RATIONALE FOR THE USE OF SUBASSEMBLIES IN PRODUCTION SYSTEMS: A COMPARATIVE LOOK AT SEQUENTIAL AND ARBORESCENT SYSTEMS

by

GUILLAUME PIERRE AMBLARD

S.B., Electrical Engineering
Massachusetts Institute of Technology
(1987)

Submitted to the Department of Electrical Engineering
in partial fulfillment of the Requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1989

Signature of Author [illegible]
Operations Research Center
Electrical Engineering Department
May 24, 1989

Certified by [illegible]
Stephen C. Graves
Professor of Management Science
Thesis Supervisor

Accepted by [illegible]
Amedeo R. Odoni
Professor of Aeronautics and Astronautics and of Civil Engineering
Co-director of the Operations Research Center

DISCLAIMER

This work was supported by DARPA and the Balanced Technology Initiative program under Contract MDA 972-88-C-0027.

[illegible]

Guillaume Pierre Amblard

RATIONALE FOR THE USE OF SUBASSEMBLIES IN PRODUCTION SYSTEMS: A COMPARATIVE LOOK AT SEQUENTIAL AND ARBORESCENT SYSTEMS

by

GUILLAUME PIERRE AMBLARD

Submitted to the Department of Electrical Engineering and Computer Science on May 19, 1989 in partial fulfillment for the Degree of Master of Science in Operations Research

ABSTRACT

In designing a production system for the fabrication of a product, a manufacturing engineer usually has some control over the extent of the use of subassemblies. He/she may have the freedom to decide whether or not parts or groups of parts should be assembled together before being added to the main assembly. In this thesis, we study the pros and cons associated with that choice.

We present a list of qualitative advantages and disadvantages of subassembly use, based on a broad literature search that covered the conventional manufacturing and assembly literature, as well as the literature in areas that exhibit subassembly-like entities such as modules or subroutines.

We also develop a simulation to compare the output rate of the production system configurations which display the most extreme use of subassemblies: the sequential and arborescent configurations. Specifically, we compare the output rate of non-synchronous arborescent and sequential production systems with reliable stations with stochastic processing times for different buffer allocation schemes. This part of the thesis is meant to mark the beginning of a comprehensive study of the quantitative impact of subassembly use in assembly and production systems.

To facilitate such a study, we present algorithms to determine the most arborescent and most sequential assembly sequences possible for a product, based on the algorithmic liaison sequence generation method of De Fazio and Whitney (1986).

Thesis Supervisor: Professor Stephen C. Graves
Title: Professor of Management Science

ACKNOWLEDGEMENTS

I would like to express my gratitude to my adviser, Professor Stephen C. Graves. His guidance and support have been instrumental to the completion of this thesis.

I would also like to thank Dr. Daniel Whitney, my Draper supervisor for his unflagging interest in this thesis. I have appreciated the stimulating discussions I have had with him, Dr. Tom De Fazio and Mr. Richard Gustavson. They provided an engineering insight that broadened my perspective on manufacturing.

My fellow students (no pun intended), Tom Abell, Dan Baldwin, and Suguna Pappu have been a great source of help throughout the year. Specifically, Tom Abell's explanation of the Liaison Sequence Analysis data structure was instrumental to the development of the "Generation of the Most Arborescent Sequences".

I would like to extend a special thanks to the students at the ORC, specifically the French (/Belgian) Connection, who have made these two years of graduate work enjoyable and maybe even fun. Thanks also to Marcia Chapman, Prof. Amedeo Odoni, Prof. Alvin Drake fo, their care and concern, as well as to my roommates at 20 Prospect Hill for their understanding.

Finally, I would like to thank my family, i.e. my parents Marc and Geneviève, my brothers Frédéric and François and my little sister, Caroline for their moral and emotional support. It is nice to know that no matter where I am, whether I am 40, 4000 or 14000 miles away, I can always give them a call.

Last but not least, I would like to thank Au Bon Pain for providing sustenance in moments of need.

TABLE OF CONTENTS

LIST OF FIGURES

"The enormous number of Japanese Buddhist temples built between AD 700 and 1600 were made by flexible production methods. Each of the temples looks quite different. And yet each is put together out of essentially standardized parts, such as beams standardized to width and length; standardized roofing and roof tiles; standardized intervals between the various levels of a pagoda, and so on. The individually distinctive features such as the doors, iron grills, or the ornamentation of the tiles on the roof's edge, were only added at the very end, thus creating brilliant diversity based, however, on true mass production, that is, on standardized parts assembled according to prearranged pattern."

Peter Drucker (1974)

CHAPTER I

INTRODUCTION

1.1) OVERVIEW

Recent declines in productivity, losses in market share and increases in global competition have caused US manufacturers to reconsider their manufacturing practices. All aspects of product and process design are being reevaluated in order to find more efficient ways of producing better quality products.

At the Charles Draper Laboratory, engineers are working on a project called "Strategic Approach to Product Design" aimed at the full integration of product and process design. This thesis is a part of that effort. We consider the part of product and process design, where the choice of subassemblies is made.

In designing a production system for the fabrication of a product, a manufacturing engineer usually has some control over the extent of the use of subassemblies. He/she sometimes has the freedom to decide whether or not parts or groups of parts should be assembled together before being added to the main assembly. In this thesis, we identify and discuss some qualitative and quantitative reasons associated with that choice.

1.2) THESIS OUTLINE

In chapter II, we present a list of qualitative advantages and disadvantages of subassembly use. Due to the broad nature of the word "subassembly", our literature search was as exhaustive as possible and covered a wide spectrum of fields with assembly-like entities. The list includes the advantages of these subassembly-like entities, which are applicable to subassemblies in the manufacturing and assembly sense of the word, as well as the extrapolation of comments on the subassemblies in the sparse manufacturing and assembly literature. In this chapter, we felt the need to distinguish between two types of subassemblies: Subassemblies in a single-product manufacturing environment and subassemblies in a multi-product manufacturing environment.

An initial study of subassemblies in a single-product manufacturing environment is important, because it allows us to focus on the intrinsic structural advantages and disadvantages of using subassemblies. The joint study of subassemblies in a multi-product environment and the benefits of commonality, modularity, cannibalization and standardization that emanate from such a structural organization, while relevant to our discussion, would only muddle our initial analysis.

In a multi-product environment, the quantitative advantages of subassembly use are clear. Commonality, modularity and cannibalization result in substantial savings in inventory and production costs, as well as design time. While there are definite qualitative reasons for use of subassemblies in a single-product environment, it is less clear that there are substantial quantitative advantages to do so.

In an effort to quantify the impact of using subassemblies in a single-product environment, in Chapter III, we compare the output rate of two types of configurations, which display the most extreme use of subassemblies: the arborescent and the sequential (or serial) configurations. Specifically, we compare the output rate of non-synchronous arborescent and sequential systems for two different models of systems' stations.

The first model assumes stations with identical quantitative characteristics, in particular independent and identically distributed processing times. The purpose of this first set of simulations is to evaluate the difference in the output rates of arborescent and sequential production systems and study how that difference varies as a function of the coefficient of variation of the processing-time distribution, the type of distribution and the buffer size between the successive stages.

The second model assumes a balanced system with no buffers and two types of stations: stations with deterministic processing times and stations with exponentially distributed processing times. The purpose of this second set of simulations is to study how the two types of systems attenuate the processing time variability of isolated stations, as well as to develop intuition for the structural differences in the two types of systems.

The basic conclusion of the simulation of these two models is that arborescent and sequential systems do not differ greatly in their ability to attenuate processing time variability. While one configuration or another may be preferable depending on the location and the number of sources of variabilities, no structure is clearly always advantageous. Certainly, the relative output rate of the two types of systems should not be a factor in deciding what kind of system structure to use. That is not to say, however, that arborescent and sequential systems are equivalent, however.

This third chapter of the thesis is meant to mark the beginning of a comprehensive study of the quantitative impact of subassembly use in assembly and production systems.

To facilitate such a study, we present, in Chapter IV, a method to determine the most arborescent and most sequential assembly sequences possible for a given product. Our work is based on the algorithmic liaison sequence generation method of De Fazio and Whitney (1986). This algorithm, which is a simplification of Bourjault's "Elaboration Automatique des Séquences Opératoires,"(1984) generates all of the physically possible liaison sequences for a product by collating the answers to a series of questions, which the engineer must answer regarding the assembly opportunities between related parts.(Whitney et al, 1986)

Finally in Chapter V, we discuss possible extensions to the comparison of sequential and arborescent systems. We also mentioned ways in which quantitative techniques developed in the area of modular design, cannibalization, group technology, inventory management and testing could be used to further our understanding of the impact of subassembly use.

CHAPTER II

QUALITATIVE RATIONALE FOR THE USE OF SUBASSEMBLIES IN ASSEMBLY AND PRODUCTION SYSTEMS

2.1) INTRODUCTION, DEFINITIONS AND CONVENTIONS

A subassembly, as defined in Webster's New Collegiate Dictionary [32], is "an assembled unit designed to be incorporated in a finished product". In mathematical terms, we might define a subassembly as a non-empty subset of parts that either has only one element (i.e. only one part), or is such that every part has at least one surface contact with another part in the subset.[9] However, a random set of touching parts does not necessarily constitute a subassembly for our purposes. A subassembly should possess some element of functionality, which can be identified and tested. It should also be stable, or at least conditionally stable. A subassembly is said to stable if its parts maintain their relative position and do not break contact spontaneously.[9] It is said to be conditionally stable if it becomes stable with the help of fixtures or orientation. In some sense, the essence of a subassembly is its identifiability, its functionality and its stability.

One consequence of the functionality requirement for a subassembly is that, if one builds a product in a sequential line, i.e. by adding parts one at a time to the main assembly, there does not necessarily exist a subassembly at every stage of the assembly. Subassemblies exist only at those stages, where the assembled parts form a functional unit.

In our work we focus on the multiple and separate subassembly assemblies rather than on the one-subassembly assembly. In other words, we are concerned with assemblies, where there exists at some point in the assembly two or more disjoint subassemblies. The reason for this focus is that most of the reasons for use of subassemblies apply to the multiple subassembly case.

The term "multiple and separate subassemblies" does not imply the existence of physically separated stations. In fact, even the assembly of product in a sequential line could include multiple subassemblies. Only in Chapter III do we assume that separate subassemblies are assembled or produced in physically distinct stations, apart from the main assembly. Though in many cases, in a modular system for instance, that will be the case.

In this chapter, we sometimes use "subassemblies" to refer to the production entity where subassemblies are put together. This negligence on our part is due to the fact that it is sometimes difficult to distinguish between the product and the process. The impact of the decision to partition a product into subassemblies only manifests itself in the assembly systems that emanate from the product design.

2.2) DESCRIPTION OF LITERATURE SEARCH METHODOLOGY

The word "subassembly" has many synonyms in various fields: module, subsystem, subroutine, etc,...We try to take full advantage of the large number of these subassembly-like entities in the literature search.

Our search was as exhaustive as possible and covered literature in :

- Mechanical Engineering, more specifically in:

  product design, mechanical design, design theory, automated assembly, manufacturing and production systems, inspection and gaging, standardization and group technology, etc...

- Operations Research and Management Science, more specifically in the areas of : modular design, cannibalization and commonality

- Computer Science, more specifically in the area of:

  centralization/decentralization of computer systems and information systems, architecture and structure of computer programs.

This sweeping approach was particularly necessary, since we found no comprehensive study of rationale for subassemblies, with a few exceptions: for example, Shaftel (1972) in the area of modular design, and Leventer (1976) in the area of decentralized information systems, have established lists of the pros and cons of particular types of subassemblies. In the conventional manufacturing and assembly engineering literature, advantages of subassemblies are only mentioned in passing in the midst of the discussion of other topics.

This chapter presents the advantages of subassembly-like entities, which are applicable to subassemblies in assembly and production systems, as well as the extrapolation of comments mentioned sporadically in the manufacturing and assembly engineering literature.

In the course of our literature review, we realized the need to differentiate between two different types of subassemblies:

- Subassemblies in a single-product manufacturing environment, i.e. subassemblies in a single application

- Subassemblies in a multi-product manufacturing environment, i.e. subassemblies with multiple uses

An initial study of subassemblies in a single-product manufacturing environment is important, because it allows us to focus on the intrinsic structural advantages and disadvantages of using subassemblies. The joint study of subassemblies in a multi-product environment and the benefits of commonality, modularity and cannibalization that emanate from such a structural organization would only muddle our initial analysis.

2.3) Subassemblies in a single-product manufacturing environment

2.3.1) Advantages:

a) Simplification and reduction of the problem:

The partitioning of a product into subassemblies simplifies and reduces the problem. Many assemblies are far too complex to make complete in one pass, therefore it is more efficient to break them down into subassemblies.[20] Fewer production entities result in a simpler, better production system. In fact, Frank Riley (1983) recommends that complex products be designed so that they consist of subassemblies of no more than 12 or 13 parts, which can be combined into final assembly. Furthermore, by considering the decomposition of a product into functional subassemblies at the design stage of a product, one may be able to combine a number of parts together to form one new composite part.[19] The resulting lower part count thus reduces the complexity of the product and contributes to a shorter assembly time and a more reliable product.[1]

b)Ease of overall system optimization

Subdivision of a production system into well-defined, functionally meaningful subassemblies can ease the overall system optimization and planning. Beyond a certain size, a system becomes inefficient and unwieldy. In huge sequential system, there is tendency to optimize locally. However, the combination of optimal subsystems need not result in an optimal performance. The use of subassemblies facilitates the determination and setting of the production load and inventory policy and material requirement for each part of the system that will maximize the overall results. [4],[29]

c) Ease of assembly

Certain products are simply easier to assemble through the use of subassemblies:.[2]

• In stacked assembly for instance, where the parts are placed on top of each other in a layered fashion, it is a good idea to group the parts which are going to form a whole layer together in a subassembly.[5]

• Subassemblies may be a good ground to implement the extreme symmetry or extreme asymmetry of groups of components, which is recommended by designers for ease of positioning and holding.[19]

• Also, it is often useful to isolate sets of components, which form a functional unit. Kahler and Ahm (1984) recommend for instance, that the adjustment components of a product be designed as subassemblies.

d)Enhancement of product quality through facilitation of testing and maintenance

In small functional subassemblies, with accessible components and well defined input and output, faulty components are more easily identified.

• There is *thus incentive for early and frequent testing* of parts. Not only is it more economical to discover defective units in early stages of manufacturing, many times it is impossible to test defective workmanship buried inside a product. Early inspection, as facilitated by separate, functional subassemblies with accessible components can help prevent the cumulation of defects in products.[15]

• The quicker diagnosis of faults is crucial, considering (it is estimated by Starr (1964)) that for complex systems about 50 percent of the system's downtime is devoted solely to locating faults.

• In addition, there are lower scrap and diagnosis costs, since fewer parts are directly implicated by a failure.

e) Independence of parts of the systems:

If the system can be subdivided into independent subassemblies with well defined inputs, outputs and interface with the rest of the production systems, the advantages are considerable. Such a disaggregation:

• *Facilitates the isolation of reworks and of new and special processes*: If part of the assembly line is more problem-prone, or simply newer and more experimental than the rest of the assembly line, it is advantageous to isolate it so as to limit its impact on the rest of the system.

• Entices *local control* and *facilitates* the *determination, isolation and rapid response to local needs.* It may enable the use of less-versatile, more specific, cheaper machines, which are closer to the local subassembly's needs. Generally speaking in a single-product manufacturing environment, isolation of subassemblies enables the use of less-formal methods specific to local needs.[16]

• *Results in a more flexible, expandable and reliable system,* because of the possible decoupling of subassemblies. Deficient machines are more readily replaceable. The capacity of bottlenecks can be increased by the addition of extra machines. Entire subassemblies can be substituted by supply from stock or from a subcontractor. *The overall system is more suited to face changes in product specifications, product demands, quality requirements, input and machine availability as well as unexpected changes in the production environment.*[24]

• *Eliminates many communication and correlation problems.* The task of communication is not difficult when components are few and relations among them are simple.[12]

• *Increases worker motivation.* In a smaller group, it is easier to arouse group spirit. Brainstorming becomes more efficient. Workers are more creative because they can focus their attention on a well-defined system, large enough to be interesting, small enough to be concrete.[16],[12]

2.3.2) Disadvantages:

a) Natural subassemblies may not exist and to create some may require the addition of non-functional parts and unnecessary operations [6]

b) The only possible subassemblies may be awkward with tangling and nested components. Subassemblies may be difficult to grab and handle. Bringing large subassemblies into correct alignment may be considerably more complicated than adding pieces one part at a time. Furthermore, it may not be possible to keep the orientation of subassemblies constant throughout the assembly sequence.[19],[7]

c) Subassemblies are not always structurally sound. They may be more prone to damage. Once placed in the assembly, the parts of the subassemblies should be designed to stay in position until the assembly is completed without the aid of external "fingers" or locaters, which complicate the assembly.[11] This is not always possible however.

d) The part transfer between separate subassemblies is quite complex and is more likely to be a source of problems.

e) The use of physically separated subassemblies may result in higher costs of overhead items such as storage, material handling, floor space, electricity, air conditioning, etc.., especially so if use of subassemblies results in a multi-facility production system.

f) Use of physically separated subassemblies may result in a less integrated system. The coordination of all the activities may be quite complex. Fewer people contribute new ideas to the overall system and plant-wide construction and manufacturing practice are not as easy to implement. It is harder to make use of cross-trained workers, as it is difficult to use people idle in one subassembly in a second subassembly.[16]

2.4) Subassemblies in a multi-product manufacturing environment

In a multi-product environment, by designing a product with a large number of subassemblies, a manufacturer can reap the benefits of modularity, commonality and cannibalization

Modularity consists of the use of different variants of a subassembly in several products, where usually the total number of variants is smaller than the number of products.

Commonality refers to the common use by several products of the one same subassembly.

Finally *cannibalization* is the practice of stripping a malfunctioning product of its functioning subassemblies for the emergency maintenance of other products.[8]

It is rare to find a firm that produces only a single type of product. Today, competition is so keen and the market so segmented that most manufacturers produce a variety of products.[25] Commonality, modularity and cannibalization stem from a growing interest on the part of manufacturers to minimize the overall manufacturing, inventory and maintenance cost of a line of products.[22]

The underlying assumption in the use of these three techniques is that *subassemblies are* going to be *standardized* in order to meet a greater variety of functional and physical requirements.[30] There is therefore a trade-off involving the disutility of not providing each product/customer with a subassembly/item fitting its exact requirements versus economies of scale achieved in producing and inventorying a common item with multiple applications.[28]

The advantages and disadvantages of standard subassemblies, modularity, commonality and cannibalization are listed below:

2.4.1)Advantages

a) Economies of scale result from the reduction in product variety. Increased production volume of fewer types of different products opens the possibility for large-scale machinery. The use of large-scale machineries will in many industries lead to a decrease of the manufacturing costs per unit.

b) If the same machinery had been used for the production of several variants of a subassembly, the move to standard subassemblies will decrease the number of set-ups and change-overs needed and considerably lower the costs per unit [3]

c) If the same machinery can still be used, after a switch to standard subassemblies, lower (or equal) capital cost will result. If machines of type x were used in n different production systems, less than n machines may be needed if the operations of type x are consolidated.[3]

d) Addition to the production line is simplified. Products are rarely produced from scratch; they are usually modifications and synthesis of previous design. The repeated use of certain subassemblies in products of different generations results in lower product and process design time

e) Similarly, a greater product mix is possible.[30] Consider a plant that produces k different subassemblies and three varieties of each . In an idealized setting where a product can be made from any combination of any type of subassemblies, the total number of products would be 3^k, i.e. 531,441 if $k = 12$. While obviously there are never so many theoretically possible products, the flexibility gained from use of standard subassemblies or modules is often substantial.[28],[31]

f) Fewer items or subassemblies must be inventoried in any plant. The management of inventory is simplified. In addition, less room is required, since the storage space for a few subassemblies is much smaller than space for storing several complete products.[25]

g) Consolidation of the inventory of standard subassemblies results in lower overall safety stock, because the uncertainty of the level of production diminishes. The inventory costs are thus lower.

h) Fewer types of standard parts result in a steeper learning curve, tighter quality control and standard testing procedures.[25]

i) Repair is simplified by ease of replacement. Repairman need only carry a few types of standard parts to quickly repair products. [25],[26]

j) Cannibalization is also simplified. Consequently, scrap costs are lowered and the vulnerability of the system to machine failures and variability in processing diminishes.

2.4.2) Disadvantages:

a) More parts than required may be used. These excess parts increase the cost of production, transportation and handling.[25]

b) Production scheduling and interconnecting of production modules may be more complicated.

2.5) Conclusion:

In this chapter, we have presented the results of our study on qualitative rationale for subassembly.

In a multi-product environment, the quantitative advantages of subassembly use are clear. Commonality, modularity and cannibalization result in substantial savings in inventory and production costs, as well as design time.

While there are definite qualitative reasons for use of subassemblies in a single-product environment, it is less clear that there are substantial quantitative advantages to do so.

In an effort to quantify the possible impact of using subassemblies in a single-product environment, in the next chapter, we compare the output rate of the two types of configurations, which display the most extreme use of subassemblies: the arborescent and the sequential configurations.

References for Chapter II

(Please note that a more extensive bibliography is listed at the end of the thesis).

[1] Andreasen, M.M., S. Kahler, and T. Lund, 1982 "Design for Assembly - an integrated approach," *Assembly Automation,* August.

[2] Andreasen, M.M., S. Kahler, and T. Lund, 1983 Design for Assembly, IFS Publications Ltd. Bedford, U.K.

[3] Bongers, C.1980. Standardization: Mathematical Methods in Assortment Determination. Martinus Nijhoff Publishing, The Hague.

[4] Candea, D. I. , 1977. *Issues of Hierarchical Planning in Muli-Stage Production Systems*, Ph.D. Thesis M.I.T. Sloan School of Mgt.

[5] Dewhurst, P. and G. Boothroyd, 1987. "Design for Assembly in Action," *Assembly Engineering*, January.

[6] Graves, Stephen C., 1988. Memo to Dr. James Nevins of C.S. Draper Laboratory, July 21.

[7] Hartley, John., 1984."Systematic approach to ease assembly," *Assembly Automation,* August.

[8] Hirsch, W.M., M. Meisner, C. Boll, 1968. "Cannibalization in Multi-Component Systems and the Theory of Reliability," *Naval Res. Log. Quart.* 15, pp. 331-359.

[9] Homem de Mello, L.S., A.C. Sanderson, 1988. *Automatic Generation of Mechanical Sequences,* The Robotics Institute, Carnegie Mellon Institute, Pittsburgh, Pa.

[10] Huff, Sidney L. A , 1978. *Case Study of Centralization vs. Decentralization and Stages of Growth in the Data Processing Function within the Canadian Banking Industry,* Working Paper, Sloan School of Management, MIT.

[11] Jenkins, R.F., 1965. "Why Automatic Assembly?" Metal Production, August 18.

[12] Johnson, R.A., W.T. Newell, Roger C. Vergin, 1974. Production and Operations Management: A systems' concept. Houghton Mifflin, Boston.

[13] Kahler, S. and T. Ahm, 1984. "Design for assembly - a case study," *Assembly Automation,* August.

[14] Klein, Charles C. *Generation and Evaluation of Mechanical Assembly Sequence Alternatives,* M.I.T. S.M. Thesis, M.E. Department. C.S. Draper Laboratory Report No 934.

[15] Kennedy, C.W., E.G. Hoffman, S.D. Bond. Inspection and Gaging. Industrial Press, Inc,. New York, N.Y.

[16] Lazear, T.J, J.F. Rockart, 1976. *Organization of the System Functions in Engineering and Construction.* Report CISR-14 P852-76 Sloan School of Management.

[17] Leventer, Joav Steve, 1976. *Centralization versus Decentralization of Information Systems: A Framework for Decision Making,* Master's Thesis, Sloan School of Management, MIT.

[18] Lund, Thomas, 1984."Integration is the theme for design," *Assembly Automation,* August.

[19] Owen, Tony, 1985. Assembly with Robots, Englewood Cliff, NJ: Prentice Hall.

[20] Rathmill, Keith. Robotic Assembly. UK: IFS Ltd, Springer Verlag, 1985.

[21] Riley, Frank J, 1983. Assembly Automation: A Management Handbook, N.Y., N.Y.: Industrial Press.

[22] Rutenberg, D.P, T.L.Shaftel, 1971 "Product Design: Subassemblies for Multiple Markets," *Management Science,* Vol. 18, No 4, Part I, December, pp. 220-230.

[23] Rutenberg, D.P., 1971 "Design Commonality to Reduce Multi-Item Inventory: Optimal Depth of a Product Line," *Operations Research*, Vol. 19, March-April, pp. 491-509.

[24] Schell, Roger, 1981. *Dynamic Reconfiguration in a Modular Computer System,* Ph.D. Thesis, MIT Project Mac.

[25] Shaftel, Tim, 1972. "How Modular Design Reduces Production Costs," *Arizona Review*, Vol. 21, No 6-7, June-July, p.4.

[26] Siewiorek, D.P. and M.R. Barbacci, 1974. *Some Observation on Modular Design Technology and the Use of Microprogramming*, Department of Computer of Science, Carnegie Mellon.

[27] Starr, M.K., 1964. Production Management: Systems and Synthesis. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[28] Starr, M.K., 1965. "Modular Production - A New Concept," *Harvard Business Review,* November-December, pp. 131-142.

[29] Starr, M.K., 1976. The Practice of Management Science. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[30] Tersine, R.J., 1980. Production/ Operations Management: Concepts, Structure and Analysis, School of Business, Old Dominion University.

[31] Whitney, D.E., et al., 1986. *The Strategic Approach to Product Design*. The Charles Draper Laboratory, Cambridge, Ma.

[32] *Webster's New Collegiate Dictionary*, 1977. G. & C Merriam Company, Springfield, Ma

CHAPTER III

COMPARISON OF THE OUTPUT RATE OF ARBORESCENT AND SEQUENTIAL SYSTEMS

3.1) PROBLEM PRESENTATION AND SIMULATION ENVIRONMENT DESCRIPTION

3.1.1) Introduction

In Chapter II, we pointed out that, while there are definite qualitative reasons for use of subassemblies in a single-product environment, it is less clear though that there are substantial quantitative advantages to do so.

In this chapter, in an effort to quantify the possible impact of using subassemblies in a single-product environment, we compare the two types of production system configurations which display the most extreme use of subassemblies: the sequential and arborescent configurations.

In a *sequential system*, the production system has a linear structure and components are added to the main assembly one at a time. In a *purely arborescent system*, the production system takes on a tree-like structure, where pairs of components are combined into subassemblies, which are then joined together two at time until the assembly is completed. While the use of subassemblies is kept to a minimum in the sequential configuration, it is maximized in the arborescent configuration.

For ease of reference, we define:

- AS to be the purely *Arborescent System*
- SS to be the purely *Sequential System*

Our analysis is limited to purely arborescent and purely sequential production system. The number of parts will thus always be a power of 2 greater or equal to 4 (i.e. 4, 8, 16,32, etc...). What we take to be an AS is actually an arborescent system of order 2.

Few systems are purely arborescent. However, consideration of hybrid systems, i.e. systems that are partially arborescent and partially sequential would only muddle, at least at this time, our analysis of the differences between the two extreme configurations.

Before starting this simulation work, we expected that purely arborescent systems would yield a higher output rate. We wanted to verify that supposition, in order to see if the extensive use of subassemblies in a non-synchronous and reliable system actually raised the overall output rate. Only if that supposition happened to be true would it be necessary to consider hybrid systems, to check if the output rate should be a criterion in the decision to use physically separate subassemblies.

Our goal in this chapter is two fold:

1. To compare the *output rate* of AS and SS as a function of the number of parts, for different *buffer allocation* schemes (including the case where no buffer is allocated) and for different *stochastic models* of the operations.

2. To study the relative *capability* of the two types of systems to *attenuate* the *variability* of isolated stations.

3.1.2) Examples of products, that are both sequential and arborescent

The comparison of purely AS and purely SS is an intermediate step in the possible establishment of output rate as a criterion in the decision to use subassemblies. For, as mentioned earlier, if the output rate of AS is indeed greater than SS then we will evaluate the output rate of hybrid systems.

One should, therefore, not be concerned with the fact that few products can actually be produced in both purely arborescent and purely sequential systems. These types of products do exist though. We present two of them now. Readers, not familiar with the notion of liaisons and parts-trees, may wish to skim Section 4.1 and Section 4.2.1a) for a brief introduction to liaisons, liaison diagrams and parts-tree diagram representations.

a) Example #1: 4-part electrical socket

The 4-part electrical socket, shown in Figures 3.1a and 3.1b and represented by the liaison diagram and the precedence relations list shown in Figures 3.1c and 3.1d, can be assembled in either purely sequential or purely arborescent systems. Specifically, it can be assembled in five different ways, i.e. in four purely sequential and one purely arborescent. The parts-tree diagram representation of the five possible configurations are shown in Figure 3.1e.

b) Example #2: 8-part product

The 8-part product, shown in 3.2a and represented by the liaison diagram and the precedence relations list shown in Figures 3.2b and 3.2c, can also be built in both types of systems. The parts-tree diagram representation of one purely sequential and one purely arborescent configuration are shown in Figure 3.2d.

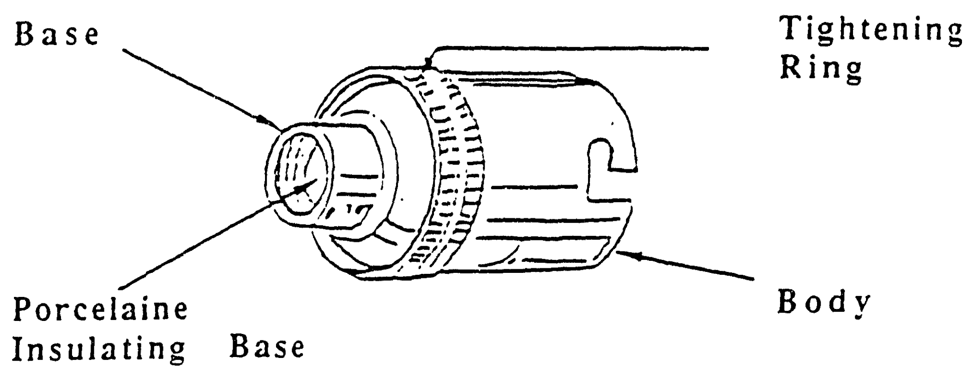


Figure3.3a: Schematic representation of an *electrical socket.* This 4-part product is the first of two sequarborescentable products presented.

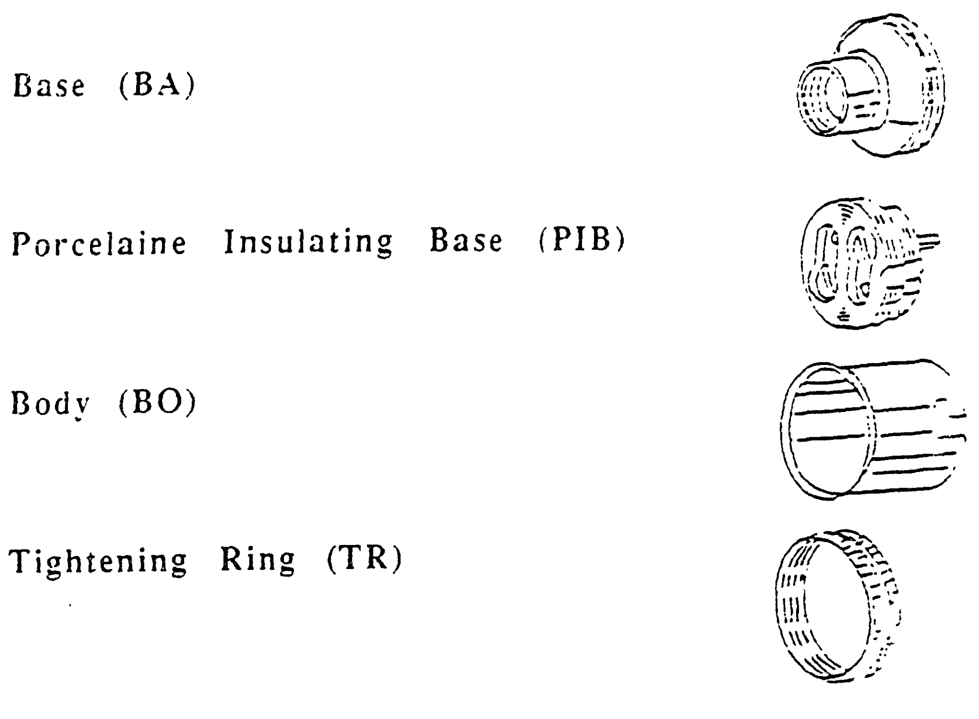


Figure3.3b: Representation of the component parts of the electrical socket example shown in Figure 3a.

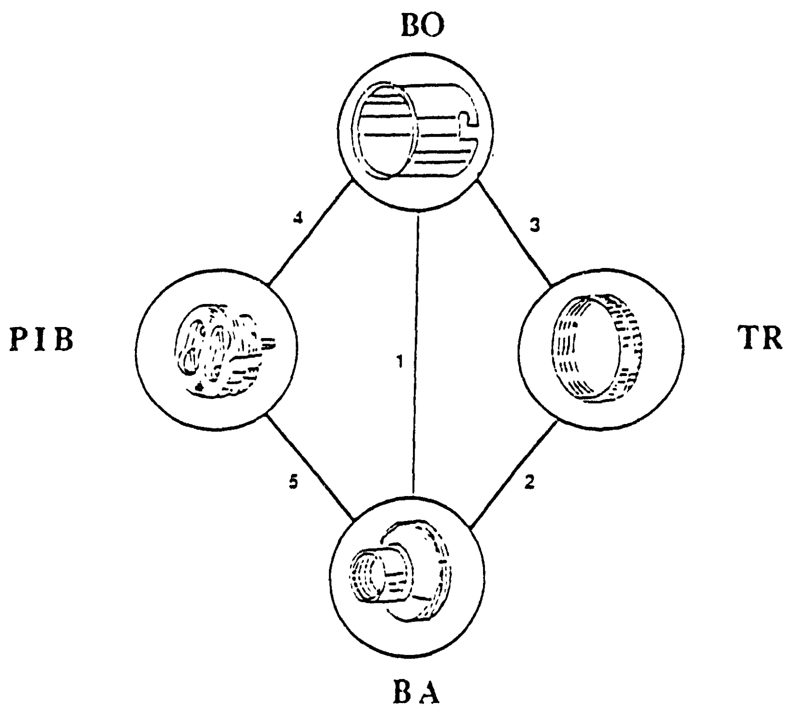


Figure3.3c: Liaison diagram of the electrical socket example shown in Figure 3a

- $1 \geq 2$
- $3 \geq 2$
- $4 \geq 1$
- $5 \geq 1$
- $5 \geq 2$

Figure3.3d: List of precedence relations between the liaisons of the liaison diagram shown in Figure 3c.
$i \geq j$ means that Liaison i must precede or be done simultaneously with Liaison j

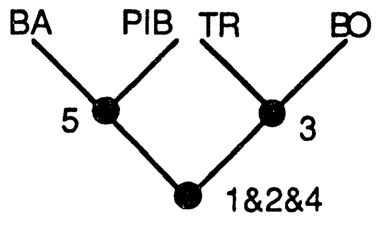


Figure 3.1.e.1:

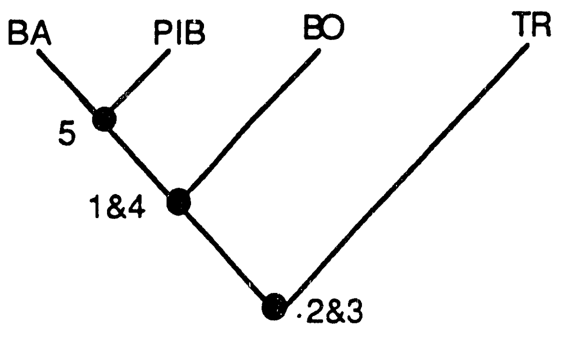


Figure 3.1.e.2:

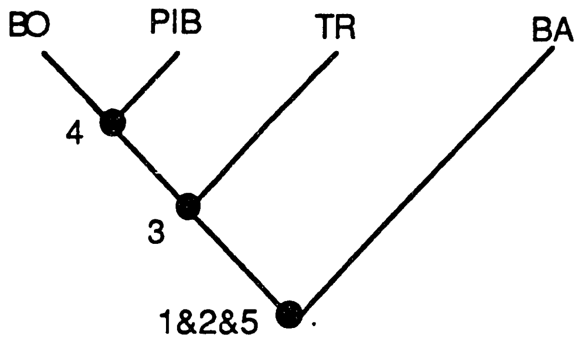


Figure 3.1.e.3:

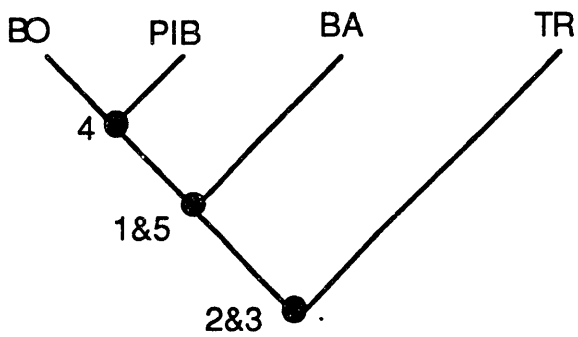


Figure 3.1.e.4:

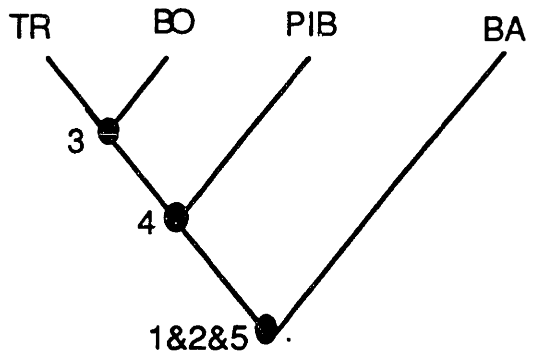


Figure 3.1.e.5:

Figure 3.1.e: Parts tree representation of the electrical socket shown in Figures 3.1a and 3.1b and described in the liaison diagram and the precedence relations of Figures 3.1c and 3.1d. The parts tree in Figure 3.1e.1 represents a purely arborescent assembly, whereas the other parts trees represent purely sequential assemblies.

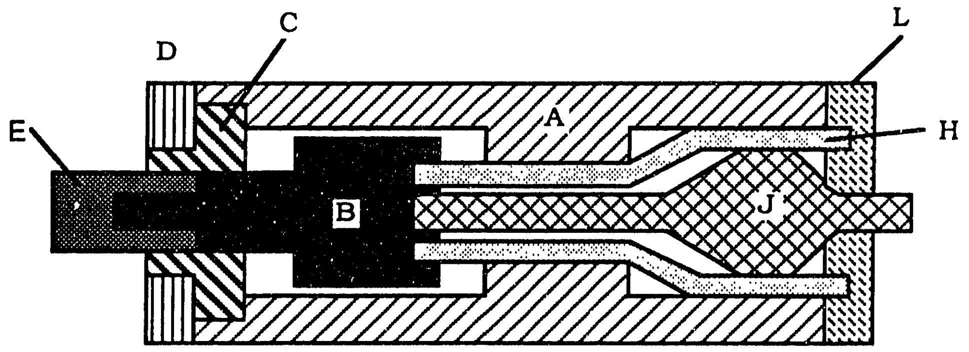


Figure 3.2a:

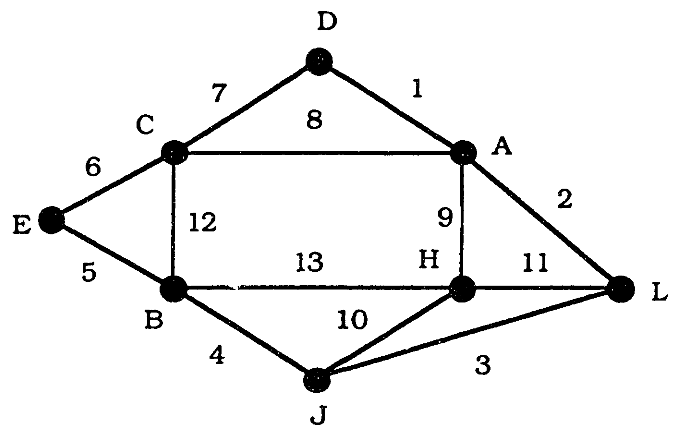


Figure 3.2b:

Figure 3.2a,b: Schematic and liaison diagram of a simplified version of the example of Assembly From Industry (A.F.I.) of De Fazio and Whitney (1986).

$1 \geq 2 \& 3 \& 4$
$1 \geq 4 \& 7 \& 12$
$1 \geq 9 \& 13$
$1 \geq 7 \& 12 \& 13$
$1 \geq 6 \& 8$
$2 \geq 3 \& 4$
$2 \geq 11 \& 13$
$2 \geq 4 \& 8 \& 11 \& 12$
$3 \geq 2$
$3 \geq 11$
$4 \geq 1 \& 2 \& 3$
$4 \geq 2 \& 3 \& 8$
$4 \geq 1 \& 9 \& 10$
$4 \geq 8 \& 9 \& 10$
$4 \geq 11 \& 13$
$4 \geq 2 \& 3 \& 13$
$4 \geq 2 \& 8 \& 12$
$5 \geq 1 \& 2 \& 3 \& 4 \& 6$
$5 \geq 6 \& 8$
$5 \geq 1 \& 9 \& 13$
$6 \geq 1 \& 7 \& 8 \& 9 \& 12 \& 13$
$7 \geq 1$
$7 \geq 6$
$8 \geq 1$
$8 \geq 2 \& 3 \& 4$
$8 \geq 4 \& 12$
$8 \geq 9 \& 13$
$8 \geq 12 \& 13$
$9 \geq 2$
$9 \geq 4 \& 8 \& 12$
$9 \geq 13$
$10 \geq 11$
$10 \geq 2 \& 3$
$10 \geq 4$
$11 \geq 2$
$11 \geq 3 \& 4$
$12 \geq 1 \& 2 \& 3 \& 4$
$12 \geq 8$
$12 \geq 1 \& 9 \& 13$
$13 \geq 4$
$13 \geq 1 \& 9$
$13 \geq 8 \& 9$
$13 \geq 2 \& 8 \& 12$

Figure 3.2c: Precedence relations for the modified version of the example of Assembly From Industry (A.F.I.), described in the schematic of Figure 3.2a and in the liaison diagram of Figure 3.2b. $i \geq j$ means that Liaison i must precede or be done simultaneously with Liaison j

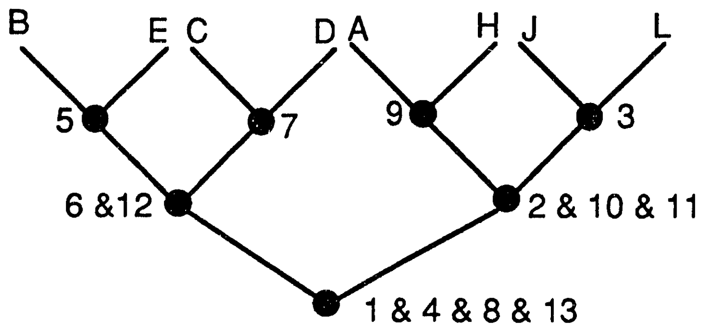


Figure 3.2d.1:

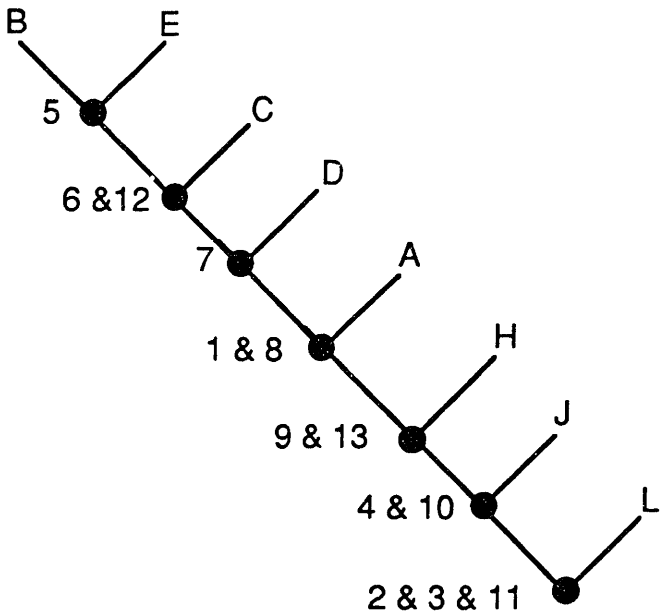


Figure 3.2d.2:

Figure 3.2d: Parts tree representation of the product described in Figures 3.2a, 3.2b and 3.2c. We have represented two ways of assembling the product: one purely arborescent (Figure 3.2d.1) and one purely arborescent (Figure 3.2d.2)

3.1.3) Modelling Assumptions

In our work here, we make the following assumptions about AS and SS:

• *All assembly operations* are conducted at *different workstations.* This assumption insures that the two systems will have the same number of workstations.

In this section where we assume that each of the operations is performed at a different workstation, we will use the "production system as seen from above" type of representation shown in Figure 3.3. The *squares*/rectangles represent *stations*, the *letters* inside the squares/rectangles represent the *assembly operation* performed at the station, and the *arrows* or arcs represent *part transfers or flows.* We sometimes refer to the station where Oi is conducted, as station Oi.

• Whenever possible, we assume that *all* n *assembly operations* in both AS and SS are attributed the *same* quantitative *characteristics,* in particular the same processing time distribution. For all assembly operations, we let t denote the stochastic processing time to complete the operation, EVT denote the expected value of the processing time, and VPT denote the variance of the processing time.

We make the assumption of identical assembly operations for simplicity and ease of analysis. We realize that in actuality, individual operations are functionally different in the two systems and it is difficult to establish a one to one correspondence between them. Oi in AS does not necessarily correspond to Oi in SS. We can see for instance that O4 in Figure 3.3a is a very different operation from O4 in Figure 3.3b. In Figure 3.3b, it combines parts 5 and 6; in Figure 3.3a, it adds parts 5 to parts 1, 2, 3 and 4. The assumption of identical assembly operations permits the study of the differences between the two types of configurations, all other factors being the same.

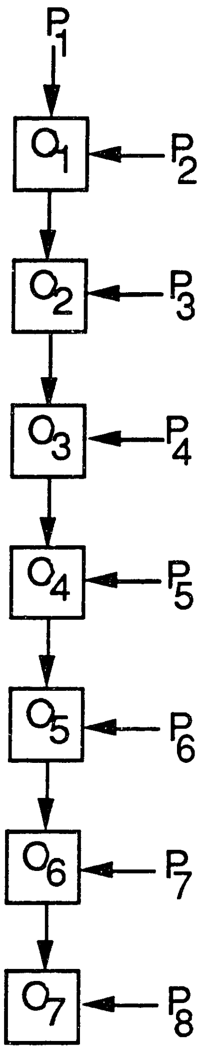


Figure 3.3a:

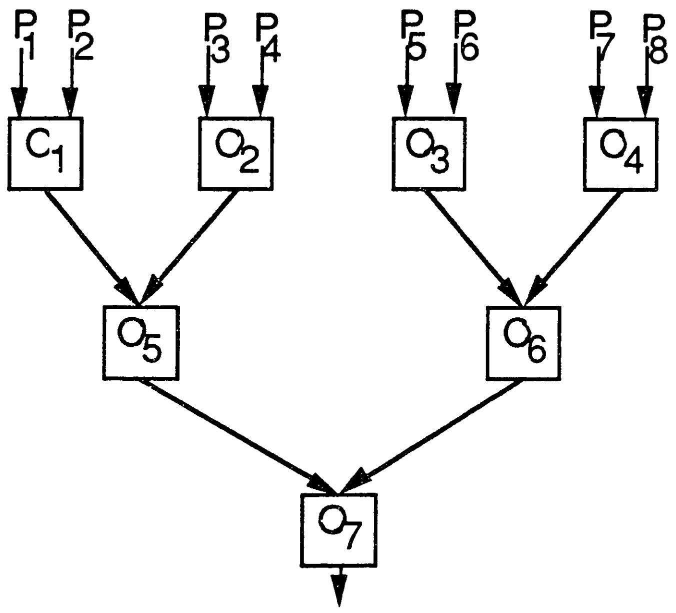


Figure 3.3b:

Figure 3.3a,b: "Production system as seen from above" representation of a *sequential* (Figure 3.3a) and of an *arborescent* system (Figure 3.3b) for an 8-part product.

In the latter part of our modelling work, we relax the assumption of stations with independent identically distributed processing times in order to observe the nuances of the structural differences of AS and SS, in particular to study the relative capability of the two types of systems to attenuate the effect of isolated sources of variability

• We assume *non-synchronous part-transfer*, i.e. each workpiece is allowed to nove to the next station as soon as processing at the current station has been completed assuming the buffer at the receiving station is not full). The case of synchronous part ransfer, which assumes simultaneous transfer of all the parts in the assembly system, would not be insightful, since it would yield identical results for the two types of systems. For instance, for both systems, the expected interarrival time, i.e. the time between the completion of parts at the last station, would simply be E(max(T1, T2, ...,Tn)), where Ti is the processing time for the assembly operation at the ith station. The interarrival time is in fact the time between successive part transfers in a synchronous system.

3.1.4) Description of the mechanics of part transfer through the production system

a) When no buffer is allowed between stations:

In AS, each of the stations has two inputs and consequently also two input slots. An *input slot* is a term we introduce to facilitate the description of the mechanics of part transfer through the production system. A given input slot of a station is *full* and thus cannot receive further input from the station upstream (which may therefore eventually be blocked) in three different states of the station:

1. The station has already received the input corresponding to this slot, but is still awaiting the arrival of its other input. In this state, the *station* is *inactive* and is said to be *partially starved.*

2. The station has received its two inputs and is processing them. In this state, the *station* is said to be *active.*

3. The station has finished processing the unit corresponding to its two present input slot parts, but cannot empty its input slots by transferring its finished unit downstream. This occurs, because the station downstream has not emptied the targeted input slot. In this state, the *station* is *inactive* and is said to be *blocked.* Note that as soon as the station downstream empties the targeted slot, the station will transfer the subassembly and be *totally starved* until the instant it receives the input parts from the two upstream stations. The station will then become *partially starved* (if it receives only one input) or *active* (if it receives two inputs).

In SS, the stations operate in a similar manner with the exception that they cannot be partially starved, since stations in SS only have one input and consequently only one input slot.

The station, which performs the last operation in the final assembly, i.e O7 in Figures 3.3a and 3.3b for instance, is never blocked in either system.

Furthermore, the stations, which are the most upstream - i.e. the stations that assemble parts, rather than subassemblies of parts - can only be in two possible states: the *active* state and the *blocked* state. We assume these stations are always supplied with the necessary raw material or input parts. Consequently they can never be *starved.* These stations are station 1 in Figure 3.3a and stations 1, 2, 4 and 5 in Figure 3.3b.

Our systems operate in a dual "push"and "pull" mode. They can be interpreted as "push" systems with unlimited availability of raw materials or input parts for the stations and unlimited demand for the finished product. The material always moves forward as soon as an input slot is available; in that sense, the material is not "pulled" by demand. However, the systems can also be interpreted as operating in a "pull mode", where stations are blocked until notice from stations downstream.

b) When buffer is allowed between stations

Buffers are often added between stations to alleviate the dependencies (and the resulting mutual slowing down) between the system's stations, as well as to minimize the effect of the uncertainty (variance) of the operations' processing times. Buffers can attenuate short transients, but are incapable of overcoming long-term imbalances.

If we have a buffer of size n after every station, we are in effect adding n spaces to the stations' targeted input slots for a total of n+1 space input slots. The implications of such a measure for our system are the following:

- When a station is partially starved (i.e. it has only received one of its two input parts), then the upstream station, which has already submitted one piece and completed the next one, is no longer blocked. In fact, this station is not blocked until the n+1 spaces of its targeted slot are full and it is ready to dispose of another unit.

- Similarly when a station is active or blocked, the stations directly upstream are no longer blocked if they are done processing the next unit and won't be until the n+1 spaces of their targeted slot are full and they are ready to dispose of another unit.

3.1.5) Literature Review

This chapter of the thesis compares the production rate of sequential and arborescent production systems with reliable stations with stochastic processing times for different buffer allocation schemes. To date, no such comparison has been conducted.

a) Sequential systems

The literature on sequential systems - otherwise referred to as "series of workstations", "serial production lines", "serial assembly lines"- is quite vast. Of particular interest to us, the output rate and the effect of buffer on the output rate of sequential systems with reliable stations with stochastic processing times has been studied extensively.

Past work on such sequential systems has involved analytical solutions as well as simulation studies and the development of empirical formulas.

As mentioned by Muth (1973), most of the analytical work has used an approach in which the system is described at any time t by its state vector X(t), which contains information about the systems' stations and buffers. The state of a station can be either 0 (idle or starved), 1 (busy/active) or 2 (blocked), whereas the state of a buffer is simply the number of items in the buffer. These state-based models assume that all the processing times are exponentially distributed, so that the process X(t) is a Markov Process whose state probabilities, in equilibrium, satisfy a system of linear equations. The equilibrium state probabilities can be obtained by solving this system of equations either analytically as done by Hatcher (1969), Hunt(1956), and Patterson (1964), or numerically as done by Hillier and Bolling(1967).

There are two problems with this approach:

1. Solutions are restricted to the case of stations with exponential processing times

2. The number of states, and consequently the number of simultaneous equations to be solved, in systems with no buffer at all, grows asymptotically as n^m, where $m = (3 + \sqrt{5})/2 \approx 2.62$. A bufferless, 10-station system, for instance, has 6765 states. The problem worsens when buffers are allowed since the number of states grows rapidly with buffer sizes. Because of this size problem, analytical solutions have only been obtained for $n \leq 3$ (Hunt 1956) and numerical solutions for $n \leq 6$ (Hillier and Bolling 1967).

Simulation offers a way to overcome these weaknesses. Many have taken this approach and not limited themselves to exponential distribution: Anderson and Moodie (1969), Barten (1962), Knott(1970) and Muth(1973) for instance. Muth's 1973 work is particularly relevant to this thesis. He finds lower and upper bounds for the output rate of n-station sequential system and shows how the difference between the two bounds increases with the number of stations and with the coefficient of variation (COV). The COV, a measure of the variability and the lack of memory of the processing time, is defined as the ratio of the standard deviation and expected value of the processing time. He also illustrates the fact that the loss of capability due to interference between workstations, in an non-synchronous line, occurs in the first few stations, long lines being only slightly worse than short ones. His later work (1979) confirmed Yamazaki and Sakasegawa's (1975) assertion that any serial line has a dual line which is identical except that the direction of material flow is reversed- the first workstation in the primal line is the last in the dual line, etc...- and *the output rate of a line and its dual line are identical.*

Simulations have also been widely used to study the effect of the size and the location of buffer spaces in a production system. Buffer space is often used to compensate for the variability of the processing times. While many, including Muth (1973) and Hatcher (1969), have studied this issue, Conway et al. (1988) provide the most extensive coverage of the matter. Their work can be summarized as follows:

1. For balanced systems with identical stations,

    • To achieve a given target production capacity, the buffer capacity should be proportional to the coefficient of variation. Buffer capacity equal to 10 times the coefficient of variation recovers 80 to 85% percent of the capability lost due to variability in these systems. As shown earlier by Hatcher (1969), the improvement diminishes rapidly with increased buffer size.

    • The best buffer allocation is symmetrical and evenly distributed if possible. In cases where the buffer cannot be allocated evenly, center placement, i.e. allocation of buffer spaces in the center of the production line, is significantly better than end-placement, i.e. allocation of buffer spaces at the end of the production line . Near-center placement, though, is almost as good as exact-center placement.

2. For unbalanced systems, buffers provide less increase in capacity. The preferable position for buffers is around the bottleneck stations.

Incidentally, in the course of these experiments, Conway et al. found that the output rate of the line was sensitive to the form of the probability distribution of the processing times, but that the optimal allocation was only sensitive to the standard deviation.

b) Arborescent systems

While the literature on sequential systems is abundant, there is little relevant literature on arborescent systems.

Researchers, such as Freeman and Jucker (1967), Svetska and Nair (1972), Buxey (1974) and Pinto et al (1981) all studied multi-stage parallel systems and the production gains and labor cost reduction associated with use of parallel stations. These systems, though, are radically different from arborescent systems, because the stations in parallel systems produce identical outputs. The station downstream from two parallel stations is "unstarved" as soon as *one of the two stations (not both)* has completed the processing of its current piece. In an arborescent system, both upstream station need to have completed their tasks, in order for the station downstream to be "unstarved".

Similarly, all the work in the area of open queueing networks, even Smith and Daskakaki's "Buffer Space Allocation in Automated Assembly Lines" (1987), which compares the output rate of series, merging and splitting topologies for different buffer allocation schemes is not relevant to this thesis. This is because, the stations in these open queueing networks have decoupled inputs, i.e. a station can start service on an input piece, even if it has not received pieces from all of its input sources.

The only work with some relevance to our arborescent systems has been done in the area of Assembly/Disassembly networks. In particular, Ammar and Gershwin's "Equivalence Relations in Queueing Models of Assembly/Disassembly Networks" (1987) gives us insight on the comparison of three-station sequential and arborescent systems.

After introducing the notion of structural equivalence - two systems are structurally equivalent if the corresponding stations have the same processing rate, the same buffer capability and corresponding buffers are connected to corresponding stations, although parts do not necessarily move in the same directions - Ammar and Gerschwin proved that structurally equivalent systems behave almost identically in a probabilistic sense.

Ammar and Gerschwin showed the equivalence of the systems shown in Figure 3.4a and 3.4b. While the Figure 3.4b system is different from the arborescent equivalent of the system, shown in Figure 3.4c, we can draw interesting conclusions from this equivalence of the Figure 3.4a and Figure 3.4b systems. In particular, *if stations two and three are identical, then the three-station sequential and arborescent systems should have very closely related output rates.*

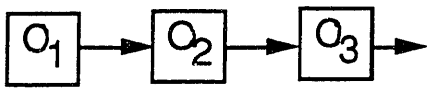


Figure 3.4a:

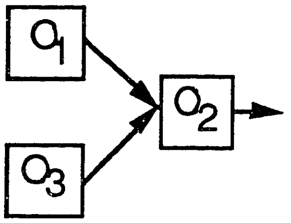


Figure 3.4b:

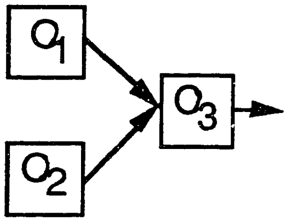


Figure 3.4c:

Figure 3.4: Ammar and Gershwin (1987) showed that the behavior of the systems in Figures 3.4a and 3.4b are closely related. Consequently, if O2 and O3 are identical, then the behavior of systems in Figures 3.4a and 3.4c must also be closely related.

Unfortunately, we cannot extrapolate from Ammar and Gershwin's work for systems with more than three stations. The arborescent and sequential systems have radically different structures for n greater than three. A sequential system with n stations only has two stations which are either never starved or never blocked for instance, whereas there are (n+3)/2 such stations in a n-station arborescent system.

3.1.6) Simulation tool

In order to overcome the size problem and the restrictive assumptions of the analytical solutions available for the study of non-synchronous systems, we decided to resort to simulation to help us quantify and analyze the differences between AS and SS. Once it was decided to simulate dynamic, stochastic models of sequential and arborescent production systems, a simulation language needed to be chosen. We chose to use PC SIMCRIPT II.5 on the AT&T 386.

SIMSCRIPT II.5 is an event-oriented or process-oriented simulation language considered by many to be one of the most powerful simulation language available. [16] It presents many advantages over other simulation languages, such as SIMAN and general-purpose languages such as FORTRAN:

• it is the only major simulation language with a package for performing statistical analysis of simulation output data.

• its English-like and free-form syntax make its programs easy to read and almost self-documenting.

• it provides a natural framework for simulation modeling, which facilitates the modification of models. Its building blocks are more closely akin to simulation modeling than those in a language like FORTRAN. [15]

• it automatically provides all the features needed in programming a simulation, resulting in a considerable savings in programming time. These features include:

1. Generating random numbers from a uniform distribution
2. Generating random variables from a specified distribution
3. Advancing simulation time
4. Determining the next event from the event list and passing control to the appropriate block of code. Languages like FORTRAN are inadequate, for instance, for process-oriented simulation, which require a co-routine structure.
5. Adding records to, or deleting records from, a list
6. Collecting and analyzing data
7. Reporting the results. It is equipped with an animated interactive graphics called SIMGRAPHICS [4]
8. Detecting error condition. Its on-line debugger, which makes full use of the system's multi-tasking and multi-window environment is an amazing time saver

• Because of the diversity of the statements available, SIMSCRIPT maintains a level of flexibility, unusual for a simulation language. Normally one gives up a certain amount of flexibility in using a simulation language instead of a general-programming language like FORTRAN. But that proved not be the case.

• Finally, the PC version of SIMSCRIPT II.5 is available to universities for $500. This proved to be a deciding factor, because even if one has access to a simulation language on a mainframe, the cost of mainframe simulations is usually prohibitive.

3.1.7) Simulation conditions

In our simulation work, we adhere to the following running conditions to insure the comparability of the runs and the validity of our results:

- We select 10 time units as the standard mean processing time for all the stations in the balanced systems and for the bottleneck stations in the unbalanced systems. Consequently, we will always be able to compare r(AS) and r(SS), the output rates of AS and SS to their theoretical capability of 0.1 unit/time unit.

- We start every run with an initialization period at the end of which we will reset the output counters but keep the state of the system (input slots and buffers). This measure eliminates the transient difference in production rate of AS and SS due to their different assembly branch lengths. It also limits the bias due to the starting conditions. In the systems with no buffer, the number of possible states of the stations is limited. The initialization period should thus be proportional to (n-1), the number of operations/ stations. In the systems with buffer, the transient period should be longer and its length proportional to (n+b-1), where b is the total number of buffer spaces. The initialization period should be long enough to exclude observations from the transient period and short enough to limit the number of wasted observations. After experimenting with different initialization period lengths and studying the sensitivity of the results to the length of the initialization period, we decided for simplicity of bookkeeping to set the length of the initialization period to be equal to the length of a single subrun or *batch*.

- Similarly, we decided to set the length of the "data collection period" of each batch to be 5,000 time units long for cases with no buffer or with 1 buffer space between stations and 10,000 time units for cases with 3 or more buffer spaces between stations.

- Each simulation run consists of 20 batches.

In our simulation work, we estimate the production rate r(AS) and r(SS) for various stochastic models of the operations and for various buffer allocation schemes. Since simulation is a sampling experiment, it is important to interpret our results within the framework of statistics. Therefore for each of our simulations, we must specify a confidence interval for the estimate of the mean output rate, r of the form:

$$E(X) - H \leq r \leq E(X) + H$$

E(X) is the observed sample mean and H is the interval half width, which is required to provide a confidence level of 1-a, say. That is, with probability 1-a, the true value of r falls within the range E(X) -H to E(X)+H. The half width depends on the standard deviation of the mean sample output rate, which in turn depends on the sample size. [18]

We use a Sequential Batch Means procedure to determine experimental values of the output rate. After the initialization period, there is a data collection period of length M, which is divided into K(=20) runs (batches) of length L(=5,000 or 10,000) (M=KL). Xj, the value of the production rate of batch j (Xj = # of arrivals during the jth batch / L) is used as an experimental value.[16] We choose this procedure over the Replication-Deletion Approach, which consist of replicating a run K times. Although this latter method insures the independence of the runs, it is quite wasteful in terms of computer time since it requires discarding observations at the beginning of each run.

Once we have obtained the K experimental values of X, we calculate the sample average

$$\overline{X} = \sum_{j=1}^{K} x_j \,/\, K$$

and the sample variance:

$$S^2(X) = \sum_{j=1}^{K} (X_j - \overline{X})^2 \,/\, (K-1)$$

If the Xj's are independent, the sample variance of $\overline{X}$ is:

$$S^2(\overline{X}) = S^2(X) \,/\, K$$

And consequently a (1-α) confidence interval for the output rate is:

$$\overline{X} - S(\overline{X})\, t_{K-1,\,\alpha/2} \le r \le \overline{X} + S(\overline{X})\, t_{K-1,\,\alpha/2}$$

, where $t_{K-1,\,\alpha/2}$ is the upper α /2 point of the t distribution with N-1 degrees of freedom.

The above confidence interval assumes normally distributed and independent X_j 's. The assumption of normality, in this case, is reasonably based on the central limit theorem, since the X_j 's are averages (i.e. average number of workpieces produced in L units of time). Anyway, for a reasonably large sample sizes (like K=20), the violation of normality should not significantly affect the results. We must verify though, that the X_j's are in deed independent. We do so for every simulation by computing the test statistic Cb, which for large values of K is an estimate of the correlation between adjacent batches.

$$C_b = 1 - \sum_{j=1}^{K} (X_j - X_{j+1})^2 / \left(2 * \sum_{j=1}^{K} (X_j - \overline{X})^2\right)$$

If the X_j's are in deed independent and normally distributed, then Cb is approximately normal with a mean of zero and a variance equal to $(K-2)/(K^2-1)$ for K as small as 8.

Thus to test the independence of the batches, we conduct the two-sided hypothesis test:

$$H_0\text{: } Cb=0 \quad \text{versus} \quad H_1\text{: } Cb\neq 0$$

We reject H_0 (i.e. independence) in favor of H_1 (i.e. correlated batches), if the absolute value of z is greater than $z_{(a/2)}$, where $z_{(a/2)}$ is the upper a/2 point on the standard normal and where:

$$z = Cb/\sqrt{[(K-2)/K^2-1)]}$$

3.2) Simulation of Models assuming Stations with Independent and Identically Distributed Processing Times

3.2.1) Introduction:

In this section we present the first of two simulation studies. This first study is based on a model assuming stations with identical quantitative characteristics, in particular independent and identically distributed processing times. Its goal was to evaluate the difference in the output rates of arborescent and sequential production systems and study how that difference varies as a function of the coefficient of variation of the distribution, the type of distribution, the buffer size and the number of stations.

3.2.2) Scenarios considered

To study how the difference between AS and SS varied as a function of n, we ran *simulations of 3, 7, 15 and 31 station AS and SS for all the scenarios considered.*

To study how the AS and SS vary in their ability to attenuate variability, as well as to study how this relative ability varies as a function of the buffer size, we considered the cases of:

- stations with *exponentially* distributed processing time (i.e. COV =1) and *buffer* sizes of *0, 1, 3, 5 and 10.*
- stations with *uniformly* distributed processing time with COV =0.5 and *buffer* sizes of *0, 1, 3, 5 and 10.*
- stations with *uniformly* distributed processing time with COV =0.3 and *buffer* sizes of *0, 1, 3, 5 and 10.*

As mentioned earlier, the expected value of all the distributions considered is 10 time units.

Finally, to study whether any difference in the output rate of AS and SS is dependent on the type of distribution, we considered the cases of systems with *no buffer* and with:

- stations with *lognormally* distributed processing time with COV of 1, 0.5 and 0.3
- stations with *Bernoulli* processing time distributions with COV of 1, 0.5 and 0.3

We compared these results to the cases of systems with no buffer involving uniform and exponential distributions.

3.2.3) Expected results

Before conducting simulations, we presumed that the simulations would prove the superiority of arborescent characterization The stations in the arborescent system seemed to be more decoupled from one another than stations in the sequential systems. The AS - because of its shorter production branches, n - $\ln_2(n)$ shorter than the sequential systems- appeared to have a higher capability to attenuate variability in the system. In particular, fewer stations in the AS seemed to be directly affected by a disruption in the system.

If for instance, the station O1 in the 8-part AS and SS shown Figure 3.10 is experiencing a temporary slow down, the effect of the slow-down will propagate more rapidly in the SS. All the stations in SS soon become starved whereas only O1, O3 and O7, the stations in O1's branch are ever starved in the AS. The effect of the slow-down on the other operations/stations in the AS is delayed until the moment their input slots are full and they become blocked.

Despite Ammar and Gershwin's finding that three-station AS and SS should have closely related behavior, the qualitative analysis of the three-station system, shown in Figure 3.10, seemed to confirm a superiority of the arborescent system:

- the two systems behave identically when O3 is lagging behind O1 and O2,
- neither system is clearly advantageous when O1 is lagging behind O2 and O3 by one piece (When both O2 and O3 have completed their pieces, in the arborescent systems O2 proceeds with the next piece, whereas in the sequential systems O3 proceeds with its next piece),
- arborescent systems are clearly advantageous when O2 is lagging behind O1 and O3 by one piece. Because, as soon as O1 and O3 have completed their pieces, in arborescent systems O1 can proceed with the next piece, whereas it is blocked until O2 finishes processing in the sequential systems.

We expected this arborescent advantage to grow proportionally with n, i.e. proportionally with the degree of arborescence of the AS.

3.2.4) Presentation of the results

The simulations indicated otherwise:

• As summarized in Figures 3.5a and 3.5b, there is *no* statistically *significant difference* between the output rate of *arborescent* and *sequential* systems

• the AS seems slightly advantageous when small values of n (n = 3 and n =7), but as n increases, the SS actually has a slight edge

• *any relative difference* is *attenuated by buffer spaces*. We plotted the ratio of r(AS) and r(SS) in Figures 3.6, 3.7 and 3.8 for various buffer space allocation schemes. And we observed that the ratio r(AS)/ r(SS) tends to 1 as the number of buffer spaces between stations grows.

• *any relative difference* is *inversely proportional* to the *coefficient of variation* (COV) of the stations' processing time distribution. We can see in Figures 3.6, 3.7 and 3.8 that the range of values of the r(AS)/r(SS) ratio decreased as the COV diminishes from 1 to 0.5 and finally to 0.3

• while the output rate of both AS and SS can vary considerably for different types of processing time distributions with the same COV- for instance, r(AS) and r(SS) are considerably higher for Bernoulli distribution than for lognormal distributions (as illustrated in the table in Figure 3.10)- the *difference does not vary* considerably as a *function* of the *type of distribution* .

Figure 3.5: Comparison of Arborescent and Sequential Systems with no buffer

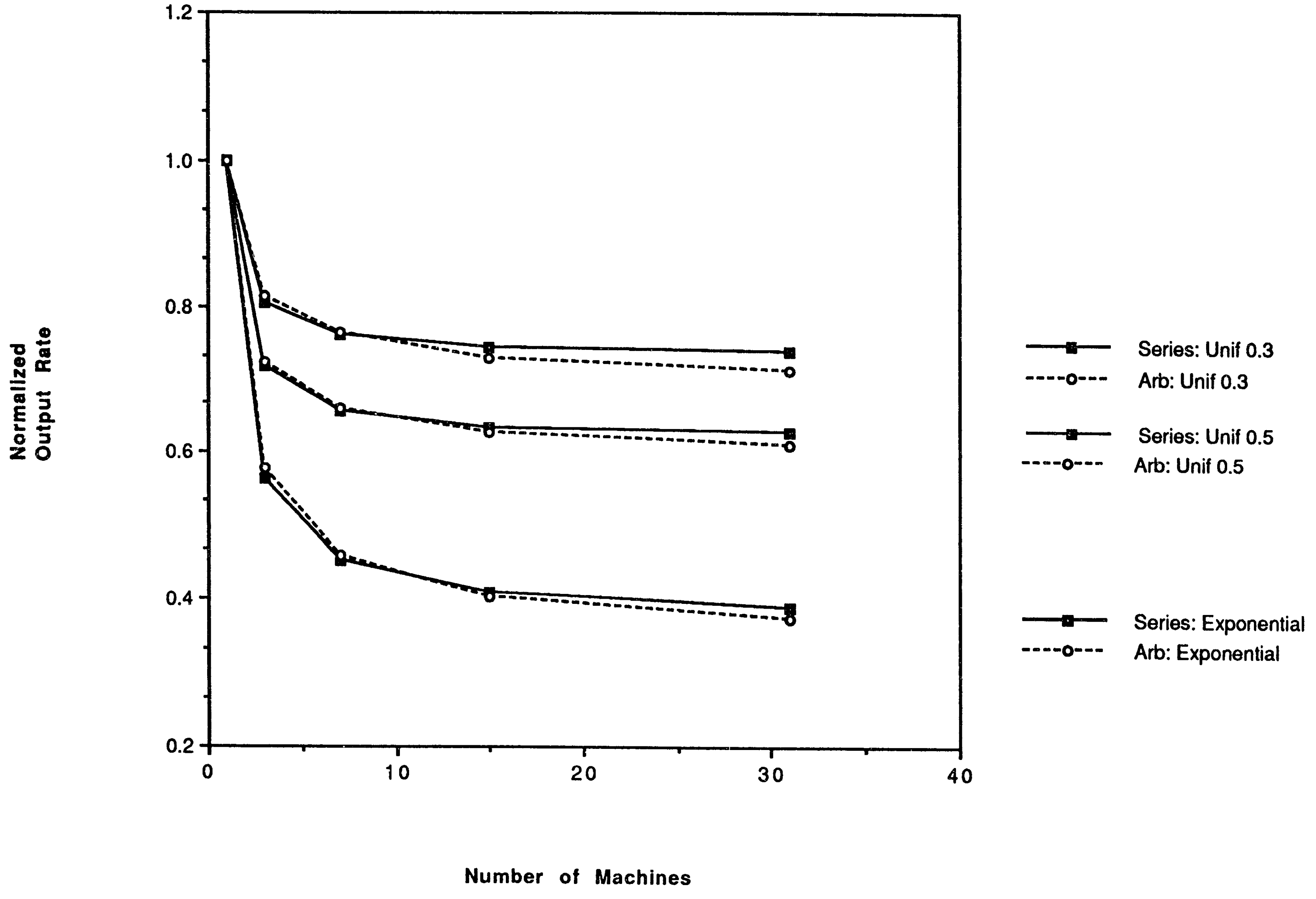
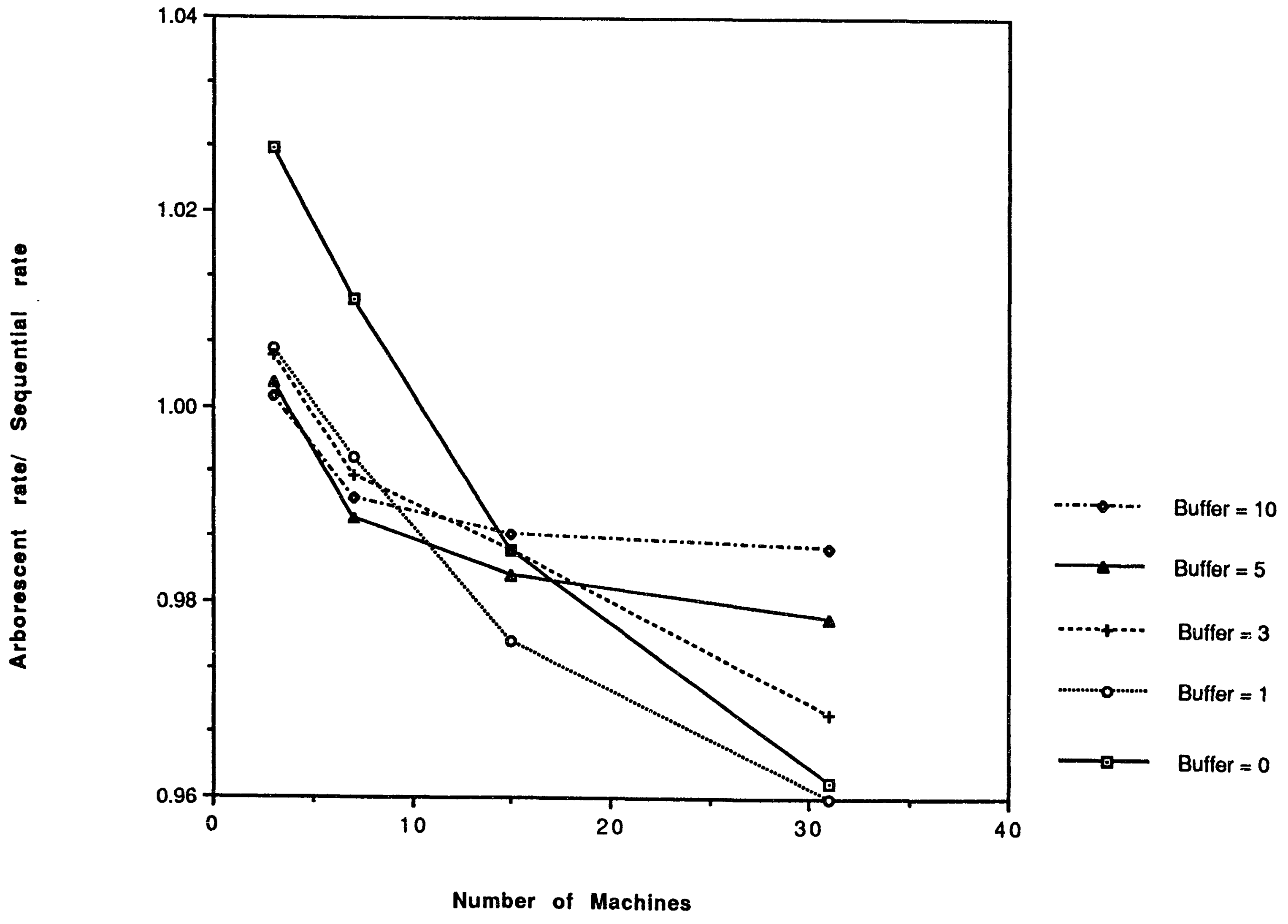


Figure 3.6: Comparison of Arborescent and Sequential Systems with stations with EXPONENTIALLY distributed processing times for different buffer size



Cases w/ no buffer |

	SERIES	ARBORESCENT
N = 3	r = 0.0563	r = 0.0578
N = 7	r = 0.0453	r = 0.0458
N = 15	r = 0.0410	r = 0.0404
N = 31	r = 0.0390	r = 0.0375

Cases w/ buffer= 1 |

	SERIES	ARBORESCENT
N = 3	r = 0.0672	r = 0.0676
N = 7	r = 0.0577	r = 0.0574
N = 15	r = 0.0543	r = 0.0530
N = 31	r = 0.0526	r = 0.0505

Cases w/ buffer= 3 |

	SERIES	ARBORESCENT
N = 3	r = 0.0775	r = 0.0779
N = 7	r = 0.0707	r = 0.0702
N = 15	r = 0.0679	r = 0.0669
N = 31	r = 0.0669	r = 0.0648

Figure 3.6b: Output rate of arborescent and sequential systems with *identical* stations with *exponentially* distributed processing times

Cases w/ buffer= 5

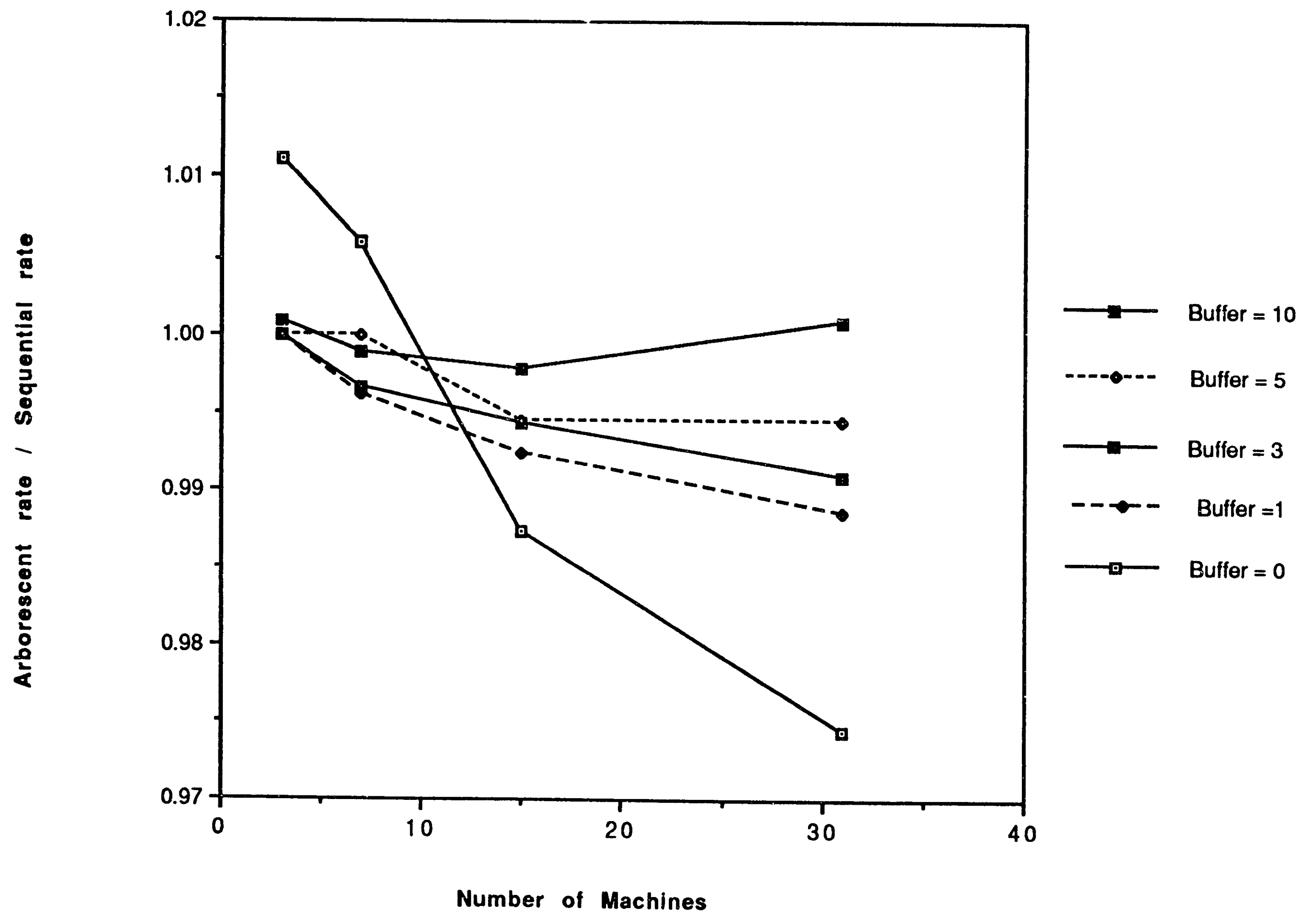
	SERIES	ARBORESCENT
N = 3	r = 0.0830	r = 0.0832
N = 7	r = 0.0778	r = 0.0769
N = 15	r = 0.0753	r = 0.0740
N = 31	r = 0.0741	r = 0.0725

Cases w/ buffer= 10

	SERIES	ARBORESCENT
N = 3	r = 0.0894	r = 0.0895
N = 7	r = 0.0860	r = 0.0852
N = 15	r = 0.0841	r = 0.0830
N = 31	r = 0.0834	r = 0.0822

Figure 3.6b(Cont'd): Output rate of arborescent and sequential systems with *identical* stations with *exponentially* distributed processing times

Figure 3.7: Comparison of Arborescent and Sequential Systems with stations with UNIFORM processing time distributions with COV = 0.5 for different buffer sizes



Cases w/ no buffer |

	Uniform w/ COV = .5		*Uniform w/ COV = .3*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0716	r = 0.0726	r = 0.0806	r = 0.0814
$N = 7$	r = 0.0655	r = 0.0659	r = 0.0760	r = 0.0763
$N = 15$	r = 0.0634	r = 0.0626	r = 0.0743	r = 0.0730
$N = 31$	r = 0.0626	r = 0.0610	r = 0.0737	r = 0.0711

Cases w/ buffer= 1 |

	Uniform w/ COV = .5		*Uniform w/ COV = .3*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0842	r = 0.0842	r = 0.0922	r = 0.0922
$N = 7$	r = 0.0806	r = 0.0803	r = 0.0902	r = 0.0898
$N = 15$	r = 0.0789	r = 0.0783	r = 0.0889	r = 0.0885
$N = 31$	r = 0.0783	r = 0.0774	r = 0.0885	r = 0.0879

Cases w/ buffer= 3 |

	Uniform w/ COV = .5		*Uniform w/ COV = .3*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0916	r = 0.0916	r = 0.0964	r = 0.0965
$N = 7$	r = 0.0895	r = 0.0892	r = 0.0954	r = 0.0953
$N = 15$	r = 0.0882	r = 0.0877	r = 0.0948	r = 0.0944
$N = 31$	r = 0.0878	r = 0.0870	r = 0.0946	r = 0.0942

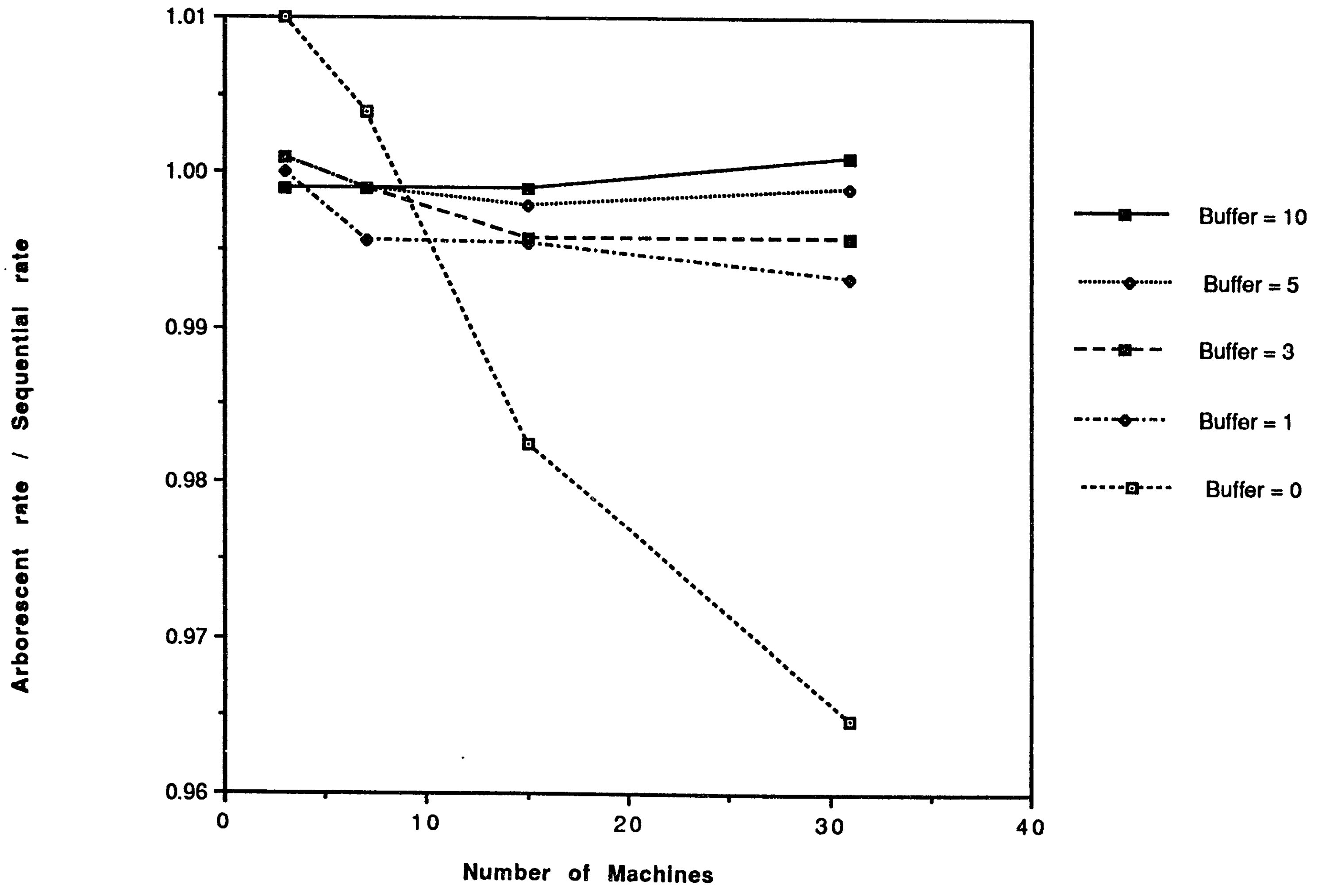
Figure 3.7b: Output rate of arborescent and sequential systems with *identical stations* with *uniformly* distributed processing times

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Cases w/ buffer= 5 |
- - - - - - - - - - - - - - - -
              Uniform w/ COV = .5         Uniform w/ COV = .3
              SERIES      ARBORESCENT     SERIES      ARBORESCENT
N = 3         | r = 0.0942 | r = 0.0942 |  | r = 0.0977 | r = 0.0978 |
N = 7         | r = 0.0926 | r = 0.0926 |  | r = 0.0970 | r = 0.0969 |
N = 15        | r = 0.0918 | r = 0.0913 |  | r = 0.0965 | r = 0.0963 |
N = 31        | r = 0.0915 | r = 0.0910 |  | r = 0.0964 | r = 0.0963 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Cases w/ buffer= 10 |
- - - - - - - - - - - - - - - - -
              Uniform w/ COV = .5         Uniform w/ COV = .3
              SERIES      ARBORESCENT     SERIES      ARBORESCENT
N = 3         | r = 0.0968 | r = 0.0969 |  | r = 0.0988 | r = 0.0987 |
N = 7         | r = 0.0957 | r = 0.0956 |  | r = 0.0983 | r = 0.0982 |
N = 15        | r = 0.0951 | r = 0.0949 |  | r = 0.0979 | r = 0.0978 |
N = 31        | r = 0.0948 | r = 0.0949 |  | r = 0.0977 | r = 0.0980 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 3.7b (Cont'd): Output rate of arborescent and sequential systems with *identical stations* with *uniformly* distributed processing times

Figure 3.8: Comparison of Arborescent and sequential systems with stations with UNIFORM processing time distribution with COV = 0.3 for different buffer sizes



Cases w/ Coefficient of Variance = 1

	Lognormal w/ COV = 1		*Bernoulli w/ COV = 1*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0581	r = 0.0592	r = 0.0582	r = 0.0589
$N = 7$	r = 0.0459	r = 0.0465	r = 0.0518	r = 0.0519
$N = 15$	r = 0.0407	r = 0.0403	r = 0.0505	r = 0.0499
$N = 31$	r = 0.0379	r = 0.0360	r = 0.0501	r = 0.0491

Cases w/ Coefficient of Variance = 0.5

	Lognormal w/ COV = .5		*Bernoulli w/ COV = .5*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0713	r = 0.0724	r = 0.0738	r = 0.0743
$N = 7$	r = 0.0626	r = 0.0633	r = 0.0683	r = 0.0684
$N = 15$	r = 0.0588	r = 0.0584	r = 0.0671	r = 0.0660
$N = 31$	r = 0.0571	r = 0.0550	r = 0.0668	r = 0.0645

Cases w/ Coefficient of Variance = 0.3

	Lognormal w/ COV = .3		*Bernoulli w/ COV = .3*	
	SERIES	ARBORESCENT	SERIES	ARBORESCENT
$N = 3$	r = 0.0803	r = 0.0811	r = 0.0824	r = 0.0828
$N = 7$	r = 0.0740	r = 0.0746	r = 0.0782	r = 0.0783
$N = 15$	r = 0.0714	r = 0.0706	r = 0.0772	r = 0.0756
$N = 31$	r = 0.0702	r = 0.0680	r = 0.0770	r = 0.0738

Figure 3.9: Output rate of arborescent and sequential systems with *identical stations* with *lognormal* or *Bernoulli* processing times

3.2.5) Interpretation of the results

While some of these results were predictable, other are quite surprising.

AS and SS behave identically in the case of stations with deterministic processing times. Consequently, any difference in the output rate of AS and SS will exist because of any difference in the ability of the two systems to attenuate processing time variability. It therefore seems logical that any difference increases as a function of the COV, the parameter which reflects the variability of the stations. Similarly, since the buffer size attenuates the system variability, it is also expected that any difference decreases with an increasing buffer size.

The relative equality of r(AS) and r(SS) for all n and for any type of distribution was considerably more surprising. We were able to formulate some explanations based on the three-station example, that partially explained the minimization of the arborescent advantage:

- First, O2 does not lag behind both O1 and O3 a large percentage of the time, maybe 20% of time since $Pr(t2<t1 \ \& \ t2<t3) = 0.334$, where t1, t2 and t3 are the random variables associated with the processing time distributions of a single piece at stations 1, 2 and 3.

- Second, the actual time saved, i.e. the time from the completion of processing on both O1 and O3 to the time of completion processing on O2 or (of the next unit) on O1, is small.

• Third, no additional time is saved when O2 is lagging O1 by more than one unit (in the case where there is no buffer spaces)

• Fourth, the time saved may not actually contribute to an increase in output rate, if O3 subsequently lags behind O1 by more that the saved amount

We remained puzzled by these results, though. We were particularly frustrated by the lack of nuance in these results. Consequently we decided to run a second batch of simulations to study the relative capability of the two types of systems to attenuate the processing time variability of isolated stations and develop intuition for the structural differences of AS and SS.

3.3) Simulation of models with bottleneck stations (i.e. stations with higher variability)

3.3.1) Introduction & Basic Definitions

a) Introduction

In this section, we relax the assumption of stations with independent, identically distributed processing times. We study bufferless, balanced systems with bottlenecks to get additional insight on the difference between arborescent and sequential systems. We are particularly interested in the relative capability of the two types of systems to attenuate the processing time variability of isolated stations. Unless otherwise mentioned,

• the bottlenecks will have exponentially distributed processing times with expected value of 10 time units,

• the systems' other stations will have deterministic processing times with the same expected value, namely 10 time units.

Note that, since all the stations in our system have the same expected values, *what we refer to as bottlenecks are not bottlenecks in the traditional sense of the word.* We use the term bottleneck to emphasize the fact that these stations with variable completion times are directly responsible for any loss in system capacity.

b) Basic definitions

In a sequential system, Station i is said *to lag* behind Station i-1, where Station i is directly downstream from station i-1 and it finishes its kth piece after Station i-1 has completed its k+1st piece. Similarly Station i-1 is said to lag behind Station i if it finishes its k+1 th piece after Station i has completed its kth piece.

In an arborescent system, Station i is said *to lag* behind Station j, where Station j is directly upstream from Station i and it finishes its kth piece after Station j has completed its k+1th piece. Similarly, Station j is said to lag behind Station i, where Station i is directly downstream from Station j and it finishes its k+1th piece after Station i has completed its kth piece.

3.3.2) First class of systems: systems with one bottleneck

a) Case # 1: Bottleneck at Station 1 (as shown in Figure 3.10)

i) Arborescent systems with a bottleneck at Station 1

Let us first consider the three-station arborescent system with a bottleneck at Station 1. Let t1 and t2 be the processing time of the kth piece on Station 1 (O1) and Station 2 (O2), respectively, and let t3 be the processing time of the k-1st piece on Station 3.

Assume that O1 and O3 start processing their kth and k-1st piece (respectively) at the same time. Note that both t2 ant t3 are deterministic, and t2 = t3 = 10.

If t1 < 10, i.e t1< t3, O1 is blocked for t3 - t1 units of times, since it cannot dispose of the kth processed piece.
If t1 > 10, i.e. t1 >t3, O3 is starved for t1 - t3 because it has received only one of the two necessary inputs.

Example: Assume that at t=0, O1 and O3 start processing their kth and k-1th piece respectively

If t1 = 7:

At t = 7, O1 completes its kth piece and becomes blocked, as it cannot dispose of the processed piece.
At t= 10, O3 completes its k-1th piece (O2 may have finished earlier as it may have starting processing before O3 if O3 was starved by O1 on the previous piece) and O1, O2, O3 start processing the next piece.

If t1 = 13:

At t = 10, O3 completes the k-1st piece and becomes starved. If O3 was also starved by O1 on the previous piece, O2 started processing before O3 and completed it before t = 10. In this case O2 is blocked until t = 10. If O3 was not starved by O1 on the previous piece, O2 started processing at the same time as O3 and also completes it at t= 10. In either case, O2 starts processing its next piece at t = 10.

At t =13, O1 finishes the kth piece. O1 and O3 start working on the next piece.

O2 never slows down the system, i.e. O3 is never starved because of O2. Consequently, this three-station arborescent system has the same output rate as a two-station (O1-O3) balanced, sequential system, where the first station is a bottleneck and the second station has deterministic processing times. In fact as a result of the reversibility of sequential systems, we can conclude that this three-station arborescent system has the same output rate as a balanced, two-station, sequential system, where one station is a bottleneck and the other has deterministic processing times.

Similarly, in larger arborescent systems with one bottleneck at Station 1, such as the seven station system in Figure 3.10, O3 is neither blocked nor slowed down by downstream stations or by stations in other branches. Consequently, *a balanced, n-station, arborescent system with n-1 stations with deterministic processing times and a bottleneck at Station 1 has the same output rate as a balanced,two-station, sequential system, where one station is a bottleneck and the other has a deterministic processing time.*

ii) Sequential systems with a bottleneck at Station 1

Similarly in these sequential systems, Station 2 is never slowed down by stations down-stream. O_{i+2} (where $i > 0$) always starts and completes the processing of its (k-i)th piece before or at the same time as O2 starts and completes the processing of its kth piece. Consequently, *balanced, n-station, arborescent and sequential systems with n-1 stations with deterministic processing times and a bottleneck at Station 1 have the same* *output rate as a balanced,two-station, sequential system, where one station is a bottleneck and the other has a deterministic processing time.*

b) Case # 2: Bottleneck at Station n (as shown in Figure 3.10)

i) Arborescent systems with a bottleneck at Station n

Let Station U1 and Station U2 be the two stations directly up-stream from Station n in an n-station arborescent system. Station U1 and Station U2 always complete the processing of the kth piece at the same time as the stations one stage up-stream complete their k+1th piece. Thus, all of Station n 's input can be represented by one station with deterministic processing time of 10. Consequently as a result of reversibility, a balanced n-station arborescent system with n-1 stations with deterministic processing times and a bottleneck at Station n has the same output rate as a balanced, two-station, sequential system, where one station is a bottleneck and the other has a deterministic processing time.

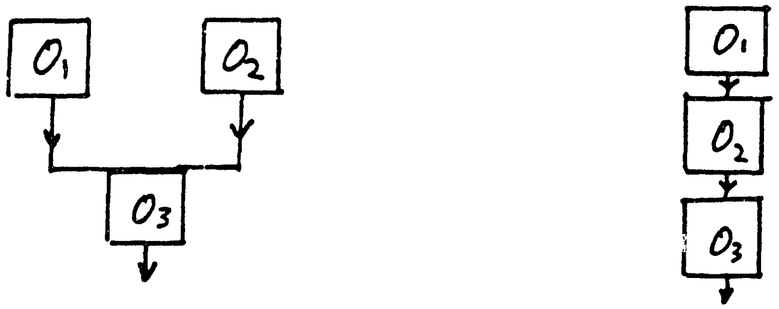
ii) Sequential systems with a bottleneck at Station n

We have already concluded that a balanced n-station arborescent system with n-1 stations with deterministic processing times and one bottleneck at Station 1, has the same output rate as a balanced, two-station, sequential system, where one station is a bottleneck and the other has a deterministic processing time. Because of the reversibility of sequential systems, we can also conclude that balanced n-station sequential and arborescent system with n-1 stations with deterministic processing times and a bottleneck at Station n has the same output rate as a balanced, two-station, sequential system, where one station is a bottleneck and the other has a deterministic processing time.

N=3

Arborescent System

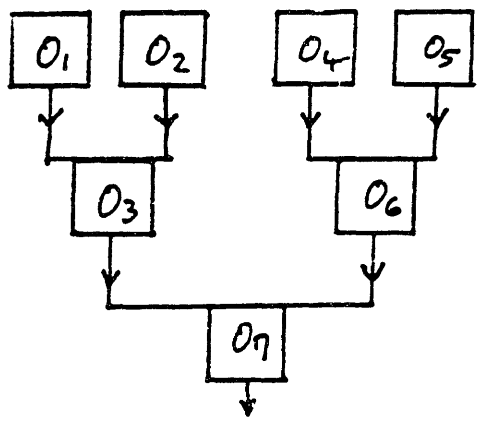
Sequential System



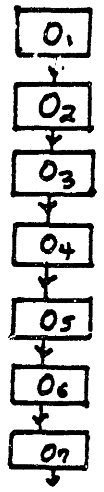
N=7

Arborescent System

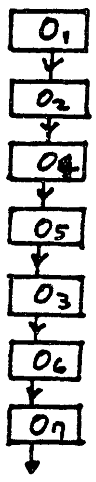
Sequential System



Alternative # 1



Alternative # 2 for cases 8 thru 11



Case # 1: O1
Case # 2: On , i.e. O7 if n = 7 and O3 if n = 3
Case # 3: On-1 , i.e. O6 if n = 6 and O2 if n = 2
Case # 4: O1 & O2
Case #5: O1 & O(n+1)/2 , i.e. O1 & O4 if n = 7 and O1 & O3 if n =3
Case #6: O1 & On , i.e. O1 & O7 if n = 7 and O1 & O3 if n =3
Case #7: O1 & On-1 , i.e. O1 & O6 if n = 7 and O1 & O2 if n =3
Case #8: All the first stage stations, i.e. O1 &O2&O4&O5 if n =7 and O1&O2 if n=3
Case #9: All the stations of the 2nd to last stage, i.e. O3&O6 if n =7 and O1&O2 if n=3
Case #10: All the non-first stage stations, i.e. O3 & O6 & O7 if n =7 and O3 if n=3
Case #11: All the stations of one branch, i.e. O1 & O3& O7 if n =7 and O1 &O3 if n=3

Figure 3.10a: Representation of 3 and 7 station arborescent and sequential systems and listing of the bottlenecks used in section 3 of Chapter II to compare the relative ability of AS and SS to attenuate the effect of isolated source of variability.

Figure 3.11: Simulation results for cases involving bottlenecks

		SERIES	ARBORESCENT
Cases w/ 1 bottleneck			
Case #1: btn at O_1	n = 3	r = 0.0730	r = 0.0730
	n = 7	r = 0.0730	r = 0.0730
	n =15	r = 0.0730	r = 0.0730
Case #2 : btn at O_n	n = 3	r = 0.0730	r = 0.0730
	n = 7	r = 0.0730	r = 0.0730
	n =15	r = 0.0737	r = 0.0736
Case #3:	n = 3	r = 0.0727	r = 0.0728
btn at O_{n-1}	n = 7	r = 0.0737	r = 0.0736
	n = 15	r = 0.0727	r = 0.0727
Cases w/ 2 bottlenecks			
Case #4:	n = 3	r = 0.0616	r = 0.0628
btns at O_1 & O_2	n = 7	r = 0.0616	r = 0.0628
	n =15	r = 0.0613	r = 0.0628
Case #5: btns at O_1	n = 3	r = 0.0616	r = 0.0628
& $O_{(n+1)/2}$	n = 7	r = 0.0631	r = 0.0651
	n =15	r = 0.0664	r =0.0659
Case #6:	n = 3	r = 0.0617	r = 0.0631
btns at O_1 & O_n	n = 7	r = 0.0651	r = 0.0613
	n =15	r = 0.0691	r = 0.0632
Case #7:	n = 3	r = 0.0613	r = 0.0628
btns at O1 & On-1	n = 7	r = 0.0650	r = 0.0642
	n = 15	r = 0.0685	r = 0.0646

Figure 3.11 (cont'd): Simulation results for cases involving bottlenecks

	SERIES	ARBORESCENT
Cases w/ all the bottlenecks in the same stage		
Case #8: Bottlenecks at all the first stage stations		
n = 3	r = 0.0613	r = 0.0628
n = 7 w/ alternative # 1	r = 0.0514	r = 0.0564
n = 7 w/ alternative # 2	r = 0.0506	
n =15 w/ alternative #1	r = 0.0476	r = 0.0520
n =15 w/ alternative #2	r = 0.0441	
Case #9: Bottlenecks at all the stations of the 2nd to last stage		
n = 3	r = 0.0617	r = 0.0628
n = 7 w/ alternative # 1	r = 0.0630	r = 0.0631
n = 7 w/ alternative # 2	r = 0.0608	
n =15 w/ alternative #1	r = 0.0660	r = 0.0628
n =15 w/ alternative #2	r = 0.0606	
Cases w/ at least two bottlenecks in each branch		
Case #10: Bottlenecks at all the non-first stage stations		
n = 3	r = 0.0733	r = 0.0733
n = 7 w/ alternative # 1	r = 0.0570	r = 0.0552
n = 7 w/ alternative # 2	r = 0.0552	
n = 15 w/ alternative #1	r = 0.0500	r = 0.0454
n =15 w/ alternative #2	r = 0.0448	
Case #11: Bottlenecks at all the stations of one branch		
n = 3	r = 0.0619	r = 0.0632
n = 7 w/ alternative # 1	r = 0.0584	r = 0.0549
n = 7 w/ alternative # 2	r = 0.0586	
n =15 w/alternative # 1	r = 0.0581	r = 0.0503
n =15 w/ alternative #2	r = 0.0577	

c) Case # 3: Bottleneck at Station i where $n/2<i<n$ (as shown in Figure 3.10)

Here i is between n/2 and n, rather than between 1 and n, because if $i < n/2$ in the arborescent case, this problem becomes identical to the one solved in section 1).

For reasons similar to those mentioned in Case # 1 and Case #2, we can conclude that balanced n-station arborescent and sequential systems with n-1 stations with deterministic processing time and a bottleneck at Station i (where $n/2 < i < n$) have the same output rate as balanced, three-station sequential systems, where the second station is a bottleneck and where the other two stations have deterministic processing times. In fact since the starvation of the downstream station always occurs simultaneously with the blockage of the upstream station, we can even say that balanced n-station arborescent and sequential systems with n-1 stations with deterministic processing time and a bottleneck at Station i (where $n/2 < i < n$) have the same output rate as balanced, two-station sequential systems where one station is a bottleneck and the other has a deterministic processing time.

d) Summary of the single bottleneck scenario

Any balanced, n-station arborescent or sequential system with n-1 stations with deterministic processing times and one bottleneck has the same output rate as a balanced, two-station sequential system where one station is a bottleneck and the other has a deterministic processing time, regardless of where the bottleneck is.

In fact *any balanced, n-station (where $n > 1$) system with n-1 stations with deterministic processing times and with one bottleneck has the same output rate as a balanced, two-station sequential system where one station is a bottleneck and the other has a* *deterministic processing time, regardless of the value of n , the location of the bottleneck and the structure of the system .*

The completion interarrival time for such a system is described by the maximum of 10 and X, where X is an exponential random variable with mean 10. The expected interarrival time can be calculated to be 13.6786. Thus the output rate should be reasonably close to 1/ 13.6786, i.e. 0.0731.

e) Simulation work for the single bottleneck scenario

The simulation work for systems with one bottleneck summarized in Figure 3.11 confirmed that balanced arborescent systems and balanced, sequential systems with one bottleneck have the same output rate, regardless of the value of n and the location of the bottleneck.

3.3.3) Second class of systems: systems with two bottlenecks

a) Case # 4: Bottlenecks at Station 1 and 2 (as shown in Figure 3.5)

i) Expected results

Balanced, n-station systems with bottlenecks at Station 1 and Station 2 are never slowed down by Station 4 through n, since these stations work in unison with Station 3.

Consequently, any balanced, n-station arborescent system with bottlenecks at Stations 1 and 2 has the same output rate as a balanced, 3-station arborescent system with bottlenecks at Stations 1 and 2. And, any balanced, n-station sequential system with bottlenecks at Stations 1 and 2 has the same output rate as a balanced, 3-station sequential system with bottlenecks at Stations 1 and 2.

Intuitively, the three-station arborescent systems should have a considerably larger output rate than the three-station sequential systems, because:

• the two systems behave identically when O3 is lagging behind O1 and O2,

• neither system has a clear advantage when O1 is lagging behind O2 and O3 by one piece (When both O2 and O3 have completed their pieces, O2 proceeds with the next piece in arborescent systems, whereas in sequential systems O3 proceeds with the next piece),

• arborescent systems have an advantage when O2 is lagging behind O1 and O3 by one piece. Because, as soon as O1 and O3 have completed their pieces, in the arborescent systems O1 can proceed with the next piece, whereas it is blocked until O2 finishes processing its piece in the sequential systems. This difference should be especially significant, since O2 is a bottleneck in this case.

ii) Simulation results

As shown in Figure 3.11, simulations confirmed that the output rate of systems with bottlenecks at Stations 1 and 2 was independent of the number of deterministic stations in the systems. Surprisingly though, the output rate of the arborescent systems proved to be only 2% greater than the output rate of the sequential systems. Several explanations are possible.

First, O2 does not lag behind both O1 and O3 a large percentage of the time, maybe 20% of time since Pr(t2<t1 & t2<t3) = 0.334.

Second, the actual time saved, i.e. the time from the completion of processing on both O1 and O3 to the time of completion processing on O2 or (of the next unit) on O1, is small.

Third, the time saved may not actually contribute to an increase in output rate, if O3 subsequently lags behind O1 by more that the saved amount.

b) Case # 5: Bottlenecks at Station 1 and (N+1)/2

(as shown in Figure 3.10)

i) Expected results

Intuitively, the difference between arborescent and sequential systems should increase in this case, because of the higher degree of independence of the bottlenecks in the arborescent systems. The effect of the occasional lag of O2 behind O1 and O3 should be minimized considerably in the arborescent systems.

In both types of systems, the output rate should increase with n because as n grows the bottlenecks are further isolated.

ii) Simulation results

As shown in Figure 3.11, simulations confirmed that in both types of systems the output rate of systems with bottlenecks at Station 1 and (N+1)/2 increased with n. The difference between the output rate of arborescent and sequential systems increased to 3% when n=7. However, for n = 15 the sequential systems turned to be just as productive as the arborescent systems. Several explanations are possible.

First, we may have undermined the isolation of the bottlenecks in the sequential systems. Small fluctuations in the production rate of Station 1 are attenuated by the deterministic stations between the bottlenecks. *Given that there is at least one deterministic station between the bottlenecks, adding other deterministic stations between the bottlenecks further decouples the bottlenecks. The added deterministic stations act as buffer spaces and contribute to an increase in output rate.* This effect is particularly significant in large sequential systems and thus limits the potential disadvantage of a sequential configuration.

Second, the sequential configuration is at a disadvantage when the bottleneck is lagging behind the preceding and succeeding stations. But this situation only occurs for the second bottleneck in the system.

c) Case # 6: Bottlenecks at Station 1 and Station n

i) Expected Results

When n = 3, the arborescent systems should be better than the sequential systems for the reasons mentioned in Case #4, i.e.:

• the two systems behave identically when O3 is lagging behind O1 and O2,

• neither system has a clear advantage when O1 is lagging behind O2 and O3 by one piece (When both O2 and O3 have completed their pieces, O2 proceeds with the next piece in arborescent systems, whereas in sequential systems O3 proceeds with the next piece),

• arborescent systems are clearly advantageous when O2 is lagging behind O1 and O3 by one piece. Because, as soon as O1 and O3 have completed their pieces, in the arborescent systems, O1 can proceed with the next piece whereas in the sequential systems O1 is blocked until O2 finishes its piece.

When n > 3, i.e. when there is at least one stage between Station 1 and Station n, the sequential systems are clearly advantageous.

For arborescent systems with more than three stations, Station 2 does not affect the overall output rate because once the first piece has been processed, it never lags behind Station 3. Similarly, stations, not in Station 3's branch, never lag behind Stations 3 and n and thus do not influence the production rate either. Consequently if n > 3, *balanced, n-station arborescent systems with bottlenecks at Stations 1 and n should have the same output rate as a balanced,* $ln_2(n+1)$ *sequential systems with bottlenecks at Stations 1 and* $ln_2(n+1)$. 7-station and 15-station arborescent systems should have the same output rate respectively, as 3-station and 4-station sequential systems with two bottlenecks. Consequently, the output rate of 7-station and 15-station arborescent systems should be less than the output rate of a 7-station sequential system, which in turn should be less than the output rate of a 15-station sequential system. For, as mentioned in the previous section, given that there is at least one deterministic station between the bottlenecks, adding other deterministic stations between the bottlenecks further decouples them and increases the output rate. Note that 3 station arborescent systems with bottlenecks at Stations 1 and n do not behave as a $\ln_2(3+1)$, i.e. 2 station sequential systems with bottlenecks at Stations 1 and 2, because station O2 in the 3-station AS occasionally slows down the system. (since there is no deterministic stations between the bottlenecks).

ii) Simulations results

As shown in Figure 3.11, simulations for systems with bottlenecks at Station 1 and n confirmed the findings of Case #4, i.e. that for 3-station systems the arborescent configuration is only slightly more advantageous than the sequential configuration.

As expected, when n>3, the output rate of these types of arborescent systems is considerably smaller that the output rate of the corresponding sequential systems. The output rate of 7-station and 15-station arborescent systems was indeed smaller than the output rate of 7-station sequential systems. In fact, the 7-station arborescent system was 6% less productive than its sequential counterpart. The difference between the two configurations increases with n, or more precisely with $n - \ln_2(n)$. The output rate of the 15-station arborescent with bottlenecks at Stations 1 and n was 9% lower than the output rate of the corresponding sequential system.

d) Case # 7: Bottlenecks at Station 1 and Station n-1

i) Expected results

For systems with more than three stations, the output rates should all be between the rates found in Cases 5 and 6.

The sequential systems (with n >3) should have slightly smaller rates than in Case #6, since the bottlenecks are now closer together. On the other hand, the output rate of arborescent systems should increase slightly since the bottlenecks are now more decoupled than in Case #6.

ii) Simulation results

As shown in Figure 3.11, simulations results were exactly as expected.

3.3.4) Third class of systems: systems where bottlenecks are all in the same stage

a) Case # 8: Bottlenecks at all first stage stations

(as shown in Figure 3.10)

i) Expected results

Let us call OU1 and OU2 the stations directly upstream from a station O.

In section 3.3.3 a), we mentioned than the arborescent configurations was particularly advantageous, when OU2 was lagging behind OU1 and O. Because, as soon as O and OU1 have completed their pieces, in the arborescent systems OU1 can proceed with the next piece where as in the sequential systems OU1 is blocked until OU2 finishes its piece.

This advantage should thus be amplified if all the first stage stations in the arborescent systems and the corresponding stations in the sequential stations are bottlenecks. Consequently, as n increases the output rate of arborescent systems should be considerably greater than the output rate of the corresponding sequential system.

There may be some questions as to what is in fact an equivalent sequential system. The two possible alternatives are shown in Figure 3.10. Both alternatives were studied, though the alternative # 1 system seems more fitting. For, if one has a choice between sequential and arborescent configurations then one also has the choice between alternative # 1 and alternative #2, and one would have to prefer alternative # 1, since its bottlenecks are better separated.

ii) Simulation results

As shown in Figure 3.11, simulations confirmed the superiority of the arborescent configuration when all the first stage station are bottlenecks.

The 15-station arborescent system, for instance, had a 10% or 17% larger output rate than the corresponding 15-station sequential system, depending on whether alternative #1 or alternative #2 was chosen as the sequential system.

<u>b) Case # 9: Bottlenecks at all stations of the second to last stage</u> (as shown in Figure 3.10)

i) Expected results

Let us call the two stations of the second to last stage, U1 and U2.

An n-station arborescent system with two bottlenecks in the second to last stage should behave exactly like the n-station arborescent system with bottlenecks at Stations 1 and 2, studied in Case #4. The sequential system should be different, though, from its Case #4 analog.

For small values of n, the arborescent configuration has the advantage that if U2 lags behind U1 and Station n, U1 can proceed with the next piece as soon U1 and Station n have completed their pieces, whereas this is not possible in sequential configurations. For n = 3, the arborescent system should have the 2% advantage it had in Case #4. However, this potential advantage should be diminished for large systems because as n increases, the bottlenecks in the (alternative # 1) sequential system become more and more decoupled by the deterministic stations between the bottlenecks, which act as buffer spaces.

ii) Simulation results

Simulation results confirmed the presence of two conflicting factors, which depending on the values of n gave an advantage to one configuration or another. As expected, for relatively small values of n, the arborescent system was 2% more productive. However, with increasing values of n, the advantage of partially decoupled bottlenecks in the sequential configurations became significant. For n equal to 15, the sequential system was 5% more productive.

3.3.5) Fourth class of systems: systems where there are least two bottlenecks in each branch

a) Case # 10: Bottlenecks at all non-first stage stations (as shown in Figure 3.10)

i) Expected results

If alternative #2 is used to represent the sequential systems, the two systems should have practically identical output rate. Because, arborescent and sequential systems with identical stations have the same output rate and if alternative # 2 is used, the (n-1)/2 bottlenecks can be thought of as a subsystem with all identical stations.

However, if one uses the more logical choice, i.e. alternative # 1, the sequential system should have a higher output rate, because in this type of system the bottlenecks are somewhat decoupled from one another.

ii) Simulation results (see Figure 3.10)

As expected, when alternative #2 was used to represent the sequential system, the two types of systems had identical output rates.

When the more logical, alternative # 1 was used to represent the sequential system, though, the sequential configuration was considerably more advantageous. The sequential system was 10 % more productive for n equal to 15 for instance.

<u>b) Case # 11: Bottlenecks at all the stations of one branch i.e. at Station 1, 3, 7, 15, etc.,</u>(as shown in Figure 3.10 and summarized in Figure 3.11b)

i) Expected results

This should be the case that gives the sequential configuration the greatest structural advantage. It is because, while the bottlenecks are partially decoupled in the sequential systems, they are directly connected and consequently pace, block and starve each other in the arborescent systems.

The alternative sequential systems should yield identical results here, because they are dual of one another in this case. In other words, Station 1 in alternative # 1 is equivalent to Station n in the alternative # 2 system, or more generally, Station i in one system alternative # 1 is equivalent to station n-i+1 in the other.

ii) Simulation results

As expected, the simulation illustrated the superiority of the sequential configuration for this scenario. This advantage increased considerably with n. The sequential system was respectively 6 and 15 % more productive for n=7 and n=15.

As expected the two alternative sequential systems yielded statistically equivalent results.

3.3.6) Conclusions

The simulation work involving bottleneck stations (i.e. stations with variability) helped us understand the nuances of the structural difference of the SS and the AS.

After this second set of simulations, it became clear that while no structure is always advantageous, depending on the location and the number of bottlenecks, one structure or another may be preferable.

The arborescent structure has an advantage whenever a station OU2 is lagging behind the station OU1 and the station O, where OU1 and OU2 are stations directly upstream from a station O. Because, as soon as O and OU1 have completed their current pieces, in the arborescent systems, OU1 can proceed with the next piece whereas in the sequential systems OU1 is blocked until OU2 finishes its piece.

This*"lateral"* advantage is most significant when all the bottlenecks are in the first stage. In that case, the arborescent system is at least 10 % more productive for systems with 15 or more stations. This "lateral" advantage does not propagate downstream when all the bottlenecks are not in the first stage, however.

This is principally due to the *"longitudinal"* advantage of sequential systems. Given that there is at least one deterministic station between bottlenecks, additional deterministic stations between the bottlenecks act as buffer spaces and contribute to an increase in output rate. This effect is particularly significant in large sequential systems and thus limits the potential disadvantage of a sequential configuration. In fact when there are bottlenecks at every stage of a given branch in the AS, the AS can be as much as 15% less productive than the corresponding SS for systems of 15 or less stations.

A generalization of this "longitudinal" advantage may exist for the models with i.i.d. stations. Whenever two stations in the same production branch of the AS are experiencing temporary slow-downs, the effect of the slow-downs may be better attenuated in the SS, where the stations in the SS are further apart.

Consequently, having production branches n station long instead of $\ln_2(n)$ station long may not necessarily be detrimental, especially since we know from Muth (1973) and Conway (1988) that most of the loss of capability in balanced, sequential systems of stations with stochastic processing time occurs in the first few stations.

3.4) Overall Conclusion of the Output Rate Comparison of Arborescent and Sequential Systems

In an effort to quantify the impact of using subassemblies in a single-product environment, we compared, in this chapter, the output rate of non-synchronous arborescent and sequential systems for two different models of systems' station.

The first model assumed stations with identical quantitative characteristics, in particular independent and identically distributed processing times. The purpose of this first batch of simulations was to evaluate the difference in the output rates of arborescent and sequential production systems and study how that difference varies as a function of the coefficient of variation of the distribution, the type of distribution and the buffer size.

The second model assumed bufferless, balanced systems with two types of stations: stations with deterministic processing times and stations with exponentially distributed processing times. The purpose of this second set of simulations was to study the relative capability of the two types of systems to attenuate the processing time variability of isolated stations, as well as to develop intuition for the structural differences of the two types of systems.

The basic conclusion of the simulation of these two models is that AS and SS do not differ greatly in their ability to attenuate processing time variability. While one structure or another may be preferable depending on the location and the number of sources of variabilities, no structure is clearly always advantageous.

Certainly, any decision to use subassemblies in an assembly or production system should not be based on the relative output rates of non-synchronous arborescent and sequential systems. That is not to say, however, that arborescent and sequential are quantitatively equivalent. Further study in such areas as testing should be pursued.

It is important at this time to put our results into perspective. Our work here assumed non-synchronous systems. Consequently, the behavior of two types systems was not expected to be so closely related.

For synchronous systems, identical results for the two types of configurations would not have been surprising since the simultaneous transfer of all the parts in those systems is paced by the slowest station. The expected interarrival time for both AS and SS would simply have been E(max(T1, T2, ...,Tn)), where Ti is the processing time at the ith station.

The comparison of the output rate of arborescent and sequential systems with stochastic completion would be complete with a study of the output rate of infinite buffer synchronous systems, where parts are transferred in batches between stations. In these types of systems, while no station would ever be blocked, the phenomenon of starving would remain and in fact would be accentuated. Intuitively, this transfer mechanism would seem likely to accentuate the "lateral advantage" of arborescent systems, while limiting the "longitudinal advantage" of sequential systems.

References made in Chapter III:

[1] Ammar, M.H. and S.B. Gershwin. 1987. "Equivalence Relations in Queueing Models of Assembly/Disassembly Networks," Georgia Institute of Technology School of Information and Computer Science Technical Report GIT-ICS-87/45

[2] Bratley P., B.L. Fox, L.E. Schrage. A Guide to Simulation. Springer-Verlag, New York, 1983.

[3] Buzacott, J. 1967. "Automatic Transfer Lines with Buffer Stocks." Int. J. Prod. Res. 6, 173-1988.

[4] CACI Products Company. "SIMGRAPHICS User's Guide and Casebook," La Jolla, Ca 1988

[5] Conway, R., W. Maxwell, J.O. Mclain and L. J. Thomas. 1988."The role of Work-in-Process Inventory in Serial Production Lines," Opns. Res, Vol. 36, No. 2, March-April.

[6] Crane, M.A. 1974. "Multi-Server Assembly Queues," Journal of Applied Probability, Vol.11 Number 3, pp 354-367.

[7] Freeman, D.R., 1968, "A General Line Balancing Model," Proceeding XIX Conference A.I.I.E., p. 230

[8] Fishman, G.S., 1978. Principle of Discrete Event Simulation, John Wiley and Sons, Inc., New York.

[9] Gershwin, S.B. 1988. "Assembly/Disassembly Systems: An Efficient Decomposition Algorithm for Tree-Structured Networks", Laboratory for Manufacturing and Productivity, M.I.T., Cambridge, MA.

[10] Groover, Mikell P. 1983. Automation, Production Systems, and Computer Integrated Manufacturing. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.
Harrison, J.M. "Assembly-like Queues," Journal of Applied Probability, Vol. 10 Number 1, pp 354-367.

[11] Hillier, F.S., and R.W. Boling. 1967. "Finite Queues in Series Exponential or Erlang Service Times - A numerical Approach." Opns. Res. 15, 286-303.

[12] Hunt, G.C. 1956. "Sequential Arrays of Waiting Lines," Opns. Res. 4, pp 674-683.

[13] Knott, A.D. 1970. "The inefficiency of a series of work stations - a simple formula," Int. J. Prod. Res., 8, 109.

[14] Latouche, G. 1981. "Queues with Paired Customers," Journal of Applied Probability, Vol.18 Number 3.

[15] Larsen, R.J, M.L. Marx. An Introduction to Mathematical Statistics and its Applications. Prentice Hall, Englewood Cliffs, New Jersey, 1986.

[16] Law A.M., W.D. Kelton. Simulation Modeling and Analysis. McGraw-Hill,1982.

[17] Muth, E.J. 1973. "The Production Rate of a Series of Workstations with Variable Service Times." Int. J. Prod., Vol. 11, No. 2, pp 155-169.

[18] Muth, E.J. 1979. "The Reversibility Property of Production Lines," Management Science, Vol. 25, No.2, February.

[19] Pedgen, C.D., Introduction to SIMAN, Systems Modeling Corporation, State College, PA, August 1984.

[20] Pinto, P.A. and D.G annenbring and B.M. Khumawala. 1981. "Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations," Int. J. Prod. Res., 1981, Vol.19, No. 5, pp 565-576.

[21] Russell, Edward C, Building Simulation Models with SIMSCRIPT II.5.CACI, Inc. 1983.

[22] Smith, J.M. and S. Daskalaki. 1988. "Buffer Allocation in Automated Assembly Lines," Operations Research, Vol.36, No. 2, March-April, pp 343-358.

[23] Svestka, J.A. and K.P.K. Nair. 1972. "Parallel Operation of Sequential Service Lines with Interline Transfer," AIIE Transactions, Vol.14, No.1, March., pp 29-34.

[24] Yamazaki, G. and H. Sakasegawa. 1975. "Properties of Duality in Tandem Queueing Systems," Ann. Inst. Stat. Math. 27, pp 201-212.

CHAPTER IV

GENERATION OF THE MOST ARBORESCENT ASSEMBLY SEQUENCES

4.1) INTRODUCTION

In the previous chapter, we compared the output rate of asynchronous arborescent and sequential production systems with reliable stations with stochastic processing times for various buffer allocation schemes. This comparison of AS and SS assumed an idealized product, which could be assembled via both types of configurations. In reality, most products neither have a number of parts n+1 such that $n+1= 2^k$ nor can they be built using both purely AS and SS. For case studies of products from industry, it would be most useful to know the most arborescent sequence possible for the assembly of a given product.

In this chapter we outline a method to determine the most arborescent and the least arborescent assembly sequence possible for a product using the Liaison Sequence Analysis (LSA) software.

4.1.1) Algorithmic methods for assembly sequence generation

The Liaison Sequence Analysis is an algorithmic method developed by De Fazio and Whitney for generating all the assembly sequences possible for a product. LSA is a simplified version of Alain Bourjault's "Elaboration Automatique des Séquences Opératoires", which introduced the notion of liaison and liaison diagram. A liaison diagram is a network wherein nodes represent parts and a line or arc between two nodes represents any user-defined relation between the two parts called a "liaison".

User-defined relations of "liaison" in a general sense follow the literal definition, i.e. "a close bond or connection" [1]. Examples of liaisons include physical part-to-part contact or interference fit between parts, or pass through without touching, such as a bolt through a hole.[3] A sample 6-part 5-liaison product and its liaison diagram are shown in Figure 4.1.

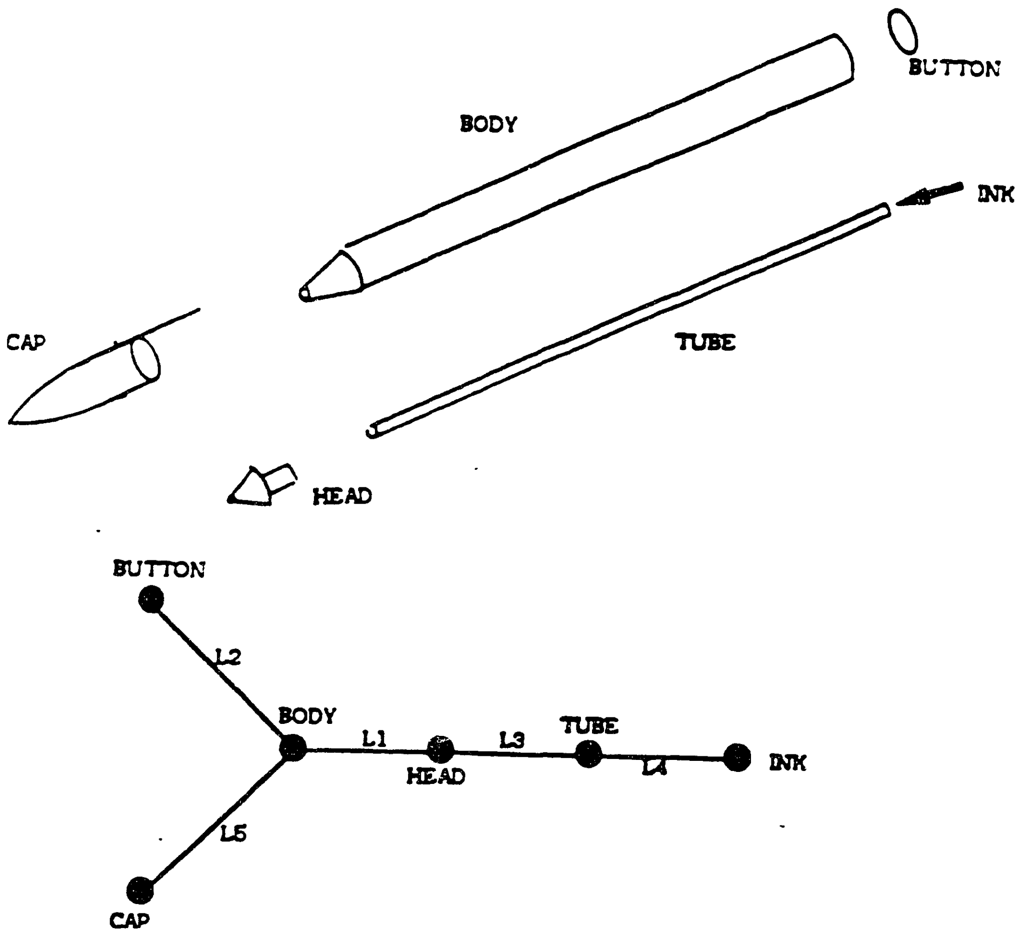


Figure 4.1: Example of a six-part five-liaison product: A ballpoint pen

Source: Charles J. Klein, 1987

Both the Bourjault and the De Fazio-Whitney methods view component assembly as the sequential completion of the liaisons between parts and determine all the possible *liaison assembly sequences* through the use of rules or precedence constraints. The two methods differ in the questions used to generate these precedence relations.

The questions used by Bourjault are of two general forms:

Question 1: Is it true that liaison Li can be established if Liaisons (Lj... Lk) have been established?

Question 2: Is it true that Liaison Li can be established if Liaisons (Lj... Lk) have not been established?

The group (Li...Lk), called the body of the question, consists of one or more liaisons and evolves overtime with the "yes" or "no" answers of the user.

The two questions asked by De Fazio and Whitney for each liaison Li are the following:

Question 1: What liaisons must be established prior to establishing Li?

Question 2: What liaisons must not be established until after establishing Li?

The answers directly correspond to the set of precedence relations for the assembly system and are either in the form of "none" or a logical combination. A sample answer could be: "3 &4 -> 6", which indicates that the establishment of both liaisons 3 and (&) 4 must precede (->) the establishment of liaison 6.

As mentioned previously the LSA software, currently implemented at Draper Labs, is based on the De Fazio-Whitney method. The main asset of this method is that it reduces the number of questions to be answered from a minimum of 2 (L^2 + L) (to a maximum of 2^L) required by Bourjault's method to 2 L, where L is the number of liaisons. This reduction makes the generation of precedence relations practical for a product with a large liaison count.[2] It is worth pointing out though, that Bourjault and engineers at Draper Labs are currently working on a simplification of the "Elaboration

Automatique des Séquences Opératoires", which will decrease the difference in the number of questions answered by the two methods.

4.1.2) Graphical representation of assembly sequences in the LSA software

In the LSA software, the assembly sequences are represented graphically by liaison sequence diagrams. These diagrams/graphs are networks of states, represented by boxes, linked together by state transitions, represented by lines as shown in Figure 4.2. Each box is divided into cells that represent liaisons. A darkened cell indicates the establishment of liaison. Every state is unique and represents an assembly state. Every state transition represents the path from one state to another. There are usually multiple state transitions to and from a state. A possible assembly sequence is a path from the unassembled state (0th rank) to the final assembled state (last, i.e. nth rank).[2]

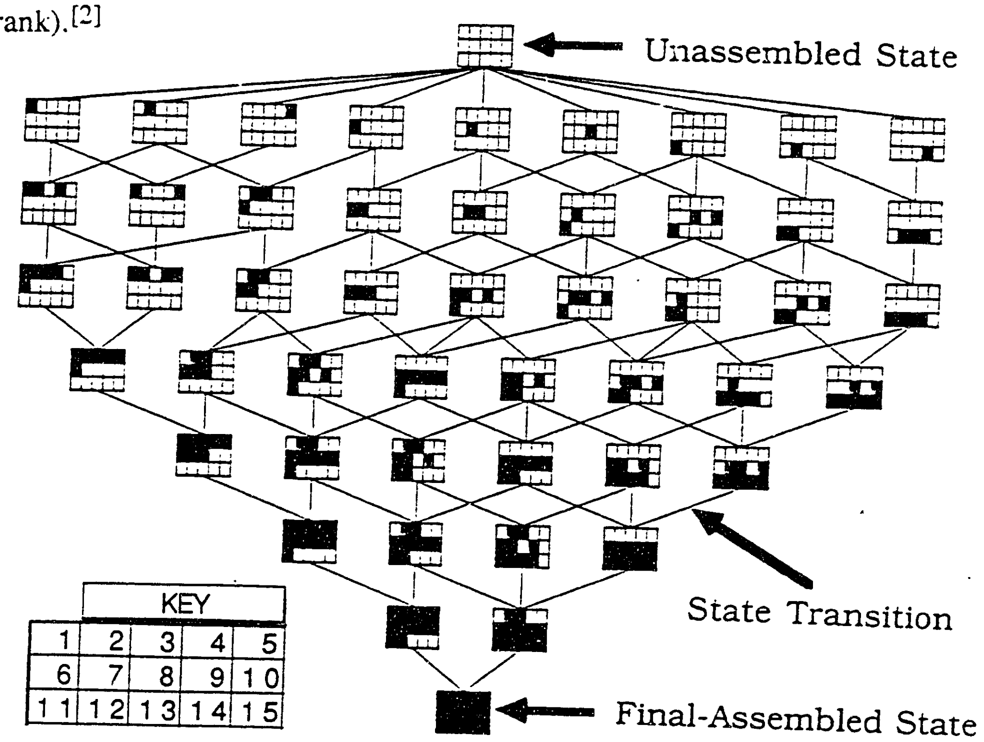


Figure 4.2: A network of sequence graph that represents all valid assembly paths

Source: Max-Cheung Lui, 1988

In our work here, we modify the state representation to ease the legibility of the liaison sequence diagram. Instead of darkening cells to represent the establishment of liaisons, we simply write in the cell the liaison number for all the liaisons completed.

4.2) Characterization and Quantitative Measurement of the Arborescence of an Assembly

In order to determine the most arborescent assembly sequence possible for a product, we need to be able to differentiate between different levels of arborescence. In this section, we characterize arborescence and find quantitative measures of the level of arborescence for an assembly sequence.

4.2.1) Problem Setup

a) Basic definitions

In this chapter, we use *parts-tree* diagrams, such as the one shown in Figure 4.3, to represent assembly systems. The Pi's represent the parts/components of the product. The nodes represent the assembly operations, which each join two parts or subassemblies and output a subassembly. The Li's next to the nodes represent the liaison(s) established at the node.

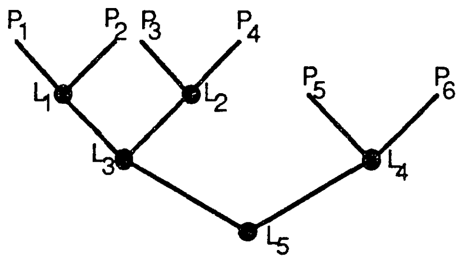


Figure 4.3: Sample parts-tree diagram

The *length* of a branch of a parts-tree is defined as the number of operations in that branch. In the figure 4.4, the length of branch # 1 is 3 and the length of branch # 2 is 2, for instance.

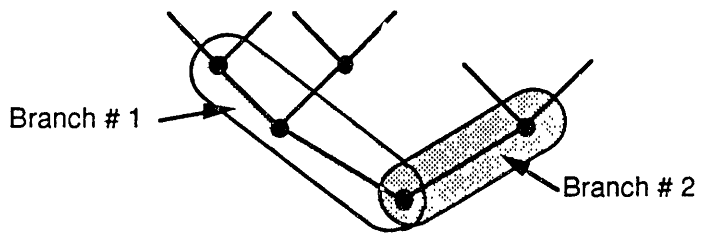


Figure 4.4: Sample Calculation of the Height and the Length of the branches of a parts-tree. Length of Branch # 1 is 3. Length of Branch # 2 is 2. Therefore the Height of the tree is 3.

The *height* of a tree is defined to be the length of its longest branch. The height of the tree shown above is therefore 3.

Two assembly sequences are said to have different *levels of arborescence*, if and only if one of the sequences is more arborescent than the another. Subsequently, we will characterize what is meant by "more arborescent."

b) Introduction

Whereas in a sequential assembly system all the operations are conducted in one main production branch, in an arborescent assembly system a large number of operations are conducted away from the main assembly, where they can be done and tested in parallel. We try to convey this fact in our definition of levels of arborescence.

We distinguish between different level of arborescence lexicographically:

- first we use the **ARBcount** which is a measure inversely proportional to the height of the tree. The bigger is the ARBcount, the more arborescent is the system.

• then in case of tie, we use the measure, described in Section 4.2.3, which is proportional to the width of the parts-tree.

4.2.2) ARBcount and the ARBcount Computing Algorithm

a) Introduction and presentation of algorithm

The ARBcount calculates the number of operations "done away" from the longest branch of the main assembly and "done away" from the longest branch of each of the subassemblies (and their subsubassemblies).

It is calculated in a recursive manner:

$$\text{ARBcount} = (\text{\# of operations}) - (\text{length of biggest subassembly of tree}) + \sum_i \text{ARBcount}_i \qquad (4.1)$$

, where the summation is over the subassemblies left after excluding the longest branch of the biggest subassembly. By bigger, we mean the subassemblies with the most operations or parts.

We illustrate the definition of ARBcount with examples in Figure 4.5.

ARBcount is computed in the following manner:

• 1. Represent the assembly sequence using a parts-tree diagram

• 2. Initialize ARBcount to 0

• 3. Determine the larger of the two subtrees joined by the last operation in the tree. Break ties arbitrarily.

• 4. Calculate (# of operations on the tree) - (length of the longest branch of the biggest subtree). Add that number to the ARBcount.

• 5. Delete the the longest branch of the biggest subtree.

• 6. Apply steps 3, 4 , 5 and 6 to each of the subtrees left on the "disintegrated tree."

b) Sidenotes

Note that determining (# of operations on the tree) - (length of the longest branch of the biggest subtree) in step 4 is equivalent to counting the number of operations in the tree that are not done in the longest branch of the biggest subassembly.

As illustrated in Figure 4.6, breaking ties arbitrarily in step 2 does not affect the value of the ARBcount.

Since ties are broken arbitrarily in step 2, the length of the biggest subassembly (used in equation 4.1) may be different from the height (or depth) of the tree. This point is illustrated in the left-most parts-tree in Figure 4.5. In that tree, the height of the tree is four, whereas the length of the subassembly, chosen to be the biggest, is three. We in fact will take full advantage of this flexibility in breaking ties in the Final ARBcount Maximization Module of Section 4.4.

In equation 4.1, the summation is over subassembly left after the longest branch of the biggest subassembly has been deleted rather than over subassemblies left after the biggest subassembly has been deleted. As illustrated in Figure 4.7, there is a difference between the two when the biggest subassembly has eight or more parts. This difference enables us to account for the arborescence of the main subassembly.

The ARBcount of an assembly system is inversely related to the height (or depth) of the parts-tree. For systems with seven or more operations though, the ARBcount contains more information than the Height. As illustrated in Figure 4.8, the ARBcount enables us to distinguish more readily between systems (with the same part count) that have identical heights and but differing levels of arborescence, i.e.

ARBcount(AS1) = ARBcount(AS2) => Height(AS1) = Height(AS2) but

Height(AS1) = H(AS2) $\not\Rightarrow$ ARBcount(AS1) = ARBcount(AS2)

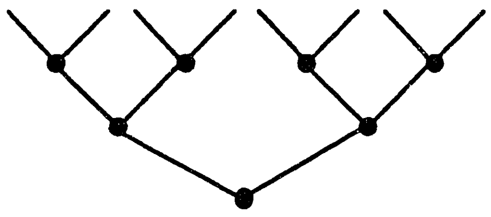
, where the *height of a system* is defined to be the length of the longest production branch of a system

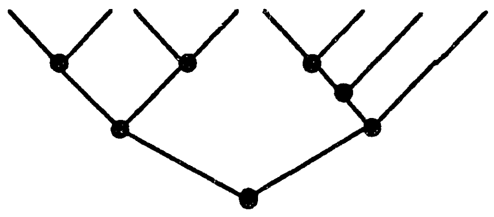
4.2.3) Distinguishing between systems with identical ARBcounts

If two assembly systems AS1 and AS2 have different ARBcounts, then the system with the biggest ARBcount is the most arborescent. Having the same ARBcount does not imply equivalent levels of arborescence, however.

Consider for instance the two assembly systems shown in Figure 4.9. They both have the same ARBcount, i.e they both have one separate subassembly. In AS2 though, the "separate" subassembly is incorporated into the main assembly at a later point in time. Consequently, the operations in the main production branch of AS2 are more decoupled from AS2's "separate" operation. The parts-tree of AS2 is wider than the parts-tree of AS1 tree. AS2 is therefore more arborescent than AS1. Interestingly enough we will see in Section 4.4 that ARBcount can also be used to implement this tie-breaking measure.

To further illustrate the concepts of arborescence, ARBcount, width and height, we have represented in Figure 4.10 all the assembly structures possible for 3, 4, 5, 6, 7-part product in decreasing order of arborescence.

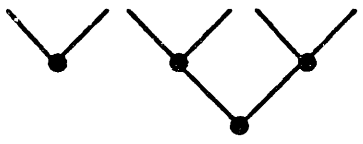


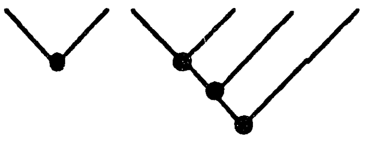


$$\text{ARBcount} = (\text{\# of operations}) - (\text{length of biggest subassembly of tree}) + \sum_i \text{ARBcount}_i$$

=> $\text{ARBcount} = 7 - 3 + \sum \text{ARBcount}_i$ $\quad\quad$ $\text{ARBcount} = 7 - 3 + \sum \text{ARBcount}_i$

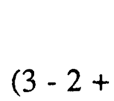
Figure 4.5a: Part-tree representation of two different eight-part, 7-operation arborescent assembly systems. The value of the ARBcount of the trees after the **fourth step** of the ARBcount computation algorithm is shown. In both trees, we arbitrarily picked the left-most 4 part subassembly as the biggest subassembly





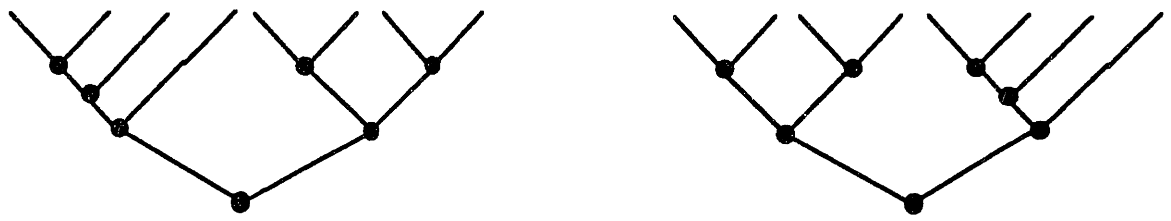
=> $\text{ARBcount} = 4 + \sum \text{ARBcount}_i$ $\quad\quad$ $\text{ARBcount} = 4 + \sum \text{ARBcount}_i$

Figure 4.5b: Representation of the "disintegrated tree" of the two above eight-part arborescent assembly systems, resulting from the **fifth step** of the ARBcount computation algorithm



$\text{ARBcount} = 4 + (1-1+0) + (3 - 2 + 0) = 5$ $\quad\quad$ $\text{ARBcount} = 4 + (1-1+0) + (3-3+0) = 4$

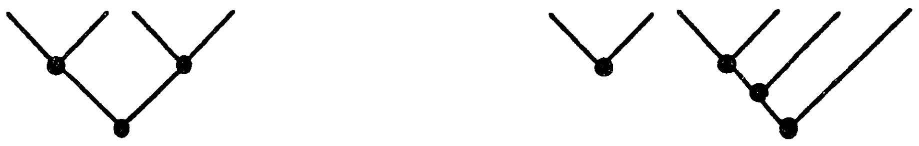
Figure 4.5c: Representation of the "disintegrated tree" of the two above eight-part arborescent assembly systems, resulting from steps 3, 4 ,5 and 6 of the ARBcount computation algorithm have been applied to the trees of Figure 4.5b.



ARBcount = (# of operations) - (length of biggest subassembly of tree) + $\sum_i$ ARBcount $_i$

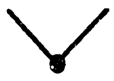
=> ARBcount = 7 - 4 + $\sum$ ARBcount $_i$ ARBcount = 7 - 3 + $\sum$ ARBcount $_i$

Figure 4.6a: Two different part-tree representations of an eight-part 7-operation arborescent assembly system. To illustrate the point that breaking ties arbitrarily in step 2 of the ARBcount Computing Algorithm does not affect the value of the ARBcount, we assume that the left most 4-part subassembly is the biggest subassembly in each of the two trees. The value of the ARBcount of the two trees after step 4 is shown.



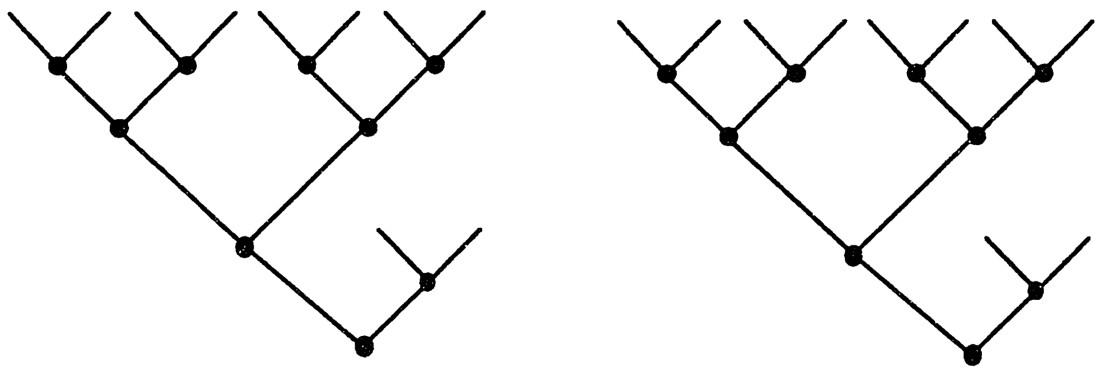
ARBcount = 3 + $\sum$ ARBcount $_i$ ARBcount = 4 + $\sum$ ARBcount $_i$

Figure 4.6b: Representation of the "disintegrated tree" of the two part-trees from Figure 4.6a, resulting from the fifth step of the ARBcount Computation Algorithm



ARBcount = 3 + 1 = 4 <=> ARBcount = 4 + 0 = 4

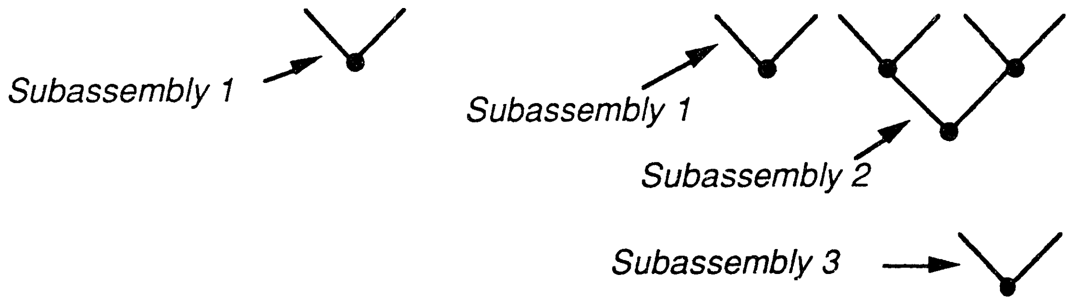
Figure 4.6c: Representation of the "disintegrated tree" of the two above eight-part arborescent assembly systems, after steps 3, 4, 5 and 6 of the ARBcount computation algorithm have been applied to the disintegrated trees of Figure 4.6b. As shown here, breaking ties arbitrarily in step 2 of the algorithm does affec: the final value of the ARBcount



ARBcount = (# of operations) - (length of biggest subassembly of tree) + $\sum_{i}$ ARBcount $_i$

=> ARBcount = 9 - 4 + Σ ARBcount $_i$ ARBcount = 9 - 4 + Σ ARBcount $_i$

Figure 4.7a: Duplication of the part-tree representation of a ten-part 9-operation arborescent assembly system. The left-most tree is used to calculate the ARBcount that would have resulted had we deleted the biggest subassembly in Step 5 of the ARBcount Computation Algorithm (A.B.C.) . The right-most tree is used to calculate the ARBcount as it is defined in Section 4.2.2. The value of the ARBcount of the two trees after step 4 is shown.



ARBcount $_1$ = 1 - 1 + 0 = 0

ARBcount = 5 + Σ ARBcount $_i$

=> ARBcount = 5 + 0 = 5

ARBcount $_1$ = 1 - 1 +0 = 0
ARBcount $_2$ = 3 - 2 + 0 = 1
ARBcount $_3$ = 1 - 1 + 0 = 0
ARBcount = 5 + Σ ARBcount $_i$
ARBcount = 5 + 0 + 1 + 0 = 6

Figure 4.7b: Shown on the left is the "disintegrated tree", that would have resulted from the fifth step of the A.B.C.had we decided to delete the biggest subassembly instead of the longest branch of the biggest subassemblies. The "disintegrated tree" on the right results from the A.B.C as it is defined in Section 4.2.2. The calculation of the ARBcount of the subassemblies of the disintegrated trees is shown. The tree on the right has a higher ARBcount. Deleting the longest branch of the biggest subassembly instead of the whole biggest subassembly allows us to account for the arborescence of the biggest subassembly.

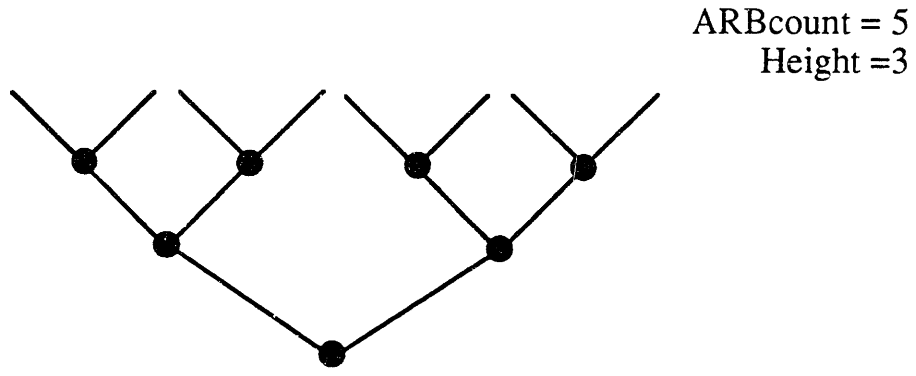


Figure 4.8a:

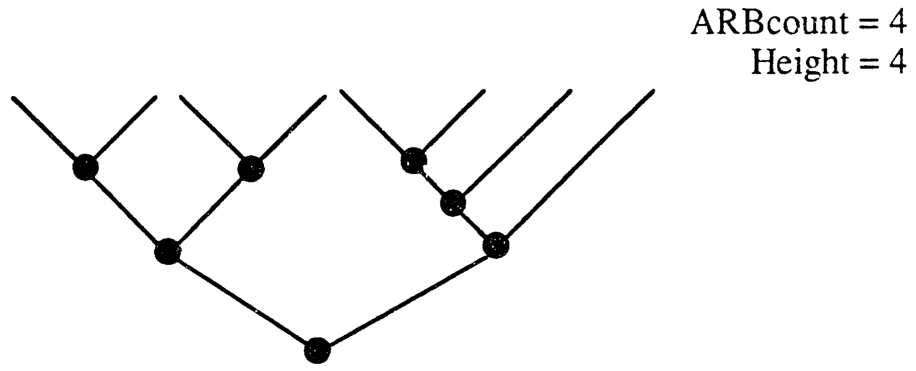


Figure 4.8b:

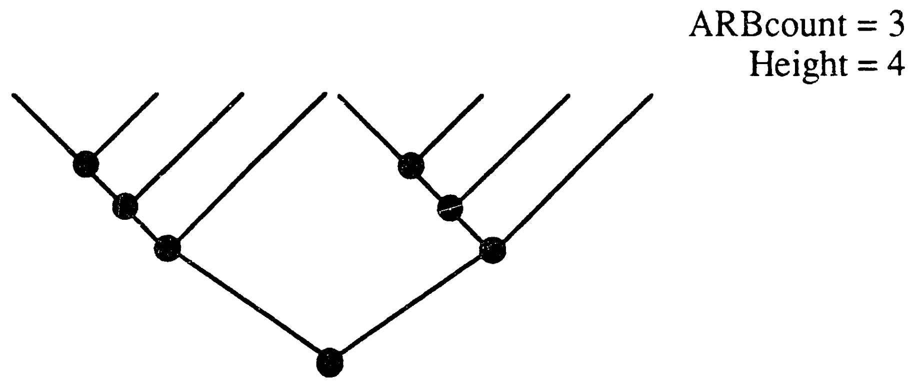


Figure 4.8c:

Figures 4.8a,b,c: Parts-tree diagram representation of three eight-part arborescent systems with differing levels of arborescence. Note that while the ARBcount makes a distinction between the level of arborescence of the systems in Figure 4.8b and Figure 4.8c, the Height does not.

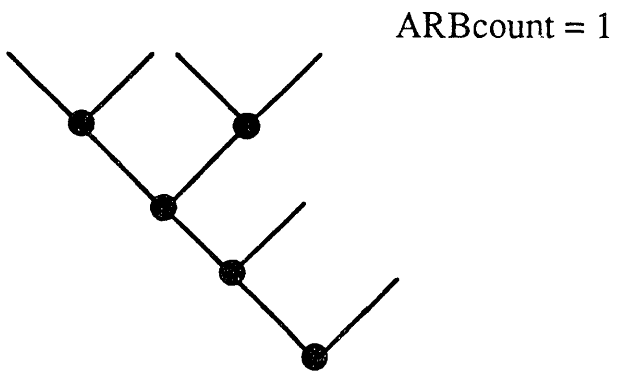


Figure 4.9a: Parts tree diagram representation of a generic 7-operation assembly system called AS1.

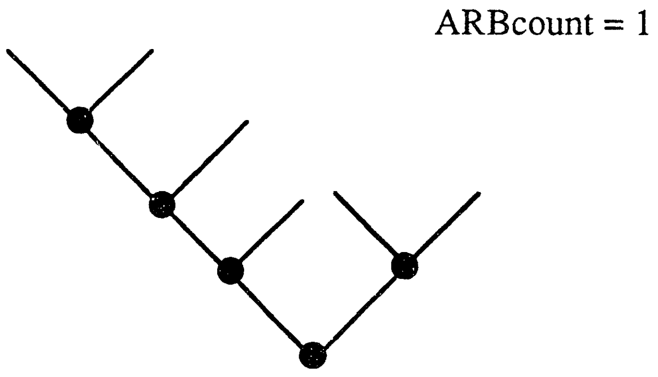


Figure 4.9b: Parts tree diagram representation of a generic 7-operation assembly system called AS2.

Figures 4.9a,b: AS1 and AS2 have the same ARBcount. But since AS2's parts tree is wider than AS1's, AS2 is the more arborescent of the two trees.

ℕ = 3

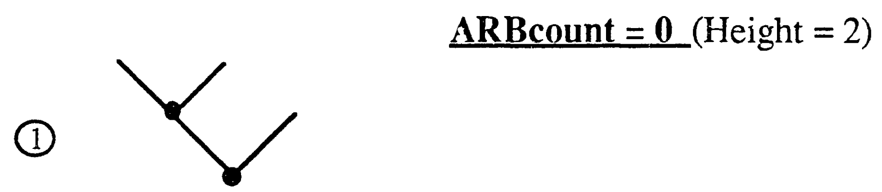


Figure 4.10a : Parts-tree diagram representation of the only assembly structure possible for the assembly of 3-part products

ℕ = 4

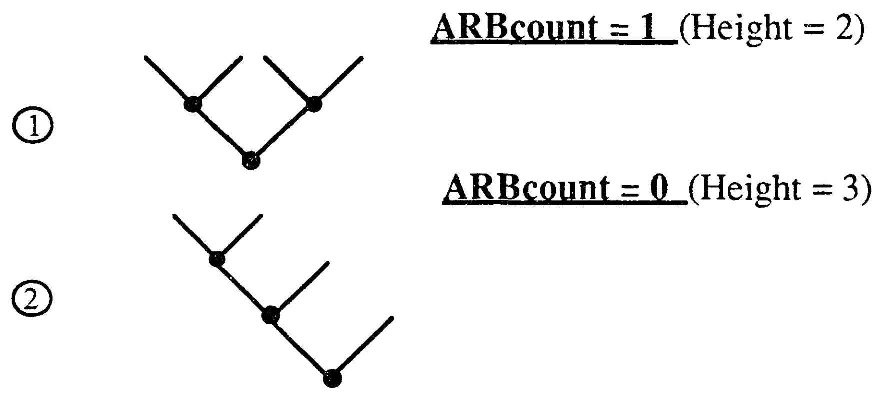


Figure 4.10b: Parts-tree diagram representation of all the assembly structures possible for the assembly 4-part product, in decreasing order of arborescence

ℕ = 5

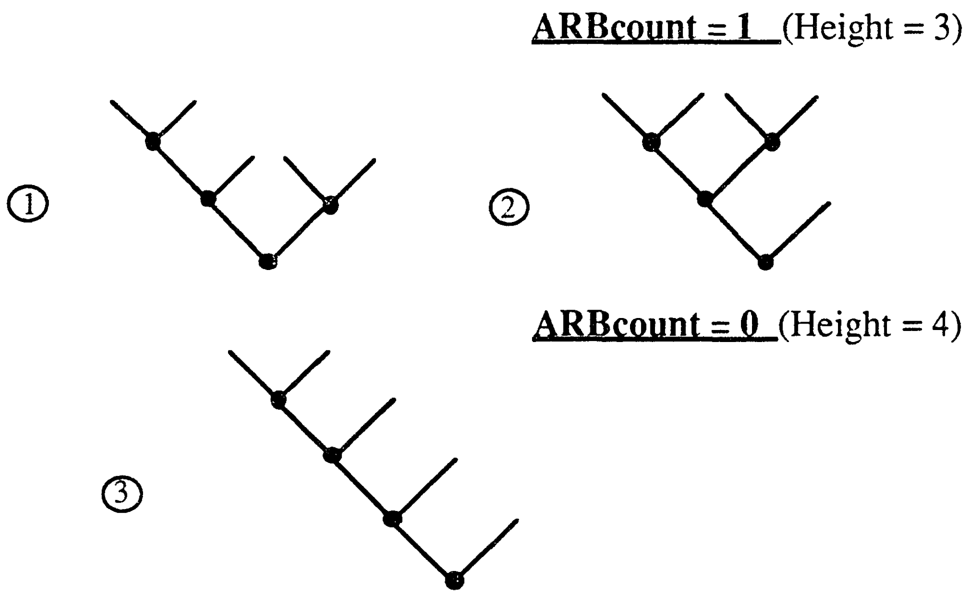
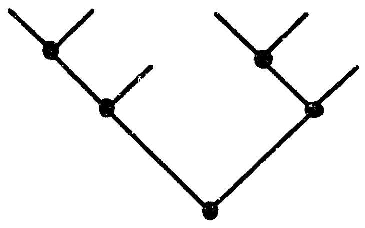
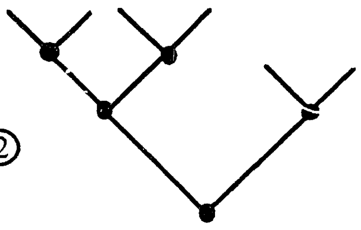


Figure 4.10c: Parts-tree diagram representation of all the assembly structures possible for the assembly 5-part product, in decreasing order of arborescence

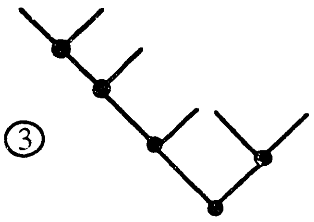
ℕ = ⑥

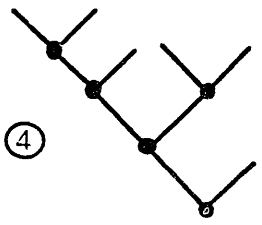
ARBcount = 2 (Height = 3)

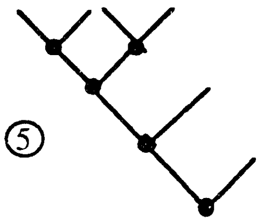
①


②


ARBcount = 1 (Height = 4)

③


④


⑤


ARBcount = 0 (Height = 5)

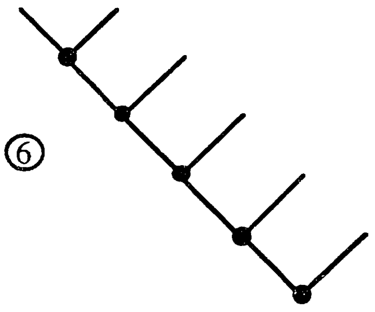
⑥


Figure 4.10d: Parts-tree diagram representation of all the assembly structures possible for the assembly 6-part product, in decreasing order of arborescence

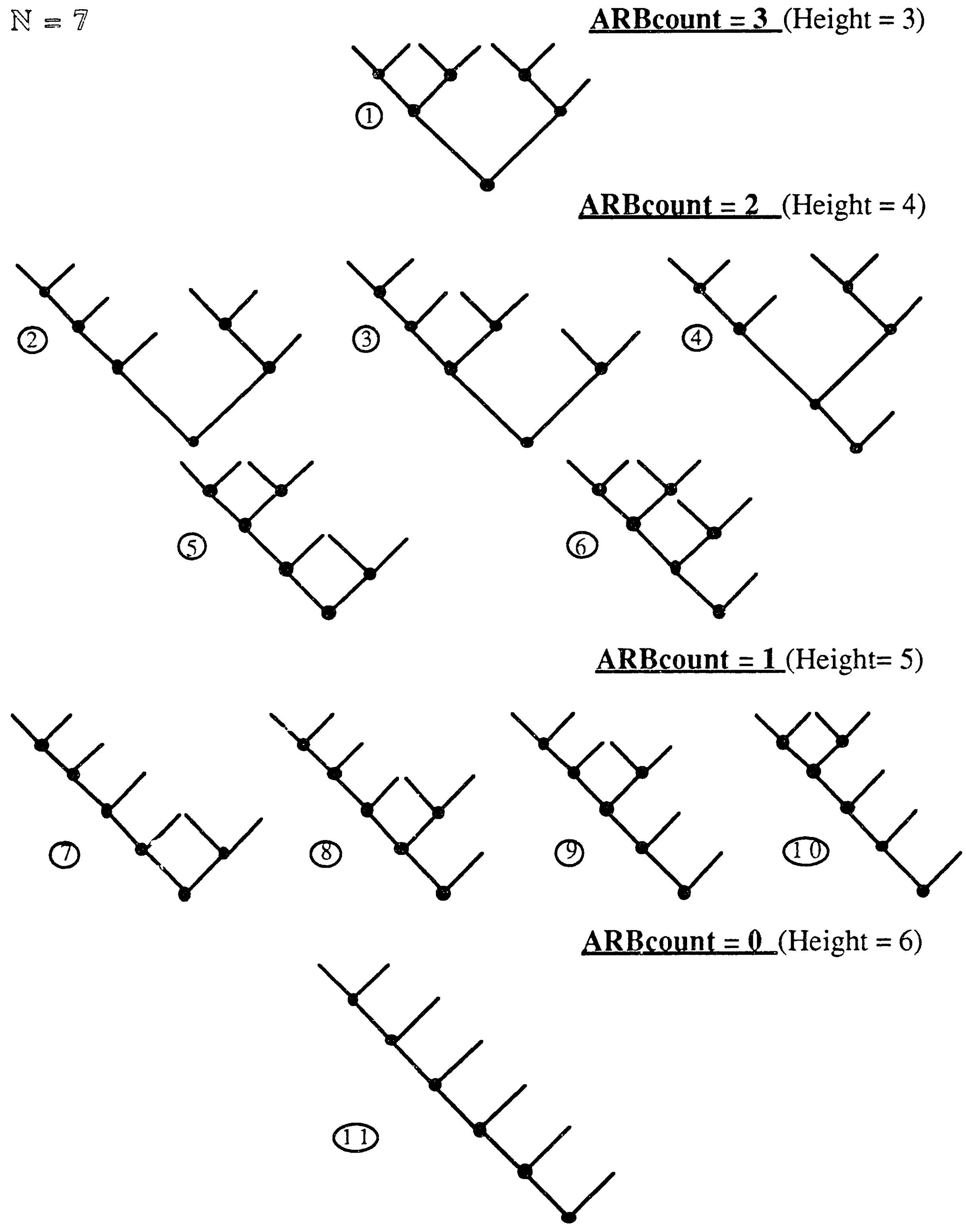

N = 7
ARBcount = 3 (Height = 3)
1
ARBcount = 2 (Height = 4)
2
3
4
5
6
ARBcount = 1 (Height= 5)
7
8
9
10
ARBcount = 0 (Height = 6)
11

Figure 4.10e: Parts-tree diagram representation of all the assembly structures possible for the assembly 7-part product, in decreasing order of arborescence

4.2.4) Hints for implementation of the algorithm

(The reader, who only wants to get a broad understanding of the issues involved, may wish to skip to section 4.3)

• When N is less that 5, then two systems with the same ARBcount necessarily have the same level of arborescence.

• When N is less than 7, if we do not wish to use the ARBcount approach, we can also distinguish between the level of arborescence of two systems by comparing the size of the smallest subassembly of the main subassembly at each stage, starting with the last stage, until either we find a stage where the two systems have differing smallest subassembly size or until stage 1 is reached. The bigger the smallest subassembly, the less disproportionate in size are the subassemblies being mated and the more arborescent the system. For these cases, the ARBcount does not necessarily need to be calculated. Unfortunately as illustrated in Figure 4.11, when N $\geq$ 7, a system could have a larger subassembly at the last stage and be less arborescent. Consequently the ARBcount needs to be calculated for these values of n.

• When N is less than 8 or when the size of the largest "separate" subassembly is less than 4, two systems have the same level of arborescence if and only if the size of the smallest subassembly of the main subassembly of the two systems is the same at every stage.

Unfortunately, if the largest "separate" subassembly is made up of four or more parts, then two systems, with subassemblies of the same size at every stage, have the same level of arborescence only if the "separate" subassemblies of more than four parts have the same level of arborescence.

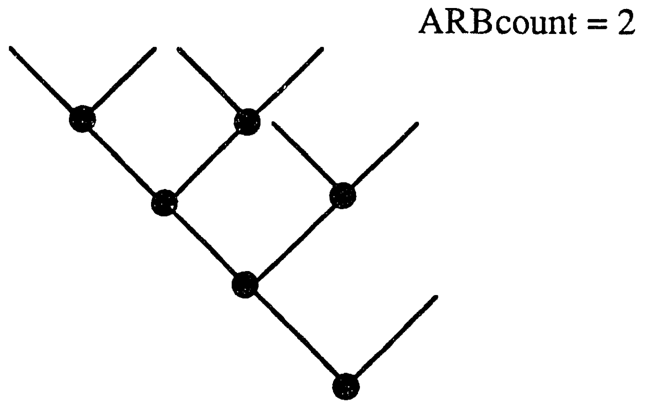


Figure 4.11a:

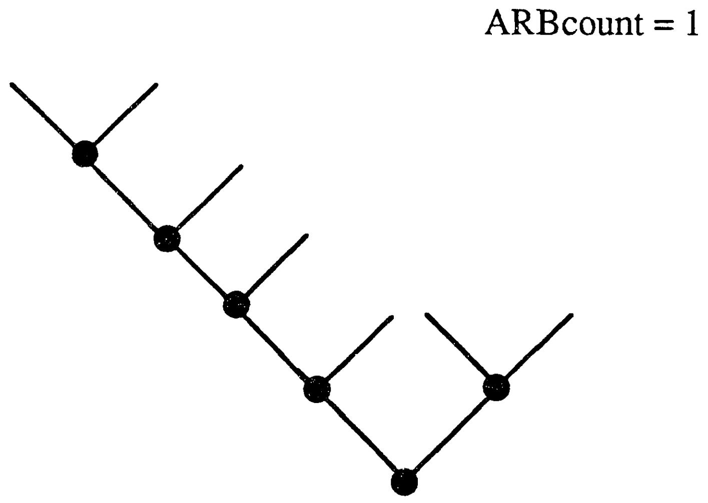


Figure 4.11b:

Figures 4.11a,b: Parts tree diagram representation of two different arborescent part groupings for a generic, seven part product. For products with less than seven parts, if two systems have smallest subassemblies of differing size at the last stage, the system with the largest smallest subassembly is the most arborescent. However, as illustrated here, for products with seven or more parts, a system, like the one in Figure 4.11b, could have a larger subassembly in the last stage and be less arborescent.

4.3) THE ARBORESCENT PURGE MODULE

4.3.1) Introduction and Motivations

There is a certain amount of redundancy in the LSA software in the generation of assembly sequences which include multiple and separate subassemblies. Consider an eight-part product, with parts P1, P2 ... P8. The purely arborescent assembly, shown in Figure 4.12a, where P1 is mated with P2, P3 is mated with P4, P1&P2 are mated with P3&P4, and where P5 is mated with P6, P7 is mated with P8, P5&P6 are mated with P7& P8, and where finally P1&P2&P3&P4 is mated with P5&P6&P7&P8, generates the 80 sequential assembly sequences described in Figure 4.12b. These 80 "sequences" essentially all describe the same *part grouping*, i.e. the same strategy for assembling the product. Certainly, all 80 have the same level of arborescence. A product designer using LSA should have the option of deleting the79 or so "redundant" sequences.

Having this pruning option is especially crucial for large products since the Number of Sequential assembly Sequences corresponding to a Purely Arborescent part grouping, (call it NSSPA) rises faster than n/2!. For n=4, NSSPA = 2, for n =8, NSSPA = 80 and for n= 16, NSSPA >> 2 10^6 (i.e. >> 8! * 4! * 2! 1!)

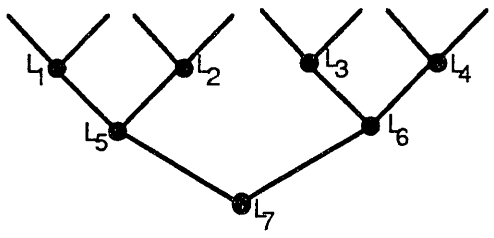


Figure 4.12a: Part-tree representation of the eight-part, 7-liaison purely arborescent assembly system, whose eighty sequential assembly sequences are described in Figure 4.12b. Here, L1, L2, L3 and L4 are the first-stage liaisons. L5 and L6 are the second-stage liaisons.

Assembly sequences where all the first stage liaisons are done first:

	— — — —		— —	—		
	First stage Liasions (L1, L2, L3, L4)		2nd Stage Liaisons (L5, L6)	3rd Stage L7		
# of such sequences:	(4 * 3 * 2 * 1)	*	(2 * 1)	* 1	=	**48**

Assembly sequences where three first stage liaisons are done first and where the first second stage liasion is done before the fourth first-stage liasion:

	— — —		—		— —	—		
	3 of the first stage Liasions (L1, L2, L3 or L4)		1st 2nd Stage Liaison		Last 1st stage Liaison followed by last 2nd stage Liaison	3rd stage L7		
# of such sequences:	(4 * 3 * 2)	*	1	*	(1 * 1)	* 1	=	**24**

Assembly sequences where two first stage liaisons are done first and where the second-stage liasion joining these two subassemblies is done before the other two first-stage liasions:

	— —	—		— —	—	—		
	2 first stage Liasions (L1&L2 or L3&L4)	1st 2nd Stage Liaison		Last 2 1st Stage Liaisons	2nd 2nd-stage Liaison	3rd Stage L7		
# of such sequences:	(4 * 1)	* 1	*	(2 * 1)	* 1	* 1	=	**8**

Total number of sequential sequences corresponding to the arborescent parts gouping shown in Figure 4.10a = **80**

Figure 4.12b: Breakdown of all the sequential assembly sequences corresponding to the arborescent parts grouping shown in Figure 4.12a

Note that for a purely sequential assembly, no distinction needs to be made between part groupings and assembly sequences, because each possible way of grouping parts sequentially has one and only one corresponding sequential assembly sequence.

4.3.2) Description of the pruning algorithm

Before developing an algorithm to delete the "redundant" sequential sequences of an arborescent part grouping, we needed to decide which one of the many sequential sequences corresponding to each arborescent part grouping to keep. We explain our decision in terms of parts-tree diagram representation.

We decided to keep the assembly sequence, which corresponds to the assembly of parts and subassemblies from top-to- bottom, left-to-right as they are shown in a parts-tree diagram, where the larger of any two subassemblies, to be mated by an operation, is always represented on the left. The implication of this convention in the liaison sequence diagram is the following.

In a liaison sequence diagram (e.g. Figure 4.13), if there exists an intermediate assembly state, call it STi, consisting of multiple, distinct subassemblies, then we only keep the assembly sequence leading to STi where all the subassemblies are built one at a time from start to finish in decreasing order of size until state STi is reached. In other words, we first build the biggest subassembly in its entirety, then we build the second biggest subassembly in its entirety, and so on until we finally build the smallest subassembly and reach STi.

Once the LSA software has generated the entire liaison sequence diagram, we implement this pruning in the following manner:

Consider a state STi, which consists of two or more subassemblies S1, S2 ... Sk, for $k \geq 2$

- we first identify Ss, the smallest subassembly in terms of number parts. If there is a tie, we break the tie in a manner described below.
- we then delete all the transitions to state STi from parent states in which the smallest subassembly of Ss is already established. This step insures that Ss is the last of the k subassemblies done.

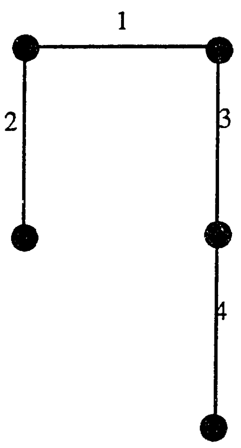


Figure 13a:

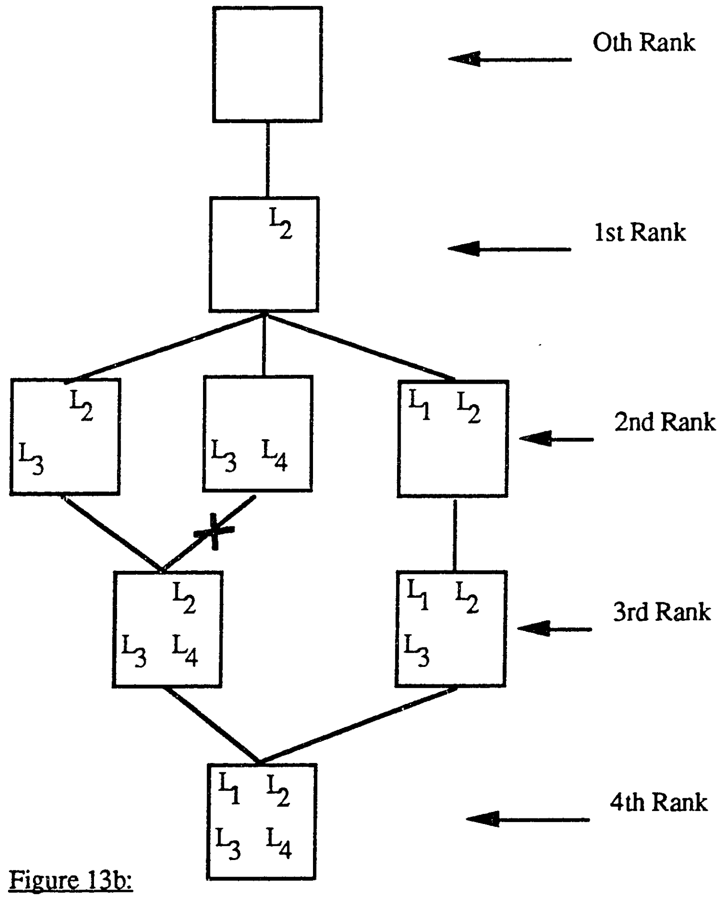


Figure 13b:

Figure 13a,b: The liaison diagram and the sequence diagram of a generic product are shown in Figures 13a and 13b. In Figure 13b, the left-most state of the 3rd rank consists of one three-part subassembly and one two-part subassembly. Consequently we only keep the transitions from the parent state, where the three part subassembly is built in its entirety.

By applying this technique to every state STi we insure, that we only keep those assembly sequences leading to STi, where all the subassemblies are built one at a time from start to finish in order of decreasing size.

Note that no sequential assembly sequence or partially sequential assembly sequence is deleted accidentally, since only the states (in ranks 2 through n -1) with two or more subassemblies are examined.

We have made no mention thus far of how to break off ties between subassemblies with the same number of parts. Ties could be broken in any arbitrary way. However, for reasons that will be apparent when we describe the Arborescence Maximization Module, we have chosen the following tie-breaking rule, which is more computationally intensive in the short run, but which is overall more efficient, since it makes trivial the finding of the most arborescent assembly:

- if two subassemblies S1 and S2 are made up of the same number of parts, the subassembly with the highest level of arborescence is built first. Here, S1 is said to have a higher level of arborescence than S2 if S1's most arborescent configuration is more arborescent than S2's most arborescent configuration. We point this out, since individual subassemblies are built in "subsequences" with varying levels of arborescence.
- if the two subassemblies have the same level of arborescence, the tie is broken in an arbitrary manner. The subassembly with the liaison with the highest index is built last.

4.3.3) Implementation of the pruning algorithm

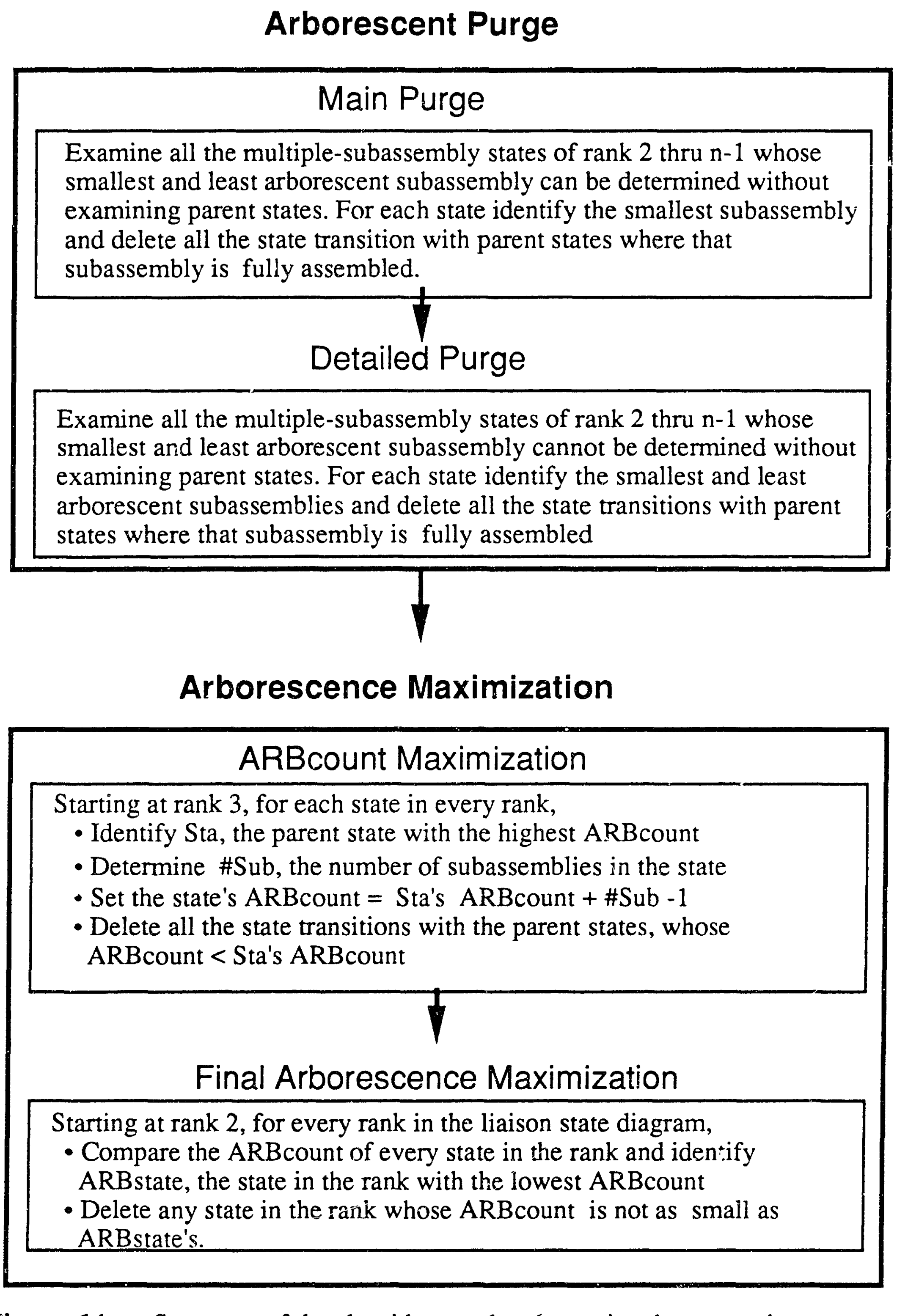
For states with one smallest subassembly or with multiple smallest subassemblies of less than four parts, the above pruning technique is straight forward to implement, because in either case the level of arborescence is not a factor. In the one-smallest subassembly case, there is no ambiguity about the identity of the smallest subassembly. In the case of several smallest subassemblies of less than four parts, i.e in case of smallest subassemblies of two or three parts, the level of arborescence is necessarily the same in subassemblies of equal size.

For states with several smallest subassemblies of four or more parts, the implementation of the technique requires the determination of the level of arborescence of each of smallest subassemblies and is thus more computationally intensive. The fewer the number of parent states, the quicker the determination of the level of arborescence. Consequently, we postpone the determination of the level of arborescence for these cases until the latest possible moment.

As shown in the flowcharts of Figures 4.14a and 4.14b , the Arborescent Purge Module is thus divided in two sections:

- The Main Purge
- The Detailed Purge

In the Main Purge, we examine all the multiple-subassembly states of rank 2 thru n-1 with one smallest subassembly or with several smallest subassemblies of less than four parts. For each of these states we identify the smallest subassembly and delete all the transitions from parent states where the entire smallest subassembly is assembled. A flowchart of the Main Purge is shown in Figure 4.14c.



Arborescent Purge

Main Purge

Examine all the multiple-subassembly states of rank 2 thru n-1 whose smallest and least arborescent subassembly can be determined without examining parent states. For each state identify the smallest subassembly and delete all the state transition with parent states where that subassembly is fully assembled.

Detailed Purge

Examine all the multiple-subassembly states of rank 2 thru n-1 whose smallest and least arborescent subassembly cannot be determined without examining parent states. For each state identify the smallest and least arborescent subassemblies and delete all the state transitions with parent states where that subassembly is fully assembled

Arborescence Maximization

ARBcount Maximization

Starting at rank 3, for each state in every rank,
- Identify Sta, the parent state with the highest ARBcount
- Determine #Sub, the number of subassemblies in the state
- Set the state's ARBcount = Sta's ARBcount + #Sub -1
- Delete all the state transitions with the parent states, whose ARBcount < Sta's ARBcount

Final Arborescence Maximization

Starting at rank 2, for every rank in the liaison state diagram,
- Compare the ARBcount of every state in the rank and identify ARBstate, the state in the rank with the lowest ARBcount
- Delete any state in the rank whose ARBcount is not as small as ARBstate's.

Figure 14a: Summary of the algorithm used to determine the most arborescent assembly sequence possible for the assembly of a product.

Arborescent purge

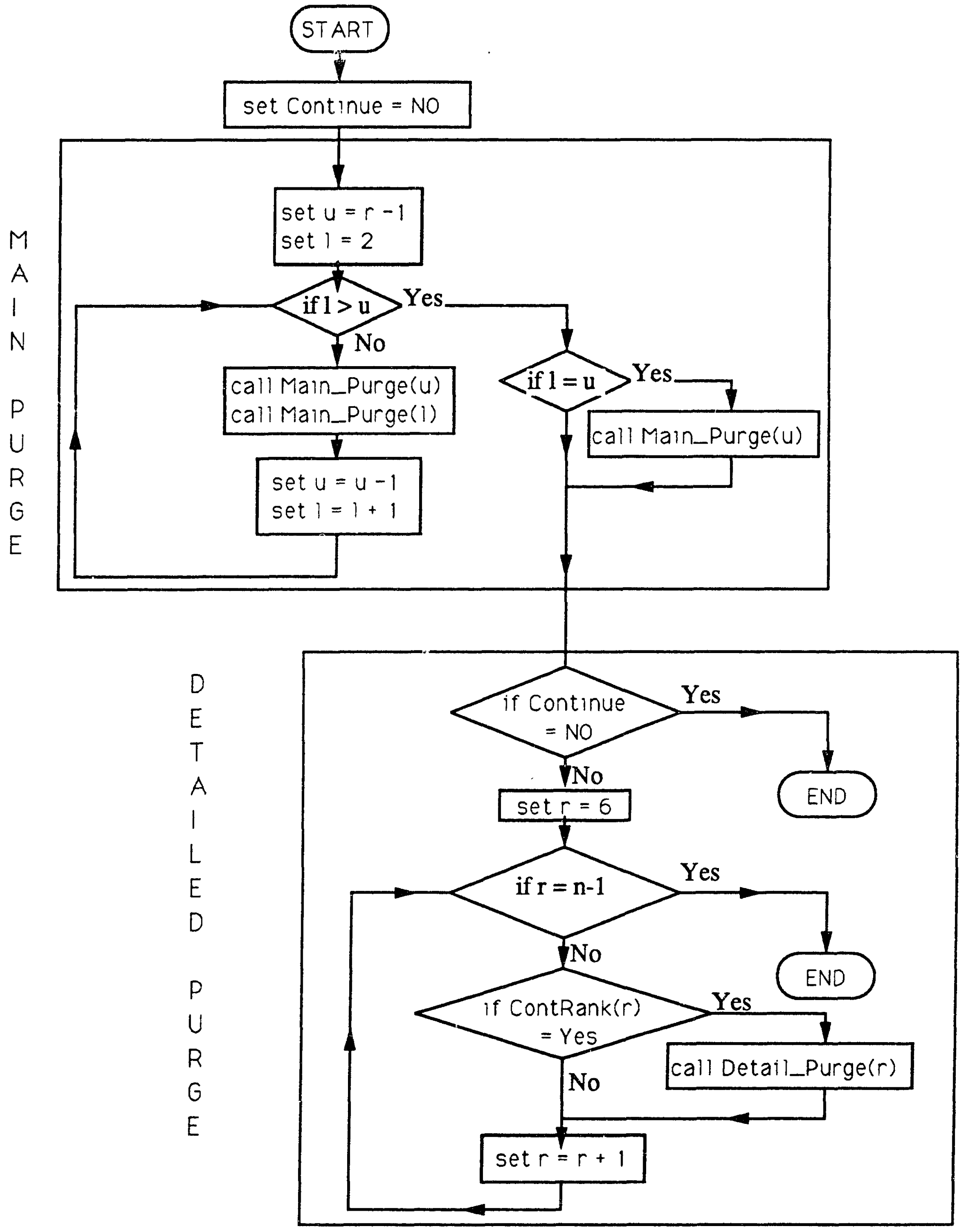
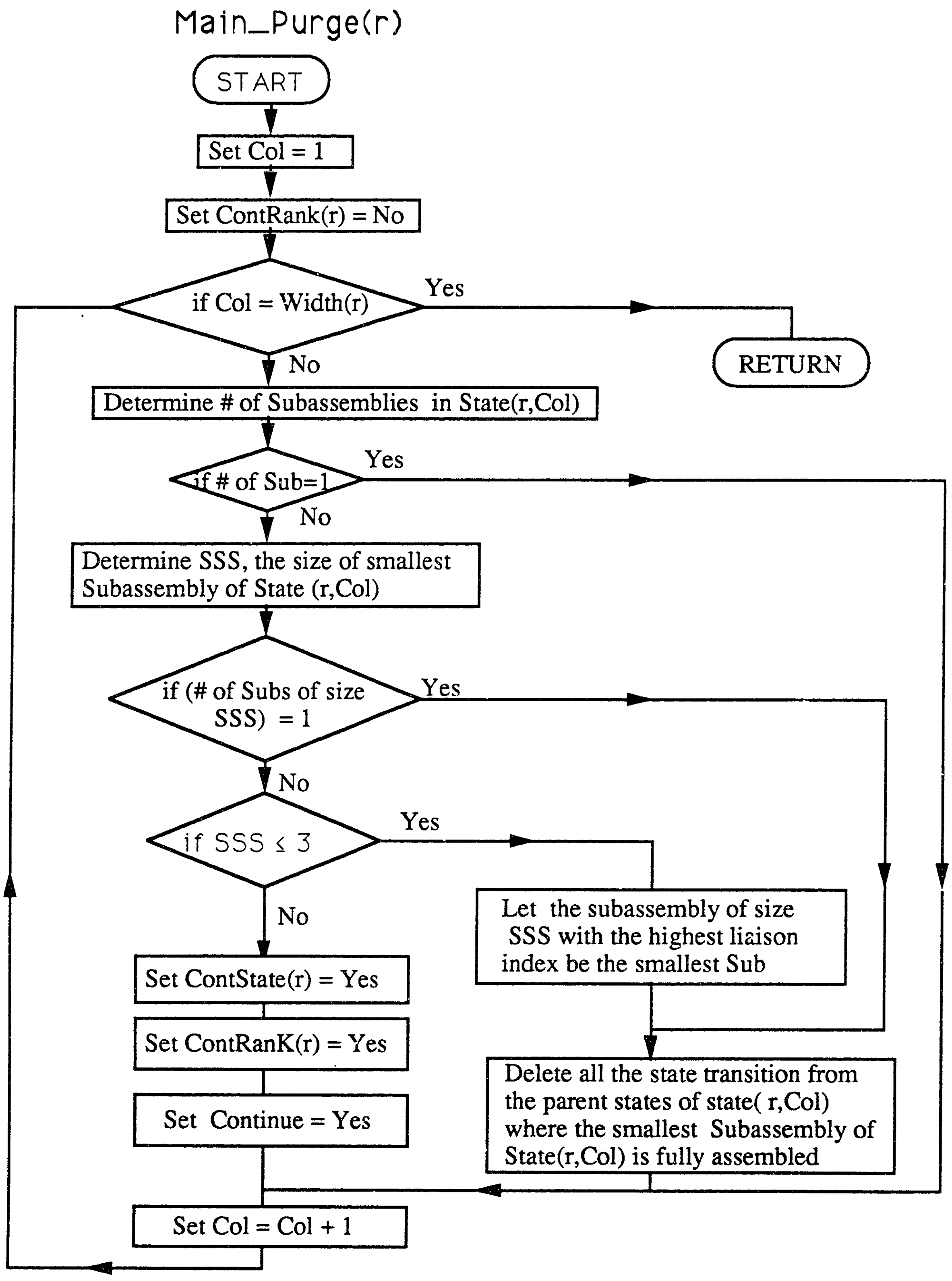


Figure 4.14b: Flowchart for the Arborescent Purge Module. This module deletes all the "redundant" arborescent assembly sequences in a product's liaison sequence diagram. The flowchart for the Main_Purge routine is shown in Figure 4.14c, the one for the Detailed_Purge is shown in Figure 4.14d. Note that r stands for rank.



Main_Purge(r)
START
Set Col = 1
Set ContRank(r) = No
if Col = Width(r)
Yes
No
RETURN
Determine # of Subassemblies in State(r,Col)
if # of Sub=1
Yes
No
Determine SSS, the size of smallest Subassembly of State (r,Col)
if (# of Subs of size SSS) = 1
Yes
No
if SSS ≤ 3
Yes
No
Let the subassembly of size SSS with the highest liaison index be the smallest Sub
Set ContState(r) = Yes
Set ContRanK(r) = Yes
Set Continue = Yes
Delete all the state transition from the parent states of state(r,Col) where the smallest Subassembly of State(r,Col) is fully assembled
Set Col = Col + 1

Figure 4.14c: Flowchart for the Main_Purge routine, which is called from the Main Purge Module shown in Figure 4.14b and described in Figure 4.14a. Note that Width refers to the width of a rank, i.e. the number of states in the rank.

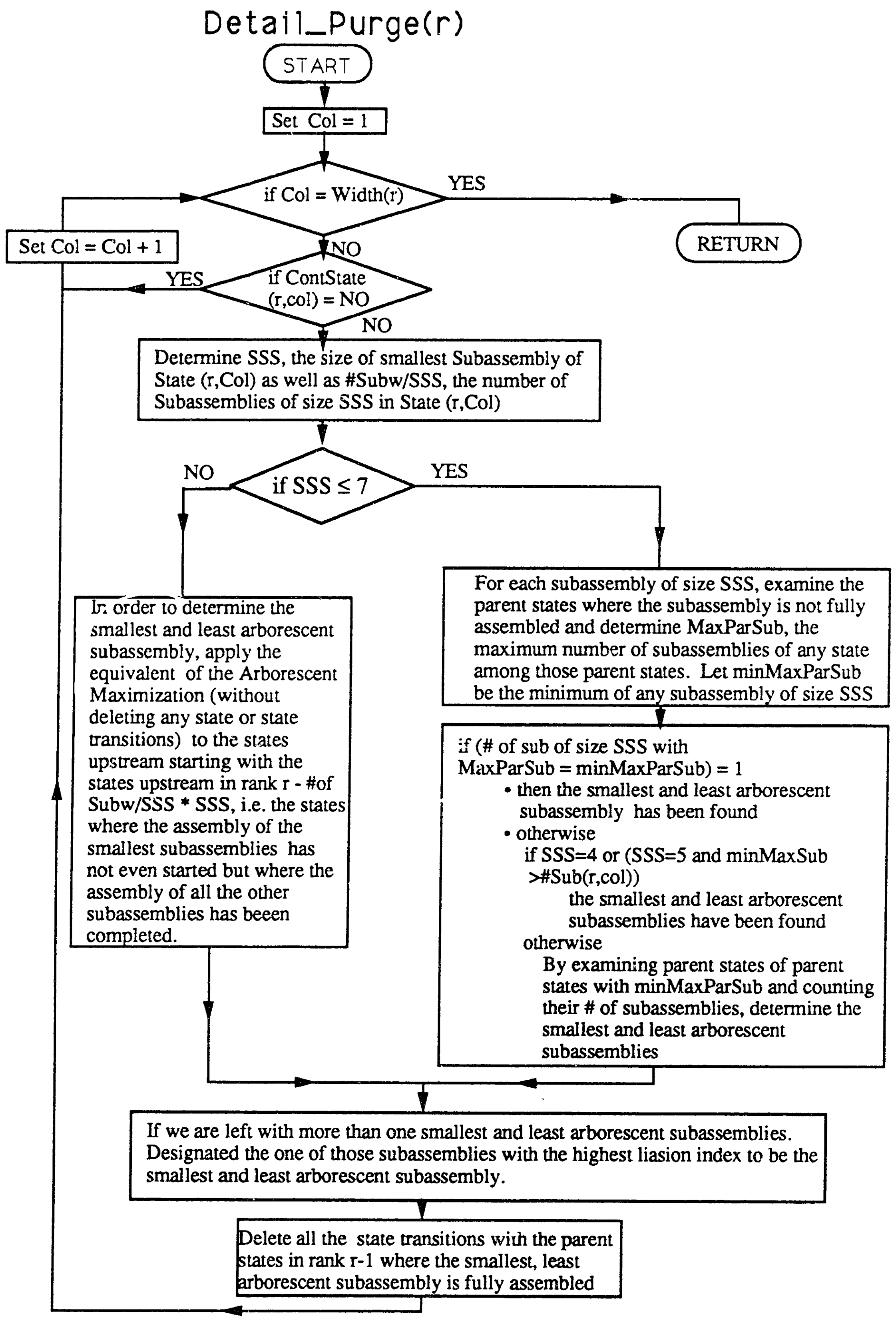

Detail_Purge(r)
START
Set Col = 1
if Col = Width(r)
YES
RETURN
Set Col = Col + 1
NO
YES
if ContState (r,col) = NO
NO
Determine SSS, the size of smallest Subassembly of State (r,Col) as well as #Subw/SSS, the number of Subassemblies of size SSS in State (r,Col)
NO
if SSS ≤ 7
YES
In order to determine the smallest and least arborescent subassembly, apply the equivalent of the Arborescent Maximization (without deleting any state or state transitions) to the states upstream starting with the states upstream in rank r - #of Subw/SSS * SSS, i.e. the states where the assembly of the smallest subassemblies has not even started but where the assembly of all the other subassemblies has beeen completed.
For each subassembly of size SSS, examine the parent states where the subassembly is not fully assembled and determine MaxParSub, the maximum number of subassemblies of any state among those parent states. Let minMaxParSub be the minimum of any subassembly of size SSS
if (# of sub of size SSS with MaxParSub = minMaxParSub) = 1
• then the smallest and least arborescent subassembly has been found
• otherwise
if SSS=4 or (SSS=5 and minMaxSub >#Sub(r,col))
the smallest and least arborescent subassemblies have been found
otherwise
By examining parent states of parent states with minMaxParSub and counting their # of subassemblies, determine the smallest and least arborescent subassemblies
If we are left with more than one smallest and least arborescent subassemblies. Designated the one of those subassemblies with the highest liasion index to be the smallest and least arborescent subassembly.
Delete all the state transitions with the parent states in rank r-1 where the smallest, least arborescent subassembly is fully assembled

Figure 4.14d: Flowchart for the Detail_Purge routine, which is called from the Detailed Purge Module shown in Figure 4.14b and described in Figure 4.14a.

In the Detailed Purge, we examine all the multiple subassembly states of rank 6 through n-1 in which the smallest subassembly is not unique and has four or more parts. For each of those states, we determine the level of arborescence of each of the smallest subassemblies and identify the smallest and least arborescent subassembly. We then proceed to delete all the transitions from parent states where the entire smallest subassembly is done.

a)Hints for implementation

In the Detailed Purge, we distinguish between states whose smallest subassemblies are less than than 7 parts and those with smallest subassemblies have more than 7 parts.

As mentioned in section 4.2.4, we can distinguish between the level arborescence of two subassemblies of 7 or less parts by comparing the smallest subassemblies of these subassemblies at each stage until either we find a stage where their smallest subsubassemblies differ in size or until we reach the stage where they are totally disassembled. However, for subassemblies of more than 7 parts, a subassembly, such as the one shown in Figure 4.11, could have a larger subsubassembly at the last stage and be less arborescent. We therefore need to calculate the ARBcount of these subassemblies in order to distinguish between their level of arborescence. A flowchart of the Detailed Purge is shown in Figure 4.14d

4.3.4) Example of the Arborescent Purge:

In Figure 4.15c thru 4.15g we have represented the liaison sequence diagram for a generic product with the liaison diagram shown in Figure 4.15a, at different stages of the Arborescence Purge.

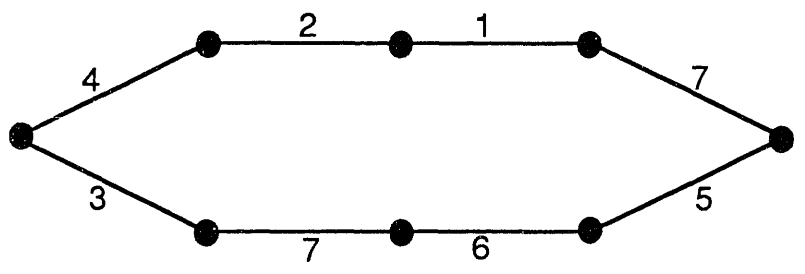


Figure 4.15a:Liaison diagram for the generic eight-part. For the sake of simplicity, it is assumed that the part grouping shown in Figure 4.15b is the only possible part grouping for this product.

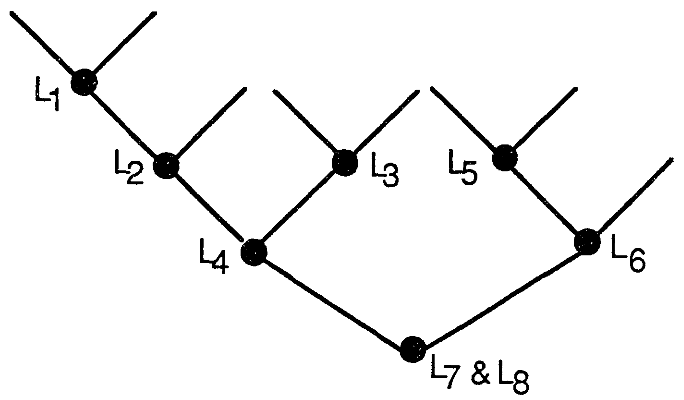


Figure 4.15b:Parts tree diagram representation of the only possible part grouping possible for the product whose liaison diagram is shown in Figure 4.15a. The Liaison Sequence Diagram for this product is shown at different stages of the Arborescent Purge in Figures 4.15c thru 4.15e

We also applied the Arborescent Purge Algorithm to the eight-part modified version of the example of Assembly From Industry (A.F.I.) shown in Figure 4.16. Because of the large number of liaisons and degrees of freedom in this example, the liaison sequence diagram for this example is too large to be shown here. The impact of the Arborescent Purge in this case is worth mentioning though. Whereas the original liaison sequence diagram contained 1008 assembly sequences, only 81 were left after the Main Purge and 71 after the Detailed Purge.

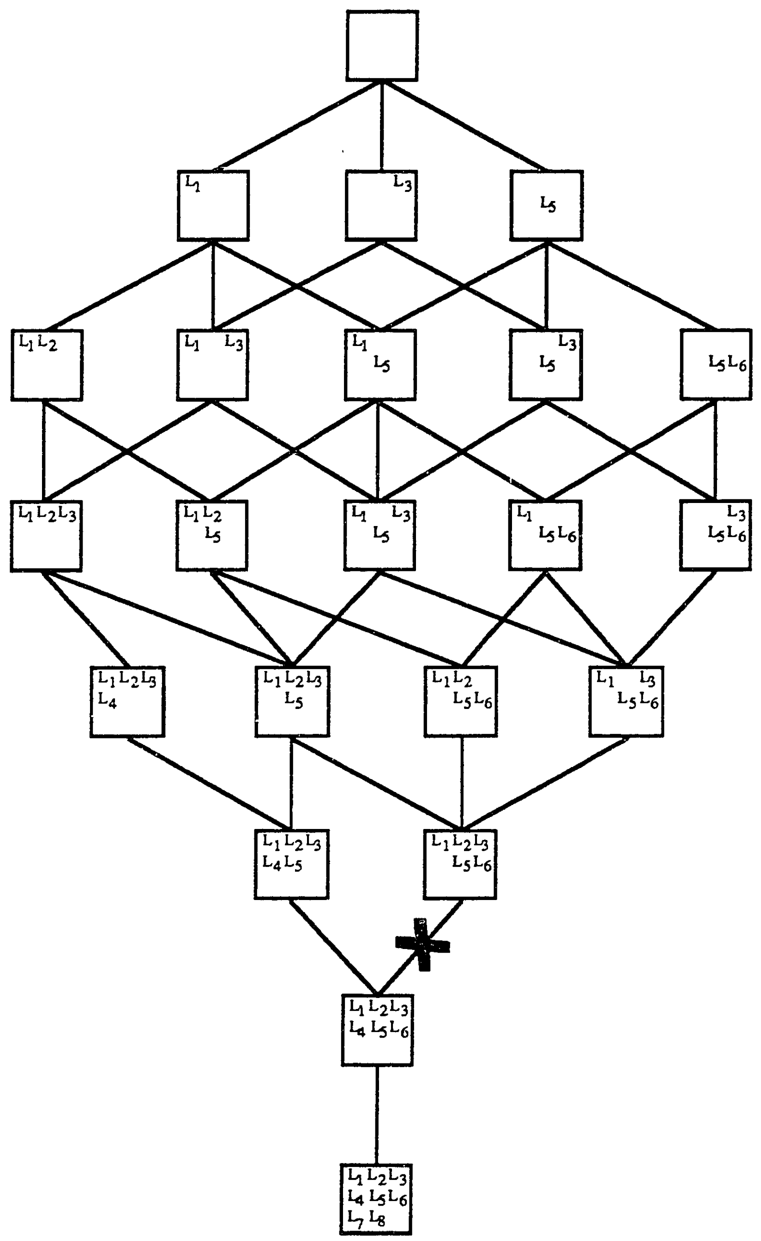


Figure 4.15c: Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.15a before any of the "redundant" arborescent assembly sequences have been deleted. For the sake of simplicity, it assumed that the part grouping shown in Figure 4.15b is the only possible parts grouping for this product.

In reality, there are probably other possible part groupings. Consequently, during the arborescent purge we delete state transitions with parent states - not the state themselves - where all the liasions of the smallest subassembly are done.

In the above figure, the state in the second to last figure is made up of two subassemblies, one consisting of five parts, the other of three. Consequently, we delete the state transtion with the parent state where the smallest subassembly is fully assembled to insure that the smallest subassembly is done last.

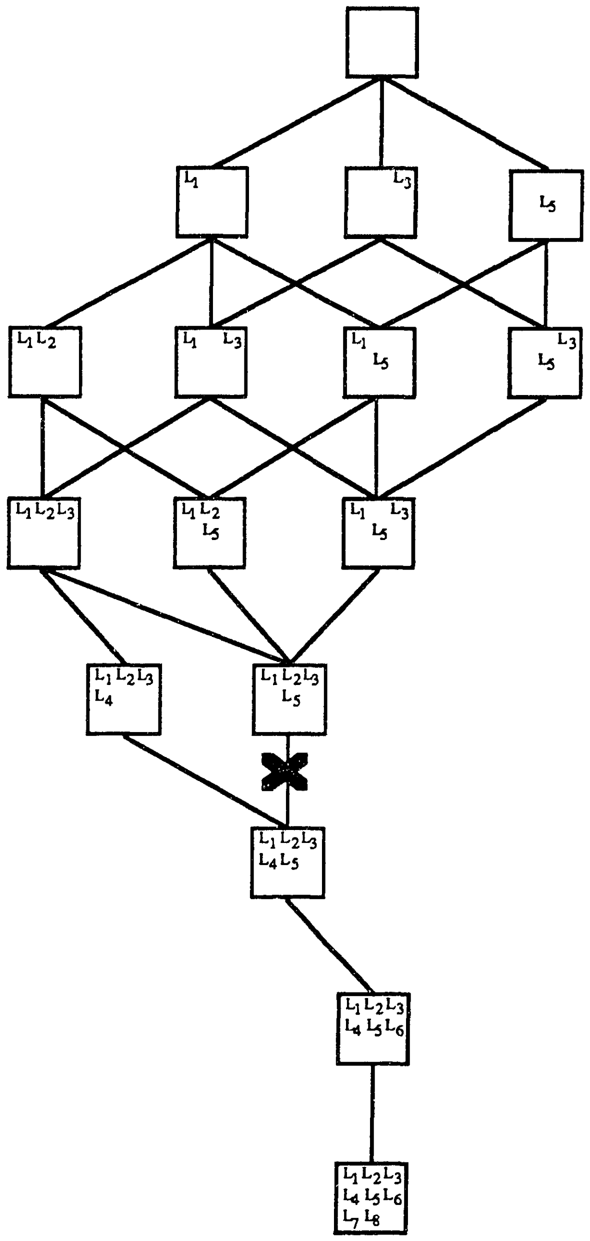


Figure 4.15d: Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.15a after the Arborescent Purge Algorithm has been applied to the last two ranks.

In the above figure, the state in the third to last rank is made up of two subassemblies, one consisting of five parts, the other of two. Consequently, we delete the state transtion with the parent state where the smallest subassembly is fully assembled to insure that the smallest subassembly is done last.

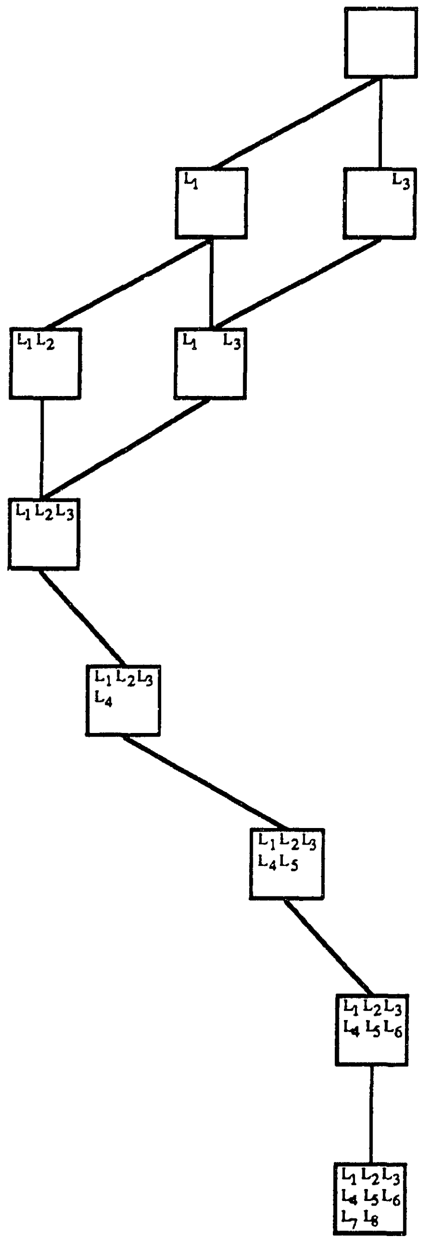


Figure 4.15e:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.15a after the Arborescent Purge Algorithm has been applied to the last three ranks.

In the above figure, the state in the fourth rank is made up of only one subassembly, consequently this state is not even examined

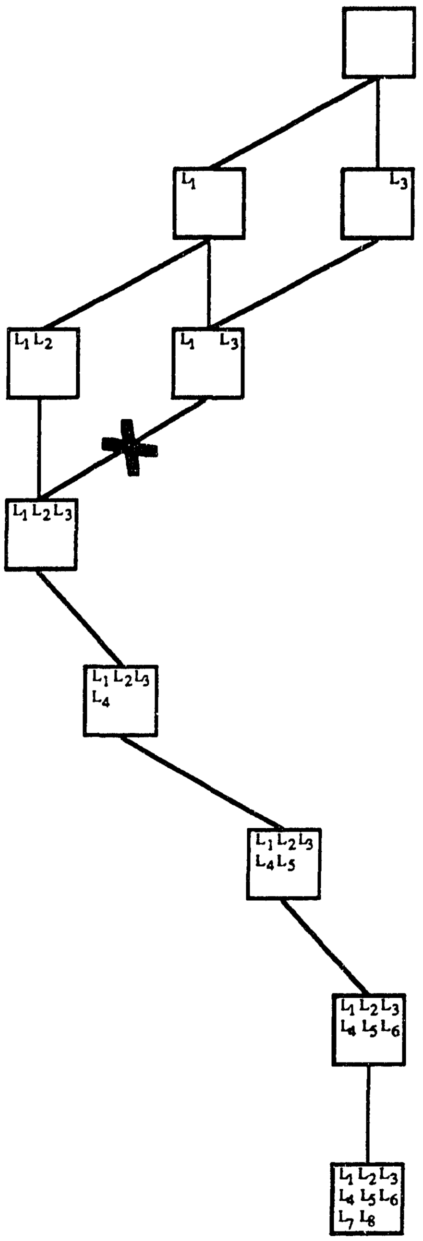


Figure 4.15f:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.15a after the Arborescent Purge Algorithm has been applied to the last four ranks.

In the above figure, the state in the third rank is made up of two subassemblies, one consisting of three parts, the other of two. Consequently, we delete the state transtion with the parent state where the smallest subassembly is fully assembled to insure that the smallest subassembly is done last.

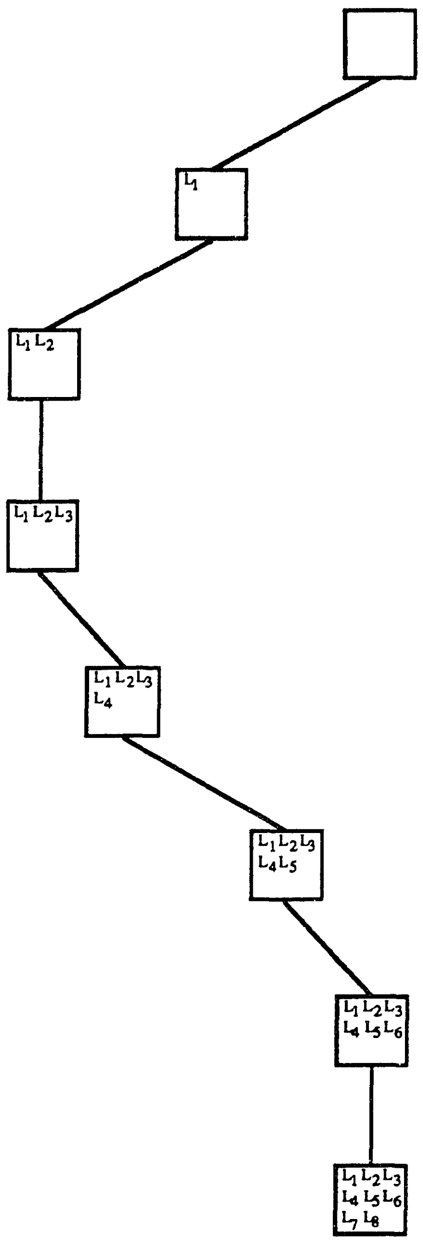


Figure 4.15g:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.15a after all the "redundant" arborescent assembly sequences have been deleted.

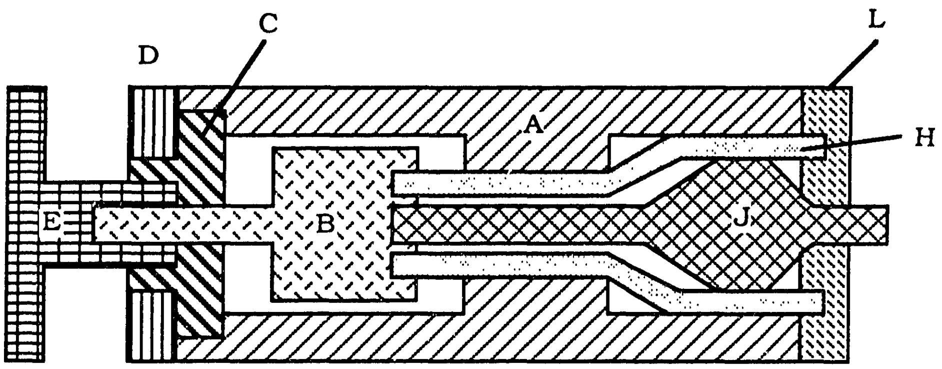


Figure 4.16a:

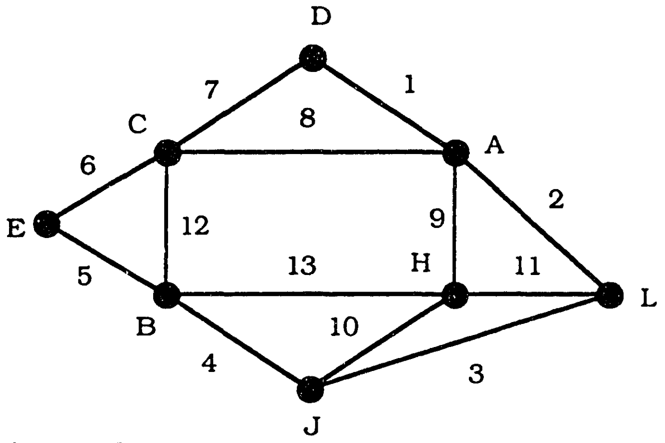


Figure 4.16a,b: Schematic and liaison diagram of a simplified version of the example of Assembly From Industry (A.F.I.) of De Fazio and Whitney (1986).

Source: Baldwin, 1989

4.4) The Arborescence Maximization Module

Once all the "redundant" sequential sequences of the arborescent assemblies of a product have been eliminated from the liaison state diagram, it is reasonably easy to determine the most arborescent assembly sequence.

As shown in the flowchart of Figure 4.14a, the Arborescence Maximization Module is divided into two parts:

- The ARBcount Maximization Submodule

1 >= 2 & 3 & 4 & 5
1 >= 4 & 7 & 12
1 >= 5 & 9 & 13
1 >= 7 & 12 & 13
1 >= 6 & 8
2 >= 3 & 4
2 >= 11 & 13
2 >= 4 & 8 & 11 & 12
3 >= 2
3 >= 11
4 >= 1 & 2 & 3
4 >= 2 & 3 & 8
4 >= 1 & 9 & 10
4 >= 8 & 9 & 10
4 >= 11 & 13
4 >= 2 & 3 & 13
4 >= 2 & 8 & 12
5 >= 1 & 2 & 3 & 4 & 6
5 >= 6 & 8
5 >= 1 & 9 & 13
6 >= 5
6 >= 1 & 7 & 8 & 9 & 12& 13
7 >= 1
7 >= 6
8 >= 1
8 >= 2 & 3 & 4 & 5
8 >= 4 & 12
8 >= 5 & 9 & 13
8>= 12 & 13
9 >= 2
9 >= 4 & 8 & 12
9 >= 13
10 >= 11
10 >= 2 & 3
10 >= 4
11 >= 2
11 >= 3 & 4
12 >= 5
12 >= 1 & 2 & 3 & 4
12 >= 8
12 >= 1 & 9 & 13
13 >= 4
13 >= 1 & 9
13 >= 8 & 9
13 >= 2 & 8 & 12

Figure 4.16c: Precedence relations for the modified version of example of Assembly From Industry (A.F.I.) described in the schematic of Figure 4.16a and in the liaison diagram of Figure 4.16b

• The Final Arborescence Maximization Submodule (FAMS)

The ARBcount Maximization Submodule eliminates all the assembly sequences which do not have the maximum possible ARBcount. The FAMS distinguishes between the level of arborescence of the assembly systems with the same maximal ARBcount and leaves us with the most arborescent sequences.

4.4.1) The ARBcount Maximization Submodule

We start by initializing the ARBcount of all the states in rank 1 to zero We then apply the following steps to every state STr from rank 2 to rank n:

• 1. We identify STa, the parent state of STr with the highest ARBcount.

• 2. Set *ARBcount(STr) to (ARBcount(STa) + #Sub(STr) -1)*, where #Sub is the number of subassemblies.

• 3. Before moving to the next state, we delete the transitions from parent states which have a lower ARBcount than STa, since these states represent assembly sequences leading to STr, which are less arborescent than the sequence going through STr.

We now explain why the succinct formula in step two calculates the correct ARBcount:

• *If STr has only one subassembly*, (=> #Sub(STr) -1 = 0), we set STr's ARBcount equal to STa's ARBcount, because the operation corresponding to the transition from STa to STr could not have possibly increased the number of operations performed "away from" the main assembly.

• *If STr has two subassemblies*, (=> #Sub(STr) -1 = 1), then we set STr's ARBcount to be one greater than STa's ARBcount, because the operation, corresponding to the transition from STa to STr, was necessarily performed away from the main assembly. If the last operation had been done in the main assembly, then Sr could not possibly have had multiple subassemblies, since that would be a violation of our practice of always completing the main assembly before starting any smaller and less arborescent subassemblies.

• *If STr has three subassemblies*, (=> #Sub(STr) -1 = 2), the transition from STa to STr was necessarily done in the third biggest subassembly (for the same reasons as above). Furthermore, the third biggest subassembly is necessarily going to be added to the second biggest subassembly, before being incorporated into the main assembly. For, if the second biggest subassembly was supposed to be incorporated into the main assembly without the third biggest subassembly, this would have been done before even starting the third biggest subassembly, since we always build the biggest subassembly in its entirety before starting on a new subassembly. Thus we set STr's ARBcount to be two greater than STa's ARBcount in order to account for the contribution of the transition from STa to STr to the arborescence of both the second biggest subassembly and the biggest subassembly.

• Generally speaking, *if STr has two or more subassemblies*, we add the (#Sub- 1) to the ARBcount in order to account for the fact that the operation done in the transition from STr to Sta is performed away from the main production branch of the main assembly and away from the main production branches of (#Sub -2) "separate" subassemblies. This operation contributes to the arborescence of the separate subassemblies and by the same token to the overall arborescence of the system. This step corresponds to adding the number of operations left on the "disintegrated tree" to the ARBcount after deleting the main branches of the separate subassemblies, and doing so repeatedly until no operation is left on the "disintegrated tree".

Only because all the separate subassemblies were assembled one at a time, in decreasing order of size in the Arborescent Module does this algorithm compute the correct ARBcount, i.e. the correct number of operations done away from the main assemblies in such a succinct fashion. If the subassemblies had not been represented in the assembly sequences in decreasing order of size, the subassemblies of each sequence would have to be ranked in decreasing size and the ARBcount of each sequence would have had to be computed individually one at a time. This last step would have necessitated examining all the states in all the sequences, (and # of sequences * # of state in each sequences >> total # of states in the liaison diagram), before even starting to delete any liaison state transitions. Most important of all, we would lose the efficiency of the Final Arborescence Maximization

4.4.2) Final Arborescence Maximization Submodule

All the assembly sequences left after the ARBcount Maximization Submodule have the same, maximal ARBcount, i.e. in all these assembly systems the same number of operations are done away from the main assembly branch and away from the main production branches of the "separate" subassemblies. The Final Arborescence Maximization Submodule therefore needs to check at what point the "separate" subassemblies are incorporated into the main assembly branch, in order to distinguish between various level of arborescence within this class of systems. For, as mentioned in Section 4.3, the later the "separate" subassemblies are incorporated into the main assembly branch, the more decoupled are the operations in the main production branch from the operations done away from the main subassembly, and consequently the more arborescent the system.

We are comparing assembly sequences with the same overall ARBcount in this section. Consequently if one sequence has a lower ARBcount at an earlier rank, then the ARBcount in that sequence is incremented at a later point time. The operations done away from the main subassembly must therefore be incorporated into the main assembly at a later point in time, since subassemblies are started and completed in decreasing order of size and *arborescence*. Thus, a system with a lower ARBcount at an earlier rank is more arborescent. The ARBcount gives a measure of both the width and the height of a tree, when one considers how it grows over an assembly sequence.

The above conclusions enable us to distinguish between sequences with the same ARBcount but with different levels of arborescence. We:

• 1. start at rank 2 of the liaison state diagram

• 2. compare the ARBcount of each state in the rank and identify ARBstate, the state in the rank with the lowest ARBcount

• 3. and, before moving to the next rank , delete any *state* in the rank whose ARBcount is not as small as ARBstate's, since these states represent assembly sequences which are necessarily less arborescent than the sequences going through ARBstate.

4.4.3) Arborescence Minimization

A manufacturing engineer, who wants to use a sequential assembly sequence but can't because of constraining liaisons precedence relations, may wish to find out the most sequential or least arborescent assembly sequence possible for the assembly of a given product.

With just a few simple modifications, the Arborescence Maximization method outlined in sections 4.4.1 and 4.4.2 can be transformed into the Arborescence Minimization Method, which determines the least arborescent sequences. We have italicized the necessary changes below.

ARBcount *Minimization*
• 1. We identify *STs*, the parent state of STr with the *lowest* ARBcount.
• 2. ARBcount(STr) = ARBcount(STs) + #Sub(STr) -1, where #Sub is the number of subassemblies.
• 3. Finally, before moving to the next state, we delete the transitions from parent states wh[illegible] [illegible]ave a *higher* ARBcount than *STs*, since these states represent assemb[illegible] sequences leading to STr, which are *more* arborescent than the sequence going through STr.

Final Arborescence *Minimization*
• 1. starting at rank 2 of the liaison state diagram
• 2. comparing the ARBcount of each state in the rank and identify ARBstate, the state in the rank with the *highest* ARBcount
• 3. and, before moving to the next rank, deleting any state in the rank whose ARBcount is not as *high* as ARBstate's, since these states represent assembly sequences which are necessarily *more* arborescent than the sequences going through ARBstate.

4.4.4) Examples

In Figures 4.17c thru 4.17e we have represented the liaison sequence diagram for a generic product with the liaison diagram shown in Figure 4.17a, at different stages of the Arborescence Maximization.

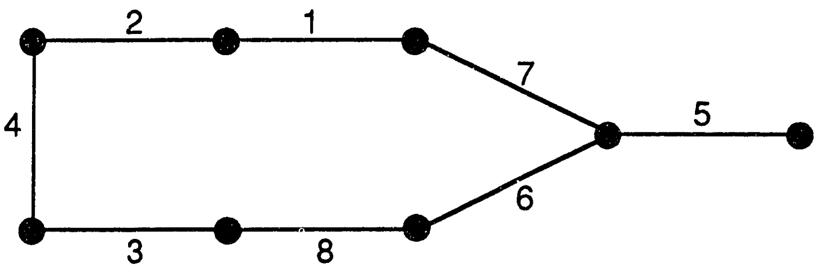


Figure 4.17a: Liaison diagram for the generic product. For the sake of simplicity, it is assumed that the six part groupings shown in Figure 4.17b are the only possible parts groupings for this product.

It is now interesting to reconsider the eight-part modified version of the example of Assembly From Industry (A.F.I.) introduced in Section 4.2.4 and shown in Figure 4.16, to study the impact of our various modules. Whereas the original liaison sequence diagram contained 1008 assembly sequences, only 81 were left after the Main Purge, 71 after the Detailed Purge and 55 after the ARBcount Maximization. Finally, 3 assembly sequences remained after the Final Arborescence Maximization

4.5) CONCLUSION

In this chapter, we have presented a method to determine the most arborescent and most sequential sequences possible for the assembly of a product, based on the algorithmic liaison sequence generation method of De Fazio and Whitney (1986). The algorithm consists of two parts: the deletion of redundant arborescent sequences generated by the Draper Laboratory's LSA software and the deletion of assembly sequences which did not have the maximum level of arborescence. It is based on an "arborescent classification," generated by calculating the ARBcount, a measure of both the height and the width of a parts-tree.

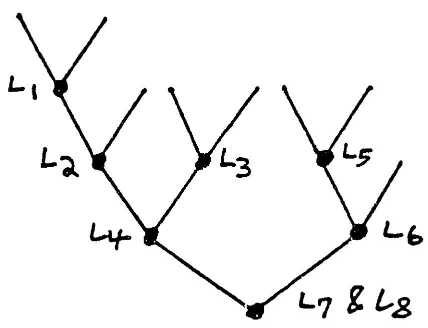
The deletion of redundant arborescent sequences will facilitate the choice of assembly sequence using the LSA software. For products with an *average number of potential subassemblies*, the number of assembly sequences will be reduced by a factor of 10 for products with about 8 parts and by a factor of 10^5 for products with about 16 parts.

REFERENCES MADE IN CHAPTER IV

[1] De Fazio, T.L. and D.E. Whitney. Simplified Generation of All Mechanical Assembly Sequences, C.S. Draper Laboratory Report No 2709, Cambridge, Ma, June 1988

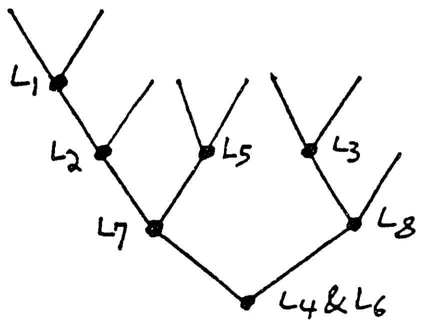
[2] Lui, Man-Cheung Max. . *Generation and Evaluation of Mechanical Assembly Sequences using the Liaison-Sequence Method*, M.I.T. S.M. Thesis, M.E. Department. C.S. Draper Laboratory Report No 990, Cambridge, Ma, May 1988.

[3] Klein, Charles C. *Generation and Evaluation of Mechanical Assembly Sequence Alternatives*, M.I.T. S.M. Thesis, M.E. Department C.S. Draper Laboratory Report No 934, Cambridge, Ma, May 1988



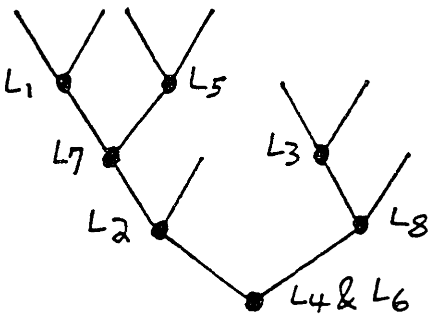
ARBcount = 3

Figure 4.17b.1:



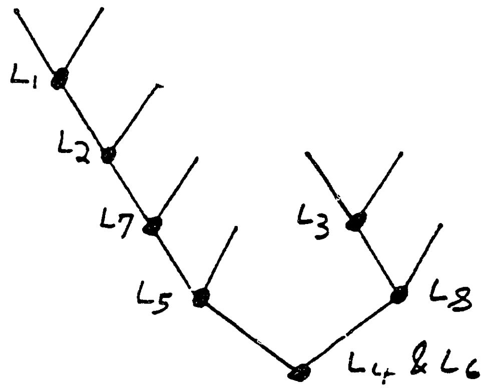
ARBcount = 3

Figure 4.17b.2:



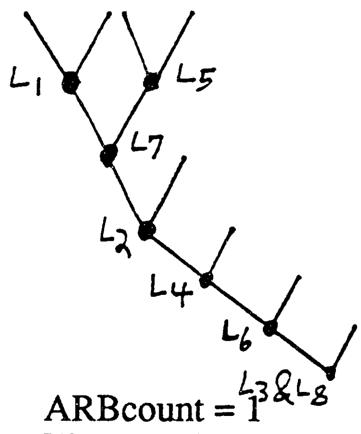
ARBcount = 3

Figure 4.17b.3:



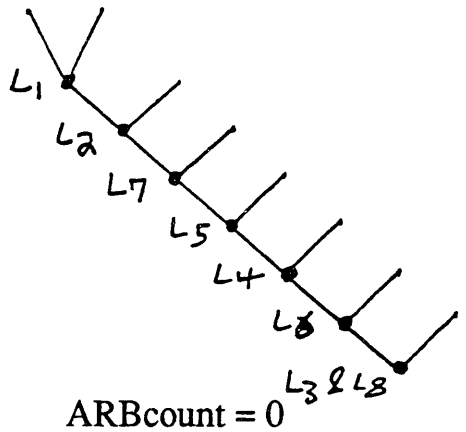
ARBcount = 2

Figure 4.17b.4:



ARBcount = 1

Figure 4.17b.5:



ARBcount = 0

Figure 4.17b.6:

Figure 4.17b: Parts-tree diagram representation of the six possible parts groupings for the generic product shown in Figure 4.17a

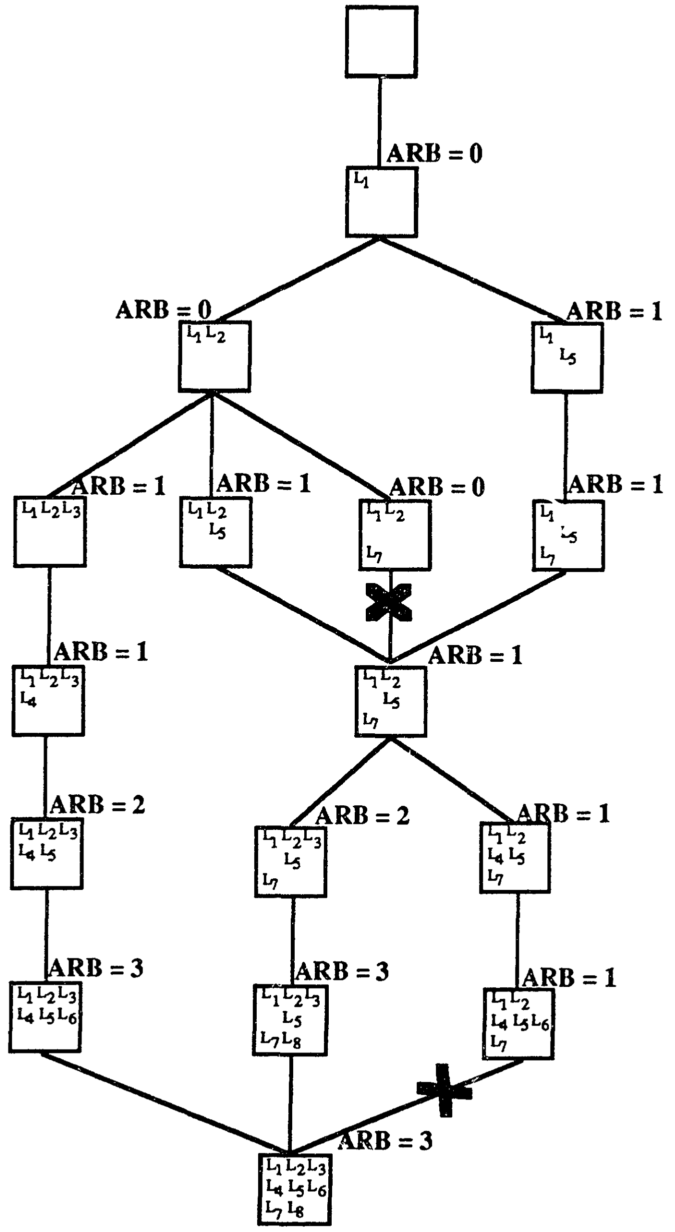


Figure 4.17c:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.17a after all the "redundant" arborescent assembly sequences have been deleted. For the sake of simplicity, it is assumed that the six part groupings shown in Figure 4.17b are the only possible parts groupings for this product.

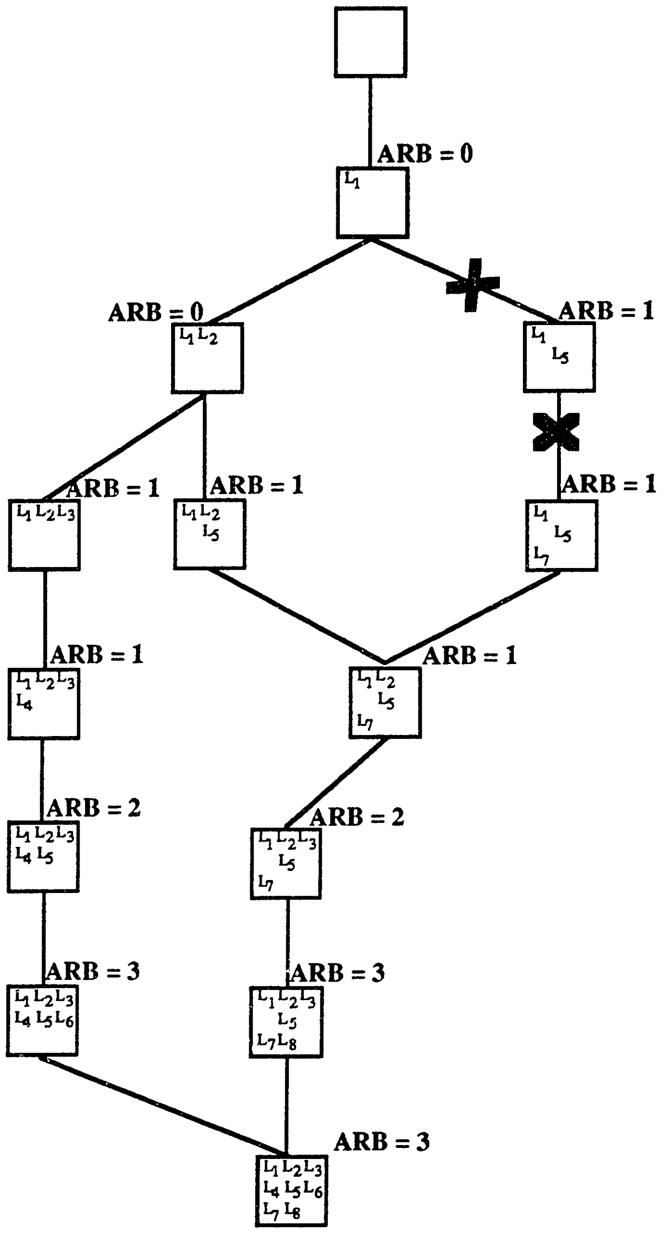


Figure 4.17d:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.17a after all the assembly sequences which do not have the maximum, possible ARBcount have been deleted.

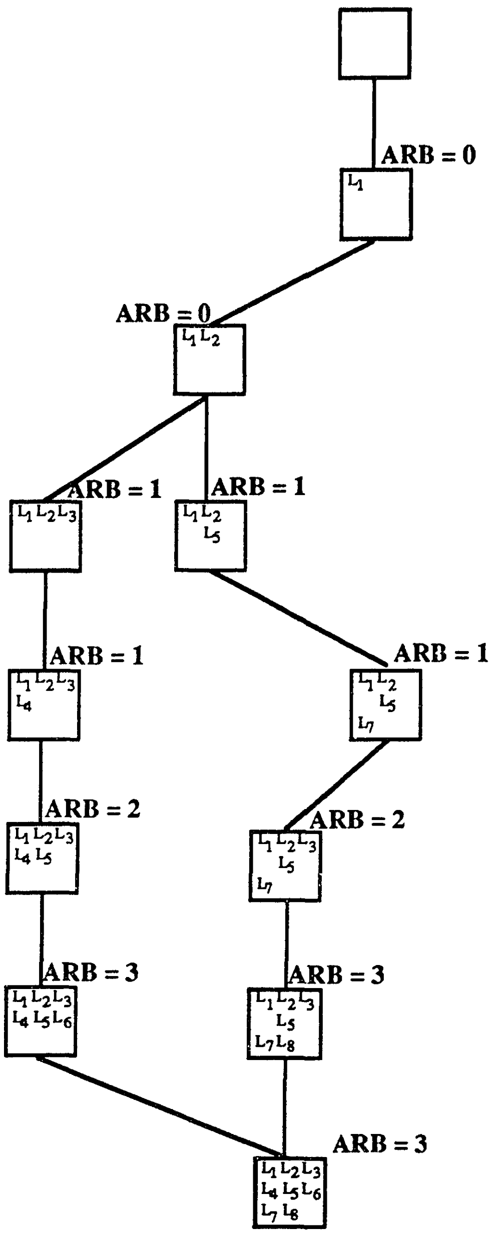


Figure 4.17e:Liaison sequence diagram for the generic product whose liaison diagram is shown in Figure 4.17a after all the assembly sequences which do not have the maximum possible level of arborescence have been deleted. The two assembly sequences left correspond to the part groupings represented in Figures 4.17b1 and 4.17b2.

CHAPTER V
RECOMMENDATIONS FOR FUTURE WORK

This thesis was meant to mark the beginning of a comprehensive study of qualitative and quantitative rationale for the use of subassemblies. Consequently many extensions of this thesis are possible.

5.1 FURTHER COMPARISON OF SEQUENTIAL AND ARBORESCENT SYSTEMS

5.1.1. Comparison of the Output Rate of Systems with Batch Part Transfers

As mentioned at the end of chapter III, we could further our comparison of the sequential and arborescent systems by studying the output rate of infinite buffer systems, where parts are transferred in batch between stations. In these types of systems, while no station would ever be blocked, the phenomenon of starving would remain and in fact would be accentuated. Intuitively, this transfer mechanism would seem likely to accentuate the "lateral advantage" of arborescent systems, while limiting the "longitudinal advantage" of sequential systems.

5.1.2. Study of the impact of delayed testing

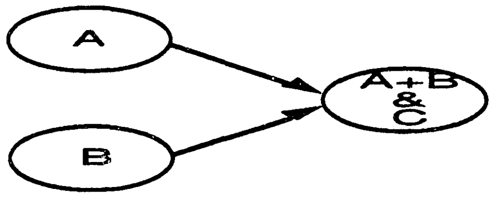
We could extend our comparison of sequential and arborescent systems to the area of testing. The goal would be to capture what is arguably the essence of the "subassembly edge" in a single-product environment, i.e. early and frequent testing. It would be interesting to model the impact of the trade-off between early and delayed testing, specifically to quantify the impact of this trade-off on scrap, diagnosis and repair costs.

5.1.3 Conclusion to the comparison of sequential and arborescent systems

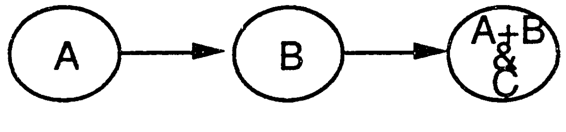
The comparison of arborescent and sequential systems broadens our understanding of the two types of configurations, which display the most extreme use of subassemblies. We now know that for all practical purposes, reliable arborescent and sequential system have identical output rate. Certainly, their relative output rate should not be a decisive factor in the decision to use subassemblies.

The impact of the comparison of AS and SS, though, is restricted by the difficulty of establishing one to one correspondence between operations in the two systems. Because of this difficulty, as well as for simplicity and ease of analysis, we had to initially make the assumption of stations with identical quantitative characteristics. Any meaningful conclusion therefore is bound to be quite limited.

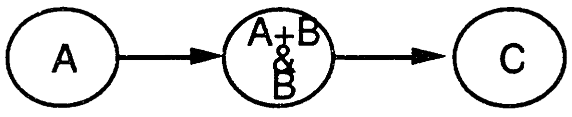
The fundamental problem is that a modification in system configuration changes the functionality of the individual operations. For instance in a three station system, the sequential equivalent of



could be



as well as



, where A+B corresponds to the mating of parts of A and B, and where C consist of an operation performed on the product of A+B. The only real way to overcome these problems is to work with real data.

5.2 APPLICATION OF O.R. TECHNIQUES TO PRODUCT CASE STUDIES

While the modelling of the use of subassemblies is restricted in an academic setting, where one is forced to make idealized assumptions, quantitative techniques could still be instrumental in our study of rationale for subassemblies.

A formal case study of two or three products in different industries, for which actual data is available should be undertaken. Its goal would be to point out and quantify the savings and costs implication of subassembly use in all aspects of product and process design: production and inventory, capital investment, design time, testing, diagnosis, repair and scrap costs, etc.. Ideally this work would be conducted by an operation researcher, with a mechanical engineering background, in joint collaboration with local manufacturers.

The first step in such a project would involve the collection of quantitative tools and techniques available. Much work has been done in relevant areas, in particular in determining optimal:

- testing strategies
- standardization of subassemblies and modules, as discussed in the modular design, standardization and cannibalization literature
- grouping of parts with similar features, as discussed in the group technology literature

References to relevant literature in those research areas are included in the bibliography.

Judicious use of these quantitative tools would yield a better appreciation for the pros and cons of use of subassemblies, especially since no comprehensive study of rationale for subassemblies have never been conducted.

In addition, this exercise of studying product design from the eye of a "subassembly decision maker" would help us to develop a methodological approach to the choice of subassemblies.

5.3 DEVELOPMENT OF RULE-BASED EXPERT SYSTEM FOR USE OF SUBASSEMBLIES

The ultimate goal of the product case study project could be to develop a Rule-Based expert-system for Choice Of Subassembly (RBCOS), that could be integrated in the sets of computer-tools of the Strategic Approach to Product Design.

5.3.1 Questions and rules

RBCOS would be based on the rules emanating from the product cases studies and the list of pros and cons list of Chapter II. Some of the relevant questions to be asked when considering a subassembly would be:

- How many parts does this subassembly have? Is the number of parts limited to 12 or 13 as suggested by Riley (1983)?
- Does this subassembly have a well defined function(s) that can be tested?
- How quickly can the various functions of the subassemblies be tested? How difficult and costly are the diagnosis and the repair of the subassembly?
- Is the subassembly stable? Conditionally stable?
- How easily is the subassembly grabbed, handled or oriented?

- How sÿmmetric or asymmetric is the subassembly? The most extreme, the symmetry or asymmetry the better.
- How experimental or problem-prone is the subassembly? How likely is it to slow-down the system?
- How variable is the production volume of this subassembly? Do "redundant" machines need to be added at certain times or seasons?
- Is this subassembly similar to ones used in previous designs or in other products?
- Can the subassembly be standardized to be used in other products?
- Is the interface of this subassembly with the rest of the system well defined?

5.3.2 Linkage to the LSA software

RBCOS should be linked to the LSA software. In considering a subassembly, one should be able to study the impact of the use of that subassembly on the rest of the assembly. Consequently, the LSA should have a built-in module, that erased all the assembly sequences, where the parts, specified by a user, are not built as a subassembly. When one has decided on using a certain subassembly, the sequences, that don't include this subassembly, become irrelevant and should thus be eliminated.

5.3.3 Linkage to a Group-Technology based database

RBCOS should also be linked to a Group-Technology type database that includes:

- a classification of parts used in the industry
- a classification of proprietary parts
- a classification of standard subassemblies, used in previous designs
- a list of the precedence relations and liaison diagrams for previously considered assembly and subassembly sequences

Products are rarely produced from scratch; they are usually modifications and synthesis of previous designs. A designer should be able to reap the full benefits of those previous designs.

He should be able to make an educated selection of parts, study the impact of that choice on the assembly sequence and modify his selection of parts repeatedly, until the final product and the final assembly sequence have been chosen. Concurrently, he should be able to consider a wide range of subassemblies and study their impact on the assembly sequence.

The stored information on parts would insure that no part is overlooked in the part selection process. Similarly, the stored information on previously used assemblies and subassemblies would enable the designer to quickly prune the set of potentially optimal assembly sequences. Finally, the stored information on previously considered assembly sequences and subassemblies would limit the number of questions that the designer must answer to generate all the assembly sequences possible.

This integrated computer-aided approach would result in quicker, better quality product design. It would certainly fit nicely within the framework of the Draper Laboratory's Strategic Approach to Product Design.

CHAPTER VI

CONCLUSION

This thesis is meant to mark the beginning of an extensive study on rationale for the use of subassemblies in assembly and production systems.

We first presented a study of qualitative advantages and disadvantages of subassembly use. This survey was based on a broad literature search, which covered the conventional manufacturing and assembly literature as well as areas, which exhibit subassembly-like-entities. Since no comprehensive study of rationale for subassemblies had been conducted, we listed the advantages of subassembly-like entities, which are applicable to subassemblies in the manufacturing and assembly sense of the word, as well extrapolations of comments on the subassemblies in the sparse manufacturing and assembly literature.

In this initial work, we felt the need to distinguish between two types of subassemblies: subassemblies in a single-product manufacturing environment and subassemblies in a multi-product manufacturing environment.

In a multi-product environment, the quantitative advantages of subassembly use are clear. Commonality, modularity, cannibalization and standardization contribute to substantial savings in inventory and production costs, as well as decrease in design time. While there are definite qualitative reasons for use of subassemblies in a single-product environment, it is less clear that there are substantial quantitative advantages to do so.

In an effort to quantify the impact of using subassemblies in a single-product environment, we compared the output rate of the two types of configurations, which display the most extreme use of subassemblies: the arborescent and the sequential configurations. Specifically, we simulated non-synchronous arborescent and sequential systems for two different models of the systems' stations. One model assumed stations with identical quantitative characteristics, in particular independent and identically distributed processing times. The other assumed bufferless, balanced systems with two types of stations: stations with deterministic processing times and stations with exponentially distributed processing times.

We concluded that for all practical purposes sequential and arborescent systems with stations, with independent and identically distributed processing times have equivalent output rates. Any relative difference between the two types of configurations is attenuated by buffer space and is inversely proportional to the coefficient of variation. Furthermore, while r(AS) and r(SS), the output rates of AS and SS can vary considerably for different types of processing time distributions with same COV, - for instance, r(AS) and r(SS) are considerably higher for bernoulli distribution than for lognormal distributions - any difference between the two does not vary considerably as a function of the type of distribution.

The simulation of the second type of models did not really alter our conclusions. AS and SS do not differ greatly in their ability to attenuate the processing time variability of isolated stations. While one configuration or another may be preferable depending on the location and the number of sources of variabilities, no structure is clearly always advantageous. We were able to distinguish a "lateral advantage" of the arborescent systems, when the bottlenecks (i.e. more variable stations) were in the same stage of production and a "longitudinal advantage" of sequential systems, when the bottlenecks were in the same branches of production. These so-called "lateral " and "longitudinal" advantages do not reflect a fundamental difference between the two types of structures, but rather they reflect a difference in the "separation" of the sources of variability in the two systems.

Certainly, any decision to use subassemblies in an assembly or production system should not be based on the relative output rates of non-synchronous arborescent and sequential systems. That is not to say, however, that arborescent systems and sequential systems are equivalent. Further study, in such areas as testing and commonality, should be pursued.

For case studies of products from industry, it would be most useful to know the most arborescent sequence possible for a product. In Chapter IV, we presented a method to determine the most arborescent and most sequential assembly sequences possible for a product, based on the algorithmic liaison sequence generation method of De Fazio and Whitney (1988). The algorithm presented consists of two parts: the deletion of redundant arborescent sequences generated by the Draper Laboratory's LSA software and the deletion of assembly sequences which did not have the maximum level of arborescence. It is based on an "arborescent classification," generated by calculating the ARBcount, a measure of the height and the width of a tree. The deletion of redundant arborescent sequences will facilitate the choice of assembly sequence using the LSA software. For products with an average number of potential subassemblies, the number of assembly sequences will be reduced by a factor of 10 for products with about 8 parts and by a factor of 10^5 for products with about 16 parts.

This thesis has taken a first step in the study of the rationale for the use of subassemblies in production systems. As mentioned in Chapter V, several possible quantitative extensions are possible. The most insightful continuation of this thesis, though would consist of a formal case study of two or three different products, for which actual data are available. The initial goal would be to make judicious use of O.R. techniques to point out and quantify the savings and cost implications of subassembly use in all aspects of product and process design, as well as to develop a methodological approach to the choice of subassemblies. Such a project, while logistically complex, would be useful, as it would culminate in a Rule-Based expert-system for the Choice Of Subassemblies (RCOS), that could be integrated in the sets of computer-tools for the Strategic Approach to Product Design.

BIBLIOGRAPHY

Assembly Automation, Design for Assembly and Product Design

Andreasen, M.M., S. Kahler, and T. Lund, 1982 "Design for assembly - an integrated approach," *Assembly Automation,* August

Andreasen, M.M., S. Kahler, and T. Lund, 1983 Design for Assembly, IFS Publications Ltd. Bedford, U.K.

Boothroyd G and P. Dewhurst, 1986. "Production Design... Key to Successful Robotic Assembly," *Assembly Engineering,* September and October.

Boothroyd G., C. Poli and L.E. Murch, 1982. Automatic Assembly, N.Y., N.Y.: Marcel Decker.

Boothroyd G., A.H. Redford, 1968. Mechanized Assembly, London: McGraw Hill.

Bolz, Roger W, 1977. Production Processes: The Productivity Handbook. Fifth Edition. N.Y., N.Y.: Industrial Press.

CIRP (15th) Manufacturing Systems seminar. Conference Report, 1983: "Easy assembly and feeder design feature at seminar," *Assembly Automation,* August.

Dewhurst, P. and G. Boothroyd, 1987 "Design for Assembly in Action," *Assembly Engineering*, January.

Hartley, John., 1984."Systematic approach to ease assembly," *Assembly Automation,* August.

Institution of Production Engineers, 1984. Seminar on " Automated Assembling Systems Choice - Dedicated vs. Flexible," London, February

Institution of Production Engineers. 1985. Conference report: "Opportunities for rationalising assembly," *Assembly Automation*, August.

Jenkins, R.F., 1965. "Why Automatic Assembly?" *Metal Production*, August 18.

Johnson, R.A., W.T. Newell, Roger C. Vergin. Production and Operations Management: A systems' concept.

Kahler, S. and T. Ahm, 1984. "Design for assembly - a case study," *Assembly Automation*, August.

Lotter, Bruno, 1984. "Using the ABC analysis in design for assembly," *Assembly Automation*, May.

Lund, Thomas, 1984."Integration is the theme for design," *Assembly Automation,* August.

Mangin, Charles-Henri, 1988. "Design for Producibility," *Assembly Engineering,* February .

Nevins, J.L., D.E. Whitney, 1986 "What progressive Companies are doing to raise productivity." C.S. Draper Laboratory Report No 2729, Cambridge, Ma.

Owen, Tony, 1985. Assembly with Robots, Englewood Cliff, NJ: Prentice Hall.

Owen, Tony, 1984. Flexible Assembly Systems: Assembly by Robots and Computerized Integrated Systems. N.Y.,N.Y: Plenum Press.

"Proceedings of the 1st International Conference on Assembly Automation, 1980." Brighton, UK: IFS Ltd, March 25-27.

Rathmill, Keith. Robotic Assembly. UK: IFS Ltd, Springer Verlag, 1985.

Redford, A.H, 1984. "Product design for general purpose assembly," *Assembly Automation,* August.

Riley, Frank J, 1983. Assembly Automation: A Management Handbook, N.Y., N.Y.: Industrial Press.

Schroder, Manfred and Robert Bosch, 1982. "Meeting a variety of future needs with flexible assembly," *Assembly Automation*, February.

Starr, M.K, 1963. Product Design and Decision Theory, Englewood Cliffs, NJ: Prentice Hall.

Streer, Kenneth, 1979. Automated Assembly, Dearborn, Michigan: Society of Manufacturing Engineers.

Talje, E, 1979. A Short Course in Industrial Design, London: Neunes-Butterworth.

Waterbury, Robert C, 1987. "Process Planning Perfects Production," *Assembly Engineering*, November.

Whitney, D.E., et al., 1986. *The Strategic Approach to Product Design.* The C. S. Draper Laboratory Report 2742, Cambridge, Ma.

Assembly Sequence Generation

Baldwin, D.F. *Software Implementation of A Modified Version of Bourjault's Assembly Sequence Generation Method,* MAT Memo 1332, C.S. Draper Laboratory, Cambridge, Ma.

Bourjault, A, 1984. *Contribution à une Approche Méthodologique de l'Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires*, Ph.D. Thesis to obtain Grade des Séquences Opératoires," Ph.D. Thesis de L' Université de Franche-Comté en Sciences Physiques .

De Fazio, T.L. and D.E. Whitney, 1988. *Simplified Generation of All Mechanical Assembly Sequences*, C.S. Draper Laboratory Report No 2709, Cambridge, Ma.

Homem de Mello, L.S. and A.C. Sanderson, 1986. *And/Or Graph Representation of Assembly Plans,* Carnegie Mellon.University, The Robotics Institute, Pittsburgh, Pennsylvania Report No. CMU-RI-TR-86-8, April.

Homem de Mello, L.S., A.C. Sanderson, 1988. *Automatic Generation of Mechanical Sequences,* The Robotics Institute, Carnegie Mellon Institute, Pittsburgh, Pa.

Hoummady, Abdellah. Conception et Développement d'un logiciel d'Elaboration Automatique des Gammes d'Assemblage. Ph.D. Thesis Université de Franche-Comté en Automatique et en Informatique

Lui, Man-Cheung Max, 1988. *Generation and Evaluation of Mechanical Assembly Sequences using the Liaison-Sequence Method*, M.I.T. S.M. Thesis, M.E. Department. C.S. Draper Laboratory Report No 990, Cambridge, Ma.

Klein, Charles C.1988. *Generation and Evaluation of Mechanical Assembly Sequences* Alternatives, M.I.T. S.M. Thesis, M.E. Department C.S. Draper Laboratory Report No 934, Cambridge, Ma.

Group Technology and part classification

De Fazio, T.L., D.E. Whitney, 1983. "Part and Assembly Technique Classification," C.S. Draper Laboratory Report No 1643, Cambridge, Ma.

Gallagher, C.C, W.A. Knight, 1986. Group Technology Production Methods in Manafacture John Wiley &Sons, New York.

Hyde, W.F., 1981. Improving Productivity by CLASSIFICATION, CODING, AND DATA BASE STANDARDIZATION, Marcel Dekker, Inc., New York.

Kondoleon, A.S., 1976. *Application of Technology - Economic Model of Assembly Techniques to Programmable Assembly Machine Configuration,* S.M. Thesis, MIT Mechanical Engineering Department.

Opitz, H., 1967. A Classification System To Describe Workpieces. Pergamon Press, Oxford.

Reliability, Inspection, Diagnosis, and Testing

Lloyd, D.K., M. Lipow. 1984. Reliability: Management, Methods and Mathematics. The American Society for Quality Control, Milwaukee, Wisconsin.

Kennedy, C.W., E.G. Hoffman, S.D. Bond. Inspection and Gaging. Industrial Press, Inc,. New York, N.Y.

Pappu, Suguna, 1989. *A Dual-Ascent Algorithm for Finding an Assembly Test Strategy*, M.I.T. S.M. Thesis, Operations Research Center, C.S. Draper Laboratory Report (in preparation), Cambridge, Ma.

Pau, L.F., 1975. Diagnosis and Performance Monitoring. Marcel Dekker, Inc. New York and Basel.

Raz, Tzvi, 1986. "A Survey of Models for Allocating Inspection Effort in Mulitstage Production Systems," *Journal of Quality Technology*, Vol.18, No.4, October.

Modularity, Commonality, Cannibalization and Standardization

Bongers, C.1980. Standardization: Mathematical Methods in Assortment Determination. Martinus Nijhoff Publishing, The Hague.

Charnes, A., M. Kirby, 1965. "Modular Design, Generalized Inverses and Convex Programming," *Operations Research*, Vol 13, pp. 836-847.

Evans, D.H., 1963 "Modular Design- Special Case in Non-Linear Programming," *Operations Research,* Vol 11, pp. 637-647.

Passy, U., 1970. "Modular Design- Special Case in Non-Linear Programming," *Operations Research,* Vol 18, pp.441-453.

Hirsch, W.M., M. Meisner, C. Boll, 1968. "Cannibalization in Multi-Component Systems and the Theory of Reliability," Naval Res. Log. Quart. 15, pp. 331-359.

Rutenberg, D.P., 1971 "Design Commonality to Reduce Multi-Item Inventory: Optimal Depth of a Product Line," *Operations Research*, Vol. 19, March-April, pp. 491-509.

Rutenberg, D.P, T.L.Shaftel, 1971 "Product Design: Subassemblies for Multiple Markets," *Management Science*, Vol. 18, No 4, Part I, December, pp. 220-230.

Shaftel, Tim, 1971. "An Integer Approach to Modular Design," *Operations Research*, Vol. 19, January-February, pp. 130-134.

Shaftel, Tim, 1972. "How Modular Design Reduces Production Costs," *Arizona Review*, Vol. 21, No 6-7, June-July, p.4.

Starr, M.K., "Modular Production - A New Concept," *Harvard Business Review,* November-December, pp. 131-142.

General Operations Management

Candea, D. I. , 1977. *Issues of Hierarchical Planning in Muli-Stage Production Systems*, PHd Thesis M.I.T. Sloan School of Mgt.

Groover, Mikell P. 1987. Automation, Production Systems, and Computer Integrated Manufacturing. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Prenting, T.O., R. M. Battagin, 1964. The Precedence Diagram: A Tool for Analysis in Assembly Line Balancing. *The Journal of Industrial Engineering*, Volume XV, No 4, pp208-213.

Starr, M.K., 1964. Production Management: Systems and Synthesis. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Starr, M.K., 1976. The practice of Management Science. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Tersine, R.J., 1980. Production/ Operations Management: Concepts, Structure and Analysis, School of Businesss, Old Dominion University.

Simulation and Statistics

Bratley P., B.L. Fox, L.E. Schrage, 1983. A Guide to Simulation. Springer-Verlag, New York.

CACI Products Company,1988. SIMGRAPHICS User's Guide and Casebook, La Jolla, Ca

Fishman, G.S., 1978. Principle of Discrete Event Simulation, John Wiley and Sons, Inc., New York.

Larsen, R.J, M.L. Marx, , 1986. An Introduction to Mathematical Statistics and its Applications. Prentice Hall, Englewood Cliffs, New Jersey

Law A.M., W.D. Kelton,1982. Simulation Modeling and Analysis. McGraw-Hill.

Pedgen, C.D., 1984. Introduction to SIMAN, Systems Modeling Corporation, State College, PA.

Russell, Edward C,1983. Building Simulation Models with SIMSCRIPT II.5.CACI, Inc.

Modeling of Stochastic Systems

Ammar, M.H. and S.B. Gershwin. 1987. "Equivalence Relations in Queueing , Models of Assembly/Disassembly Networks," Georgia Institute of Technology School of Information and Computer Science Technical Report GIT-ICS-87/45

Buzacott, J. 1967. "Automatic Transfer Lines with Buffer Stocks." *Int. J. Prod. Res.* 6, 173.

Conway, R., W. Maxwell, J.O. Mclain and L. J. Thomas. 1988. "The role of Work-in-Process Inventory in Serial Production Lines," *Opns. Res*, Vol. 36, No. 2, March-April.

Crane, M.A. 1974. "Multi-Server Assembly Queues," *Journal of Applied Probability*, Vol.11 Number 3, pp 354-367.

Gershwin, S.B. 1988. *Assembly/Disassembly Systems: An Efficient Decomposition Algorithm for Tree-Structured Networks*, Laboratory for Manufacturing and Productivity, M.I.T., Cambridge, MA.

Harrison, J.M. "Assembly-like Queues," *Journal of Applied Probability*, Vol. 10 Number 1, pp 354-367.

Hillier, F.S., and R.W. Boling. 1967. "Finite Queues in Series Exponential or Erlang Service Times - A numerical Approach" *Opns. Res.* 15, 286-303.

Hunt, G.C. 1956. "Sequential Arrays of Waiting Lines," *Opns. Res.* 4, pp 674-683.

Knott, A.D. 1970. "The inefficiency of a series of work stations - a simple formula," *Int. J. Prod. Res.*, 8, 109.

Latouche, G. 1981. "Queues with Paired Customers," *Journal of Applied Probability*, Vol.18 Number 3.

Muth, E.J. 1973. "The Production Rate of a Series of Workstations with Variable Service Times." *Int. J. Prod.*, Vol. 11, No. 2, pp 155-169.

Muth, E.J. 1979. "The Reversibility Property of Production Lines," *Management Science*, Vol. 25, No.2, February.

Pinto, P.A. and D.G. Dannenbring and B.M. Khumawala. 1981. "Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations," *Int. J. Prod. Res.*, 1981, Vol.19, No. 5, pp 565-576.

Smith, J.M. and S. Daskalaki. 1988. "Buffer Allocation in Automated Assembly Lines," *Operations Research*, Vol.36, No. 2, March-April, pp 343-358.

Svestka, J.A. and K.P.K. Nair. 1972. "Parallel Operation of Sequential Service Lines with Interline Transfer," *AIIE Transactions*, Vol.14, No.1, March., pp 29-34.

Yamazaki, G. and H. Sakasegawa. 1975. "Properties of Duality in Tandem Queueing Systems," *Ann. Inst. Stat. Math.* 27, pp 201-212.

Organization of computer software and Centralization /Decentralization of computer/information systems

Huff, Sidney L. *A* , 1978. *Case Study of Centralization vs. Decentralization and Stages of Growth in the Data Processing Function within the Canadian Banking Industry*, Working Paper, Sloan School of Management, MIT.

Hussey, John Worthen Jr. "A Methodology to Aid in the Centralization-Decentralization Decision for Large Decentralized Organizations," Master's

Lazear, T.J, J.F. Rockart, 1976. *Organization of the System functions in Engineering and Construction.* Report CISR-14 P852-76 Sloan School of Management.

Leventer, Joav Steve, 1976. "Centralization versus Decentralization of Information Systems: A Framework for Decision Making," Master's Thesis, Sloan School of Management, MIT.

Schell, Roger, 1981. "Dynamic Reconfiguration in a Modular Computer System," PHd Thesis, MIT Project Mac.

Siewiorek, D.P. and M.R. Barbacci, 1974. "Some Observation on Modular Design Technology and the Use of Microprogramming," Department of Computer of Science, Carnegie Mellon.