tech from August 1992 to August 1994 and from August 1995 to the present. He has held Graduate Teaching Assistant positions while completing his Masters degree. He plans to continue his studies in the Manufacturing Systems Option in pursuit of a Doctorate degree.

Michael P. Greco