A Monte-Carlo approach to tool selection for sheet metal punching and nibbling

Emad Summad

ABSTRACT

Selecting the best set of tools to produce certain geometrical shapes/features in sheet metal punching is one of the problems that has a great effect on product development time, cost and achieved quality. The trend nowadays is, where at all possible, to limit design to the use of standard tools. Such an option makes the problem of selecting the appropriate set of tools even more complex, especially when considering that sheet metal features can have a wide range of complex shapes. Another dimension of complexity is limited tool rack capacity. Thus, an inappropriate tool selection strategy will lead to punching inefficiency and may require frequent stopping of the machine and replacing the required tools, which is a rather expensive and time consuming exercise.

This work demonstrates that the problem of selecting the best set of tools is actually a process of searching an explosive decision tree. The difficulty in searching such types of decision trees is that intermediate decisions do not necessarily reflect the total cost implication of carrying out such a decision.

A new approach to solve such a complex optimisation problem using the Monte Carlo Simulation Methods has been introduced in this thesis. The aim of the present work was to establish the use of Monte Carlo methods as an "assumptions or rule free" baseline or benchmark for the assessment of search strategies. A number of case studies are given, where the feasibility of Monte Carlo Simulation Methods as an efficient and viable method to optimise such a complex optimisation problem is demonstrated.

The use of a Monte Carlo approach for selecting the best set of punching tools, showed an interesting point, that is, the effect of dominant "one-to-one" feature/tool matches on the efficiency of the search. This naturally led on to the need of a search methodology that will be more efficient than the application of the Monte Carlo method alone. This thesis presents some interesting speculations for a hybrid approach to tool selection to achieve a better solution than the use of the Monte Carlo method alone to achieve the optimum solution in a shorter time.

A Monte-Carlo approach to tool selection for sheet metal punching and nibbling

By
Emad Summad
B.Sc. and M.Sc. [Industrial Engineering]

Thesis submitted to the University of Durham
for the degree of Doctor of Philosophy

Centre for Industrial Automation and Manufacture
School of Engineering
University of Durham
Durham DH1 3LE
United Kingdom

April, 2001

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Declaration

I, Emad Summad [Emad-Summad@altavista.net], hereby declare that the work reported in this thesis has not been previously submitted for any degree. All material in this thesis is original, except where indicated by reference to other work.

This work is dedicated to my father and my mother for their patience, understanding and support in its completion.

Acknowledgements

I owe my wife Aisha, my daughter Lana, and my twin boys Yaseen and Zaynadeen, a large debt for their patience, understanding, encouragement, and support during the course of the work. However, my debt to Aisha is especially great, she cheerfully put up with me for the duration of the study.

In addition, I owe more than the usual debt to my supervisor Professor Ernie Appleton. His support and contribution clearly transcends supervisor responsibility. Needless to say, all shortcomings in the structure and presentation of this work are entirely my own.

CHAPTER 1

INTRODUCTION

A major shortcoming of current product development practice is that designers do not anticipate the manufacturing implications of their decisions and manufacturing engineers produce components in a way which violates the designer's intent. In recent years, it has become widely acknowledged that integrating product design and manufacturing issues brings about an improvement in product development results. These results can be seen in the form of a reduction in product development time and cost, as well as helping to achieve higher product quality. Process planning, as an example, should be considered whilst product design is in progress and not left until after the design has been completed. This allows rapid reaction to changes in the functional requirements of products and to new technological opportunities. Integrating product design and manufacturing issues is known as Simultaneous or Concurrent Engineering. In essence the fundamental concept is to progress the design to simultaneously satisfy functionality, reliability, manufacturability and marketability. The present work is a contribution to progress in this general direction.

1.1. OBJECTIVE OF THIS WORK

The objective of this work is to study a novel, Monte Carlo, approach to tool selection for the sheet metalworking processes, punching and nibbling. Understanding and development of this approach should help to assist designers or production planners to quickly identify an optimum set of tools to carry out the punching of a particular product. Making tool selection concurrent with the design process should, in part, assist the designer to optimises the design with respect to manufacturing cost and enables the

production facility to more directly respond to the designer's intent. The journey from a desire for simultaneous engineering through to a method of selecting tools for a particular process is a long one and involves a number of interrelated decisions.

The features of the product, in part, dictate the manufacturing processes which the product must undergo for manufacture. Having a prior knowledge of the manufacturing processes can help to establish the design constraints that will guide the designer through the design of the product. Therefore, recognition of product features will enable the evaluation of designs and the development of process plans. This can be done by mapping the features to the manufacturing processes and to the tools needed.

A number of different feature recognition methods exist, for example, 'Syntactic-Pattern' methods and 'Volume-Decomposition' methods. In 'Syntactic-Pattern' methods, the sequences of geometric elements, such as straight-lines and circular arcs are used to describe 2-D geometric patterns. 'Volume-Decomposition' methods partition a design model into several small volumes or 'design features'. This is beneficial in the process planning stage because the method can be used to identify product features using some form of basic rules of logic. The rules are based on looking for certain patterns of elements or relationships so that a set of elements can be identified which can be classified as a feature.

Opponents of feature recognition systems [1], argue that when a designer starts the process of designing a product most of the features are already known by their abstract names, for example, hole, bend, notch, etc. However, during the process of presenting the design using a CAD or paper-based system, features are described using only straight lines and/or circular curves. Thus, much intrinsic information about features is

lost. This information must then be recreated in process planning by feature recognition methods and intuition. The suggestion implicit to a feature based approach is to provide the designer with a library of features, which can be used with a set of 'operators', for example, "Add", "Delete" and "Modify". However, the "design by features" approach still does not negate the need for complex geometric analysis or intuition because of complex feature interactions. It is these interactions which frequently provide the constraints on the manufacturing processes and tools to be used.

1.2. PROCESS PLANNING

Manufacturing features are the elements of the part that can be manufactured by one or a sequence of processes. It is therefore appropriate to relate design features to manufacturing processes. This step of conversion from design to manufacturing information is termed the "design interface for process planning". To achieve concurrency it is desirable to develop a systematic interface between the design system and process planning.

In process planning departments the process planners usually determine the manufacturing method on the basis of experience. Initially, they aim to meet the requirements for quality, as conveyed by the part drawings, whilst also considering the manufacturing costs. The "dream" of automated process planning is to have the ability to generate a process plan to manufacture a part automatically using no human input, the only input being the computer databases containing the required design and manufacturing information.

Two approaches to "Computer Aided Process Planning" (CAPP) are traditionally recognised; the 'Variant Approach', [2, 3] and the 'Generative Approach', [2, 3]. The variant approach is comparable to the traditional manual approach. Here, a process plan

for a new part is created by identifying and recalling an existing plan for a similar part. The "old" plan is then modified to accommodate the design features necessary for the new part. The main disadvantage of this system is that the "quality" of the process plan still depends on the knowledge background of the process planner. Here, a computer is simply a tool to assist in the manual process planning activity. In the generative approach, the process plans are generated by means of decision logic, formulae, technology algorithms and geometry based data. These are used to perform the many processing decisions required to convert a part from "raw material" to a finished state. When using the system, a specific process plan for a specific part can be generated without the involvement of the process planner. The term "Semi-Generative Approach" may be defined as a combination of the generative approach and the variant approach.

1.3. SHEET METAL PROCESSES

There is a large variation in the range of shapes and sizes of parts that can be produced by sheet metal forming processes. However, shearing processes are nearly always used to prepare the workpiece blank. This results in the need for a comprehensive understanding of the shearing or cutting process, for example, in blanking, punching, nibbling and notching.

In deciding which of the shearing operations to use, the total production rate is a major consideration. For example, if the total production run is sufficiently large to justify "progressive dies" then all blanking and forming can be done in one transfer press. In this case, parts can be produced, complete, at a rate of thousands per hour. However, such a high production rate may not be favoured in today's manufacturing environment where product life cycle is decreasing and faster delivery of components is required to ensure the commercial success of a product. In such situations, tooling of universal or wide-range applicability is utilised rather than dies dedicated to a single product. Thus,

in a flexible process a workpiece may be moved on a machine's worktable whilst different tools and dies are used to add the required features. For example, holes and notches of various sizes and shapes can be punched in various positions and nibbling can produce variously shaped blanks. This work considers the need for a better understanding of the process of selecting tool sets from the available standard or universal tool store.

1.4. TOOL SELECTION

The process of selecting tools to punch out sheet metal part features is a crucial process and has its influence on the quality of the part produced and on the cost of the production process. The literature review in Chapter 2 shows that the work undertaken to date to solve the problem of tool selection within the sheet metal industry can be considered at two different levels. The first is the design level, that is, selecting the best tools to produce a specific geometrical shape. The second is the planning level, which occurs prior to production. Here, the aim is to decide the sequence the forming operations should take and which tools should be placed in a limited capacity tool magazine. The main objective of the planning level is to reduce the total processing time including the process of stopping the machine, switching tools and positioning the required tools.

As will be seen in Chapter 2, all of the work done so far in this area of tool selection has been focused on selecting the best tools to produce particular geometrical shapes or has been restricted to certain shapes such as squares, rectangles and/or circles. Such restriction may be a wise start if it does not restrict the approach to being tailored to only a specific or narrowly defined problem.

In this work it is argued that concentrating on finding the best tool or tools for a particular feature does not guarantee the selection of the best set of tools to produce the part as a whole, bearing in mind the costly process of tool change due to limited tool magazine capacity. As a result, this work has not limited the search for the best set of tools by focusing on a particular feature and argues that by selecting, as an example, the fourth best option of tools for a certain feature might save one or more tool changes which is more costly than the cost of an extra one or two blows.

In trying to analyse such an optimisation problem, it was found, [4, 5, 6, 7] that the problem in hand is a problem of optimising an explosive decision tree. Any attempt to generate an analytical solution by mathematically formulating the problem would probably require the imposition of some restrictions on definition of the problem, which would then result in a tailored solution rather than a generic solution for the problem in hand. Thus, in this work a new approach to tool selection was taken in which the Monte Carlo method was investigated. The approach showed very promising results indicating the feasibility of such an approach.

1.5. THESIS LAYOUT

The material of this thesis is distributed over the next seven chapters. Chapters 2 and 3 are devoted for discussion of the related literature. Chapter 2 concentrates on the literature that could be considered to have a direct relationship to the present problem, that is, publications relating to tool selection for punching. Chapter 3, "Enabling Theory", explores related areas that may not have found direct application to tool selection so far but has been used in this novel approach. Chapter 4 discusses the Monte Carlo method and the way it has been utilised in this work. Chapter 5 discusses the computer software produced to facilitate conducting a number of experiments, which were necessary to examine the feasibility and integrity of the proposed approach. Then,

Chapter 6 discusses a number of case studies and theoretical issues related to the methodology, the chapter forming the basis of the results of the work and their discussion. Chapter 7 discuses some real industrial case studies. The thesis concludes with Chapter 8 where some concluding remarks are presented.

CHAPTER 2

LITERATURE REVIEW

The structure of this chapter and its intention requires some supporting explanation. It consists of two parts, the first part discusses previously published material and references relating to a contextual introduction to sheet metal work and tool selection. Hence, the initial part supplies a large number of references covering a wide spectrum but does not take a critical view. The intention is to allow a non-specialist reader to gain a wide understanding of this work's context, if that is required. The second part of this chapter, starting at section 2.4, focuses on specific references that have exercised a direct influence on the present work, its direction and techniques. In these instances it is quite appropriate for the author to take a more critical view of the material presented. However, the direct correspondence between earlier work described in this chapter and the current work is dealt with in later chapters. At that time the useful aspects of the previous work can be directly related or recognised as the foundation of the current research.

Traditionally, product development practices are conducted in a sequential manner, that is, activities associated with a certain stage of product development are completed prior to the beginning of the next stage of product development (Pahl, G. and Beitz [8]). Frequently, design work returns to an earlier stage for some corrections or changes in a reiterative manner. This can be a time consuming and costly process. Further, designers usually have to finish their designs before transferring them to the manufacturing facility. This is often done without the designer realising what implications their designs may have on the manufacturing process (Cutkosky, M. R. and Tenenmaum, J. M. [9]).

Manufacturing staff must not only understand the designers intention but they may also require some modifications to the current proposed designs in order to suit the manufacturing processes available. In some cases, they may have to return the design to the designers with suggestions for major changes. The disadvantage of such a practice is that it is a time consuming and costly process. Further, designers may not learn from their mistakes because these correction iterations are often not documented. Recently, Concurrent Engineering (CE) has started to be widely accepted as an approach that can achieve a better and more effective product development process (Jo, H. H. et al [10]).

2.1 CONCURRENT ENGINEERING

Concurrent Engineering is considered a "philosophy" for product development. If related product development activities are running in parallel, the product development time and cost can be reduced. In recent years industrial products have started to have shorter life cycles, thus, companies are under increasing pressure to shorten their product development cycle. For this reason, "philosophies" such as Concurrent Engineering have started to gain a significant industrial following (Finger, S. et al [11], Vujosevic, R. and Kusiak, A. [12]).

2.1.1 CONCURRENT ENGINEERING APPROACHES

A number of different approaches have been suggested to implement the Concurrent Engineering philosophy. Some of these approaches are listed below with some of their associated references: -

1. Team design (Cutkosky, M. R. and Tenenmaum, J. M. [9], Jo, H. H. et al [10], Vaiyapuri, V. and Okogbaa, O. G. [13], Young, R. E. et al 14]).
2. Design for assembly (Jo, H. H. et al [10], Vujosevic, R. and Kusiak, A. [12], Subramanyam, S. and Lu, S. C.-Y. [15], Appleton, E. and Garside, J. [16]).

3. Design for manufacture (Jo, H. H. et al [10], Subramanyam, S. and Lu, S. C.-Y. [15], Finger, S. and Dixon, J. R. [17], Bralla, J. G. [18], Appleton, E. and Garside, J. [19]).
4. Taguchi methods (Subramanyam, S. and Lu, S. C.-Y. [15]).
5. Multi-agent negotiation: different agents possess different knowledge and evaluation criteria (Sycara, K. P. [20], Huang, G. Q. et al [21], Wallace, A. and Boldyreff, C.[22]).
6. Quality Function Deployment (QFD): clearly defining the customer's needs and converting them into measurable factors (Akao, Y. [23]).
7. Rapid prototyping (Cutkosky, M. R. and Tenenmaum, J. M. [9]).
8. Cost prediction - Machining process evaluation and selection (Wong, J. P. [24], Yu, J-C. et al [25]).
9. Artificial intelligence constraint networks - Concurrent design is a constraint-based decision making process during which a design is analysed (Vujosevic, R. and Kusiak, A. [12], Young, R. E. et al [14]).

The features that exist in a designed part dictate the processes that the part must undergo during manufacture. Thus, having a prior knowledge of the processes that are available can help to establish the design constraints that will guide the designer in the design of the part (Nnaji, B. O. et al [26]). Since design features can dictate the way in which manufacturing decisions are taken, (Jo, H. H. et al [10], Young, R. E. et al [14]), then it is important to concentrate on the way in which these design features are analysed and translated into different manufacturing requirements. In the next section these design feature issues are discussed.

2.2 DESIGN FEATURES

Recognition of design features that exist in a part, will enable the part to be evaluated and the process plan to be developed. That is, the features can be mapped into the manufacturing processes and decisions can be made about the tools required to produce the final form (Gindy, N. N. Z. et al [27], Joshi, S. and Chang, T-C. [1]). The question that arises here is; what is a feature and how can features be represented and recognised?

2.2.1 FEATURES DEFINITION

The term "feature" has been used loosely by several researchers to define "regions of interest in a part, specifically, for the purpose of design, manufacture and analysis" (Case, K. and Gao, J. [28]). Each of the different stages of a manufacturing process "interprets" design features in different ways, for example, manufacturing form features, assembly features, fixturing features. (Allada, V. and Anand, S. [29], Feru, F. et al [30], Salomons, O. W. et al [31]). In the following section different techniques for feature representation are discussed.

2.2.2 FEATURE REPRESENTATION

There are several ways to describe a part and its features. Traditionally, designs are presented in a 2D engineering drawing, or physical prototype. Recently, with the increased use of computers in the design process, a number of different methods for representing part design and features have been utilised (Joshi, S. and Chang, T-C. [1], Lenau, T. and Mu, L. [32]). Some of these methods are listed below: -

1. Wireframe models (Case, K. and Gao, J. [28], Sing, N. and Qi, D. [33], Rooney, J., and Steadman, P. [34]).
2. Surface models (Sing, N. and Qi, D. [33], Rooney, J., and Steadman, P. [34]).

3. Solid models (Case, K. and Gao, J. [28], Salomons, O. W. et al [31], Sing, N. and Qi, D. [33], Rooney, J., and Steadman, P. [34], Bronsvoort, W. F. and Jansen, F. W. [35]).

A detailed discussion of these techniques is given in the references but is beyond the scope and need of this thesis.

2.2.3 FEATURE EXTRACTION

Traditionally, the feature extraction process is done by inspection by a human process planner in a process known as "reasoning" (Yu, J-C. et al [25], Salomons, O. W. et al [31]). However, since the use of computers has become more widespread the "dream" of fully automated manufacturing systems and ideas on automated process planning systems have started to merge. This has led researchers to question the way in which data is stored in computers and how suitable this data is for generating automatic process plans. This area of research has become increasingly known as the "Design/Planning Interface". There are two stages involved in any "Design/Planning Interface" the first is "Part Decomposition" and the second is "Feature Recognition" (Joshi, S. and Chang, T-C. [1]). As the reader will appreciate from the above terms, "Part Decomposition" is the separation of the features from the part model and "Feature Recognition" is the identification classification of the semantics/meanings of the feature, discussed in more details below.

2.2.4 FEATURE RECOGNITION

The feature recognition process involves recognising higher level features, for example, a hole, from the lower level geometrical entities represented within the part, for example, straight lines or circular curves (Nnaji, B. O. et al [26], Joshi, S. and Chang, T-C. [1], Sing, N. and Qi, D. [33], Pratt, M. J, [36]). Depending on the way the part

model is presented (2-D or 3-D), a number of different techniques have been introduced for recognising features. Some of these techniques are listed below.

1. Syntactic pattern recognition (Joshi, S. and Chang, T-C. [1], Allada, V. and Anand, S. [29], Bronsvoort, W. F. and Jansen, F. W. [35]).
2. Volume-Decomposition approach (Joshi, S. and Chang, T-C. [1], Sing, N. and Qi, D. [33]).
3. Expert systems approach (Joshi, S. and Chang, T-C. [1], Bronsvoort, W. F. and Jansen, F. W. [35]).
4. Graph based approach (Joshi, S. and Chang, T-C. [1], Allada, V. and Anand, S. [29], Bronsvoort, W. F. and Jansen, F. W. [35], Fu et al [37]).

Further detailed discussion of the above techniques is also beyond the scope of this thesis.

Some researchers argue, (Joshi, S. and Chang, T-C. [1]), that when the designer starts the design process, he/she is already aware of the features required. As such, during the process of converting them to the Computer Aided Drafting [CAD] system, information is lost and has to be recreated by feature recognition. Researchers have started to consider different ways of efficiently implementing the design/planning interface. Here, in one instance, designers could design their products using a predetermined library of features (Salomons, O. W. et al [31]). Such an approach is called "Design with Features" or "Feature-Based Design".

2.2.5 FEATURE-BASED DESIGN

The concept of "Feature-based design" is based upon providing the designer with a library of features which can be used with a set of operators such as "Add", "Delete" or

"Modify". The system then creates a features representation, which is also used to create a boundary representation of the object (Case, K. and Gao, J. [28], Feru, F. et al [30], Salomons, O. W. et al [31], Bronsvoort, W. F. and Jansen, F. W. [35], Young, R. I. M. and Bell, R. [38]). Since each application may have its own different set of features, then, there might be a design view, a process plan view and an assembly view of the same feature. Therefore, during the whole design and manufacturing process of a part, the different views must be maintained simultaneously and kept consistent in order to support the Concurrent Engineering philosophy (Salomons, O. W. et al [31], Bronsvoort, W. F. and Jansen, F. W. [35]). As mentioned earlier, the features that exist in a part dictate the manufacturing processes that the part must undergo i.e., the process plan to produce the part. The following section discusses the process planning stage.

2.2.6. PROCESS PLANNING

Process Planning determines how a product is to be manufactured. Individual features on a part and the relationship between individual features must be identified and represented (Smith, J. S. et al [39]). This allows the manufacturing processes and their sequence to be determined. A process planner's first task is to interpret the drawing and convert it to meaningful manufacturing information. This can be in the form of manufacturing features, datum surfaces, fixturing and/or locating faces. These can then be used to plan the details of the manufacturing process (Halevi, G. and Weill, R. D. [2], Molengraaf, J. C. M. V. D. et al [40], Narayanan, V. [41]).

Manufacturing processes can be broadly divided into the following categories (Halevi, G. and Weill, R. D. [2]): -

1. Forming from liquid - for example, casting and moulding.
2. Forming from solid by deformation, including: -

2.I. Hot working, for example, hot rolling, forging and extrusion.

2.II. Cold working, for example, stamping, bending, spinning, shearing, cold rolling, extrusion, deep drawing.

2.III. Forming from powder, for example, powder metallurgy and plastic moulding.

3. Forming from solid by material removal, for example, milling, turning, drilling.
4. Forming by joining parts - for example, the use of welding, brazing, soldering and adhesive to join several components into a more complex assembly.
5. Forming by assembly - for example, the mechanical joining of parts by threading, bolts, rivets.
6. Forming by material increase -Material is progressively added until the whole part is created, such as stereolithography.

The process planner, when deciding upon a suitable manufacturing process, usually considers a number of design factors. These can have a bearing on the choice of the manufacturing process used (Halevi, G. and Weill, R. D. [2]). For example,

1. Production Quantity
2. Part Geometry
3. Part Size
4. Part Material
5. Part Dimensional accuracy
6. Cost of raw material, possibility of defects and scrap rate
7. Subsequent processes

Early attempts to create automated planning systems consisted of building computer-assisted systems. These were used for planning and reporting on part generation, storage, and retrieval. Most Computer Aided Process Planning [CAPP] systems developed in the 1970s used Group Technology (GT) classification as the major

approach. In the 1980s more generative and knowledge based approaches were used in CAPP development (Chang, T-C. [3]).

Study of the literature available on Computer Aided Process Planning (CAPP), (Smith, J. S. et al [39], Alting, L. Zhang, H. [42]), shows that little research has been carried out on the processing of sheet metal parts. However, as the reader will appreciate there are many thousands of products made from sheet metal. Also, the majority of previous research undertaken into sheet metal CAPP systems adopted a variant approach which recalls an existing plan for a similar part and then makes the necessary modifications required for the new part. As sheet metal products vary in complexity and therefore may require complex manufacturing processes, difficulties may occur in maintaining consistency in the editing practices and the accommodation of various combinations of geometry, size and material. This has led to a need for a better understanding of sheet metal parts and their processes. In the following section, sheet metal processes are discussed briefly.

2.3. SHEET METAL PROCESSES

Despite the variation in the range of shapes and sizes of parts produced by sheet metal forming, (Bralla, J. G. [18], Mielnik, E. M. [43]), it is usually found that almost any shape can be produced by use of one or more common processes, for example, shearing, drawing or bending. However, shearing is nearly always used in the preparation of a work piece blank. This has resulted in the need for a greater understanding of shearing and cutting processes and their application.

2.3.1. SHEARING PROCESSES

Shearing or cutting processes, (Bralla, J. G. [18], Mielnik, E. M. [43], Niebel, B. W. et al [44]), may be classified as follows:

1) Operations required to produce blanks include: -.
   a) Cropping - The cutting action is along a straight line.
   b) Cut-off - The cutting action is along a line which is not necessarily straight.
   c) Parting - Produces a small amount of scrap.
   d) Blanking - Produces a workpiece or part with an enclosed contour.

2) Operations for the shear cutting of holes include: -
   a) Punching - Produces holes with a variety of shapes and sizes.
   b) Nibbling - Separating by continuously punching a small slot. The shapes produced can be varied.
   c) Slotting - Forms elongated or square holes.
   d) Perforating - Holes are pierced, usually close together.

3) Operations involving restricted or partial cutting.
   a) Notching - A piece of metal is removed from the edge of a blank or strip to form a notch.
   b) Semi-notching - Metal is removed from the central portion of a strip to facilitate subsequent bending or for providing part attachment along the edges for progressive forming processes.

4) Operations for size control.

a) Trimming - Cutting excess or damaged metal after a forming operation such as deep drawing.

b) Slitting - Cutting a wide coil into several narrow coils in a rotary shear called a slitter.

c) Shaving - Cutting off metal in a "chip like" fashion to remove the rough fractured edge of the sheet and therefore obtaining accurate dimensions.

In deciding which of the above shearing operations to use, the total production rate becomes the deciding factor. For example, if the total production run is sufficient to justify progressive dies then all blanking and forming can be done in one press stroke. In this case, complete parts can be produced at a rate of thousands per hour (Bralla, J. G. [18]). However, as mentioned earlier, such high production rates may not be favoured in today's manufacturing environment, especially as product life cycles are decreasing. Subsequently, components must be delivered quickly to ensure commercial success of a product. In such situations, tooling of 'universal' or 'wide-range', [45], applicability is utilised rather than dedicated dies which are made for a specific operation. In a "universal" process, a workpiece may be punched out or produced using one or more tools, while the workpiece is moved on the machine's worktable. For example, a key-shaped hole can be punched out using a standard circular tool and a standard rectangular tool instead of having a dedicated key-shaped tool. Therefore, holes and notches of various sizes and shapes can be punched, and nibbling can produce variously shaped blanks. However, punching machines have a limited capacity, which restricts the number of tools that can be available during one set-up. Therefore, if a tool is needed and it happened that it is not available in the machine turret, then a machine switch off is needed in order to install the required tool. Such a process is usually called "tool set-up" and is a relatively costly process.

For the reasons given above, it can be seen that a better understanding of the process of tool selection from a standard or universal tooling range is required in order to select the best set of tools and minimise cost. The following section considers the tool selection process for the sheet metal industry in more detail.

2.4 TOOL SELECTION

During this literature review it became apparent that the work undertaken to date to solve the problem of tool selection within the sheet metal industry has been undertaken on two different levels. The first is the design level; selecting the best tools to produce specific geometrical shapes. The second is the planning/scheduling level prior to production. This is a purely operational level; the aim is to decide the sequence which the manufacturing jobs should take and which tools should be placed on the limited capacity tool magazine. The main objective of the planning level is to reduce the total processing time, for example reducing the times the machine is stopped for switching tools and re-locating the required tools. The rest of this section is devoted to discussing some of the related literature. The approach taken from this point on is to discuss previous relevant work on an individual author level, which is then followed by a general conclusion for the whole chapter and a description of the proposed research.

During this research an interest was taken in developing a design-supporting tool which could be used at the design level as well as at the planning level. The idea was that if tools could be optimally selected for a current design specification, then the impact of such tool selection could be demonstrated at the design level. Here, such a tool would give designers the opportunity to make any changes necessary to the current design in order to improve the efficiency of the production process. Thus, the planning department may not have to "re-invent the wheel" as the optimum set of tools would have been nominated at the design stage.

In order to fulfil such an objective a full and thorough survey of the reported literature for tool selection within the sheet metal industry was conducted. The scope of this search has been limited to the production of flat sheet metal components using punching, and/or nibbling operations.

Research into the job scheduling problem relating to a flexible manufacturing machine has been undertaken by Tang, C. and Denardo, E., [46,47]. This type of problem would regularly occur in the planning department of any metal working industry where the problem is to decide on the sequence by which to process the jobs and the tools to place on the machine before each job is processed. They assumed that 'n' jobs can be processed on a machine that has sufficient capacity for 'C' tools, but that no job required more than 'C' tools. If the required tools for the next job are not on the machine, then one or more tool changes must occur before the job can be started. It might be worth mentioning here that a "tool switching instant" is that moment of time when at least one tool is "switched" i.e., a tool is removed from the machine and a different tool is inserted on the machine. They considered this scheduling problem with two different performance criteria in mind, leading to two entirely different solutions. The first criterion was to minimise the total number of tool switches. This is adequate for situations where the time needed to switch tools is significant relative to the job processing time i.e., tool changing time is proportional to the number of tool changes (individual tool changes). The second criterion was to minimise the total number of occasions/instances at which tools require changing. This criterion is appropriate when the tool changing time is roughly constant and independent of the number of tools being changed (whole magazine changes). The above shows the impact that a decision taken at the design level can have on the total time required to process a job and on the production process as a whole.

The problem of rationalising tool selection in flexible systems for sheet metal manufacturing was also considered by Daskin, M. et al, [48] who focused their work specifically on circular interior holes. If a design requires a round hole, the nominal diameter of the hole, as well as its required tolerance, must be specified. For example, a circular hole may have a nominal diameter of 0.50 cm with tolerances of + 0.02 cm and – 0.02 cm. Each hole specification should therefore define an interval on the real line. For example, the above mentioned hole will have the interval of [0.48 , 0.52]. Similarly, each tool should define an interval on the real line, see Figure 2.1. For example, a tool with a corresponding interval of [0.49 , 0.51] can be used to punch the above mentioned hole. Features and tools together define an (m x n) 0-1 matrix, the rows of which correspond to the holes and the columns of which correspond to the tools. The objective of this is to minimise the number of holes that cannot be punched by 'standard' tools. The work of Daskin, M. et al, [48], was limited to round holes, applying such an approach to non-round holes might not be possible since it is difficult to establish an (m x n) 0-1 matrix. Hence, this restricts the approach with respect to it being generic.

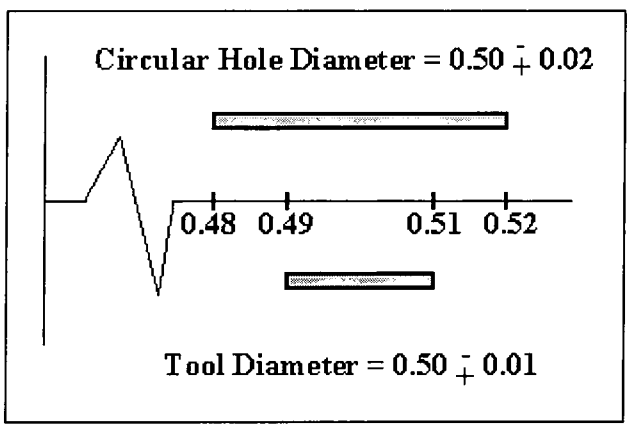


Figure 2.1. Example hole and tool intervals on the real line.

The work of Daskin, M. et al, [48] was extended and incorporated the problem of selecting tooling as a problem of minimising the production costs. Hsu, V. N. et al, [49]

presented an argument which was based upon the fact that eliminating as much custom tooling as possible will lead to the need for fewer tools and this would in turn yield significant savings. Thus, in their work they tried to answer the question of which tools to select to optimise part production if the tool rack capacity was limited. Once again, they limited their work to round holes. However, they discussed the idea of 'penalty costs' for not punching a given hole type. Further, they discussed fixed costs for each incomplete part, regardless of the number of unpunched holes. These fixed costs could include the cost of removing the part from the machine, rescheduling and reloading the part in another production shift, or sub-contracting the part to another supplier. This costing based approach may help when examining the redesign of some products.

An expert system designed for the generation of cutting plans was introduced by Ehrismann, R. et al, [50,51,52,53]. The task of this expert system was to determine the most economical method of manufacturing a flat sheet part as well as the generation of the appropriate Numerical Control (NC) program for a laser cutter and punching machine combination. To determine the required punching tools, the individual contours of the part are considered. The geometry processor analyses the sheet part geometry and sub-divides the contours of the part into inner contours and outer contours. It also distinguishes between unknown shapes and known shapes, such as, circles, squares and rectangles. In addition, it manages the available punching tools which are also classified into geometrical groups. The selection of the optimum tool is performed using "contour rules". Each contour has specific properties which are stored in the expert system as “starting facts”. Rule groups are activated based on these starting facts. If, for example, a contour belongs in the "class of holes" and if the contour has a known form, that is, it is a circle or a square, then the "one-punch rules" are activated and they select a tool which can punch the contour with a single blow. However, if the

contour belongs in the "class edge" or if it has no recognised form, then the "one-punch rules" are not activated.

A different tool selection approach incorporated within an automatic process planning system, (PART-S) was developed by Liebers, A. et al, [54], for the small batch manufacture of sheet metal parts. Form features are used to relate geometrical patterns and their parameters, to specific production methods. For every feature in the set-up, the method selection module determines the tool type and its maximum size capability. The system interrogates the method selected for all features in the set-up, specifically their tool types and the corresponding parameter ranges. The tool that can be used for the largest number of features is selected. A selection procedure like this may lead to the selection of a 'needle' tool, which would be suitable for punching nearly all features. To avoid this deficiency lower bounds are used. Such a lower bound is defined as a minimum tool size expressed as a percentage of the size of the feature.

PC-based software capable of automatic tool selection for Computer Numerical Control (CNC) nibbling has been developed by Choong, N. F. et al, [55,56]. The selection process was divided into two stages. In stage one, each profile is considered independently and a suitable set of tools is selected. Profiles are categorised into eight different types, they are; general external profile, general internal profile, square hole, rectangular hole, triangular hole, oblong hole, arc-slot hole and circular hole. Each type of profile has a set of technological and heuristic rules governing the types of punches that can be chosen. In stage two the software reconsiders these independently selected tools and subsequently plans for a single tool set-up to produce the complete part. The tool selection criteria is based upon the analysis of the characteristics of three different types of nibbling machines. Two steps are involved in this stage. Step one reduces the

total number of primary selections to the number of tool stations that can be accommodated. Tools selected in the primary selection are reselected based upon the following criteria, listed here in order of priority: -

1. Tools that are manually selected for certain parts of a profile.
2. For lines with small vertex angles.
   2.1. Triangular tools if the machine has no laser cutting facility.
   2.2. Laser cuts if the machine has suitable facilities.
3. Tools that are frequently selected and have the highest priority, that is, they would be the first choice for a profile.

Based on tools selected in step one, step two selects only one solution from the primary selections obtained for a specific profile.

A different approach for tool selection using parametric design techniques which relates the geometrical shape of features to the geometry of tools has been discussed by Tilley, S., [57]. Each shape is reduced to more basic shapes, until only circular and rectangular shapes remain. This method is based on the assumption that the tools required to punch the circular and rectangular features are the most commonly available. According to the results of the initial tool selection module, a final tool selection is made. The aim is to minimise the cost of operation. This cost depends on: -

1. The set-up time of the machine, this in turn depends upon the number of tools that have to be changed on the turret.
2. The number of tool strokes.
3. The total length of the tool path.

Tilley, S., [57] argues that the calculation of the optimum tool settings for a machine turret would have to be exhaustive and would need complicated calculations and consequently would take too much time. Hence, he suggested the tooling set-up problem should be left for the user of the system who would define a strategy to select the set-up of the machines. Therefore, the user of the system would allocate the priority of one punching method over another, for example, nibbling over laser cutting, area punching over contour punching, single stroke over multiple strokes. These priorities can be specified for all types of materials and their thicknesses. Tilley, S., [57] reflected on the complex nature of the problem of selecting the best set tools in a way that involved cost, time and quality issues.

Introducing their approach for automatic tool selection and the determination of their locations and orientations for a given sheet metal part, Cho, K. H. and Lee, K., [58], state that each part and its tool boundary curve can be represented by a set called the "shape-index-set". This "shape-index-set" is composed of four elements which are; index of the angle with preceding curve, index of curvature, index of length, index of the angle with proceeding curve. A tool with a boundary curve matching the complete or partial boundary of a given sheet metal part is defined as a "candidate tool". The following four factors determine whether a tool is a candidate;

1. A tool that has a sequence of its boundary curves matching the corresponding sequence of the part's boundary curves.
2. A tool which has a matching corner angle.
3. A tool that has one boundary curve matching the corresponding part's boundary curve, completely or partially.

4. A tool by which some boundaries of the part can be punched out, approximately, within the predefined tolerance.

The "candidate tools" are stored in a descending order based upon the number of matching curves. After the candidate tools are ordered, a module called "interference test and punching" is invoked. In this module, the tools are tested for interference according to their order and the punching operation is performed with the first interference-free tool found. Limited tool turret capacity which limits the number of tools to be considered has not been discussed. This work by Cho and Lee had a significant influence upon the approach taken in the present research. In the present work the set of boundary curve elements is known as a "curve signature".

It can be concluded from the above discussion that most of the previous reported work has restricted the shape of features and tools to basic shapes which resulted in the generation of a non generic methodology that works best for particular geometrical shapes, for example, internal circular holes. On the other hand others have restricted the search for a particular type of tools called "candidate tools" using rule-type searching. Such an approach has the main drawback that there will always be new cases which will violate the existing rules and new rules must be continuously introduced and maintained. Therefore, most of the previous work can be generalised as methods of reducing the search space of all possible solutions and thereby simplifying the tool selection problem. This in itself reflects the complex nature of the problem under study. It is also worth mentioning here that selecting the best set of tools for punching one particular feature does not guarantee the selection of the best set of tools for producing the part as a whole, especially when considering the limited space of a tool cartridge. Therefore, it is important to be able to decide which punching tools to select in order to

optimise part production when tool rack capacity is limited. In general, any given sheet metal part may require holes of different shapes and sizes. Optimising part production is different from optimising hole production, especially when the tool rack capacity is limited. For example, a product with ten different hole-shapes could be most efficiently produced, from a blow count point of view, by using ten dedicated tools. However, if the tool rack capacity is only six then an expensive tool change would be required. In contrast, if the tools selected were restricted to the rack capacity of six an expensive and time consuming tool change can be avoided and one of the existing tools could be used to punch out a larger shape by using a series of blows in a nibbling mode. Summad, E. and Appleton, E. [4,5,6, 7], proposed an approach that did not restrict the geometry of the features considered and did not restrict the search for the best tool for a particular feature. However, the problem resulting from this approach was the searching of an explosive decision tree.

2.5. COMMERCIAL SOFTWARE - REVIEW

From searching the literature of the available commercial software for the sheet metal industry, a number of different software companies were found. These companies compete in order to provide a complete range of services. These services range from drawing/creating a sheet metal part, data import and export between different CAD systems, shape recognition, tool selection, tool path optimisation and nesting. However, it is worth noting here that the review indicated that all of this software offers automatic assignment of punching and nibbling tools as a function of preferred rules, configured either by the user or the system developer. Such rules are biased to use the least number of hits/blows and do not take into consideration that using fewer blows might mean an extra, expensive, tool set-up which could be avoided by accepting a solution using more blows. In order to overcome such a problem, software usually offers the possibility of interactive modifications to optimise the turret station operations. The frequent need for

such interactive modifications often leads users to abandon the software approach and to do the tool selection manually.

In common with many other adaptive process planning approaches, rule based tool selection is often application or user specific and hence not generic. It also has the disadvantage of embedding current practice rather than seeking an optimum solution. In the following section a representative group of the available commercial software will be briefly discussed. However, none of the software suppliers publish detailed information on their tool selection strategy because of the commercial nature of products business and the detail included here is often by deduction.

The software reviewed includes:

1. Radan - Radpunch [http://www.radan.com/radpunch_prof.php3#radpunch]
2. Pro/ENGINEER - NC Sheetmetal
   [http://www.ptc.com/products/proe/production/ncsheetmetal.htm]
3. Alma - act/cut [http://www.alma.fr/productique/ang/act_cut_tech_ang.htm]
4. Camtek - PEPS Expert Punching [http://www.camtek.co.uk/prod07.html]
5. DRI - Proto DB-32 [http://www.drifab.com/Sheet_Metal/sheet_metal.html]
6. Tol - TOLpunch [http://www.tol.fr/tolfab/htmlgb/complet.html]

2.5.1. RADAN - RADPUNCH

Radan is one of the world's leading producers of CAD and CAM systems for the sheet metal working industry. From the start of the development of Radpunch in 1976, Radan's aim has been to develop comprehensive CAM systems for every type of CNC sheet metal working machine tool.

Radpunch includes comprehensive 2D CAD software for component geometry entry with dimensioning, geometry editing and standard parts libraries. Alternatively, data can be imported into the system from other CAD systems by IGES, DXF or by direct transfer. Radpunch offers comprehensive sheet metal CAM programming for punching and nibbling machine technologies and includes a database of tools, machine data and materials data. However, Radpunch only offers automatic tool selection as a function of preferred rules, i.e. rule based selection. These rules are divided into four areas, set-up, runtime, max size to scrap and tool fit tolerance. Optimisation for set-up will ensure that tools which are already loaded into the turret will be considered before any others. However, optimising for runtime will consider all the tools which are specified in the Master Tool File and select those tools which produce the least number of hits. Selecting 'Max size to scrap' will make sure that small cutouts are always cut to scrap, while large ones are nibbled out to retain usable material. Finally, 'Tool fit to tolerance' will specify how closely the tooling must match the part geometry. Max undersize and Max oversize are the amounts by which a tool can be too small, and too large, respectively.

In addition, all arc slots are nibbled whereas, straight edges will only be nibbled with a circular tool as a last resort, that is, when no suitable rectangular or square tools can be found.

Many Radan customers who have used such a system over a considerable time period, have a lack of confidence in the automatic tool selection module and prefer to do selection manually.

2.5.2. PRO/ENGINEER - NC SHEETMETAL

Pro/ENGINEER, a powerful and productive CAM solution, aims to provide a complete solution including tool path generation, NC simulation and NC post-processing. After parts are designed in Pro/ENGINEER or imported through a DXF file, the NC sequences are defined and nests created automatically and sent to the machine tool. Once again, Pro/ENGINEER relies on pre designed rules for selecting the required tools. There is a lack of published material on these rules for obvious commercial reasons.

2.5.3. ALMA - ACT/CUT

Alma, created in 1979, is a leading company in the development of CAD/CAM software for industrial cutting and robotics, which offers a range of computer assisted manufacturing software products for the optimisation of cutting processes for the sheet metal industry.

Act/*cut* software developed by Alma, also offers automatic assignment of punching and nibbling tools as a function of preferred rules with the possibility of interactive modifications to optimise the turret station.

The software has an integrated knowledge base to take account of the operator's know how including:

- Automatic calculation of roughness values and overlap ratio.
- Automatic duplication of repetitive machining.
- Automatic gouging of recesses.
- Optimisation of tool strikes for gouging.

- Choice and management of removal methods (trap doors, suction cups, micr-junctions).
- Display of tool volumes and sizes.
- Simulation of tool paths.
- Display of turret station with optional tool substitution.
- Automation definition and chaining of tool trajectories on the sheet per working zone with the possibility of interactive modification.
- Management of clamps repositioning.
- Management of parts removal and/or holding (trap doors, suction cups, micro-junctions, etc.)

2.5.4. CAMTEK - PEPS EXPERT PUNCHING

PEPS (Production Engineering Productivity System) was the first CAD/CAM system to be redeveloped and released for use under Windows. The PEPS Expert Punching module is used for punching and nibbling operations on a wide range of turret punch machines. PEPS Expert Punching provides automatic shape recognition, rule-based tool selection, auto tool path creation and the full graphical simulation of tool path but the published literature gives no detailed information on the tool selection strategy.

2.5.5. DRI - PROTO DB-32

Over the last 22 years, Digital Resources have focused on providing automatic programming for sheet metal punching and laser cutting. The Proto DB-32 automatically performs tool selection using a rule-based approach that reflects user's tooling preferences. A few examples are discussed below.

Rules for linear and radial nibbling include:

- Ability to designate specific tool to use for nibbling.
- User defined maximum allowable nibbling tool diameter.
- User specified maximum allowable scallop.

Rules for punching and nibbling round holes include:

- Plus and minus tolerances to determine whether single or multiple hits are required.
- If nibbling is required, which tool will "rough out" the interior portion of the hole, and which tool will be used for "finishing"
- Maximum scallop parameters are enforced if nibbling occurs.

Rules for obround (round-ended rectangle) tool use include:

- Use of longer tool for 3-sided obround (obround cut on edge of part) is user defined.
- Detailed control of obround tool is afforded on punching of part perimeter.

Rules for trimming/edge parting include:

- Parting tools can be pre-designated.
- Parting may be turned on or off for any edge individually.
- Width constrains may be applied to tools which may be selected for edge parting.
- Over-cut (using a tool that is longer than the edge being cut) may be allowed and controlled easily.
- Edge overlap may be forced on outside corners of part to ensure clean corner cut.
- Amount of tool overlap on inside corners is user defined to eliminate nibble marks.
- Automatic tab creation is easily invoked and controlled.
- Bridge punching (for heavy material) may be automatically invoked based on material type and thickness.

- Minimum percentage of tool to be used may be defined in terms of both tool length and width.
- Overlap minimums may also be specified in terms of tool length and width and is automatically enforced.

The system is biased towards using single hits for punching hole shapes. The tolerance allows the system to automatically map, for example, a round punch to a circular hole as long as it falls within the allowable variance. However, if no tool is found that falls within the allowable variance, pre-defined punching parameters are applied for roughing out the core of the hole and then nibbling constraints are applied for finishing off the hole edge. These constraints include specifying the tools to be used and the allowable scallop for nibbling.

2.5.6. TOL - TOLPUNCH

TOL S.A. has been in the business of cutting technologies, punching, nibbling and laser cutting, for more than 10 years. TOL S.A. developed a range of programming systems for 2D numerically controlled cutting machines. TOLpunch, automatically computes the tool path for the parts to be cut. The operator can modify an element of the tool path or the whole tool path. Once again there is a dearth of published information about this package with respect to tool selection strategy.

The next chapter describes other published material and synthesises ideas from a range of non-manufacturing areas such a computer vision, data structures, optimisation etc. Such background and techniques had significant influence on the direction of this research.

CHAPTER 3

ENABLING THEORY

This work has attempted to take a somewhat radical view of tool selection. As such, it is based upon the synthesis of ideas from a range of non-manufacturing disciplines. The previous chapter has dealt with literature that could be considered to have a direct influence upon the present work, where views and ideas have been absorbed and used directly. The literature in this chapter should be considered as the embodiment of the conditioning of the thoughts used and, as such, it explores related areas that may or may not have found direct application to the present approach to tool selection. Literature is presented here because it describes techniques that have been used in different areas but, in essence, where there is a similarity between the initial problem and the problems formulated in the current work.

Today, tools are very expensive, as such, it is no longer acceptable to buy a "special" tool for each geometrical shape to be produced. The trend is, where possible, to limit the use to 'standard' tools. Therefore, it is possible to create complex geometrical shapes out of simple standard geometrical sub-shapes by using a set of "standard tools". Tool selection for this shape generation strategy led to consideration of the capability of computer vision analysis, which is concerned with identifying an object (shape) as one of a collection of known model objects (shapes). In a complex version of the problem, the objects may be partially occluded that is several objects may overlap. [59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]. In a similar way, in punching operations it is acceptable to allow the overlap of simple geometrical shapes, such that one tool can be used, provided a part of the feature boundary is being cut. The whole boundary can be

generated by a series of blows by "overlapping" tools. Therefore, almost any tool can be used, provided it does not damage the workpiece. However, issues such as limited machine tool capacity, production cost and time will restrict such a wide freedom of choice.

3.1. COMPUTER VISION

In the human visual perception process, it is considered that object recognition capability is based on an impressive and extensive visual memory of images. A visual input is related to some previously existing objects in the world. For example, looking at a picture, a human being can recognise a tree, based on the visual memory of the tree image. However, if somebody is given an aerial picture of a seaport and is asked to identify a ship the person may find it difficult, especially, if the ship's appearance from an arial perspective does not exist already in the person's visual memory. With this in mind, computer vision consists of a number of different activities, they are,

3.1.1. IMAGE FORMATION [70, 71, 72]

Image formation occurs if a sensor registers radiation that has interacted with physical objects. This activity concentrates on a mathematical representation of an image and methods for obtaining a digital image within the computer. The image of both the tools and the features within this current research are assumed to be held in a Computer Aided Drafting [CAD] database and therefore image formation is not a problem within the scope of the present work.

3.1.2. EARLY PROCESSING [70, 71, 72]

Once sensor data has been obtained it is usual to do some preliminary processing in order to enhance the appearance of the objects. Such processing may include filtering the image in order to detect a particular feature or features within an image, provided

that the appearance of the features within the image are accurately known. Once again, this was not considered to be of direct relevance for the reasons described above.

3.1.3. BOUNDARY DETECTION [70, 71, 72, 73, 74, 75]

The boundary of an object plays a very important part in the hierarchical structure which is used to describe such an object. Obviously, an image consisting only of disconnected edge elements is featureless. Therefore, this image would require additional processing to group the edge elements into meaningful structures, called features. Thus, the objective is to make a coherent, one-boundary feature from many edge elements, for example, the making of a square boundary feature from four connected straight-line edge elements. Different algorithms and methods are used here, [70, 63], based on the amount of information available. However, they all use the same approach which is described as "boundary recover by edges following".

Boundary detection probably represents the start of an overlap of interest between computer vision and tool selection, due to the fact that both have an interest in edges. However, it is considered that the different algorithms and methods employed in computer vision are beyond the scope of this research. Such areas of research are well known within manufacturing engineering as "feature recognition". This has been briefly discussed in the previous chapter. The automatic recognition of features is beyond the main scope of this work therefore, the detection of features is done manually where input to the proposed work is a list of all features and their entities i.e., straight lines and circular curves. This activity does not represent additional work for the designer or process planner because it is an inherent part of constructing CAD drawings of parts and tools.

3.1.4. MATCHING [70, 71, 72]

A matching process is used to establish a correspondence between the input data and the pre-existing representations. This area of computer vision can be considered to be a direct comparison between tooling and features and therefore has a direct influence on tool selection. Although matching using computer vision has not been applied directly to the present work, techniques and tools within this area have been used, for example, decision trees, [70] and curve signatures, [70, 66].

Computer vision is strongly reliant upon the management of data and its representation, [70]. Therefore, an interest in data structure and data representation resulted from an understanding of computer vision.

3.2. DATA STRUCTURE AND PRESENTATION

As mentioned in Chapter 2, tool selection problems can involve the searching of an explosive decision tree where the size and hierarchical structure of such a decision tree is case specific and initially unknown. Thus, when choosing a certain data structure great care must be taken in order to handle the possibility of an explosive decision tree. Therefore, an understanding of the different types of data structure is helpful in selecting a data structure type for handling such an optimisation problem.

A number of different data structures were studied in order to determine their suitability to the problem being considered. Some of these structures are discussed below.

3.2.1. LISTS [76, 77, 78]

Lists are particularly flexible structures because they can grow and shrink on demand and elements can be inserted or deleted at any position within a list. Inserting an element into the middle of the list requires shifting all of the following elements within

the list to make room for the new element. Similarly, deleting any element, except the last, also requires shifting elements to close up the gap. If there is an "n" element list and it is desirable to insert a new element between the first and second elements, then the last elements (n-1) of the list must be moved and so on up the list, thus making room for the new element. For a list that contains many nodes, this is a rather inefficient way of performing an insertion task, especially if many insertions are to be performed.

In a number of applications, given a new element, it may be desirable to determine the preceding and following elements quickly. For example, in a tool selection problem, in allocating a curve signature to a particular curve entity, linear curve, both entities, also lines or curves, before and after should be determined. In such situations, each element on a list is given a pointer to both the next and previous elements on the list, see Figure 3.1. In the case of linked allocation, an addition is performed in a straightforward manner. If a new element is to be inserted following the first element, this can be accomplished by merely interchanging pointers. It is clear that the insertion and deletion operations are more efficient when performed on linked lists than on sequentially allocated lists. There are also some special kinds of lists such as stacks, queues, discussed below, which might be useful for other applications.

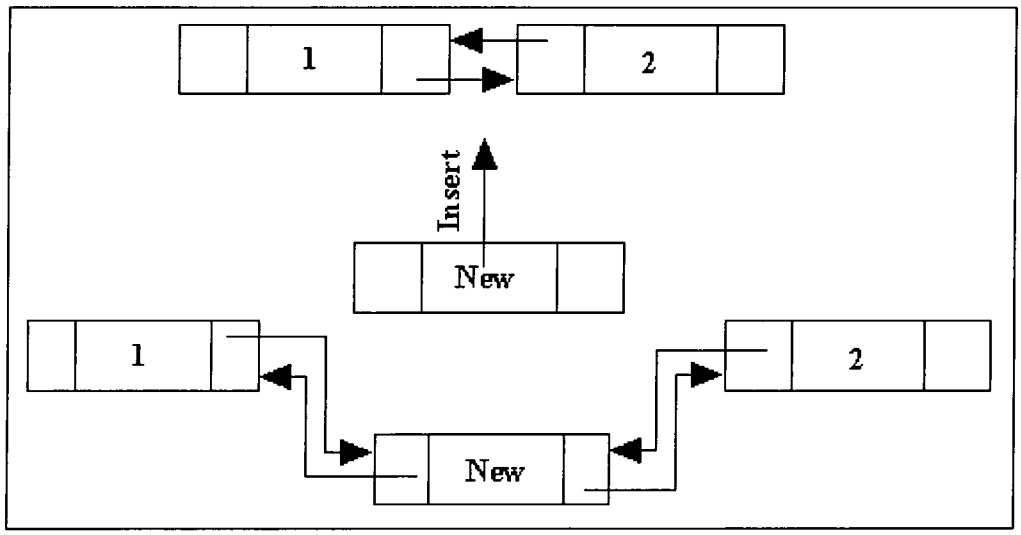


Figure 3.1. Insertion in the middle of a double linked list [78].

3.2.1.1. Stacks [76, 77, 78]

A stack is a special kind of list in which all insertions and deletions take place at one end, called the "top". Other names for a stack are "pushdown list", and "LIFO" (Last-In-First-Out) list. Once again, stacks do not represent structure, as structure would necessarily involve hierarchy.

3.2.1.2. Queues [76, 77, 78]

A queue is another kind of special list. Here, items are inserted at one end (the rear) and deleted at the other end (the front). Another name for a queue is a "FIFO" (First-In-First-Out) list. Queues, also, do not represent structure as, once again, structure would involve hierarchy.

3.2.2. TREES [76, 77, 78]

Trees are useful in describing any structure which involves hierarchy. Familiar examples of such structures are family trees. Sheet metal parts and their features also represent an example of a description containing a structure that involves hierarchy. Linked allocation can be used to represent the tree structures. A cell having two pointer fields and one or more information fields can represent a node. Such a representation

appears to be more popular due to the ease with which nodes, for example, feature edges can be inserted into and deleted from a tree.

The diagram in Figure 3.2. shows how a tree structure has been used in this work. The full line indicates the decision taken and the node indicates the accumulated cost function value to reach this point. The broken lines are indication of the vast number of decision options that were not selected. This illustrates how the decision tree expands explosively.

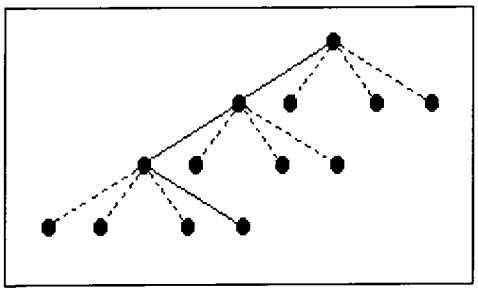


Figure 3.2. An example of a decision tree.

If unconstrained tree structures can grow to an unwieldy and unpredictable size the efficiency of searching may become critical. Hence, it may be useful to measure the performance of a tree search method by counting the number of key comparisons required to find a particular record or answer. In many applications, for example tool selection, it is not only necessary to find a particular node within a tree, but it may also to be necessary to move through the nodes of the tree, visiting each one in turn. If there are "n" nodes in the tree, then there are "n!" different ways of visiting them. Thus, it can be seen how difficult the search may be if the tree is of an explosive nature and the number of nodes to reach a solution is affected by the decision route taken through the tree structure. In the next section a number of optimisation techniques are introduced and examined in order to be able to decide on the most adequate tree searching technique for the application in hand.

3.3. OPTIMISATION TECHNIQUES

The ever-increasing demand on engineers to lower production costs to withstand competition and improve performance has prompted engineers to look for robust methods of decision making, such as optimisation methods. Optimisation methods, coupled with modern tools of computer-aided design, are also being used to enhance the creative process of conceptual and detailed design of engineering systems. With rapidly advancing computer technology, computers are becoming more powerful, and correspondingly, the size and the complexity of the problems being solved using optimisation techniques are also increasing.

The purpose of this section is to present the techniques and applications of engineering optimisation in a simple manner and to establish their relevance to tool selection. Optimisation is the act of obtaining the best result under given circumstances. Since the effort required or the benefit desired in any practical situation can be expressed as a function of certain decision variables, optimisation can be defined as the process of finding the conditions (the variable values) that give the maximum or minimum value of a function, in this work known as the cost function. The particular form of the cost function to be used here will be described in more detail in the next chapter.

Optimisation is commonly considered to be applicable to continuous functions but it is in some instances applicable to discrete systems. Although tool selection in considered to be a discrete function process, foundation work in the optimisation of continuous functions gives a valuable insight. Consequently the optimisation of both continuous and discrete functions are introduced here.

It can be seen from Figure 3.3. that if a point x^* corresponds to the minimum value of function f(x), the same point also corresponds to the maximum value of the negative of

the function, -f(x). Thus, without loss of generality, optimisation can be taken to mean minimisation since the maximum of a function can be found by seeking the minimum of the negative of the same function.

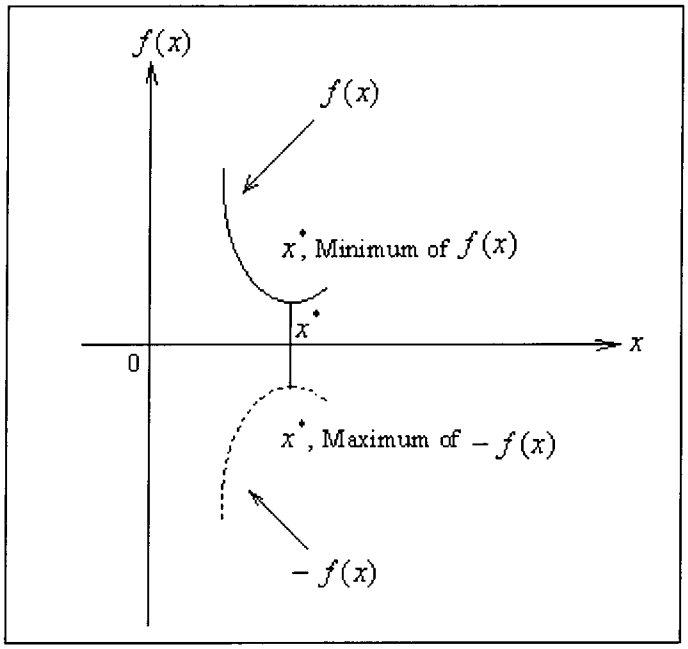


Figure 3.3. Minimum of $f(x)$ is same as maximum of - $f(x)$ [79].

Thus, in this application optimisation can be applied to the minimisation of a cost function that relates to tool selection. There is no single method available for solving all optimisation problems efficiently. Hence a number of optimisation methods have been developed for solving different types of optimisation problem.

Optimum seeking methods are also known as mathematical programming techniques and are generally studied as a part of Operations Research. Below is a list of various mathematical programming techniques together with other well-defined areas of Operations Research [79].

3.3.1. MATHEMATICAL PROGRAMMING TECHNIQUES

Mathematical programming techniques are useful in finding the minimum of a function of several variables under a prescribed set of constraints. An extensive list of techniques are listed below [79]: -

1. Calculus methods
2. Calculus variations
3. Non-linear programming
4. Geometric programming
5. Quadratic programming
6. Linear programming
7. Dynamic programming
8. Integer programming
9. Stochastic programming
10. Separable programming
11. Multi-objective programming
12. Network methods: CPM and PERT
13. Game theory
14. Simulated annealing
15. Genetic algorithms
16. Neural networks

3.3.2. STOCHASTIC PROCESS TECHNIQUES

Stochastic process techniques can be used to analyse problems described by a set of random variables having known probability distributions. Such techniques are listed below:

1. Statistical decision theory
2. Markov processes
3. Queuing theory
4. Renewal theory
5. Simulation methods
6. Reliability theory

3.3.3. STATISTICAL METHODS

Statistical methods enable the analysis of experimental data and building of empirical models to obtain the most accurate representation of the physical situation.

1. Regression analysis
2. Cluster analysis, pattern recognition
3. Design of experiments
4. Discriminate analysis (factor analysis)

The description by Rao [79], gives the above list but does not describe adequately their individual functions or interrelationships.

The list gives a range of mathematical techniques ranging from well established "classical" optimisation methods based upon calculus through to more recently devised methods such as genetic algorithms and neural networks. These more recent methods are usually based upon the computational power available in modern computers. Other methods, such as simulated annealing, have been developed to deal with problems that occur with the more classical approaches.

A further difficulty in optimisation relates to processes that are stochastic, that is, non-deterministic processes i.e., random processes which evolve over time. In order to deal with the optimisation of these processes a number of methods have been developed. Once again the fundamentals of some of these approaches are based upon classical search methods but they include the ability to deal with variation. For example, the Markov theory allows a historical view of changes to be taken into account by giving emphasis on recent events compared with more distant events.

Although these approaches may be used as part of a "real world" exploration or experiment, many of these methods are used in conjunction with simulation. Simulation allows decision scenarios to be explored before any "real world" action is taken.

The list C given by Rao [79], suggests a number of statistical methods applied to experimental or simulation data and as a consequence may be used in conjunction with methods described in the previous two lists. Another classification for optimisation methods by Taylor [80] is more directly useful and differentiates between continuous and discrete functions, as shown in Table 3.1.

	Characteristics						
	Continuous						Discrete
Properties of Independent Variables	Unconstrained					Constrained	
	Univariate		Multivariate				
Properties of Performance Space	Smooth	Non-Smooth	Smooth		Non-Smooth	Smooth/ Non-Smooth	
Optimisation Methods	Univariate minimisation with derivatives	Univariate minimisation without derivatives	Conjugate Gradient	Quasi-Newton	Direct Search	Linear Programming	Combinatorial Optimization
Examples of Minimisation Methods	Newton-Raphson	Brent's Method	Fletcher-Reeves	Fletcher-Powell	Nelder-Mead Method	Simplex Method	Simulated Annealing Methods

Table 3.1. Optimisation method classification based on independent variable properties [80].

In tool selection the objective is the selection of the best set of tools to punch out a given sheet metal part. This is expressed as a cost function of certain decision variables, in this work, mapped tools and features curves. Obviously, there will be more that one possible solution to punch a given part but there may be only one best solution. Hence, the tool selection optimisation problem is actually a problem of searching the space of all possible solutions, in order to arrive at the best one. Searching the space of all possible solutions can best be represented as traversing different paths of a decision tree where the node is the cost after a particular decision has been made, and the line is the decision that has been made. Travelling through any path has it own accumulated cost or penalty, and the final cost of traversing a certain path, in this work, a cost function value, reflects the accumulated cost caused by the impact of the decisions that have been made.

3.4. SPACE SEARCH ALGORITHMS

Traditionally, a number of different space search algorithms have been implemented which can be clustered into two groups. The first group is called "Blind/uninformed" search algorithms. This group contains a number of searching methods, discussed below, such as "Breadth-first search", "Depth-first search", "Depth-Limited search" and "Iterative-deepening search". These methods work with no information about the search space, other than to distinguish the goal state, that is, the completion of the punching of all features. The second group is called "Informed search methods" where the search algorithms have some additional knowledge about the state space that "blind/uninformed" search algorithms do not have. This additional knowledge allows the algorithm to make an estimate of how much further the algorithm must go to reach the goal state. To use this additional knowledge, the search algorithm uses an evaluation function that calculates the "cost" to reach the goal. The second group contains a number of methods, for example, "Best-First Search", "Greedy Search", "A* Search", "Beam Search", "Hill-Climbing Search", "Simulated Annealing Search" and "Genetic Algorithms".

Most of these search strategies are discussed below but Simulated Annealing and Genetic Algorithms are not discussed until the end of Chapter 6 where these approaches can be compared with Monte Carlo methods and the results of simulation can be used to indicate where Simulated Annealing and Genetic Algorithms may have some advantage.

3.4.1. BLIND/UNINFORMED SEARCH METHODS

3.4.1.1. Breadth-first search Method [81, 82]

"Breadth first search" is an exhaustive graph search algorithm where the search goes through the tree level by level, visiting all of the nodes on the top level first, then all the nodes on the second level, and so on. This requires all current paths to be kept in memory simultaneously. This strategy has the benefit of being complete, that is, if there is a solution, it will be found, and the solution will be optimal as long as the shallowest solution is the best solution, which may indeed not be the case in tool selection. However, the breadth-first search achieves its optimum solution by keeping all of the leaf nodes in memory and hence in this application will require a prohibitive amount of memory when searching anything more than a very small tree. The time complexity, [81], of breadth-first search is *O(b^d)* where b is the branching factor, as an example, b will equal 2 for binary trees, and d is the depth of the solution. Johnson, K. [83], see Table 3.2., estimated, using branching factor b=10, how huge the time and memory requirements can be using a breadth first search method. It was assumed that 1000 nodes can be goal-checked and expanded per second and that a node needs 100 bytes of storage.

Depth	Nodes	Time	Memory
0	1	1 millisecond	100 byes
2	111	1 second	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

Assume:

- **branching factor (b) = 10**
- **1000 nodes can be goal-checked and expanded per second**
- **A node needs 100 bytes of storage**

Table 3.2. Breadth-first search method space/time requirements.[83].

Thus, the Breadth first search might prove useful in any application where connectivity is important, for example, if there is a need:

1. to perform some function on all the nodes connected to a given node,
2. to determine if there is a path from one vertex to another
3. to implement some sort of network
4. to make sure that every vertex could be reached from every other vertex.

Therefore, for all the above reasons, breadth first search would not be adequate for the tool selection optimisation problem. It would involve the search of an explosive decision tree with unpredictable branching and depth, which would require a long time and a massive computer capacity.

3.4.1.2. Depth-First Search Method [81, 84, 85]

Depth-first search, an exhaustive graph search algorithm, goes through the tree branch by branch, going all the way down to the leaf nodes at the bottom of the tree before backtracking to the last decision point and trying the next alternative path over. This

strategy requires much less memory than breadth-first search, since it only needs to store a single path from the root of the tree down to the leaf node in order to find a solution. However, it is potentially non optimal, for example if there is a solution at the fourth level in the first branch tried, and a solution at the second level in the next branch over, the solution at the fourth level will be returned. The time complexity of depth-first search is *O(b^m)* where b is the branching factor and m is the maximum depth of the tree. Its space complexity is only *b*m*. However, if a depth first search is to be used for tool selection i.e., to find the best solution, that is the global optimum it will then involve searching the whole space and it would thus have the same time complexity as a breadth first search.

Thus, for the reasons given above both the breadth first search and the depth first search would not be adequate for the tool selection optimisation problem. The fundamental problem is that in any practical case the problem would involve the search of an explosive decision tree of unpredictable branching and depth.

3.4.1.3. Depth-Limited Search Method [81]

A depth-limited search essentially does a depth-first search with a predetermined imposed cut-off at a specified depth limit. When the search hits a node at that depth, it stops going down that branch and moves over to the next branch. This avoids the potential problem with depth-first search of going down one branch indefinitely, especially when there may be solutions at shallower depths.

The time complexity of depth-first search is O(b^l) where b is the branching factor and l is the depth limit. Its space complexity is only b*l. However, a depth-limited search is incomplete, so if there is a solution, but only at a level deeper than the limit, it will not

be found. Therefore, it is probably not adequate for the tool selection optimisation problem.

3.4.1.4. Iterative-Deepening Search Method [81, 86]

When searching for a path through a graph from a given initial node to a solution node with some desired property, a depth-first search may never find a solution if it enters a cycle in the graph. This problem is not likely to happen in this work since a tool selection problem would only rarely contain cycles. However, an explicit check for cycles can be added. If a cycle is detected it is necessary to carry out an iterative deepening search which consists of repeated depth-limited searches at different depth limits.

The time complexity of an iterative deepening search is *O(b^d)* where b is the branching factor and d is the depth of the solution. The space complexity is *O(bd)*. Once again an iterative deepening searching strategy would require visiting all the nodes in order to achieve the optimal solution. Hence, it is rather an expensive and exhaustive searching method which may still prove inadequate for tool selection.

It can be concluded from the above discussion that "Blind" search methods are not an adequate approach if the aim is to search for the optimal cost function minima, as they will involve an exhaustive search of a large solution space, which is probably prohibitively expensive.

3.4.2. INFORMED SEARCH METHODS

The searching methods belonging to this group use an evaluation function to estimate how much further the search must go to reach the goal. Such approaches seem to be more "intelligent" in their search for the optimal solution. These types of searching

methods seemed to be attractive for the problem in hand and therefore a number of these searching methods were considered and will be discussed here.

3.4.2.1. Best-First Search Method [87, 88, 89]

The Best-First search is a graph search algorithm which optimises a breadth first search by ordering all current paths according to some heuristic. Hence, the best possible node is always expanded first. The heuristic attempts to predict how close the end of a path is to a solution. Therefore, it must incorporate some estimate of the cost of the path from the present state (node) to the goal state (node). So, in deciding which tool edge should be selected for cutting a particular feature edge, all possible tool edges would be ranked based on, for example, their length, so the longest tool edge would be preferred. Such a search mechanism would initially appear to be very promising but it does not guarantee that the optimal solution will be found. On the contrary, such a mechanism might decide on not expanding a node, tool edge, which would eventually prove to be the best. For example, if it is required to punch a rectangular hole of 100 x 60 by one of two tools of either 80 x 20 or 50 x 30, the best-first search method, based upon the above rule, would assign the tool edge 80 to cut the feature edge 100. Such a decision would result in 6 blows being needed to finish the feature. However, by inspection, selecting the tool edge 50 would involve only four blows. This problem gets more complex when tool set up due to the limited capacity of the turret is taken into account. In this case selecting a second best tool edge option might well result in saving an expensive tool change.

3.4.2.2. Beam Search Method [90]

Beam search is like the breadth-first search in that it progresses level by level. Unlike the breadth-first search, beam search expands several partial paths and purges the rest. However, a beam search moves downward only through the best nodes at each level; the other nodes at that level are ignored. Beam search is considered as an "optimisation"

of the best first search graph search algorithm where a predetermined number of paths are kept as candidates. The number of paths is the "width of the beam". If more paths than this are generated, the worst paths are discarded. This reduces the space requirements of best first search. It is most useful when the local optima are not too common. The "genetic algorithm" is a variant of the beam search [91] because it expands several partial paths and purges the rest.

Once again the Beam search might be "fooled" by its heuristic decision and may not explore a potentially good branch. For example, using the previous example of a rectangular feature 100 x 60, a beam search would expand the tool edge 80 branch and purge the tool edge 50 branch although, as indicated earlier, the 50 edge would lead to a better solution. Furthermore, the fact that the beam search is not useful when local optima are common makes it an unreliable method specially if a complete understanding of the space to be searched is not available.

3.4.2.3. Hill-Climbing Search Method [92]

The name "Hill climbing" comes from the idea of trying to find the direction to the top of a hill from a current position. In essence, hill climbing search methods, sometimes known as gradient methods, find an optimum by following the local gradient of the function. It is worth mentioning here, that hill climbing methods assume that the problem space being searched is continuous in nature, and that derivatives of the function representing the problem space exist. This is not true of many real world problems such as tool selection where the problem space is discrete and discontinuous. Hill climbing search works well, fast and takes little memory, if an accurate heuristic measure is available in the domain, and if there are no local minima/maxima. However, a major disadvantage of using hill climbing is that hill climbing algorithms only find the local optimum in the neighbourhood of the current point. They have no way of looking

at the global picture. Even, parallel methods of hill climbing, used to search multiple points in the problem space, would still suffer from the problem that there is no guarantee of finding the optimum value, especially in very noisy spaces with a multitude of local peaks or troughs.

To conclude, all informed search methods rely strongly, in one way or another, on heuristic based mechanisms of defining the best next move. Such a mechanism forms a clear weakness in such methods because of the very basic fact that no matter what rules are devised there will be exception to those rules and the more effort made to tighten the rules the more complex the approach will become. This led the present work to investigate different approaches that can be utilised for such optimisation problems and coincidentally establish a "rule free" baseline or benchmark for the assessment of search strategies.

3.4.3. MONTE CARLO METHOD

Although the Monte Carlo method has not been listed by Rao [79] or Taylor [80], it was considered to be a useful approach for tool selection for reasons that will be outlined and discussed below.

The name of the Monte Carlo method is inspired by the gambling casinos at the city of Monte Carlo in Monaco. The Monte Carlo method involves an interesting combination of sampling theory and numerical analysis [93]. The mathematical techniques used by this method are in fact based on the selection of random numbers [94]. Monte Carlo methods can be used to simulate the behaviour of a physical or mathematical system. They are distinguished from other simulation methods by being stochastic [95], that is,

non-deterministic, in some manner. This stochastic behaviour in Monte Carlo methods generally results from and is instigated by the use of random number sequences.

In principle, the Monte Carlo (MC) method can be considered as a very general mathematical tool for the solution of a great variety of problems. An important feature of all MC techniques is that more precise results can be obtained by generating a larger number of points [94, 96]. Using the casino analogy, if the stake to gamble is negligibly small it pays to keep betting until you hit the jackpot. In the case of tool selection the gambling stake is inexpensive computing time and the jackpot is a significant reduction in production cost. Being based on random numbers, the results obtained with a MC procedure are never exact, but rigorous in a statistical sense i.e., the exact results lie in given intervals with given probabilities [94]. The uncertainty of obtaining good results is strictly related to the variance of the possible outcomes, that is uncertainty is smaller if the size of the sample is larger [94, 96, 97]. It is then clear that (i) a great effort has been devoted by MC experts to devise variance-reducing techniques and (ii) the enormous development of MC applications that occurred in the last decades is due to, and has been pushed because of the even more extraordinary development of electronic computers.

The applications of MC methods can be divided into two major groups. One consists of direct simulation of systems that are already statistical in their nature; in such cases it is not even necessary to have a well defined mathematical equations to describe the behaviour of the system. The second group consists of MC methods devised for the solution of well defined mathematical equations. In such cases the methods are used to solve the equations that describe the problem of interest. In the former case, the simulation may yield more information than that obtainable from the solution of the equations. On the other hand, the direct simulation of a statistical problem is at times

very inefficient [94]. If a direct simulation is performed, a large amount of computation is devoted to the "uninteresting" normal situations and very little to the rare occurrences of interest. In this work, for example, direct simulation is used for searching the space of all possible solutions for a tool selection optimisation problem. Obviously, the interest is not in those inefficient solutions which are frequently found. The interest is in those rarely found ones i.e., the global optimum solution, hence, it is necessary to distort the simulation by applying more sophisticated/intelligent MC techniques that reduce the variance of the quantity of interest, giving up the advantages offered by direct simulation.

Thus, MC calculations can be considered as simulated experiments. As in real experiments, the final results of the simulation are not the end of the story: they must be interpreted in physical terms in order to obtain a better understanding of the problem at hand [94]. In this respect MC is again a useful tool and the MC method is increasingly becoming a much used modelling tool.

It was decided to explore the Monte Carlo method approach for the following reasons:

1. The tool selection optimisation problem is of a statistical nature, that is, with a different tool being selected randomly a different route toward an optimal solution is dictated.
2. The difficulty of defining the mathematical equations that would describe the problem.
3. For a tool selection optimisation problem, there are a large number of sub optimal solutions and the aim is to find the global or a near global optimum solution. To achieve the global optimisation, one needs to cover the whole of the solution space.

One appealing way to do so is to use a random walk which in this instance takes the form of a Monte Carlo approach [97].

4. Directly simulating the process of tool selection will give a better understanding of the problem than just solving mathematical equations to find a sub or even global minima.

The next chapter will introduce the Monte Carlo theory and describe the approach taken for tool selection.

CHAPTER 4

MONTE CARLO METHODS

So what does a computer simulation technique have in common with the world famous Monte Carlo casino, in the second smallest country in the world, the principality of Monaco? - The element of chance is used in both, so that the desired result occurs in the long run. The random nature of a game of chance is designed so that the owners of the casino can be assured, that in the long run, the casino will make a profit while the individual gambler has a reasonable chance of winning. The random nature of a Monte Carlo simulation is designed so that the programmer can be assured that in the long run, the simulation will approach equilibrium values, while an individual move has a realistic chance of taking the simulation away from equilibrium.

Selecting the best punching tools for a particular sheet metal part represents a complex optimisation problem. In the previous chapter a number of optimisation techniques have been discussed. A major drawback of these techniques is their reliance on a heuristic-based method in their convergence of the search for the optimal solution. For example, hill climbing methods follow the local gradient of the function, which might lead to a local optimum, in the neighbourhood of the current point, but may not lead to the global optimum.

In the search for an optimal set of tools, one of several alternatives can be taken. The first of which is the physical approach, where a selected set of tools is tried on the shop floor and by a process of trial and error the tool setter will seek improvement. Such an approach might seem adequate if the same product is to be produced for a long period of time. However, such an approach is highly inefficient, specially at the design level where the designer needs an indication of the impact of the proposed design on the manufacturing process. Also, product life cycles are getting shorter and shorter which

would require products to be "right first time" in order to compete in today's highly competitive market. Therefore, a physical approach would seem inefficient. An alternative option is not to seek the optimal solution and to be satisfied with a reasonably good solution. Arguably, most of the available systems come under such a classification, where restriction on the definition of a candidate tool is imposed early in the process of the search for a set of punching tools. Such approaches do not guarantee the optimum solution for the reason that these approaches do not have a global picture of the search space.

A third alternative is to seek and find the optimal solution i.e. the global minima. These approaches tend to mathematically model the system in hand, trying to find the best parameters to achieve the best solution. In previous chapters, earlier work and enabling theory has led to the idea of modelling the tool selection process. It is commonly believed that the more detail such a model has the better it resembles reality. However, there is no guarantee that the time and effort devoted to modelling will return "useful" and therefore satisfactory results, that is, that the model will lead to the selection of an optimal set of tools. Conversely, there is a tendency for a modeller to treat a particular description of a problem as the best representation of reality. Further, if the conditions under which the model was designed, change, then the model must be modified, therefore, control over the solution must be established. Thus, the addition of more details could make the solution more difficult to be solved analytically which then may change the proposed method for solving the problem in hand from being an analytical method to a numerical method.

In this work a different approach has been adopted in which the interactions of the components of the complex system are directly simulated. Although simulation, in the

past, was often viewed as a "method of last resort", recent advances in computer technology has made simulation a more widely used tool. Simulation permits considerable modelling freedom so that a model may closely correspond to the system being studied i.e., mimicking the interactions of the components of the complex systems understudy. Hence, simulation is an extremely useful tool in situations where analytical solutions are restricted or otherwise inappropriate. It is cheap, quick and easy to use. However, simulation provides only statistical estimates rather than exact results, [98] as it builds its analysis on a sample taken from the population of all possible solutions i.e., the "search space", and it only compares alternatives rather than generating the true, global optimum solution. Most searching techniques discussed in the previous chapter, however, also compare alternatives rather than generate the true global optimum solution.

As mentioned earlier, simulation in its technical sense, involves using a model to produce results, [97]. If the model has a stochastic element, that is, it is non-deterministic [95], it is then called stochastic simulation [97], or a statistical sampling experiment [98]. Such experiments involve observing a random phenomenon [97]. Due to the fact that sampling from a particular distribution involves the use of random numbers, stochastic simulation is sometimes called "Monte Carlo" simulation [98]. Here, it is important to emphasise that the observations in the Monte Carlo methods, as a rule, are independent [98], that is, each new observation or derived solution is not a dependant of the previous solutions, i.e. each new derived solution resembles a totally random walk in the "search space".

Most of the available searching and optimisation techniques have, as mentioned earlier, the main drawback of being heuristic based in their search for the optimal solution.

However, Monte Carlo methods represent a "rule- free" searching method. In this way the Monte Carlo approach can be used to establish a base line i.e., to establish the adequacy of other methods for the selection of the best set of punching tools. Also, to gain a better understanding of the nature of the complex interactions of the different components involved i.e., mapped tools and feature edges. The introduction of a more rational, heuristic or rule based search, if found necessary, should then achieve a better solution than the Monte Carlo method or should achieve the optimum solution in a shorter time. Hence, it has been decided on investigating the way that Monte Carlo can be used in solving the undertaken optimisation problem.

4.1. MONTE CARLO METHODS

For the sake of completeness, it has been decided to provide the reader with a basic introduction to Monte Carlo methods. The term "Monte Carlo" refers to a group of methods, which use random or pseudo random numbers for the solution of physical or mathematical problems [98, 95, 99]. Such methods avoid the need to generate equations that describe the behaviour of the system [100]. Monte Carlo methods are used in different fields, from economics [101] to nuclear physics [98] and even for regulating the flow of traffic [102].

According to the electronic book of the "Computational Science Education Project – sponsored by U.S. Department of Energy", [100], the primary components of a Monte Carlo simulation method include the following seven elements:

1. Probability distribution functions
2. Random number generator
3. Sampling rule
4. Scoring or tallying
5. Error estimation
6. Variance reduction techniques
7. Parallelisation and vectorisation algorithms

In the following sections the seven elements mentioned above will be discussed. However, it should be remembered that the way in which Monte Carlo methods are applied varies from field to field and thus each section will be related to the tool selection problem. However, strictly speaking, in order to call something a Monte Carlo experiment, all that is needed is the use of random numbers to examine the problem [103].

4.1.1. PROBABILITY DISTRIBUTION

As mentioned earlier, in many applications of Monte Carlo methods the physical process is simulated directly and there is no need to generate equations that describe the behaviour of the system. The only requirement is that the physical or mathematical system is described by one or more probability density functions, which may even have their origins in experimental data or in a theoretical model which describes the process [100, 104]. A probability distribution is a mathematical function which describes the probabilities of possible events in a sample space [104]. There are two major types of probability distributions; the first of which are the "Discrete Probability" distributions and the second are the "Continuous Probability" distributions. In order to decide on the most adequate probability distribution that can best describe the physical system of matching tools and features edges, a brief discussion of some of the characteristics related to these probability distributions types was found necessary.

4.1.1.1. Discrete Probability Distribution

In discrete probability distribution types, there are a finite number of events in the sample space, and a positive probability applies to each. For example, if two dice are rolled, the sample space of possible events (results of the dice toss) are 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12, see Figure 4.1. Hence, dice toss is an example of a discrete probability distribution.

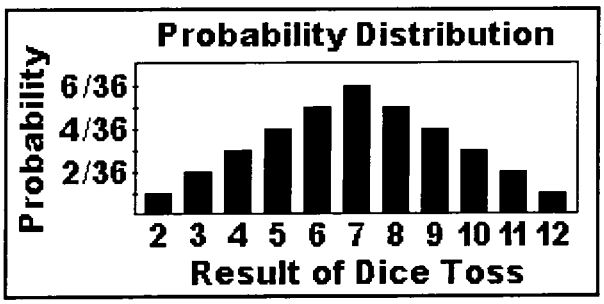


Figure 4.1. An example of a discrete probability distribution [104].

4.1.1.2. Continuous Probability Distributions

A continuous probability distribution is one that has an infinite sample space. The normal distribution is an example of continuous distributions, see Figure 4.2. In this case probabilities are not assigned to specific events in the sample space. Instead, they are assigned to subsets of the sample space [104]. For example, the height of a randomly selected person is a continuously distributed random variable. Hence, it does not make sense to ask what the probability is that an individual's height will be exactly 1.645 meters. That, after all, is just one possibility out of an infinite number of possible heights. However, it makes more sense to ask the question, "What is the probability that an individual's height will fall between 1.60 and 1.70 meters?". Accordingly, continuous distributions are described by probability density functions [104]. For example, the normal distribution is described by:

$$Y = \frac{N}{\sigma\sqrt{2\pi}} e^{-(X-\mu)^2/2\sigma^2}$$

Where,

Y = Frequency of a given value of X,

X = Any score on the distribution,

μ = Mean of the distribution,

σ = Standard Deviation of the distribution,

N = Total frequency of the distribution,

π = A constant of 3.1416, and

e = A constant of 2.7183.

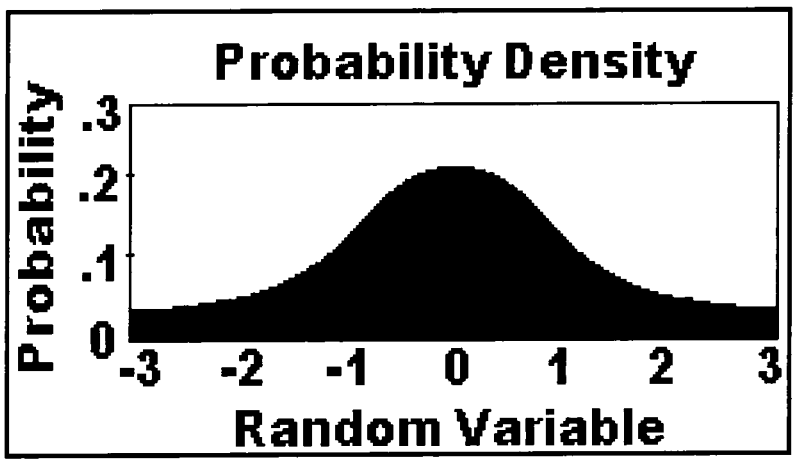


Figure 4.2. Normal distribution; a continuous probability distribution [104].

4.1.1.3. Partly Discrete and Partly Continuous Probability Distributions

A third type of probability distribution is a mixed distribution. These are distributions which are partly discrete and partly continuous. For example [104], rainfall in a given day has a mixed distribution. There is a positive probability that no rain will fall in any day. This is the discrete component of the distribution. If rain does fall however, the amount that actually falls is continuously distributed, see Figure 4.3.

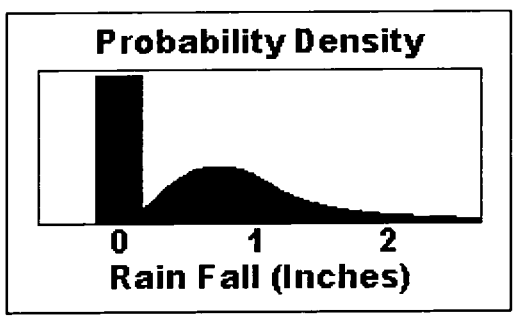


Figure 4.3. An example of a mixed probability distribution [104].

In the case of the simulation of tool selection the probability distribution of possible events in a sample space is a discrete probability distribution as there are a finite number of solutions in the sample space i.e., similar to the above example of rolling two

dice. For a large feature and tool population these observations or results are close to each other and consequently it has been assumed in this work that approximating this discrete probability distribution to a continuous probability distribution is statistically valid. This assumption can be seen to be reasonable by considering, for example, a tool magazine capacity of, say, 40 tools punching, say, 20 features on a part. The total distribution could contain a space of 120^{80} choices.

4.1.2. RANDOM NUMBER GENERATOR

A random number generator is a deterministic algorithm that produces numbers with certain distribution properties, that is, these numbers should behave similar to realisations of independent random variables. Random variables are useful because they allow the quantification of random processes, they also facilitate the definition of a "Mean" and a "Standard Deviation". For example [100], if one were drawing balls of different colours from a bowl, then, it would be difficult to envisage an average colour. However, if numbers were assigned to the different coloured balls, then an average could be computed. In the same way, in tool selection it is difficult to quantify the process of randomly selecting a tool edge to punch out a randomly selected feature edge. It would be difficult to envisage an average tool-feature match. In a tool selection problem, a tool feature match corresponds to one blow. Hence, if numbers are assigned to different tools and feature edges, then a total number of blows can be computed. Now that a "random-value" output, in this work, total number of blows, an average value of the random outputs over the possible events, that is, simulation trials, can be defined. Thereafter, over a number of simulation runs a "Mean" and "Standard Deviation" of the total number of blows, the cost function value, could be computed. This average value is called the "Expectation Value" or "Mean" for the cost function. The average of the square of the cost function leads to an important quantity, the "Variance". Further, the square root of the variance is the "Standard Deviation".

4.1.3. SAMPLING RULE

In essence, the goal of the Monte Carlo method is to replace the physics and mathematics by random sampling of possible solutions. Samples are then used as a representation of a larger group. In fact, the main interest lies in this larger group, which is called the target population [105]. In this work, the target population is the search space, that is, all different possible values of the cost function i.e., all possible solutions. Therefore, a sample is an informative, abbreviated way of describing the population under study, without having to list all of the individual observations, which may be practically impossible. The use of a sample has a clear advantage in the proposed use of the Monte Carlo method for searching a huge unpredictable explosive search space where it may be difficult or impossible to list all possible solutions.

The main rule for a statistical sample is that it should not lead to a conclusion that does not represent the target population. Therefore, the main point of concern is an adequate sample size that would fairly represent the targeted population. This point will be discussed in Chapter 6 where the law of large number and other relevant background will be introduced.

4.1.4. SCORING (OR TALLYING)

The outcomes of the sampling process must be accumulated into overall scores for the quantities of interest. In this work the score is the Cost Function Value. The Cost Function Value is described in more detail later in section 4.2.6. Thus in this application of Monte Carlo the score is the summation of all of the components of cost, appropriately weighted, resulting from each decision in a successful decision tree navigation. It should be emphasised that the intermediate scores have little value as they may or may not achieve a successful outcome and are unreliable for indicating the likelihood of achieving an optimal solution, as will be illustrated later.

4.1.5. ERROR ESTIMATION

Statistical sampling is a very useful means for the representation of a larger population of interest but false or manipulated data gathering can affect the conclusion in an adverse manner. In this work, it might lead to a wrong estimation of the target value, in other words, it might not lead to the global optimum. Thus, it is very important to know how far away the "sample mean" is from the "true mean" of the population. An estimate of the statistical error (variance) as a function of the number of trials and other quantities must be determined. Once again, this is discussed in more detail in Chapter 6.

4.1.6. VARIANCE REDUCTION TECHNIQUES

The "variance" of a set of scores is simply the square of the "Standard Deviation", [106, 107], and this measures the spread by looking at how far the observations are from their mean. Therefore, "Variance" and "Standard Deviation" will be large if the observations are widely spread about their "Mean", and small if the observations are all close to the "Mean". The law of large numbers [107], the foundation of such business enterprises as gambling casinos, explains that the larger the number of simulation trials made, the less the error of the outcome observations will be. That is, the "Sample Mean" gets closer to the "Population Mean" as more simulation trials are made. However, the law of large numbers does not say how many trials are needed to guarantee a sample mean close to the population mean. That depends on the variability of the random outcomes. The more variable the outcomes, the more trials are needed to ensure that the "Sample Mean" is close to the "Population Mean". Hence, larger samples have a clear advantage because they are much more likely to have a "Sample Mean" close to the true value of the "Population Mean", because there is much less variability among large samples than among small samples [107, 108]. This assumes that the statistical estimation, in this work the minimum cost function value, will be accurate if enough observations i.e., long computational time, can be afforded.

"Variance Reduction Techniques" are methods used to reduce the "Variance" of the estimated solution which would then reduce the computational time for the Monte Carlo simulation to find the global optimum i.e. the selection of the best set of tools with the minimum cost function value. This will be discussed in more details in Chapter 6.

4.1.7. PARALLELISATION AND VECTORISATION

Parallelisation and vectorisation are computer algorithms concerned with issues related to memory allocation and coding architecture, which would allow the Monte Carlo methods to be implemented efficiently and quickly using advanced computing. However, these algorithms are beyond the scope of this work as the aim of this work is not to launch commercial software but to explore and establish the base line of the adequacy of the use of Monte Carlo methods for the optimisation search of an explosive decision tree. The following sections discuss how Monte Carlo simulation methods have been used in the current application, that is, the selection of the "best" set of tools for punching and/or nibbling any given two dimensional sheet metal part.

4.2. THE IMPLEMENTATION OF MONTE CARLO METHODS IN TOOL SELECTION

As has been mentioned earlier, the way Monte Carlo methods are implemented differ from one application to another. This section shows the way Monte Carlo simulation methods are used in this work to search explosive decision trees for the global optimum. In general, any given sheet metal part may require holes of different shapes and sizes. However, optimising the part production is different from optimising individual hole production, especially if the tool rack capacity is limited [49].

Monte Carlo methods can start by use of a random sample from a population of all possible solutions in order to have an abbreviated informative, global picture of the

larger population of interest. The author [4, 5, 6, 7] has already published several papers on this topic but for completeness a description of the Monte Carlo approach is given here. The approach taken to solve the tool selection optimisation problem is to randomly select a tool to punch a randomly selected feature. Then, a tool edge is randomly selected and its shape is matched and mapped with a randomly selected feature edge. Providing that such a match does not cause any tool-feature interference the tool selection is accepted. Here, "no tool-feature interference" means the tool does not remove more material than is required. Another tool is then randomly selected for another randomly selected feature and so on until the simulation program indicates that all the features of the part are completely "punched out". Carrying out such a simulation will result in the selection of a set of tools for producing the whole part. Running this simulation task a number of times will result in the generation of a large number of different sets of tools. However, most of these solutions will be unacceptably inefficient. For this purpose, a cost-function value has been introduced to evaluate each solution and the set of tools with the minimum cost value is the one selected for punching the part in hand. Subsequently, an interesting and obvious question arose; "how many simulation runs are required to derive an acceptable set of tools?"

Monte Carlo methods are based upon the assumption that the results from a simulation with random inputs will be of a known distribution, which in the case of this work is assumed to be a Normal Distribution. In this application it is the values of the cost function that are considered to be distributed normally. Normal Distributions [107] are usually good approximations to the results of many kinds of chance outcomes, such as tossing a coin many times. Hence, in the case of this work, a Normal Distribution at this point is assumed to be statistically acceptable. In practice it is recognised that the distribution will not be normal and will only approach normality where a large number

of features and tools are present. The problem of the influence of deviation from the assumption of normality is dealt with in Chapter 6. The next sections discuss the simulation system and its elemental components in more detail.

4.2.1. DESCRIPTION OF THE SIMULATION SYSTEM

The diagram in Figure 4.4. shows a flow diagram of the proposed simulation system. The basic principle is that a random tool is selected for punching a randomly selected feature. One tool curve is then randomly selected and matched to a randomly selected feature. Providing that such a match does not cause any tool-feature interference, the tool is accepted and a blow is added to the blow count. Another tool is then randomly selected for another randomly selected feature and so on until the part is completely "punched out". The system keeps a record of all tools used and each successful tool feature match represents a punch blow. To keep a record of features that still require punching becomes complicated because of the allowed overlapping of punching operations and the fact that a single blow may fortuitously cut two or more feature curves at one time. Once an acceptable set of tools has been found, the repetition of the whole process can produce a number of different solutions. Each proposed solution can be evaluated, in this case, by a cost function. Plotting the obtained cost function values against their frequencies of occurrence gives the sample distribution curve, assumed to correspond to a normal distribution curve, that best represents an informative abbreviated description of the larger population of interest i.e., the search space.

In the next subsection the necessary geometrical aspects of the proposed simulation system are discussed.

4.2.2. FEATURE AND TOOL DESCRIPTION

This section shows the data structure used in the proposed simulation approach. In the sheet metal industry today, the use of Computer-Aided Drafting (CAD) packages is very common. In this work, the format of AutoCAD, a popular and easily available CAD software package, has been adopted. In AutoCAD, design data is stored in the form of a list, which contains entities (straight lines and/or circular curves). For example, a square will be stored as four separate straight lines. A straight line is defined by the co-ordinates of its starting (x_1, y_1) and ending (x_2, y_2) points while a circular curve is defined by the co-ordinates of the centre point (x_0, y_0), the radius of curvature (R) and the starting (A_S) and finishing (A_F) angles from the centre point. If the entity is a circle, it will be stored by the co-ordinates of the centre point and the radius of the circle. This structure for representing tool or feature edges can be applied directly in the determination of curve signatures as described in the following section.

4.2.3. ASSIGNING A CURVE SIGNATURE FOR EACH CURVE ENTITY

A curve signature, similar to the shape-index-set used by Cho and Lee, [58], is used in this work to uniquely describe a tool or a feature edge. These curve signatures are composed of four components; they are entity length, entity radius of curvature, angle formed with preceding entity and angle formed with proceeding entity. This section will discuss the computation of each of these components.

4.2.3.1. Length of the entity

As discussed earlier an entity can be either a straight line, circular arc, or circle. In the following sub-sections each entity will be discussed separately.

4.2.3.1.1. Straight line entities

To calculate the length of a straight line entity, the distance between the two points representing the starting and finishing points of the entity must be determined. Subsequently, the following equation can be used;

$$l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where l = length of a straight line

x_1 = X co-ordinate of straight line starting point

x_2 = X co-ordinate of straight line finishing point

y_1 = Y co-ordinate of straight line starting point

y_2 = Y co-ordinate of straight line finishing point

4.2.3.1.2. Circular Arc entities

The length of a circular arc is computed using the following equation.

$$l = radius * \theta(radians) = r * \theta^o \left(\frac{\pi}{180^o}\right),$$

where l = length of a circular arc

θ = angle contained within the curve

r = Radius of curvature

π = 3.142857

The computation of the length of the arc is useful when calculating the number of blows required to carry out a nibbling operation or to cut a circular curve using a circular tool.

As previously explained AutoCAD stores a circular arc as a centre point, radius, starting angle and ending angle. It is important to know the co-ordinates of the starting and ending points in order to know the exact position of the arc in relation to other adjacent

entities. This is also useful when calculating the angles of the preceding and proceeding entities as will be shown in a later section.

The starting and ending points of a circular arc can be computed using the following equations:

$$X = x_0 + r * \cos(\theta)$$

$$Y = y_0 + r * \sin(\theta)$$

where X = X co-ordinate of point in question

Y = Y co-ordinate of point in question

θ = angle value of point in question

x_0 = X co-ordinate of curve centre point

y_0 = Y co-ordinate of curve centre point

4.2.3.1.3. Circle entities

Since circles present a closed loop boundary, it is not necessary to define the position of the circle in relation to its adjacent entities, that is, there is no angle between the circle and a preceding or a proceeding entity. Thus, for the circle, it is simply a matter of calculating the circumference. Here, the difference between the starting and finishing angle within a circle is 360^o.

Therefore, the circumference of a circle = $r * 360^o * \frac{\pi}{180^o} = 2\pi r$.

4.2.3.2. The angle between entities

This element of the curve signature is very important, as the angle between two entities uniquely represents the two dimensional spatial relationship between these two entities. The angle θ between the two entities should be measured from the positive axis, in an

anticlockwise direction if θ is positive, and in a clockwise direction if θ is negative [109]. In the following subsections further details regarding the calculation of angles between two entities are given.

4.2.3.2.1. The angle between two straight line entities

Calculating the angle between any two straight lines may be done in more than one way. In this work vector rotation mathematics has been used for this purpose.

To calculate the angle between two straight lines, it is assumed that each line is a vector, thus, by back rotating both vectors to bring the nearest on to the positive X-axis, the angle between the two lines can be calculated. A full description of these calculations is given in Appendix A.

It was found that by calculating only the sin or the cosine of an angle it was not possible to obtain the correct value for the angle, for example, the sin of both 30 and 150 equals 0.5. Therefore, by calculating both the sin and cosine values for the angle in question, the true value of the angle can be obtained. In this work the programming language, Visual Basic for Microsoft Excel, supports only the inverse tan function, therefore, the tan value is calculated by dividing the absolute value of sin by the absolute value of cosine. After determining the correct quadrant the correct value of the angle can be obtained.

4.2.3.2.2. The angle between two circular curve entities

The angle θ between two curves, which intersect at a point, (x_P, y_P) is defined as the angle between the tangent lines to the curves at (x_P, y_P). This can be calculated using the formula:

$\tan\theta = \frac{m_2 - m_1}{1 + m_1 m_2}$, once the slopes of the tangent lines are known.

where, θ = angle between two curves

m_1 = slope of first curve tangent

m_2 = slope of second curve tangent

A line is tangential to a curve if it touches the curve at exactly one point. The question that arises here is how to calculate the slope of the tangent at a specific point on the circle.

Given the circle described by:

$(x-a)^2 + (y-b)^2 = R^2$

where a = X-co-ordinates of the centre point of the circle

b = Y-co-ordinates of the centre point of the circle

R = Radius of the circle

and the point described by $P(x_P, y_P)$ on the circle, then the equation of the tangent to the circle through P is given by $(x_P - a)(x-a) + (y_P - b)(y-b) = R^2$. Further, using the tangent equation the slope of the tangent can be calculated.

4.2.3.2.3. The angle between a straight line entity and a circular curve entity

The angle θ between a straight line and a circular curve which intersect at a point (x_P, y_P) can be defined as the angle between the straight line and the line tangential to the curve at (x_P, y_P). The calculations used for determining the angle between two circular curves can still be applied here, however, it should be realised that the straight line used to calculate the slope uses the start and end points as co-ordinates.

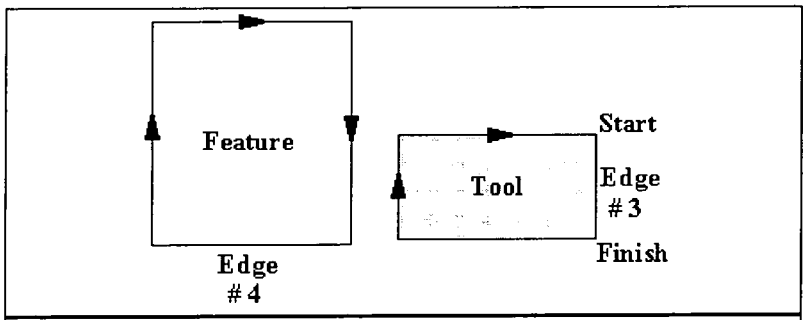
In this work, the simulation program has been restricted to angles between straight lines. The aim is to demonstrate the feasibility of the approach rather than writing complete comprehensive code. However, it is clear that calculating the angle between a straight line and a circular curve is possible.

4.2.4. TOOL-FEATURE CURVE MATCHING

The location and orientation of the tool for punching are computed by matching of the tool and feature curves using simple vector operations. Tool curves are aligned with the corresponding matching curves of the sheet boundary as follows:

1. Move the tool and bring the starting point of the selected random tool edge to point (0,0).
2. Work out the required tool rotation angle for the tool/feature mapping.
3. Move the tool, after rotation, to bring the starting point of the selected random tool edge to the starting point of the random selected feature edge.

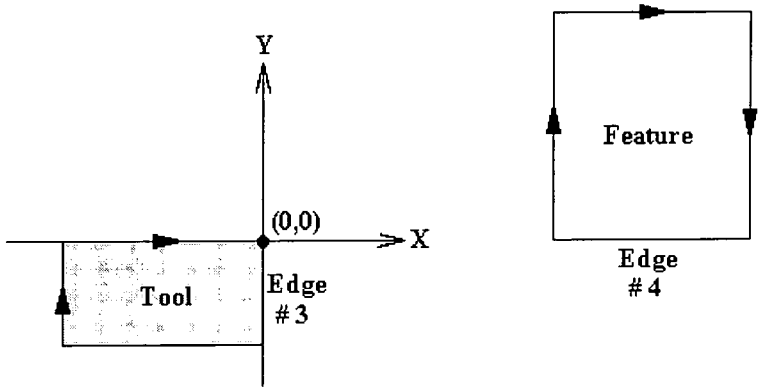
Note that this will work for both internal and external feature edges.



Example: Tool Edge #3 is randomly matched with Feature Edge # 4

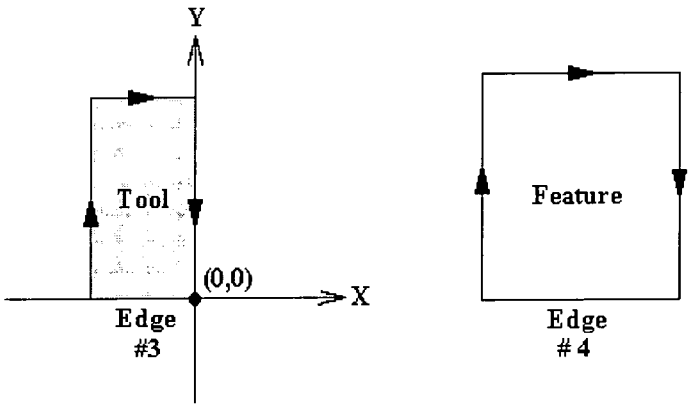
Step 1

Move the tool and bring the starting point of the selected random tool edge to point (0,0)



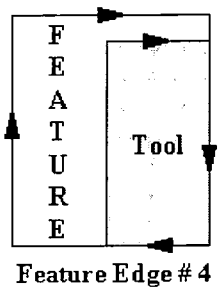
Step 2

Work out the required tool rotation angle for the tool/feature mapping



Step 3

Move the tool, after rotation,
to bring the starting point of the selected random
tool edge to the starting point of the random selected feature edge.



Feature Edge # 4

4.2.5. TOOL-FEATURE COLLISION TEST

This is one of the most important modules in the program. The main function of this module is to ensure that the selected tool does not remove more of the workpiece material than required. From the previous literature, it can be seen that different approaches to this tool-feature clash problem have been taken. However, this matter is purely a geometrical issue but needs to be resolved in order to determine whether the tool has cut within of the required boundary of the feature and is hence not cutting into the required part. While mapping the tool to the feature and defining the intersecting points between the tool and feature edges, the interference test can be carried out simultaneously in order to save computation time. Such an approach was found to be efficient and is described here in more detail. Since the computer program that is operational at this stage focused on straight lines, the description of the algorithm will be restricted to straight lines only. However, this approach can be easily extended to suit non straight-line entities.

A point on a straight line can be parametrically described as $t(x_1, y_1) + (1-t)(x_2, y_2)$, see Figure 4.5. So if $t = 0$ then the point is (x_2, y_2) the end point of the straight line. If $t = 1$ then the point is (x_1, y_1) the start point of the straight line. Now, consider, for example, two lines, see Figure 4.6.

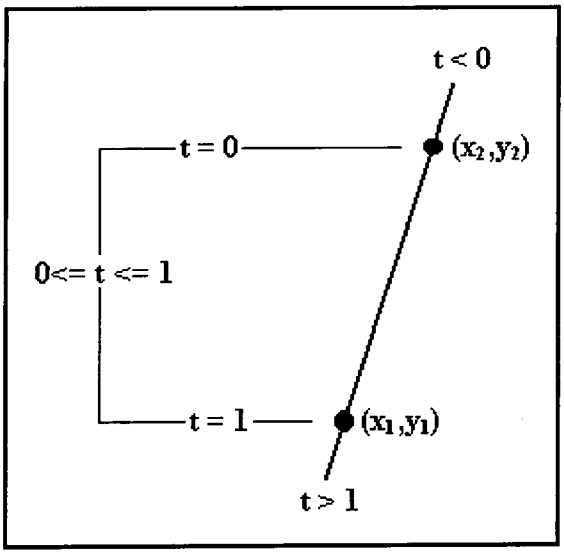


Figure 4.5. Straight line parametric description.

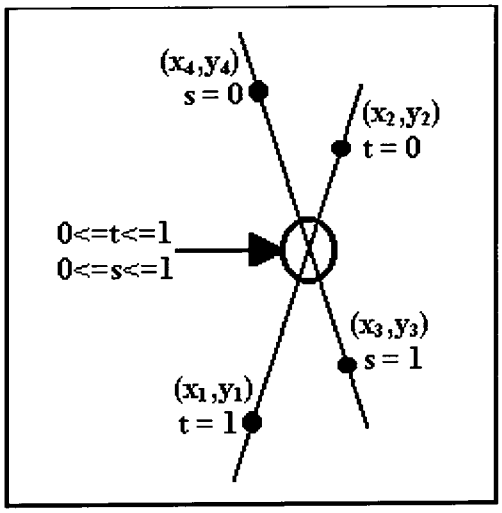


Figure 4.6. An example of two intersecting straight lines.

Line 1 $t(x_1, y_1) + (1-t)(x_2, y_2)$

Line 2 $s(x_3, y_3) + (1-s)(x_4, y_4)$

If these two straight lines intersect, then at the point of coincidence,

$$t(x_1, y_1) + (1-t)(x_2, y_2) = s(x_3, y_3) + (1-s)(x_4, y_4),$$

which is the same as: -

$$t(x_1, y_1) + (x_2(1-t), y_2(1-t)) = s(x_3, y_3) + (x_4(1-s), y_4(1-s))$$

Since both sides of the equation represents the same point, then by comparing the x-co-ordinates of the intersecting point,

$$tx_1 + x_2(1-t) = sx_3 + x_4(1-s) \tag{4.1}$$

and comparing y-co-ordinates

$$ty_1 + y_2(1-t) = sy_3 + y_4(1-s) \tag{4.2}$$

from (4.1) $$t = \frac{s(x_3 - x_4) + (x_4 - x_2)}{(x_1 - x_2)} \tag{4.3}$$

from (4.2) $$t = \frac{s(y_3 - y_4) + (y_4 - y_2)}{(y_1 - y_2)} \tag{4.4}$$

"t" is the position of the point on the line. In this case "t" should be between 0 and 1 for the point to be on the line.

By equating (4.3) and (4.4),

$$s = \frac{(x_4 - x_2)(y_1 - y_2) - (y_4 - y_2)(x_1 - x_2)}{(y_3 - y_4)(x_1 - x_2) - (x_3 - x_4)(y_1 - y_2)} \tag{4.5}$$

from (4.1)

$$s = \frac{t(x_1 - x_2) + (x_2 - x_4)}{(x_3 - x_4)} \tag{4.6}$$

from (4.2)

$$s = \frac{t(y_1 - y_2) + (y_2 - y_4)}{(y_3 - y_4)} \tag{4.7}$$

from (4.6) and (4.7)

$$t = \frac{(x_3 - x_4)(y_2 - y_4) - (x_2 - x_4)(y_3 - y_4)}{(x_1 - x_2)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_2)}$$

$0 \le t \le 1$ and $0 \le s \le 1$ else, there is no intersection.

This approach has been tested for a number of cases to determine its integrity, it has been found that some extra rules were necessary in order to deal with lines that are aligned, for example, to avoid such problems as illustrated in Figure 4.7.

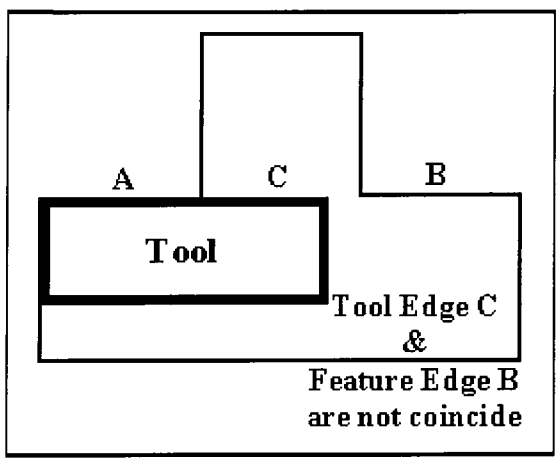


Figure 4.7. An example of not truly coincide straight lines.

If a candidate tool passes the interference test, the boundary curves that are being punched out can be determined. Note; all the intersection points have already been calculated during the interference test. If an intersecting point is located in the middle of a feature curve, then the corresponding feature curve is temporarily split into two feature curve segments at that point, see Figure 4.8.

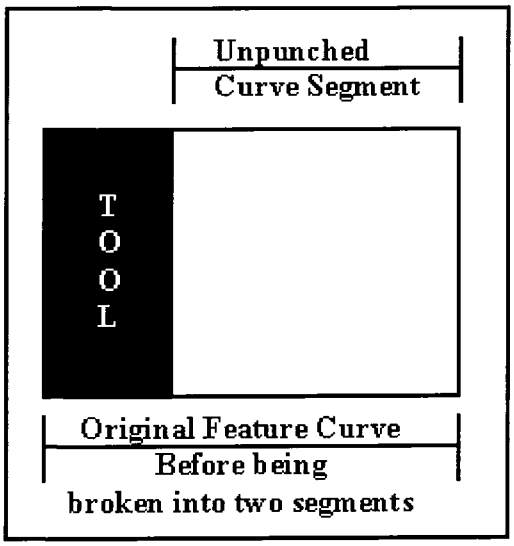


Figure 4.8. An example of split feature edge.

4.2.6. COST FUNCTION

The complex nature of the process of selecting tools to punch out sheet metal part features has its effects on the cost, time and quality of the part to be produced. In punching operations almost any tool can be used to nibble out the product, provided it does not violate the shape to be generated. However, issues such as set-up time of the machine, which depends upon the number of tools and machine tool capacity, production time and cost will restrict such a wide freedom of choice. In selecting a set of tools a number of decisions have to be made i.e., mapping tool edges to feature edges. Hence, taking any decision has it own impact on cost, time and quality. For example, selecting a certain tool for a certain feature may increase the number of blows above the optimum or might result in the need for a tool change which will then increase the time and cost of production. Therefore, the final impact of a number of decisions, in this work, a cost function value, reflects the accumulated impacts caused by the decisions that have been made.

Moving tool selection to be concurrent with the design process could enable cost prediction and should help to ensure that the designer optimises the design with respect

to manufacturing cost, time and quality. In this application of Monte Carlo the impact of the selected set of tools is the summation of all of the components of the cost function, appropriately weighted, resulting from each selection decision. Obviously, different cost function components would have different impacts on the total value of the cost function. For example, an extra single tool set-up has a much greater impact on the production time and cost than an extra single blow.

For this purpose, a cost-function value has been introduced to evaluate each solution and the set of tools with the minimum total cost value is the one selected for punching the part in hand. The main idea of the cost function is to apply a penalty on every component of the process that has an effect on quality, time, and cost of the produced part. Since different components can have varying effects on the quality, time and cost, as such, each component is multiplied by a certain "weight" value, which reflects the amount of the effect. These "weight" values could be deterministic or variable e.g., the load on the punch, based on certain processing parameters.

These "weight" values might be different from one company to another or even from one machine to another. For instance, in the sheet metal industry, the cost of tooling contributes a considerable part to the overall manufacturing cost of a component. Hence, it is imperative to keep this cost down by ensuring the tool operates without breakdown for long periods of time. This can be achieved by reducing the stress on the tool during punching which would prolong the tool life. Therefore the cost function should reflect such an influence on the total cost. Also, the influence of quantity of a part to be made contributes considerably to the overall manufacturing cost of a component, consequently influencing the tools to be selected and the manner in which they are used.

The proposed cost function may take the form: -

Cost Function Value = K_1 (Total number of blows) + K_2 (Number of tool changes) + K_3 (Number of tool set-ups)

For example, in this work the following arbitrary weight values could be, $K_1 = 1$, $K_2 = 10$, and $K_3 = 100$.

This type of proposed cost function is flexible, that is, its terms can be adjusted to suit different types of tool management system. For instance, it could be extended to include laser cutting by adding, say, K_4 (length of laser cut) + K_5 (number of laser cut starts) etc. Furthermore, since the cost function takes into account all of the components that are believed to have an influence on the cost and the quality of the part, it is able to overcome the need for a traditional "adaptive" or "blind selection" process.

One of the advantages of the proposed cost function is its simplicity and flexibility. The fact that it is described in general terms and is not based on specific types of tools or machines allows it to be applied to a wide range of processing situations, as illustrated by its simple extension to laser cutting.

4.2.7. SIMULATION MATHEMATICAL ANALYSIS

In this study, after a large number of simulation runs the distribution of the computed values, the cost function values derived from a series of simulation runs, is assumed to be normal. Here, both the Mean (μ) and Standard Deviation (σ) can now be found.

For a Normal Distribution [106],

$$Y = \frac{N}{\sigma\sqrt{2\pi}} e^{-(X-\mu)^2/2\sigma^2} \qquad (4.8.)$$

Where,

Y = Frequency of a given value of X,

X = Any score on the distribution,

μ = Mean of the distribution,

σ = Standard Deviation of the distribution,

N = Total frequency of the distribution,

π = A constant of 3.1416, and

e = A constant of 2.7183.

The first requirement is to determine the cost function value that represents a close approximation to the optimum value. In Figure 4.9. the shaded area is the area below the maximum acceptable score. For example, if 1% of the total distribution area lies below the target then it is assumed that 99% must lie above the target.

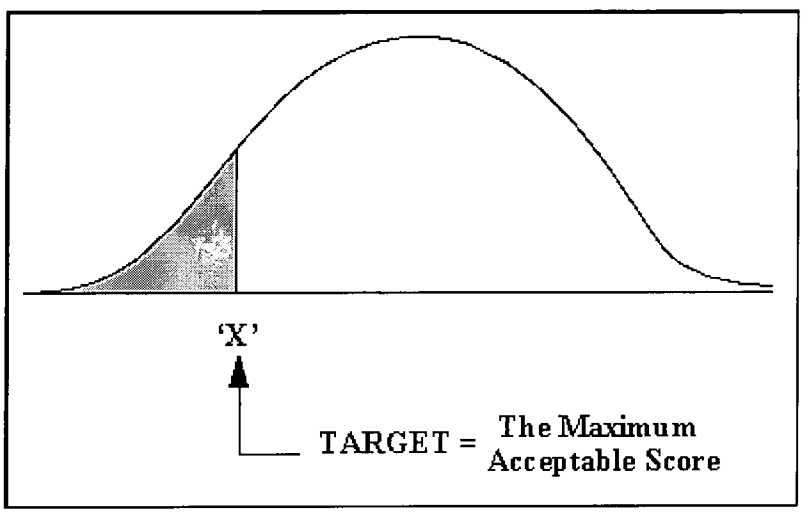


Figure 4.9. Distribution curve.

Using an "area under the normal distribution curve" table, it is now possible to determine a corresponding z score value. Substitution of the z score into

$$z = \frac{X - \mu}{\sigma} \qquad (4.9.)$$

Gives the value of X as,

$$X = z\sigma + \mu \qquad (4.10.)$$

Where, X is the target cost function value.

It is now necessary to determine how many simulation runs are required to expect an X value to fall on or above the target value. The number of simulation runs is "N", the total frequency of the distribution. Hence, using equation (4.8.)

$$1 = \frac{N}{\sigma\sqrt{2\pi}} e^{-(X-\mu)^2/2\sigma^2}$$

Note: "Y" is given a value 1 because the calculation is seeking a single occurrence of the target value.

Other than N, the "unknown", all other terms are constant and known from a preliminary search. Hence, if the preliminary search is statistically valid the number of simulations required to achieve a target cost function value can be computed.

The next chapter will focus on the software produced during this work. This software was necessary to assess the practical performance of the methodology described in the chapter now completed.

CHAPTER 5

THE SIMULATION PROGRAM

The main principle behind this simulation program, see Figure 4.4. page 68, is to demonstrate the feasibility of the proposed approach of using Monte Carlo simulation methods in tool selection. The objective was not to develop commercial software. A professional programmer will see ways of improving the current simple but inefficient programming style. Such an improvement will lead to a better performance which should make the approach more appealing for industrial use. The current computer program consists of 35 Modules, grouped into three main functional groups, see Figure 5.1. The first of which, see Figure 5.2., contains 25 "Data input" related modules. The second group see Figure 5.3., the main body of the programme, deals with repeating the process of selecting a set of tools to punch all required features. Thus, this group deals with selecting and mapping a random tool edge for a random feature edge and checking that the selected tool edge does not cause any damage to the work piece. The last group of modules is concerned with Feature update and the computations, described in the previous chapter for predicting the global minimum cost function value and eventually selecting the best set of tools.

The following sections will discuss in more details some of the above mentioned simulation modules.

5.1. SYSTEM MODULES

The computer program is divided into three functional groups. Modules are discussed under their functional group headings. Simple "housekeeping" modules have not been discussed here.

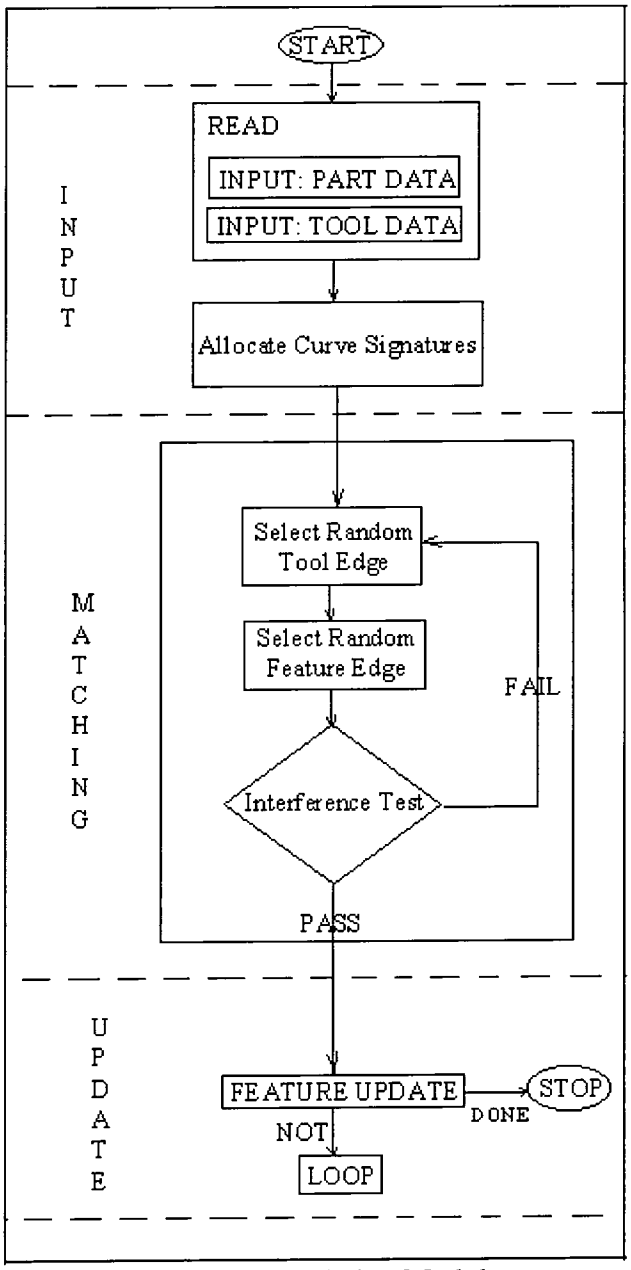


Figure 5.1. Simulation Modules.

- Sub Auto_Open()
- Sub Which_Part_Data()
- Sub Which_Set_Of_Tools()
- Sub How_Many_Feature_Are_There()
- Sub Number_Of_Rows()
- Sub Close_Part_Data_File()
- Sub P_Data_Sheet()
- Sub Design_P_Data_Sheet()
- Sub Clear_P_Data()
- Sub Preceding_Curve_P_Data()
- Sub Proceeding_Curve_P_Data()
- Sub Curve_Status_P_Data()
- Sub Curve_Curvature()
- Sub Curve_Length()
- Sub Angle_With_Preceding_Curve_P_Data()
- Sub Angle_With_Proceeding_Curve_P_Data()
- Sub Tool_Data_File()
- Sub Number_Of_Rows_In_T_Data()
- Sub Number_Of_Tools()
- Sub Copy_T_Data_TO_P_Data_Sheet()
- Sub Preceding_Curve_T_Data()
- Sub Proceeding_Curve_T_Data()
- Sub T_Curve_Length()
- Sub T_Curve_Curvature()
- Sub Angle_With_Preceding_Curve_T_Data()
- Sub Angle_With_Proceeding_Curve_T_Data()

Figure 5.2. Input Related Modules.

- Sub CurveMatchingAndInterfernceTest()
- Sub GenerateRandomFeatureNumber()
- Sub GenerateRandomToolNumber()
- Sub GenerateRandomFeatureEdgeNumber()
- Sub GenerateRandomToolEdgeNumber()
- Sub CheckWhichCurveMatchingCase()
- Sub CalculatingTheNecessaryToolEdgeRotationAngle()
- Sub InterferenceTest()

Figure 5.3. Tool Selection Related Modules.

5.1.1. SUB AUTO_OPEN()

This module can literally be considered as the main module that calls upon other related modules for the execution of assigned tasks. The next sub sections discuss some of these modules. These sub modules are in the main associated with curve signatures, curve matching and interference testing.

5.1.1.1 Tool and Feature Curve Signature Modules

These modules are concerned with assigning a curve signature value for each tool and/or feature curve. Some of these modules purely act as pointers in a doubly linked list, as has been discussed in Chapter 3, and where a linked list is used to describe the hierarchical structure in the decision tree. In this work each tool and feature entity occupies a complete row in a spreadsheet. Two cells are used to point to the preceding and proceeding entities. This arrangement has proved very useful in situations where the feature curve is subsequently split into two or more curves. Without such an efficient record of structure and punching precedence it would be very difficult to handle the process of adding newly created feature edges and removing punched feature curves/ entities. The modules for handling these matters are:

Sub Preceding_Curve_P_Data()	Sub Proceeding_Curve_P_Data()
Sub Preceding_Curve_T_Data()	Sub Proceeding_Curve_T_Data()

The next set of modules discussed, below, deal with different curve signature values i.e. length, radius of curvature, angle with preceding entity and angle with proceeding entity.

5.1.1.1.1. Curve Length:

Both modules 'Sub Curve_Length()' and 'Sub T_Curve_Length()' deal with computing the length of feature and tool entities respectively. First, the type of the entity in hand (straight line or a circular curve) is checked in order to decide the related sub modules to call for calculation of the required length. Details of the analysis used for calculation where discussed in Chapter 4.

5.1.1.1.2. Radius of Curvature

Once again the entity type has to be identified first. If the entity in hand is a circular curve then the radius of the curvature is actually the radius of the curve which is extracted from AutoCAD data. However, if the entity in hand is a straight line then synthetically the entity is given a radius of curvature of '999'. Two modules 'Sub Curve_Curvature()' and 'Sub T_Curve_Curvature()' deal with allocating the value of radius of curvatures for both features and tools respectively.

5.1.1.1.3. Angles with preceding and proceeding entities

Four sub modules,

Sub Angle_With_Preceding_Curve_P_Data()

Sub Angle_With_Preceding_Curve_T_Data()

Sub Angle_With_Proceeding_Curve_P_Data()

Sub Angle_With_Proceeding_Curve_T_Data()

deal with finding the angle with preceding entity and the angle with the proceeding entity. Once again entity type is required in order to decide on the type of the match, such as, angle between two straight lines or angle between two circular curves. It is worth mentioning here that the focus of the present work was on straight lines only. Although some work has been done for angles between straight lines and circular curves, incomplete work is pointed out clearly in the code in Appendix 'B'.

5.1.1.2. Curve Matching And Interference Test

This group contains a number of modules for selecting random feature and tool edges. First, the type of tool-feature curve matching in hand is defined, that is, for example matching a straight line tool edge with a straight line feature edge. Hence, the necessary tool rotation angle is calculated and its newly mapped exact position is defined. Finally the tool is checked against any damage it might cause to the work piece by removing

excessive material. If the tool is safe then the tool is selected and the details of the feature in question are updated. As discussed in earlier chapters, the selected tool might partially punch out some of the feature edges, which causes the generation of new feature entities which need new curve signature values. The punched feature is then removed from the list of "not punched" curves. The same process carries on until all features are completed. A cost function value is then computed and allocated to the selected set of tools. This process of selecting a set of tools can be repeated as required. The next chapter deals with a number of case studies undertaken using the software described here.

CHAPTER 6

RESULTS AND DISCUSSION

It is proposed that the approach to solving this tool selection problem is considered in three phases. This chapter explores the use of simulation to show the applicability of each of the phases. The first phase is to establish the validity of the simulation planning, particularly the determination of sample size. Once a statistically significant sample size is obtained it should be possible to use that sample to establish a Mean and Standard Deviation for the distribution of the cost function value. In phase two, the objective is to use the knowledge of the distribution to estimate how many trials would be required to find a tool selection that is acceptably near to the optimum. The experimental work described here shows that the assumption of a Normal Distribution is a reasonable basis for the calculation of a minimum number of trials to achieve an acceptable results. The chapter ends with a discussion of specific cases and further explores some of the potential difficulties experienced using this type of approach, particularly the effect dominant tools have on the selection process. Although dominant tools can cause difficulties they also represent the possibility of an improved method using a Monte Carlo - Rule based hybrid, Simulated Annealing, or Genetic Algorithms.

To avoid confusion and interruption while reading this text, some figures have been placed at the end of this chapter. The reader will be refered to these figures by their page numbers.

6.1. ESTABLISHING AN ACCEPTABLE SAMPLE SIZE

Of initial concern is the sample size. Usually when a statistical experiment is to be conducted the size of the random sample which would best represent the population should be computed. In drawing a random sample from any population, the 'Mean' of the sample would change as further samples are taken. Eventually, as more samples are obtained the mean of the sample will approach the 'Mean' of the population. This argument is supported by what is known as "The Law of Large Numbers" [107].

As has been discussed in Chapter 4, in carrying out a tool selection search, the first requirement is to estimate the "target" cost function value that represents a close approximation to the optimum value, that is, in a simple case, the minimum number of blows. In doing so, the operator first conducts a number of simulation runs and computes the mean and standard deviation of the cost function values for those runs. Based on the assumption that over a large number of simulation runs the distribution of the computed values will give a Normal Distribution, the number of simulation runs that are needed to achieve a target cost function value can be computed.

During this research it was found that a sample of 100 simulation runs provided an acceptable representation for the range of tool selection populations considered. It was found that in most cases the mean cost function value of the simulation runs approached a constant value even at much lower simulation runs than 100. The features and tools used in these cases are shown in Figures 6.1. to 6.5. These results are illustrated by the graph shown in Figure 6.6., page 132. This graph shows that for the five case studies, containing different populations of features and tools, the mean cost function values obtained for increasing sample sizes stabilises before a sample size of 100 was reached.

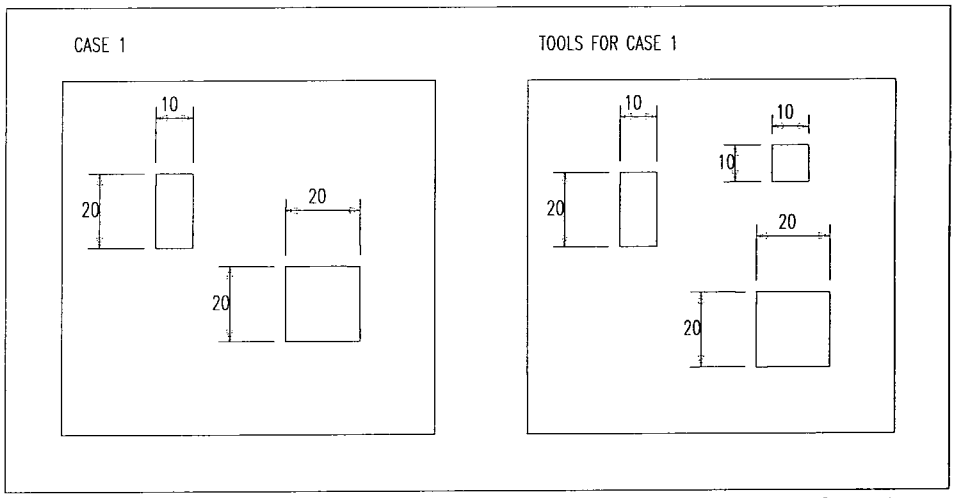


Figure 6.1. Stability of the mean cost function at 100 runs – Case 1.

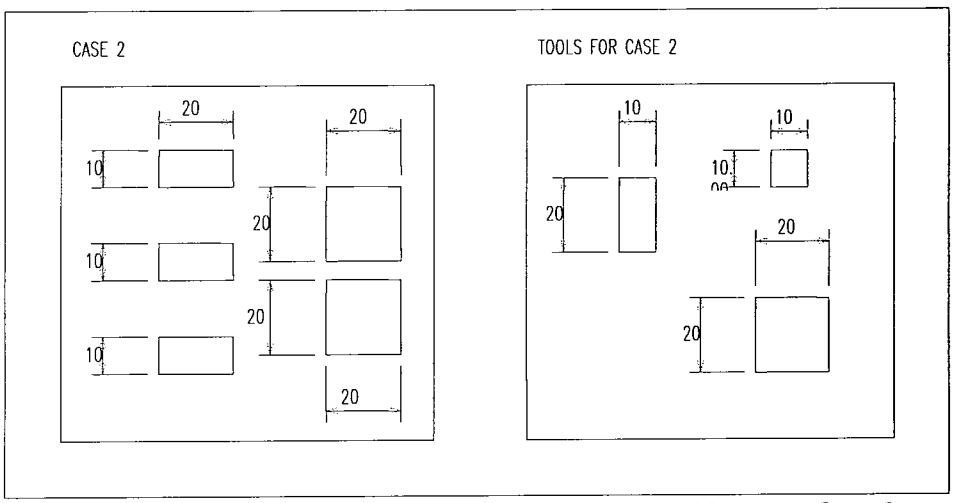


Figure 6.2. Stability of the mean cost function at 100 runs – Case 2.

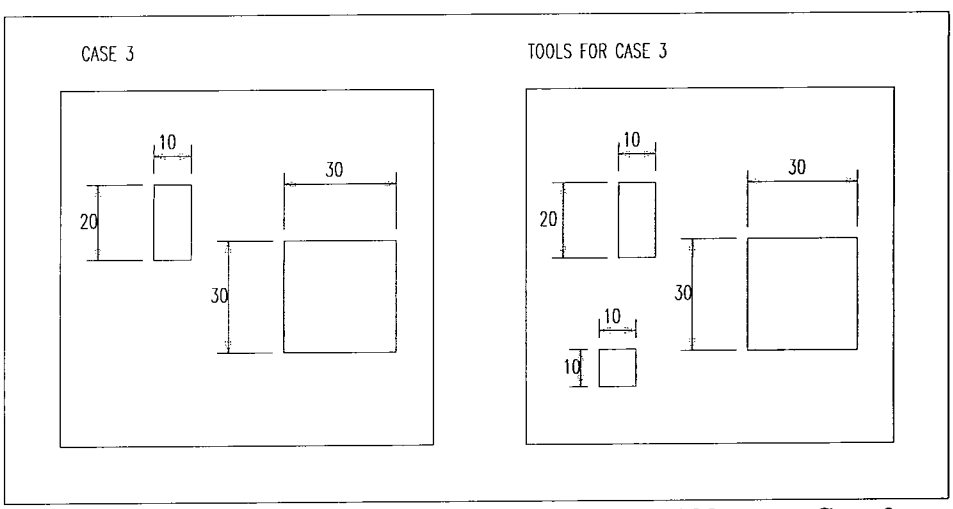


Figure 6.3. Stability of the mean cost function at 100 runs – Case 3.

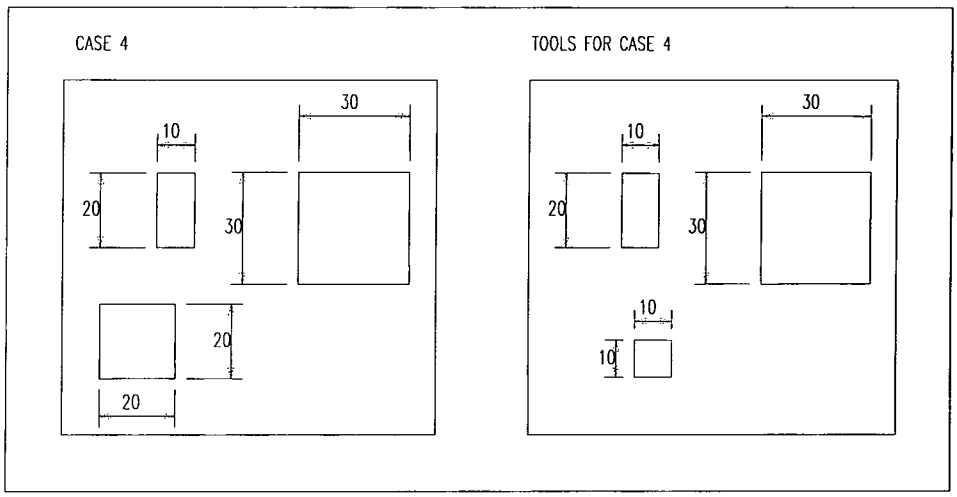


Figure 6.4. Stability of the mean cost function at 100 runs – Case 4.

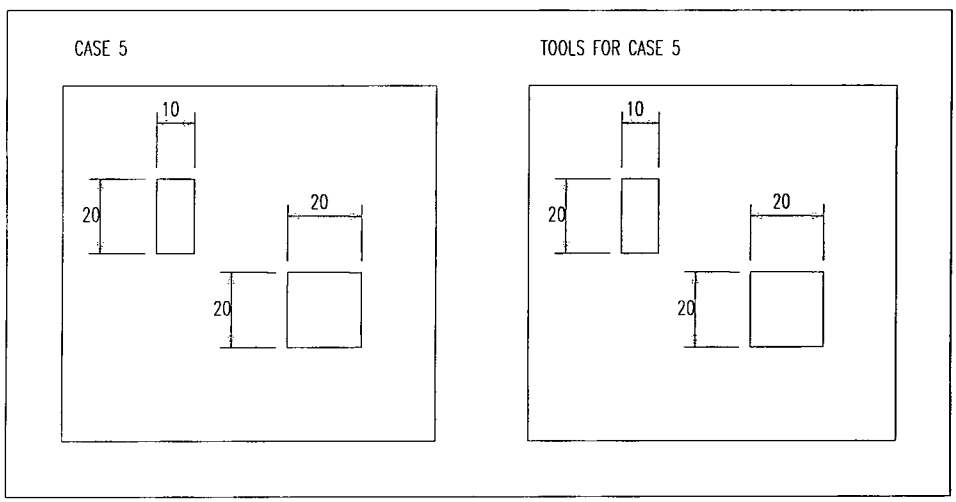


Figure 6.5. Stability of the mean cost function at 100 runs – Case 5.

It is clear from the graph in Figure 6.6, page 132, that the effect of the Law of Large Numbers will be influenced by the particular tool and feature population. However, in all the cases considered, a value of sample size of 100 was found to be adequate. This may not be the case for larger tool and feature population and this is a matter for further investigation.

In a similar manner, the graph in Figure 6.7., page 133, shows a variation in the 'Standard Deviation' as the sample size increases. The 'Standard Deviation' is a useful value when computing the prediction of the number of simulations which may have a reasonable chance of achieving a near optimum solution. As previously mentioned, a sample size of 100 seems adequate for presenting a useful value for the 'Standard

deviation'. Thus, a sample size of 100 is proposed on the basis that it will produce useful values of 'Mean cost function value' and 'Standard Deviation' for the population at an acceptable computational cost.

6.2. CALCULATING THE REQUIRED SIMULATION RUNS TO ACHIEVE THE TARGETED MINIMUM COST FUNCTION VALUE

Having established that a sample size of 100 runs acceptably represents the population in hand, the next step, based on the available information regarding the population, is to compute the theoretically expected minimum cost function value. Further, it is necessary to calculate how many simulation runs it should take to achieve this target value.

The results of each 100 run trial are given in Figures 6.8 to 6.12, pages 134 to 138, and correspond to cases 1 to 5, respectively. The empirical based predictions of the distribution have been represented by the 'Mean cost function value' and 'Standard Deviation', given in Table 6.1. Further, these results can now be used to compute the requirements, in terms of number of runs, to achieve an acceptable tool selection. These calculations fall into two phases. The first phase is the calculation required to obtain a theoretical estimate of the minimum cost function value. This first estimate can be obtained by targeting a result where 99.98% of the population lies above the target value. This will give a very attractive estimate of the theoretical minimum cost function value. However, to find the tool selection required to achieve this performance would probably involve an unacceptably large computation time, see Table 6.1. However, this theoretical value is a useful benchmark for assessing results from shorter simulation runs. The theoretical minimum cost function value (e.g. number of blows) estimated for the cases considered are given in Table 6.1. Note that some of these values are unrealistic, for example, one blow. These values are caused by the assumption of a

normal distribution, where it is not strictly appropriate. Further, in a qualitative sense they represent the extreme lower end of the "number of blows" spectrum and hence, are useful from this point of view for identifying a reasonable number of trials required to achieve a "low" blow count.

The second phase of the calculation relates to obtaining a value for the number of simulation runs required to achieve an optimum value for the cost function, as well as the tool selection required to achieve that value. In this phase the normal distribution is used to determine both the target number of blows and the number of simulation runs, such that 99 % of the population will lie above the target value. In this case, the number of simulation runs can be computed from the sample mean cost function value and standard deviation and the assumption of a normal distribution. The results of these calculations are given in Table 6.1.

As can be seen from Table 6.1 the computational requirement is considerably reduced when compared with the 99.98% population. From the results obtained so far it has been found that the assumption of a 99% target should guarantee an optimum or near optimum value for the minimum cost function value required for these practical cases.

	Mean cost function value	Standard Deviation	Theoretical Computed Minimum Number of Blows		The required Simulation Runs to Achieve the Target Number of Blows	
			99.98%	99%	99.98%	99%
Case 1	4.84	0.762	4	4	29	5
Case 2	11.44	1.122	9	10	42	7
Case 3	4.43	1.822	1	3	69	11
Case 4	7.66	2.095	3	5	79	12
Case 5	2.65	0.716	1	2	27	5

Table 6.1. Theoretical estimated minimum cost function values.

To investigate the validity of the calculations for the theoretical minimum cost function value and the number of simulation runs required, a different case study has been investigated, see Figure 6.13A and 6.13B.

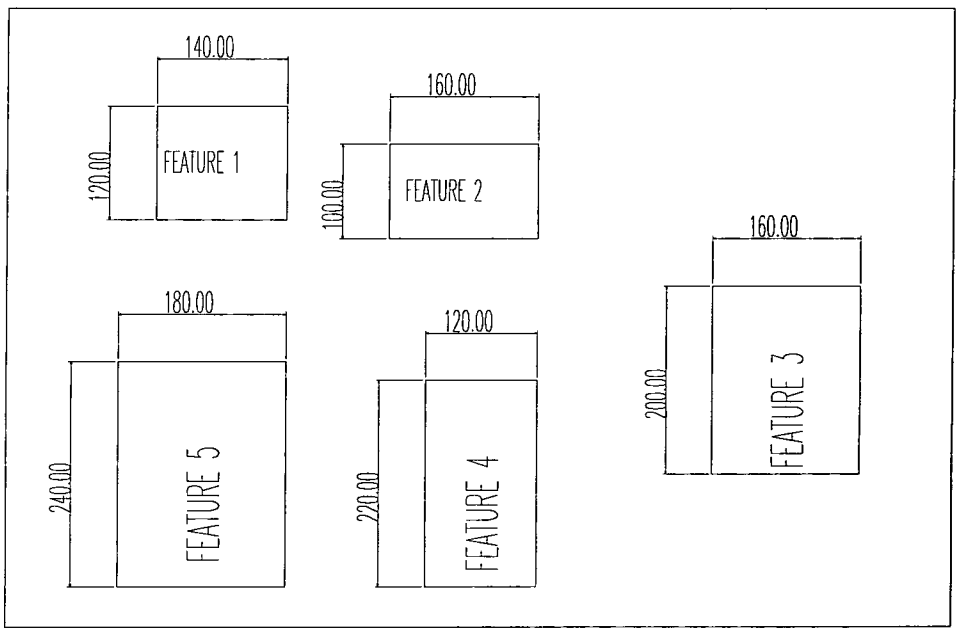


Figure 6.13A Case Study - Features

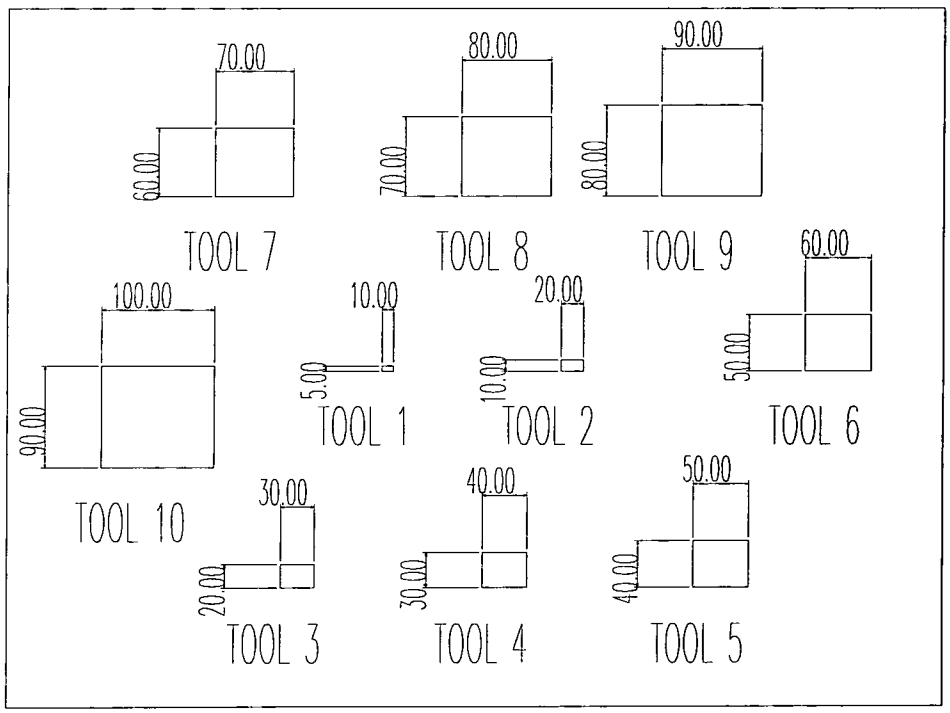


Figure 6.13B Case Study - Tools

In this case study, a sample of 100 simulation runs was generated, see Figure 6.14, page 139. For clarity, the cost function value was simplified to only include the number of blows to punch out the product that is there were no tool changes etc. It was found that the theoretical minimum number of blows required was 45 and the recommended number of simulation trials to achieve this theoretical minimum number of blows was 259.

By thoroughly analysing the generated simulation runs, it was found that the first 100 simulation runs returned a value of 40 blows at the 99^{th} simulation run. After running the recommended number of trials (259 runs), it was found that a result of 37 blows was achieved at the 155^{th} run. Further, it was found that by running an additional 241 runs on the top of the recommended number of trials no further improvements was obtained, see Figure 6.15, page 140.

To investigate the validity of the calculation, another 100 samples were generated, see Figure 6.16, page 141. Here, the theoretical minimum number of blows was also 45 blows. However, the recommended number of simulation trials was 258. Once again, by thoroughly investigating the results it was found that during the first 100 runs a result of 44 blows was achieved, at the 49^{th} trial. By running the recommended number of trials, that is, 258 runs, a result of 42 blows was achieved at the 132^{nd} run. Once again, by running an additional 242 simulation runs on top of the recommended number of trials it was not possible to obtain an improved solution, see Figure 6.17, page 142.

A further 100 samples were generated, see Figure 6.18, page 143, these gave a theoretical minimum number of blows of 43 and 267 simulation trials were recommended. Here, a result of 41 blows was achieved at the 47^{th} trial. By running the

recommended number of runs it was not possible to obtain an improved solution. In this case, by carrying out an additional 233 simulation runs over and above the recommended number, a better result of 38 blows was obtained at the 430th run, see Figure 6.19, page 144.

A fourth trial of 100 simulation runs, see Figure 6.20, page 145, gave a theoretical minimum number of blows of 42 and 278 simulation runs. The minimum result of 44 blows was found at the 10th run. By running the recommended 278 simulation runs, a better result of 39 blows was found at the 156th run and that remained the minimum number of blows over an additional 222 simulation runs, see Figure 6.21., page 146.

A fifth and final attempt was conducted, see Figure 6.22, page 147, here, a minimum of 42 blows was achieved on the 67th run. It was found that this result agreed with the theoretical computed value. The program recommended an additional 267 simulation runs. Here, a better result of 39 blows was found at the 129th simulation run and this remained the lowest number of blows even after running an additional 233 simulation runs, see Figure 6.23, page 148.

In all of the above cases a better result than the theoretically predicted value was obtained. These investigations have established the practical value of the calculations for the minimum number of blows and the recommended number of simulation runs.

An understanding of the effectiveness of the above approach can be gained by comparing these results with a more comprehensive search which used 500 runs. Here, a comparison can be made between the minimum number of blows determined by a 100 run sample and 500 runs. This minimum can be seen by reference to the results in

Figure 6.24., page 149. The results shown are for runs of the same product and tool populations, shown in Figures 6.13A and 6.13B, and they relate to the five experiments described above.

The envelope of the points shown indicate that a sample of 100 runs gives an acceptable result in terms of the number of runs required to achieve an acceptable value for the number of blows. However, with the first sample of 100 runs the minimum number of blows may or may not be near to the minimum number of blows for a longer search.

In the following sections, a number of different case studies will be discussed in an attempt to gain a better understanding of the behaviour of cost function value distributions.

6.3. CASE STUDIES

The case studies start by analysing very simple features and tool populations. These initial, simple cases, made manual checking of the results possible. As the cases became more complex, in terms of the number of features and tools, it becomes more difficult to check the results manually. However, it is those very complex cases that benefit from the time savings produced by such an application.

6.3.1. CASE STUDY 1

In the first case study shown in Figure 6.25, and Figure 6.26, page 150. there are two different features and each feature can only be punched by one tool. Therefore, it is obvious in this case, that the total number of blows equals the total number of features, as each feature will be punched by only one blow. As anticipated, the computed result of two blows agrees with the answer obtained by inspection.

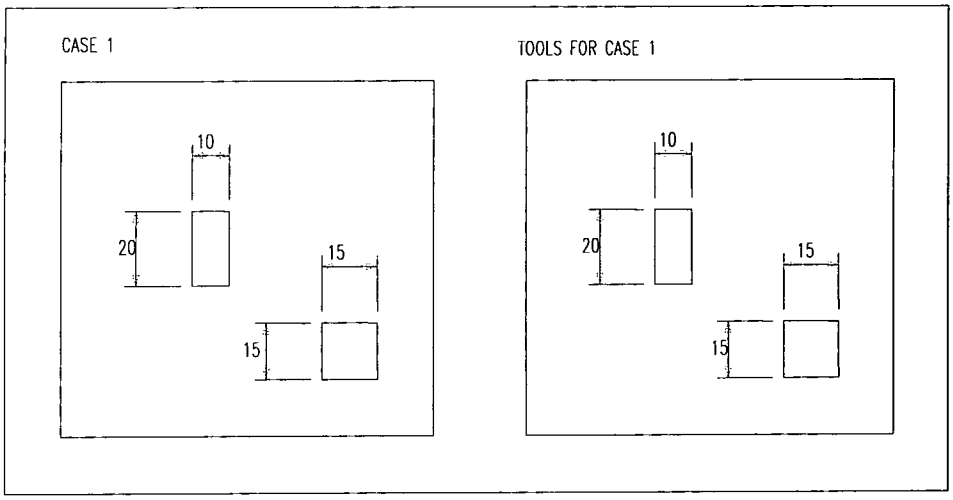


Figure 6.25. Case study 1.

6.3.2. CASE STUDY 2

In case two there are two features, these are shown in Figure 6.27. One feature can be punched by only one of the tools, whilst the other can be punched by either one tool or a combination of the two tools or by multiple punches of a single tool. In this case, the simulation has a limited number of possible answers for the total number of blows. It can be seen from Figure 6.28., page 151, that a single tool per feature is used more frequently than a combination of tools. Once again, the computer generated solutions agree with those determined manually, even though the four blow solution is extremely inefficient.

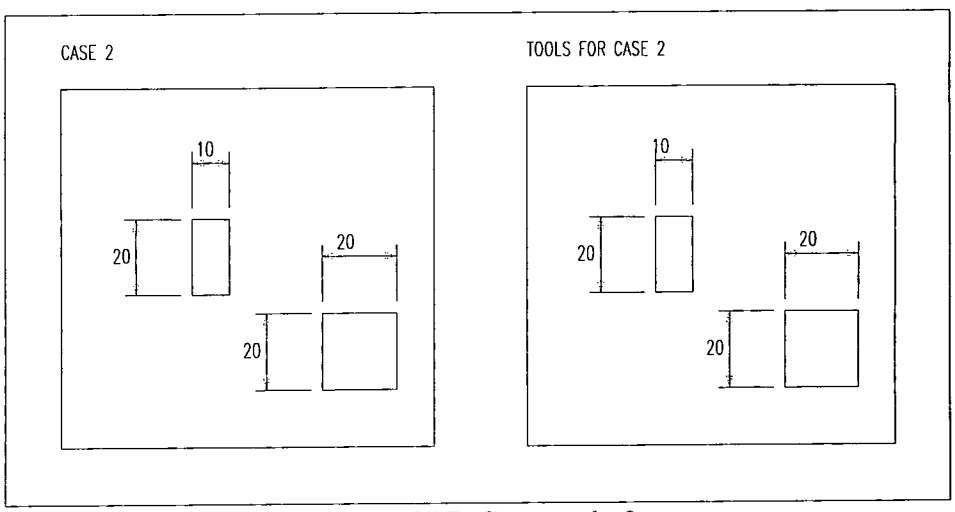


Figure 6.27. Case study 2.

6.3.3. CASE STUDY 3

In case study three both features and tools are shown in Figure 6.29. The features can be punched by either a single tool or more than one tool. The first feature, 10 x 20, can be punched out by only one or two blows. The second feature, 20 x 20, can be punched by either one, two, three, or four blows, giving a minimum and maximum number of blows of 2 and 6, respectively. Figure 6.30, page 152, shows the computed results which predicted the manual search but which also indicates the frequency of occurrence of particular selections.

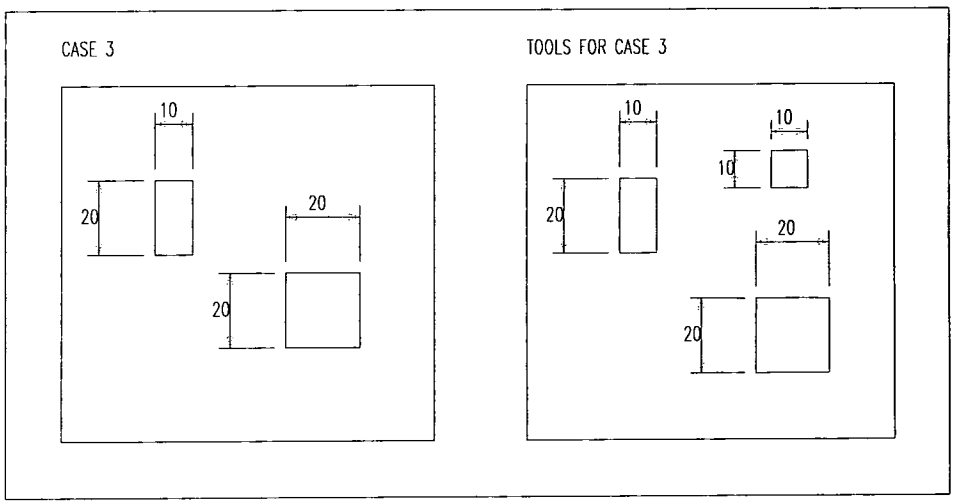


Figure 6.29. Case study 3.

6.3.4. CASE STUDY 4

In case study 4, described by figures Figure 6.31., below, and 6.32., page 153, only one tool is available. Here, the tool is used to nibble all the features. In this case, the total number of blows remains constant at a value of 6 blows.

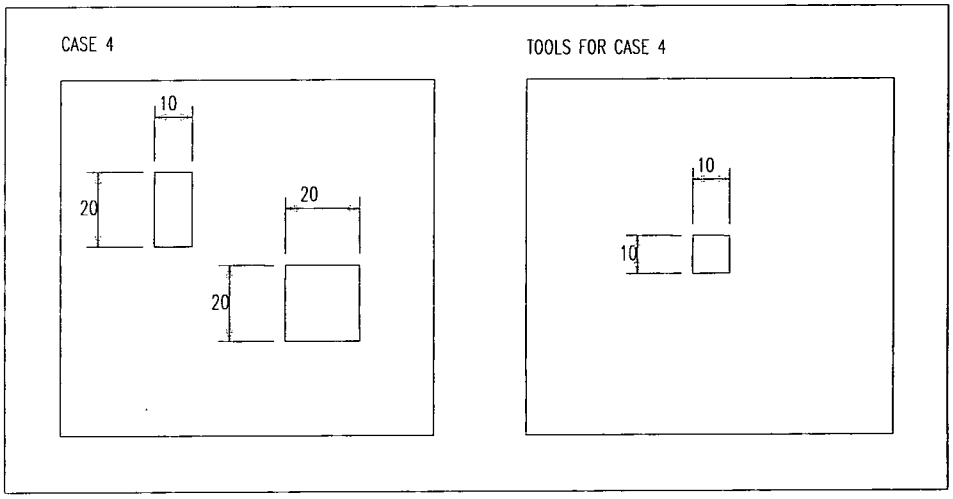


Figure 6.31. Case study 4.

It is worth noting at this point that even with a single tool it is possible to use a different number of blows to achieve the same finished product. An example has already been given, in part, in Case Study 2, here, the feature (20 x 20) can be punched by two or three blows using the tool (10 x 20). However, in this case, using only the small tool the computed value derived over 100 runs was 6 and this was also found to be the only possible solution by manual inspection.

6.3.5. CASE STUDY 5

This case study is illustrated by Figure 6.33., and is similar to Case Study 3 in that it uses the same tool population. It can be noted that as the number of features increases the cost function value distribution starts to approximate to a normal distribution curve. Note that by manual inspection the minimum number of blows required would be 5 however the 100 run trial was unable to find this value, see Figure 6.34, page 154.

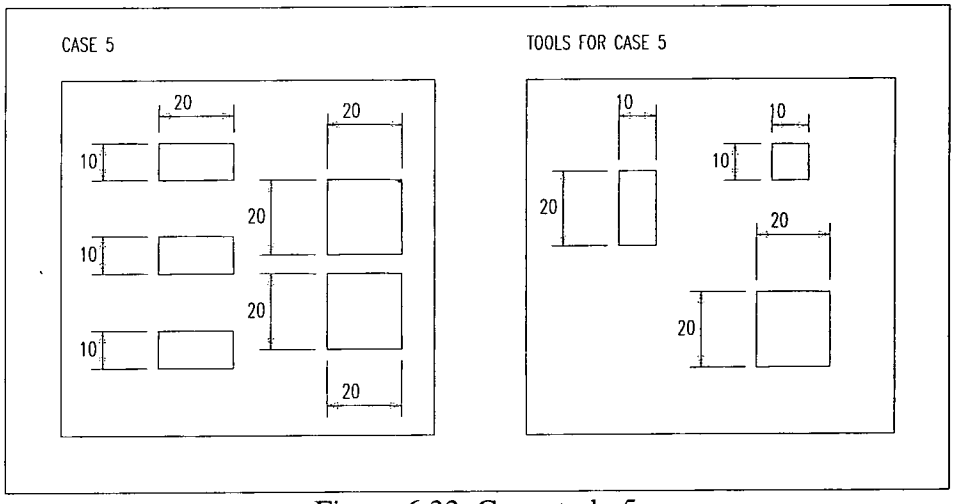


Figure 6.33. Case study 5.

6.3.6. CASE STUDY 6

Case Study 6, shown in Figure 6.35., and 6.36, page 155, presents some interesting results . Although this case is similar to Case Study 3, in that it involves two features and three tools, the inclusion of a large feature (30 x 30) has opened up a number of possible solutions. Thus, the range of possible solutions increases and the distribution starts to show more than one peak, as the curve deviates from a smooth distribution. This leads to the later discussion on the effect of dominant tools and features. Inspection, indicates that the minimum and maximum number of blows should be two and ten, respectively. The computer predicted minimum number of blows was 2. In this instance, the trial of 100 runs determined the minimum value but did not find the maximum value. The current work has no interest in maximum values other than the fact that it is an indication of some asymmetry in the cost function value distribution.

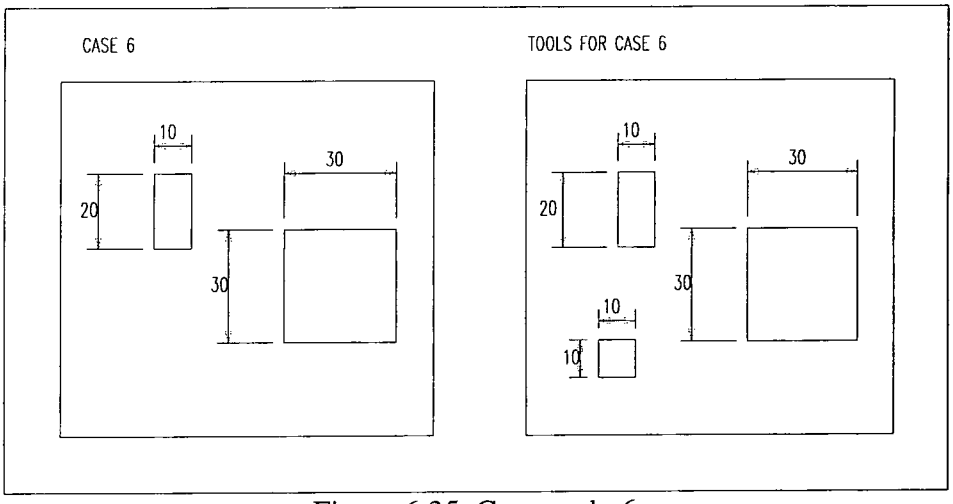


Figure 6.35. Case study 6.

6.3.7. CASE STUDY 7

Case Study 7 is shown in Figure 6.37. The idea of more than one peak becomes clearer as there are clear dominant features, and hence a large number of possible solutions. This is an interesting case if compared to Case 6. It shows the introduction of a second dominant feature. However, that feature is smaller than the large 300 x 300 feature in both cases and so has a smaller effect, see Figure 6.38., page 156.

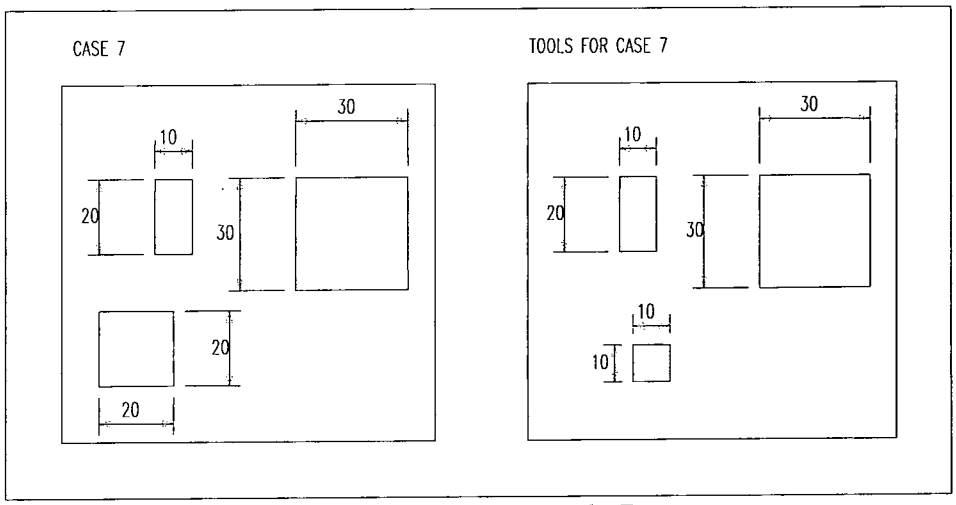


Figure 6.37 Case study 7.

6.3.8. CASE STUDY 8

Case Study 8 is illustrated by Figure 6.39. This case can be compared to Case 7, however the dominant tool has been removed, thus reducing the range of possible solutions for the feature (30 x 30). The tool was removed in order to investigate the

change in "smoothness" of the curve, as illustrated in Figure 6.40., page 157. This case is as expected, that is, smoother with a shorter range and a single peak. Once again, manual inspection gives a minimum value of 2 and a maximum value of 4.

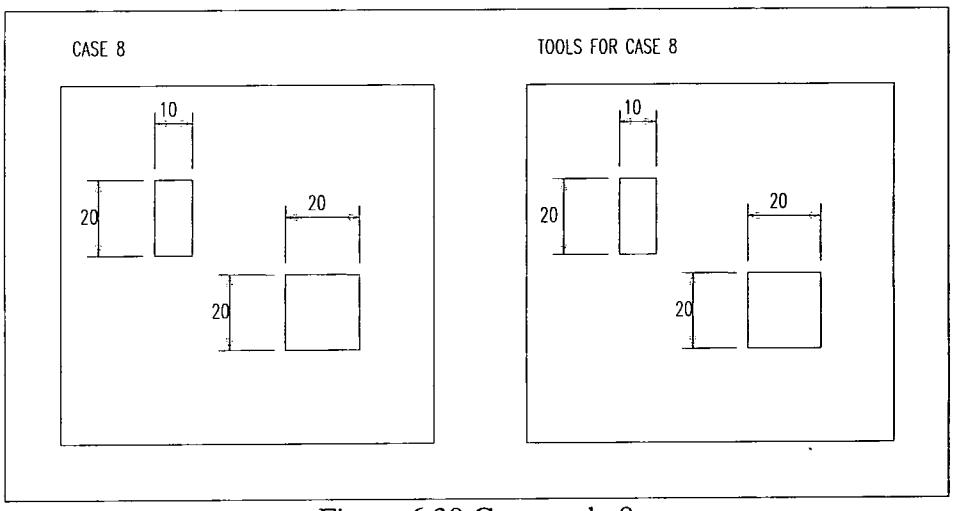


Figure 6.39 Case study 8.

6.3.9. CASE STUDY 9

So far the cases considered have been "simple" and it has been possible to determine the maximum and minimum number of blows by inspection. As cases become more complex determining the number of blows becomes, at best, more time consuming. Case Study 9, see Figures 6.41. and 6.42., represents a more complicated case in that there are an increased number of features and many more tools. The maximum number of blows can still be computed by selecting the smallest edge on the smallest tool and aiming to nibble out all of the features. The minimum number of blows is achieved by selecting tools on a "one to one" match with the features. In this case not all of the features can be punched on a "one to one" basis. Additionally, there may be a tool magazine limitation on the number of tools used in one setting. Thus, determining the minimum number of blows in this relatively simple case is far from obvious.

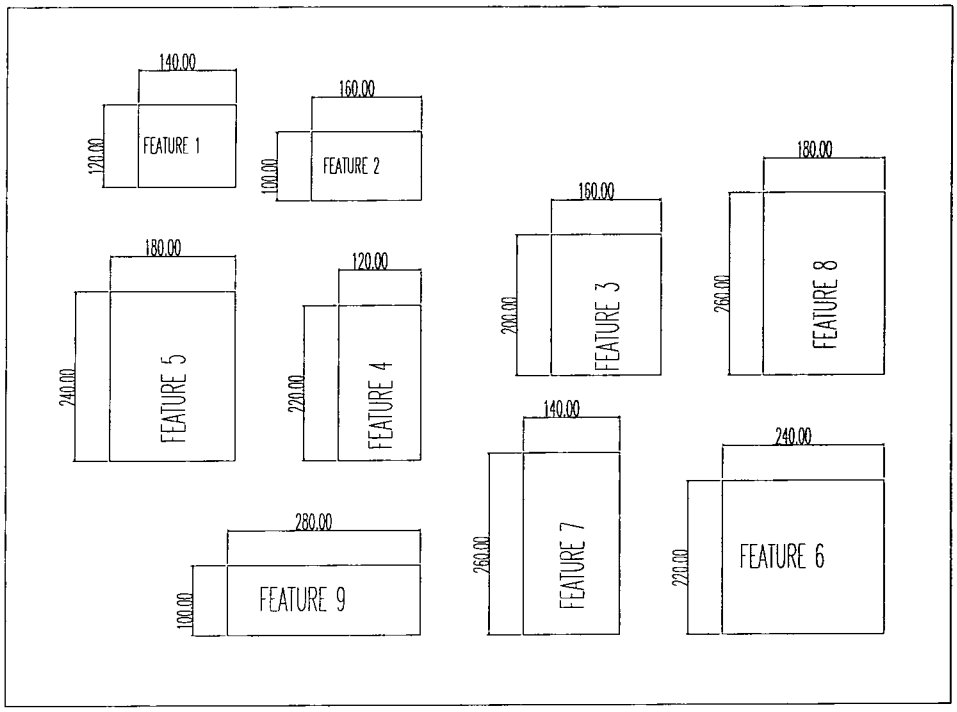


Figure 6.41. Case study 9 - Features.

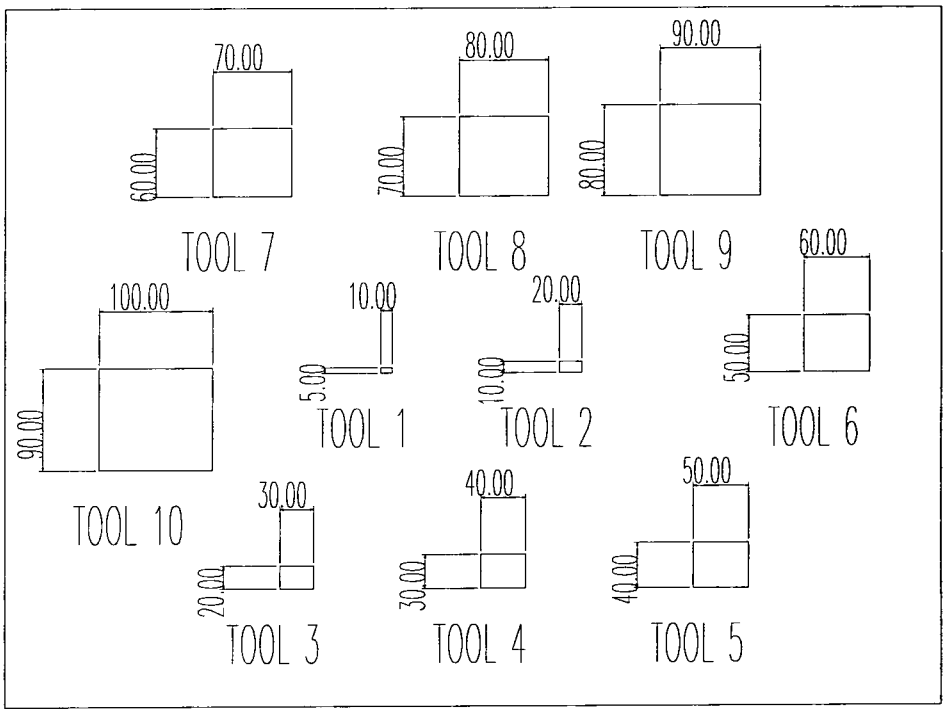


Figure 6.42 Case study 9 – Tools.

In this case study, 100 initial Monte Carlo runs have been conducted, where the best three solutions of a cost function values of 93, 102 and 102 were selected. Applying the next phase of the Monte Carlo search, namely 'Gene Mixing and Re-Sequencing' a dramatic improvement on the current prime search was achieved. For example, Feature

9 of the first solution had 6 blows as shown in Figure 6.43. However by Gene Re-Sequencing the six blows were reduced to only 4. Another example, see Figure 6.44., by mixing two solutions for Feature 8 which were originally 10 and 13, a new improved solution of only 6 blows has emerged.

Applying "Gene Mixing and Re-Sequencing" to the 3 picked solutions has resulted in a solution of 52 blows, see Figure 6.45. The tools in Figure 6.45. are marked with A, B and C letters which reflects to which solution they originally belong. These 52 blows have been achieved using 8 tools which are tools 3, 4, 5, 6, 7, 8, 9, and 10.

This solution can be compared with the case of using the largest tool i.e. tool number 10 which resulted in a solution of 46 blows as follow:

Feature	Number of Blows
1	4
2	2
3	**4**
4	6
5	**6**
6	8
7	**6**
8	6
9	4

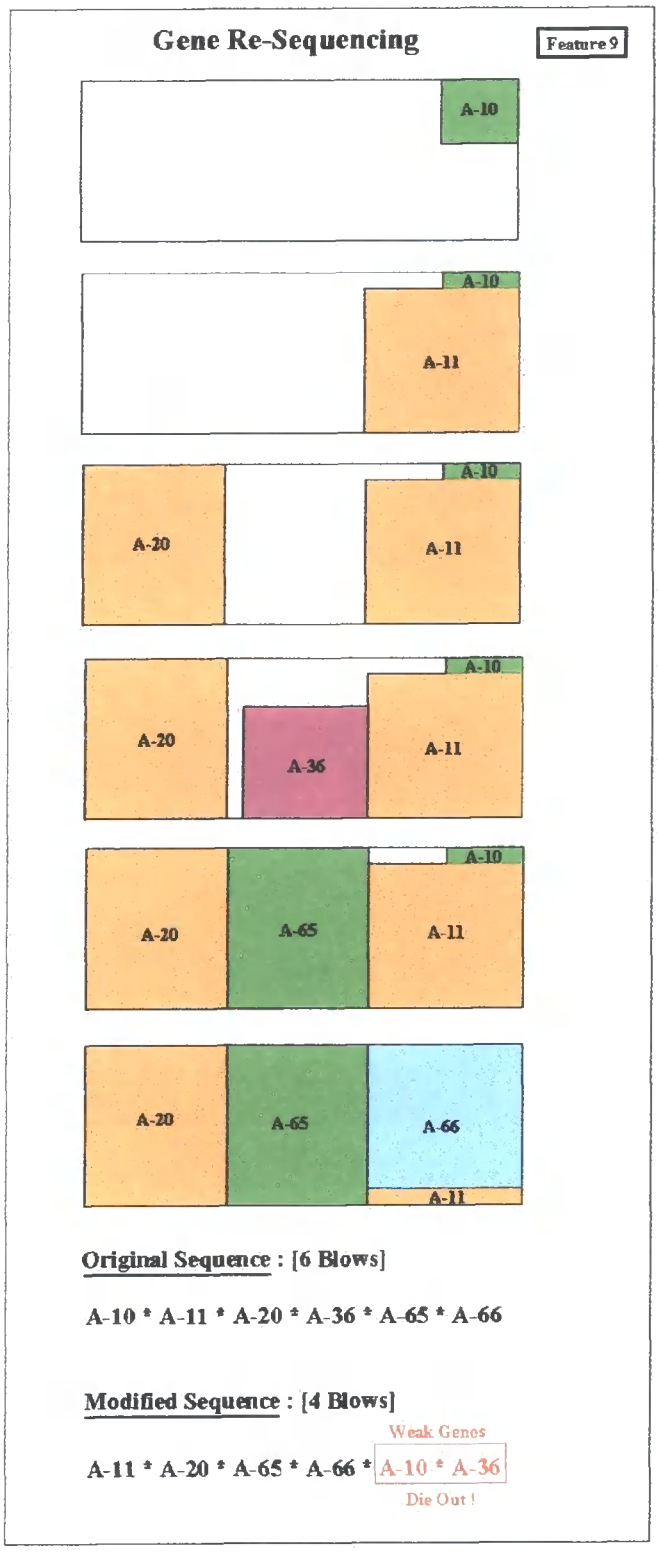


Figure 6.43. Feature 9 – Gene Re-Sequencing

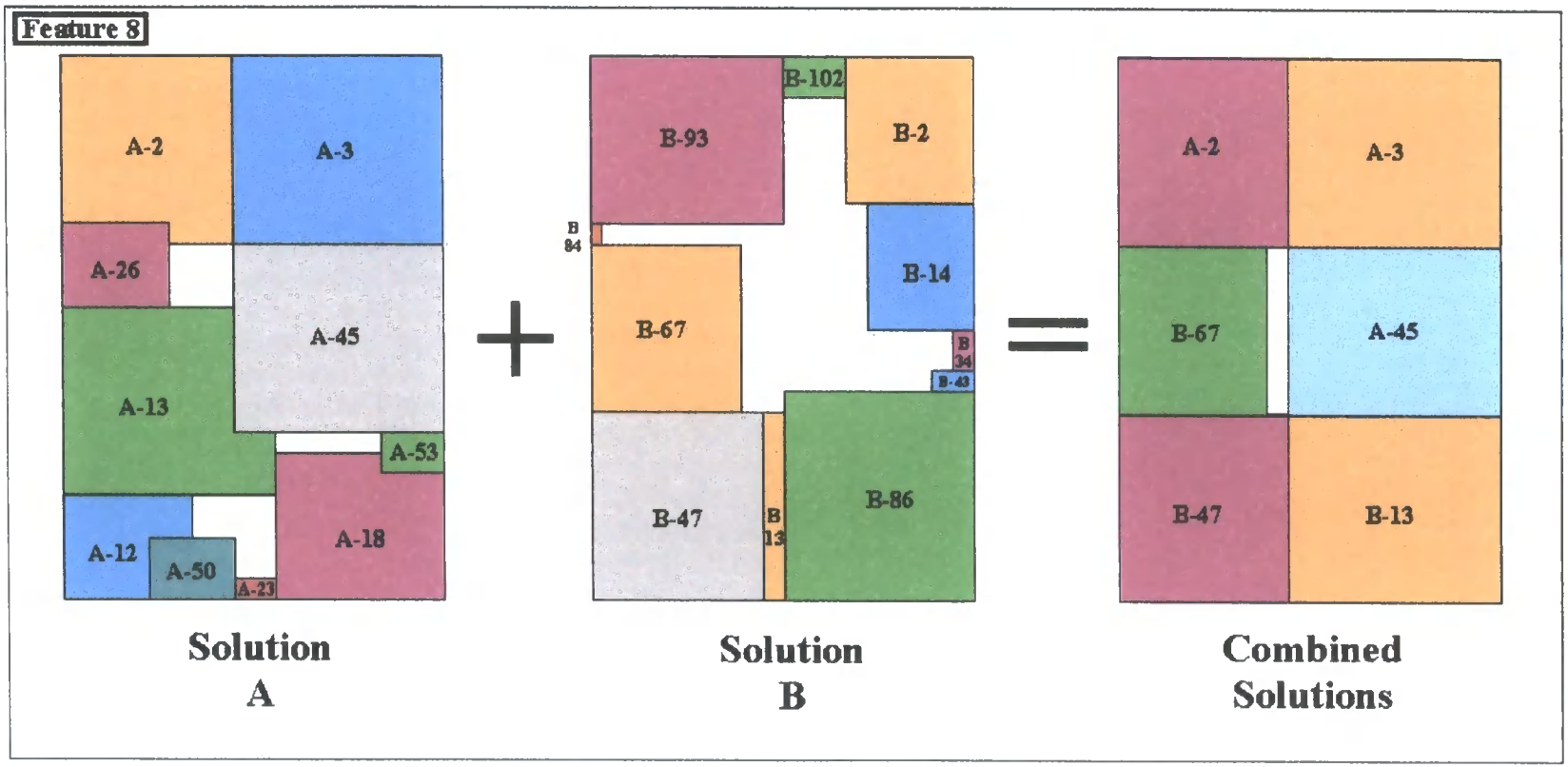


Figure 6.44. Feature 8 – Gene Mixing and Re-Sequencing

Figure 6.45. Case Study 9 – Final Solution

Compared to the 52 blows:

Feature	Number of Blows
1	4
2	2
3	**6**
4	6
5	**8**
6	8
7	**8**
8	6
9	4

As can be seen from above, three features, namely features 3, 5 and 7 are the cause of these extra 6 blows. However, if the uniform tool loading problem, see Figure 6.46., is to be considered then the solution of less 6 blows is not of that great advantage considering the concerns of tool wear and short tool life. It can be argued hear that for small and medium batches these extra 6 blows will not cause a major production time delay.

It has been recognised that in cases where there is no "one-to-one" matches, see Figure 6.41. and Figure 6.42., the greater the number of runs the smoother the curve becomes, see Figure 6.47., page 158. Figure 6.47 shows the distributions for 100, 200, 300, 650, 750, 850, and 1000 simulation runs, plotted as an individual graph. Figure 6.47 clearly supports the argument of a two pass approach. In this approach a first search is undertaken which looks for "one-to-one" matches, thus, the spread of the distribution

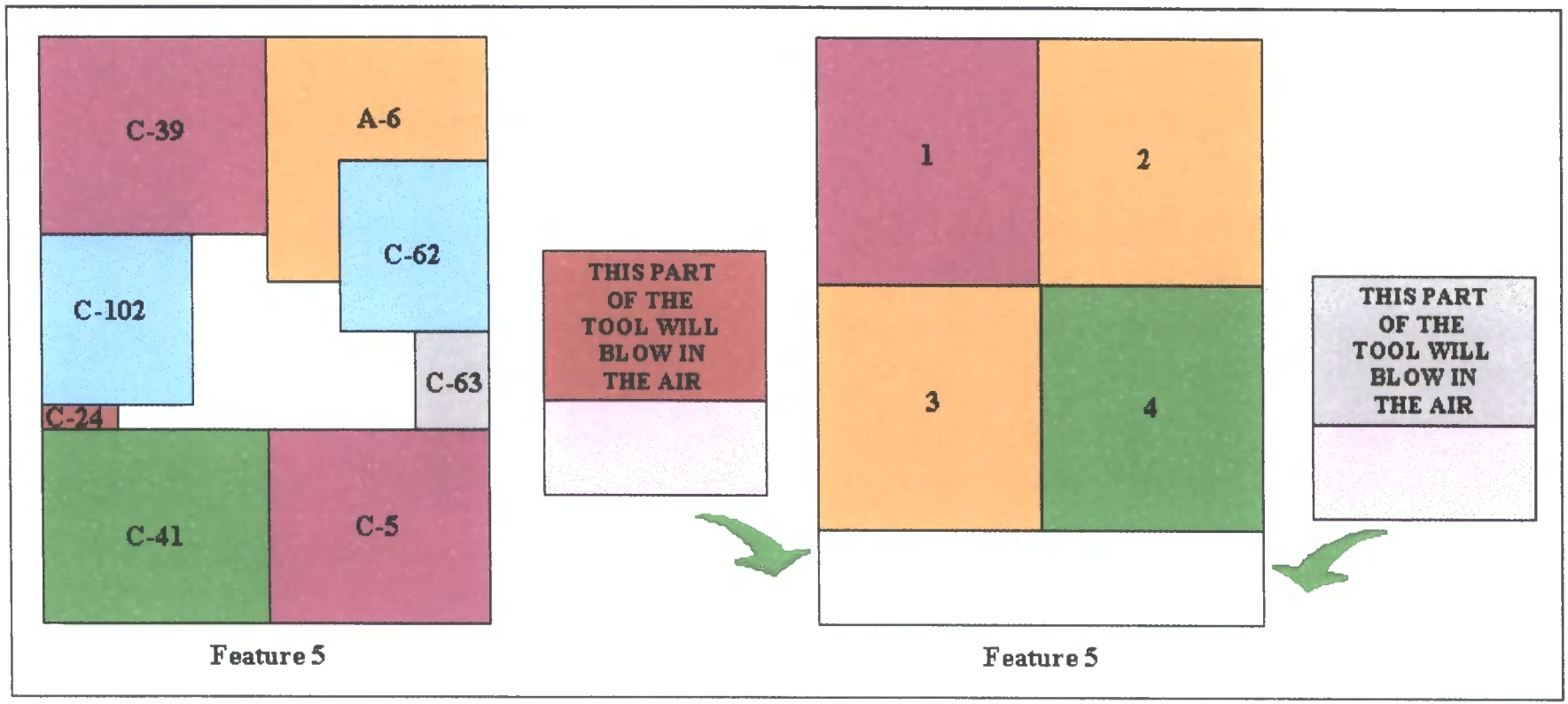


Figure 6.46. Feature 5 – Tool Loading

will be considerably reduced and the search time to achieve a near to optimum solution will be reduced.

6.4. DISCUSSION

This section, divided into 4 sub sections, discusses a number of related topics in the undertaken study.

6.4.1. SAMPLE SIZE

In establishing an acceptable sample size, the "Law of Large Numbers" [107], has been discussed. However, the "Law of Large Numbers" cannot predict directly how many trials are needed to guarantee that a sample mean for the cost function value distribution will be close to the whole population mean for the cost function value distribution (μ). The number of trials needed depends on the variability of the cost function distribution. Hence, the more variable the distribution, the more trials are needed to ensure that the trial mean is close to the whole distribution mean (μ). In general terms, within an accepted sampling error, the mean ($\bar{x}$) for a number of random observations will eventually approaches the mean (μ) of the population. Hence, the required sample size depends on two factors [108], the degree of tolerated sampling accuracy and the extent of variation in the population in regard to the key characteristics of the study, in this work, different feasible tool-combinations. It is worth mentioning here that for accuracy of sampling, the size of the population is largely irrelevant and what is important is the absolute size of the sample [108]. The only exception [108], is when the sample size represents a sizeable proportion of the population (e.g. 10%). In such cases even a slightly smaller sample is equally accurate because the global population is already well described.

It is then clear that before deciding on the size of the sample, it is necessary first to decide on how much sampling error will be tolerated. From large number theory it can be seen that an increase in sample size can lead to increased accuracy. However, with larger samples, increasing the sample does not necessarily give the same payoff in terms of accuracy, that is beyond a certain point the cost of increasing the sample size is not worthwhile in terms of extra precision achieved. Thus, the desired accuracy is not the only factor in working out the sample size; cost and time are also key factors [108]. Therefore, the final sample size will be a compromise between cost and accuracy.

The second factor that has an affect on sample size is the extent to which there is variation in the population, in this work, different feature-tool combinations. Table 6.2 lists the required sample sizes depending on the degree of accuracy and the estimated population variation for the key study variables. As can be seen from Table 6.2, the greater the diversity the larger the sample size should be. However, in the tool selection problem the split or variability of output might not be known or could not even be predicted. In tool selection, the variation can be very large, especially with larger tools and features populations, as there will be many possible tool-combinations. The use of the table can be illustrated by means of an example. Although the actual distribution of cost function values over a number of runs is unknown, assuming that the maximum cost function value for manufacturing a particular feature was 22, and the minimum cost function value was 2, then the spread of possible solutions would be 20. For any particular run a solution would be a whole number between 2 and 22 therefore the percent of population expected to give a particular answer would b 5 or 95. That is, 5% of the population would give the specified cost function value and 95% would give a different value. Thus from Table 6.2. a sample size of 76 will give a sample error of 5%. This example would represent a very simple case of tool selection. Under more realistic

tool selection conditions the population homogeneity would be lower than given in the table and the results shown in Figure 6.15., Figure 6.18. and Figure 6.21. for example show spreads of 43, 38 and 43, respectively. These cases would, therefore, have a percentage of population of a particular answer of 2 or 98, 3 or 97 and 2 or 98, respectively.

Acceptable sampling error[a]	Per cent of population expected to give particular answer					
	5 or 95	10 or 90	20 or 80	30 or 70	40 or 60	50/50
1%	1 900	3 600	6 400	8 400	9 600	10 000
2%	479	900	1 600	2 100	2 400	2 500
3%	211	400	711	933	1 066	1 100
4%	119	225	400	525	600	625
5%	76	144	256	336	370	400
6%	—[b]	100	178	233	267	277
7%	—	73	131	171	192	204
8%	—	—	100	131	150	156
9%	—	—	79	104	117	123
10%	—	—	—	84	96	100

Notes:
a At the 95 per cent level of confidence
b Samples smaller than this would normally be too small to allow meaningful analysis

Table 6.2. Required sample sizes on population homogeneity and desired accuracy [108]

By inspection it can be seen that the extrapolation of the data in Table 6.2. to cover more percentages would predict that a sample sizes of 100 would give sampling errors in the region of a few percent, say less than 4%. For all practical purposes such errors would correspond to insignificant differences in cost function values. In addition, the table also shows that an increase in sample size of, say, 100 to 500 can decrease the sample error. However, as has been shown, a sample of 100 already gives a small sampling error and further reduction is probably not necessary. Once again, as has been mentioned earlier, interest in the design process will be in ranges of good solutions rather than an exact value for a cost function, therefore once again a sample of 100 is justified.

6.4.2. DISTRIBUTION PATTERNS

The case studies show some interesting characteristics regarding the distribution of the blows (cost function values) required to complete a product. These distributions can be considered as "combinations of distribution" relating to distributions associated with individual, as well as, combinations of features. For example, features that can only be produced by single blows of tools and are specific to that particular feature shape give a single point solution, that is, the number of blows equals the number of features, see case study 1, section 6.3.1. At the other end of the spectrum lies a single point solution. Here, a single nibbling tool is used to cut out the profile of all the features, see case study 4, section 6.3.4. In the "simple" cases considered in this chapter and for all the practical cases, the solution for the number of blows will be a combination of the above cases. That is, in some instances a single tool choice may punch out a single feature and small tools will be used to nibble out the remaining features using a variable number of blows. The extent of the dominance of any one or number of "one-to-one" matches will depend upon the nature of the punching task and the tools available.

For example, in the case of a few features and a few tools, if "one-to-one" matches exist, these distributions are not likely to be smooth or normal. The effect will be that any one fortuitous match will dominate the distribution with a small surrounding distribution. This type of case is further complicated by the influence of the relative sizes of the features. That is, if a "one-to-one" match is found for a large feature, hence, large tool, the distribution of blows for the remaining features with small tools is likely to be narrow. In contrast, if the "one-to-one" match requires a small tool to punch a small feature then there will be a large number of ways for the remaining tools to punch out the remaining features, hence, the spread will be wide. However, in practical cases the number of features will be large and although some features will be punched out by

dedicated tools there will be a wide variation in non-specific tools required to punch out the remaining features. Hence, for large populations it is reasonable to consider the distribution to be normal.

A comparison of distributions in Figures 6.38 and 6.40. gives an interesting insight into the effect of dominant "one-to-one" matches and leads on to the proposal of a search methodology that will be more efficient than the application of the Monte Carlo method alone, see section 6.4.3. Figure 6.38 shows the distribution curve associated with the features and tooling shown in Figure 6.37. In this case the matching of the (300 x 300) tool to the (300 x 300) feature requires only a single blow. The remaining features can be punched out by as little as three further blows (four in total) and as many as six additional blows (seven in total). Similarly, if the "one-to-one" match is a small (20 x 10) feature then the remaining spread ranges from four further blows up to maximum of 15 blows. If the "one-to-one" match is the intermediate (20 x 20) shape then the blow distribution will have a spread of between four and eleven. These distributions combine to give the two peaks shown by the distribution curve in Figure 6.38.

However, if one of the features and its matching tool was removed from consideration, for example, by a preselection process, then it would be fair to assume that one of the peaks in Figure 6.38 would be removed, a single peak would remain. This has been found to be the case, as shown in Figures 6.40.

Finally, Figure 6.47. shows the results of a more complex case. The results are difficult to interpret for a few simulation runs but as the number of runs increases the curve develops and appears more closely representative of a normal distribution curve. For a first approximation this may be a reasonable conclusion to draw.

The case studies considered above show that a large number of simulation runs may be required in order to improve the chance of obtaining a near optimum solution. However, if a first search is made looking for "one-to-one" matches, then, the spread of the distribution will be considerably reduced and the search time to achieve a near to optimum solution will be also reduced. This two-pass approach leads to a hybrid search which uses rule based intelligent choices in a series of passes followed by a Monte Carlo approach to "mop up" the difficult features that require multi-punching strokes.

6.4.3. THE POTENTIAL FOR INTELLIGENT SEARCHING

The use of this investigation of the Monte Carlo approach for selecting the best set of punching tools, as mentioned in Chapter 4, was to establish a base line for selection efficiency and effectiveness. However, the case studies showed that the effect of dominant "one-to-one" matches caused distributions not to be smooth or normal. As a consequence, a large number of MC simulation runs may be required in order to improve the chance of obtaining a near optimum solution. This naturally leads on to the need of a search methodology that will be more efficient than the application of the Monte Carlo method alone. The introduction of a more rational, heuristic or rule based search may achieve a better solution than the use of the Monte Carlo method alone or may achieve the optimum solution in a shorter time. This section discusses three well known searching techniques namely, “Hill-Climbing”, “Simulated Annealing” and “ Genetic Algorithms”. Although these methods are often quoted as generic techniques within optimisation theory, the way each of these three search methods operate differs from one method to another. Hill climbing methods, for instance, concentrate on local conditions to seek the nearest local optimum. Hill climbing works best for smooth functions with a single optimum. Hence, the utility of hill climbing can be greatly reduced if it is used for the optimisation of functions with numerous local optima. On

the other hand, simulated annealing is used to avoid this draw back of hill climbing and deals with situations where the search becoming locked in local minima. Simulated annealing is a strategy that allows a search to jump out of local minima and continue the search to the minimum or a low sub optimum. Finally, genetic algorithms are used to improve the efficiency of searching or optimisation by using knowledge gained in the early part of a search. The method steers the search towards areas where there is a good likelihood of finding an optimum solution. The following sections explain the way these methods relate to decision tree searching and tool selection in particular.

6.4.3.1. Hill-Climbing Methods

Traditional hill-climbing methods (gradient descent, Quasi-Newton methods, and simplex method) start with an initial solution or guess and proceed to find the nearest maximum (or minimum) solution. They mainly concentrate on local gradients and seek the first local optima within the search space, focusing solely on precision and computation time at the expense of reliability [110]. Hill climbing methods will always find the best answer if (a) the function being explored is smooth, and (b) the initial variable values place you near the optimal solution. If either condition is not met, hill climbing can end up at a local solution rather than a global solution. Therefore, the use of "hill climbing" methods alone is not likely to bring a substantial improvement on the Monte Carlo Methods described here.

6.4.3.2. Simulated Annealing

Simulated annealing tries to avoid the draw back of hill climbing and is not forced to focus the search within the local neighbourhood. Simulated annealing simulates the slow cooling of a physical metallurgical system, so if the temperature decreases sufficiently slowly, then the probability of an "atom" being in a global optimum tends to certainty [111, 112, 113]. The idea behind simulated annealing is that there is a cost

function which associates a cost with each state of the system. The method works using "hill climbing" to find a minimum. However, the search is able to proceed beyond the first minimum because of an "energy" allowance that allows the search to move to a search space position with a higher cost function value. As the search proceeds the "energy" allowance is progressively reduced until the allowance is insufficient for the search to climb out a minimum value at which point the search is terminated.

In tool selection, if for example, the final decision point that leads to a final solution required a tool change then it may be worthwhile fort searching to be returned to an earlier decision level. That is, moving back through a decision tree and commencing searching along another branch, searching for a tool that can complete the job without the need for a tool change. It would be wise to do this search before proceeding into the final decision level because whichever tool is selected at the final decision level, a finishing blow will still be required. In this way, moving back in the decision tree is similar to SA. As with other applications of SA a convergence strategy is required if the search is not to be exhaustive. In the application of SA to this case the movement out of local minima can be seen as a jump back up the decision tree. In order to converge the search the extent of movement back up the decision tree could be reduced as the search progressed. Thus it is suggested that SA can be combined with a MC approach as illustrated below. Assume that a MC search finds a solution, see Figure 6.48.

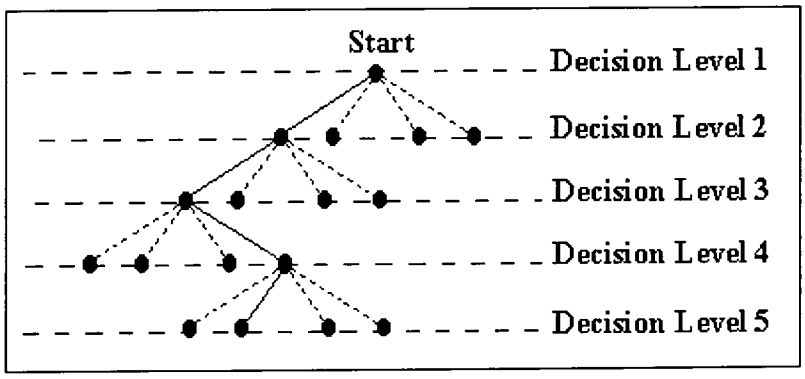


Figure 6.48. Monte Carlo-Simulated Annealing combined search.

This solution is assumed to be sub optimal and so the SA technique moves back up the decision tree to an earlier decision level, say level 1. Using the MC search strategy again the search randomly makes a selection at level 1 and proceeds with a MC search down the decision tree to a solution. If the second solution produces a cost function value which is considerably less that the earlier search then the second decision at level 1 is consolidated into the decision sequence. If the first search is still found to have the lowest cost function value then the first decision at level 1 is consolidated. The above process may be repeated several times depending upon the SA convergence strategy. For example there could be five MC searches. From these five solutions the search (tool selection) with the lowest cost function value is selected. Using this decision tree route the SA strategy moves to decision level 2. Once again MC searches are made from level 2 downwards. The number of MC searches made at level 2 could be the same or higher/lower depending upon the convergence strategy. Assuming the number of searches remain the same then, say, five MC searches would be made at decision level 2 and the route leading to the lowest cost function values would be selected for further exploration. In this way the MC-SA search strategy would move down the decision tree levels. Although this approach cannot guarantee finding the global optimum it does tend to focus the search in areas where decisions are more likely to find low cost function values.

The one-to-one matching described earlier by the author, [6], is in fact a particular variant on the above SA approach but where the matching of tools to features is by inspection rather than by random selection. Once all of the one-to-one matches have been used the remaining "mopping" up of features is completed by MC selection which may or may not continue to use SA.

6.4.3.3. Genetic algorithms

In tool selection the fundamental problem is the searching of a decision tree where only the final solution can be used to assess the individual route performance. The inherent difficulty of using genetic algorithms is the fact that intermediate objective function values give little indication of the final cost function value or even the likelihood of success of the search. The idea here is to select optimum or near optimum solutions to tool selection without constraining or prejudging the selection procedure.

The approach is to consider the acceptable matching of a tool edge to a feature edge as a gene. Thus, a sequence of genes could represent a solution which is equivalent to a viable chromosome. So, for example a viable chromosome may consist of a string of genes represented by letters; as follows,

A-B-C-D-E-F.

Although it may be known that each of these genes makes its own contribution to the punching of a part it is not known which genes make a large contribution (strong gene) and which genes make a small, or even a redundant "contribution". Two different techniques are suggested for future investigation.

The first assumes that a good solution has been generated, perhaps by an MC search. Then by performing what could be termed "gene manipulation" the current gene sequence could be changed to seek a better solution, perhaps even searching all the different possible gene sequences. Each newly generated sequence could be evaluated by calculating its cost function value. The idea here is that eventually the bad genes, for example, redundant blows, would come at the end of the gene sequence where they can be removed out of the sequence leaving the good genes as an improved solution. The diagram in Figure 6.49. illustrates the progress in punching out a feature using a sequence of tool edge-feature edge matches. From this sequence it can be seen that one

of the punch blows, represented by gene (B) is made redundant by a subsequent punch blow (C). At the time of the punch blow (B) decision the use of punch blow (C) had not been decided. The number of blows in forming the required feature is obviously equal to the number of genes in the sequence. Redundancy in the gene sequence can be removed by reordering the gene sequence, that is, keeping the same genes but changing the sequence order. For example, in Figure 6.49, if the first successful sequence was

A-B-C

then the sequence could be reordered to

B-C-A

or A-C-B

or C-A-B

etc

etc

There are six ways of writing this sequence. However, in searching these combinations it will be found that those sequences that move redundant or duplicated blows to the back of the sequence will actually stop earlier because the punching task has been completed. Keeping the biological analogy it can be considered that unneeded genes at the end of a chromosome will atrophy and are thus removed. In this way a simple list manipulation technique can be used to remove any redundancy from an already successful, randomly generated, blow sequence. Figure 6.50., shows another example with an initial sequence A-B-C-D-E. Note that a re-sequence of this order could be A-E-D-[2 redundant genes B and C].

The other possible approach for implementing a genetic algorithms is to import a good gene from another different gene sequence i.e., the use of effective tool edge-feature

edge combinations. For example by randomly matching tool-edges to feature-edges several solutions may be found, see Figure 6.51.

A-B-C-D

and A-E-F-B

The approach suggested here is to create a combined gene pool A-B-C-D-E-F. This pool contains redundant blows but redundancy can be easily removed by gene order manipulation, as described above, to give the solution A-D-[redundant blows].

The method of pooling genes can be cascaded through generations of chromosomes. That is, an improved chromosome produced by using the technique above can be gene pooled with another improved chromosome, produced in the same way. Then the resulting redundancy can be removed to produce a second generation improved chromosome. Thus, this combination of the MC. method for randomly generating chromosomes and gene pooling and sorting to breed improved chromosome performance is an attractive computational method for selecting optimum or near optimum solutions to tool selection without constraining or prejudging the selection procedure.

At this point of research further investigation into combining genetic algorithms and/or simulated annealing would require more time than was available. Hence, this work is left for future investigation. The author is already in the process of publishing these additional investigations.

6.4.4. THE POTENTIAL FOR IMPROVED COMPUTATION

It is worth mentioning here, the simulation programme used for the work was written in visual basic for Microsoft Excel, which is an interpreted programming language rather than a compiler. Also, this programme was designed to read and write every entry and

computed value to a working spreadsheet rather than to do these operations directly within the CPU. This simple approach to computation was used in order to trace the program and to check the correctness of the program. The first aim was to establish trust in the program and ensure that it did the computation correctly. In this way it was possible to establish the feasibility of the use of the Monte Carlo simulation approach. This simple software approach of course has a negative implications on the computational time. However, since the credibility of the approach has now been established, object oriented approaches and different programming languages such as C++, could now be used to reduce the current computational time by a factor of 100 to 1000. Thus, for example, if within the current version of the program it takes 24 hours to finish a certain number of simulation runs, then with improved programming it would take only 2 to 15 minutes. This obviously would have a great impact on the practicality of such an approach when used at the design level.

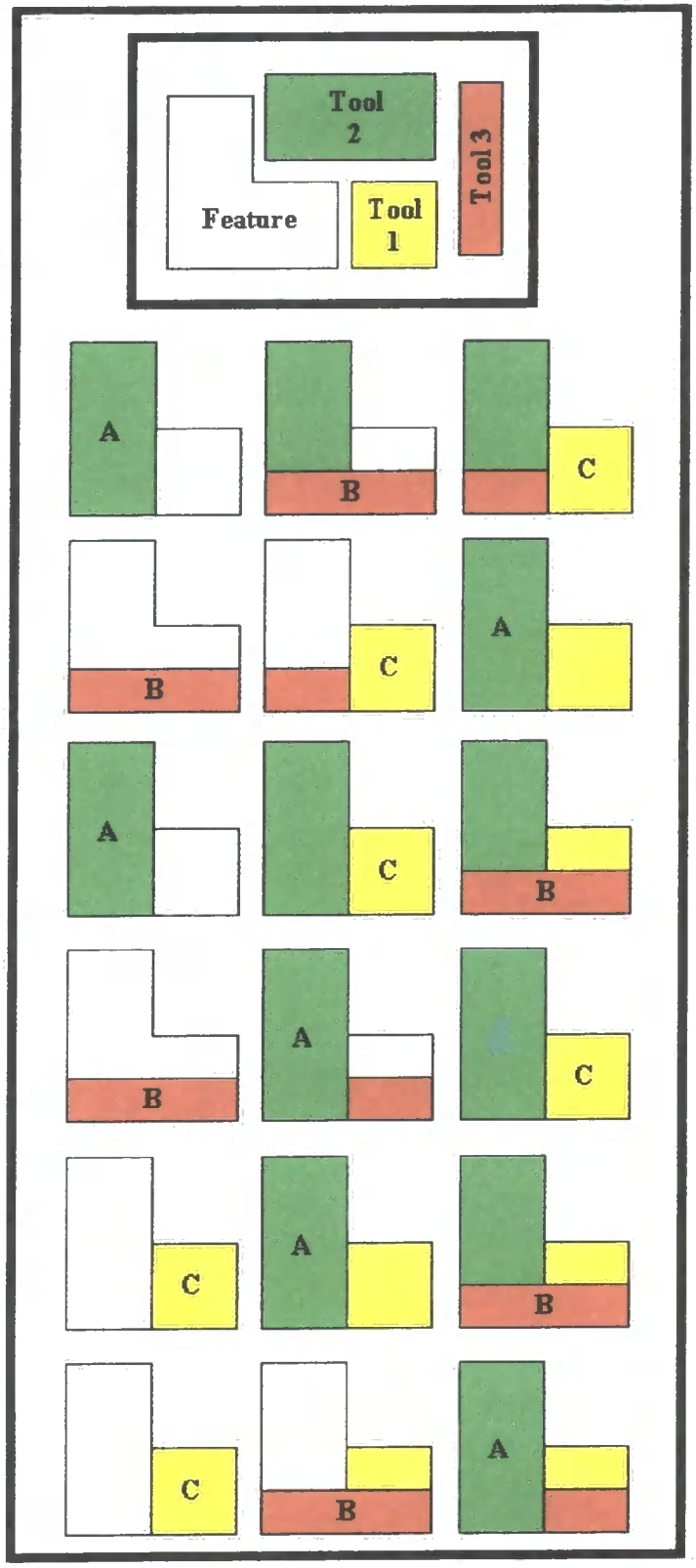


Figure 6.49. Example 1 – Monte Carlo Genetic Algorithms Approach

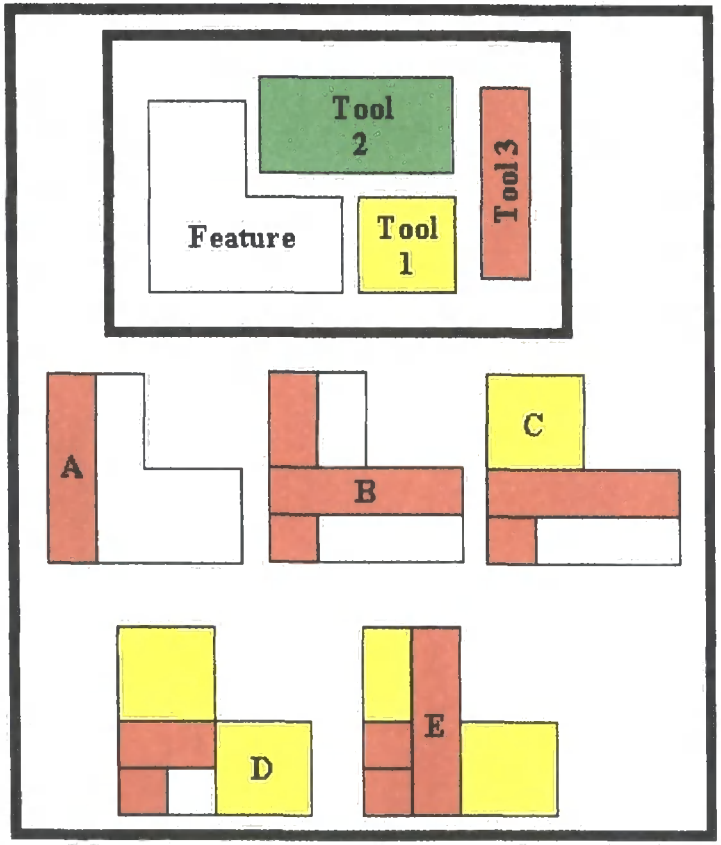


Figure 6.50. Example 2 – Monte Carlo Genetic Algorithms Approach

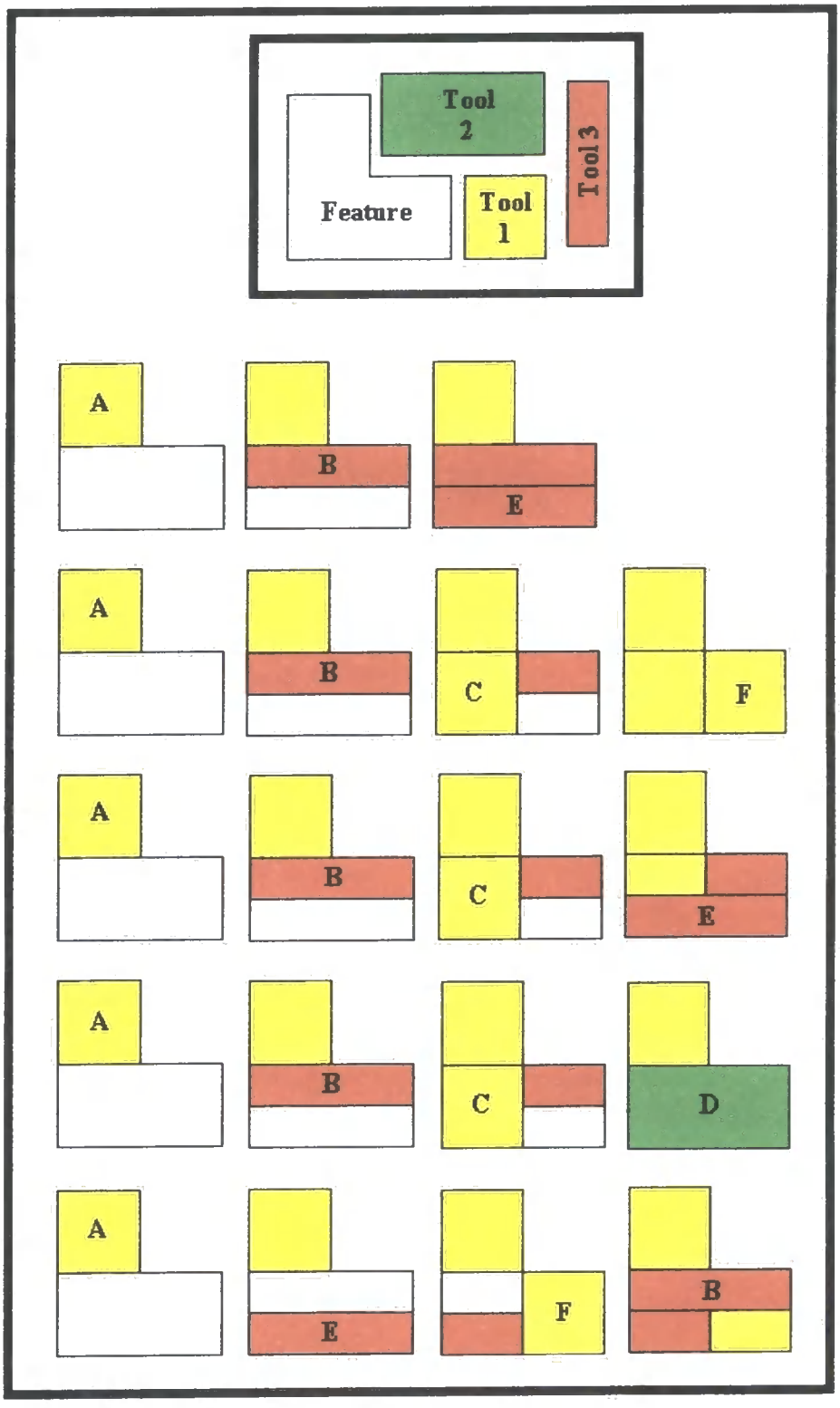


Figure 6.51. Example 3 - Monte Carlo Genetic Algorithms Approach

Figure 6.6. - Variation in the Mean

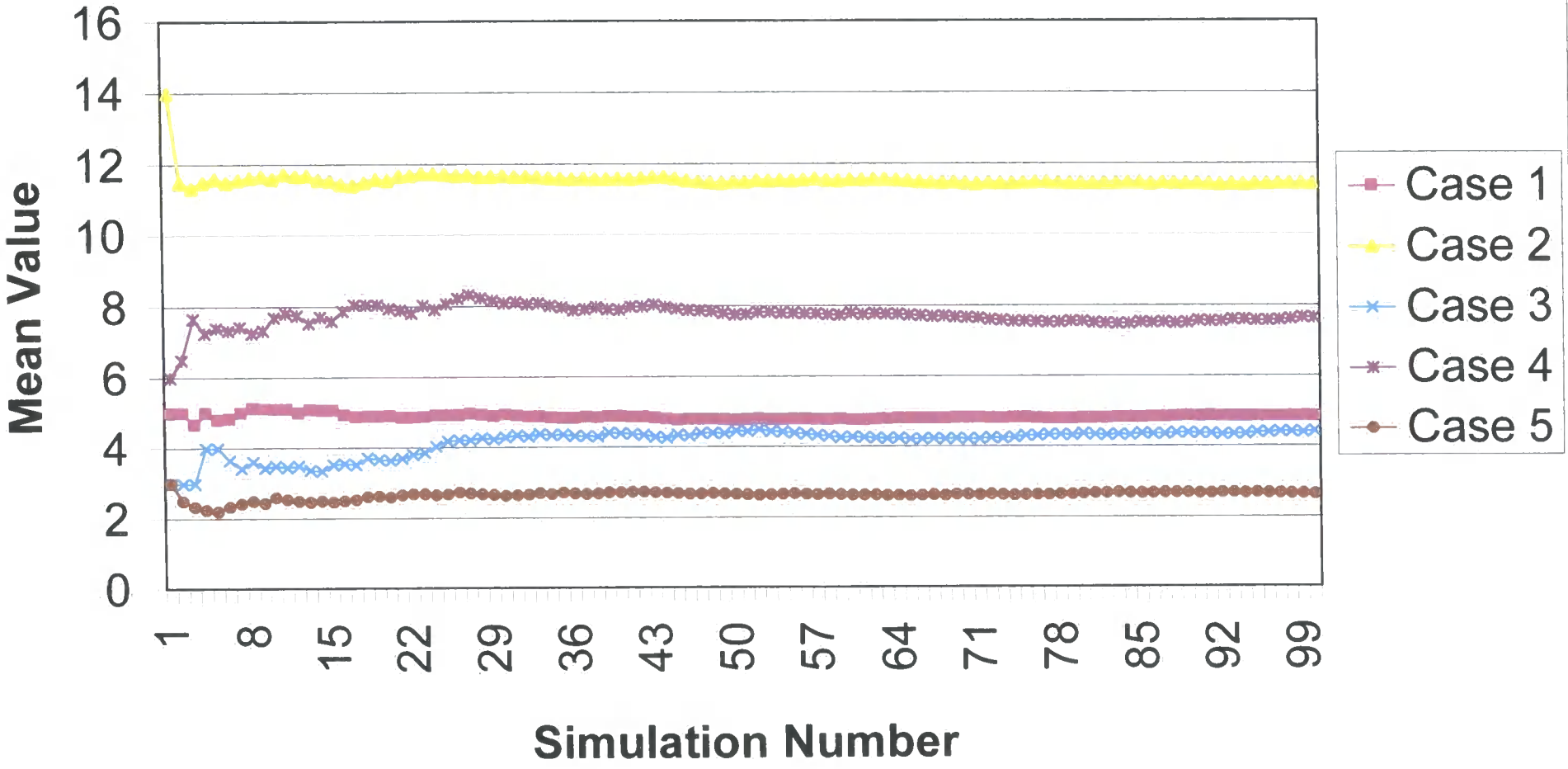


Figure 6.7. - Variation in the Standard Deviation

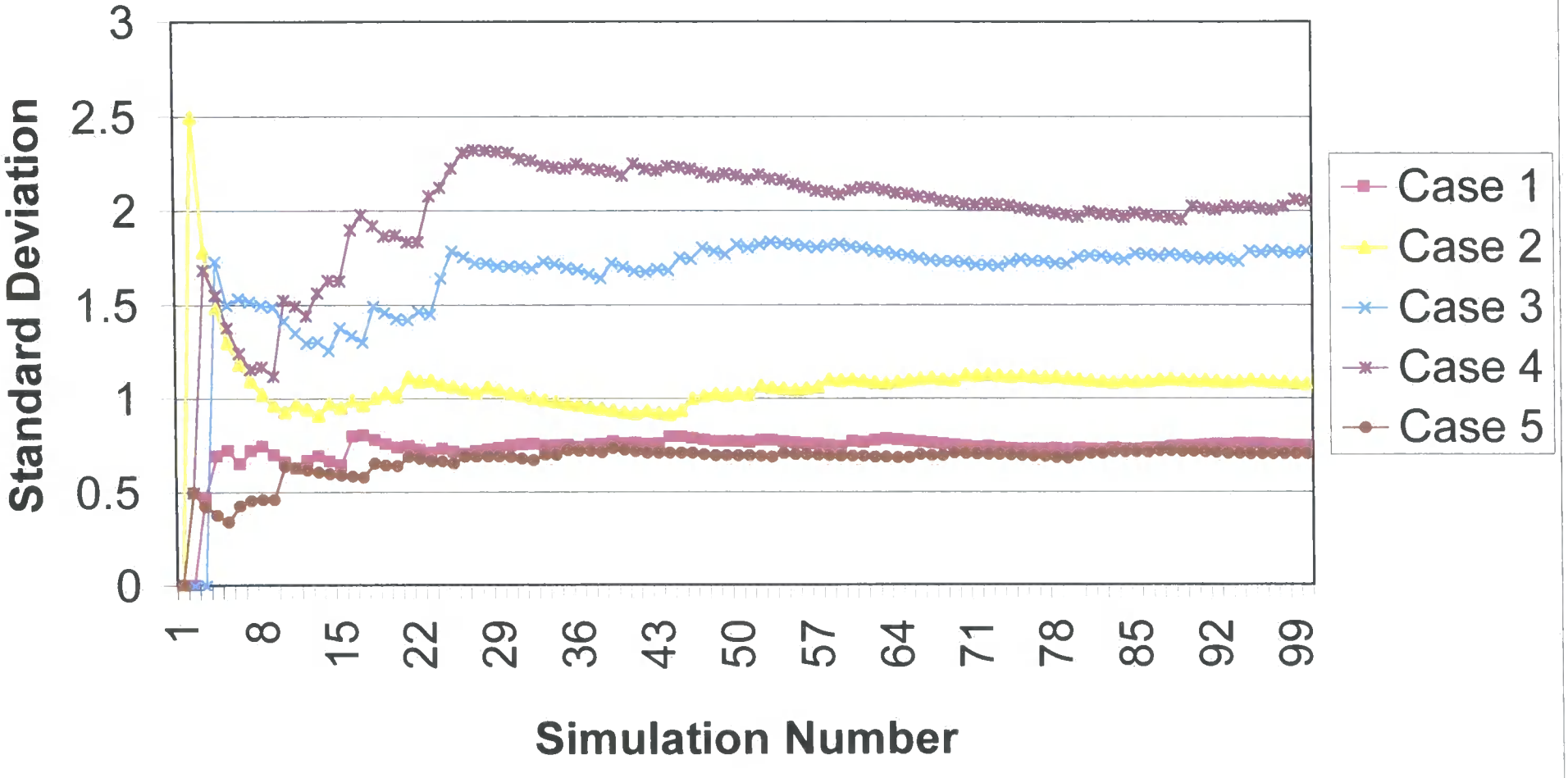


Figure 6.8. - Case 1 [100 Simulation Runs] Results

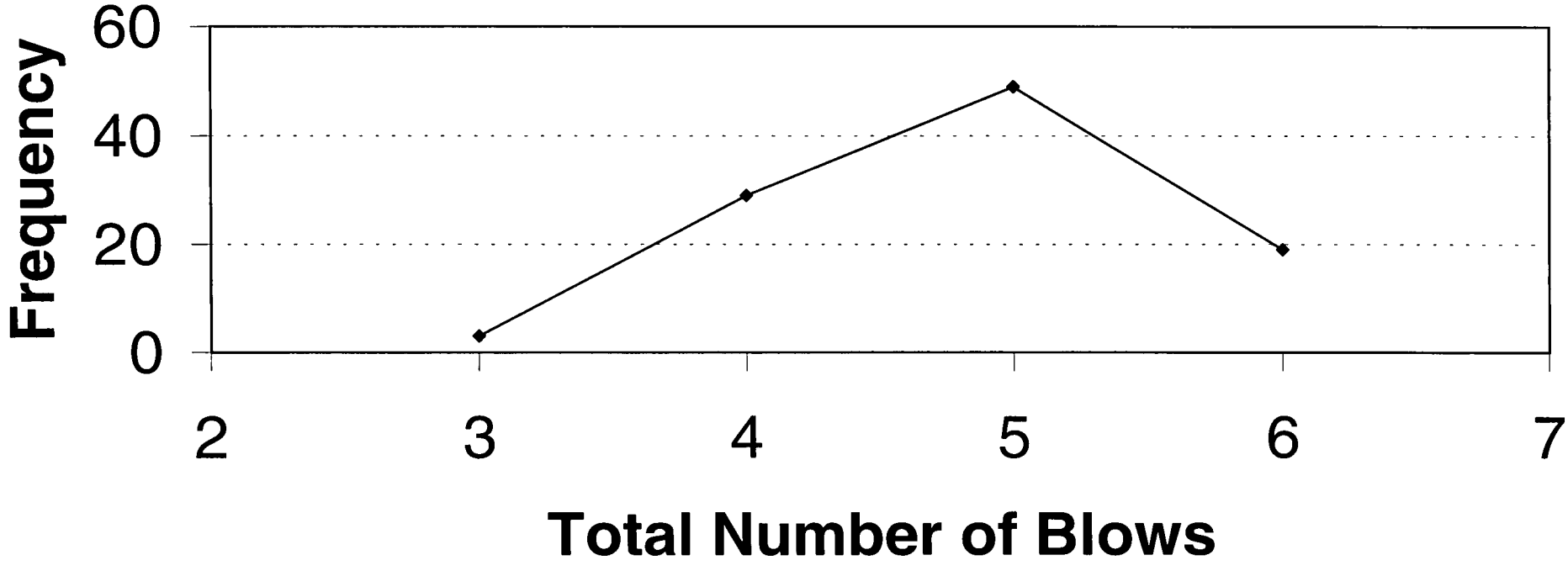


Figure 6.9. - Case 2 [100 Simulation Runs] Results

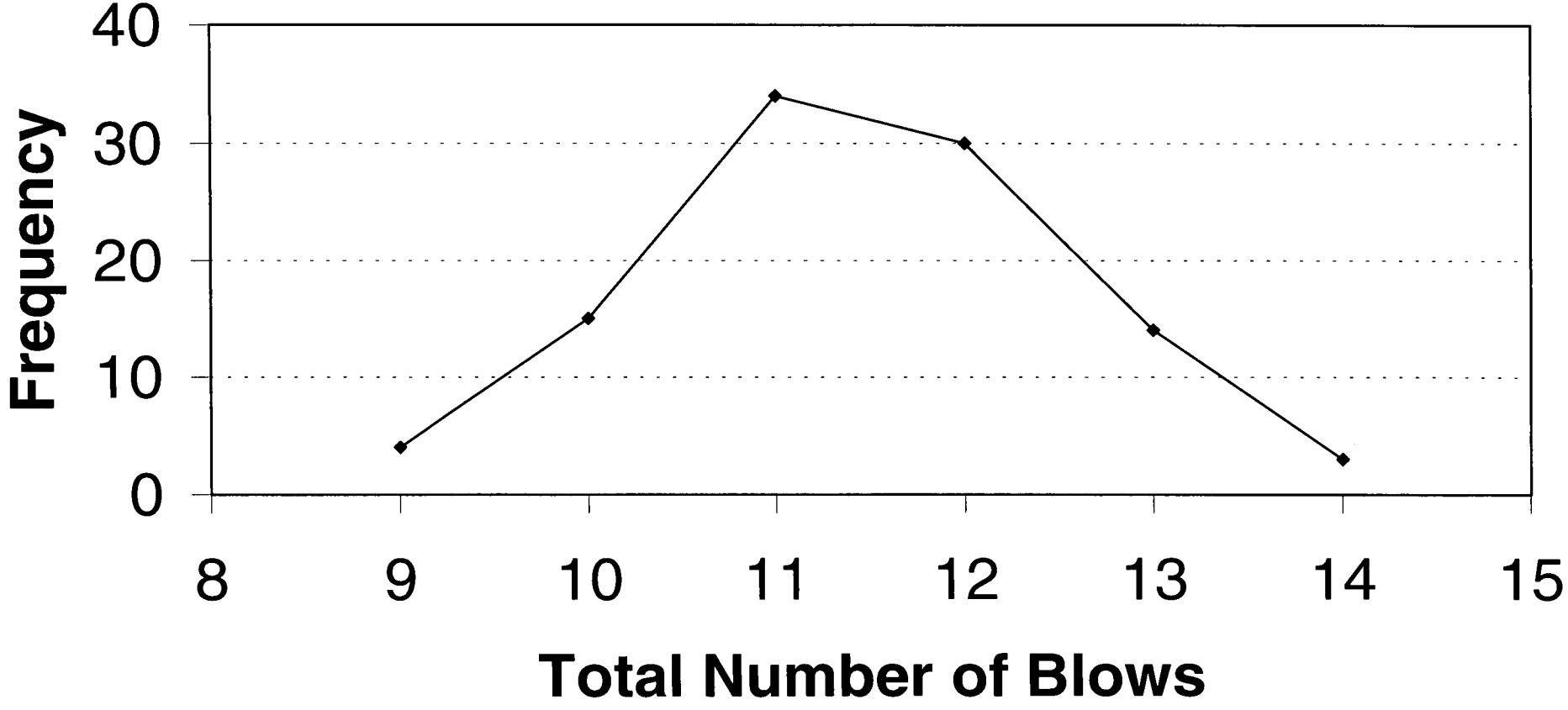


Figure 6.10. - Case 3 [100 Simulation Runs] Results

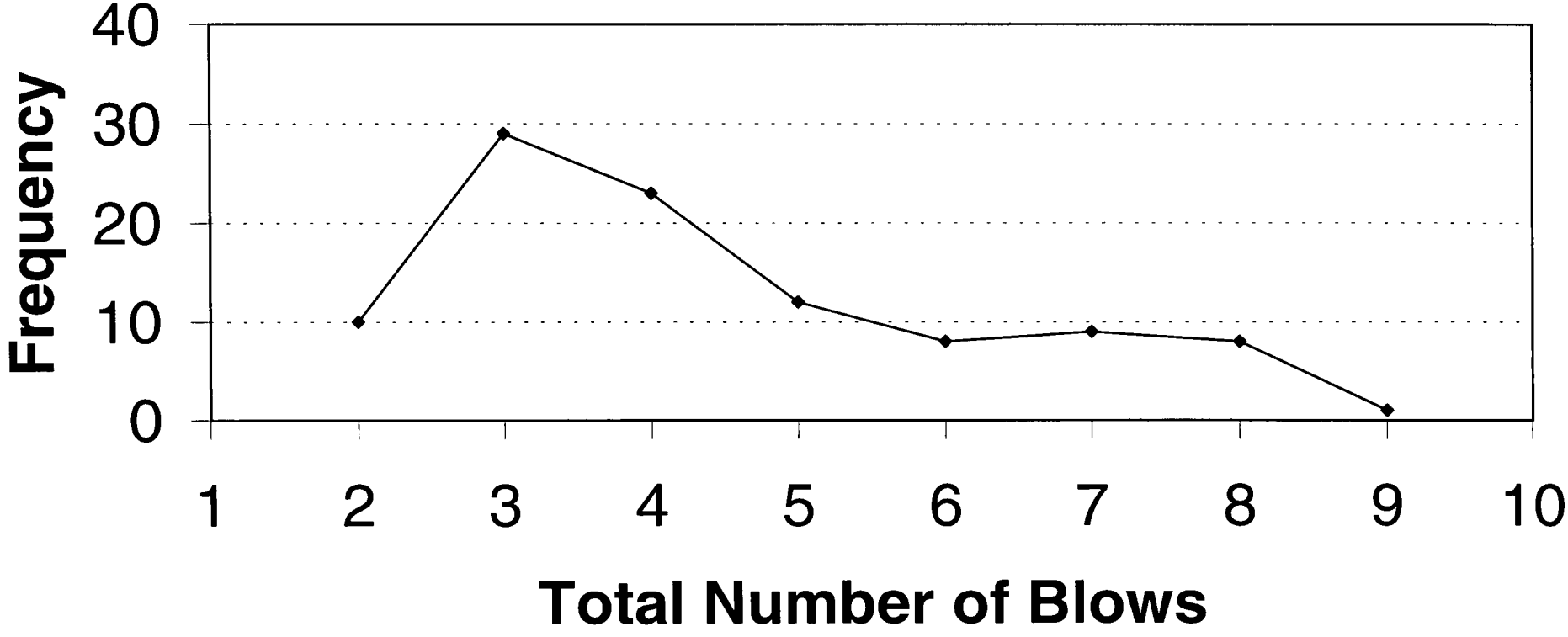


Figure 6.11. - Case 4 [100 Simulation Runs] Results

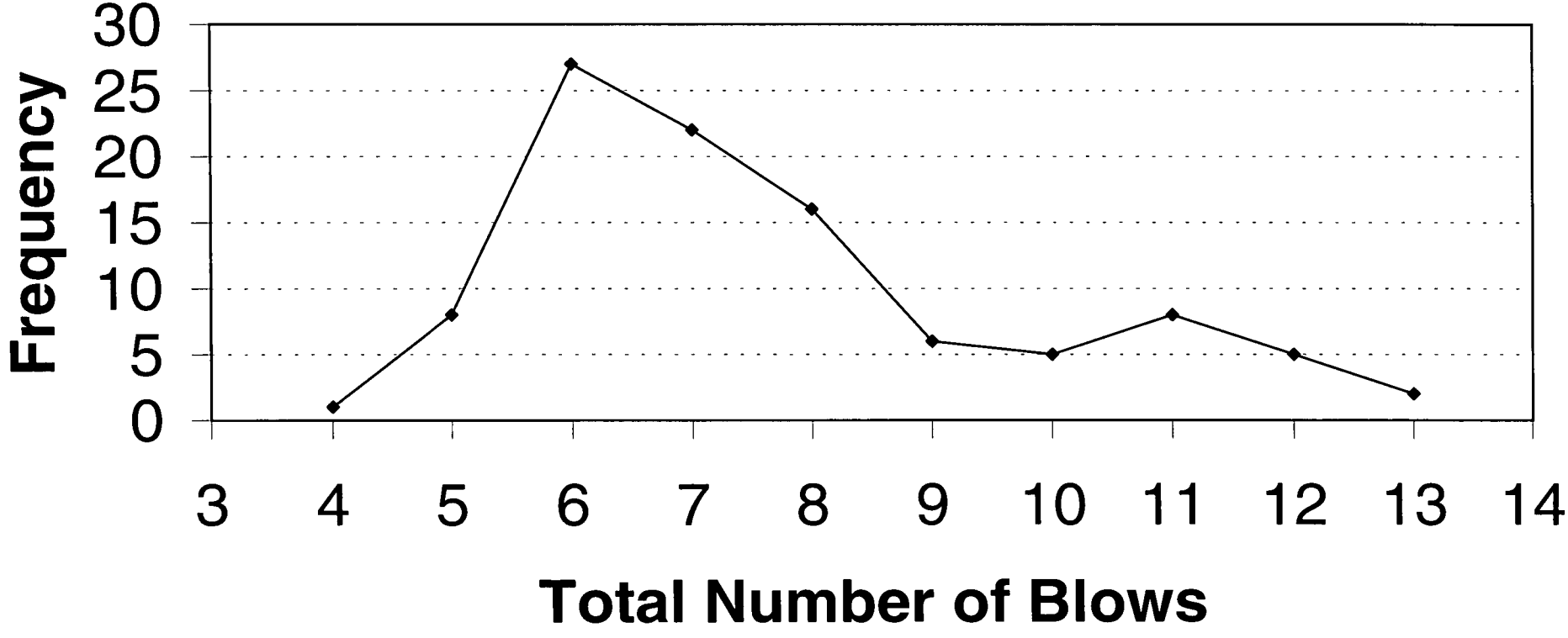


Figure 6.12. - Case 5 [100 Simulation Runs] Results

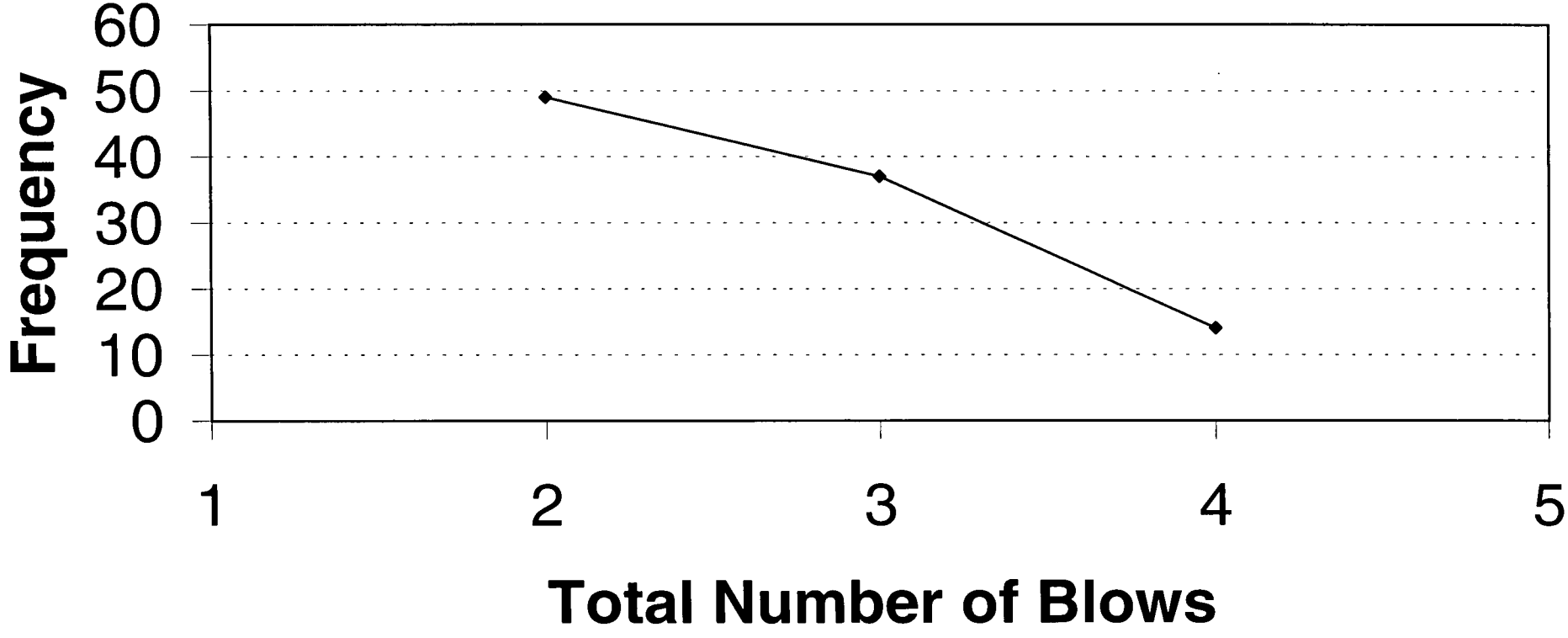


Figure 6.14. - Case Study [Figures 6.13 A & B]

100 Simulation Runs - Trial A

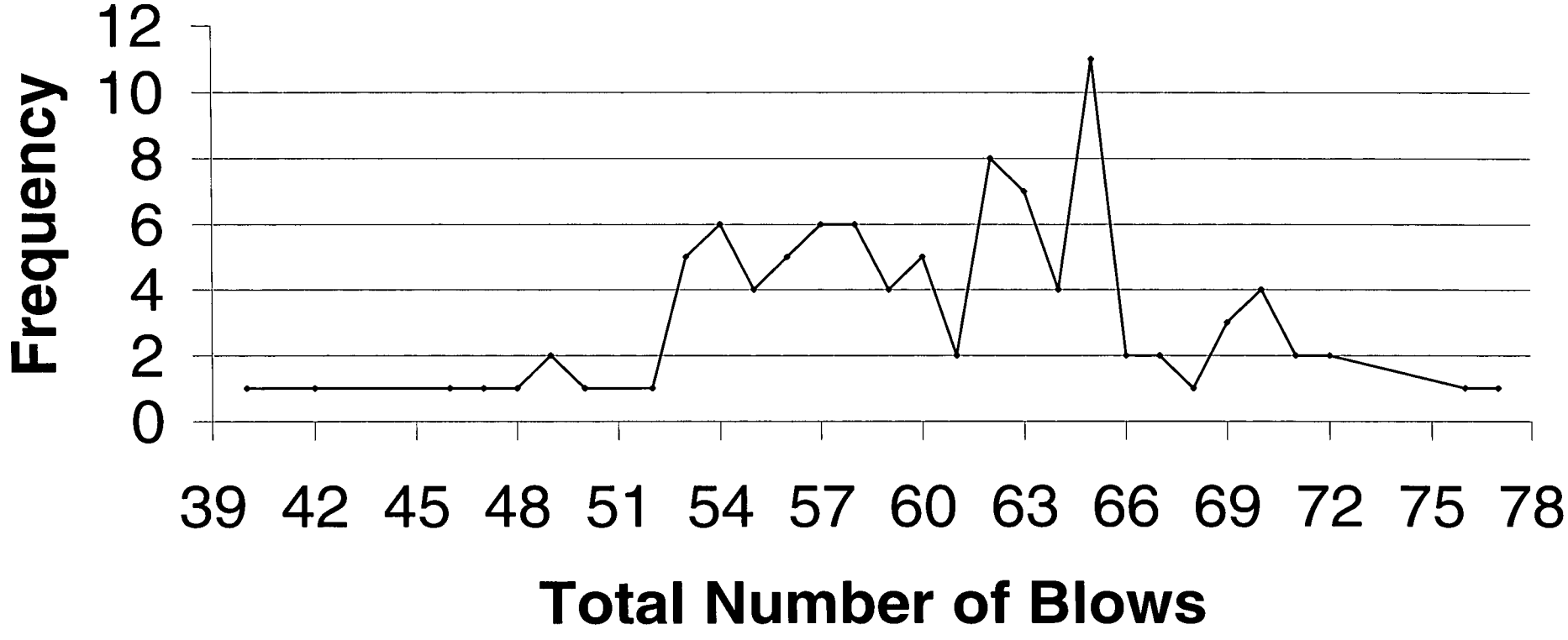


Figure 6.15. - Case Study [Figures 6.13 A & B]

500 Simulation Runs - Trial A

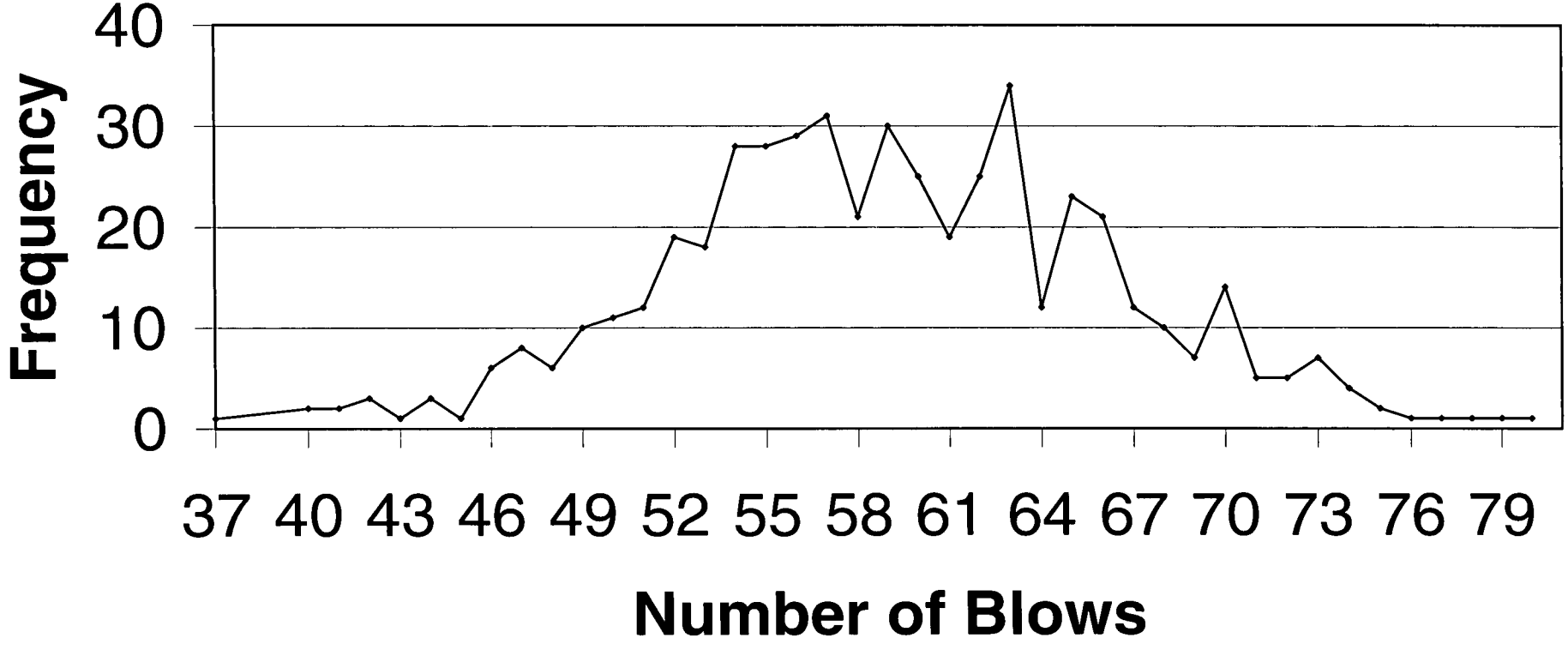


Figure 6.16. - Case Study [Figures 6.13 A & B]

100 Simulation Runs - Trial B

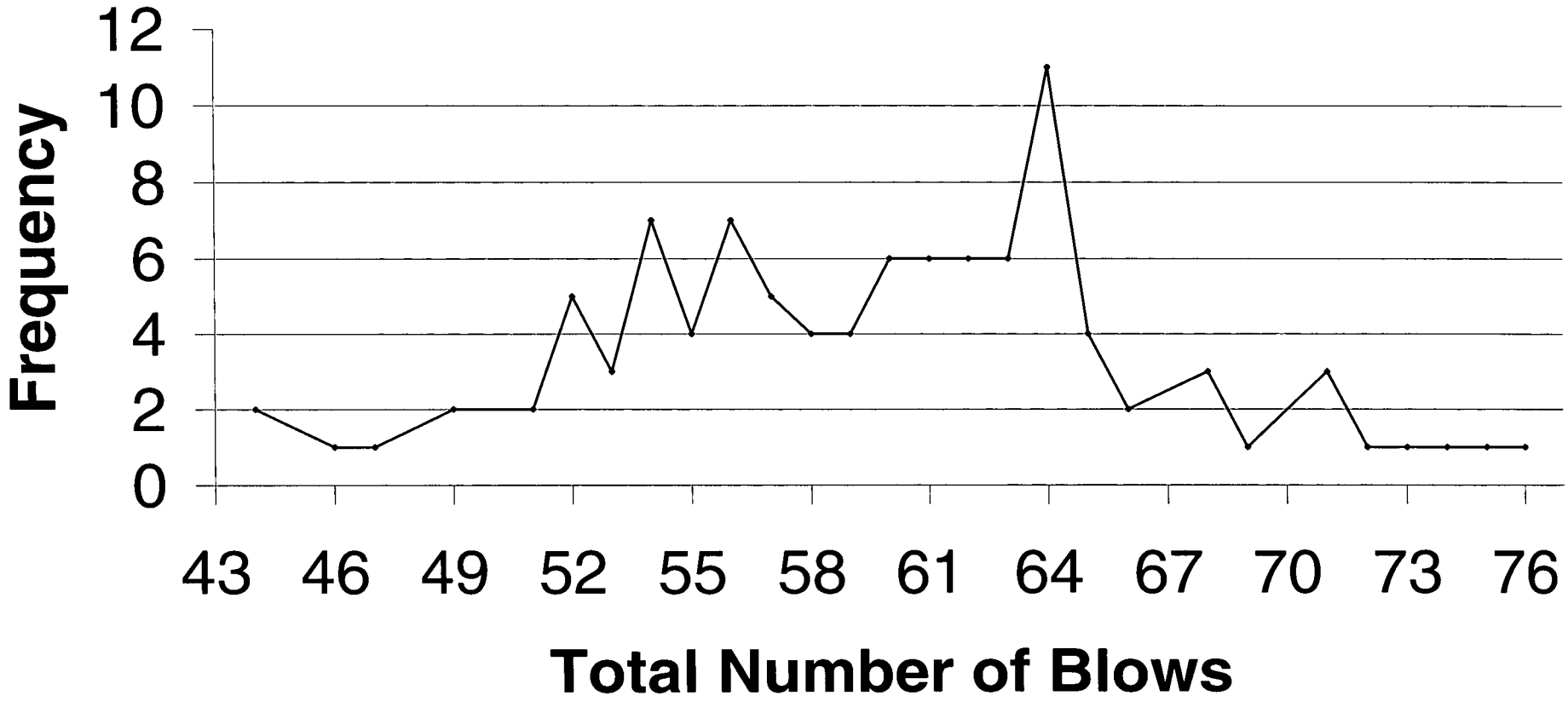


Figure 6.17. - Case Study [Figures 6.13 A & B]

500 Simulation Runs - Trial B

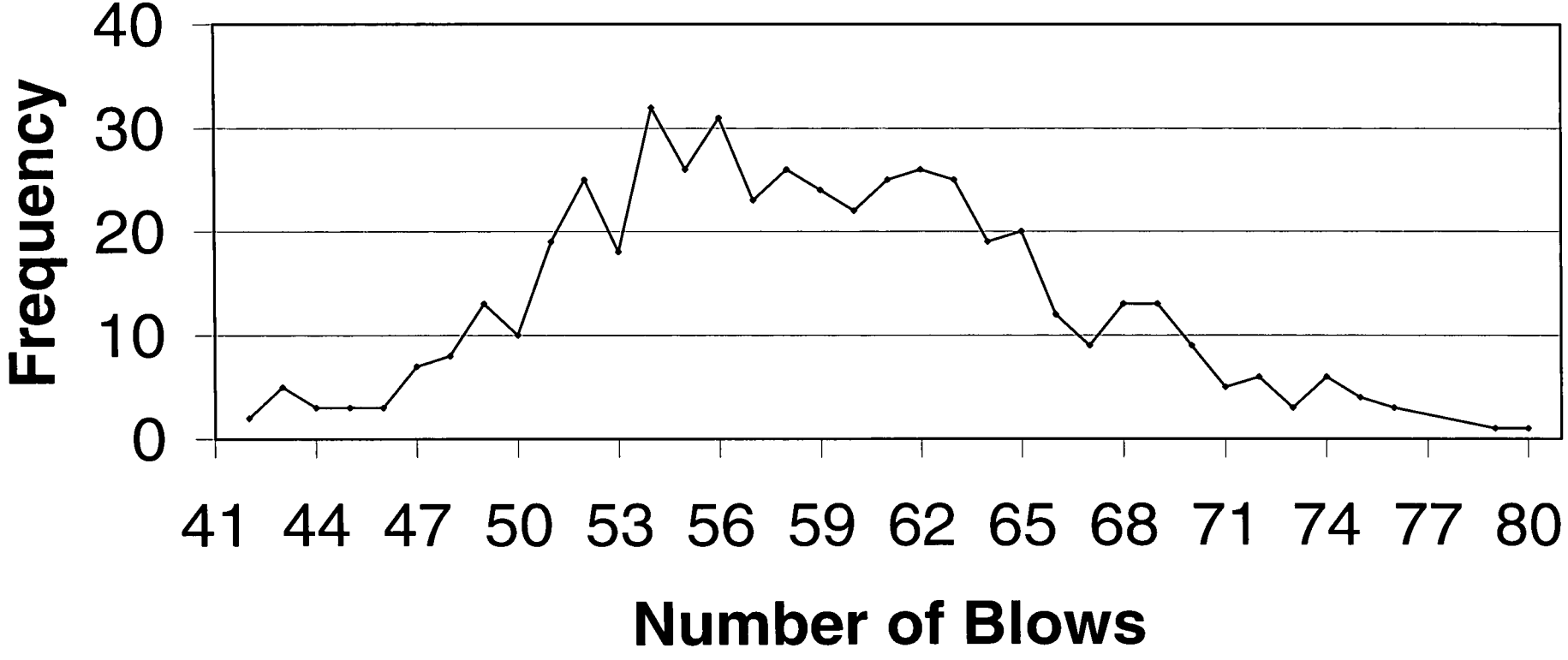


Figure 6.18. - Case Study [Figures 6.13 A & B]

100 Simulation Runs - Trial C

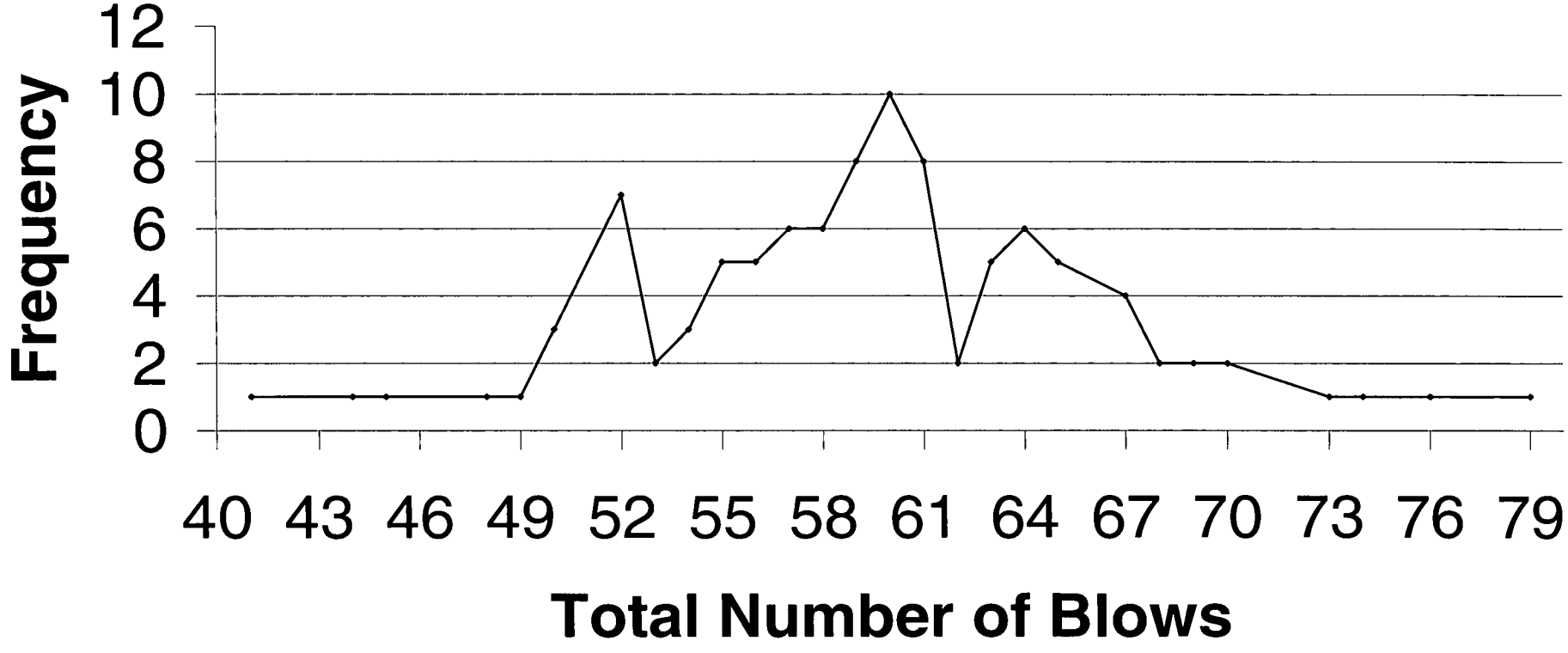


Figure 6.19. - Case Study [Figures 6.13 A & B]

500 Simulation Runs - Trial C

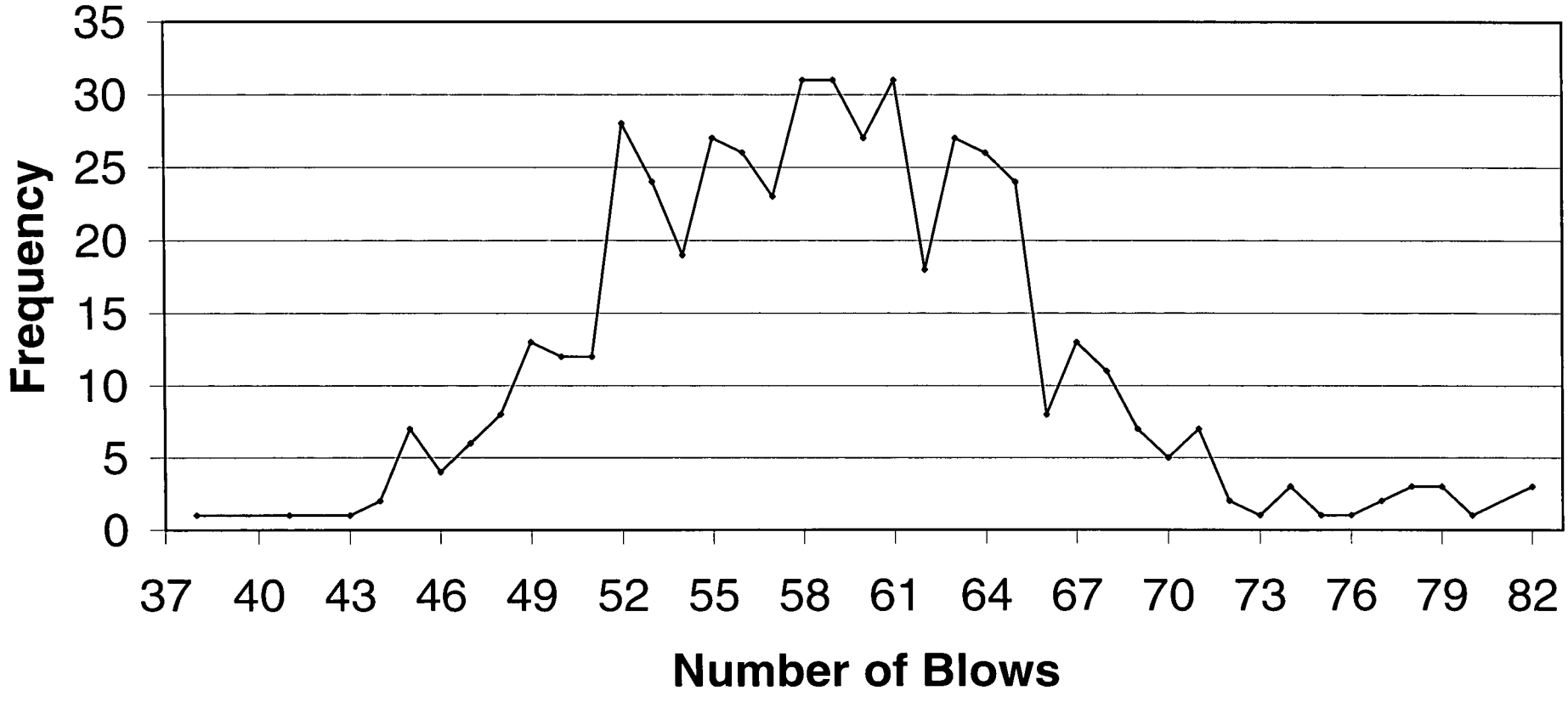


Figure 6.20. - Case Study [Figures 6.13 A & B]

100 Simulation Runs - Trial D

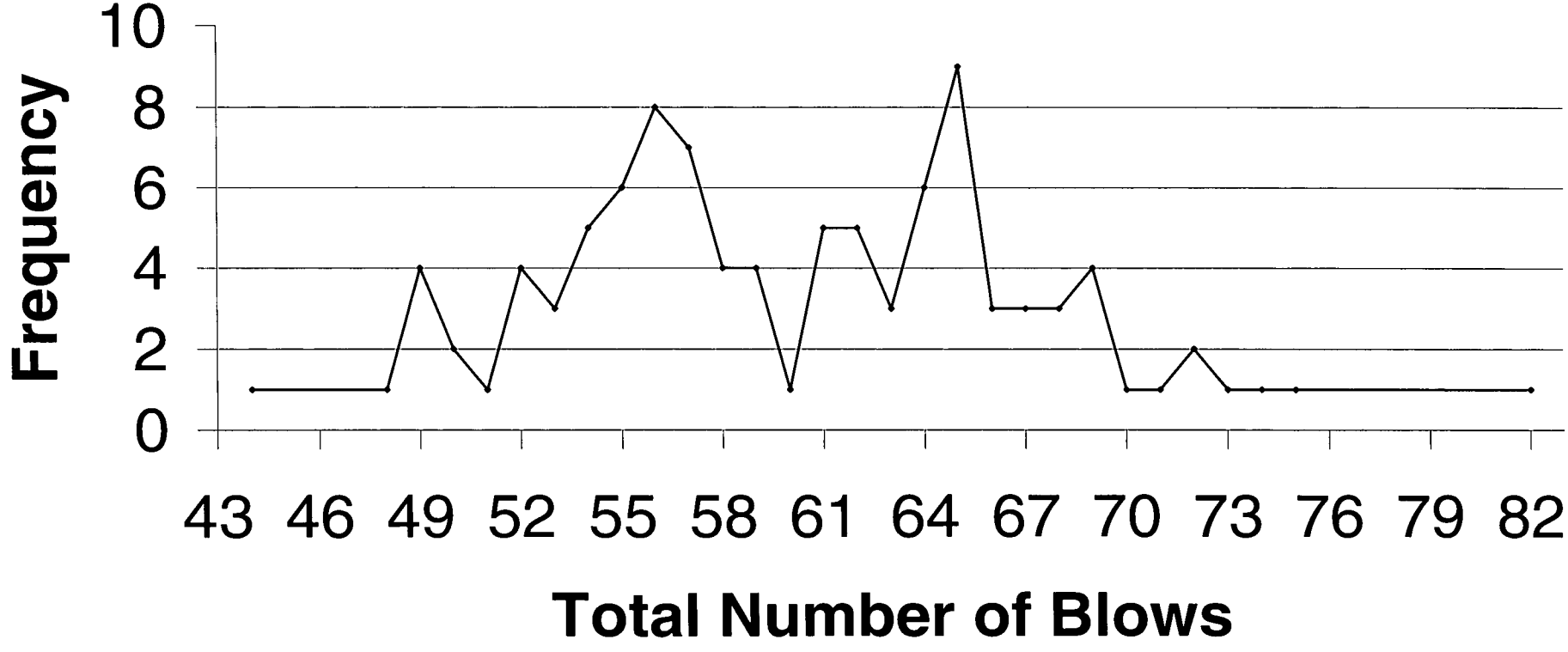


Figure 6.21. - Case Study [Figures 6.13 A & B]

500 Simulation Runs - Trial D

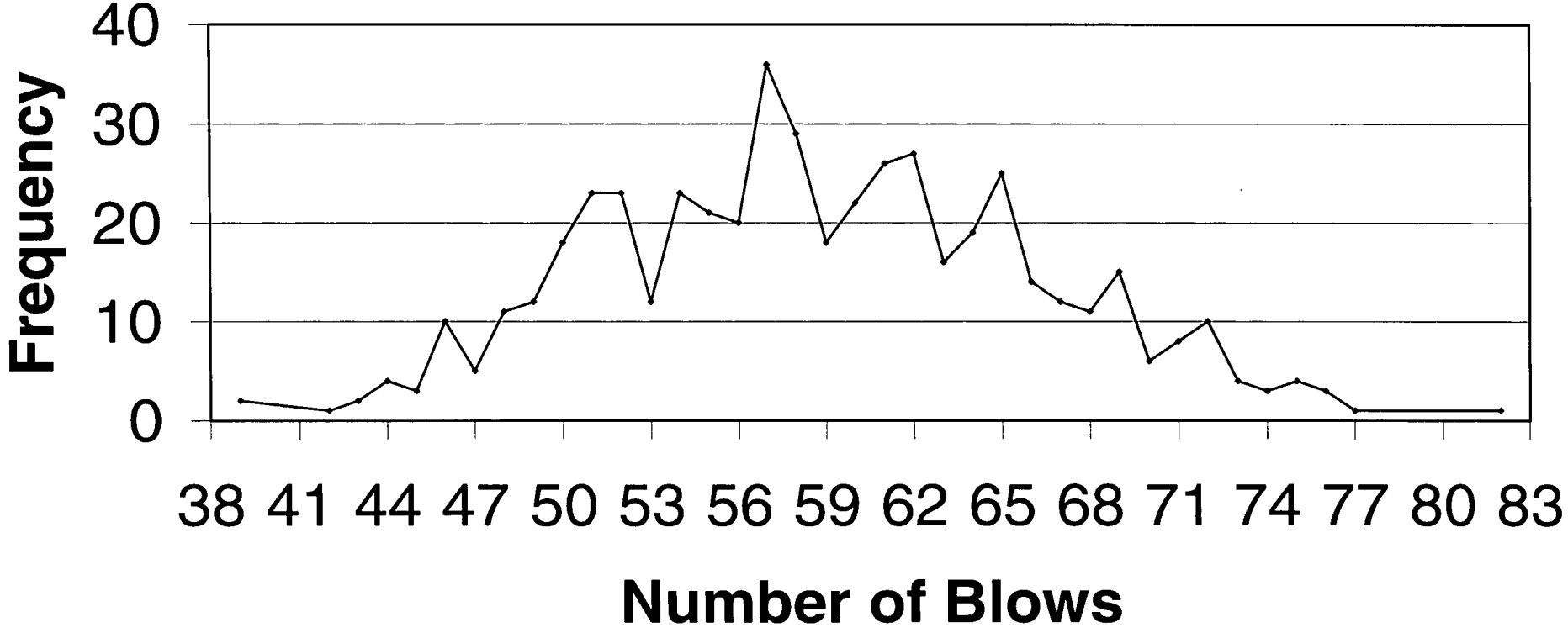


Figure 6.22. - Case Study [Figures 6.13 A & B]

100 Simulation Runs - Trial E

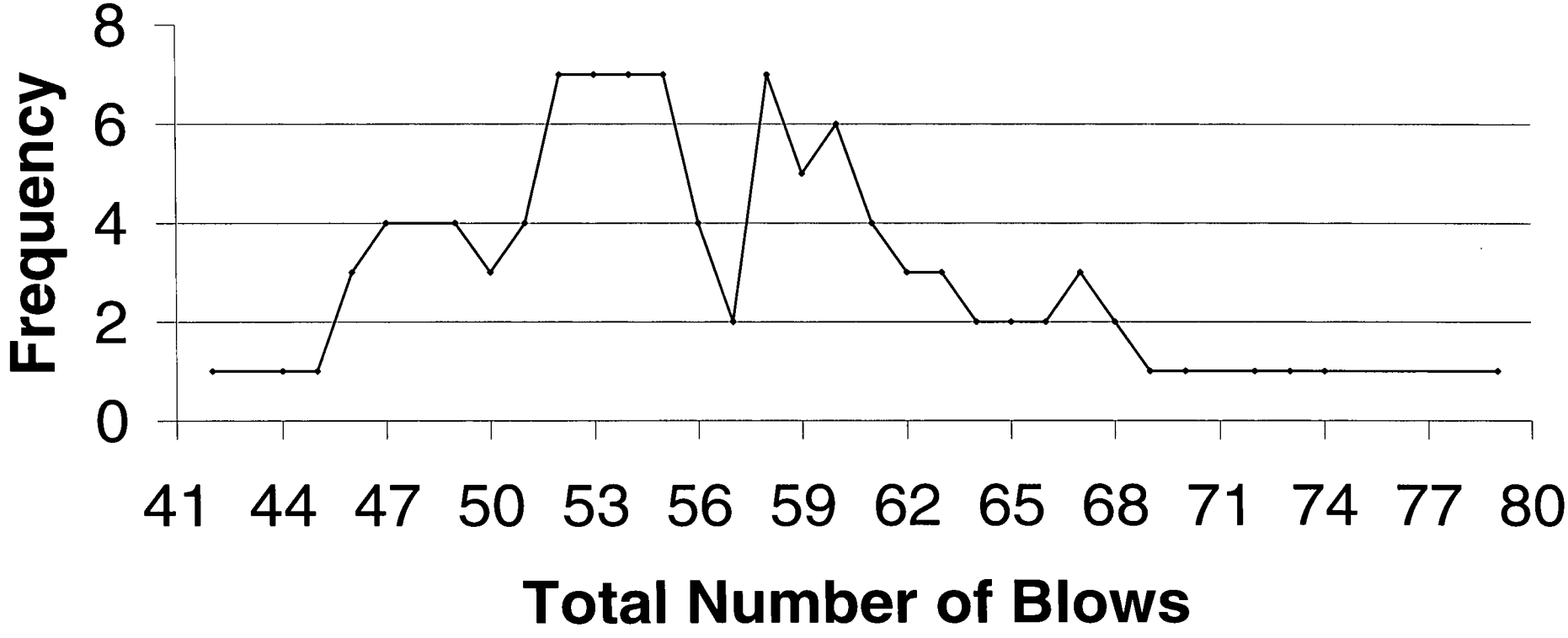
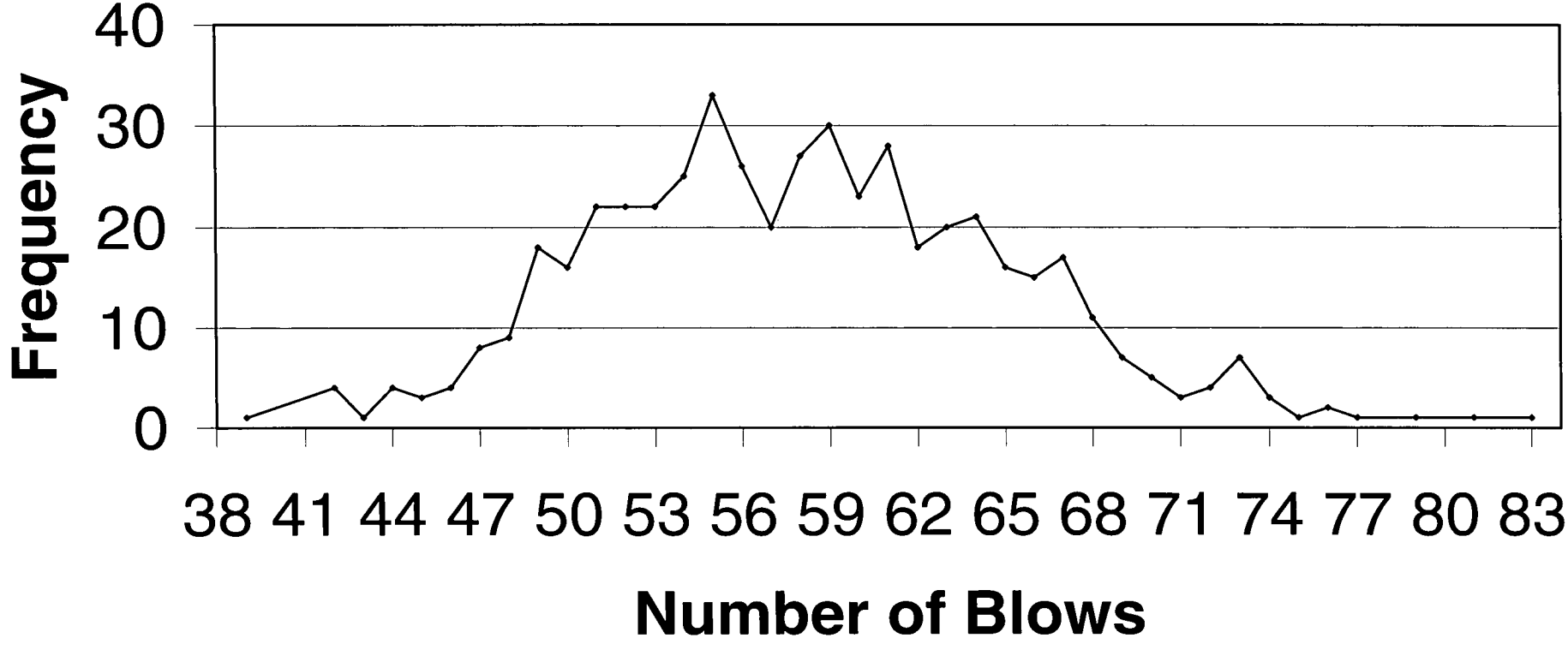


Figure 6.23. - Case Study [Figures 6.13 A & B]

500 Simulation Runs - Trial E



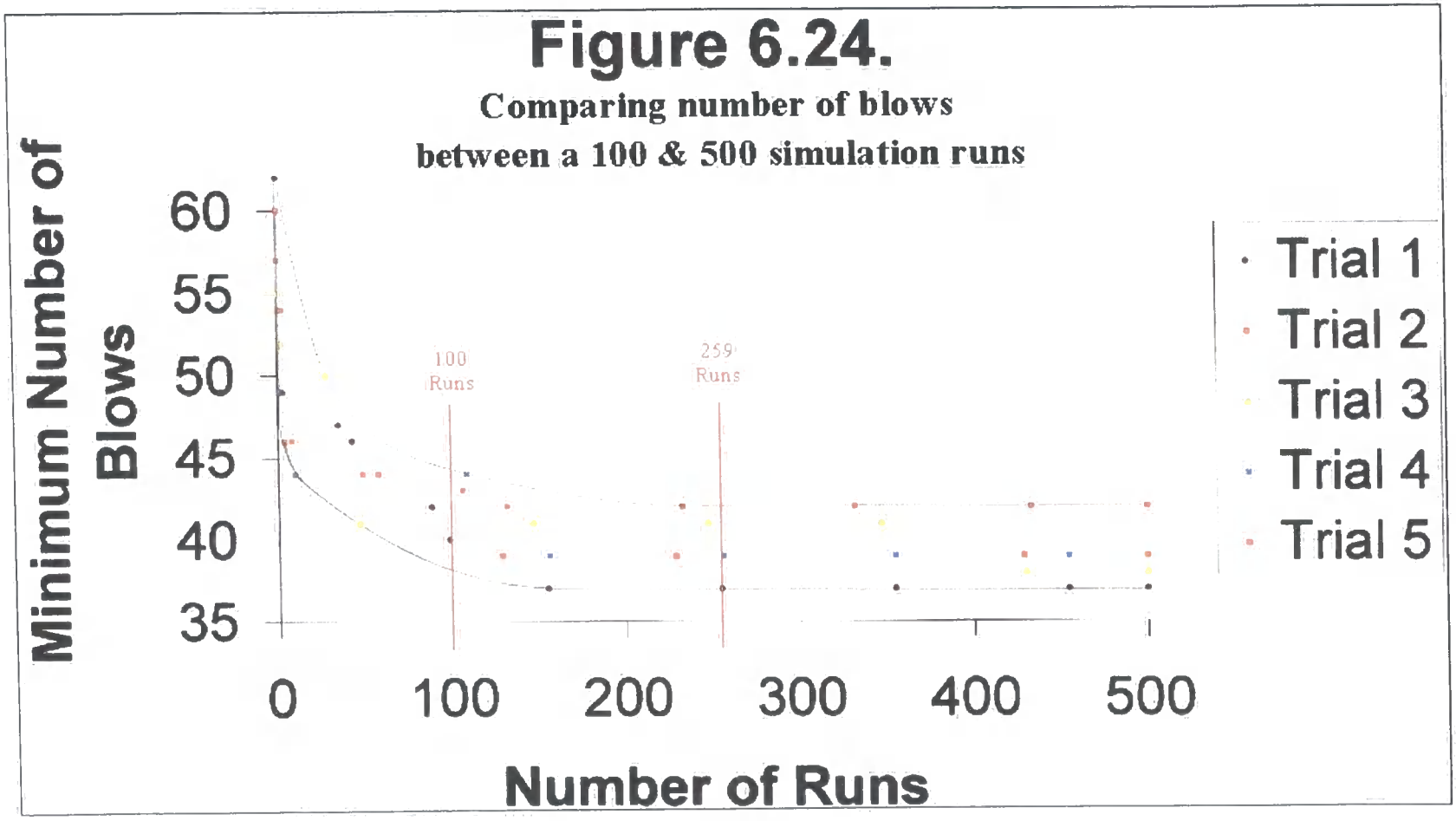


Figure 6.24.
Comparing number of blows
between a 100 & 500 simulation runs
Minimum Number of Blows
60
55
50
45
40
35
100 Runs
259 Runs
0
100
200
300
400
500
Number of Runs
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

Figure 6.26. - Case Study 1 - 100 Simulation Runs

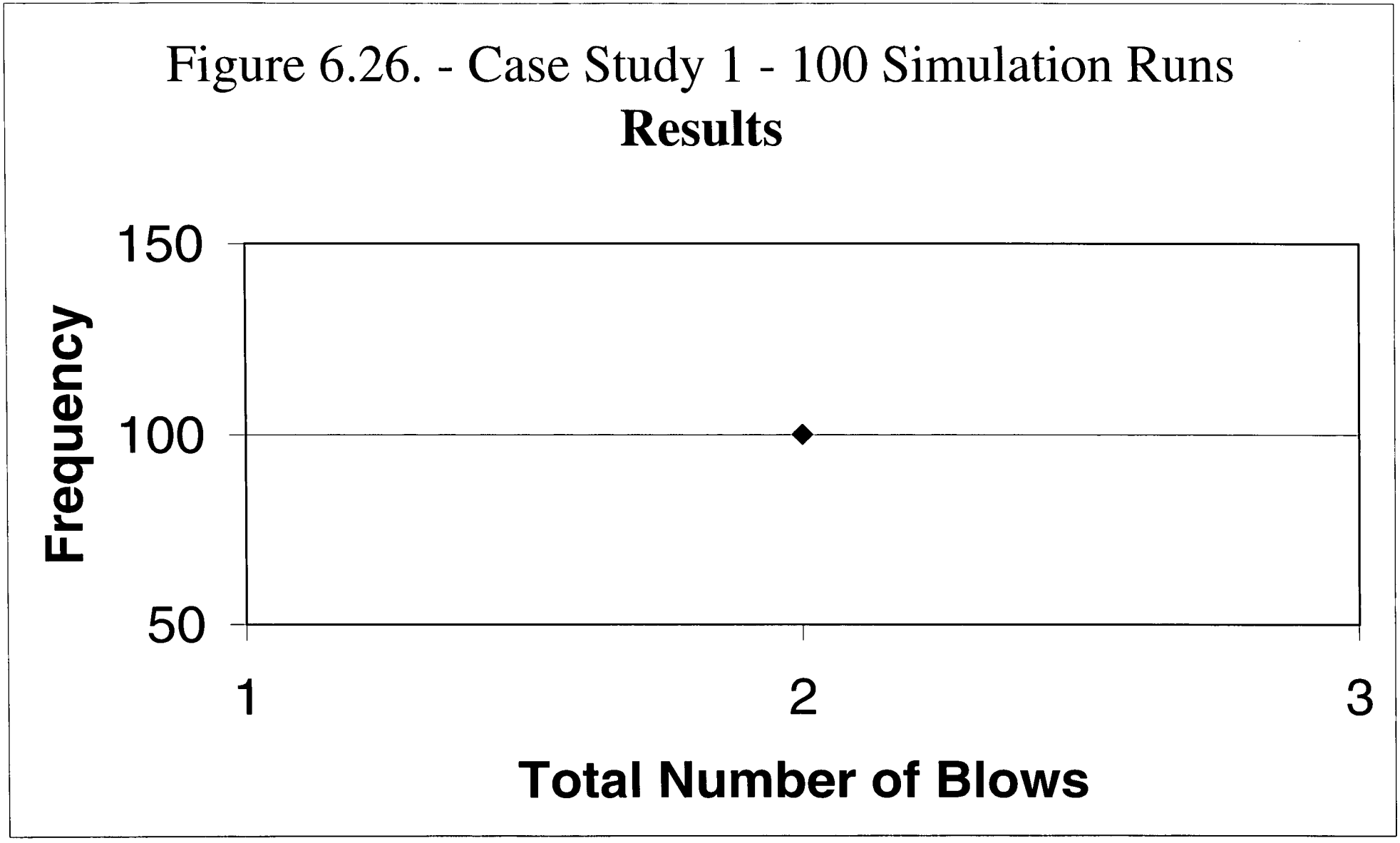


Figure 6.28. - Case Study 2 - 100 Simulation Runs

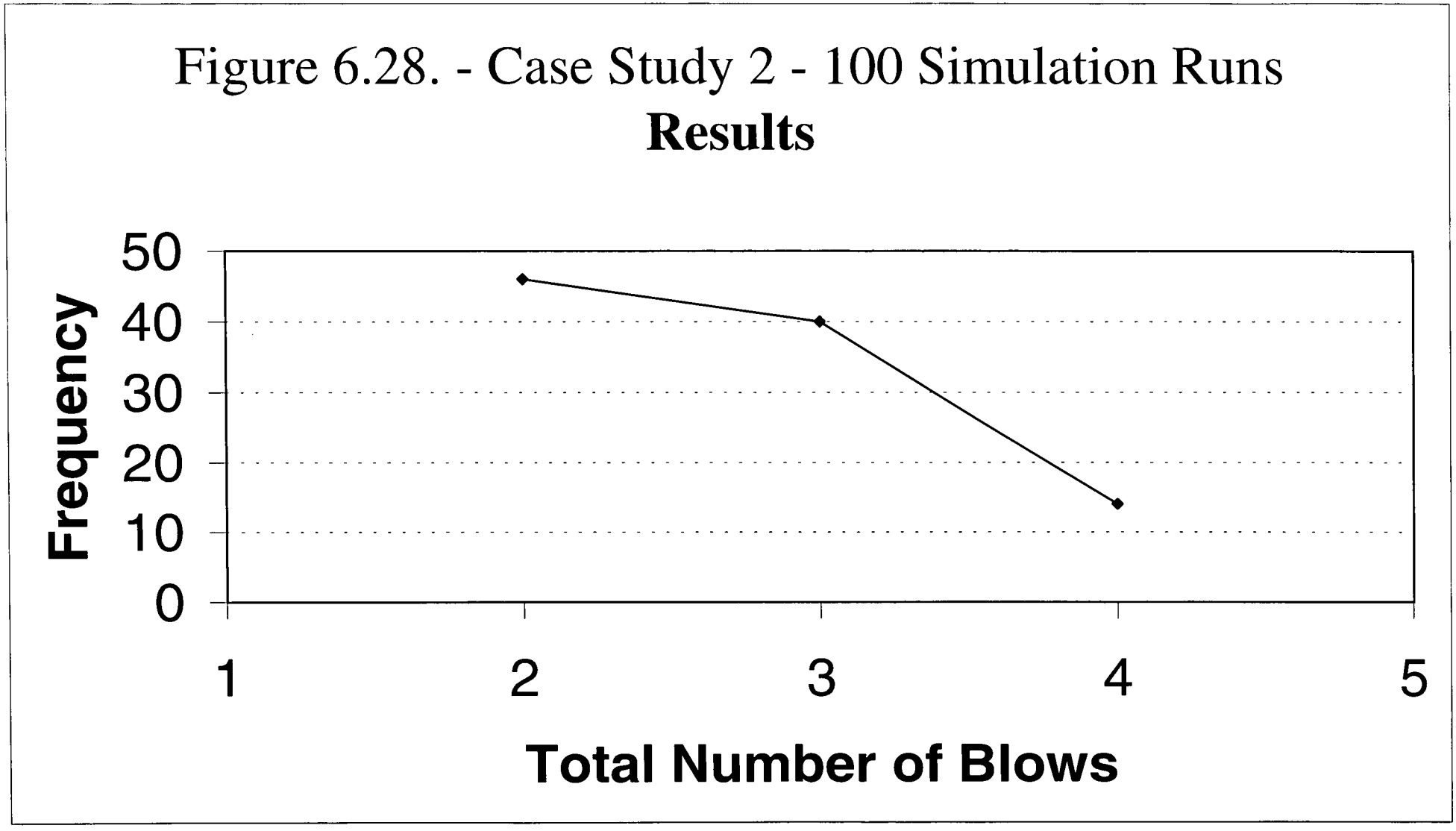


Figure 6.30. - Case Study 3 - 100 Simulation Runs

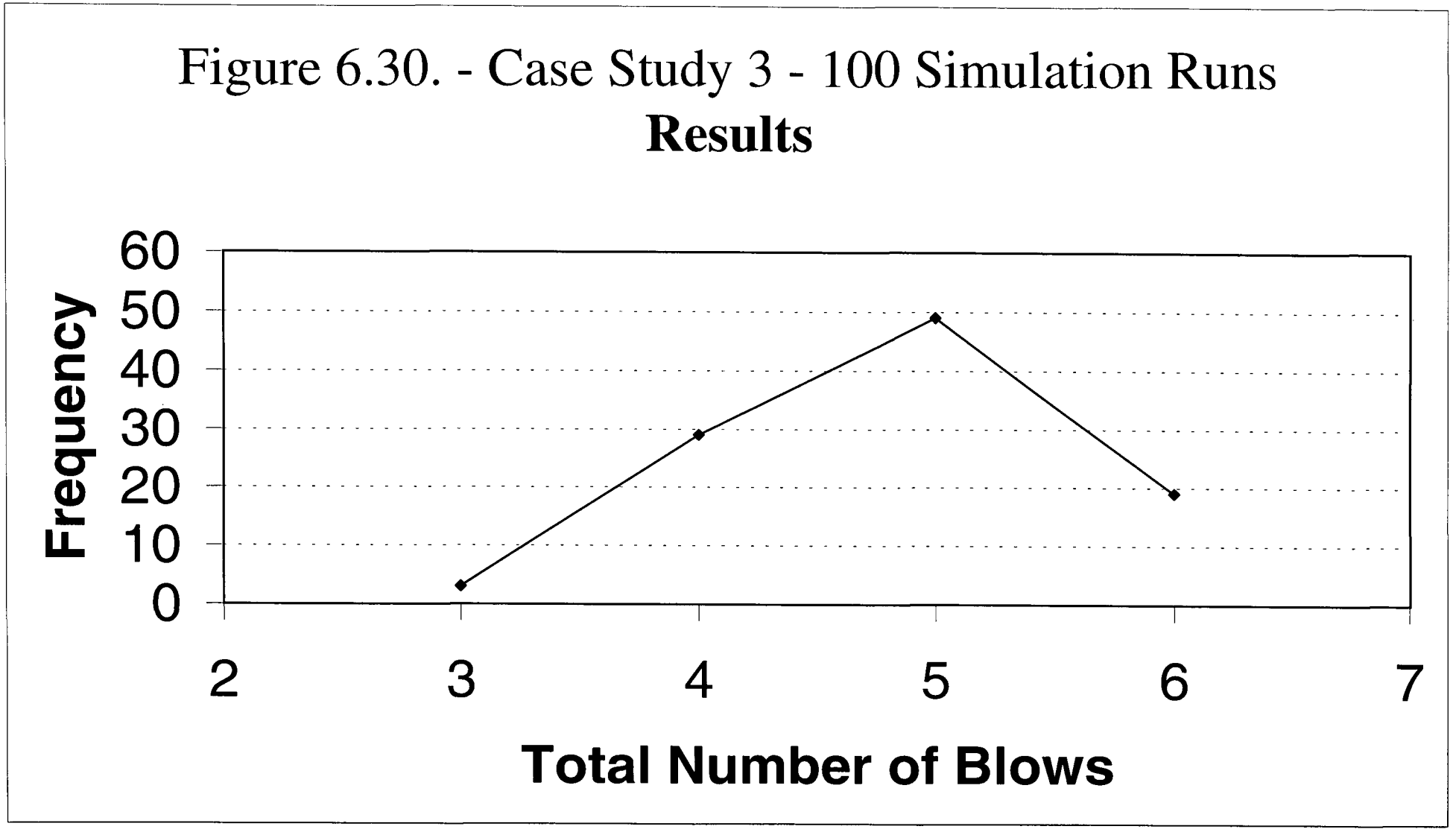


Figure 6.32. - Case Study 4 - 100 Simulation Runs

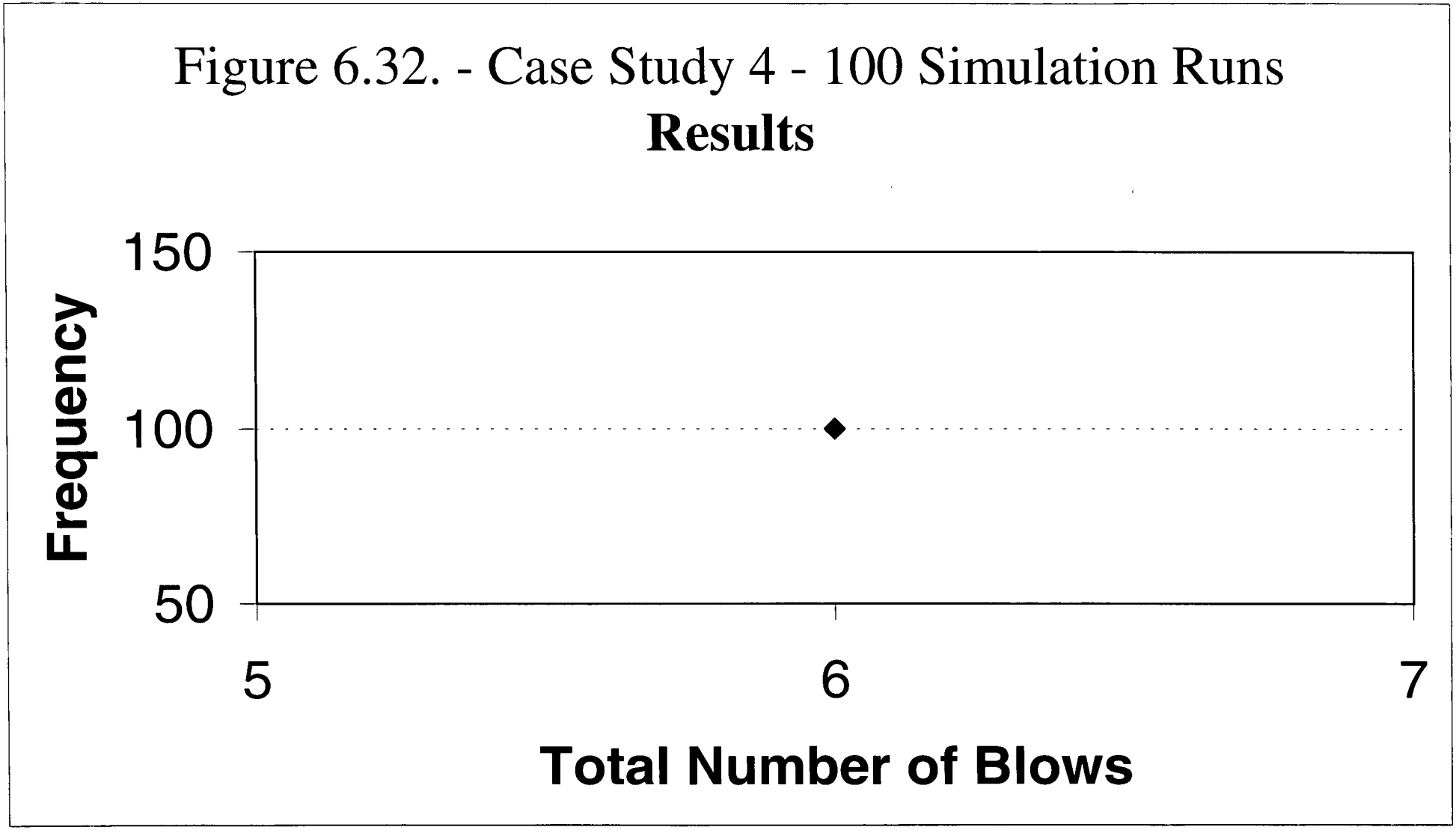


Figure 6.34. - Case Study 5 - 100 Simulation Runs

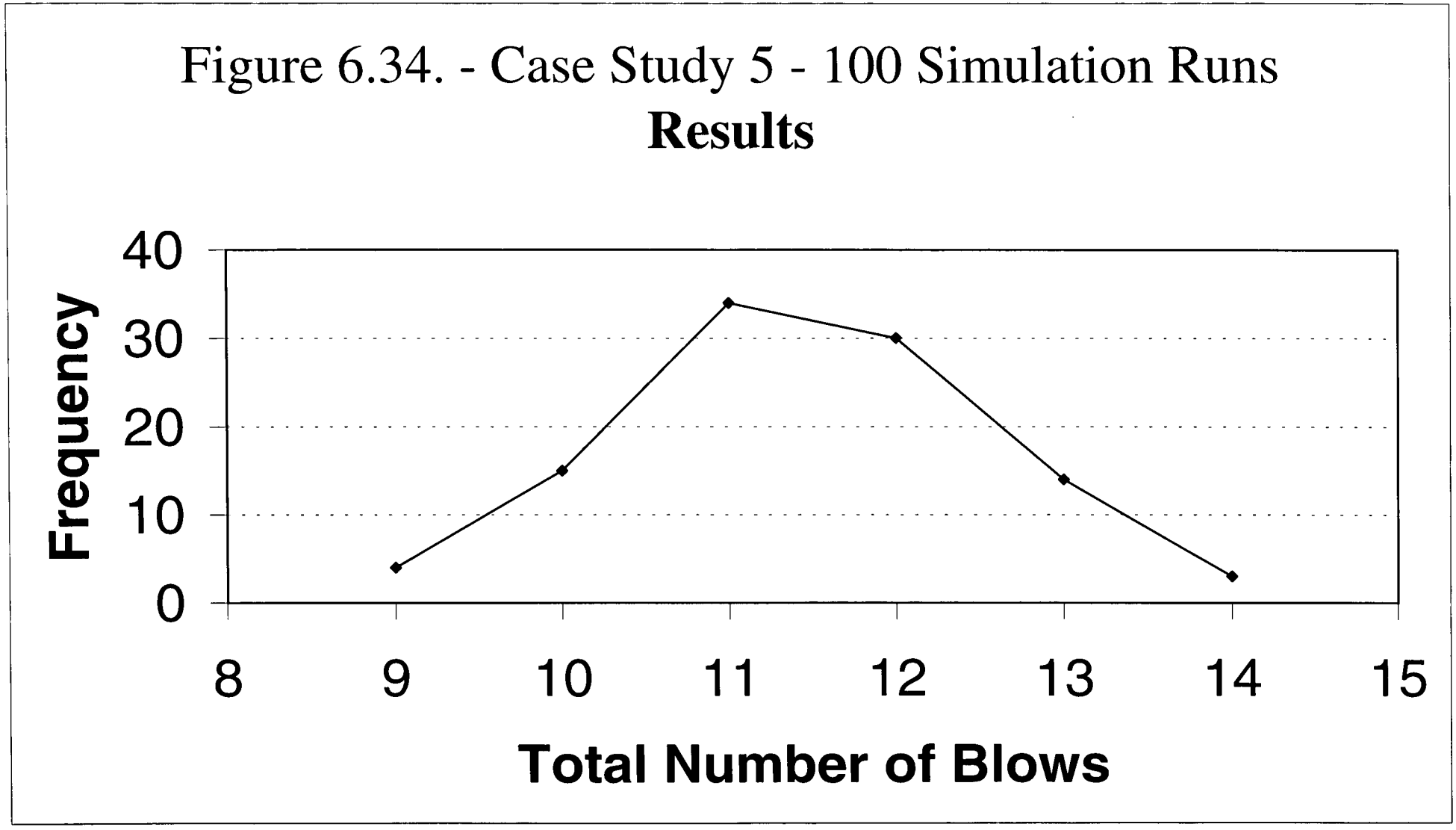


Figure 6.36. - Case Study 6 - 100 Simulation Runs

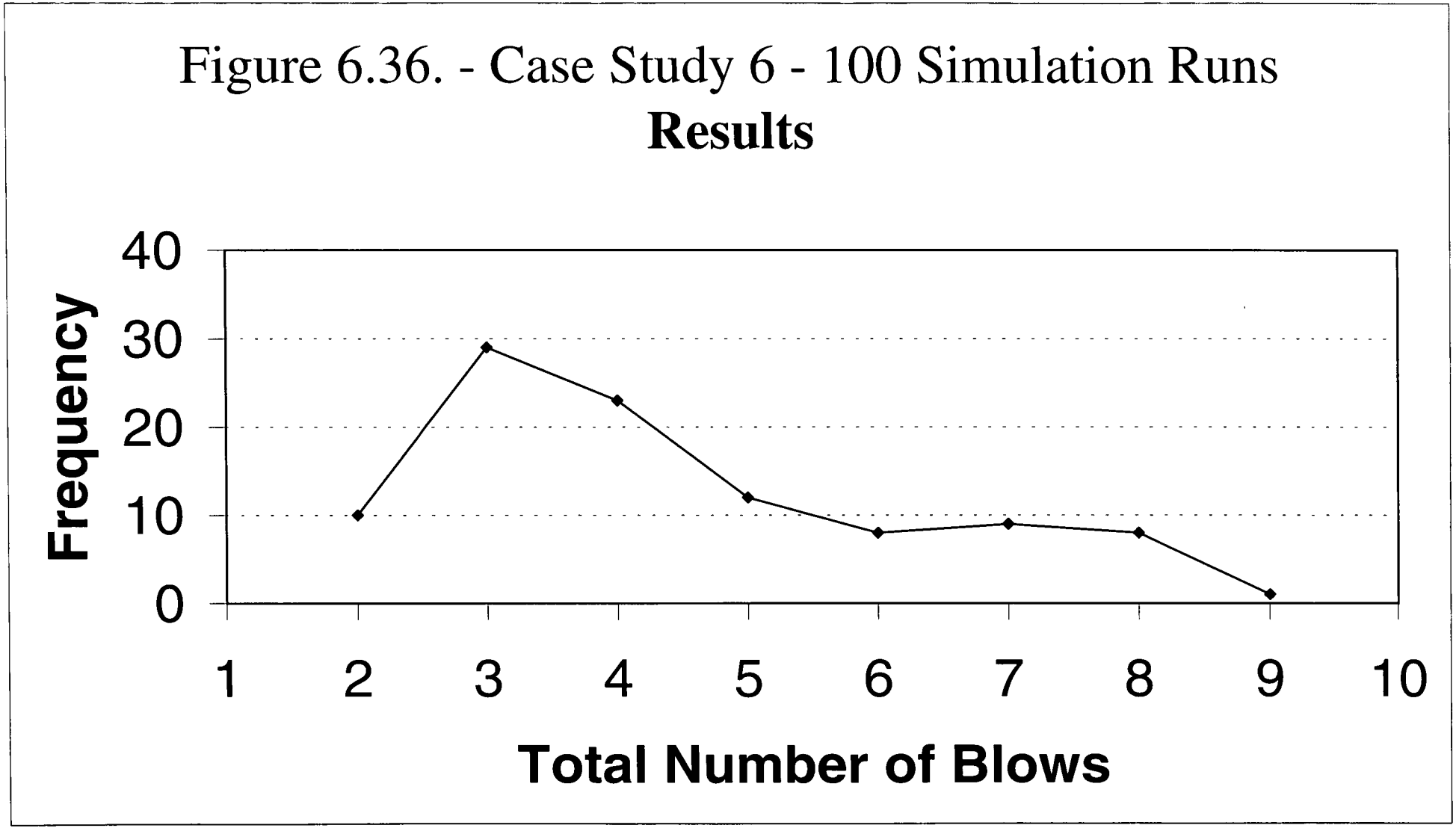


Figure 6.38. - Case Study 7 - 100 Simulation Runs

Results

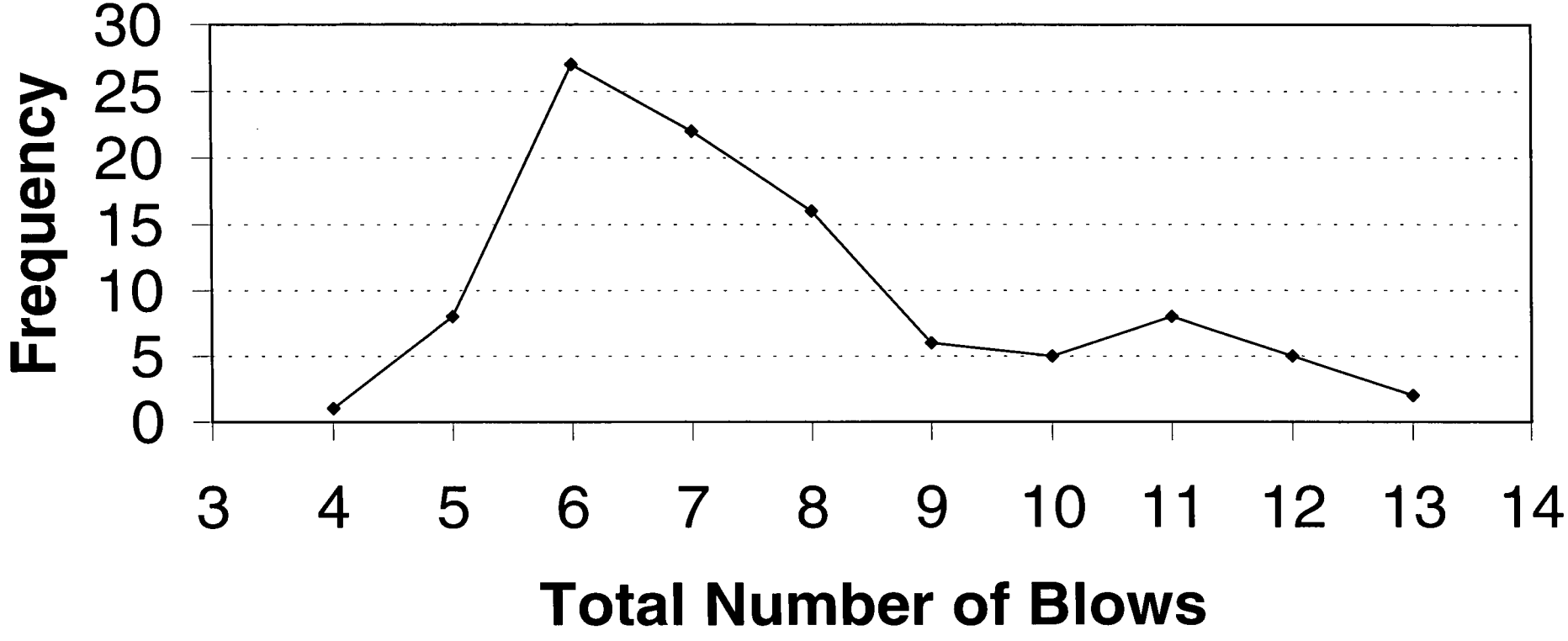


Figure 6.40. - Case Study 8 - 100 Simulation Runs

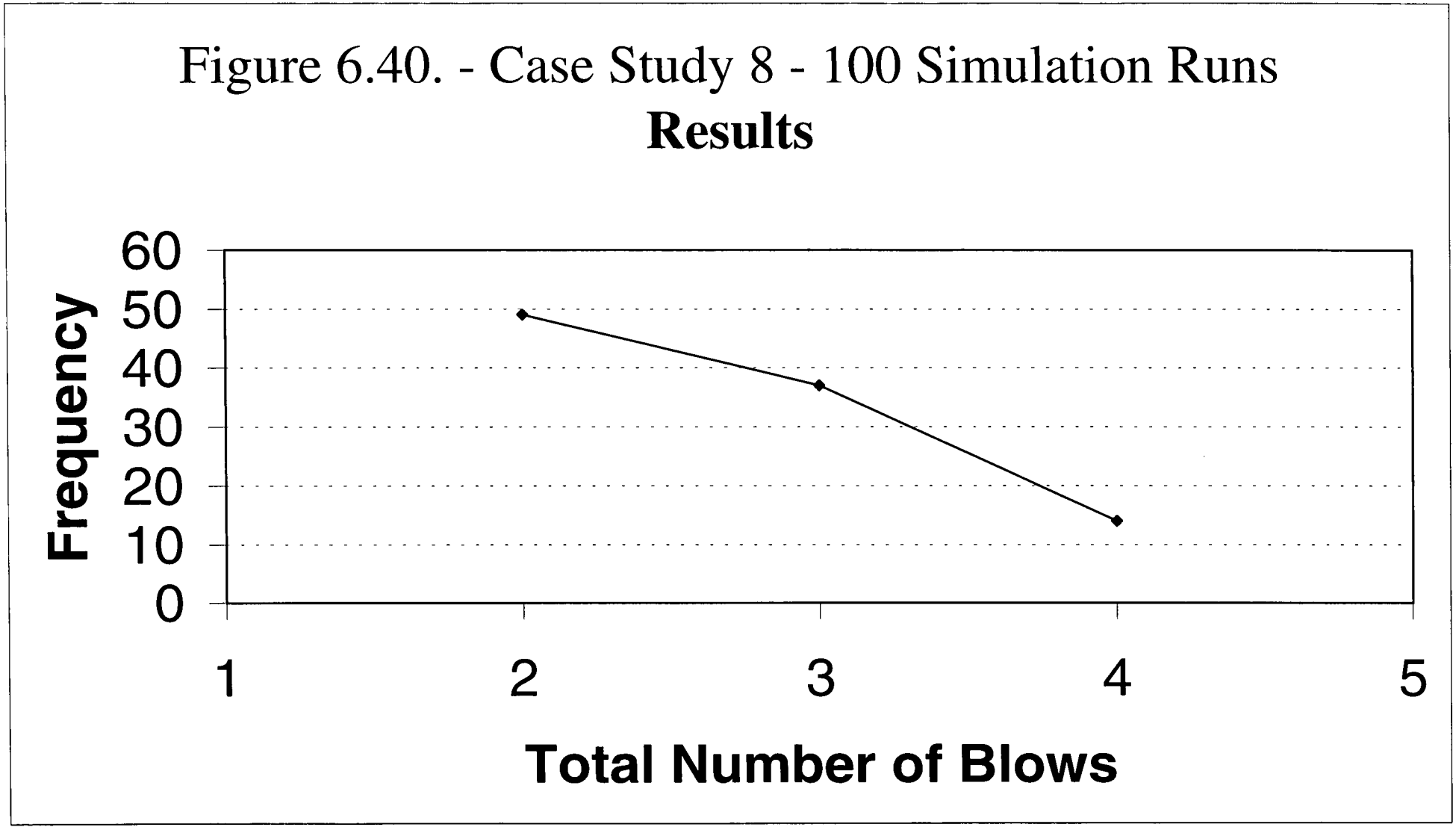
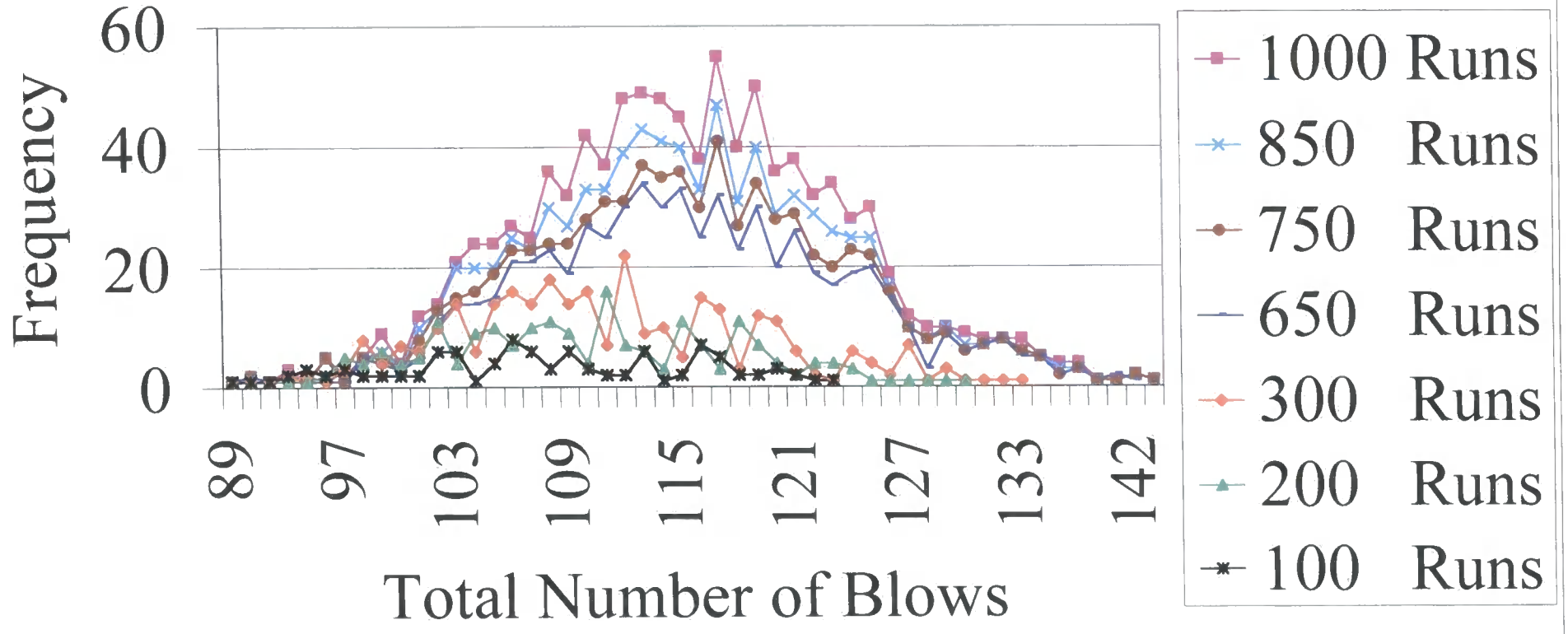


Figure 6.47. - Case Study 9
[Simulation Results]



CHAPTER 7

INDUSTRIAL CASE STUDIES

At the beginning of this chapter the author would like to express his gratitude to Ms. Judith Porter the Managing Director of Hydram Engineering – Sheet Metalwork www.hydram.co.uk and Mr. Andrew Jordan of Hydram for their collaboration during the preparation of this chapter. They always gave an open and warm welcome to come and discuss the work.

The structure of this chapter consists of four sections each of which discusses a different industrial case study. In each of these four sections a descriptive drawing of the sheet metal product under study was followed by three different randomly generated Monte Carlo solutions, as shown. The three generated solutions were then exposed to the second phase of the Monte Carlo search namely, "Gene Mixing and Re-Sequencing". The tools that have been used in the first three industrial case studies are those used by Hydram Engineering and are listed in Appendix 'B".

The author believes that it is necessary here to explain what is a "Gene" and what is "Gene Mixing and Re-Sequencing". As has been explained over previous chapters, in this work a random tool edge is mapped to a randomly selected feature edge. In other words a "Gene" is that unique position of the selected tool to punch a selected feature edge; see Figure 7.1. It can be seen in Figure 7.1. that the same tool can form a number of different genes. As regards "Gene Mixing and Re-Sequencing", a gene re-sequence is the change of the sequence/order of the blows, see Figure 7.1., where one or more genes can be made redundant, and hence improve the solution in hand. Gene re-sequence is

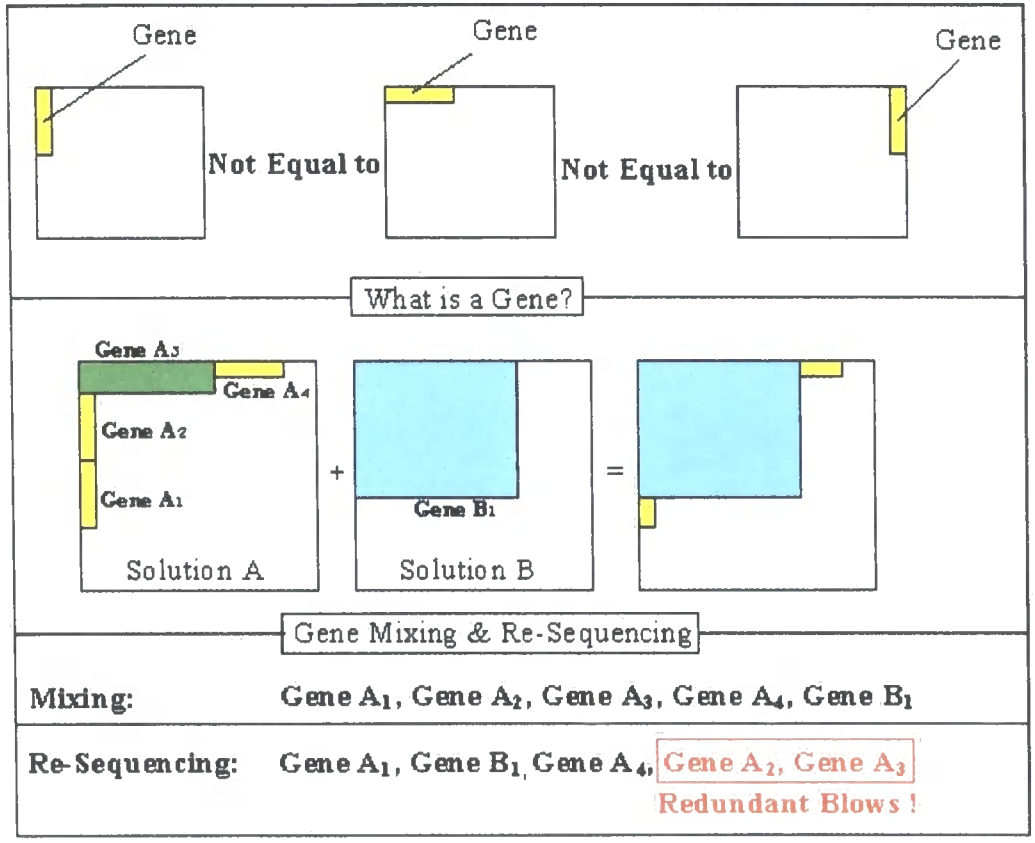


Figure 7.1. Gene Mixing and Re-Sequencing

done at two levels. Firstly at the one solution level and secondly when mixing more that one solution, see Figure 7.1.

The author hopes that by the end of this chapter the reader should clearly see how efficient and reliable the Monte Carlo – a non-rule based – search is.

The remainder of this chapter deals with the afore mentioned industrial case studies.

7.1. INDUSTRIAL CASE STUDY 1

The product undertaken in this case study is described in Figures 7.2. (a) to (c). This product consists of 24 gene areas. 100 initial Monte Carlo runs have been conducted from which the best three solutions were picked. The three solutions had cost function values of 67, 67 and 71. These three different Monte Carlo generated solutions are shown in Figures 7. 3. to 7. 5.

The next step was to expose these three generated solutions to the "Gene Mixing and Re-Sequencing" process. A visual description of such a process to a number of gene areas is shown in Figure 7.6.

A comparison between the three initial Monte Carlo generated solutions against the "Gene" manipulated Monte Carlo solution is presented in Table 7.1. The table clearly shows, see gene areas number 16, 22 and 24, how the gene manipulation process has lead to a much-improved solution of a cost function value of only 55, see Figure 7.7. This solution could possibly be improved by mixing and re-sequencing the genes of more trials i.e., more than three trials.

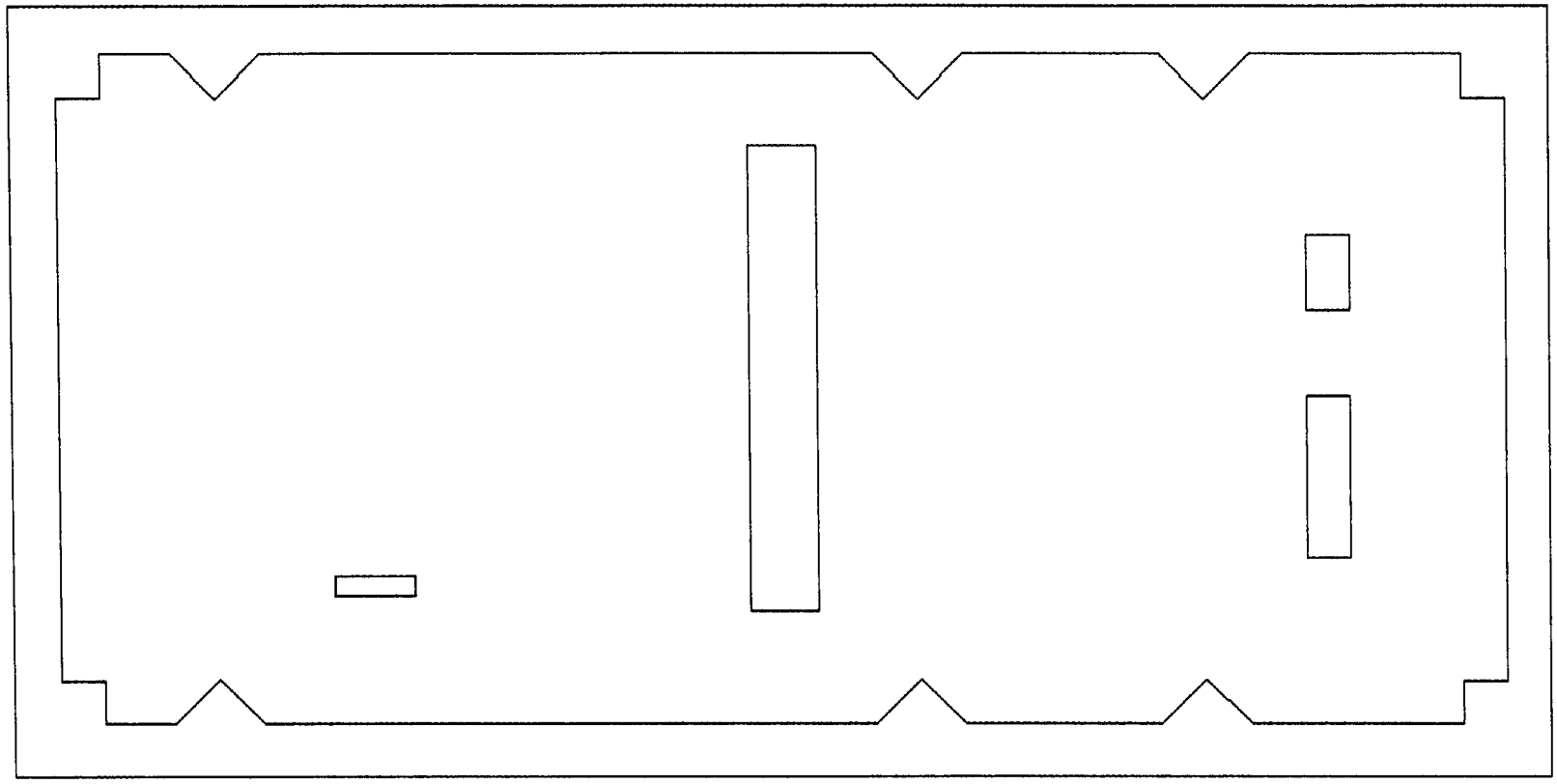


Figure 7.2. (a). Industrial Case Study # 1 – Part Description.

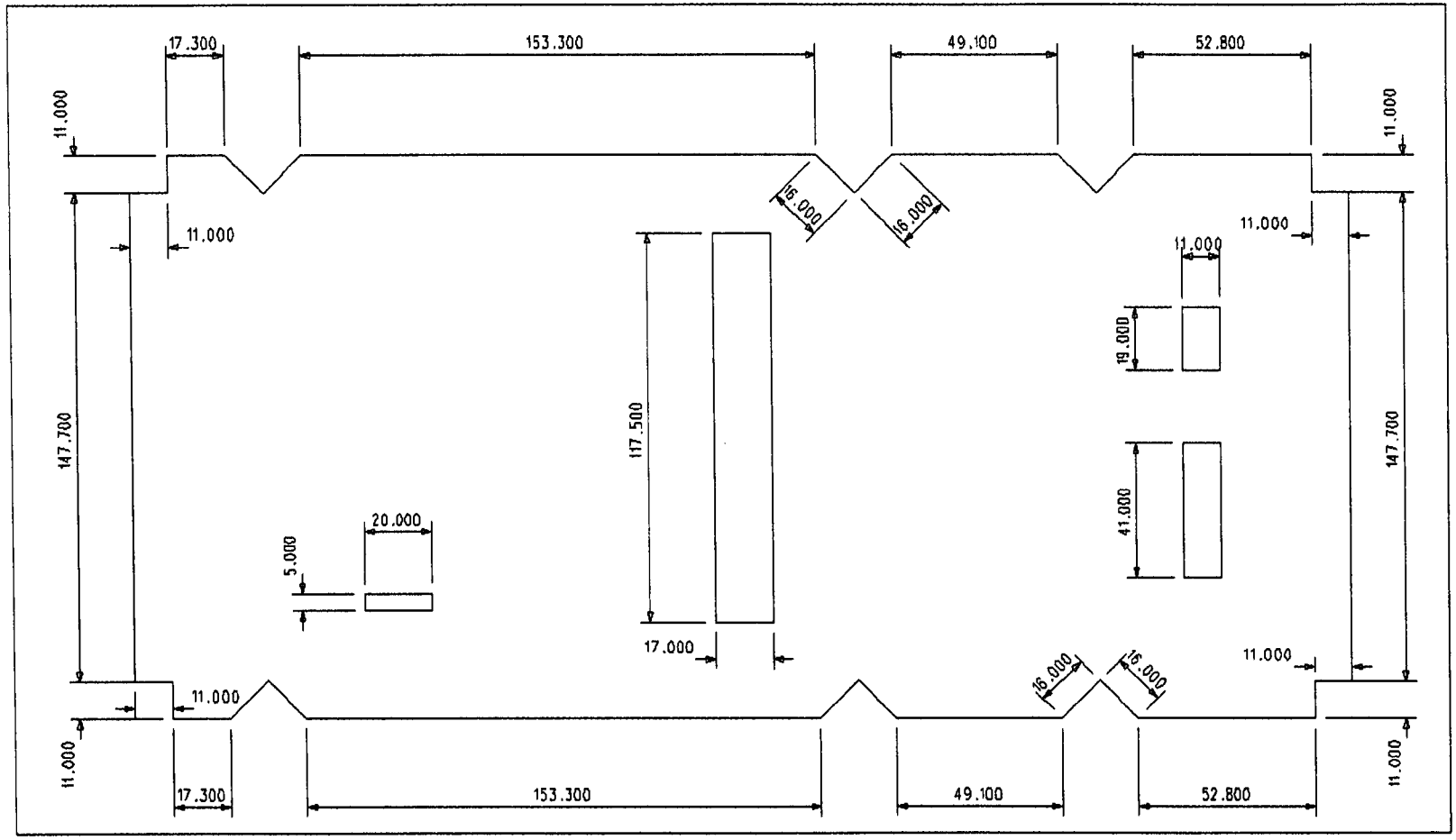


Figure 7.2. (b). Industrial Case Study # 1 – Part Description.

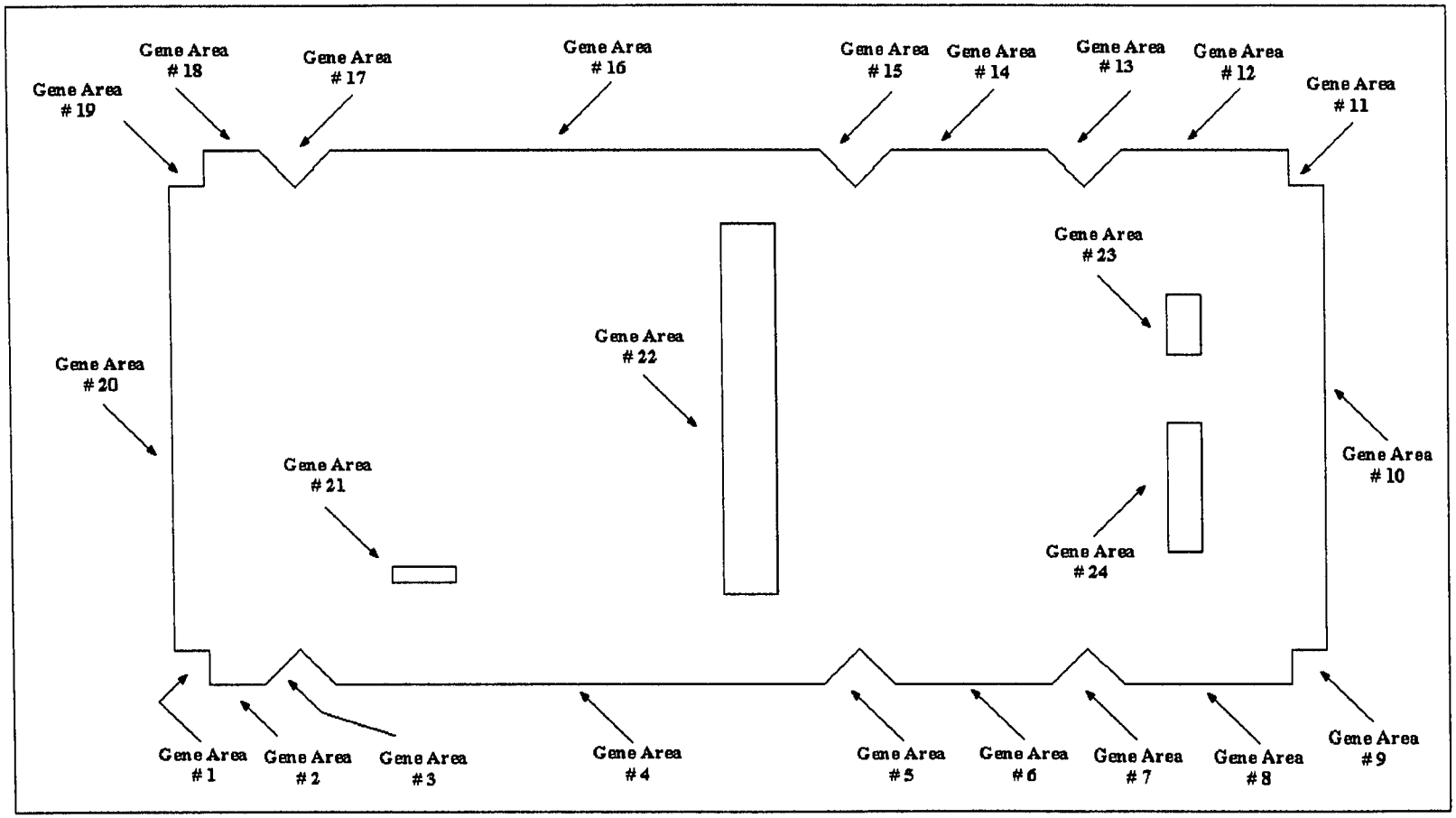


Figure 7.2. (c). Industrial Case Study # 1 – Part Description.

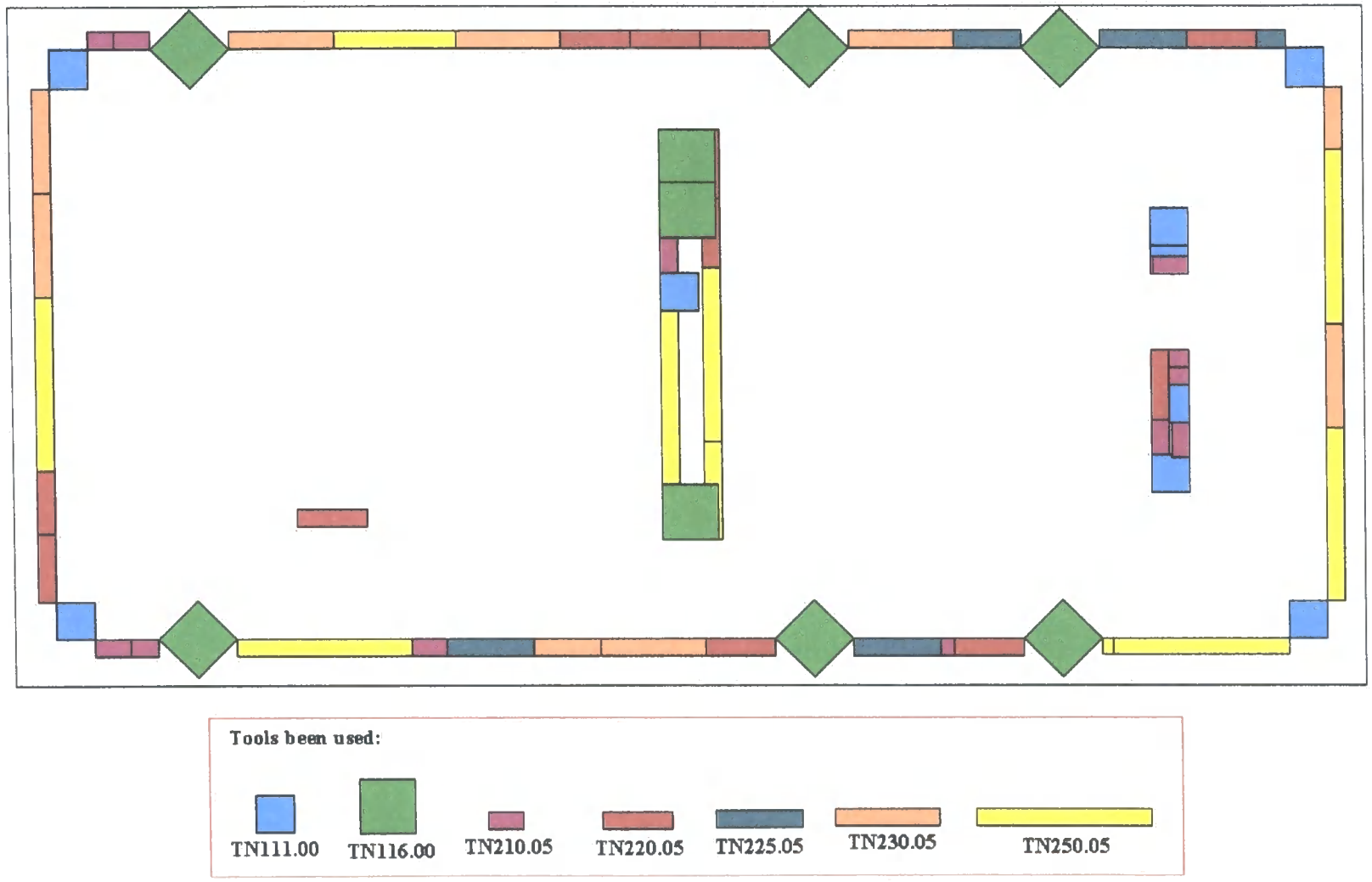


Figure 7.3. Industrial Case Study # 1 – Trial A.

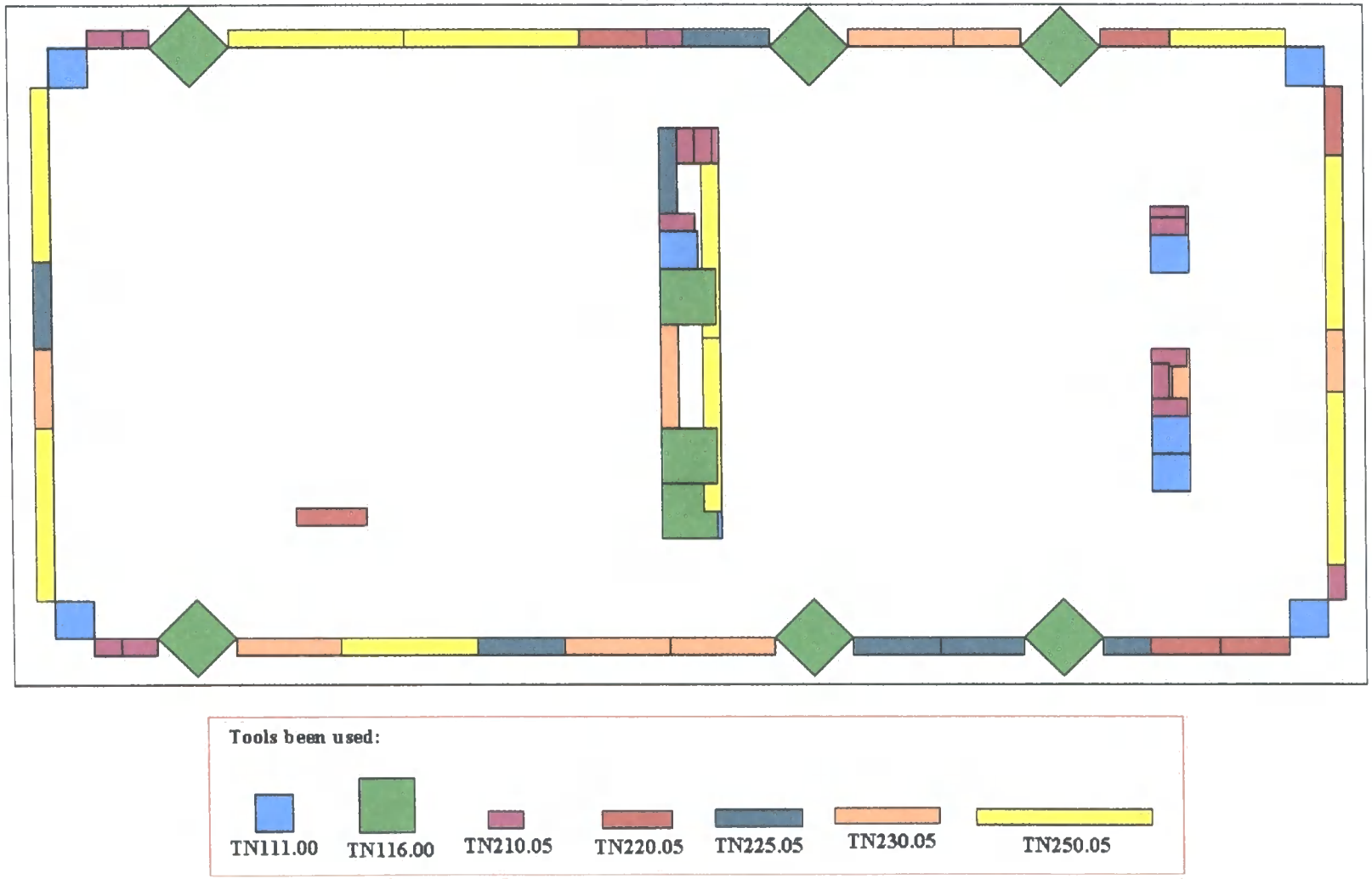


Figure 7.4. Industrial Case Study # 1 – Trial B.

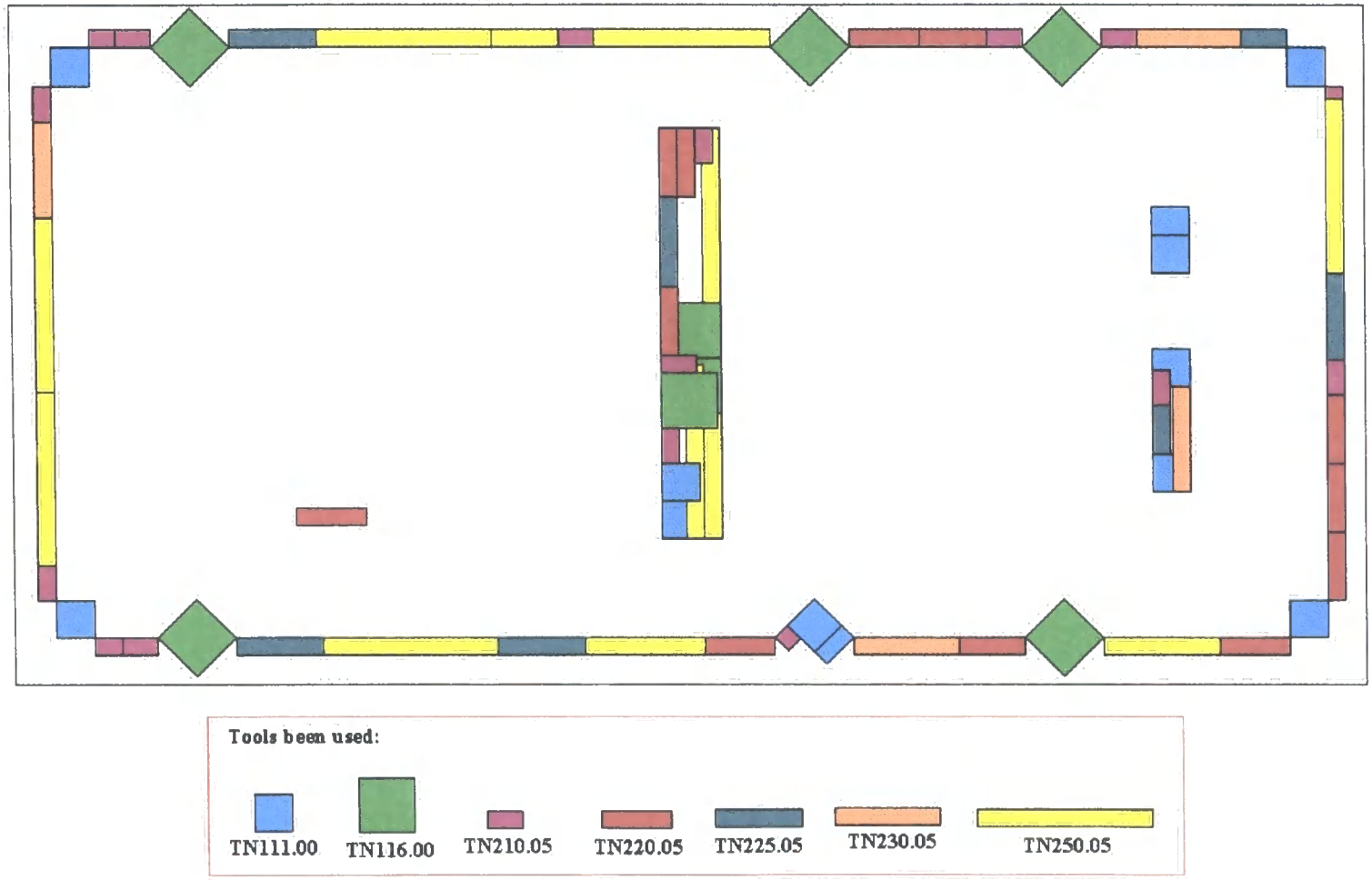


Figure 7.5. Industrial Case Study # 1 – Trial C.

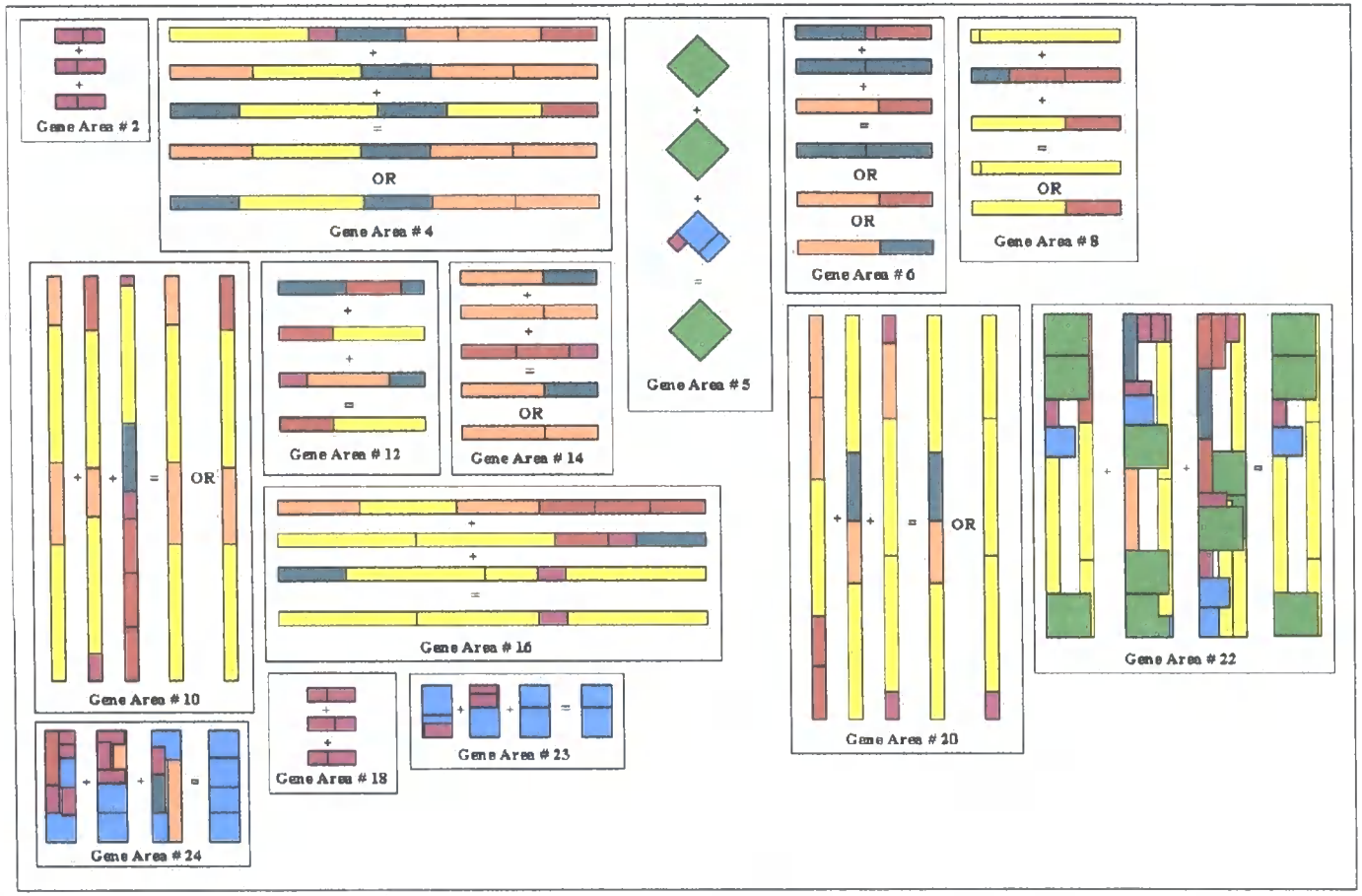


Figure 7.6. Industrial Case Study # 1 – Gene Manipulation.

	67 Blows ! Trial A	67 Blows ! Trial B	71 Blows ! Trial C	55 Blows ! Gene Manipulated Trial
Gene Area Number	Number of Blows	Number of Blows	Number of Blows	Number of Blows
1	1	1	1	1
2	2	2	2	2
3	1	1	1	1
4	6	5	5	5
5	1	1	3	1
6	3	2	2	2
7	1	1	1	1
8	2	3	2	2
9	1	1	1	1
10	4	5	7	4
11	1	1	1	1
12	3	2	3	2
13	1	1	1	1
14	2	2	3	2
15	1	1	1	1
16	6	5	5	4
17	1	1	1	1
18	2	2	2	2
19	1	1	1	1
20	5	4	5	4
21	1	1	1	1
22	10	13	15	9
23	4	5	2	2
24	7	6	5	4

Table 7.1. Industrial Case Study # 1 – Results comparison

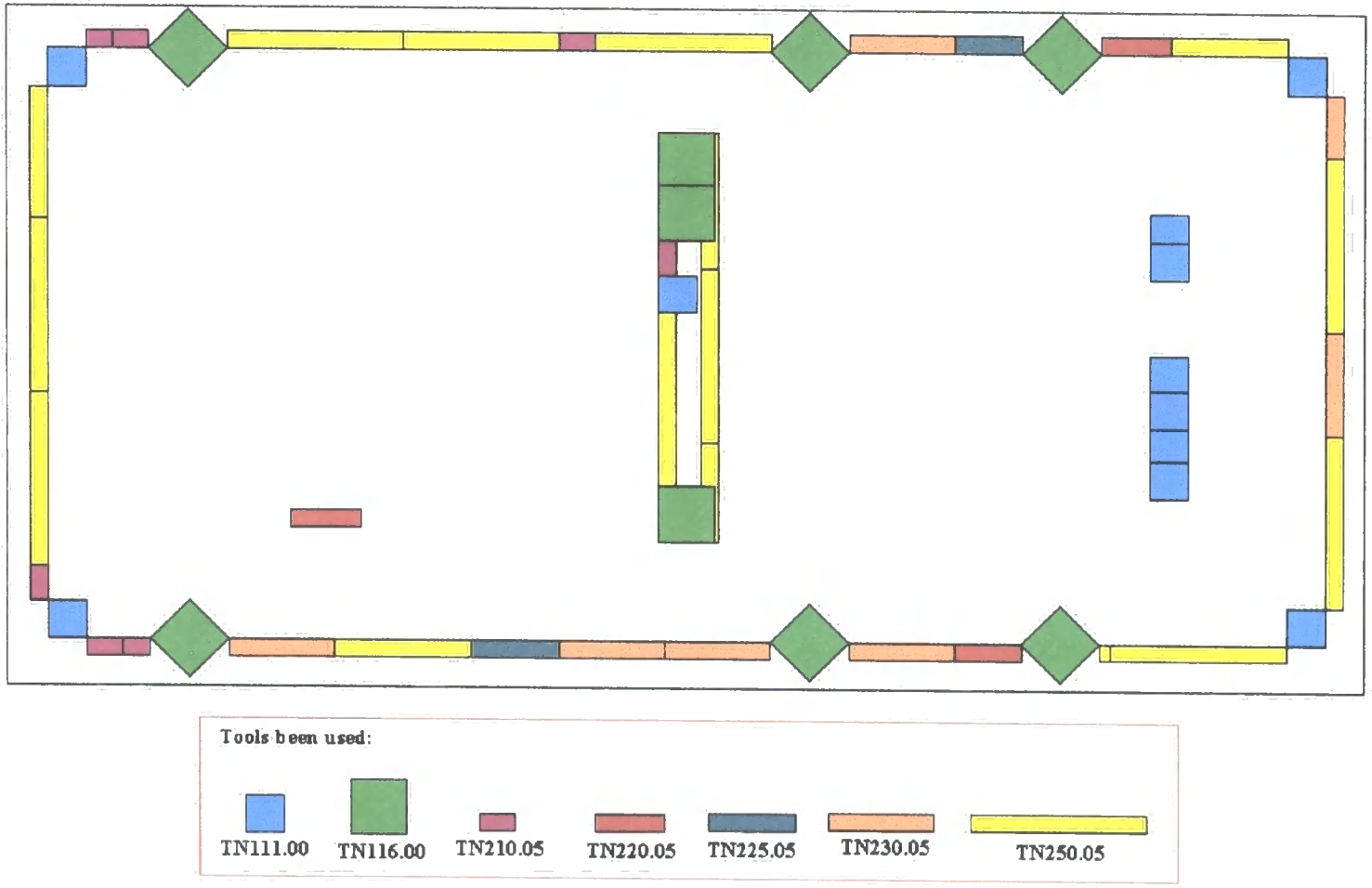


Figure 7.7. Industrial Case Study # 1 – MC Gene Manipulated Trial.

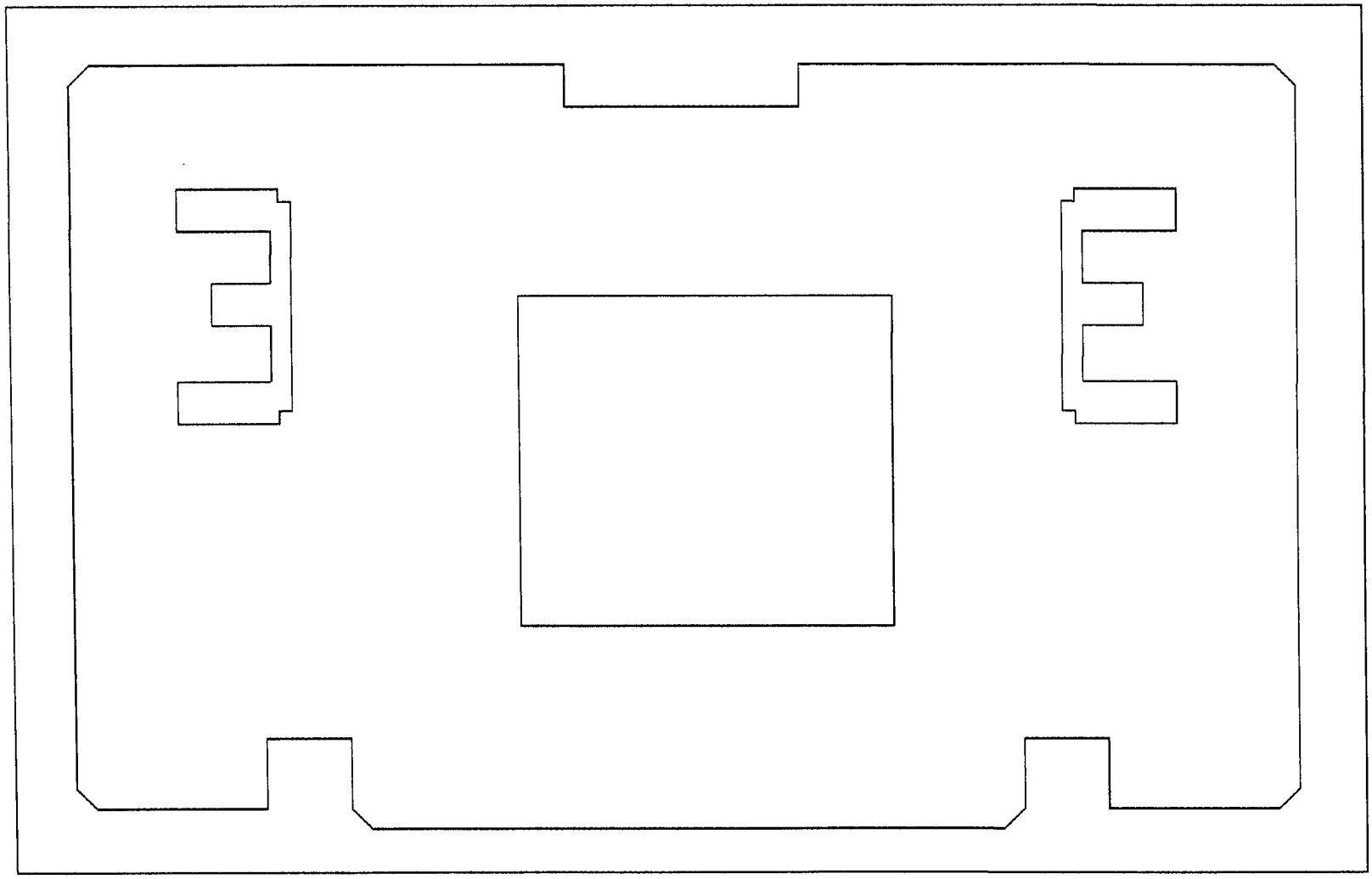


Figure 7.8. (a). Industrial Case Study # 2 – Part Description.

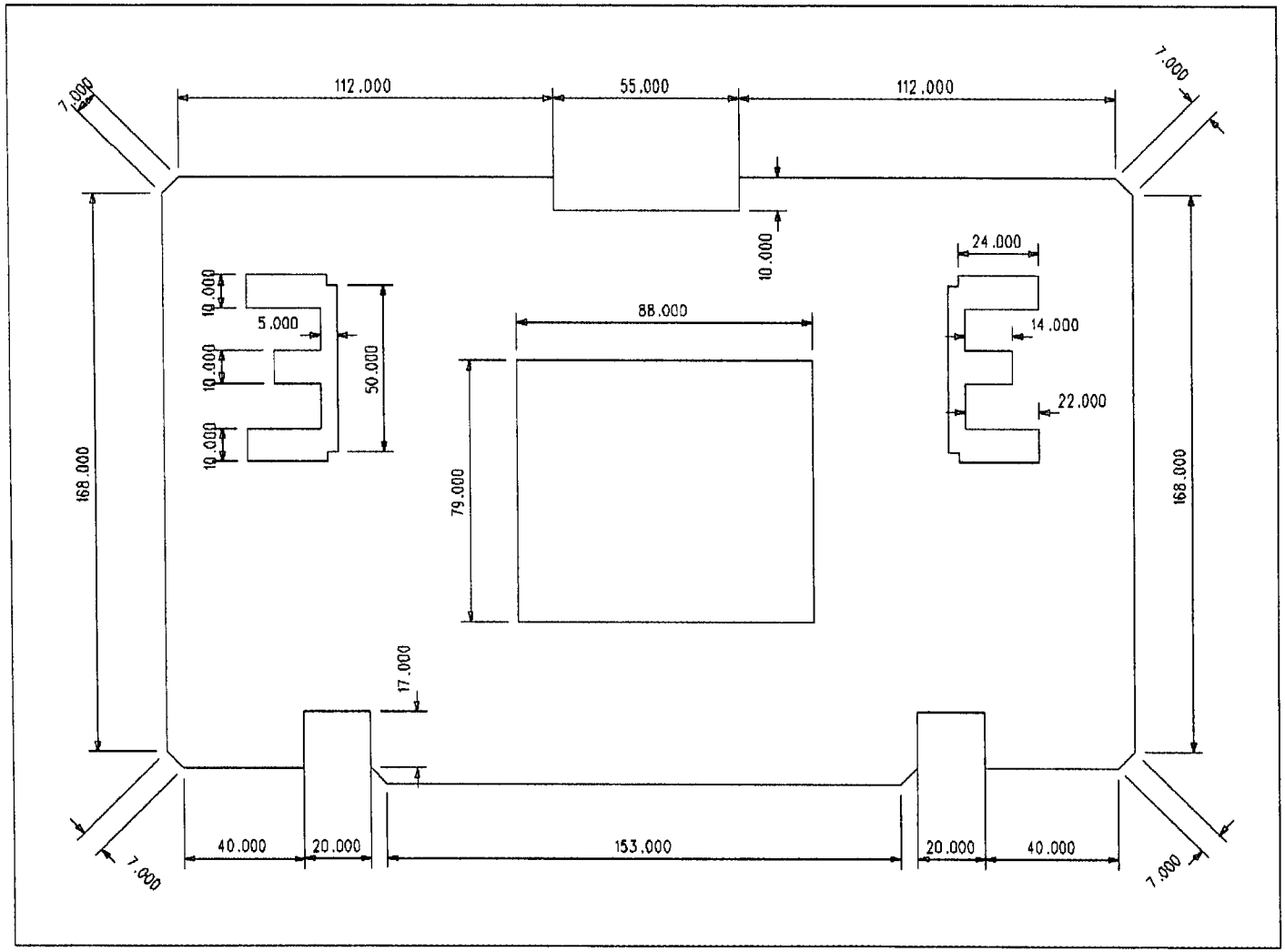


Figure 7.8. (b). Industrial Case Study # 2 – Part Description.

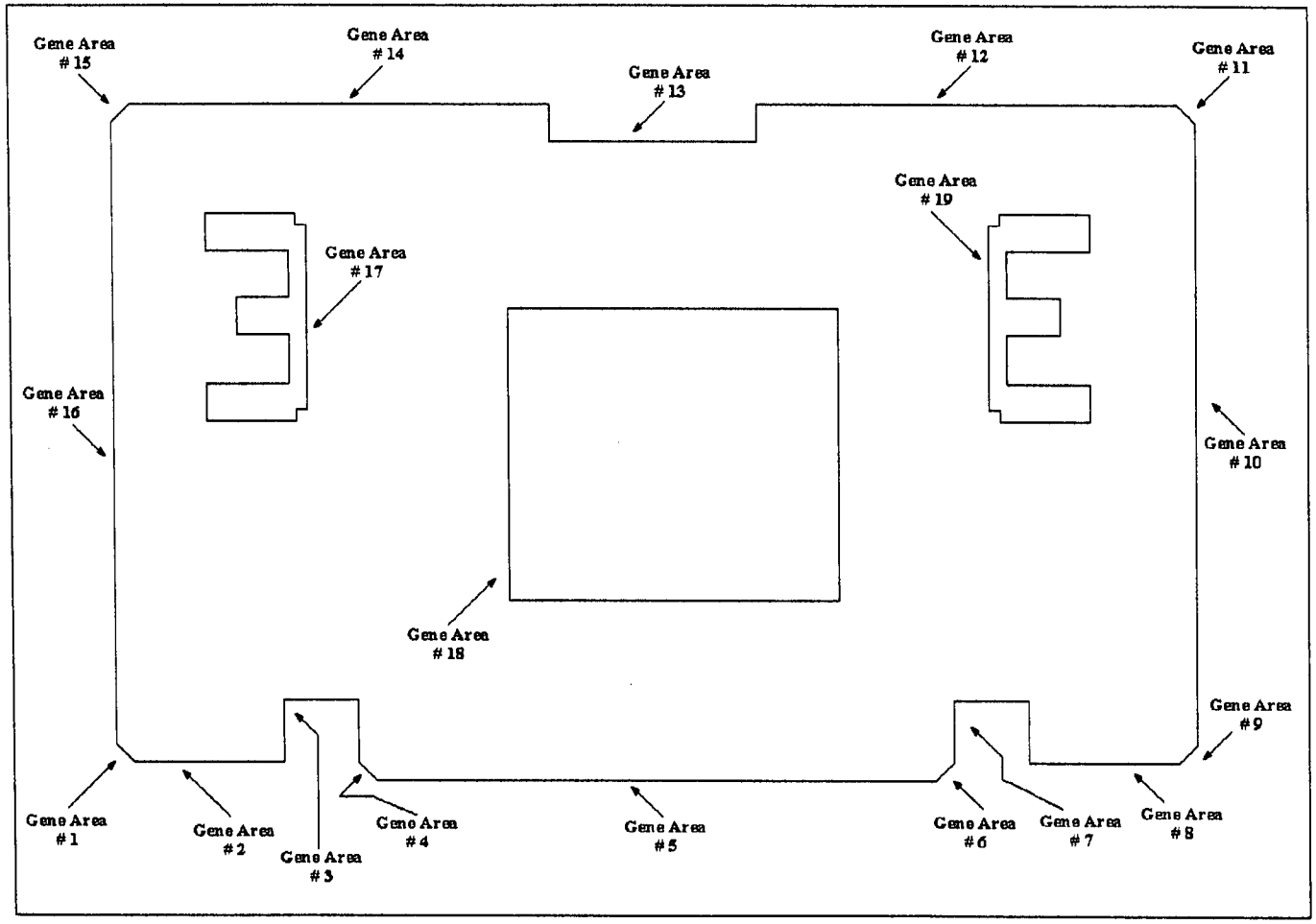


Figure 7.8. (c). Industrial Case Study # 2 – Part Description.

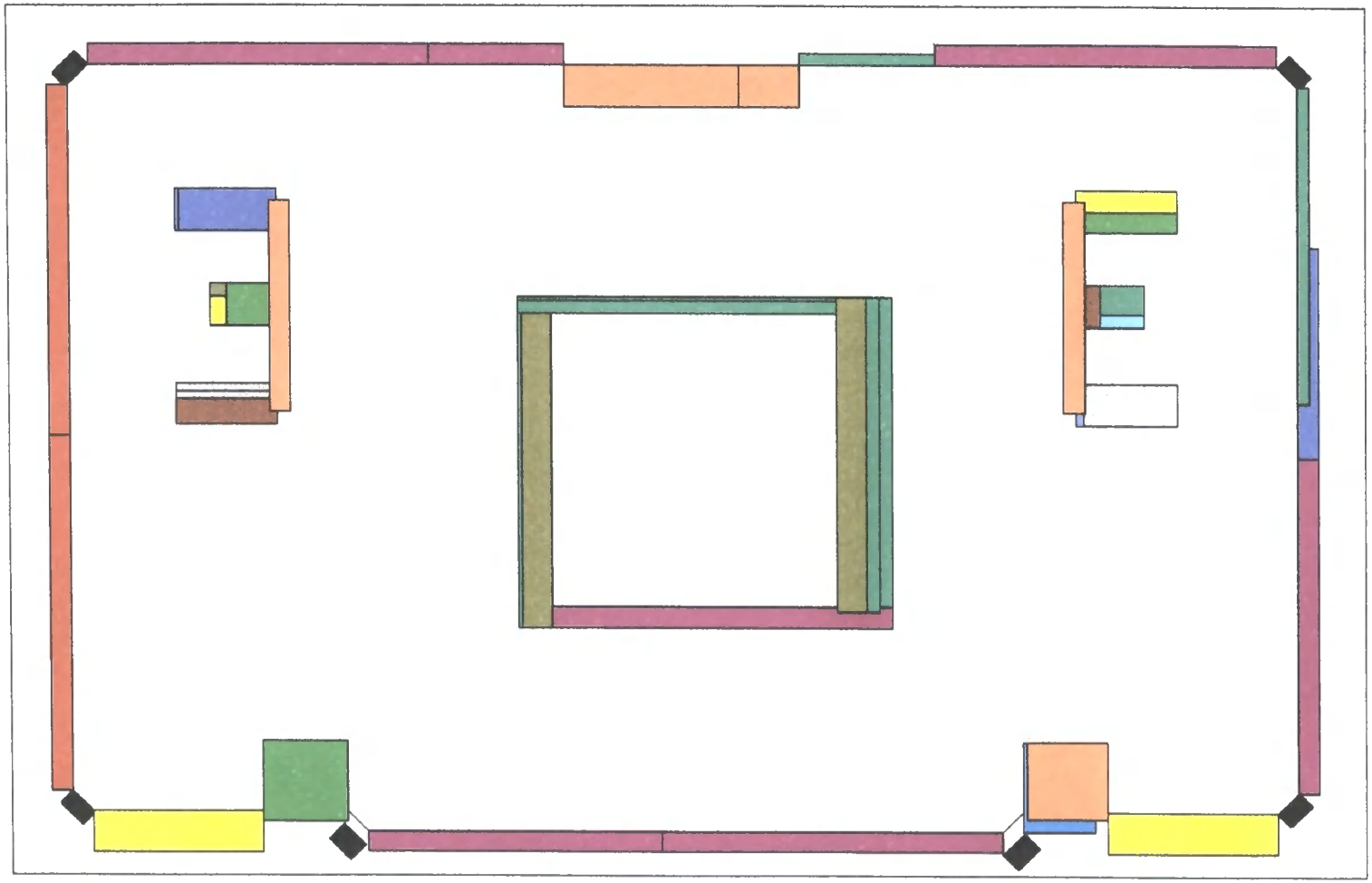


Figure 7.9. Industrial Case Study # 2 – Trial A.

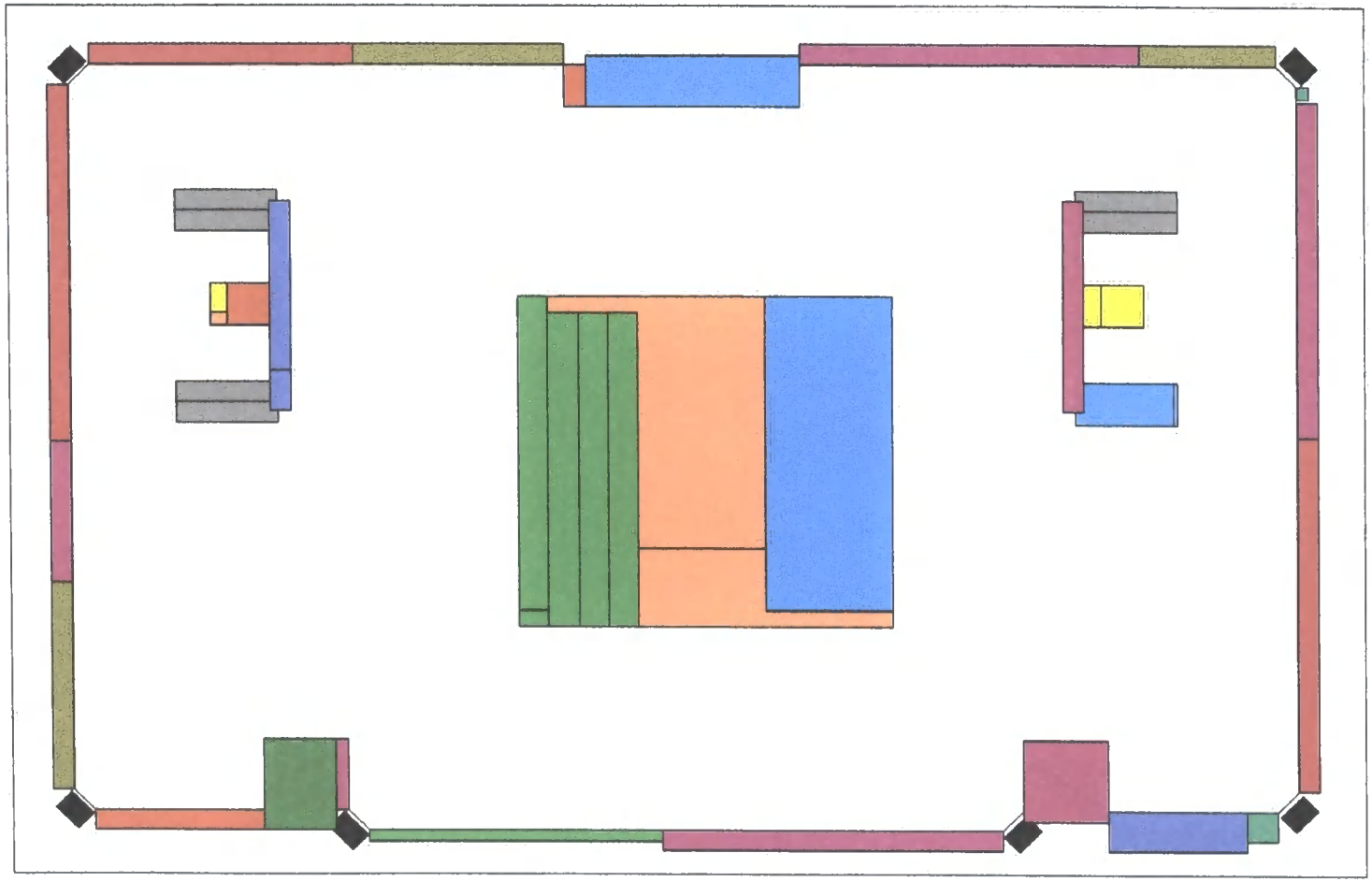


Figure 7.10. Industrial Case Study # 2 – Trial B.

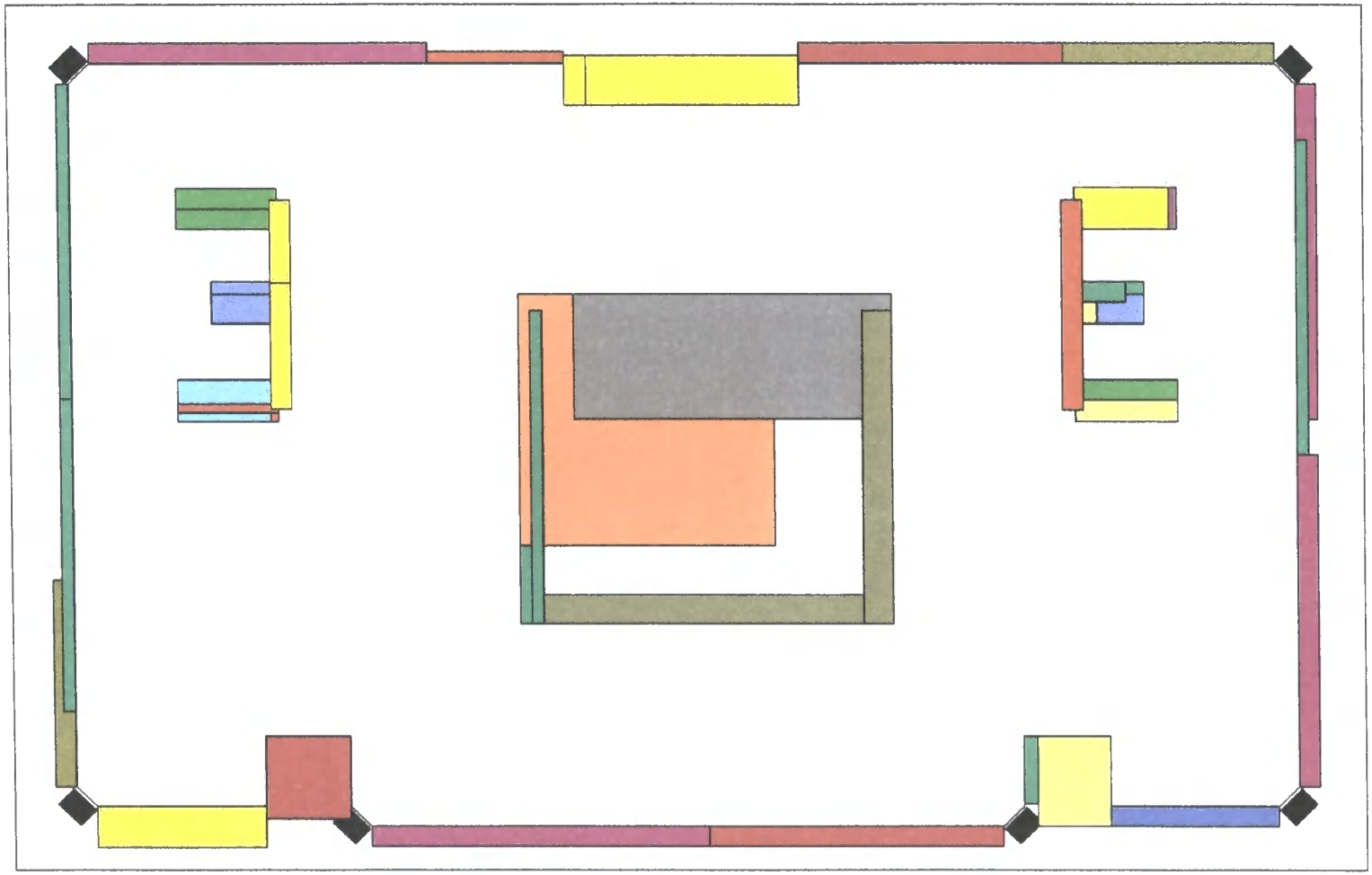


Figure 7.11. Industrial Case Study # 2 – Trial C.

	Trial A	Trial B	Trial C
Gene Area # 1			
	TN272.52	TN272.52	TN272.52
Gene Area # 2			
	TN240.10	TN240.05	TN240.10
Gene Area # 3			
	TN120.00	TN322.17	TN120.00
Gene Area # 4			
	TN272.52	TN272.52	TN272.52
Gene Area # 5			
	TN280.05	TN280.05 TN375.03	TN280.03 TN285.05
Gene Area # 6			
	TN272.52	TN272.52	TN272.52
Gene Area # 7			
	TN119.05 TN322.17	TN120.00	TN116.30 TN322.17
Gene Area # 8			
	TN240.10	TN233.07 TN233.09	TN240.05
Gene Area # 9			
	TN272.52	TN272.52	TN272.52
Gene Area # 10			
	TN250.05 TN280.05 TN375.03	TN280.05 TN285.05 TN375.03	TN280.05 TN375.03
Gene Area # 11			
	TN272.52	TN272.52	TN272.52
Gene Area # 12			
	TN280.05 TN375.03	TN250.05 TN280.05	TN250.05 TN285.05

	Trial A	Trial B	Trial C
Gene Area # 13			
	TN240.10	TN240.10 TN250.19	TN250.19
Gene Area # 14			
	TN280.05	TN250.05 TN285.05	TN247.03 TN280.05
Gene Area # 15			
	TN272.52	TN272.52	TN272.52
Gene Area # 16			
	TN285.05	TN250.05 TN280.05 TN285.05	TN250.05 TN375.03
Gene Area # 17			
	TN110.00 TN210.05 TN223.10 TN224.02 TN224.06 TN250.05 TN311.07	TN110.00 TN210.05 TN224.05 TN240.05 TN311.07	TN222.10 TN224.02 TN224.05 TN230.05 TN314.07
Gene Area # 18			
	TN275.03 TN275.07 TN280.05 TN375.03	TN160.00 TN275.07 TN275.30	TN160.00 TN275.07 TN275.30 TN375.03
Gene Area # 19			
	TN210.04 TN210.05 TN222.10 TN223.10 TN224.05 TN224.06 TN250.05 TN314.07	TN110.00 TN223.10 TN224.05 TN250.05	TN210.05 TN222.10 TN223.10 TN224.05 TN224.06 TN250.05 TN311.07 TN312.06

Table 7.2. Industrial Case Study # 2 – Tools selected

can accommodate only 20 tools, the cost function value will favour the solution of trial "B" on both solutions of trials "A" and "C", where the cost of one extra blow is far less than the cost of one extra tool change and tool set-up.

The next step was to expose these generated solutions to the "Gene Mixing and Re-Sequencing" process. Figure 7.12. shows the improved Monte Carlo solution of a cost function value of only 43.

A comparison between the three Monte Carlo generated solutions against the "Gene" manipulated Monte Carlo solution, is presented in Table 7.3. It can be seen in this table, see gene area number 17, how the gene manipulation process has improved the solution for a cost function value of only 43, see Figure 7.12. Once again, this solution can possibly be improved by mixing and re-sequencing more genes of more trials i.e., more than three trials.

7.3. INDUSTRIAL CASE STUDY 3

The product undertaken in this case study is described in Figures 7.13 (a) to (c). This product consists of 34 "gene" areas. Similarly, 100 initial Monte Carlo runs have been conducted where the best three solutions were picked. The three different Monte Carlo generated solutions are shown in Figures 7.(14) to 7.(16). These three solutions had cost function values of 64, 60 and 60. The tools used in each of these trials are shown in Table 7.4.

In these three trials, 20 tools were used in trial "A", 19 tools in trial "B" and 23 in trial "C". These also show a good example of the cost function calculation. For example, both trial "B" and "C" involved 60 blows but trial "B" used 19 tools whereas trial "C"

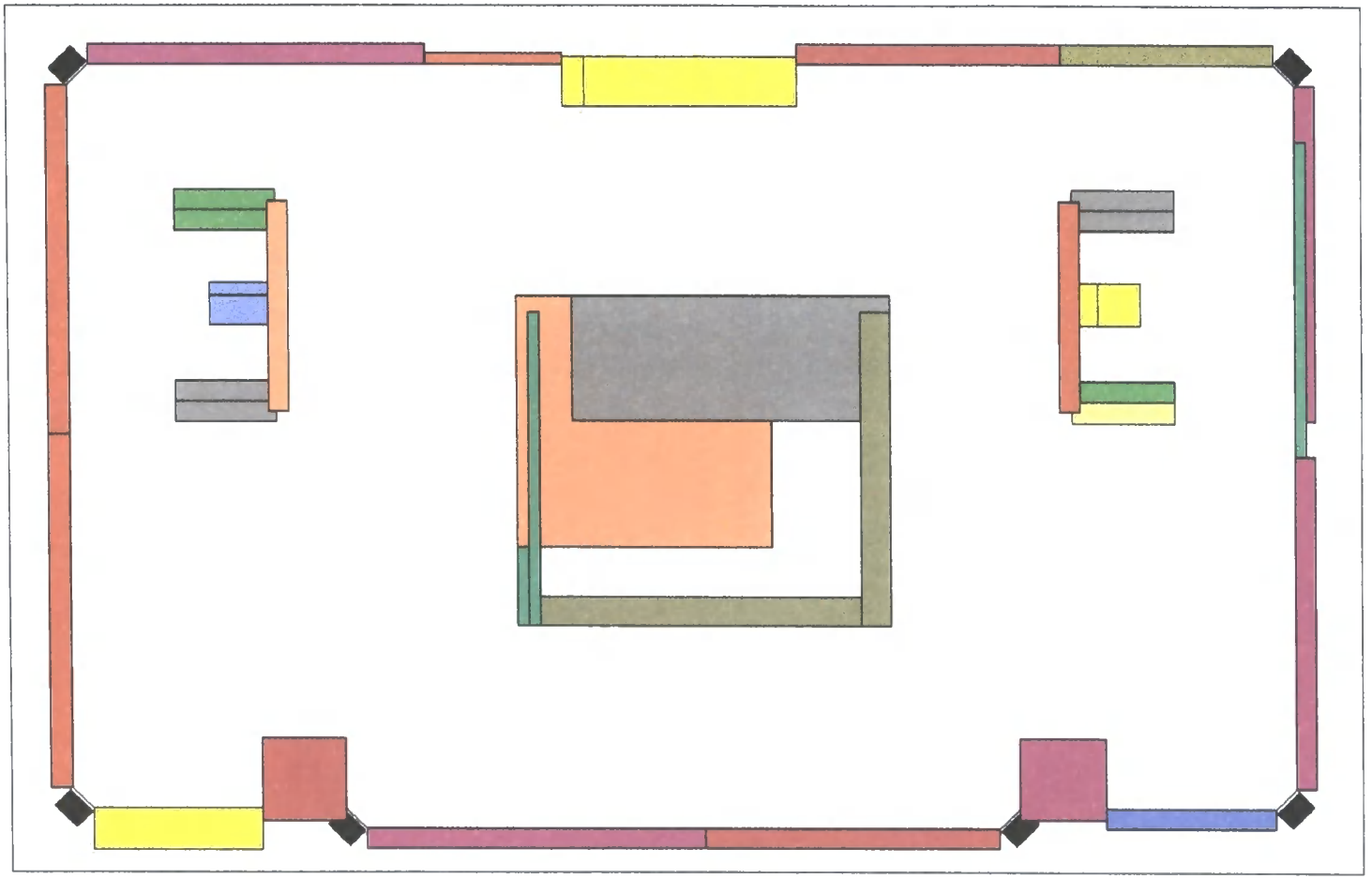


Figure 7.12. Industrial Case Study # 2 – MC Gene Manipulated Trial.

	49 BLOWS ! Trial A	50 BLOWS ! Trial B	49 BLOWS ! Trial C	43 BLOWS ! Gene Manipulated Solution
Gene Area Number	Number of Blows	Number of Blows	Number of Blows	Number of Blows
1	1	1	1	1
2	1	1	1	1
3	1	2	1	1
4	1	1	1	1
5	2	2	2	2
6	1	1	1	1
7	2	1	2	1
8	1	2	1	1
9	1	1	1	1
10	3	3	3	3
11	1	1	1	1
12	2	2	2	2
13	2	2	2	2
14	2	2	2	2
15	1	1	1	1
16	2	3	3	2
17	9	9	9	7
18	8	8	6	6
19	8	7	9	7

Table 7.3. Industrial Case Study # 2 – Results comparison

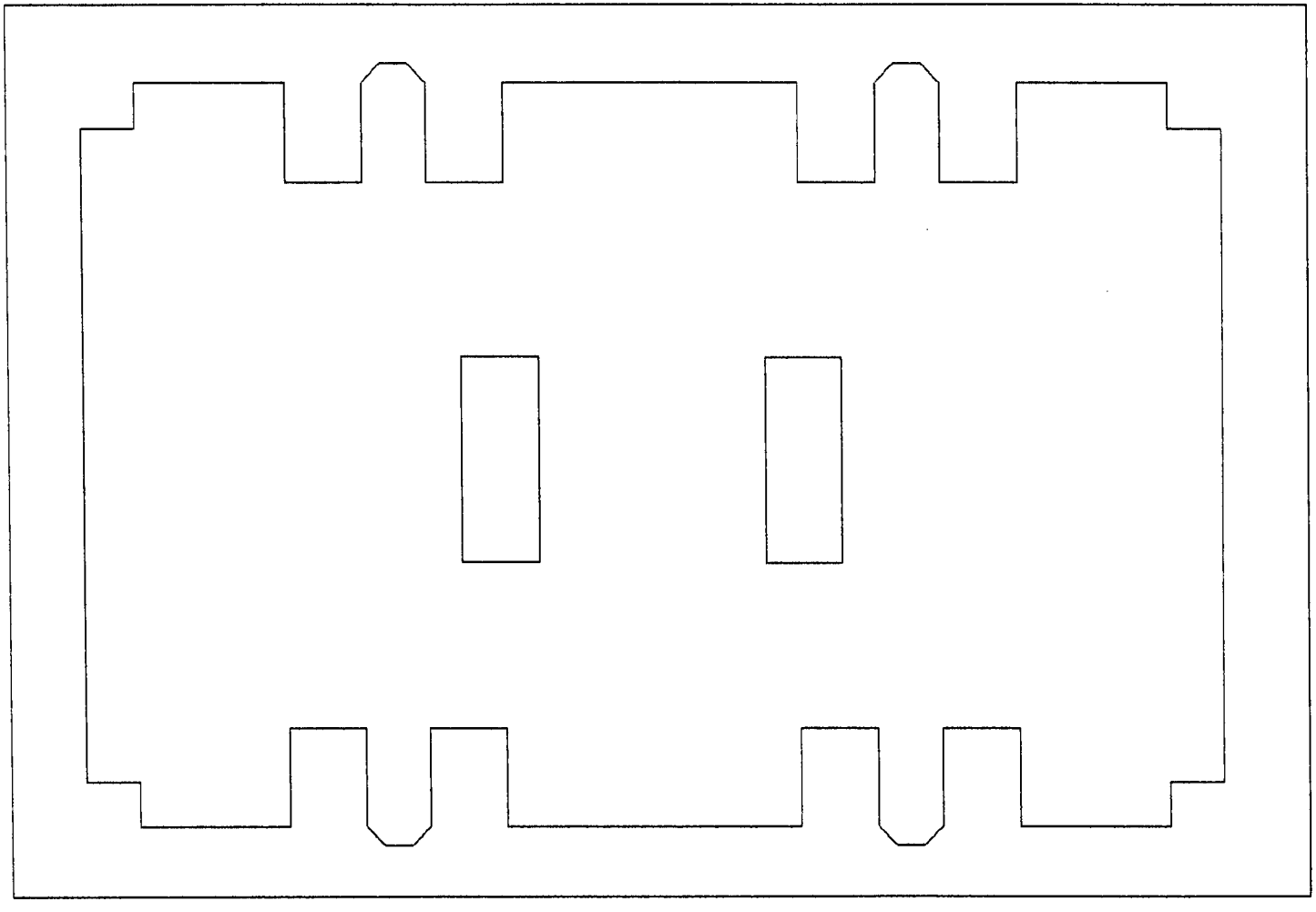


Figure 7.13. (a). Industrial Case Study # 3 – Part Description.

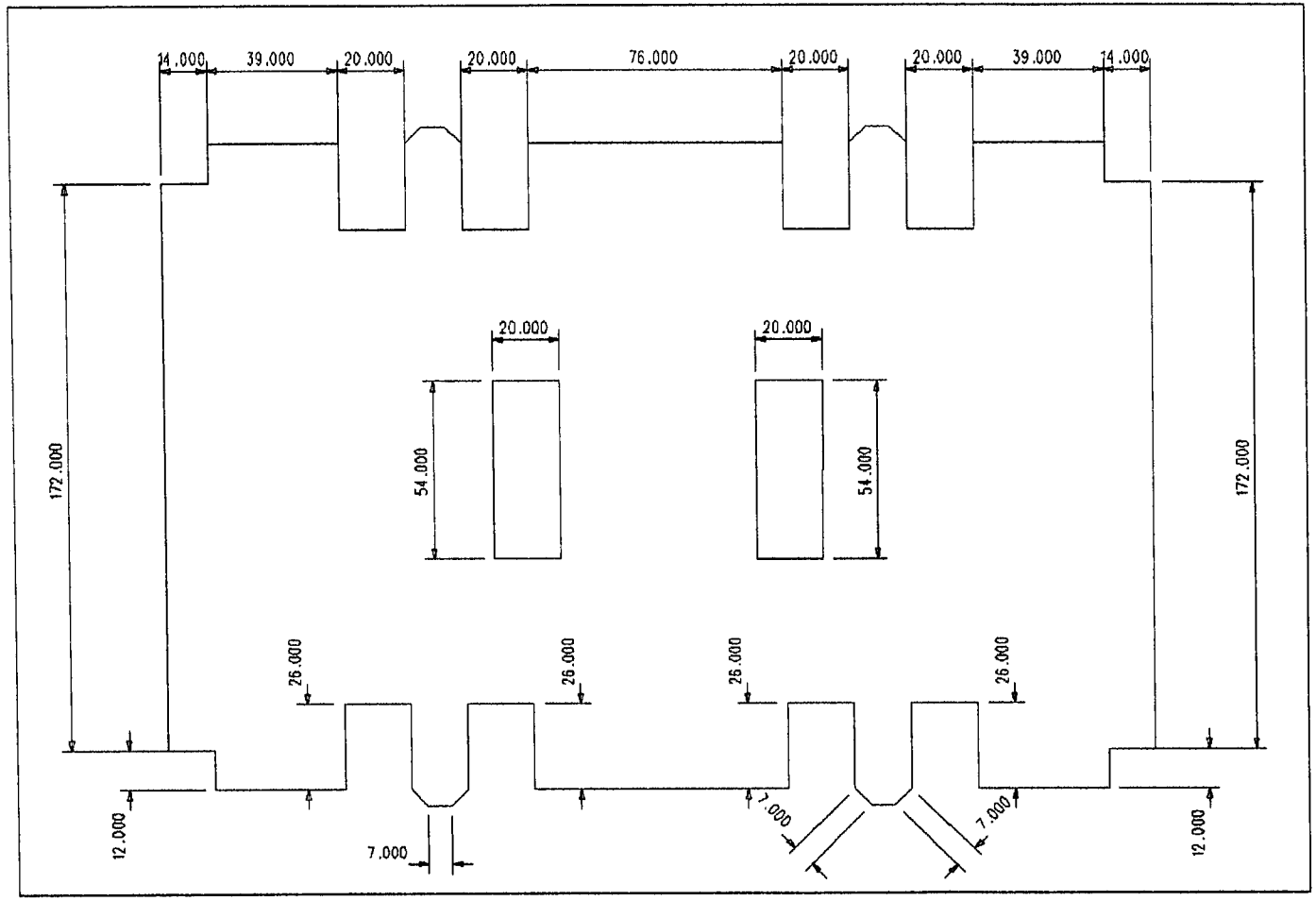


Figure 7.13. (b). Industrial Case Study # 3 – Part Description.

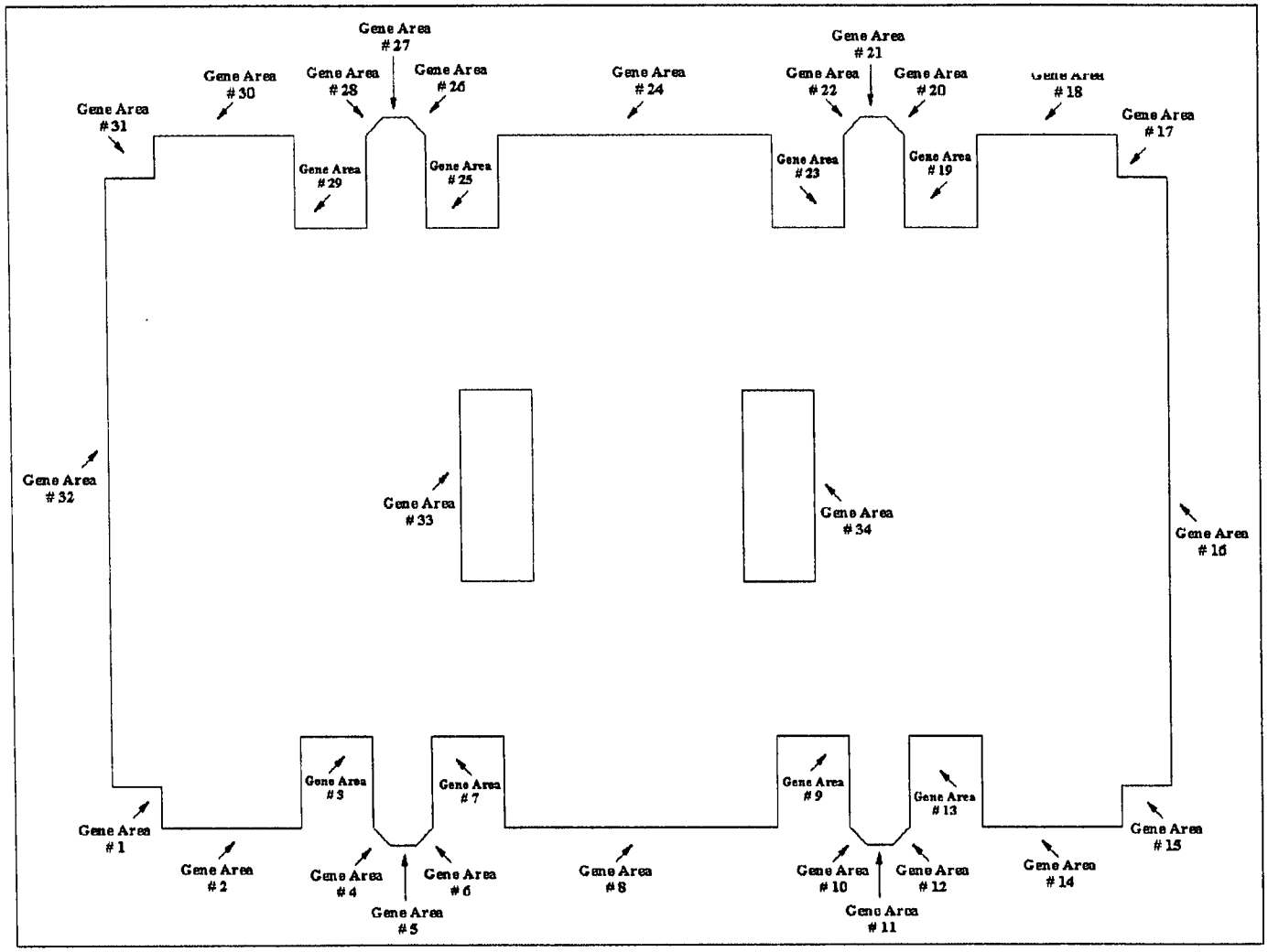


Figure 7.13. (c). Industrial Case Study # 3 – Part Description.

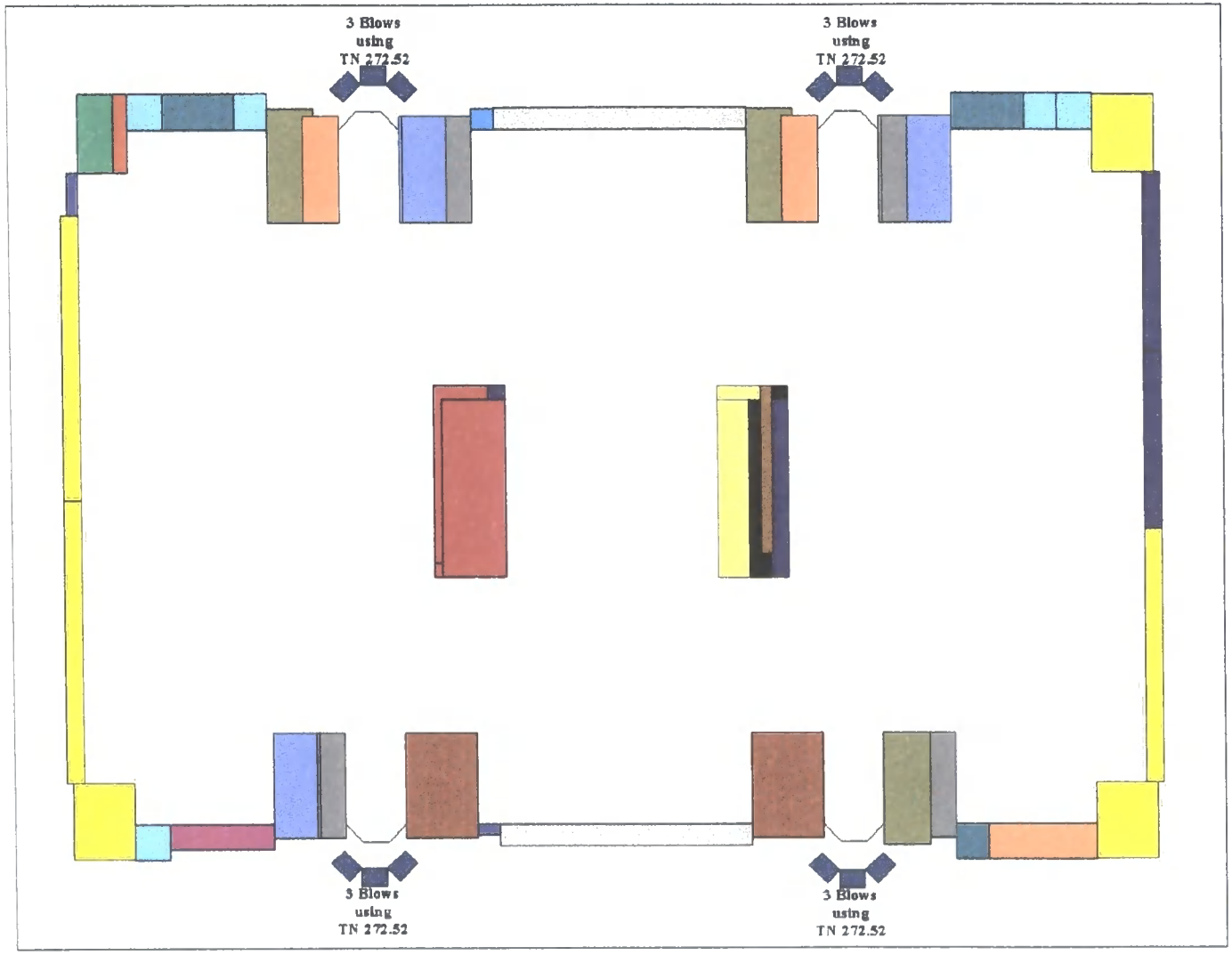


Figure 7.14. Industrial Case Study # 3 – Trial A.

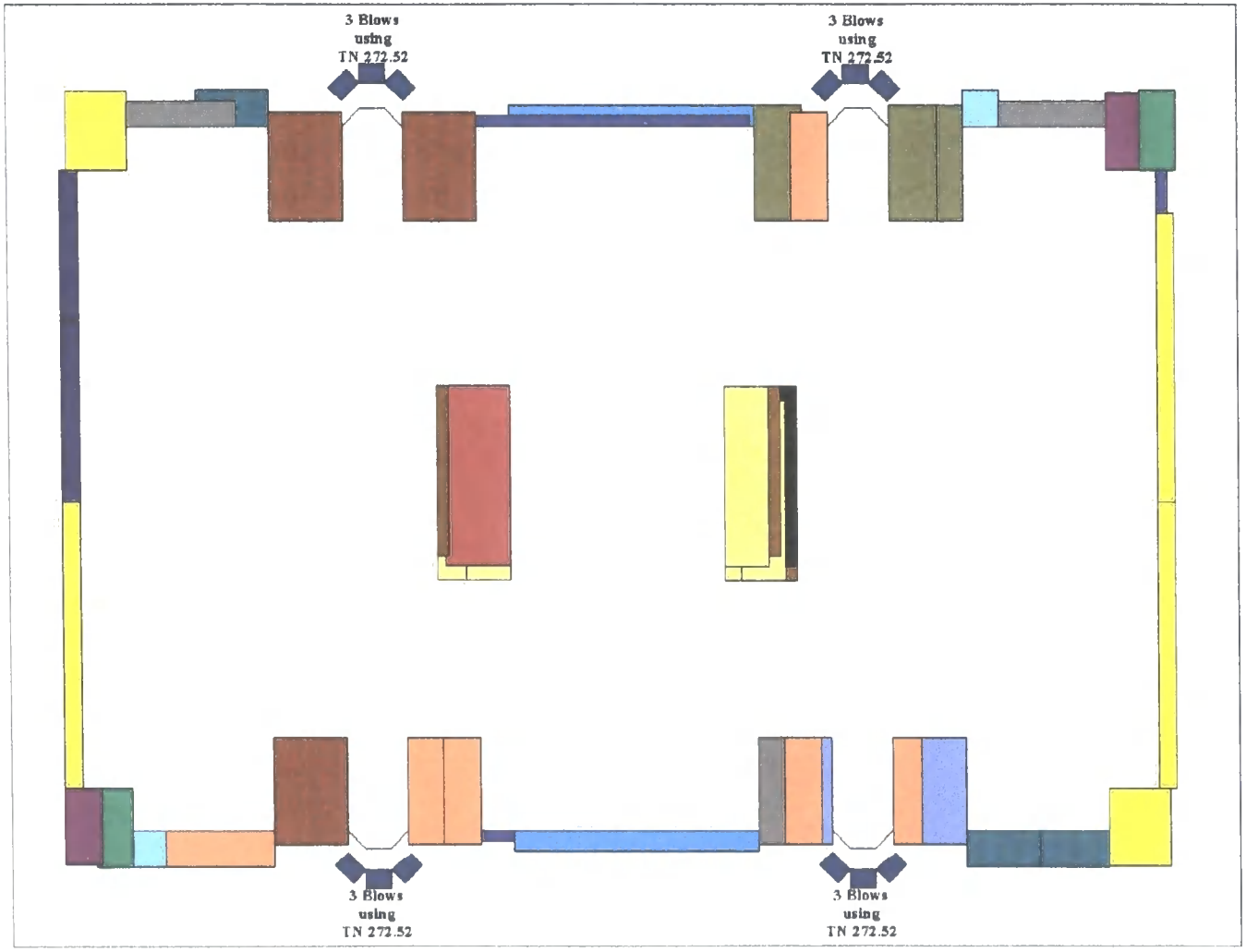


Figure 7.15. Industrial Case Study # 3 – Trial B.

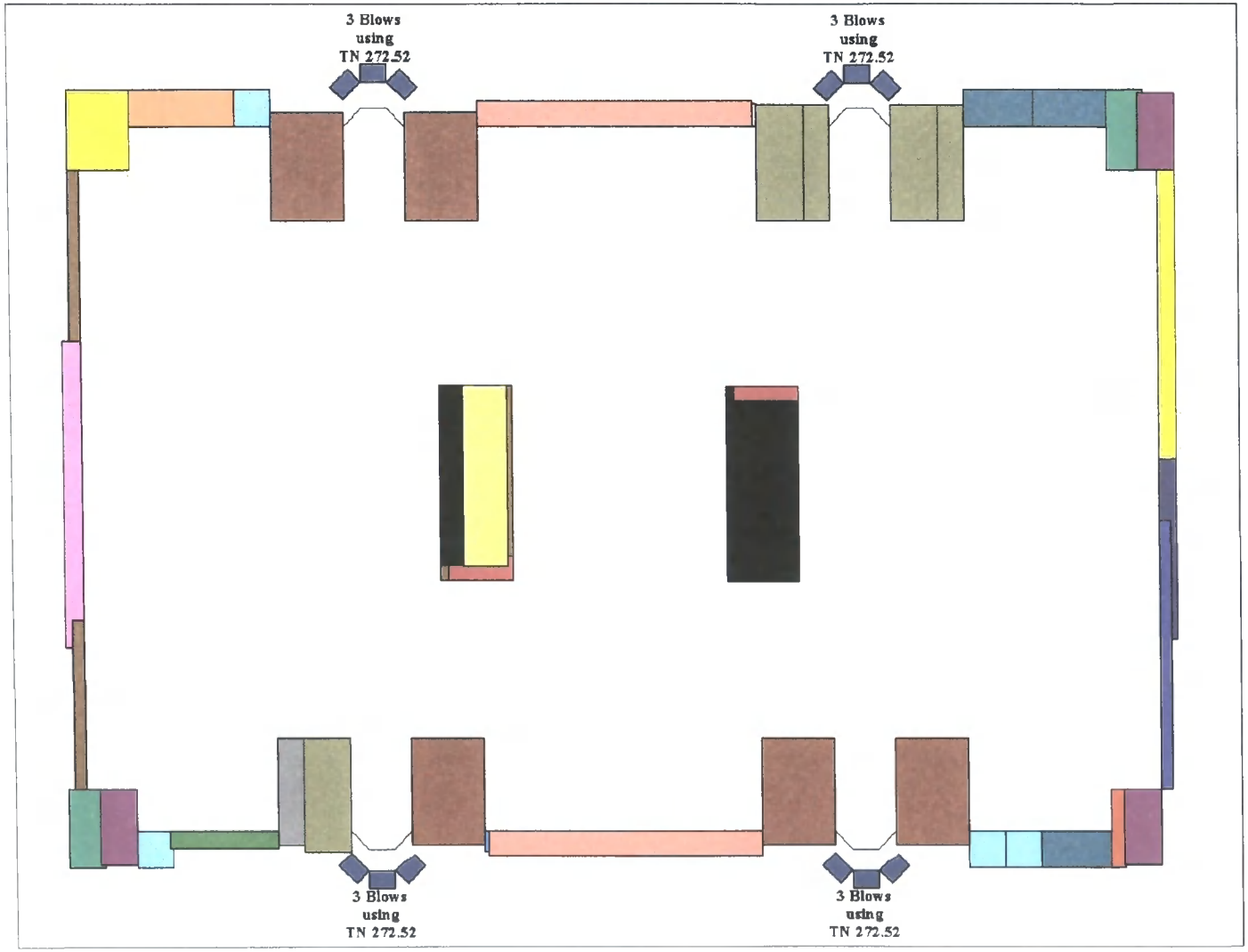


Figure 7.16. Industrial Case Study # 3 – Trial C.

Trial A		Trial B		Trial C	
Gene Area Number	Tools been used	Gene Area Number	Tools been used	Gene Area Number	Tools been used
Gene Area # 1	TN322.17	Gene Area # 1	TN222.10 TN321.10	Gene Area # 1	TN222.10 TN321.10
Gene Area # 2	TN110.00 TN330.69	Gene Area # 2	TN110.00 TN230.10	Gene Area # 2	TN110.00 TN230.05
Gene Area # 3	TN330.12 TN330.69	Gene Area # 3	TN330.20	Gene Area # 3	TN232.13 TN330.69
Gene Area # 4	TN272.52	Gene Area # 4	TN272.52	Gene Area # 4	TN272.52
Gene Area # 5	TN272.52	Gene Area # 5	TN272.52	Gene Area # 5	TN272.52
Gene Area # 6	TN272.52	Gene Area # 6	TN272.52	Gene Area # 6	TN272.52
Gene Area # 7	TN330.20	Gene Area # 7	TN230.10	Gene Area # 7	TN330.20
Gene Area # 8	TN270.06 TN375.03	Gene Area # 8	TN267.06 TN375.03	Gene Area # 8	TN267.06 TN275.07
Gene Area # 9	TN330.20	Gene Area # 9	TN230.10 TN330.12 TN330.69	Gene Area # 9	TN330.20
Gene Area # 10	TN272.52	Gene Area # 10	TN272.52	Gene Area # 10	TN272.52
Gene Area # 11	TN272.52	Gene Area # 11	TN272.52	Gene Area # 11	TN272.52
Gene Area # 12	TN272.52	Gene Area # 12	TN272.52	Gene Area # 12	TN272.52
Gene Area # 13	TN232.13 TN330.69	Gene Area # 13	TN230.10 TN330.12	Gene Area # 13	TN330.20
Gene Area # 14	TN220.10 TN230.10	Gene Area # 14	TN220.10	Gene Area # 14	TN110.00 TN220.10
Gene Area # 15	TN322.17	Gene Area # 15	TN322.17	Gene Area # 15	TN222.04 TN321.10
Gene Area # 16	TN250.05 TN280.05	Gene Area # 16	TN280.05 TN375.03	Gene Area # 16	TN250.05 TN280.05 TN375.03
Gene Area # 17	TN322.17	Gene Area # 17	TN222.10 TN321.10	Gene Area # 17	TN222.10 TN321.10

Trial A		Trial B		Trial C	
Gene Area Number	Tools been used	Gene Area Number	Tools been used	Gene Area Number	Tools been used
Gene Area # 18	TN110.00 TN220.10	Gene Area # 18	TN110.00 TN330.69	Gene Area # 18	TN220.10
Gene Area # 19	TN330.12 TN330.69	Gene Area # 19	TN232.13	Gene Area # 19	TN232.13
Gene Area # 20	TN272.52	Gene Area # 20	TN272.52	Gene Area # 20	TN272.52
Gene Area # 21	TN272.52	Gene Area # 21	TN272.52	Gene Area # 21	TN272.52
Gene Area # 22	TN272.52	Gene Area # 22	TN272.52	Gene Area # 22	TN272.52
Gene Area # 23	TN230.10 TN232.13	Gene Area # 23	TN230.10 TN232.13	Gene Area # 23	TN232.13
Gene Area # 24	TN267.06 TN270.06	Gene Area # 24	TN267.06 TN375.03	Gene Area # 24	TN270.06 TN275.07
Gene Area # 25	TN330.12 TN330.69	Gene Area # 25	TN330.20	Gene Area # 25	TN330.20
Gene Area # 26	TN272.52	Gene Area # 26	TN272.52	Gene Area # 26	TN272.52
Gene Area # 27	TN272.52	Gene Area # 27	TN272.52	Gene Area # 27	TN272.52
Gene Area # 28	TN272.52	Gene Area # 28	TN272.52	Gene Area # 28	TN272.52
Gene Area # 29	TN230.10 TN232.13	Gene Area # 29	TN330.20	Gene Area # 29	TN330.20
Gene Area # 30	TN110.00 TN220.10	Gene Area # 30	TN220.10 TN330.69	Gene Area # 30	TN110.00 TN230.10
Gene Area # 31	TN222.04 TN222.10	Gene Area # 31	TN322.17	Gene Area # 31	TN322.17
Gene Area # 32	TN280.05 TN375.03	Gene Area # 32	TN250.05 TN280.05	Gene Area # 32	TN247.03 TN285.05
Gene Area # 33	TN250.05 TN350.18	Gene Area # 33	TN247.03 TN250.19 TN350.18	Gene Area # 33	TN247.03 TN250.06 TN250.19 TN350.18
Gene Area # 34	TN247.03 TN250.05 TN250.06 TN250.19	Gene Area # 34	TN247.03 TN250.06 TN250.19	Gene Area # 34	TN250.06 TN350.18

Table 7.4 Industrial Case Study # 3 – Tools selected

used 23. If it is assumed that the machine can accommodate only 19 tools then the solution of trial "C" will need an extra tool change and tool set-up which will acquire a cost penalty. Therefore, the cost function value, if the machine turret can accommodate only 19 tools, will reflect such a cost penalty which will favour less the solution of trial "C". Similarly, if the turret can accommodate 20 tools , then the cost function will even show that the solution of trial "A" is more favoured than the solutions of trial "C". Although trial "A" would involve four more extra blows, it will not require an extra tool set-up as with trial "C" to accommodate the extra three tools needed.

The next step was to expose these generated solutions to the "Gene Mixing and Re-Sequencing" process. Figure 7.17. shows the improved Monte Carlo solution of a cost function value of only 53.

In Table 7.5. a comparison between the three Monte Carlo generated solutions against the Monte Carlo "Gene" manipulated solution is presented. Table 7.5. shows, see gene area number 34, how the gene manipulation solution can improve the solution, which leads to an enhanced cost function value of only 53.

INDUSTRIAL CASE STUDY 4

The product in this case study is described in Figures 7.18 (a) and (b). This product consists of 67 "gene" areas. Table 7.6. shows the sizes of these 67 gene areas and the tools available in the machine turret. The 67 gene areas can be grouped into 8 size-groups as shown in Table 7.6 and can therefore be treated as only 8 discrete gene areas. It was decided to treat each of these gene areas individually in order to see how the Monte Carlo search mechanism would operate in a situation where human intelligence may decide that the best solution can be by a "one to one" feature-tool match, resulting in 67 blows. Again 100 initial Monte Carlo runs have been conducted where

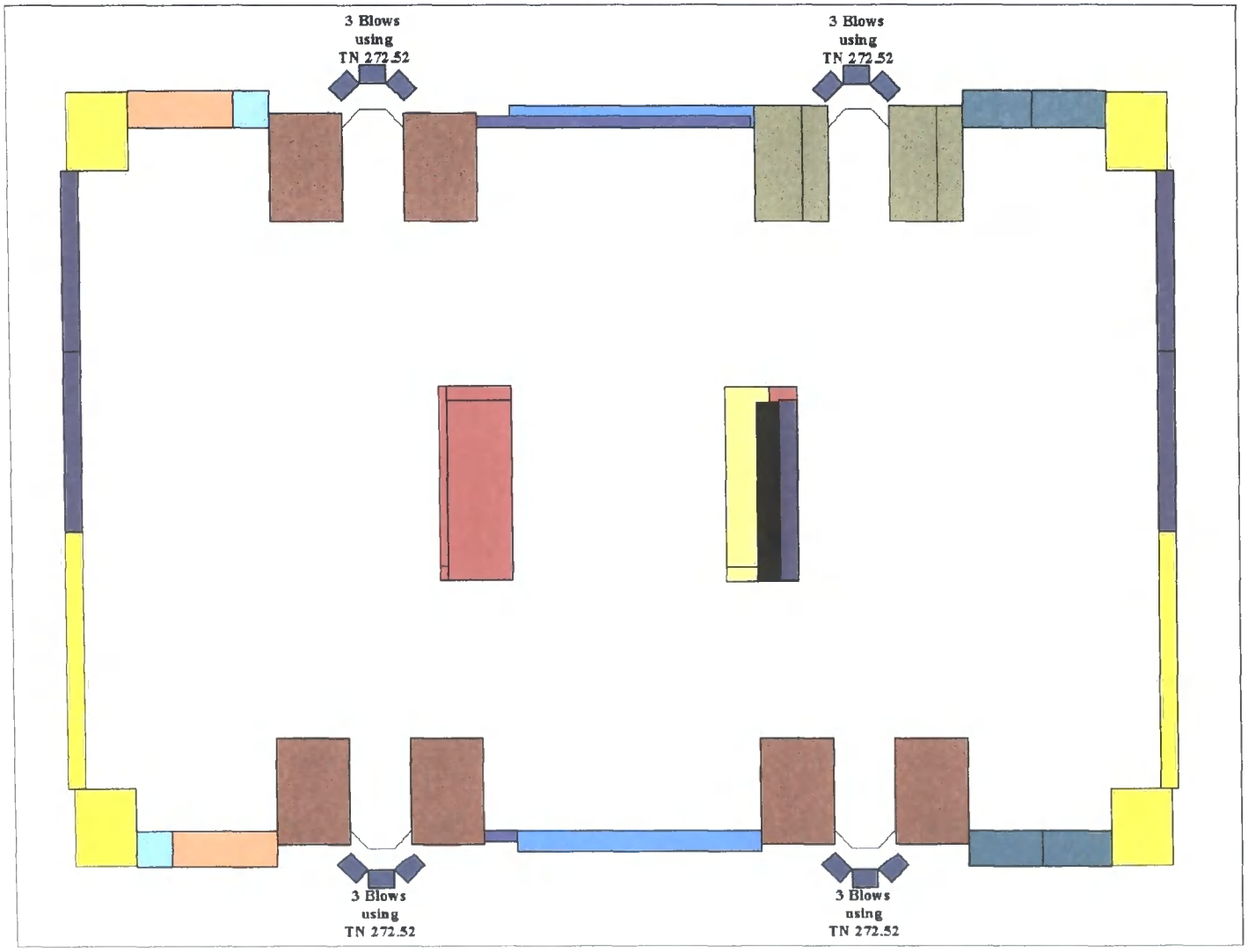


Figure 7.17. Industrial Case Study # 3 – MC Gene Manipulated Trial.

	64 BLOWS ! Trial A	60 BLOWS ! Trial B	60 BLOWS ! Trial C	53 BLOWS ! Gene Manipulated Solution
Gene Area Number	Number of Blows	Number of Blows	Number of Blows	Number of Blows
1	1	2	2	1
2	2	2	2	2
3	3	1	2	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	2	1	1
8	2	2	2	2
9	1	3	1	1
10	1	1	1	1
11	1	1	1	1
12	1	1	1	1
13	2	2	1	1
14	2	2	3	2
15	1	1	2	1
16	3	3	3	3
17	1	2	2	1
18	3	2	2	2
19	3	2	2	2
20	1	1	1	1
21	1	1	1	1
22	1	1	1	1
23	2	2	2	2
24	2	2	2	2
25	3	1	1	1
26	1	1	1	1
27	1	1	1	1
28	1	1	1	1
29	2	1	1	1
30	3	2	2	2
31	2	1	1	1
32	3	3	3	3
33	4	4	5	4
34	6	6	6	5

Table 7.5. Industrial Case Study # 3 – Results comparison

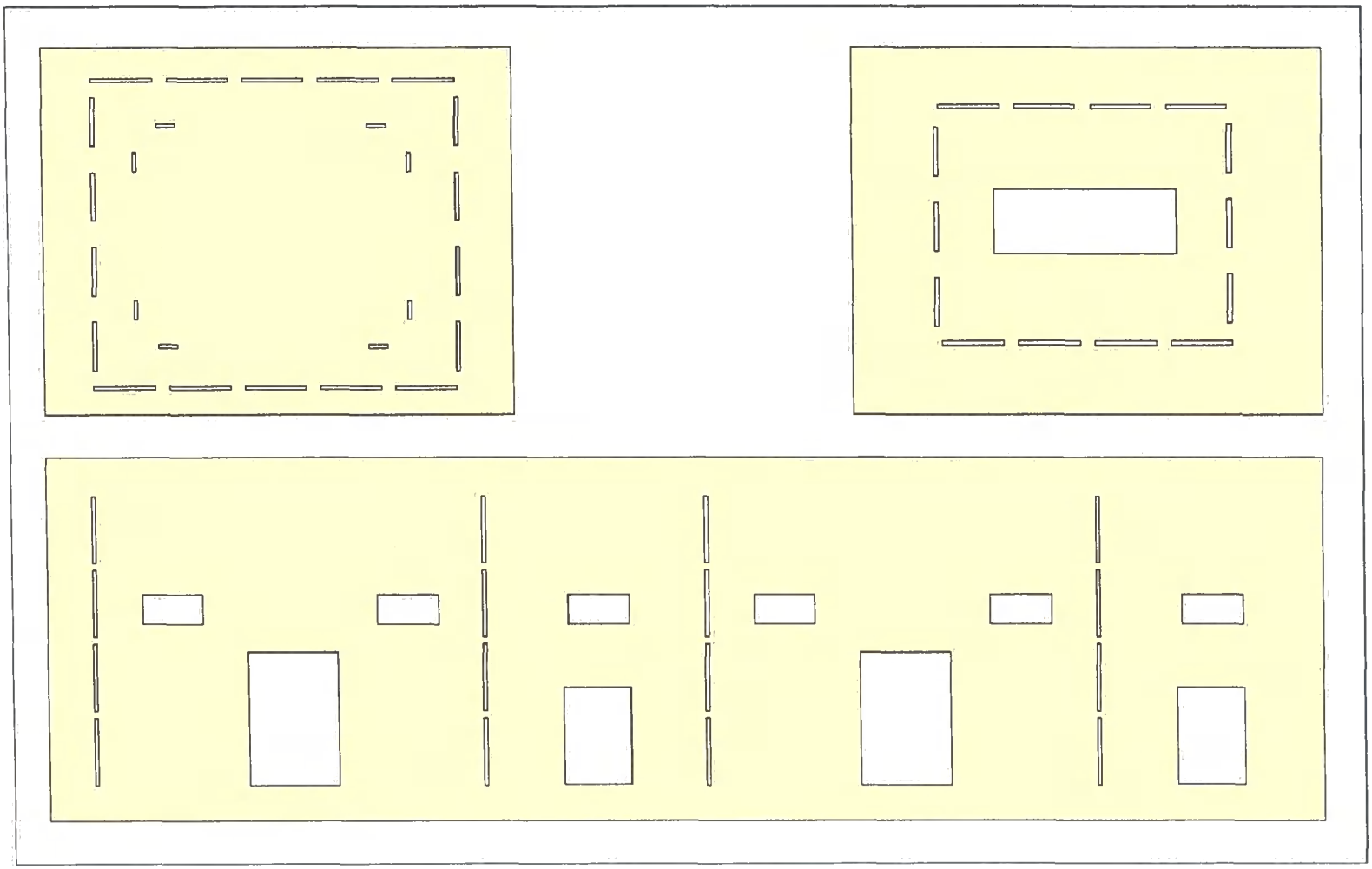


Figure 7.18. (a). Industrial Case Study # 4 – Part Description.

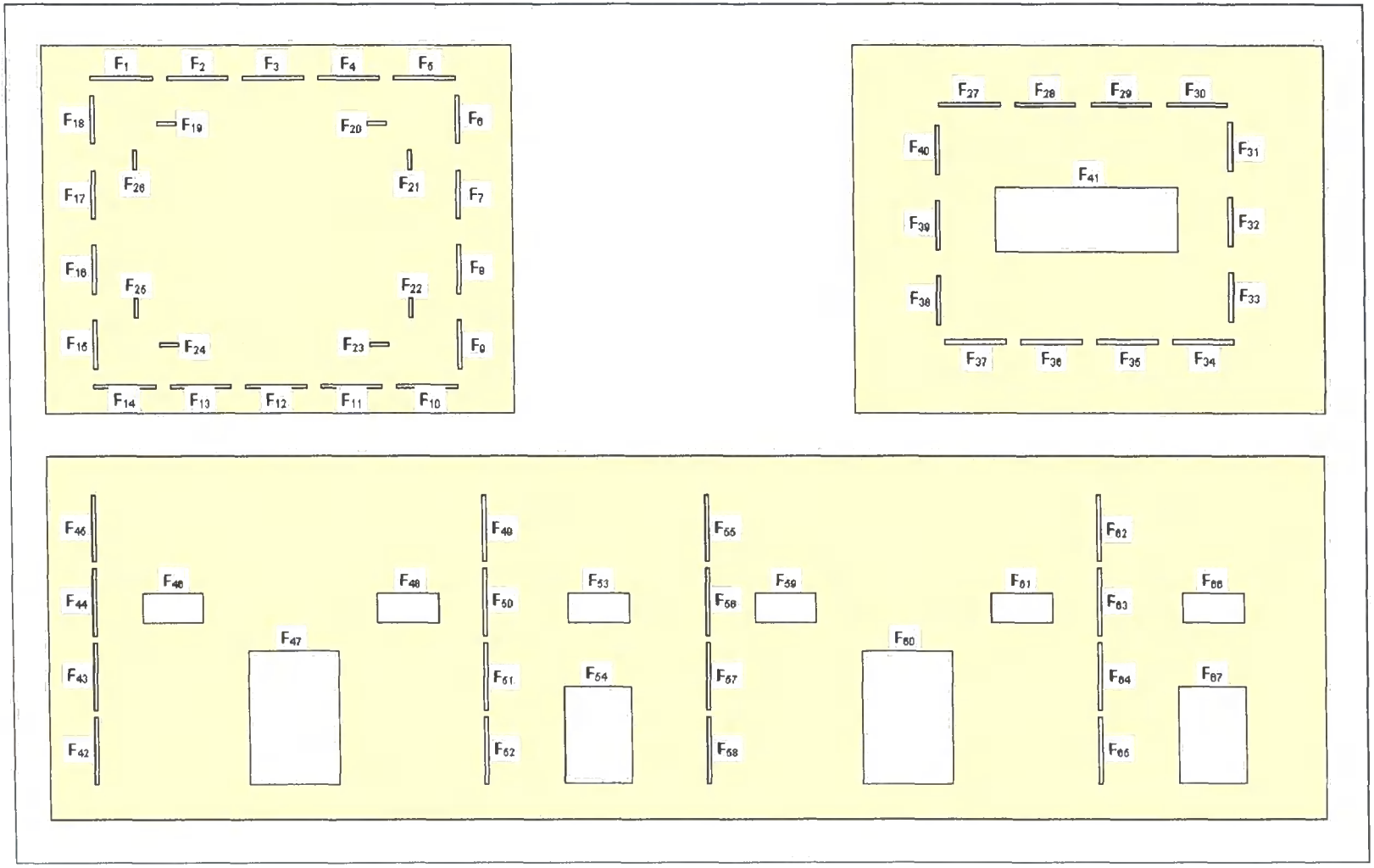


Figure 7.18. (b). Industrial Case Study # 4 – Part Description.

Gene Area	Size
F_{41}	48 x 17
F_{47}	36 x 24
F_{60}	36 x 24
F_{54}	26 x 18
F_{67}	26 x 18
F_{46}	16 x 8
F_{48}	16 x 8
F_{53}	16 x 8
F_{59}	16 x 8
F_{61}	16 x 8
F_{66}	16 x 8
F_{42}	18 x 1
F_{43}	18 x 1
F_{44}	18 x 1
F_{45}	18 x 1
F_{49}	18 x 1
F_{50}	18 x 1
F_{51}	18 x 1
F_{52}	18 x 1
F_{55}	18 x 1
F_{56}	18 x 1
F_{57}	18 x 1
F_{58}	18 x 1
F_{62}	18 x 1
F_{63}	18 x 1
F_{64}	18 x 1
F_{65}	18 x 1

Gene Area	Size
F_1	16 x 1
F_2	16 x 1
F_3	16 x 1
F_4	16 x 1
F_5	16 x 1
F_{10}	16 x 1
F_{11}	16 x 1
F_{12}	16 x 1
F_{13}	16 x 1
F_{14}	16 x 1
F_{27}	16 x 1
F_{28}	16 x 1
F_{29}	16 x 1
F_{30}	16 x 1
F_{34}	16 x 1
F_{35}	16 x 1
F_{36}	16 x 1
F_{37}	16 x 1
F_6	13 x 1
F_7	13 x 1
F_8	13 x 1
F_9	13 x 1
F_{15}	13 x 1
F_{16}	13 x 1
F_{17}	13 x 1
F_{18}	13 x 1
F_{31}	13 x 1
F_{32}	13 x 1
F_{33}	13 x 1
F_{38}	13 x 1
F_{39}	13 x 1
F_{40}	13 x 1
F_{19}	5 x 1
F_{20}	5 x 1
F_{21}	5 x 1
F_{22}	5 x 1
F_{23}	5 x 1
F_{24}	5 x 1
F_{25}	5 x 1
F_{26}	5 x 1

Tool Number	Size
Tool 1	48 x 17
Tool 2	36 x 24
Tool 3	26 x 18
Tool 4	16 x 8
Tool 5	18 x 1
Tool 6	16 x 1
Tool 7	13 x 1
Tool 8	5 x 1

Table 7.6. Industrial Case Study # 4 – Gene areas and tools sizes

the best three solutions were picked. These three solutions had cost function values of 113, 148 and 157. These three different Monte Carlo generated solutions are summarised in Table 7.7.

The next step was to expose these three generated solutions to the "Gene Mixing and Re-Sequencing". Table 7.7. and Figure 7.19. show how the second phase of the Monte Carlo search, namely "Gene mixing and Re-Sequencing", has improved Monte Carlo solution for a cost function value of only 75.

In Table 7.3., see gene area number $\mathbf{F}_{47}$, the gene manipulation process has lead to an improved solution giving a cost function value of only 75. Obviously, this solution can be improved by conducting the process of gene mixing and re-sequencing for more trials i.e., more than three, bearing in mind that gene area numbers $\mathbf{F}_{10}$ and $\mathbf{F}_{35}$ belong to the size-group "16 x 1", where the system has already found the one to one match for the other similar gene areas. Additionally, gene areas $\mathbf{F}_{42}$, $\mathbf{F}_{52}$ and $\mathbf{F}_{58}$ belong to the size-group "18 x 1" and the system has also already found a one to one match for the other similar gene areas. Similarly for gene area **F**47 the system found a one to one match for the sister gene area of size "36 x 24".

Finally, the author hopes that the above discussed industrial case studies have shown how efficiently the Monte Carlo search can operate in searching for the best set of tools to punch a given sheet metal part. It is worth mentioning here that with the advancement of computer technology, the generation of 100 Monte Carlo solutions does not represent a time concern especially in the light of ever faster hardware becoming available.

	113 Blows ! Trial A	148 Blows ! Trial B	157 Blows ! Trial C	75 Blows ! Gene Manipulated Trial
Gene Area	Number of Blows	Number of Blows	Number of Blows	Number of Blows
F_1	1	2	2	1
F_2	1	1	1	1
F_3	1	1	1	1
F_4	1	2	2	1
F_5	1	1	1	1
F_6	1	3	1	1
F_7	3	3	1	1
F_8	3	1	1	1
F_9	1	1	1	1
F_{10}	2	2	4	2
F_{11}	1	1	1	1
F_{12}	1	1	2	1
F_{13}	4	1	1	1
F_{14}	1	1	2	1
F_{15}	1	1	1	1
F_{16}	1	1	1	1
F_{17}	1	3	1	1
F_{18}	1	1	1	1
F_{19}	1	1	1	1
F_{20}	1	1	1	1
F_{21}	1	1	1	1
F_{22}	1	1	1	1

Table 7.7. Industrial Case Study # 4 – Results comparison

	113 Blows ! Trial A	148 Blows ! Trial B	157 Blows ! Trial C	75 Blows ! Gene Manipulated Trial
Gene Area	Number of Blows	Number of Blows	Number of Blows	Number of Blows
F_{23}	1	1	1	1
F_{24}	1	1	1	1
F_{25}	1	1	1	1
F_{26}	1	1	1	1
F_{27}	1	1	1	1
F_{28}	1	2	1	1
F_{29}	1	2	2	1
F_{30}	1	1	4	1
F_{31}	3	1	1	1
F_{32}	3	1	1	1
F_{33}	1	1	1	1
F_{34}	1	2	2	1
F_{35}	4	2	2	2
F_{36}	4	2	1	1
F_{37}	2	1	1	1
F_{38}	1	1	3	1
F_{39}	1	3	1	1
F_{40}	1	1	3	1
F_{41}	1	24	1	1
F_{42}	2	2	2	2
F_{43}	1	1	2	1
F_{44}	1	2	3	1

Table 7.7. Industrial Case Study # 4 – Results comparison Continue...

	113 Blows ! Trial A	148 Blows ! Trial B	157 Blows ! Trial C	75 Blows ! Gene Manipulated Trial
Gene Area	Number of Blows	Number of Blows	Number of Blows	Number of Blows
F_{45}	1	2	2	1
F_{46}	1	1	1	1
F_{47}	7	12	12	4
F_{48}	13	1	13	1
F_{49}	1	2	2	1
F_{50}	2	2	1	1
F_{51}	2	2	1	1
F_{52}	2	2	2	2
F_{53}	1	12	8	1
F_{54}	1	1	10	1
F_{55}	1	2	1	1
F_{56}	1	2	1	1
F_{57}	2	1	2	1
F_{58}	2	2	2	2
F_{59}	1	1	1	1
F_{60}	1	9	15	1
F_{61}	1	1	1	1
F_{62}	2	1	1	1
F_{63}	2	3	1	1
F_{64}	2	1	1	1
F_{65}	1	1	1	1
F_{66}	1	1	12	1
F_{67}	1	1	1	1

Table 7.7. Industrial Case Study # 4 – Results comparison Continue...

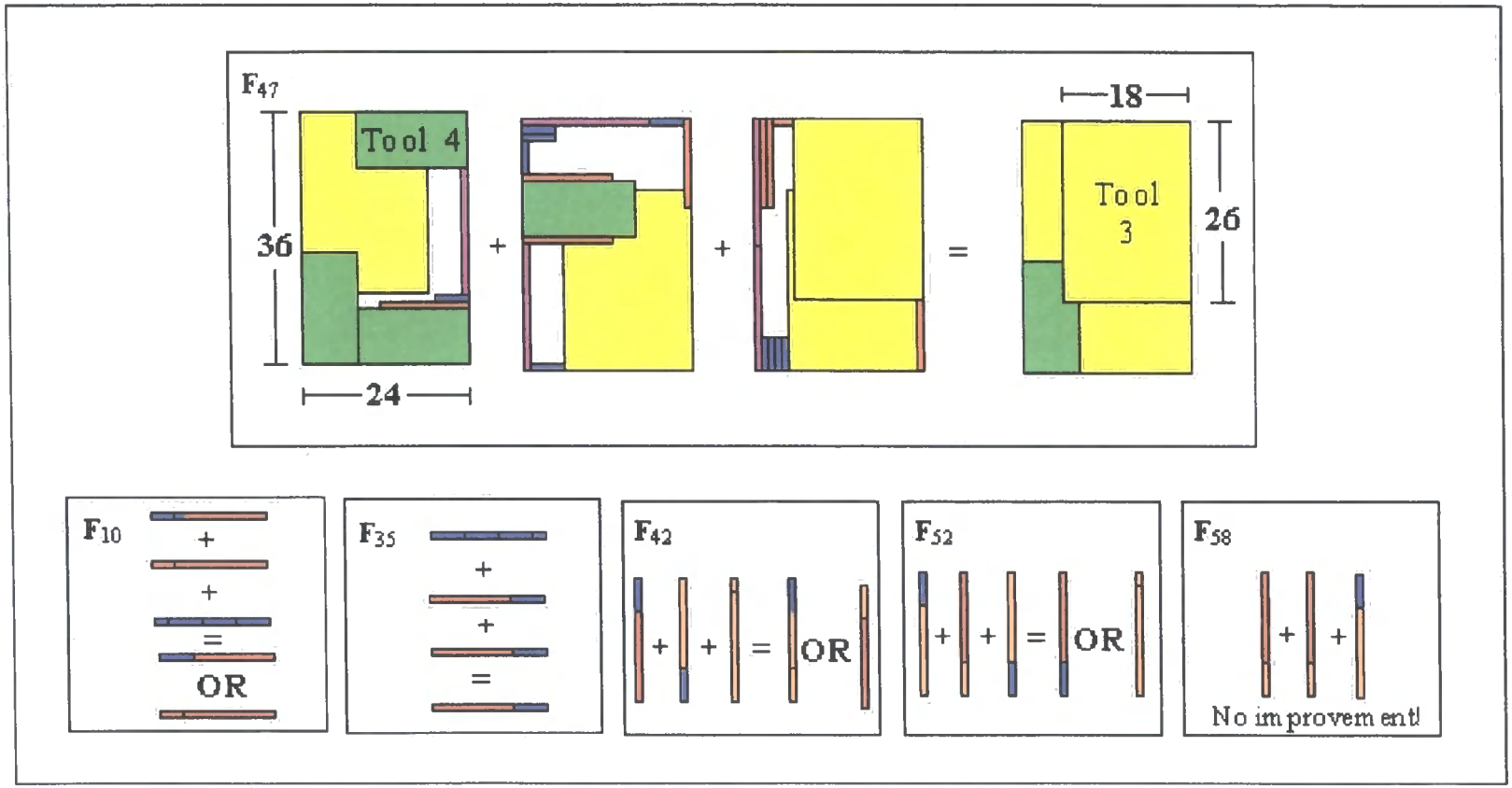


Figure 7.19. Industrial Case Study # 4 – Gene Manipulation.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK

8.1. CONCLUSIONS

1. Tool selection is a difficult and critical problem in the sheet metal industry and has a clear impact upon the cost and quality of a product.
2. In previously published work, tool selection has been considered at two levels. The design level which selects the best tool for a particular geometry and the planning level which selects tool for a particular tool magazine capacity or sequence.
3. For an application such as punching tool selection a cost function provides a flexible and useful measure of performance of a particular tool selection. The terms in the cost function can be adapted to suit particular applications.
4. Tool selection involves the searching of an explosive decision tree. Attempts by earlier researchers have "restricted" or focussed the problem by dealing with particular features. The disadvantage of this reductionist approach is that solutions may not be global optimisations.
5. Any analytical approach is likely to be restricted by the necessary simplifying assumptions. Thus, a Monte Carlo approach was investigated as an "assumption free" method.
6. The Monte Carlo approach described and demonstrated here has been found to be feasible for a range of case studies.

7. The Monte Carlo approach used here could have further application potential in problem solving for the searching of explosive decision trees and if the solutions can be simulated and assessed, as in the case of tool selection.
8. For tool selection, approximating the output distribution of a selection process which involves the use of randomness and assumes a normal distribution, is a good base for the calculations required to estimate the target cost function value and also to estimate the number of simulation runs required to achieve the target value.
9. Working out the target cost function value and the required number of simulation runs using a sample size of 100 simulation runs is statistically sound. The Mean and Standard Deviation of the generated sample would stabilise before reaching the 100^{th} sample and, thus, approaches the population mean and standard deviation of the whole population.
10. Targeting a cost function value where 99.98% of the area under the normal curve will fall above will involve a large and unnecessary number of simulation runs. However, the cost function value at this point gives an indication of a minimum benchmark target.
11. Targeting a cost function value where 99% of the area under the normal curve will fall above, involves a reasonable number of simulation runs and guarantees an optimum or near to optimum cost function value.
12. The shape and the pattern of the distribution of the generated cost function values depends on the particular tools and feature populations. Where there are dominant tools and features the distribution starts to deviate from a normal distribution. This might cause some deficiency in the computed target values based upon the normal distribution assumption.
13. A solution to the tool and feature dominance problem might be to carry out a pre-selection before starting the Monte Carlo methods. Here, the "one-to-one" tool

feature matches are removed from the tool feature population. This would result in a "cleaner" normal distribution and would show as a single peak in the distribution curve.

14. An alternative solution to give a more efficient search and to deal with the tool and feature dominance problem could be to use a combination of the MC method with simulated annealing and/or genetic algorithms.

8.2. FURTHER WORK

Clearly, global or extensive searches for an optimum or near optimum tool selection can be expensive in computation time. The suggestion is that some pre-selection of "one to one" matches between tools and features could reduce computation time considerably. However, such pre-selection could lead to operational inefficiency by requiring excessive tool changing. A partial pre-selection process obviously stimulates the "which part" question. Further research is obviously needed but this work suggests that a Monte Carlo-Rule based hybrid or a combination of MC with simulated annealing and/or genetic algorithms would be worthy of further consideration and research.

The constraints of time have made it necessary to restrict the present work to tools and features having straight edges. Clearly it would be necessary to extend the approach to include circular edges in the future. Although the inclusion of circular edges sets some geometric problems, the concept of the approach would not require significant change.

References

[1]. Joshi, S. and Chang, T-C., Feature extraction and feature based design approaches in the development of design interface for processing planning, Journal of intelligent manufacturing, 1990, vol. 1, pp. 1-15.

[2]. Halevi, G. and Weill, R. D., Principles of process planning, Chapman & Hall, UK, 1995.

[3]. Chang, T-C., Expert Process Planning for Manufacture, Addison-Wesley Publishing Company, USA, 1990.

[4]. Summad, E. and Appleton, E., The Use of the Monte Carlo Method for Tool Selection in the Sheet Metal Industry, Proceedings of the seventh international FAIM conference, June 25-27, UK, 1997, pp. 365-374.

[5]. Summad, E. and Appleton, E., Generic algorithm for sheet metal working tool selection, using curve matching and tree searching, Journal of Materials Processing Technology, 1998, vol. 80-81, pp. 501-506.

[6]. Summad, E. and Appleton, E., Applications of the Monte Carlo Simulation Method in Tool Selection for Punching Operations in the Sheet Metal Industry, IEE - International Conference on Simulation, England, 30 September – 2 October 1998, pp. 277-283.

[7] Appleton, E. and Summad, E., Tool selection for sheet metal punching operations - searching techniques, International NAISO Congress on Information Science Innovations [ISI'200], March 17-21, 2001 at the American University of Dubai, U.A.E.

[8]. Pahl, G. and Beitz, Engineering Design, Spring-Verlag, 1996.

[9]. Cutkosky, M. R. and Tenenmaum, J. M., A methodology and computational framework for concurrent product and process design, Mech. Mach. Theory, vol. 25, no. 3, 1990, pp. 365-381.

[10]. Jo, H. H. et al, Concurrent engineering: The manufacturing philosophy for the 90's, Computers and industrial engineering, vol. 21, nos. 1-4, 1991, pp. 35039.

[11]. Finger, S. et al, Concurrent design, Applied artificial intelligence, vol.6, no.3, 1992, pp. 257-283.

[12]. Vujosevic, R. and Kusiak, A., Data base requirements for concurrent design systems, Engineering databases: An engineering resource, ASME 1991, pp ?.

[13]. Vaiyapuri, V. and Okogbaa, O. G., An integrated design approach for feature based manufacturing using concurrent engineering, 2nd Industrial engineering research conference proceedings, Institute of industrial engineers, May 26-27, 1993, pp. 16-20.

[14]. Young, R. E. et al, An artificial intelligence-based constraint network system for concurrent engineering, International journal of production research, vol. 30, no. 7, 1992, pp. 1715-1735.

[15]. Subramanyam, S. and Lu, S. C.-Y., The impact of an AI-based design environment for simultaneous engineering on process planning, International journal of computer integrated manufacturing, vol. 4, no. 2, 1991, pp. 71-82.

[16]. Appleton, E. and Garside, J., A Team Based Design for Assembly Methodology, Assembly Automation, The international journal of assembly technology and management, Vol. 20, no. 2, May 2000, pp 162-169.

[17]. Finger, S. and Dixon, J. R., A review of research in mechanical engineering design. Part II: Representation, analysis, and design for the life cycle, Research in engineering design, vol. 1, 1989, pp. 121-137.

[18]. Bralla, J. G., Design for manufacturability, 1998, McGraw Hill, USA.

[19]. Appleton, E. and Garside, J., A Team-Based Design for Manufacture Methodology. To appear in the proceedings of ASME 2000 International Design Engineering Technical Conferences and the Computers and Information in Engineering

Conference, September 10-13, 2000, Omni Inner Harbor Hotel, 101 Fayette Street, Baltimore, Maryland.

[20]. Sycara, K. P., Cooperative negotiation in concurrent engineering design, Lecture notes in computer science, 1991, vol.492, pp. 267-297.

[21]. Huang, G. Q. et al, Agents for co-operating expert systems in concurrent engineering design, AI EDAM: Artificial Intelligence for Engineering Design, Analysis and manufacturing, vol. 7, no. 3, 1993, pp. 145-158.

22 Wallace, A. and Boldyreff, C., Agents and agent based design approaches to engineering design and manufacturing, Proceedings of the 15th international conference on computer-aided production engineering, CAPE.99, pp. 345-351.

[23]. Akao, Y., Quality function deployment - Integrating customer requirements into product design, Productivity press, USA, 1988.

[24]. Wong, J. P., An integrated cost estimating system for concurrent engineering environment, Computers and industrial engineering, vol. 21, nos. 1-4, 1991, pp. 589-594.

[25]. Yu, J-C. et al, Computer-aided design for manufacturing process selection, Journal of intelligent manufacturing, vol. 4, 1993, pp. 199-208.

[26]. Nnaji, B. O. et al, Feature reasoning for sheet metal components, International journal of production research, vol. 29, no. 9, 1991, pp. 1867-1896.

[27]. Gindy, N. N. Z. et al, Feature-based component model for computer-aided process planning systems, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 20-26.

[28]. Case, K. and Gao, J., Feature technology: an overview, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 2-12.

[29]. Allada, V. and Anand, S., Feature-based modelling approaches for integrated manufacturing: state-of-the-art survey and future research directions, International journal of computer integrated manufacturing, vol. 8, no. 6, 1995, pp. 411-440.

[30]. Feru, F. et al, Feature-based modeling: state of the art and evolution, Manufacturing in the era of concurrent engineering, Editors: Halevi, G. and Weill, R. D., Elsevier sciences publishers, 1992, pp. 29-50.

[31]. Salomons, O. W. et al, Review of research in feature-based design, Journal of manufacturing systems, vol. 12, no. 2, 1993, pp. 113-132.

[32]. Lenau, T. and Mu, L., Features in integrated modelling of products and their production, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 65-73.

[33]. Sing, N. and Qi, D., A structural framework for part feature recognition: A link between computer-aided design and process planning, Integrated manufacturing systems, vol. 3, no. 1, 1992, pp. 4-12.

[34]. Rooney, J., and Steadman, P., Principles of computer-aided design, 1987, Pitman publishing, UK.

[35]. Bronsvoort, W. F. and Jansen, F. W., Feature modelling and conversion - Key concepts to concurrent engineering, Computers in industry, vol. 21, 1993, pp. 61-86.

[36]. Pratt, M. J, Applications of feature recognition in the product life-cycle, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 13-19.

[37]. Fu et al, A graph grammar approach to feature representation and transformation, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 137-151.

[38]. Young, R. I. M. and Bell, R., Design by features: advantages and limitations in machine planning integration, International journal of computer integrated manufacturing, vol. 6, nos. 1 & 2, 1993, pp. 105-112.

[39]. Smith, J. S. et al, Process plan generation for sheet metal parts using an integrated feature-based expert system approach, International journal of production research, vol. 30, no. 5, 1992, pp. 1175-1190.

[40]. Molengraaf, J. C. M. V. D. et al, Selection procedures for manufacturing processes for design engineers, Robotics and computer-integrated manufacturing, vol. 10, no. 1/2, 1993, pp. 57-64.

[41]. Narayanan, V., Intelligent process routing, IEEE, 1992, pp. 270-277.

[42]. Alting, L. Zhang, H., Computer aided process planning: the state-of-the-art survey, International journal of production research, vol. 27, no. 4, 1989, pp. 553-585.

[43]. Mielnik, E. M., Metal working science and engineering, McGraw Hill, USA, 1991.

[44]. Niebel, B. W. et al, Modern manufacturing process engineering, McGraw Hill, USA, 1990.

[45]. Chiles, V. et al, Principles of engineering manufacturing, 3rd ed., Arnold, 1996.

[46]. Tang, C. S. and Denardo, E. C., Models arising from a flexible manufacturing machine, part I: Minimzation of the number of tool switches, Operations research, vol. 36, no. 5, 1988, pp. 767-777.

[47]. Tang, C. S. and Denardo, E. V., Models arising from a flexible manufacturing machine, part II: Minimzation of the number of switching instants, Operations research, vol. 36, no. 5, 1988, pp. 778-784.

[48]. Daskin, M. et al, Rationalizing tool selection in a flexible manufacturing system for sheet-metal products, Operations research, vol. 38, no. 6, 1990, pp. 1104-1115.

[49]. Hsu, V. N. et al, Tool selection for optimal part production: a Lagrangian relaxation approach, IEE Transactions, vol. 27, 1995, pp. 417-426.

[50]. Ehrismann, R. and Reissner, J., Intelligent manufacturing of laser cutting, punching and bending parts, Robotics and computer-integrated manufacturing, vol. 4, no. 3/4, 1988, pp. 511-515.

[51]. Raggenbass, A. and Reissner, J., Stamping - Laser combination in sheet processing, Annals of the CIRP, vol. 38, no. 1, 1989, pp. 291-294.

[52]. Raggenbass, A. and Reissner, J., Automatic generation of NC production plans in stamping and laser cutting, Annals of the CIRP, vol. 40, no. 1, 1991, pp. 247-250.

[53]. Raggenbass, A. and Reissner, J., An expert system as a link between computer aided design and combined stamping-laser manufacture, Proceedings of the institution of mechanical engineers, vol. 205, 1991, pp. 25-34.

[54]. Liebers, A. et al, Tool selection and tool path calculation for nibbling, Internal report, Laboratory of production and design engineering, University of Twente, P. O. Box 217, 7500 AE Enschede, the Netherlands.

[55]. Loh, H. T. et al, A PC-based software package for automated punching-selection and tool-path generation for a CNC nibbling machine, Journal of materials processing technology, vol. 23, 1990, pp. 107-119.

[56]. Choong, N. F. et al, The implementation of an automatic tool selection system for CNC nibbling, Computers in industry, vol. 23, 1993, pp. 205-222.

[57]. Tilley, S., Integration of CAD/CAM and production control for sheet metal components manufacturing, Annals of the CIRP, vol. 41, no. 1, 1992, pp. 177-180.

[58]. Cho, K. H. and Lee, K., Automatic tool selection for NC turret operation in sheet metal stamping, Journal of engineering for industry, vol. 116, pp. 239-246.

[59]. Haralick, R. M., and Shapiro, L. G., Computer and robot vision, Vol. 2, 1993, Addison-Wesley, USA.

[60]. Ansari, N. and Delp, E. J., Partial shape recognition: A landmark-based approach, IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 5, 1990, pp. 470-483.

[61]. Schmidt, W. A. C., Modified matched filter for cloud clutter suppression, IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 6, 1990, pp. 594-600.

[62]. Wolfson, H. J., On curve matching, IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 5, 1990, pp. 483-489.

[63]. Gorman, J. W. et al, Partial shape recognition using dynamic programming, IEEE transactions on pattern analysis and machine intelligence, vol. 10, no. 2, 1988, pp. 257-266.

[64]. Schwartz, J. T., and Sharir, M., Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves, The international journal of robotics research, vol. 6, no, 2, 1987, pp. 29-44.

[65]. Dubois, S. R., Glanz, F. H., An autoregressive model approach to two-dimensional shape classification, IEEE transactions on pattern analysis and machine intelligence, vol. PAMI-8, no. 1, 1986, pp. 55- 66.

[66]. Kalvin, A. et al, Two-dimensional, model-based, boundary matching using footprints, The international journal of robotics research, vol. 5, no. 4, 1986, pp. 38-55.

[67]. Ayache, N., and Faugeras, O. D., Hyper: A new approach for the recognition and positioning of two-dimensional objects, IEEE transactions of pattern analysis and machine intelligence, vol. PAMI-8, no. 1, 1986, pp. 44-54.

[68]. Hong, J. and Wolfson, H. J., An improved model-based matching method using footprints, IEEE, 1988, pp. 72-78.

[69]. Bolles, R. C., Cain, R. A., Recognizing and locating partially visible objects: The local-feature-focus method, The international journal of robotics research, vol. 1, no. 3, 1982, pp. 57-82.

[70]. Ballard, D. H., and Brown, C. M., Computer vision, Prentice-Hall, USA, 1982.

[71]. Parker, J. R., Algorithms for image processing and computer vision, John Wiley and Sons, USA, 1997.

[72]. Rosenfeld, A., Image analysis and computer vision: 1993, CVGIP: Image understanding, 1994, vol. 59, no. 3, pp. 367-404.

[73]. Lin, C. C., and Chellappa, R. Classification of partial 2-D shapes using Fourier descriptors, IEEE transactions on pattern analysis and machine intelligence, vol. PAMI-9, no. 5, 1987, pp. 686-690.

[74]. Persoon, E., and Fu, K-S., Shape discrimination using Fourier descriptors, IEEE transactions on systems, man, and cybernetics, 1977, pp. 170-179.

[75]. Zahn, C. T. and Roskies, R. Z., Fourier descriptors for plane closed curves, IEEE transactions on computers, vol. C-21, no. 3, 1972, pp. 269-281.

[76]. Aho, A. V. et al, Data structures and algorithms, Addison-Wesley, USA, 1983.

[77]. Kruse, R. L., Data structures and program design, Prentice-Hall, USA, 1984.

[78]. Tremblay, J. P., and Sorenson, P. G., An introduction to data structures with applications, McGraw-Hill. USA, 1976.

[79]. Rao, S. S., Engineering optimization: Theory and practice, 3rd edition, John Wiley & Sons, Inc., USA, 1996.

[80]. Taylor, M. G., Intelligent Improvement of Repetitive Robot Assembly Tasks, 1992, PhD Thesis, Engineering Department, Cambridge University, UK.

[81]. Novik, N., Blind Search Algorithms, http://www.netspace.org/~shalott/searchapplet/, 1998.

[82]. Breadth-First Search, http://cindy.cis.nctu.edu.tw/AI/ai1/4-bfs.html, Computer and Information Science Institute, National Chiao Tung University, 1999.

[83]. Johnson , K., State Space Analysis, http://www.sahs.uth.tmc.edu/kajohnson/HI6350/04-search/sld007.htm, University of Texas at Houston, 1999.

[84]. Breadth-First Search, http://cindy.cis.nctu.edu.tw/AI/ai1/4-dfs.html, Computer and Information Science Institute, National Chiao Tung University, 1999.

[85]. Search, http://www.ai.usma.edu:8080/overview/SEARCH.HTM, Office of artificial intelligence analysis and Evaluation at the United States Military Academy, West Point, Department of Electrical Engineering and Computer Science, 1996.

[86]. Doyle, P., Search Methods, http://www-cs-students.stanford.edu/~pdoyle/quail/notes/pdoyle/search.html#Iterative Deepening, Computer science department, Stanford University, 1997.

[87]. Finin, T., Informed Search [Best-First Search], http://www.csee.umbc.edu/471/lectures/informed-search/sld006.htm, UNIVERSITY OF MARYLAND, BALTIMORE, 1998.

[88].Doyle, P., Search Methods, http://www-cs-students.stanford.edu/~pdoyle/quail/notes/pdoyle/search.html#Best-First Search, Computer science department, Stanford University, 1997.

[89]. Best-First Search, http://cogsci.ucsd.edu/~batali/108b/lectures/heuristic.html#best, Cognitive Science 108b Lecture Notes, University of California at San Diego, 1999.

[90]. Breadth-First Search, http://cindy.cis.nctu.edu.tw/AI/ai1/4-beam.html, Computer and Information Science Institute, National Chiao Tung University, 1999.

[91]. Heuristic Search, http://cogsci.ucsd.edu/~batali/108b/lectures/heuristic.html, Cognitive Science 108b Lecture Notes, University of California at San Diego, 1999.

[92]. Cawsey, A., Heuristic Search, http://www.cee.hw.ac.uk/~alison/ai3notes/subsubsection2_6_2_3_1.html, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, 1994.

[93]. Curtiss, J. H., Monte Carlo Method, U.S. government printing office, Washington, 1951.

[94]. Jacoboni, C., and Lugli, P., The Monte Carlo method for semiconductor device simulation, Springer-Verlag Wein New York, 1989.

[95]. Hammond, B. L. et al, Monte Carlo methods in Ab Initio Quantum chemistry, 1994, World Scientific Publishing Co. Pte. Ltd, Singapore.

[96]. Berthier, G., Lecture Notes in Chemistry, Springer-Verlag, Berlin, 1981.

[97]. Ripley, B. D., Stochastic simulation, John Wiley & Sons, Inc., 1987.

[98]. Rubinstein, R. Y., Simulation and the Monte Carlo method, 1981, John Wiley and Sons, USA.

[99]. Rubinstein, R. Y., Monte Carlo optimization, simulation and sensitivity of queuing networks, 1986, John Wiley and Sons, USA.

[100]. Computational Science Education Project [Electronic book], Sponsored by U.S. Department of Energy, 1996, http://csep1.phy.ornl.gov/mc/node2.html.

[101]. Smith, V. K., Monte Carlo methods their role for econometrics, 1973, D. C. Heath and company, USA.

[102]. Introduction to Monte Carlo Methods, , Department of Energy, http://csep1.phy.ornl.gov/mc/mc.html, 1995.

[103]. Woller, J., The basics of Monte Carlo simulations, http://wwitch.unl.edu/zeng/joy/mclab/mcintro.html, University of Nebraska-Lincoln, 1996

[104]. Monte Carlo Simulation, http://www.contingencyanalysis.com/glossarymontecarlosimulation.htm, Contingency Analysis, 1998.

[105]. Williams, B., A sampler on sampling, John Wiley and Sons, USA, 1978.

[106]. Pagano, R. R., Understanding statistics in the behavioral sciences, 2 ed., West publishing company, USA, 1986.

[107]. Moore, D. S., and McCabe, G. P., Introduction to the practice of statistics, 3 ed., W. H. Freeman and company, USA, 1999.

[108]. de Vaus, D. A., Surveys in Social Research, 3rd ed., UCL Press, London, 1994.

[109]. Jordan, D. W., and Smith, P., Mathematical techniques : an introduction for the engineering, physical, and mathematical sciences, 2nd ed., Oxford : Oxford University Press, 1997.

[110].Combinatorial and Real Function Optimisation, http://iridia.ulb.ac.be/Projects/combi.html, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Belgium, 1996.

[111]. Some Search Algorithms, http://yoda.cis.temple.edu:8080/UGAIWWW/lectures95/search/more-search.html#4, Computer and Information Sciences Department in the College of Science and Technology of Temple University, Philadelphia, Pennsylvania, USA

[112]. Genetic Algorithms vs. Traditional Methods, http://http1.brunel.ac.uk:8080/departments/AI/alife/ga-versu.htm, Artificial Intelligence Site, Brunel University, 1998.

[113]. Black, P. E., Simulated annealing, http://hissa.ncsl.nist.gov/~black/DADS/HTML/simulatdannl.html, Software Quality Group, 2000.

The angle θ is to be measured from the positive x-axis, in the anticlockwise direction if θ is positive, and in a clockwise direction if θ is negative.

The angle between two nonzero vectors a and b is defined to be the angle θ, where $0 \le \theta \le \pi$., formed by the corresponding directed line segments whose initial points are the origin.

The angle θ os obtained from

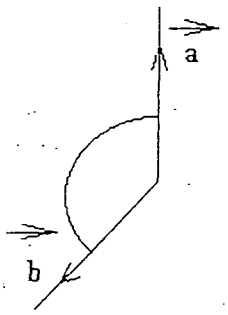
$a \cdot b = \|a\| \ \|b\| \cos\theta$

The condition that nonzero vectors a and b be perpendicular is that $a \cdot b = 0$.

$$\|a\| = \sqrt{\left((x_{a_1} - x_{a_0})^2 + (y_{a_1} - y_{a_0})^2\right)}$$

$$\|b\| = \sqrt{\left((x_{b_1} - x_{b_0})^2 + (y_{b_1} - y_{b_0})^2\right)}$$

$$a \cdot b = (x_{a_1} - x_{a_0})(x_{b_1} - x_{b_0}) + (y_{a_1} - y_{a_0})(y_{b_1} - y_{b_0})$$



Also,

$$\|axb\| = \|a\| \ \|b\| \ \sin\theta \qquad (0 \le \theta \le \pi)$$

$$axb = (x_{a_1} - x_{a_0})(y_{b_1} - y_{b_0}) - (y_{a_1} - y_{a_0})(x_{b_1} - x_{b_0})$$

Problem: Visual basic, used software, does not support $\cos^{-1}$ function but only supports $\tan^{-1}$.

Solution: The value of sin i.e. positive or negative is known.
The value of cos i.e. positive or negative is known.
Calculate the angle of absolut (sin/cos)
Work out which quadrant the angle is, hence the tue value can be computed.

Sin +ve	**All** +ve
Tan +ve	**Cos** +ve

Reference:

Jordan, D.W. and Smith, P., Mathematical techniques: an introduction for the engineering, physical, and mathematical sciences, Oxford University Press, 1994.

S. No.	Tool No.	Type	Size
1	TN104.00	Square	4.000 x 4.000
2	TN105.50	Square	5.500 x 5.500
3	TN106.50	Square	6.500 x 6.500
4	TN108.00	Square	8.000 x 8.000
5	TN109.00	Square	9.000 x 9.000
6	TN109.50	Square	9.500 x 9.500
7	TN110.00	Square	10.000 x 10.000
8	TN110.50	Square	10.500 x 10.500
9	TN111.00	Square	11.000 x 11.000
10	TN111.60	Square	11.600 x 11.600
11	TN112.00	Square	12.000 x 12.000
12	TN112.70	Square	12.700 x 12.700
13	TN114.00	Square	14.000 x 14.000
14	TN116.00	Square	16.000 x 16.000
15	TN116.30	Square	16.300 x 16.300
16	TN119.05	Square	19.050 x 19.050
17	TN120.00	Square	20.000 x 20.000
18	TN122.00	Square	22.000 x 22.000
19	TN122.23	Square	22.230 x 22.230
20	TN126.00	Square	26.000 x 26.000
21	TN140.00	Square	40.000 x 40.000
22	TN150.00	Square	50.000 x 50.000
23	TN152.00	Square	52.000 x 52.000
24	TN156.00	Square	56.000 x 56.000
25	TN160.00	Square	60.000 x 60.000
26	TN204.02	Rectangular	4.000 x 2.000
27	TN204.03	Rectangular	4.400 x 3.000
28	TN204.16	Rectangular	4.000 x 1.600
29	TN204.30	Rectangular	4.200 x 3.500
30	TN205.02	Rectangular	5.300 x 2.700
31	TN205.03	Rectangular	5.800 x 3.180
32	TN205.05	Rectangular	5.800 x 5.500
33	TN205.10	Rectangular	5.600 x 1.200
34	TN205.30	Rectangular	5.000 x 3.000
35	TN206.03	Rectangular	6.000 x 3.000
36	TN206.04	Rectangular	6.000 x 4.200
37	TN207.02	Rectangular	7.500 x 2.700
38	TN207.03	Rectangular	7.000 x 3.000
39	TN207.05	Rectangular	7.000 x 5.500
40	TN208.01	Rectangular	8.000 x 1.000
41	TN208.02	Rectangular	8.000 x 2.500
42	TN208.03	Rectangular	8.000 x 3.000
43	TN208.05	Rectangular	8.500 x 5.000
44	TN208.53	Rectangular	8.000 x 5.300
45	TN209.03	Rectangular	9.500 x 3.500
46	TN209.04	Rectangular	9.320 x 4.800
47	TN209.05	Rectangular	9.500 x 5.800
48	TN209.34	Rectangular	9.500 x 3.400
49	TN209.48	Rectangular	9.300 x 4.800
50	TN210.02	Rectangular	10.000 x 2.000
51	TN210.03	Rectangular	10.000 x 3.300
52	TN210.04	Rectangular	10.000 x 4.000
53	TN210.05	Rectangular	10.000 x 5.000
54	TN210.09	Rectangular	10.000 x 9.001
55	TN211.04	Rectangular	11.000 x 4.700
56	TN212.02	Rectangular	12.500 x 1.000
57	TN212.06	Rectangular	12.500 x 6.500
58	TN215.03	Rectangular	15.000 x 3.000
59	TN215.05	Rectangular	15.000 x 5.000
60	TN215.06	Rectangular	15.000 x 6.000
61	TN215.13	Rectangular	15.000 x 13.000
62	TN216.06	Rectangular	16.000 x 6.000
63	TN217.02	Rectangular	17.000 x 2.500
64	TN217.04	Rectangular	17.000 x 4.000
65	TN217.10	Rectangular	17.500 x 10.500
66	TN218.02	Rectangular	18.000 x 2.000
67	TN218.07	Rectangular	18.000 x 7.000
68	TN218.10	Rectangular	18.000 x 10.100
69	TN220.01	Rectangular	20.800 x 1.250
70	TN220.05	Rectangular	20.000 x 5.000
71	TN220.10	Rectangular	20.000 x 10.000
72	TN220.11	Rectangular	20.000 x 11.000
73	TN220.14	Rectangular	20.000 x 14.000
74	TN221.06	Rectangular	21.000 x 6.500
75	TN221.15	Rectangular	21.500 x 15.500
76	TN222.04	Rectangular	22.000 x 4.000
77	TN222.10	Rectangular	22.000 x 10.000
78	TN223.10	Rectangular	23.000 x 10.000
79	TN223.11	Rectangular	23.000 x 11.900
80	TN224.02	Rectangular	24.000 x 2.000
81	TN224.05	Rectangular	24.000 x 5.000
82	TN224.06	Rectangular	24.000 x 6.000
83	TN225.05	Rectangular	25.000 x 5.000
84	TN225.06	Rectangular	25.500 x 6.350
85	TN225.07	Rectangular	25.000 x 7.500
86	TN225.10	Rectangular	25.000 x 10.000
87	TN225.12	Rectangular	25.400 x 12.400
88	TN225.13	Rectangular	25.500 x 13.300
89	TN225.15	Rectangular	25.000 x 15.500
90	TN225.60	Rectangular	25.200 x 6.060
91	TN226.65	Rectangular	26.000 x 6.500
92	TN229.05	Rectangular	29.000 x 5.000
93	TN230.03	Rectangular	30.000 x 3.000
94	TN230.05	Rectangular	30.000 x 5.000
95	TN230.06	Rectangular	30.000 x 6.000
96	TN230.10	Rectangular	30.000 x 10.000
97	TN230.12	Rectangular	30.000 x 12.000
98	TN230.14	Rectangular	30.500 x 14.000
99	TN232.13	Rectangular	32.000 x 13.000
100	TN233.07	Rectangular	33.000 x 7.500
101	TN233.09	Rectangular	33.000 x 9.500
102	TN236.17	Rectangular	36.700 x 17.200
103	TN236.30	Rectangular	36.000 x 30.000
104	TN240.05	Rectangular	40.000 x 5.000
105	TN240.10	Rectangular	40.000 x 10.000
106	TN241.16	Rectangular	41.000 x 16.000
107	TN242.14	Rectangular	42.500 x 14.000
108	TN247.03	Rectangular	47.000 x 3.000
109	TN250.05	Rectangular	50.000 x 5.000
110	TN250.06	Rectangular	50.000 x 6.000
111	TN250.10	Rectangular	50.000 x 10.500
112	TN250.12	Rectangular	50.800 x 12.500
113	TN250.19	Rectangular	50.000 x 12.000
114	TN260.50	Rectangular	60.000 x 50.000
115	TN267.06	Rectangular	67.000 x 6.000
116	TN268.07	Rectangular	68.000 x 7.000

S. No.	Tool No.	Type	Size
117	TN270.06	Rectangular	70.000 x 6.000
118	TN270.18	Rectangular	70.000 x 18.000
119	TN271.22	Rectangular	71.000 x 22.500
120	TN272.52	Rectangular	7.000 x 5.000
121	TN275.07	Rectangular	75.000 x 7.000
122	TN275.30	Rectangular	75.000 x 30.000
123	TN279.04	Rectangular	79.600 x 4.600
124	TN280.05	Rectangular	80.000 x 5.000
125	TN285.05	Rectangular	85.000 x 5.000
126	TN304.03	Rectangular	4.470 x 3.200
127	TN304.30	Rectangular	4.500 x 3.500
128	TN306.04	Rectangular	6.500 x 4.500
129	TN306.05	Rectangular	6.500 x 5.000
130	TN306.50	Rectangular	6.500 x 5.000
131	TN307.04	Rectangular	7.500 x 4.500
132	TN307.05	Rectangular	7.200 x 5.500
133	TN308.03	Rectangular	8.000 x 3.500
134	TN308.04	Rectangular	8.000 x 4.000
135	TN310.05	Rectangular	10.000 x 5.000
136	TN310.06	Rectangular	10.300 x 6.300
137	TN310.07	Rectangular	10.000 x 7.500
138	TN311.05	Rectangular	11.000 x 5.500
139	TN311.07	Rectangular	11.000 x 7.000
140	TN312.04	Rectangular	12.680 x 4.500
141	TN312.06	Rectangular	12.000 x 6.000
142	TN312.64	Rectangular	12.800 x 6.400
143	TN314.07	Rectangular	14.000 x 7.000
144	TN315.06	Rectangular	15.500 x 6.500
145	TN315.09	Rectangular	15.880 x 9.500
146	TN316.09	Rectangular	16.000 x 9.000
147	TN317.06	Rectangular	17.000 x 6.000
148	TN319.05	Rectangular	19.700 x 5.000
149	TN319.07	Rectangular	19.050 x 7.500
150	TN320.04	Rectangular	20.000 x 4.000
151	TN320.06	Rectangular	20.000 x 6.000
152	TN320.07	Rectangular	20.000 x 7.000
153	TN320.14	Rectangular	20.000 x 14.000
154	TN321.10	Rectangular	21.000 x 10.000
155	TN322.17	Rectangular	22.000 x 17.000
156	TN324.02	Rectangular	24.000 x 2.000
157	TN325.01	Rectangular	25.000 x 1.500
158	TN326.06	Rectangular	26.500 x 6.500
159	TN330.02	Rectangular	30.000 x 2.500
160	TN330.07	Rectangular	30.000 x 7.500
161	TN330.12	Rectangular	30.000 x 12.000
162	TN330.20	Rectangular	30.000 x 20.000
163	TN330.24	Rectangular	30.000 x 24.000
164	TN330.69	Rectangular	30.000 x 7.000
165	TN330.70	Rectangular	30.000 x 7.300
166	TN345.03	Rectangular	4.500 x 3.00
167	TN350.18	Rectangular	50.000 x 18.000
168	TN353.04	Rectangular	53.600 x 4.600
169	TN353.56	Rectangular	53.600 x 5.600
170	TN357.58	Rectangular	57.800 x 5.800
171	TN358.06	Rectangular	58.800 x 6.800
172	TN375.03	Rectangular	75.000 x 3.000
173	TN402.00	Rectangular	17.000 x 7.000
174	TN416.00	Square	24.000 x 24.000
175	TN419.10	Rectangular	19.000 x 10.000
176	TN433.00	Rectangular	16.600 x 10.000
177	TN441.00	Square	24.000 x 24.000
178	TN442.00	Square	11.000 x 11.000
179	TN465.00	Rectangular	3.000 x 17.000
180	TN487.00	Square	32.000 x 32.000
181	TN498.00	Square	39.000 x 39.000
182	TN508.00	Square	20.800 x 20.800
183	TN519.00	Square	16.000 x 16.000
184	TN520.00	Square	24.000 x 24.000
185	TN581.00	Square	30.000 x 30.000