Optimisation and Decision Support during the Conceptual Stage of Building Design

New techniques based on the genetic algorithm.

Jim David Mathews

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

Doctor of Philosophy

School of Civil and Structural Engineering
Faculty of Technology

February 2000

For

Clive Burrows,
friend and teacher.

OPTIMISATION AND DECISION SUPPORT DURING THE CONCEPTUAL STAGE OF BUILDING DESIGN – NEW TECHNIQUES BASED ON THE GENETIC ALGORITHM by JIM DAVID MATHEWS.

ABSTRACT

Modern building design is complex and involves many different disciplines operating in a fragmented manner. Appropriate computer-based decision support (DS) tools are sought that can raise the level of integration of different activities at the conceptual stage, in order to help create better designs solutions. This project investigates opportunities that exist for using techniques based upon the Genetic Algorithm (GA) to support critical activities of conceptual building design (CBD). Collective independent studies have shown that the GA is a powerful optimisation and exploratory search technique with widespread application. The GA is essentially very simple yet it offers robustness and domain independence. The GA efficiently searches a domain to exploit highly suitable information. It maintains multiple solutions to problems simultaneously and is well suited to non-linear problems and those of a discontinuous nature found in engineering design.

The literature search first examines traditional approaches to supporting conceptual design. Existing GA techniques and applications are discussed which include pioneering studies in the field of detailed structural design. Broader GA studies are also reported which have demonstrated possibilities for investigating geometrical, topological and member size variation. The tasks and goals of conceptual design are studied. A rationale is introduced, aimed at enabling the GA to be applied in a manner that provides the most effective support to the designer. Numerical experiments with floor planning are presented. These studies provide a basic foundation for a subsequent design support system (DSS) capable of generating structural design concepts.

A hierarchical Structured GA (SGA) created by Dasgupta et al [1] is investigated to support the generation of diverse structural design concepts. The SGA supports variation in the size, shape and structural configuration of a building and in the choice of structural frame type and floor system. The benefits and limitations of the SGA approach are discussed. The creation of a prototype DSS system, abritrarily called Designer-Pro (DPRO), is described. A detailed building design model is introduced which is required for design development and appraisal. Simplifications, design rationale and generic component modelling are mentioned. A cost-based single criteria optimisation problem (SCOP) is created in which other constraints are represented as design parameters.

The thesis describes the importance of the object-oriented programming (OOP) paradigm for creating a versatile design model and the need for complementary graphical user interface (GUI) tools to provide human-computer interaction (HCI) capabilities for control and intelligent design manipulation. Techniques that increase flexibility in the generation and appraisal of concept are presented. Tools presented include a convergence plot of design solutions that supports cursor-interrogation to reveal the details of individual concepts. The graph permits study of design progression, or evolution of optimum design solutions. A visualisation tool is also presented.

The DPRO system supports multiple operating modes, including single-design appraisal and enumerative search (ES). Case study examples are provided which demonstrate the applicability of the DPRO system to a range of different design scenarios. The DPRO system performs well in all tests. A parametric study demonstrates the potential of the system for DS. Limitations of the current approach and opportunities to broaden the study form part of the scope for further work. Some suggestions for further study are made, based upon newly-emerging techniques.

TABLE OF CONTENTS

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BoQ	Bill of Quantities
BR	Building Regulations
BRE	Building Research Establishment
BSC	British Steel Corporation
C	*computer language*
C++	*computer language*
CAd	Computer Aided Draughting
CAD	Computer Aided Design
CBD	Conceptual Building Design
CBR	Case-Based Reasoning
CICA	Construction Industry Computing Association
CIOB	Chartered Institute of Builders
CIRIA	Construction Industry Research & Information Association
COP	Code of Practice
CSD	Composite Steel Deck floor
D&B	Design-and-Build contract
DS	Decision Support
DSS	Decision Support System
EAS	Evolutionary & Adaptive Search
ES	Exhaustive (or, Enumerative) Search
FEA	Finite Element Analysis
FORTRAN	Formula Translation *computer language*
GA	Genetic Algorithm
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HSE	Health and Safety Executive
IBDS	Integrated Building Design System
ICE	Institution of Civil Engineers
IDSS	Intelligent Decision Support System
IMS	Intelligent Machine System
IS	In situ (concrete)
ISDS	Integrated Structural Design System

IStructE	Institution of Structural Engineers
IT	Information Technology
KBES	Knowledge-Based Expert System
KBDS	Knowledge-Based Design System
LISP	List Processing *computer language*
LWC	Light-Weight Concrete
MS	Microsoft Corporation
NWC	Normal-Weight Concrete
OOP	Object-Oriented Programming
OS	Operating System
RIBA	Royal Institute of British Architects
PC	Personal Computer
PCC	Precast Concrete
PEDC	Plymouth Engineering Design Centre
PROLOG	Programming in Logic *computer language*
PSC	Prestressed Concrete
RC	Reinforced Concrete
RCC	Reinforced Concrete Council
RIBA	Royal Institution of British Architects
RSS	Remainder Stochastic Sampling selection method
SCI	Steel Construction Institute
SGA	Structured Genetic Algorithm
UB	Universal Beam
UC	Universal Column
WRW	Weighted Roulette Wheel selection method

List of Figures and Tables

Authors declaration

The work described in this thesis was carried out in the School of Civil and Structural Engineering at the University of Plymouth. At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award. All the material described herein is the original work of the author unless otherwise acknowledged.

Signed........................ Date.19/05/2000

Training

The following modules from the University of Plymouth B.Sc. Computer Science degree course were attended during this research: -

"Event Driven Programming", **Mr. Peter Melhuish**
"AI Programming" and "Machine Learning", **Dr. Charles Ellis**

The following modules from the University of Plymouth M.Sc. Intelligent Systems degree course were attended during this research: -

"Congnitive Science", **Dr. Kenneth Coventry**
"Knowledge Engineering" and "AI Machine Learning", **Dr. Emmannuel Ifeachor**

The author's previous employment as a Software Engineer has also been beneficial.

Events Attented

Nov 1993	"Interfacing of Draughting and Design Technology", presentation by Mott MacDonald IT Division, IStructE, London.
Sept 1994	"First International Conference on Adaptive Computing for Engineering Design and Control", University of Plymouth.
Oct 1994	"Optimising Floor Slab Options for Suspended Floor Construction", Continuing Professional Development Course, IStructE, London.
Jun 1995	"Knowledge-Based Approaches to Automation in Construction", Colloquium, IEEE, London.
Aug 1995	"Sixth International Conference on the Application of Civil and Structural Engineering", University of Cambridge.
Feb 1996	"Design, Manufacture and Application of Precast Concrete Floors", IStructE Devon & Cornwall Branch, School of Engineering, University of Exeter; speaker, **Mr. Edward Fisher**, Richard Lees Ltd.
Mar 1996	"Second International Conference on Adaptive Engineering Design and Control", University of Plymouth.
Jun 1996	"The Construction Industry and the Environmental Challenge" Seminar / Exhibition orgainized by the CIOB, Windsor House, Plymouth.
Sept 1996	"Knowledge-Based Systems – Current Developments and Future Trends", Seminar, IStructE, London.

Published Work

Mathews J.D. & Rafiq M.Y., 1994
"Adaptive Search for Decision Support in the Preliminary Design of Structural Systems",
in ACEDC '94: Proc. 1st Int. Conf. Adapt. Comput. for Eng. Des. Ctrl., Parmee I.C. (ed.).,
pp.169-75.

Mathews J.D. & Rafiq M.Y., 1995
"Adaptive Search to Assist in the Conceptual Design of Concrete Buildings", in AI CIVIL-
COMP '95: Proc. 4th Int. Conf. Applic. AI to Civ. Struct. Engg., Topping B.H.V. (ed.),
Vol. F., pp.179-87.

Mathews J.D., Rafiq M.Y. & Bullock G.N., 1996
"A Prototype for a Conceptual Structural Building Design System Using the Genetic
Algorithm", in ACEDC '96: Proc. 2nd Int. Conf. Adapt. Comput. for Eng. Des. Ctrl.,
Parmee I.C. (ed.), pp.287-90.

Rafiq M.Y. & Mathews J.D., 1996
"An Integrated Approach to Structural Design of Buildings using Genetic Algorithms" in
Proc. 2nd World Congress of Integrated Design and Process Technology, IDPT-Vol.3,
Esat I.I., Veniali, F., Rasty J., Gransburg, D.D. & Ertas, A. (eds.), Dec., pp.84-90.

Mathews J.D. & Rafiq M.Y., 1996
"Integration and Optimisation in Conceptual Building Design Using Genetic Algorithms"
in "Knowledge-Based Systems in Structural Engineering: Current Developments and
Future Trends ", Anumba C.J. (ed.)., IStructE Seminar 11th Sept., pp.63-73.

Rafiq M.Y. & Mathews J.D., 1998
"An Integrated Approach to Structural Design of Buildings Using a Structured Genetic
Algorithm" in J. Intgrtd. Des. & Proc. Tech., Vol. 2., No.3, pp.20-31.

Funding

Acknowledgements

1 Introduction

1.1 Motivation

Building design is a multidisciplinary activity. The successful creation of a building requires close co-operation between many parties, including clients, architects, engineers from various disciplines, quantity surveyors and contractors. Tasks that need to be addressed include the functional design, structural design, construction planning and costing, and the construction itself. As design tasks are inter-related, decisions taken by one discipline normally have significant implications upon the activities of the others. The building industry is often described as being fragmented. A marked lack of integration between the various disciplines contributes to poor quality decision-making, leading to incompatibility and construction difficulties, which might otherwise be avoided. These problems cause delays, increase costs and reduce the quality and performance of the final structure.

Current design practice can be regarded as a convergent process. A broad design concept is developed into a detailed design solution. To raise the quality of building design, integration and collaboration must start early on in the design process, at the conceptual design stage, where critical decisions affecting the future of the project are made. The challenge in achieving greater integration has implications upon attitudes to design and design practice. Lack of time to consider different options before committing to a particular design concept is a highly significant factor.

It has been hoped the electronic computer might improve integration. Many commercial software tools have been developed to assist with detailed design activities. They have been particularly effective for supporting structural analysis and draughting. Unfortunately, whereas detailed design tasks are formalised, conceptual design tasks are not. CBD is a loosely structured activity containing elements of uncertainty. It requires a varying degree of importance to be given to different considerations in different circumstances. Accordingly, software created for the purpose of assisting with CBD, and capable of integrating different activities, needs different capabilities.

Early research work involving computer software for supporting CBD processes was associated primarily with a field of artificial intelligence (AI) called knowledge-based expert systems (KBES). KBESs processed design knowledge symbolically. The knowledge consisted of heuristic facts and rules. Rules were applied to make decisions

1

automatically. Early KBESs operated in a rigid manner. Later systems sought to provide greater flexibility. Notably, research into KBESs did not produce new commercial tools. Limitations upon the practicality, validity and scope for applying inductive techniques are all factors that have encouraged investigation of alternative approaches to provide DS in the CBD domain. The current research describes a new approach based on the GA. The GA is a type of evolutionary / adaptive search (EAS) technique that has been applied with considerable success to a broad range of real problems in the engineering design domain. Indeed, this success has inspired the present study.

2.2 Purpose

The GA permits a novel, systematic approach to conceptual design in which the relaxation of rule-based control is compensated by greater exploration and qualitative evaluation of potential solutions. Techniques that are based upon, or complementary to the GA, are presented and their relevance is discussed. The techniques were specifically intended to be applicable in helping designers assess the suitability of different structural concepts for medium-size office buildings, as well as other types.

2.3 Purpose

This study contributes to collective knowledge of techniques that ultimately seeks to provide practical computer-based support for CBD activities. In the field of building design, GAs permit an original approach to concept development. From the GA perspective, the building domain represents a novel application for studying established and proprietary AI techniques. Numerous alternative and complementary techniques are mentioned, for which research is ongoing and has shown promise. The additional challenges faced in supporting concept development effectively, as compared to those encountered when applying a GA to a detailed design problem are presented. The study aims to demonstrate a broad and flexible approach to CBD. It focuses specifically on architectural, structural and cost engineering aspects of design in order to present fundamental ideas and techniques, because of time constraints.

Building design support has been studied widely. This work is guided by past and present research in associated fields. A wealth of literature exists, too extensive to describe completely. Reference is made to other research with particular significance to the current project or illustrative of the diversity of techniques that are currently being investigated.

1.4 Order of Contents

The thesis begins with background information. It was convenient to divide background literature into two chapters. Chapter 2 provides an overview of CBD at the broadest level. It introduces the nature of the building design process, and then proceeds by introducing aspects related to software for conceptual design support and related research efforts. Non-GA approaches, mainly based upon KBESs, are described in this chapter. In chapter 3, the GA is introduced and its past application in the structural design domain and conceptual design domain are described. In chapter 4, attention turns to general issues relating to the effective implementation of DS in the field of CBD, and considerations for an evolutionary design approach. Findings from studies that applied the GA to domains other than the CBD domain were considered along with the finding of other studies that applied non-adaptive techniques to support CBD, to help present a rationale for design development. The aims and capabilities for creating appropriate design models are mentioned. Chapter 5 describes preliminary numerical experiments with the GA that demonstrate its potential in floor planning activity. These studied provided guidance in the creation of a broader DSS. Chapter 6 addresses design system modelling, including the representation of design knowledge and suitable genetic structures needed to implement a CBD DSS effectively. Chapter 7 considers how flexibility can be enhanced through appropriate HCI techniques. Software architecture and functionality is described that supports design exploration. Chapter 8 demonstrates capabilites of the system using suitable examples. The applicability of the GA is discussed in relation to the approach adopted and alternative techniques that exist. A summary of the research, conclusions and opportunities for further research are given in chapter 9.

2 Building Design Processes and Software Tools

2.1 Introduction

This chapter addresses the following: -

- the evolution of building design and the design team,
- stages in the design process,
- conceptual design goals and requirements,
- the disparity in the application of IT for detailed design and conceptual design, and
- previous approaches to CBD which have influenced the present study.

2.2 The Nature of Building Design

In ancient times, a single master builder led structural work and coordinated the activities of other workers and craftsmen in person, communicating information mainly by way of speech. Early craftsmen used rules-of-thumb and passed down their knowledge about geometry and proportioning structures from one generation to another. Masonry and timber were common construction materials. In more recent times, the introduction of new, more versatile building materials provided the scope for greater variation in structural form. Rapid growth in design and construction knowledge, including a better understanding of the behavior of materials and other technological advances revolutionized structural engineering and saw new specialist disciplines emerge within building design. By the turn of the 20th Century, structural engineering and other disciplines had adopted a much more scientific approach.

The previous statement, which is based on Rafiq et al [2], introduces matters related to the growth and changes that have taken place within structural building design from ancient times to modern day. Arora [3] asserted that traditionally the best designs have been achieved through a combination of "intuition, experience and repeated trials" and said this process has worked well as evidenced by the existence of many fine buildings and other structures. Building design has evolved into a complex, multidisciplinary and highly fragmented industry in modern times in which the rule-of-thumb approach has been supplemented with technological advancements, empirical knowledge and theory developed during the last centuries.

Design activities have become more formalised with the development of national and international standards and legislation, which in the UK include British Standards

4

Institution Codes of Practice (COP), Eurocodes, Building Regulations (BR), and legislation relating to quality assurance, construction management and safety. The latter are enforced by bodies like the Health and Safety Executive Construction National Industry Group. Professional institutions like the Royal Institute of British Architects (RIBA), the Institution of Civil Engineers (ICE), the Institution of Structural Engineers (IStructE) and the Chartered Institute of Builders (CIOB) promote good practice and strive to educate their members about recent developments. Groups such as the Construction Industry Computing Association (CICA) exist for the purpose of providing specialist advice to those in the industry. The Steel Construction Institute (SCI) and the Reinforced Concrete Council (RCC) and other trade association with a clear commercial focus offer support to designers as a part of their duty to promote specific products. The Building Research Establishment (BRE) and the Construction Industry Research and Information Assocation (CIRIA) are actively engaged in research in their respective areas.

Bedard et al [4] highlighted the fact that the process of developing building designs in modern times was substantially different from any design process in other fields of engineering. One important distinction is that building design has become a complex activity that requires input from professionals in different disciplines, not all of them engineering. Bedard et al [4] identified that modern building design demands the collective skills of numerous specialists, unlike other engineering projects, which: -

"remain for the most part under the control of one discipline - for example, road building by civil engineers."

Those specialists required to work in harmony together on a building project represent the building *design team*. The building design team encompasses architect, quantity surveyor, consulting structural engineer, mechanical and electrical services engineer and contractor as well as other parties. It is acknowledged that clients and accountants often exert significant influence during design development and as such have also been counted as members of the design team.

Howie [5] reported that, around 150 years ago, the architectural and engineering professions began to part company. After this time civil engineers became the lead designers in bridges, tunnels, harbour works and other large structures, and architects concentrated their efforts on buildings. Buildings both old and new clearly demonstrate the contribution made by architects, whose concerns have included aesthetic, functional and spatial matters. However, it is clear that modern buildings comprise several different

subsystems that interact with each other yet which operate according to different principles. As well as the architectural function, the structural system and foundations, the environmental system, the building services and transportation systems within a building must also be considered. Each subsystem is normally designed separately by a different discipline, in conjunction with the rest, on the assumption that the collection of individual designs together forms a complete, efficient unit. Functional design issues also frequently extend to address cultural implications as well.

The process of building design seeks to satisfy a number of goals simultaneously. Blockley [6] identified that, in general, the large scale of building design and the need to satisfy a somewhat flexible and possibly open-ended design brief has provided considerable opportunity in the past for designers to meet various requirements in novel ways. Lawson [7] mentioned some common architectural intentions, such as the provision of satisfactory functional spaces, proper proportioning and comfortable conditions for occupants. Billington [8] said the role of the structural engineer in designing safe, effective structural systems has remained unchanged while design and construction techniques have improved. Billington illustrated structural engineering innovation during the 19th Century and 20th Century with examples of buildings that were both highly efficient in their use of materials and behaviour, and which were also structurally elegant. Whilst the structural engineer is engaged in structural aspects, mechanical and electrical service engineers are involved with tasks such as achieving optimum energy consumption.

The contribution of every member of the design team is clearly important in modern times. The multidisciplinary nature of building design has raised awareness as to the importance of compatibility between the activities of each discipline. Coordination among parties has become a priority when setting organizational patterns and has often seen architects *novated* to project leaders, and assume responsibility for overall project co-ordination in addition to any other specialized duties. Within such an arrangement, responsibilities are normally delegated and it has been common practice for the architect to guide the structural engineer and other disciplines considerably in their respective duties.

Unfortunately, whilst acting as project leaders architects have also been known to study functional and spatial aspects and to make critical decisions, such as stipulating a general arrangement, without first consulting technical disciplines who have relevant experience in such matters, and whose own activities these decisions directly affect. Bedard et al [4]

highlighted the common situation where engineers were involved at a relatively late stage to make a design concept 'work' by developing what is, in effect: -

"a series of sub-optimal solutions consistent (or not conflicting) with the overall concept."

This has been found to be a highly unsatisfactory situation, and has been repeatedly criticized for resulting in over-specification and for requiring costly bespoke changes to be made at a late stage of design – see BRT [9].

Moore [10] reported that in the early 1960s, and with reference to the building industry, the observation made that in no other industry was the responsibility for design so far removed from the responsibility for construction; a fact that has contributed to the frequency with which disputes have arisen between designers and contractors. Bedard et al [4] later expressed greater concern about the extent of fragmentation throughout the entire building industry. He said that paradoxically, the successful creation of a single building structure relied on the coordinated efforts of groups,

"whose own aims may conflict, who have an incomplete awareness of each other's needs, who do not share the same model of design, who do not communicate in the same terms, and who may even follow different guidelines."

These factors amount to an *information gap* that is manifested in design solutions that fail to satisfy cost, quality and time constraints and fail to meet functional requirements. Having said this, it must also be mentioned that good communication and well-informed multidisciplinary decision-making is in evidence in completed buildings that have managed to effectively satisfy cost, time and functionality constraints. A good example is the Helicon retail / office development in London, described recently in engineering journals – see Russell [11]. This high-risk building project demanded an uncommon degree of collaboration between the design team members. Members were bound by an agreement to do everything possible to avoid conflict and modest additional monies were provided in the construction budget to appoint specialist subcontractors by reputation. Due to height restrictions, post-tensioned floors slabs were used to provide a slender, long-span floor slab 300mm deep, allowing a maximum number of stories (11) to be created and providing 20535m^2 of floor space.

Quality assurance systems such as BS5750 were introduced during the 1980s with the specific intention of bringing into effect structured management techniques to reduce communication problems. New forms of contractual arrangement were also introduced as

7

alternatives to more traditional contracts and procurement systems, in recent decades, to improve co-ordination and to reduce disputes. New contracts transferred responsibility for the engagement of parties required for the construction and supervision of works from clients directly to construction companies but maintained specific and statutory requirements. Through these changes, architects and consulting engineers were engaged by the contractor, and as a result engineers became more closely involved with other parties in initial, concept development than before. Thompson [12] reported that Design and Build contracts[1] (D&B), first introduced in 1981, have increased in popularity with the prospect of closer integration. In a recent CIOB publication [13], the advantages of D&B were cited as including: -

"single responsibility, speed of building, financial control, completion on time, economic building and (better) client relationships."

The need for reform in the construction industry was highlighted by the recommedations of the Latham report, being addressed[2]. The report suggested ways to achieve greater competitiveness through improved efficiency and cost savings. The recommedation includes changes to legislation[3] and greater cooperation led by industry governing bodies in areas such as management processes, teamworking / project partnering, greater use of standardised components, better understanding of the performace and life-cycle costs involved in construction and steps to avoid adversarial relationships.

2.2.1 The Evolution of the Design Process

Over time, design and construction activities have evolved and have become more distinct. Turk [14] accounted for the gradual separation of design (information-based) and construction (material-based) phases in the building life-cycle with the need for more precise co-ordination of engineering activities, which has also seen technical documentation and drawings supersede speech as the primary communication medium of engineering. Nevertheless, the complexity of modern buildings presents a significant challenge to the successful integration of the various disciplines.

Currently, the RIBA Handbook of Architectural Practice and Management [15], a standard reference, identifies four main stages of activity associated with building design. These stages are *assimilation* (or, design specification), *general study* (conceptual design), *development* (detailed design) and *communication* (construction), as shown in figure 2.1.

---

[1] including Construction Management contracts.
[2] NCE July 1994.

8

It is acknowledged that the design process involves a continual and necessary amount of revision and backtracking to accommodate new information and to resolve compatibility issues as they emerge. Hence, the boundaries of the design stages indicated within the model are not intended to be precise. Nonetheless, each stage embodies important actions. Initially, basic requirements and constraints imposed upon the building design have to be identified, generally from a vague client brief.[4] Design activity progresses towards the production of detailed specifications and drawings necessary for construction. Figure 2.1 and similar design models have been the basis of many design-support studies.

```
┌──────────────┐    ┌──────────────┐    ┌──────────────┐    ┌──────────────┐
│ Assimilation │ ─► │General Study │ ─► │ Development  │ ─► │Communication │
└──────────────┘    └──────────────┘    └──────────────┘    └──────────────┘
```
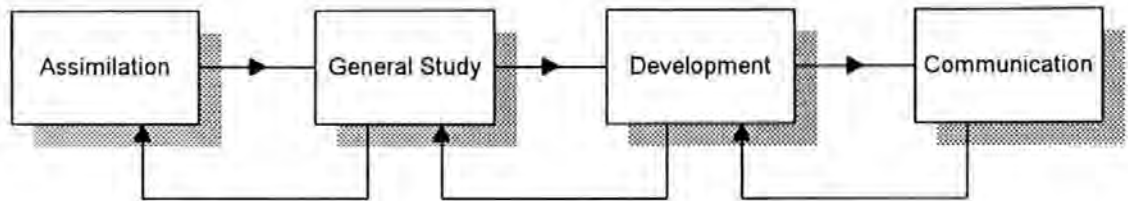
Figure 2.1: The RIBA plan-of-work map of the design process.

The RIBA Handbook contains several other associated design models. In another model created by Marcus et al (see RIBA [15]), interaction between sub-processes is shown to happen within each design stage. Others models presented in the literature illustrate activities advancing in parallel and coming under the control of different disciplines at different times, which emphasizes the importance of co-operation and information exchange within the design process. Models have also been used to differentiate *feasibility study* from *conceptual design* and *preliminary design*. Such distinction is not required in the scope of this study. For present purposes, it is sufficient to regard all early activity that follows on from design specification as conceptual design activity.

2.2.2 Conceptual Design Goals

Conceptual design follows a *synthesis-analysis-evaluation* cycle as shown in figure 2.2 to reach decisions (see Maver [16]). Due to time constraints, it is common to select one concept as the focus for all subsequent design activity at a very early stage of design. To this end, design guides like the IStructE Manual for the Design of Reinforced Concrete Building Structures [17] (hereafter, 'RC Manual'), acknowledge there is a need to produce alternative schemes for initial design at short notice, which can be assessed for architectural and structural suitability and compared for cost. The RC Manual says: -

---

[3] Specifically, adoption of the ICE New Engineering Contact.
[4] here, a vague brief means one that is flexible and open-ended.

"Although based on vague and limited information on matters affecting the structure, viable schemes must nevertheless be produced on which cost estimates can be based."

The conceptual design stage encapsulates the critical decision-making related to consideration of different alternatives for the purpose of determining a preferred solution. The goal of conceptual design is to broadly identify beneficial high-level options that greatly influence the final appearance and cost of a building, whilst also ensuring that it meets functional requirements. Ideally, a preferred concept could be determined from a detailed study of a number of alternative concepts. In practice, however, economic factors usually prohibit a sufficiently thorough investigation and critical decisions continue to rely heavily upon the perception, experience, preferred practice and other influences of senior designers within an architectural or engineering practice.
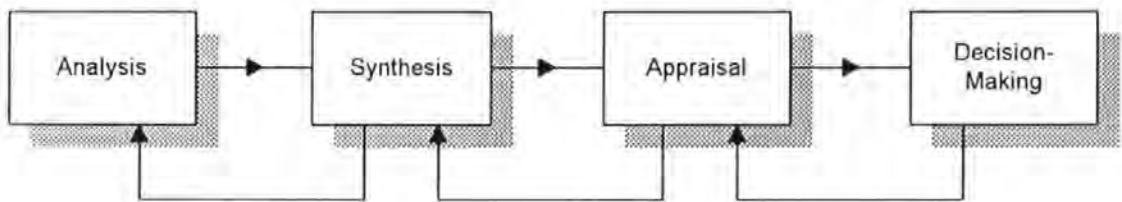
Figure 2.2: The decision-making process.

Structural aspects that are significant and which require broad consideration during CBD include the overall geometry of the proposed building layout, the column grid arrangement, the structural frame type including vertical and horizontal load subsystems, the foundations and the building envelope. The building envelope itself comprises cladding, roofing, windows, and means of access.[5] Other significant engineering aspects linked to the superstructure and normally considered in broad terms include transportation systems and building services.

Once a concept has been adopted it is then continuously refined to a greater level of detail until a compatible and satisfying solution is reached. After general aspects have been fixed, the subsequent local design of individual members and subsystems in order to satisfy various design criteria can proceed, marking a transition from conceptual to more detailed design activity.

---

[5] including fire escapes.

2.3 Information Technology for Conceptual Design

Rafiq et al [2] noted that as new construction materials have become available so new construction techniques and tools have emerged to meet the needs of the time. Manual techniques have been developed to improve design practice, such as BS5750 mentioned previously. However, the electronic microcomputer is perhaps singular amongst recent tools in its capacity to revolutionize the design process by offering first-hand support to designers in their activities. The desktop microcomputer has become commonplace in the modern structural design practice as a result of miniaturization and successive improvements in price-to-performance ratio.

Computer aided design (CAD) tasks of a discrete, sequential, analytical and numerical nature have generally been well suited to software-based support. Amongst the more familiar applications of commercial software supporting the detailed design and communication stages are programs for structural analysis and conformance checking against design COPs and computer-aided draughting (CAd) packages. These applications have had a major impact on working practice and have improved efficiency during the later stages of design[6] – see Taffs [18]. They also illustrate the disparity that exists in software developed to assist in conceptual design tasks compared with that aimed at supporting detailed design and construction activities. Research suggests that around 80% of the total resources required to construct a building are indirectly committed by the decisions made in the first 20% of the design life-cycle, during the conceptual phase, which further illustrates the imbalance of support tools in conceptual design and later stages – see Deiman et al [19] and Evbuomwan et al [20].

The many possible uses for information technology (IT) in the design practice, where IT refers particularly to the electronic computer and communication systems, extend to tasks such as administration, project management and planning as well as design itself. Within design, there are further differences in the ways and extents to which assistance can be provided. Survey results have shown that software support has tended to concentrate on discrete tasks.[7] Taffs [21] reported in 1994 that spreadsheets and draughting programs were amongst the minority as programs that were popular and widely used by practising structural engineers, at the time. In reviewing IT developments to 1998, Grierson [22] asserted that, within the last century: -

---

[6] As manual tasks have become automated and as greater reliance is made of computers, considerations regarding the nature of education and training of engineers and designers must be made, including how best to develop the requisite CAD skills for utilising new technology whilst retaining an intuition for design.

"it is information handling and distribution and not information production that has experienced explosive growth."

Thompson [12] noted that general-purpose numerical problem-solving software tools like the spreadsheet brought benefits through processing and presenting design data, whilst databases and document management tools were amongst those to offer a limited form of support by being able to collate, record and retrieve relevant information. New technologies such as compact disc media, electronic mail and the Internet offer additional benefits to businesses. Thompson [12] asserted, however, that construction companies generally have been unable to exploit greater benefit from IT because of the "unavailability of dedicated and integrated IT systems and expertise" to support principal design activities. In particular, Evbuomwan et al [20] have advocated that suitable software tools are much in need to raise awareness of 'downstream' design issues during 'upstream' decision-making as a step towards concurrent engineering practice.

These opinions support the view that there remains the opportunity for appropriate software with some kind of multidisciplinary appreciation to provide practical DS at the conceptual stage of building design. An important capability of such software is to empower designers with both the knowledge and the means to be more effective at creating highly satisfactory solutions. In this way, it is hoped that software tools may not only help to improve efficiency through supporting necessary activities, but may also realize improvement in the quality of the final product as well.

2.3.1 CAD Tools

The fact that CBD support tools within commercial practice are conspicuous through their absence belies the considerable research effort that has been dedicated to their study in recent decades. The breadth of the subject has encouraged investigations using a variety of techniques, which have in turn addressed a number of key issues. Whilst many approaches have shown some promise, few techniques have been incorporated into new commercial applications. Part of the difficulty that has been encountered has been attributed to the fact that the overall process of design comprises a loose collection of other processes of a *non-monotonic* (or, non-sequential) nature. A sufficiently broad and flexible approach is required to meet the needs of designers.

---

[7] Active support involving decision-making has been distinguished from passive support involving automatic design assessment based on code conformance and direct cost.

Current CAD tools can be discriminated according to their function and, more specifically, according to their decision-making capabilities. Many classical optimisation techniques and methods of structural analysis that use numerical methods are only helpful in obtaining a specific solution to a well-defined problem. Standards-processing software has similarly been developed to conform to specific design methods as found in COPs. Software suites have been developed for both types of program, collecting together different design methods for versatility. For example, Kousmousis et al [23] created a general-purpose engineering optimisation tool called ADS Expert that brought together a number of classical optimisation techniques. In addition, commercial applications like SCALE[8] assist in designing various structural components in accordance with design standards. In addition to these types of programs, other approved proprietary design and construction techniques have been made available in computerized form to assist designers and to promote the use of particular products. Software has also been developed specifically for use in educating and training engineers in particular design methods.

In contrast to these types of program, most CAd systems, whether stand-alone or those supporting co-operative design development (by co-ordinating access and modification to construction drawings), possess little or no specific design knowledge particular to their application.[9] Instead, economic benefits have been mainly derived from directly improving working practice. CAd has been reported to have been particularly effective in situations where simple drawing elements are repeated and where it has been necessary to amend construction drawings regularly, see Taffs [21]. The technology for supporting greater collaboration amongst geographically remote designers and for working in an electronic, rather than paper-based environment are amongst other CAd benefits that are likely to become more significant in the future.

2.3.2 Artificial Intelligence Tools and Techniques

There are potentially a vast number of design alternatives, and combinations of constituent parts that can make up a design and which deserve examination. After consideration of design compatibility, economic aspects, functional performance and other measures of acceptability only small proportion of highly satisfactory solutions are likely to remain. The quest for a pseudo-intelligent, machine-based approach supporting efficient concept development is led by these factors. Unlike standard CAd programs, *intelligent systems*

---

[8] SCALE is product of Fitzroy Computer Systems Ltd., 50 Fairmile Lane, Cobham, Surrey, KT11 2DF.
[9] Specialist products continue to be developed to integrate with general-purpose draughting programs, such as for steel detailing or RC detailing. Once again, these commercial developments are focussed on detailed design and communication phases. A few commercial programs support parametric design studies.

13

include those that contain and process pertinent design knowledge. This requires an appreciation of design activities, including design criteria, requirements and constraints and alternative solution strategies that may be suitable. Computational DS involves software systems that can synthesize design knowledge and reasoning to the advantage of the designer, and the study of intelligent systems is a generic category of AI research.

2.3.3 Current Aims

Whilst this study has explored the opportunity for applying one type of evolutionary technique – the GA – in CBD, many very important lessons and guiding principles for the successful creation of support systems in the field originate from studies involving KBESs. KBESs and evolutionary techniques have developed as separate fields of AI with some commonality in purpose. This thesis aims to show that EAS techniques can surpass the capabilities of KBESs in some ways. The remainder of this chapter presents relevant findings of research based mainly on KBESs.

2.3.3.1 Design Support using Knowledge Based Expert Systems

The KBES is a generic type of computer program capable of representing and processing knowledge used by human experts to perform specialized tasks, see Bedard et al [24]. The KBES was arguably the most significant outcome of AI research in the decade of the 1980s, according to Adeli et al [25]. Since KBESs have always been costly to develop they have been created with the intention of realizing a significant benefit in supporting complex processes. This has tended to involve areas of human knowledge where uncertainty has prevailed and where knowledge has been incomplete, often because the breadth of the subject area has precluded any one person from being fully aware of all the related knowledge, its inter-relationships and implications at any one time.

The usefulness of KBESs is derived from their ability to automatically make inferences about supplied information using the pre-programmed knowledge they contain and in making the reasoning, outcome and consequences available to a user. Notably, only a small proportion of KBESs have been created to support formative activities like design;[10] the majority of KBESs support deductive activities which include monitoring and fault diagnosis.[11] Jackson [26] asserted that this had much to do with difficulties inherent in modelling broad, non-monotonic design processes, as encountered in CBD.

---

[10] This refers to information generation activities mentioned previously by Grierson [20].

KBESs supporting design tasks have also been called Knowledge Based Design Systems (KBDS). KBDSs made up a significant proportion of early CBD support tools and employed a combination of established factual knowledge for designing specific types of structure and heuristic knowledge, which was often previously undocumented and elicited directly from appropriate, experienced experts. Design information was structured hierarchically in these systems using programming languages like Smalltalk, ADA, Eiffel, Lisp and PROLOG in order to provide a *knowledge-base*. These languages are all symbolic, and some exist in different dialects and support the OOP paradigm.[12] The separate part of the KBES which applied reasoning in order to select or reject alternative subsystems, components and parameter dimensions so as to direct search within an initial large domain towards a satisfactory solution became known as the *inference engine*.

The earliest reported structural design systems intended for commercial application had very limited scope and appeared in the late 1970s. They included SACON (see Bennett et al [27]) and SPECON (see AISC [28]). These systems were developed to automate the selection of suitable design components from comprehensive, predefined lists by asking users questions about the circumstances in which they were to be used. Before this time, computational DS had been attempted in the more limited form of *Management Information Systems*, developed during the 1960s for collating previously disparate design and construction knowledge. The most significant advances in supporting building design by means of KBESs were made later, during the 1980s and 1990s. The next section describes developments during this era and the present situation.

2.3.3.2 Discoveries and Difficulties

KBESs used a pre-programmed set of rules and conditions as instructions for solving a generic problem. A sufficient number of decision rules were required to enable comparisons to be made between a significant number of alternatives. It was rare for the factors that determined the most suitable design alternatives to be so simple as to be easily and accurately classified using a set of general rules. More often, rule-based knowledge would unintentionally introduce assumptions into the rule-base, at the time of their development, regarding what the best solutions were.

During the synthesized reasoning process that a KBES employed, it was often necessary for key decisions to be resolved in advance of other aspects. As such, certain domains were found to be more readily separable into subdomains, in which design knowledge

---

[11] The only known commercial design-oriented KBES called ELSIE was used in direct cost estimating.

relating to specific aspects could be structured, than others. Early studies were quick to exploit facts and *hard* (or, inflexible) constraints such as statutory requirements that could be formulated as *elimination rules* and which were guaranteed to favourably reduce a problem domain. Many systems also used generalizations based on *soft* constraints which acted like intelligent guesses and which benefited from qualification. Complex problem domains involving mostly elimination-type rules were rare and yet were evidently more amenable to KBES representation.

Demonstrational building design support tools like the HI-RISE system for the preliminary structural design of very tall buildings by Maher et al [29] in 1985, and the INDEX system for the design of industrial buildings by Kumar et al [30] in 1988, are notable examples of systems that exploited domains containing rigid design knowledge. Both of these domains were also unusual in that structural aspects required special attention and tended to dominate the decision-making process. Unfortunately, subsequent studies have shown that more common types of structure, like medium-rise office buildings of the sort this study aims to help design, present a greater challenge to KBES creators than specialist types. This is because greater choice is afforded during design, and because often a compromise must be reached to satisfy the goals of different disciplines.

Besides certain domains being directly more amenable than others to KBES representation, other shortcomings were apparent. The broad nature of building design forced almost all KBESs to be specialized and have narrowed scope. Some systems assumed that a particular construction material was to be used for the building frame from the outset, which immediately limited their scope. The pre-supplied rules and relationships between component parts meant that often the path to a suitable solution was largely inflexible and pre-determined for given inputs, despite considerable efforts on the part of the inference engine creators to make them less so. Furthermore, sometimes-unjustified assumptions were made or important provisions were ignored. Whilst this aided development it also made systems more unrealistic and impractical. Effective designer interaction was arguably the most important feature of all to be lacking in early systems to make them genuinely helpful.[13] Many researchers studying KBESs addressed themselves to these important issues.

In 1985, Lane et al [31] were amongst those who recognized that decisions taken by designers during conceptual design amounted to informal optimisation. This suggested to

---

[12] This is a popular paradigm for modelling design processes.

others that the use of formal computer-based optimisation could beneficially supplement captured knowledge where the building domain was less explicitly constrained, and where uncertainty or incomplete knowledge existed.[14] Lane et al [31] recognized that applying optimisation at the conceptual stage could realize greater benefits than introducing it later, when decisions become more restricted. An approach in which decision-making was interspersed with calculations was developed, despite at first appearing contradictory to the *conceptualize-analyse-detail* approach of earlier investigators, wherein numerical analysis had been applied after heuristic knowledge. In 1993, Reddy et al [32] successfully implemented procedural numerical analysis within a KBES as Lane had proposed for structural component optimisation.

After Lane et al, many bespoke techniques appeared for combining declarative programming styles for design synthesis with procedural programming for analysis, standards-checking and for providing explanation facilities, rather than for optimisation *per se*. Researchers such as Kumar et al [33,34] and Ades [35] developed techniques that enabled procedural languages such as FORTRAN and C to be interfaced with declarative languages like PROLOG, in 1988 and 1991, respectively. The combination of declarative and procedural techniques was made easier by the creation of expert system shells that used specially developed *knowledge engineering languages.*[15] It is important to note that some advances in software came about through ongoing developments in computer hardware at the time, in areas such as hard-disk capacity, display technology and processor speed.

Sriram [36] and Harty [37] separately considered improvements that could transform the HI-RISE program that applied a rigid set of rules, into a more practical system. Both interested parties thought it would be beneficial to allow alternative concepts to be more easily generated. In 1986, Sriram [36] tried an exhaustive, generate-and-test approach to find all feasible building concepts using a KBES called DESTINY. Practicality was marred by the problem of combinatorial explosion; whilst giving too few options to the designer was found to be restrictive, too many present an overwhelming choice. The time required to develop solutions was also prohibitive, of the order of 40 minutes. Note that by 1987, DESTINY had become incorporated into a large *integrated structural design system* (ISDS) called ALL-RISE, which was devised to become a general-purpose successor to HI-RISE. ALL-RISE was described by Sriram [38] in 1997.

---

[13] mainly because the necessary enabling technology was also still developing at the time.
[14] Not only this, but design knowledge can also be unreliable, used out-of-context, inaccessible until other aspects are resolved, or else may simply be wrong.

A different approach taken by Harty [37] in 1997 to improve upon HI-RISE involved assigning *operator-weighting factors* (or, certainty factors) to construction cost, time and *buildability* (or, constructability) considerations in order to model variations that were observed in practice in the relative importance of these criteria. The result was a more flexible system called DOLMEN. Through the introduction of a GUI, a designer was able to adjust design objectives and the extent of constraint satisfaction interactively, to help determine a satisfactory solution. The system was refined, as reported in 1994, see Harty et al [39].

2.3.3.3 Supporting Non-Monotonic Reasoning

Early systems were recognizable through the common features and standard operating modes they shared. Over time, KBES for structural design led to further work involving ISDSs that were mostly based at larger research centres and with which it was hoped the entire building design process could be modelled. ISDSs were often the product of combining separate modules developed within individual research projects to address specific aspects of building design such as the generation, appraisal and verification of solutions. Later systems included modules that represented the activities of different disciplines, such as structural design, energy efficiency analysis, daylight calculation and noise transmission modelling. These systems also became known as *integrated building design systems* (IBDSs).

An example of an ISDS is ALL-RISE, mentioned previously, which comprised of four main programs: a structural design module based on HI-RISE and DESTINY, and other modules called FLODER, LOCATOR and STRUPLE. FLODER was created by Karakatsanis [40] in 1985 to determine gridlines for locating structural elements. LOCATOR was created by Smith [41] in 1986 to assist in providing lateral load resistance, and STRUPLE was used to help identify subsystems and components from existing buildings that might be applicable in new designs.

The flexibility of modular systems was seen to be an important factor in supporting different aspects of design effectively, and led to the development of sophisticated control methods that included the *blackboard architecture*. This was a significant paradigm that allowed modular processes within ISDSs to make information available to one another via a central data repository, rather than having to rely on direct inter-communication. Many KBESs had featured either *backwards-chaining* rule processors that were goal-driven using

---

[15] These tended to be slower in execution because they used interpreted, rather than compiled, languages.

consequent rules or else *forward-chaining*, source-driven systems using antecedent rules.

In 1988, Paek et al [42] showed the possible advantages of a combined approach using a KBES called FRAMEX which was developed to support the design development of uniform steel-framed structures. Other search strategies, including hierarchical breadth-first and depth-first search were noted to have various benefits and drawbacks. Trial-and-error was commonly used for determining appropriate structural member sizes in early systems, later replaced by analysis methods. Separate component databases also emerged as another common feature after they were found to be easier to maintain by users.

2.3.3.4 Supporting Interaction

Some unique characteristics of the building design process were mentioned at the start of this chapter. In 1990, Bedard et al [4] identified another aspect that differentiates building design from other engineering disciplines like mechanical engineering – namely, that the building design process normally involves the creation of an unique artifact in a natural environment rather than one that undergoes lengthy prototype testing before final mass production in a controlled environment.[16]

Early KBESs overlooked the often unique nature of the design specification, and anticipated that a predefined and rigorous question-and-answer execution sequence might suffice in making satisfactory design decisions, to obtain satisfactory solutions. It became clear that such hopes were unrealistic. The introduction of additional functionality, as demonstrated by Harty et al [39] via a GUI in 1994, heralded an important change in perspective as to how uncertain and project-dependent knowledge could be effectively handled. The significance of being able to develop a design solution through an iterative/recursive process rather than from start-to-finish in a single rigid consultation cycle for offering practical DS gradually became apparent. In this way, the designer is given limited power to explore various possibilities instead of being forced to automatically accept (or reject, as was more often the case) a single solution outright.

An ISDS developed at the University of Strathclyde highlighted another important advance. The system featured a number of modules called GOAL, ANM, RELATOR and SCG. These modules were created by Rafiq et al [43] in the late 1980s with the exception of GOAL, which was created several years earlier by Sussock [44]. GOAL was used to appraise layouts using heuristic information. ANM was a space frame analysis module.

RELATOR was a numerical rule-processor for component sizing and costing. SCG was a structural concept generator module, and represented an innovative development. It contained a CAd-style interface that enabled an architectural building layout to be manually generated in plan. The system employed PROLOG rules to check the suitability of a layout and was capable of making adjustments that it considered appropriate. The system was also able to allocate structural nodes for subsequent analysis and member sizing. The CAd interface was recognized as being necessary for supporting interactive design development. It was used to manually manipulate aspects of the design. Further work on SCG was described by Ades [35] in 1991.[17]

Perceiving similar practical benefits, and in 1991, Jain et al [45] described functionality provided in another KBES that enabled a designer to graphically manipulate the dimensions of the column grid and structural core location of a building in order to determine an initial, suitable floor plan. However, interaction in this system was restricted to initial layout configuration only, and the system still maintained certain unjustified assumptions regarding the presence and location of certain design features. Later still, in 1994, a computer-aided architectural design tool called KAAD was purposely designed by Carrera et al [46] to emulate: -

"an architectural design process characterized by the parallel development, and gradual reconciliation of design requirements."

Particular attention was given here to making the computer subservient to the needs of the designer, again via a central CAd interface. In 1997, Najafi [47] described how the KAAD system built on previous efforts and strove to follow what was termed the Partnership Paradigm principle where labour could ideally be divided between user and machine in a manner that imposed: - "neither a pre-defined sequence, nor a pre-defined task allocation."

Design development was regarded as a convergent process, comprised of standard and non-standard activities. The user was involved in cooperatively developing a solution. KAAD demonstrated that DS could be realistic given a software environment in which the capabilities of a computer are utilized in tasks suitable for automation or semi-automation, and these processes linked to other tasks that required a greater degree of manual involvement.

---

[16] Indeed this aspect that has sometimes led to criticism regarding the unnecessary amount of bespoke design and has been a cause of criticism of the relevance of BS5970 to construction processes.

2.3.3.5 Supporting Creative and Routine Design

Whilst Bedard et al [4] and others acknowledged the requirements and constraints upon a building collectively create a unique specification, Harty et al [39] and others noticed similarities amongst structures that shared the same purpose. Office blocks, for example, were identified as being broadly alike in function and form. Since building design often involved the appropriate recombination of design elements and subsets from a vast but otherwise relatively familiar set of alternatives, over a period of time the building designer could be expected to become a reasonable judge of good alternatives. This was the view expressed by Harty et al [39] and shared by many researchers whose KBESs, in effect, attempted to replicate the actions of the designer using knowledge elicitation and rule-based knowledge representation techniques.

Experiments by Lawson [48] in 1972 revealed that architects and engineers use analogy, pattern recognition, pattern manipulation as well as formal design knowledge in problem solving. In practice, it is clear that the building designer, whether architect or engineer, seldom relies entirely on familiar examples and methods without also exploring the opportunity for change and adaptation, thereby blending imaginative and innovative ideas with their own practical experience, see Maher et al [49]. Whilst conceptual design certainly does involve an element of *routine* (or, repetitive) activity, it usually affords the opportunity for creative thinking in appropriate ways, a fact recognized by Sandgren [50]. Harty et al [39] asserted that in the design of office blocks the greatest opportunities for novelty and efficiency were via "geometrical and topological variation", and through "structural and architectural preferences." The former suggests scope for variation in aspects such as overall building dimensions, bay spacing and storey height, whilst the latter includes choice of structural and cladding materials and the presence of options features like a service core, shear walls or atrium.

2.3.3.6 Intelligent Decision-Making and Learning Capabilities

Adeli et al [51] made several important contributions to the development of KBESs. He observed that most early, 'first-generation', KBESs failed to exhibit a capacity for learning; such KBESs usually needed specialists to update their static knowledge-base and few systems had any kind of knowledge acquisition or memory capability beyond that used to store the original set of rules. As information was stored locally within the system,

---

[17] Ades extends parts of the work further in his more general research in structural CAD techniques and produced the DESIGNER-M program which had an interface allowing the interaction between design parameters to be observed, e.g. how total cost of an RC beam can vary with section depth.

creativity was limited. Adeli et al [51] demonstrated an efficient, functional KBES developed using only production rules programmed in a high-level procedural language – Pascal, in 1989. This was significant because it suggested to other researchers that the way forward might not necessarily lie only in symbolic / declarative programming paradigms.

Several researchers began to consider alternatives techniques that might improve upon the performance of rule-based systems. Woodbury [52] appeared as one of the first persons to formally propose the idea of using design mutation within the framework of a KBES specifically for synthesizing the generation of non-standard building concepts. How this might be possible was a challenge he presented to the research community in 1993, a time that marks the start of a shift of emphasis away from pure KBES implementations, towards alternative and sometimes, complimentary, AI techniques. Even so, there is ample evidence to show that research into new KBESs for conceptual design has continued to bring incremental benefits, from various directions, for example in research by Sabouni et al [53] in 1996, and Najafi [47] in 1997. In particular, KBESs have made increasing use of multimedia techniques and have combined textual and graphical information to enrich support capabilities. Design interaction has also been further improved.

In 1996, Smith [54] asserted that whilst rule-based KBESs positively help to direct a designer's attention to important feasible sub-domains within a design space, their practicality is otherwise limited when developing individual and new design solutions. EAS techniques, artificial neural networks (ANN) and case-based reasoning (CBR) have emerged as alternative technologies that are also currently being eagerly investigated in the hope of advancing support for design-related activities by following different approaches. Both ANN and CBR research in CBD has been encouraged by the view that it may be more effective to attempt to model complex systems according to the characteristics observed in actual solutions, rather than by applying generalizations that may or may not have relevance in different circumstances.

ANN and CBR approaches aim to train systems with functionality using examples, instead of attempting to hard-code it within a program. Both ANN and CBR techniques employ deductive reasoning for their decision-making. CBR involves the study of successful case histories; in the present context this refers specifically to real, functional building structures. A case is like a memory that is recorded and made accessible for reference at a later date. Consulting a CBR system involves two stages: retrieving relevant past cases and adapting them to suit new requirements. A case consists of a detailed description of

the design problem (client brief), the solution and any significant steps used to obtain it, including textual and graphical design information. The retrieval of cases is usually activated from partial similarities with a current design problem. This means that various case histories may be retrieved at once, each having some similarity to different aspects of a new building. Furthermore, the same case may be retrieved for different reasons in different design scenarios. CBR is still a developing field. In 1995, it was reported that CBR showed considerable potential in the building domain where the degree of similarity amongst structures was very high - see Maher et al [55,56]. CBR techniques have been developed to permit the case-base to grow with use and become more powerful as a result. Like CBR, the study of ANNs (or, synthetic neurology) is similarly founded on deductive principles and has had widespread application, in design and beyond. In 1996, Gero [57] mentions that constantly reinforced neural networks may offer a learning capability. In 1999, Rafiq et al [58] offered the following description of ANNs: -

"Neural networks can be used to attempt to discover unknown relationships that can exist but are not known, by studying how the outcome of a process depends on the input parameters and conditions and by trying to find a pattern that fits all test cases. This relationship can then be applied to other data to see if it holds for different situations."

The application of ANNs is described in more detail in subsequent chapters.

3 Evolutionary Design and the Genetic Algorithm

3.1 Introduction

The study of GAs differs from that of KBESs, ANNs and CBR techniques in that problem-specific knowledge is not applied directly in order to take decisions – i.e. the standard implementation a GA does not utilize formal inductive or deductive reasoning techniques. Furthermore, whereas many classical optimisation methods may be likened to KBESs in that both usually involve some kind of preset directed search, the GA represents a stochastic numerical search method[1] that seeks appropriate solutions to problems largely through discovery. Note, however, that this does not imply by random chance alone, as is the case with single-point mathematical programming techniques such as the Monte-Carlo Method (see Himmelblau [59]). Instead, the GA is a multi-point search technique that is guided by selection pressure.

3.2 The Genetic Algorithm

The GA is one class of EAS technique that has found widespread application in domains that involve search, optimisation and machine learning. Previous applications are diverse, and range from attempts to create art and music to extensive studies of scientific and engineering problems. New areas of application are reported regularly. Within the engineering design domain, the GA has been applied in both detailed design and conceptual design studies. The GA has featured in structural design studies and has been applied in other fields of engineering. Past successes have encouraged and influenced the direction of the present study.

The engineering application of GAs has its origins in research carried out by Rechenberg [60,61] during the late 1960s and early 1970s, associated with aeronautical engineering. At the time, the potential for computational problem-solving methods based on evolutionary processes was poorly understood. Later, Holland [62], Goldberg [63] and Davis [64] became pioneering investigators of the biological paradigm for problem solving.[2] These researchers produced seminar literature, in the 1970s, 1980s and 1990s respectively, which illustrated the scope for practical application of GAs. Their work has established accepted theoretical methods (with allowance being made for some fine differences in opinion based on individual viewpoints).

---

[1] A non-gradient based direct search method where only the objective function value and not the first derivatives are found.
[2] Note Davis' text describes the work of other prominent investigators, including De Jong and Syswerda.

The term GA was introduced by Goldberg [63]. It was used to describe to one of a number of closely-related EAS techniques that are now specialized fields in their own right. Others include Evolution Strategy, Evolutionary Programming (better known as Genetic Programming), Cellular Automata, Simulated Annealing (SA), the Ant Colony Metaphor and parallel programming techniques. In 1999, De Jong et al [65] referred back to the origins and differences between some of these techniques to present current research directions and to predict future trends. On examination, it can be seen that GA and ES techniques are fundamentally very similar; they differ only in the emphasis placed on certain operators that they both share. (These operators are described shortly hereafter).

Whereas KBES research has always attempted to address decision-making at a broad level, the GA was extensively applied to solve detailed problems where optimal or high-quality solutions were the prime objective in the first instance, before being considered suitable for, and applied to, design tasks of a conceptual nature - see Goldberg [66]. Parmee et al [67] stated that the GA has acquired two different roles:- as a flexible DS tool for exploring a broad search space and for use in locating global optima. A concise introduction to the GA is appropriate in order to pursue this statement further.

3.3 The Simple Genetic Algorithm

In 1989, Goldberg [63] introduced the philosophy, theory and mechanics behind a GA to a wide audience. The Simple GA he described has been widely recognized as a canonical definition of a GA. Goldberg described how, in permitting an unconstrained search of a design space, the GA offered tremendous potential for modelling complex problems that can otherwise present great difficulty in formulation.

It is important to state at the beginning that the GA is not infallible. Not only does it perform poorly at finding a spike in an otherwise flat landscape but also the solution may be sensitive to uncontrollable extraneous factors. It is viewed as the best technique, where no better technique exists.

In engineering design applications that have involved the GA, appropriate design parameters are chosen and *encoded* (or, represented using some form of mapping) to create an artificial *genetic string* or *chromosome*. Each design attribute selected in this way becomes an artificial *gene* within the chromosome. In this thesis, the term *genetic experiment* is used to refer to the execution of a GA, applied to a particular task. During the course of a genetic experiment, chromosomes are perturbed in such a way that genes

frequently become modified, constituting changes to the value ,of the corresponding variables. The term *genotype* or *genome* refers to the content of the entire artificial chromosome. The term *phenotype* refers to the observable characteristics of a particular genotype - in other words how that particular genotype, when *decoded,* yields a normally unique design solution with quantifiable *fitness* (or, measure of suitability).

In 1986, Dawkins [68] used the analogy of a blueprint of a house or car as being a two-dimensional representation of some three-dimensional object to illustrate how multi-variant information can be reduced to two dimensions and ultimately to one, without there necessarily being any significant loss of detail. He suggested how an artificial, one-dimensional chromosome might define the full or partial physical form of an entity through artificial genes that use a code to represent dimensional, geometric, topological and other characteristics.

By means of a binary encoding scheme, a computer is able to represent and manipulate important features using a simple string consisting only of zeros and ones. In such a scheme, a single bit becomes equivalent to an *allele* in the field of genetics, which may be regarded as the smallest atomic unit from which genetic information is constructed. A gene can be represented by a single allele or by concatenating several alleles. Genes that take different binary values indicate variations in design attributes. Concatenated genes create an artificial chromosome. In the context of design, the power of the GA lies in its ability to efficiently manipulate segments of chromosomes that represent different design aspects, in order to produce beneficial design variations.

A genetic experiment begins with the creation of an intial set of chromosomes. In the Simple GA, chromosomes are generated randomly and adopt a binary-encoding scheme. In the context of design applications, each of these chromosomes would represent an individual design solution whose suitability (for some predetermined purpose) can be independently evaluated. The set of chromosomes constitutes an initial *design population.* The GA follows an iterative cycle, in which each iteration – or generation - produces a new design population, and represents an evolutionary step. The GA continuously strives to improve upon the intial population through the iterative process, by cumulatively seeking to select, combine and retain beneficial attributes from different individuals, whilst also rejecting and replacing disadvantageous attributes. During the iterative process, *genetic operators* are applied that are based upon evolutionary selection pressures in natural

systems. High fitness solutions normally emerge in successive generations as a result of applying these operators.

The processes of selection, reproduction and replacement are fundamental to the operation of the GA. Figure 3.1 is a flow diagram of the GA that shows these steps. Selection, reproduction and replacement are performed sequentially, within the main execution loop. During the selection process, chromosomes are selected from the population to go forward into a mating pool that is used to produce a generation of new offspring chromosomes. The likelihood of any individual entering the mating pool is determined stochastically and based on its current fitness, relative to that of its peers. Fitness is determined through design evaluation (appraisal).
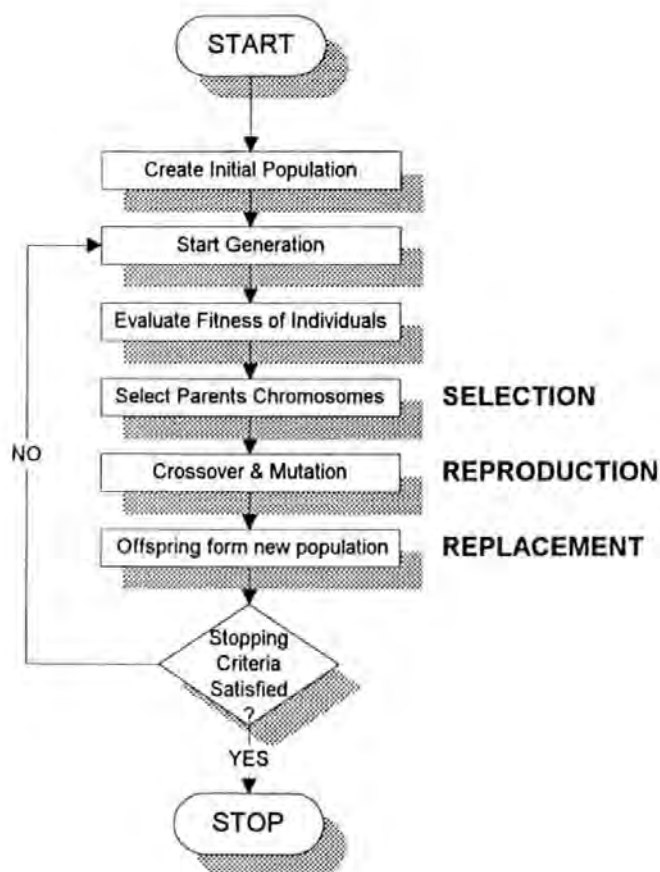
Figure 3.1: Flowchart of the Simple GA.

The evolutionary process that produces design variation takes place in the reproduction stage of each generation. Within this stage, parent chromosome pairs are artificially mated to create offspring chromosome pairs. Although there are believed to be many reproductive mechanisms at work in nature, Goldberg [63] was able to demonstrate a simple yet effective GA which applied just two; namely a synthesized recombination

operator called *crossover* and a synthesized *mutation* operator.[3] The purpose of crossover is to combine parent chromosomes in order to produce new offspring in which beneficial genetic information is retained and improved upon, to effect overall improvement in the quality of a design.

Dawkins [68] contemplated the significance of mutation and related processes in being responsible for the evolution of species in the natural world. In the Simple GA, Goldberg employed mutation as a secondary process to produce changes within offspring chromosome segments that are not necessarily present in either parent design. In this way, mutation helps to maintain diversity by introducing new, and potentially useful, genetic material into the offspring population. Crossover and mutation processes are simple to program using string manipulations, and because they are stochastic, their incidence is usually determined by respective probabilities set at the outset of an optimisation. In the Simple GA, pairs of offspring chromosomes replace parent chromosomes directly in the population.

3.4 Considerations in Applying Genetic Algorithms

As mentioned before, a significant advantage for using a GA is that it may be applied in many situations that present great difficulty for conventional calculus-based optimisation techniques. Specifically, this refers to design problems that prohibit the use of gradient methods since the functions that would be needed to implement them, and their first derivatives are not continuously available. Many practical problems involve noisy or discontinuous functions; structural engineering design is itself a field that contains a mixture of discrete and continuous variables. The GA is capable of exploiting good information rapidly and of maintaining multiple solutions, simultaneously. Conventional optimisation techniques are particularly unsuitable in domains that contain multiple, sub-optima. This is especially significant of the conceptual design domain.

Diverse studies of the GA preclude complete coverage. However, there is a practical purpose behind much GA research, and certainly in conceptual and detailed design studies. The suitability and effectiveness of GA-based techniques have been found to be highly dependent upon a number of factors common across most applications. These factors include the nature of the problem under consideration and the manner in which variables are selected and objectives are expressed (also referred to as the *environment* and the

---

[3] Whilst there is no doubt of its importance, opinion is divided as to whether reproduction, which refers to the entire process in which parent chromosomes are selected from a mating pool, and may subsequently undergo crossover and / or mutation, is an important operator in its own right.

fitness landscape). Also important is the effect that various refinements can have in improving or degrading the performance of the basic algorithm.

Numerous GA-based discoveries have been forthcoming as a result of directly addressing complex real-world problems. In addition, advanced, general-purpose techniques that have been studied at a more abstract or theoretical level have enabled scope to be extended to new areas of application. The engineering domain provides a basis for both highly practical and highly theoretical GA study. In 1995, Rajan [69] summarized this in stating that research interest in optimisation including the GA has developed over time to address the need: -

"to handle a wider class of problems, to include realistic definitions of design variables, to find techniques to locate the global optimum and to improve the efficiency of the numerical procedure."

This this describes research that investigated the suitability of the GA in supporting the activities of CBD, an aim that is clearly application-oriented. It represents a significant deviation from the majority of preceding structural studies and as such has much to offer through its novelty alone. An approach representative of the real issues and criteria involved with CBD is needed. In the context of previous research in KBESs and related fields, new techniques found to be capable of overcoming shortcomings in existing approaches would have particular significance. Since practical DSSs having no limitations or drawbacks whatsoever remain an elusive goal, the potential benefits must be gauged according to the relevance and consistency of any information generated in relation to the common role of the designer, remembering particularly lessons learnt from KBESs. The scale of application and the perceived benefits are important considerations; in 1996, referring to the application of the GA, Parmee [67] said that: -

"the detail of any system modelling must be commensurate with the degree of confidence in the available data. "

The detail to which relevant data can be usefully accommodated in a model is significant. The stochastic nature of the GA means that results normally vary between genetic experiments, and therefore require averaged data or repeated trials in order to show confidence in results, robustness according to the start condition and to be able to draw effective conclusions. Reference is often made to the best design chromosome produced during a particular generation or run of the GA, in terms of 'best-of-generation' and 'best-of-run' results.

In the same way as KBESs had narrow fields of application, certain refined GA techniques are sometimes only valid for a subset of problems. Packing problems (like the Knapsack Problem[4]) and job scheduling and transportation problems (like the Travelling Salesman Problem) are well known examples of generic types of optimisation problem that have been studied using specially modified GAs. Some of these problems have analogies in various engineering activities. The literature has recorded that modification to the way in which these problems are formulated, and refinements to their reproductive operators, have improved their success, see for example Suh et al [70], 1986. However, the benefits are usually problem-specific. The creation of robust, general-purpose techniques is an ongoing challenge. Goldberg [71] first commented on the issue of efficiency versus efficacy with the GA in 1986. Later, in 1991, Davis [64] stated that: -

"a robust general purpose approach and a specialized refined approach are mutually exclusive."

Davis [64] described many application-specific refinements that were capable of enhancing search. Whilst the complexity of methods continuously increases, modifications sometimes, though not always, draw on analogies in natural systems for inspiration. Research includes new techniques for supporting a more general-purpose approach. Some modifications to the Simple GA are appropriate in specific circumstances. Other modifications have more general application. This chapter continues by introducing some GA operators and refinements that were shown by researchers to improve the effectiveness of the Simple GA, and that are now widely employed. This is then followed by a summary of relevant work, principally selected from the structural / conceptual design domain, which use some of these modifications. This summary of past studies aims to demonstrate the variety of techniques that have been previously applied that are relevant to the current study.

3.5 Variations in GA Processes

It is usual for the control parameters used by GAs, that include the population size, the length of chromosomes, and the incidence of particular operators to be preset, but they can be permitted to vary during execution. Dynamic control parameters can be programmed that change in a predetermined manner or that respond adaptively to the progress made. For example, crossover and mutation rates have been programmed to increase or decrease at a fixed rate upon reaching a certain generation. Alternatively, the population size may be permitted to grow or to shrink as chromosomes converge towards optimal solutions.

---

[4] also known as the Bin Packing Problem.

For brevity, the terms *ChromoLen*, *PopSize*, *NoGens*, *Prob$_{Cross}$*, and *Prob$_{Mut}$* are used hereafter to indicate binary chromosome length (measured in bits), population size, number of generations (prior to termination), probability of crossover, and probability of mutation parameters, respectively.

3.5.1   Representation and Initialisation

Morphogenesis is a term given to the relationship between a chromosome genome and the decoded phenotype. It is significant in determining the effectiveness of the search process, and has prompted study of different ways to represent problems. One aspect of problem formulation is the encoding scheme used. Holland [62], and Goldberg [63] espoused the Schema Theorem, based on the notion that highly fit, low-order (short-length) building blocks (bit strings) are responsible for driving improvement. For this reason, Holland [62] and Goldberg [63] have both used binary chromosomes in their GA studies. Davis [64] and others used real number encoding and reported that it performed better in tests. Real number encoding requires the use a number system other than base-two (usually decimal). Real number schemes have advantages over binary schemes in certain circumstances. Dynamic length chromosomes have also been investigated and have enabled course-to-fine or fine-to-course design progressiont – for example, by Jenkins [72] in 1994.

In order to implement a chromosome, a data type is used to store the contents of each gene. Real-number encoding permits genes to be stored as decimal values. A binary gene can also be handled concisely as an integer value although the form (or, structure) of a binary gene is sometimes recorded literally as a character string. For example, using a binary scheme, a gene comprised of four bits (four alleles) can take the form: 0101, which treated as a binary number, is equivalent to the integer number five. In referring to binary-encoded chromosomes, the terms *the form of a gene* and *the gene value* are synonymous and are used interchangeably in this thesis. In certain circumstances, bits with values *one* are sometimes referred to as being *on* or *active*. Conversely, bits with value *zero* are sometimes referred to as being *off* or *inactive*.

PopSize is one of many control parameters that may be adjusted to suit a genetic experiment. As mentioned previously, the starting population is normally chosen at random or pseudo-randomly, but it may also be seeded with particular solutions chosen for their suitability or diversity. Davis [64] refers to the latter process as *interdigitation*.

3.5.2 Crossover and Mutation

Crossover and mutation operators are applied during reproduction when the corresponding probability is greater or less than a randomly chosen real number in the range from zero to one. The crossover rate, $Prob_{Cross}$, is typically chosen in the range 0.6 to 0.8, thus affecting 60-80% of the population during a generation, on average. The mutation rate, $Prob_{Mut}$, is typically chosen in the range of 0.001 to 0.05, affecting 0.1-5.0% of the population on average. These rates can be adjusted dynamically, but are commonly fixed. Davis [64] reported using crossover and mutation operator weightings that change non-linearly during experiments.

Goldberg [63] introduced a single-point crossover mechanism. De Jong introduced two-point crossover, which was shown to improve convergence almost universally and the idea was quickly extended to general multi-point crossover (see Davis [64]). Jenkins [72] was amongst those to have reported using multi-point crossover in structural design applications with success. Syswerda developed a technique called uniform crossover where a template was applied to the parent chromosomes to determine combination of bits to be swapped (also in Davis [64]). Using uniform crossover, the rate of mutation can be set so as to maximize the chances that each parent will pass on exactly half of their genetic material to a joint offspring. In 1997, Camp et al [73] distinguished fixed crossover and flexible crossover schemes. The former method is unusual in that it reuses the same crossover locations for an entire group of chromosomes, whereas the latter method, which is conventional, determines a new crossover site each time crossover is applied.

3.5.3 Fitness Scaling

As the population evolves through successive generations, differences between solutions tend to become smaller and normalisation is an optional refinement that is commonly employed to amplify minor differences for the purpose of making selection pressures more effective. Normalisation is also important for ensuring that an overly-fit design does not dominate the selection process. Alternative methods include using a window that bounds acceptable minima and maxima values, linear fitness scaling and linear normalisation (where fitness is normalised to a datum value e.g. 100). Linear fitness scaling is described in Goldberg [63] and linear normalisation is described in Davis [64]. The latter technique applies fitness scaling in such a manner that the maximum scaled fitness would typically become at most twice the average scaled fitness in a population, and the minimum scaled fitness would never be less than zero. Researchers including Grierson et al [74] have also

studied rank-based proportionate fitness to help maintain a diverse population of good designs that satisfy multiple criteria, simultaneously.

Since the GA is a maximising algorithm, the task of minimisation - as required for least-weight / least-cost design problems - can be achieved using an inverse fitness approach. This can be achieved by subtracting the fitness function value from an arbitrary, large, positive constant or by using the reciprocal of the original fitness value. For least-weight design optimisation problems, where the largest value can be easily calculated based upon the heaviest (most unsatisfactory) conceivable structure, then that specific value can be used in place of an arbitrary constant.

3.5.4 Selection, Reproduction and Replacement

Common selection methods used to select parent chromosome from a general population into a mating pool include the Weighted Roulette Wheel (WRW) method and the Remainder Stochastic Sampling (RSS) method, described in Goldberg [63]. The latter is generally better at maintaining high fitness solution between successive generations in a fair manner, according to the distribution of fitness values within a given population. Other selection strategies include those that determine selection probabilities from interpolation, curve-fitting or step functions. Step functions can be chosen to be reasonably uniform in order to encourage diversity, or biased in favour of high-fitness solutions. Parameters used by these techniques can also be altered, dynamically.

Replacement of parent chromosomes in the population by offspring can be automatic, or may optionally be determined according to specific criteria. Tournament Preselection is one such method that is popular. It requires an offspring to improve upon the fitness of its parents in order to survive into the general population of the next generation; if it fails to do so, a copy of the parent chromosome is retained instead. Variations on this technique include retaining an offspring chromosome if it exceeds a prescribed minimum fitness value or if it comes close to the current maximum fitness solution. Minimum and maximum limits can also be applied and adjusted dynamically.

The Elitism selection strategy is a technique commonly employed to ensure that the best chromosome in the entire population always survives into the next generation; where necessary it automatically replaces a randomly chosen or poor fitness individual. Davis[64] also described Steady State Reproduction, first studied by Whitely, where n offspring are created and replace n parents, which is similar to the Elitism technique. Here,

the replacement process can be manipulated to remove the worst chromosome(s) from the population, or to reduce duplication.

3.5.5 Convergence Criteria and Threshold

The most common stopping conditions involve either completing a specified number of generations, reaching a required fitness threshold or terminating when the population is observed to have converged to a point where the return on further execution diminishes rapidly. Testing for convergence involves examining the chromosomes that make up the population for similarity during the processing cycle. Alternatively, execution may be halted after a given number of generations have passed that continue to show no improvement.

Premature convergence and loss of diversity can be reduced in several ways. These include: applying clustering and nicheing techniques with a single population, artificially injecting new genetic material into the population and maintaining several GA populations, concurrently. These ideas featured in the GA-ANT algorithm (described in Parmee [67], in 1996) and parallel GA implementations, in which inter-population migration between multiple populations was managed.

3.5.6 Specialized Techniques

Many hybrid approaches that use the GA have been developed. Extraneous domain knowledge has been used to determine or to improve the fitness of a chromosome, and to influence selection procedures. Classical hill-climbing optimisation has been used to assist convergence in the vicinity of optima. Meta-level GAs that optimise the control parameters in lower-level GAs, and GAs that monitor and adaptively adjust their own control parameters after major and minor intervals have also been examined. Inversion, sharing, noisy GAs and messy GAs have also been developed. Some of these techniques are presented in Goldberg [63] and Davis [64]. The latter have been used to investigate problems in which the phenotype is non-static. They have also been applied in situations where chromosomes are required to shrink or to grow in order to accommodate a changing number of design variables. Messy GAs use modified operators to cut and splice string segments. Leithe et al [75] presented some modified genetic operators that were considered suitable for structural design, in 1995. Hybrid techniques that combine GAs with ANNs and other techniques are also commonly encountered.

3.6 Engineering Design Applications

3.6.1 Introduction

Investigation that involved detailed structural design problems generated interest in the application of the GA to conceptual design problems. This section introduces various applications, most of which appear in chronological order. The very first studies, and particularly those involving fixed trusses, represent problems that are clearly in the realm of detailed design. Gradually, problems of a more conceptual nature have followed.

As an aside, in 1993 - the era of the KBES - Reddy et al [32] created a tool for assisting with the design of individual reinforced concrete (RC) components.[5] The system was able to detail RC beams and slabs according to fixed requirements set out in relevant COPs. The system was called EXFORM. In the system, appropriately sized reinforcing bars were automatically chosen and arranged according to the overall dimensions and the amount of reinforcement required in the section. Reddy et al [32] described EXFORM as a tool for conceptual design and justified themselves by asserting that at component level, at least, design variations constitute different concepts.[6] Clearly, however, the consideration of individual members at this scale is an activity for an advanced stage of design.

3.6.2 Detailed Design Applications - Trusses and Frames

In 1992, Dunsmore [76] applied a GA to a classical three-bar truss optimisation problem (presented in Schmit [77], 1960). Here, a least-weight solution was sought that could satisfy certain stress constraints for various load conditions. The weight of the structure varied in relation to the cross-sectional areas of the three members, which were permitted to each take a discrete value in a single prescribed range. Nodes in the structure could not be moved and so the geometry was fixed. However, bar areas were permitted to become zero-value and this not only had the effect of producing variations in the stress in each member, but also enabled the topology of the entire structure to be altered. During the experiments, the statically indeterminate three-bar truss was able to evolve into a determinate two-bar structure, a single-bar mechanism, or a non-entity. Though variety was limited, again, for the task in hand it was noted that these independent structures represent different design concepts in their own right. Whilst one particular concept may

---

[5] This system is not unlike commercial design standards-processing software used for detailing.
[6] The term conceptual design has since adopted a more specific meaning involving holistic design. Although Reddy may be justified the reader is reminded that the priority during conceptual design is to identify beneficial high-level options having the greatest influence on cost and functionality.

yield the global optimum, viable alternatives may often exist that satisfy the load condition.

The three-bar truss represents one of the simplest structures to model and analyse. Practical structures are often considerably more complex. Optimisation problems involving truss and frame structures are relatively easy to define using variables that represent nodal positions and member dimensions, but can also easily be expanded and developed into highly dimensional problems for study. For these combined reasons there have been numerous investigations of similar structures.

Amongst the earliest applications based on biological principles were those used to optimise aircraft design. From the mid-1960's, Rechenberg [60,61] applied an optimisation technique that determined the optimum configuration of a steel plate, hinged in five sections, for minimum drag. Other early proponents of the GA considered structural design applications included Goldberg et al [78], in 1987. Schwefel [79] investigated an 18-bar truss optimisation and successfully showed that a GA could achieve a design solution having a weight only 5% short of the optimum solution, noting that a convex linear programming technique had previously only achieved an 6%-from-optimum solution, in 1989.

In 1991, Jenkins [80,81] applied the GA to structural design optimisation problems and inspired Dunsmore [76]. Jenkins pioneered the study of various other multi-member structures. These included a trussed beam and a thin-walled section. Shortly thereafter, in 1992, Rajeev et al [82] also examined a classical three-bar truss problem, followed by ten-bar, and 25-bar planar trusses, using the GA. In all cases, the prime objective was to obtain minimum-weight solutions, capable of withstanding the prescribed loading. More complex structures require more complex representations and present a greater challenge to the GA. Cai et al [83] and Jenkins [72] have studied theoretical aspects for handling very long chromosome structures efficiently, required for modelling complex structures, in 1994-5.

3.6.3 Conceptual Design of Trusses and Frames

Up to this point, the research that has been mentioned has minimal relevance to conceptual design. However, the study of trusses by Jenkins [80,81] in 1991 encouraged Grierson et al [84] to attempt simultaneous sizing, topological and geometrical optimisation problems, not possible with conventional mathematical programming techniques. In 1993, Grierson

et al [84] approached the combined optimisation problem of planar frames, by first taking a simple example and later extending the approach to a more detailed skeletal structure resembling a building in section. It was noted that before the GA, studies were restricted to fixed-layout optimisation, where member size was variable but where geometry and topology aspects were fixed. In one example, the GA was applied to optimize the design of frames in which not only was member variation permitted (according to prescribed standard sections), but also geometrical variation (by varying the length of one support member) and topological variation (through the presence or absence of a secondary support). This design problem required a binary string of total length 13 bits.

Rajan [69] conducted similar investigations using trusses instead of frames later in 1995. He also initially restricted the problem definition to a manageable number of elements. In addition to member size variation, Rajan studied techniques that simulated the removal of non-essential bracing members from a truss. Positions of nodes in the truss were also allowed to change. The effect that that such changes could have upon structural performance was investigated. A change in the connection arrangement of members constituted topological variation whereas nodal movement effected a geometrical or shape change.

Both studies constituted conceptual design. Rajan [69] employed a technique that involved creating artificially redundant structures, and controlled the presence or absence of members using single bit genes, that acted as Boolean decision variables. Interestingly, Grierson et al [84] used a similar technique in his first, simple example but later, in a more complicated case, he reverted to the same representation method as used by Dunsmore[76]. That is, a section was represented by one of *n* discrete values, of which *(n-1)* represented standard component sizes and the one remaining option represented a null section with zero-capacity, simulating the removal of that member from the corresponding design solution.

Rajan [69] and others researchers applied a technique known as *variable linking* to limit the required length of the chromosome in an effective manner. This technique used communal genes, having a discrete value range, to represent an attribute value for a group of similar members. This process was found suitable for investigations of triangular and trapezoidal trusses, in which diagonal struts, horizontal members in compression and tension, and uprights shared common genes. The length of the chromosome required was

kept in proportion to the number of different member groups and the number of discrete member sizes required within each group.

Rajan [69] also investigated shape optimisation using variables that represented the deviation of a node along an axis from a datum position. In so doing, the number of members in a structure varied according to whether or not nodes converged. Combined sizing, shape and topology optimisation was performed successfully using different segments of the chromosome to represent each aspect, in a similar manner to Grierson et al [84]. Rajan [69] discovered that an effective approach was to introduce the sizing, topology and geometric variation incrementally, carrying forward the best result from the previous stage of optimisation to seed the next, more complex task.

Grierson et al [84] applied an approximation technique to analyse the frames generated by the GA in order to significantly reduce the computational effort required.[7] Jenkins [72] formally reported the need to investigate re-analysis techniques to improve the computational efficiency in structural analysis procedures. Experiments by Grierson et al [84] produced unsymmetrical solutions for unsymmetrical load conditions. Jenkins [81] advocated analysing the final state of constraint satisfaction for optimal design solutions produced using the GA. In one example presented by Grierson et al [84], it was noticeable that while certain variables took up an intermediate value in their permissible range, whilst others, such as those that represented the height of column members in a plane frame, naturally tended to a lower bound value.

There are two further points worthy of note from the aforementioned studies. With respect to control parameters, Grierson et al [84] applied an unusually high crossover rate (90%) together with two-point crossover to produce dramatic evolutionary change. Also, a penalty approach was introduced to compensate against constraint violation, having first been carefully chosen to penalize minor and serious constraint violations, appropriately. Grierson et al [84] recognized that penalty terms were somewhat approximate; Jenkins [81] noted that penalty functions were significant in influencing the success of the optimisation process and should be chosen with great care.

In 1995, Sandgren [50] also studied cross-sectional, geometric and topological change in a ten-bar truss and in 1997, Sugimoto et al [85] investigated fully-stressed design of an equivalent fixed-topology, triangular-framed structure, under combination loading. Both

studies acknowledged the need to comply with performance factors, or serviceability constraints, in real structures in addition to the load condition. Serviceability constraints relate to the rigidity of the structure in relation to the extent of deformation and displacement of members and frequency response in vibration.

The survival-of-the-fittest paradigm is less effective when a large proportion of offspring are found to be infeasible (i.e. have null fitness). Evaluation of the entire design population and rejection of infeasible design is computationally costly. Both Sandgren [50] and Sugimoto et al [85] used finite element analysis (FEA) during the evaluation of the fitness of their structural models. The analysis of the each new population was computationally expensive. For this reason, both parties applied extraneous heuristic information to achieve better constraint satisfaction in solutions. It was found that a high proportion of viable solutions could be maintained in the population using such techniques. Sandgren [50] corrected infeasible structures using a similar principle to that employed by the automatic structural concept generator module mentioned in section 2.3.3.4, adding minimal 'dummy' members where necessary.

Sandgren [50] and Sugimoto et al [85] separately showed the GA to be capable of improving upon optimisation methods that use continuous variables where practice required a discrete value solution. The study by Sugimoto et al [85] also produced a familiar tradeoff when attempting to tackle the multi-criteria problem of minimizing nodal displacements and achieving least-weight design simultaneously, and demonstrated that a distribution of feasible designs can be generated.[8] Sandgren used a goal programming formulation that gave relative priority to a number of design objectives that were to be satisfied. He noted that: -

"the formulation of a goal programming problem contains no exact counterpart to the objective function in a nonlinear programming formulation."

Sandgren [50] modelled certain soft constraints, such as those relating to deflection, using variables that represented deviation from an ideal value. Nodes within truss structures were given freedom in space or along a surface, according to whether or not they connected to ground, and this provided greater variation than the similar, but earlier study by Rajan [69].

---

[7] The method was called the binomial-expansion reduced-basis technique, applied to the stiffness method of analysis.

Sandgren [50] and Rajan [69] independently showed that if topological variation is permissible in the design of a skeletal structure (for example, a crane jib or transmission tower) it can have a very significant effect upon the efficiency of the resulting structural system, compared with a fixed-topology truss.

3.6.4 Component Design Applications

In 1994, Kousmousis et al [86] presented GA applications in which the practicality of solutions were of paramount concern. Their research was amongst the first to combine construction factors as direct objectives of the optimisation. A practical steel roof truss arrangement appropriate for industrial / warehouse usage was the goal in a pioneering combinatorial, mixed layout and sizing optimisation problem. A GA was used to generate a layout and a rule-based PROLOG program was used for member sizing. This again represented a multi-criteria (MCOP) optimisation problem. Objectives were weighted to simulate a desirable level of compromise between separate criteria representing minimum-weight design and buildability. The aim was not necessarily to seek the global optimal solution, but rather to achieve a highly satisfactory near-optimal solution, using an efficient search procedure.

Success led Koumousis et al [87] to address the detailed design of a two-span continuous RC beam using the same approach, also in 1994. Here the optimal arrangement of reinforcement within the section was based upon criteria representing the threefold aim: - to use the fewest total number of reinforcing bars as possible, to use the fewest different bar sizes, and to provide the minimum amount of reinforcement necessary to comply with design requirements. Layouts were configured to comply with the dimensions particular to a given section. Kousmousis et al [87] reported the system was efficient at searching the large design space (which contained over 16 million potential combinations) to detail a RC beam, and said that it represented a more viable alternative to rule–processing systems applied previously to similar tasks.

In 1995, Rafiq [88] applied similar rationale to the reinforcement detailing of RC columns in bi-axial bending. Again, the goal of the optimisation was to use the fewest number of different sized steel reinforcing bars distributed in an optimal fashion. A difference here, however, was that both layout and sizing activities were incorporated directly into the optimisation algorithm. The objective function used normalized values, representing the

---

[8] A Pareto-optimal set of solutions can be found.

degree of satisfaction of each individual design criteria, to give each equal consideration. A real number scheme was also employed and was found to be effective.

In 1996, a study by Lucas et al [89] was reported that also followed the work of Kousmousis et al [86,87] in component design. This study concerned the design of rectangular RC beams. A constraint-based GA technique that combined evolutionary search with constraint satisfaction programming in Common Lisp was employed. This study went further towards considering various detailed design performance constraints such as those related to flexure and shear than its predecessors, and factors associated with buildability. Doing so, created a design problem that was over-specified (or, over-constrained), initially. The problem was resolved by using a penalty method that reflected the relative preference on the part of the designer for modifying particular independent variables in relation to others, in order to achieve constraint satisfaction. In the design model it was highly undesirable to have to modify hard constraint variables, relative to variables associated with soft constraints.

Lucas et al [89] applied the GA to search for solutions that yielded the least serious constraint violations and showed them to the user. Presented with a list of the constraint violations, the user was then required to manually relax constraints, as given in the objective function and used for fitness evaluation. Notably, the fitness function expression was modifiable through being implemented as an external procedure in Lisp, rather than being hard-coded into the GA optimisation routine. The technique was shown to be capable of producing satisfactory designs by interactive, piecewise refinement. The scope of the technique was thought to be limited since preference levels were established subjectively and it was doubted whether, using only relative measures of fitness, the method could easily be extended to handle many more criteria in a satisfactory manner.

3.6.5 Conceptual Design Aspects

In section 2.3.3.1, it was mentioned that the very first KBESs were limited to the role of recommending materials and components. Later systems were developed for selecting and combining independent and compatible elements in an autonomous manner in order to create more complex, and therein, more realistic (useful) concepts. Studies mentioned earlier permitted variation in topology, geometry and member sizing.

It is important to recognize that the value of the techniques described is itself derived from the freedom afforded to the designer - Bedford [90] said EAS techniques were significant

41

because they do not "stifle the creative process." The capability of the GA to describe and manipulate problems in a broad way has warranted the present investigation into its applicability in support of CBD; in the present context, the goal is to help designers appraise the relative merits of different design concepts, comprising of different materials and subsystems and taking various forms.

In 1993, Parmee et al [91] described how the GA had been successfully applied to the design of a hydropower scheme, which incorporated the optimal geometric design of a concrete arch dam to resist self-weight and hydrostatic forces. In this work, Parmee et al recognized that radically different geometric forms (i.e. different curvatures) embodied different design concepts that warranted consideration. Parmee et al proposed that an optimisation based on the Structured Genetic Algorithm (SGA), by Dasgupta et al [1] was appropriate for handling multiple different concepts simultaneously. The SGA is a variation on the Simple GA (Goldberg [63]) and was intended to enable mutually exclusive sets of design variables to be maintained simultaneously, for multi-state electronics applications. It is used in the present study to enable alternative design components to be represented in a single chromosome structure, and described in detail later.

In 1997, Furuta et al [92] applied the GA to aesthetic cable-stayed bridge design. This study had a more artisitic, less scientific inclination. It was a good example of many studies that required the superiority of each chromosome to be evaluated subjectively - in other words, according to user preference. A small number of concepts were generated, and a user was required to rate those with highest aesthetic appeal. In another part of the same study, an attempt was made to use an ANN to try to learn the relationship between synthesized concepts and their corresponding subjective aesthetic quality. Though the conclusions of the work were limited because of the subjective nature of the study, one aspect of this application was particularly interesting. Independent design options, such as the shape of the main bridge towers, the shape of the deck girder and the amount of cabling were discretely incorporated by mapping each alternative to different binary values.

3.6.6 Miscellaneous Studies

Hills et al [93] have applied SA to aspects of conceptual design, in 1994, in a similar manner to applications that used the GA. The SA algorithm imitates the process of annealing in metals - as they are cooled, the internal energy is minimized and an equilibrium state is achieved. An important difference, however, between the SA algorithm and the GA is that the SA technique does not maintain a population of multiple

design solutions (possibly containing different concepts) consistent in size between subsequent iteration cycles, whereas a GA does.

Hudson et al [94] presented ideas for simulating creative design from constituent parts using GAs. In the field of mechanical engineering, Mfinenga et al [95] used a GA to build a mechanical system by combining appropriate components from available sets. For example, power supplies and independent systems for translating forces were amongst the basic constituent parts.

Other applications of GAs, which are not directly related to the current study include optimisation of welded structures, in 1990 (see Deb [96]), and grillage structures in 1993 (by Hajela et al [97]) and an urban planning excercise, in 1999 (see Balling et al [98]).

3.6.7 Recent Developments

From 1997 to the present day, there has been noticeable growth in interest in the application of GAs specifically to CBD. Studies by Grierson et al [74,99], Rafiq et al [58,100] and Sisk et al [101] have particular relevance to the present study. In 1997, Grierson et al [74] considered topological building design using a GA, which will be discussed in further detail. In 1999, Grierson et al [100] applied the GA to generate architectural variations in building form. Also in 1999, Sisk et al [101] have applied considerable experience in creating bridge design software tools to aspects of CBD, and Rafiq et al [58,100] introduces hybrid GA-ANN approaches. All of these studies cover many issues related to the author's research; indeed, acknowledgment is gratefully received where made by others to the authors own work.

These studies are re-introduced at appropriate points, later in discussion.

4 General Issues for Design Modelling

4.1 Introduction

Chapter 2 introduced the building design domain and presented an overview of CAD software. KBESs were discussed at length, having been used with limited success in many former studies involving CBD. Chapter 3 introduced the GA. Various applications within both the structural design domain and the conceptual design domain were mentioned, leading up to the present research. This chapter introduces issues related to modelling the CBD domain in order to provide appropriate DS. In the following chapters, the implementation of the GA is explored in detail.

4.2 Representational Issues

Previously, computational DS has investigated various models and methods for representing design knowledge, generating new information, transferring data between processes and communicating with the designer. Processes concerned with the generation of new information (which, for this study, refer to those based upon the GA) rely upon information representation and information processing techniques.

Domain knowledge is a prerequisite for defining a search space and for developing, analysing and appraising the suitability of, different design solutions. In a GA-based approach, optimisation is unconstrained, which means that infeasible regions of a design domain are not precluded directly by the constraints that are applied. Instead, constraint satisfaction is an implicit part of the fitness evaluation and reproductive cycle. Success in applying the GA is heavily dependent upon the appropriate formulation of design variables, objectives and constraints, as well as the use of techniques to discourage or improve poor quality designs, as necessary.

Information representation and information processing are associated with the creation of structured design models and with the transformation of raw data into high-level knowledge resources, that may become an asset to designers. The variety of sources of raw data supporting CBD is vast. These sources include general design guides, specific COPs, literature describing proprietary design and construction techniques, and heuristic information elicited directly from practicing designers. It is important to consider: -

- how designers may benefit most from DS,
- how user interaction can be accommodated in an effective manner, and

44

- how design processes can be modelled in ways that offers flexibility and accommodate sufficient detail in order to offer a useful level of support.

4.3 Aspirations of Decision Support

As mentioned in section 2.3, design methods have evolved with construction practice and IT has become firmly established within detailed design for CAd, structural analysis and standards-processing tasks, amongst others. By supporting powerful techniques such as FEA, computers have contributed to the design of impressive, radical structural forms within financial and time limits like the Swiss Re building with its curvaceous frame (described by Mylius [102]) as well as other, more contemporary structures.[1]

Concessions are usually necessary in certain aspects of design to realize other benefits. Proponents of integrated design tools are keen to help designers express greater creativity when searching for an acceptable compromise in numerous design aspects. These aspects include structural efficiency, aesthetics and comfort for occupants[2], energy efficiency and environmental matters, amongst others. By supporting well-informed decision-making and enabling alternative structural forms to be more easily investigated, software tools have the opportunity to bring educational and commercial benefits.

Throughout modern history, economic constraints, buildability considerations and other factors have forced designers to be resourceful and to seek the most practical and efficient solutions that were available at the time. In the light of new technology, Moore [10] warned the modern-day building designer not to disregard these traditional intentions and be tempted to: -

"maximize architectural magnificence at the expense of other technological considerations."

Other individuals are cautious about the effect design software might have upon design practice. One notable concern is that an increase in automation might undermine the intuition that a designer has traditionally gained from applying sound principles first-hand.[3] For these reasons, the present work has purposely sought to apply techniques that emulate (and where possible, enhance) conventional design wisdom, based on well-founded

---

[1] Novelty can have unforseen disadvantages. For example, extensive glazing and sweeping curves may produce dramatic designs with heat loss / heat gain problems.
[2] In the context of satisfying the intended function of the building.
[3] This impacts design education in requiring the acquisition new skills.

45

motives for practicality, efficiency, material economy and buildability. This chapter introduces factors concerned with the formulation of suitable design models.

4.4 Data-Oriented Systems

Since design embraces many different processes, it has encouraged specialized research into highly integrated data-oriented systems, capable of encapsulating all the relevant information generated during design in a consistent manner. Collaborative research projects have attempted to extend CAd software to incorporate unified data structures and product models, supporting the flexible manipulation of design information by different disciplines. A number of major initiatives have been reported which have studied models for integrated product data management. For example, the Eureka CIMsteel initiative was established to improve the effectiveness and competitiveness of the construction steelwork industry. Using open system computing, the CIMsteel project aims to integrate design, detailing and fabrication activities, amongst other benefits. Another major collaborative project that included energy-efficient design as one of its aims was called COMBINE[4], described by Augenbroe [103]. Thorpe et al [104] described the noteworthy and large-scale ISO-STEP initiative[5], which has as its focus the creation of an open standard for data integration. The ISO-STEP project has sought to create a neutral data exchange format suitable for the communication of engineering design data, and based on the proprietary DXF[6] protocol of a popular draughting tool called AutoCAD®.[7] Progress has been gradual, due to the size of the task and the extent of non-standard design variations found in real structures. Nevertheless, whilst some semantic difficulties remain, partial integration has been achieved and enables design models to now be transferred between STEP compliant applications, including AutoCAD® and FEA applications. This capability is described by Leal [105]. The STEP protocol is used in CIMsteel and COMBINE.

Compatibility with CAd systems is increasingly common in software for integrative design. The present research has investigated general multidisciplinary tasks that stand to benefit from DS whilst research into unified data models progresses independently. Compatibility with a suitable unified product model would be likely to offer greater creative freedom to designers, once complete.

---

[4] Computer Models for the Building Industry in Europe.
[5] International Standards Organisation - STandard for the Exchange of Product data.
[6] DXF stands for Data eXchange Format. The original DXF protocol was designed to handle dimensional and textual information in technical drawing layers.
[7] AutoCAD® is produced by Autodesk Ltd., Cross Lanes, Guilford, Surrey, GU1 1UJ.

4.5   Target Applications

Support for CBD has been easier to provide in domains where one discipline has justly exerted a greater influence on the design than the rest. It has also been easier where the design brief has stipulated the presence of certain features, either explicitly or implicitly. Previously, in section 2.3.3.2, it was mentioned that buildings such as skyscrapers and industrial plants necessitated structural engineers to lead the design process. For other types of building, architects have been called upon to produce design solutions that emphasize qualities such as prestige, originality and comfort. In such circumstances, structural integrity is no less important, yet becomes subservient to the overall function of the building. Hence, structural engineers have often been required to produce a structural design solution consistent with architectural intentions, and in which the architect is in charge of producing a general arrangement. Examples include hotel, retail and exhibition buildings.

Low-rise and medium-rise office and commercial buildings were chosen as the focus for this study as they are not influenced predominantly by specific considerations relating to appearance, structural design or occupancy type, but instead by a combination of factors. The scope for variation in different design aspects encourages the generation of alternative, viable concepts with implications to cost, buildability and in other areas. It was envisaged that techniques employed in the first instance to office structures might not be limited only to these types of building, but might instead reveal broader application later, during the course of research.

4.6   Target Platforms

Techniques that have been investigated were intended to be relevant to designers given the present state of IT. The computational techniques described in this thesis were implemented in the C++ programming language[8] using the Microsoft[9] (MS) Visual C++ v.5 development environment, for MS Windows 98 / NT4 operating systems (OS). All of the computational experiments were performed on an IBM PC-compatible microcomputer[10] (PC) equipped with an Intel Pentium 200MHz processor and 32MB of random access memory. Surveys conducted into the use of IT in engineering and

---

[8] C++ is an extension of the general-purpose programming language C.
[9] Microsoft Corporation, One Microsoft Way, Redmond, Washington, USA.
[10] International Business Machines Corporation, P.O. Box 12195, Research Triangle Park, NC 27709.

architectural practice would suggest that at the present time, this hardware specification is typical of the kind of computing resource available to designers.[11]

4.7 Design Criteria and Constraints

This section considers factors that commonly influence design decisions. Availability of resources such as construction plant, materials and labour fluctuates temporally and according to geographic location. The skills set of different design teams also varies between projects. Furthermore, construction projects generally operate to narrow profit margins and often seek solutions that permit earliest possible completion. Where a number of practical solutions may exist, those that are uncomplicated and inexpensive for design, construction, in-service use, and maintenance are generally preferred and sought after. In the absence of suitable DS tools supporting concept generation and appraisal, designers continue to rely heavily on practical experience in order to identify benficial design concepts.

In various circumstances, statutory design requirements necessitate, encourage, discourage or prohibit the use of certain materials and methods of construction through COPs, often with safety concerns paramount.[12] BRs, cultural factors and environmental issues impose additional constraints. Statutory BRs govern design aspects such as safe evacuation in case of fire, apply almost universally to all modern buildings. Provision of natural lighting and use of energy-efficient materials are two examples of specific architectural intentions that have become more important design criteria for certain projects in recent times. For example, the Canon UK Headquarters building, in Reigate, Surrey was designed to minimize waste during construction and to make low energy demands during use. The consultants responsible for the structural design, Curtins Consulting Engineers, were commended during the IStructE Structural Awards 2000 for their achievement (see Stansfield [106]).

Foundation strength and zoning regulations vary according to site location and restrict the overall geometry that a structure can assume given existing construction technology, and this includes the maximum permissible height for building. Poor site access and shortage of land to build upon can present difficulties in some design situations. Restricted sites are often associated with high land costs that must be met by the client or occupier, and offset through the perceived utility of the building. Given good foundations and favourable building legislation, shortage of space and high land costs act as catalysts for buildings to

---

[11] For a survey of the application of IT in design practice to 1997 see Najafi [46].

extend upwardly. Consideration of serviceability criteria, such as those relating to wind and earthquake loading, become more significant for taller structures.

Internally, buildings are configured according to functional needs and functional preferences, and for a number of reasons most are organized in a regular manner. Architects normally divide a building into functional units using an *architectural planning module*, also known as an *architectural grid*. The architectural grid is largely determined by functional considerations. In offices and residential developments, the dimensions and locations of architectural features have been noted to influence the architectural grid. In warehouses, supermarkets and libraries, the width of shelving and aisles often influences grid dimensions. Certain architectural components have obvious importance such as windows and cladding. Other, more superficial features that affect the choice of an architectural grid include ceiling tiles and fluorescent lighting.

In a similar way, structural engineers use a *structural grid* to plan the layout of structural systems. The structural grid closely follows the architectural grid, and is determined by the function of the space that it encloses and by factors that are associated with economic construction. For office blocks with basement-level car parking, the architectural and structural grids must accommodate vehicle dimensions. In various types of building, certain functional space is used to create foyers, lobbies, hallways and meeting rooms, and requires relatively column-free zones. Usually space intended for general-purpose office accommodation tends to be more lenient towards variations in the structural layout. Large, open-plan floor areas provide maximum versatility and generally command higher rents. There is often a trade-off between the importance of having column-free areas and keeping structural costs to a minimum, and unless office buildings involve heavy load transfer, a grid spacing of 6m or more is typical for economic construction.

4.8   Structural Design Rationale

Structural engineering design is not a precise science. Individual structural components are connected to one another and to non-structural elements to form a continuum structure. The COPs which make up the statutory guidelines for good design have been largely developed through empirical study and are conservative, and analyses are approximate. COPs include safety margins, and design calculations allow for a degree of customization, to satisfy different requirements. Because of practical limitations and tolerances imposed

---

[12] For example, grade and composition of RC and steel yield strength.

through component manufacture, transportation, buildability and other factors, structures cannot usually be built in a totally optimal manner.

Safety within structural engineering extends to safe construction and maintenance techniques. Using partial safety factors, structural systems of buildings are purposely designed with reserve capacity. Buildings designed and built using these factors, and maintained properly, should be capable of satisfying structural needs for their intended service life. Safety factors are higher in situations where structural damage is likely to have severe consequences. The COPs cannot guarantee that structures will never fail although failure should always occur *in service* and therein provide warning well in advance of ultimate collapse.[13]

Section 3.6 previously introduced GA research in the structural domain that addressed the detailed design of individual frames, trusses, beams and columns. Where design was based upon RC components, design models were developed that not only complied with the requirements set out in COPs, but which also took account of various buildability considerations. The generation of diverse building design concepts similarly requires compliance with statutory requirements (both COPs and BRs) and awareness of construction practice, but at a broad level. Estimated component sizes, quantities of materials and costs are normally presented in the form of a Bill of Quantities (BoQ).

The RC Manual was mentioned in section 2.2.2 as containing simplifications of full RC design methods based on reasonable assumptions. A companion publication called The IStructE Manual for the Design of Steel Structures [107], (henceforth, 'Steel Manual') provides similar advice for steelwork designers. These guides were developed with the intention of helping designers select appropriate structural elements rapidly by providing conservative estimates of certain design calculations in order to satisfy loading and serviceability requirements. Other guides also offer similar relevant advice for other materials and specific structural systems, including RC foundations (see Reynolds [108]), aspects of steelwork design (see SCI [109]), aspects of cladding (see SETO [110]) and economic concrete frame design (see Goodchild [111]). The kind of advice offered is highly suitable for developing conceptual designs and is used later within the present study.

---

[13] Symptoms of in service failure include excessive deflection, visible cracking etc...

4.9 Previous Cost Studies

The structural designer is often required to choose between concrete or steel as the main structural material for a building, as these are the most economic and popular materials and most practiced structural design techniques. Goodchild [112], mentioned above, undertook a Cost Model Study on behalf of the RCC to provide true comparison of similar buildings that use in situ RC and steel frames. British Steel Corporation (BSC) led a similar collaborative study entitled Steel or Concrete – The Economics of Commercial Buildings for the steelwork industry, see BSC [113]. Goodchild's report compares a small number of three-storey and seven-storey buildings, in different geographical locations. Notably, the study chose buildings with similar modest grid sizes of about 7.5m. The study was limited to a small number of buildings, but very many useful conclusions were drawn. The study found structural in situ concrete to very competitive alternative to steelwork. Some of the advantages of both concrete- and steel-framed buildings are indirect, relating to the accommodation of building services more easily or erradicating the need for expensive air conditioning plant, or aesthetic quality avoiding expensive finishings. Cost savings varied between sites, number of stories and frame type. The BSC study asserted that steel frames indisputably offered the fastest form of construction.[14] The RCC study found RC to offer an efficient, cost-effective solution. Both studies acknowledge that the structural costs are but a part of the project, and that design choices are influenced by project-specific needs.

4.10 Miscellaneous Design Rationale

Whilst neither comprehensive nor selected upon merit, specific considerations that influence design decisions and that promote efficiency, are presented next. This information is based on advice largely obtained from practicing designers and general-purpose design guides.

For most structures, design rationalization offers a number of benefits. It is generally preferable to use standardized components rather than to attempt to optimise the design of each individual structural member, and widespread repetition often leads to cost savings through economies of scale. At any scale, repetition simplifies design, promotes efficient construction practice and helps to create a satisfactory appearance. For example, in the construction of in situ RC components, repetition enables a significant economy to be realized through the re-use of formwork, as formwork alone represents a significant construction cost. Certain kinds of prefabricated structural component, like precast concrete panels (as used for flooring and cladding) are mass-produced to standard sizes

and provide an efficient structural solution when used in conjunction with a regular structural layout.

The majority of the cost of the superstructure of a building normally comes from the slabs and beams that make up the floor system, rather than from columns. There are several basic types of structural floor system. In situ RC floor construction is one highly versatile type. Prestressed concrete construction is an alternative. Precast floor units and profiled steel decking are further options that use prefabricated components. Note that different manufacturers produce different units in varying shapes and with varying capacity. In general, floor slabs are designed to span either in one or in two directions. Secondary beams incorporated into a floor system design help transfer the load from a slab to the columns and foundations.

An advantage of precast floor construction over in situ construction is that it provides a rapid working platform. Construction can advance more rapidly if formwork and props do not obstruct the progress of secondary activities. A good example of fast-track construction was an office development at 288 Bishopsgate, London, described by Whitelaw [114]. Prefabricated components were used extensively to produce a cost-effective solution with a short construction schedule. Construction was able to advance at a rate of up one floor per week. Profiled steel decking similarly eliminates the need for secondary shuttering. The steel acts as permanent formwork to an overlying structural concrete slab, creating a composite floor. Prestressed concrete is a less common type of floor system. It requires specialist knowledge to design and construct and hence is more costly than RC construction. However, prestressed design can be used to achieve long clear spans (typically in the range 6-14m) using relatively slender beams. Not only does this increase available floor space, but also it allows additional stories to be incorporated into buildings where height restrictions apply. Another technique is the use of high-grade concrete in situations where a lower grade concrete would suffice; the intention being to achieve a safe service load in a shorter time to enable construction to progress rapidly.

Floor systems may be solid or hollow. The latter can offer better performance in terms of strength-to-weight ratio. Floor slabs containing hollow voids have been used to carry services discretely. Similarly, slabs with ribs and troughs may house recessed lighting and cable ducts. Where function permits, a quality finish to a floor slab may suffice, avoiding the cost and time associated with providing suspended ceilings or raised floor systems.

---

[14] Procurement times and contract duration are found to be similar for steel- and concrete-framed buildings.

This also has the effect of reducing overall storey heights and consequently, building costs. Although the role of a floor system is primarily structural, noise reduction and privacy are important in-service considerations.

It is generally desirable for the width of the bays that define a structural grid to be regularly spaced as this creates a naturally uniform appearance. It is easier to construct a repetitive floor system with regular dimensions, especially uniform depth. Column loads are more evenly distributed over a regular grid, promoting the repetitive use of standard column sections. Since steel columns need to be spliced because of economic constraints relating to fabrication, transportation and construction, those with constant internal depth simplify connection design.

Whilst the load in columns normally increases progressively at each floor level from the roof to the foundations of a multi-storey building, steel columns with constant section properties usually extend for two or three stories at a time, rather than being designed optimally for each floor. Having said this, Reid [115] illustrated how different column sections may be located internally and around the perimeter of a building, to produce an economical layout without aesthetic loss. Appropriate column positioning is very important as it determines the amount of uninterrupted floor space that ultimately becomes available within the building. Jones [116] has demonstrated how poor column positioning results in significant loss of functional space. Sometimes columns are purposely designed at a spacing that provides adequate clearance for other secondary construction activities, such as access for concrete trucks to lay foundations.[15]

Whilst COPs seek safe designs, slender RC members that use minimal amounts of concrete and reinforcing steel in an optimal distribution have proved neither the most practical nor the cheapest solution. Alternative sections are frequently easier to design, check, order, deliver, store, fabricate and erect properly, and hence become more economical, overall. Recent design publications have attempted to elucidate efficient and inefficient design principles. Based on experiences with existing buildings, design guides suggest that adopting a standard layout with components of fixed size and shape can offer cost savings. Guides such as the BRT publication [9] also recommend using constant section columns throughout a floor, in steel framed buildings. Extensive plate welding is discouraged, and fabrication of non-standard components from channel, beam and tube sections is

---

[15] The same applies to the design of piled foundations.

recommended as a better alternative, if permissible. Using repetitive, simple moment detailing is also highly recommended for steelwork design.

Research by Kousmousis et al [86, 87] and Rafiq [88] applied GAs in the design of RC beams and columns took and account of the significant labour cost incurred by manual detailing and fixing methods. Design criteria were formulated to reflect a preference for handling reinforcing steel in common sizes and for avoiding the smallest bars. In a similar way, connection detailing in steelwork is also expensive and repetition is strongly encouraged wherever possible, as might be achieved by using a single, marginally over-sized connector throughout a floor of a building. For RC slabs and walls, layered mesh reinforcement and reinforcement mats, cages and shear hoops provide practical alternatives to loose reinforcing bars, allowing rapid placement. RC column reinforcement can make concrete placement especially difficult and for this reason, the RC Manual suggests that designers regard 4% reinforcement by area as a practical upper limit for RC columns, even though the absolute statutory limit varies between 6-8%.

4.11 GA Rationale

Rationale has been extended previously to GA problem formulation. Symmetry was used to good effect to reduce the complexity of design models. Classical analysis methods have been used to determine the approximate behaviour of fixed-topology structures prior to, or during, the execution of a GA to determine acceptable upper and lower bounds for design variables. The appropriate use of variable linking has also brought benefits, as reducing the number of independent design parameters that are encoded has enabled chromosome lengths to be shortened, in turn. Other promising techniques included the use of progressive, multi-stage optimisations, in which either: -

- a simple problem is solved first, and results are used to seed a more complex one as described in section 3.5.3 by Rajan [69], or

- a problem is solved at a broad level first, and results are used to determine a specific sub-domain as the focus for a more detailed study.

4.12 Design Component Relationships

To support CBD, Sriram [38] formally described four different kinds of relationship that exist between structural components, and which have been important in the creation of KBES design models. *Generalization* and *classification* describe relationships between

constituent parts that make up the building domain, whilst *aggregation* and *alternation* are used in concept generation.

- Generalization is the grouping of a set of similar entities as a generic entity[16] in order to simplify problem representation. The opposite of generalization is specialization. As an example, flat slab construction with and without hollow sections apply similar, specialized design methods and as such, it can be sufficient to treat flat slab construction as a generic option when developing a broad design model. (Supplementary details that distinguish the two methods could be introduced at a later stage of design).

- Classification is the association of a set of instances with a collective entity. For example, catalogues produced by British Steel Corporation classify numerous standard section sizes, used for steelwork design. Larger steel sections include universal beams (UB), universal columns (UC), joists, channels and angles. Smaller items include bolts and reinforcing bars. The opposite of classification is instantiation, and a typical instance of a UC is that having section properties: depth=0.152m, breadth=0.152m, mass=23.0kg/m (a 152x152x23.0 UC section). Other component and material manufacturers classify their own product range. For example, structural concrete is available in various grades and mix proportions.

- Aggregation is the creation of a complex entity from constituent parts. For example, a structural floor system may be formed using a number of RC beams carrying a RC slab, in which each RC component requires individual design. According to the intended function, lighting, cabling and other services may be carried above or below the slab and a non-structural topping may be applied. Using aggregation, a complex building concept can be generated from elementary parts.

- Alternation is the definition of alternatives, and is important as not all buildings use the same set of components and subsystems. Different concepts result from options that are permissible and discernible to the design team at the time of design, in relation to the structure and other aspects. For example, different buildings contain different types of structural frame like RC or steel. Certain components and subsystems are easily interchangeable whilst others are more limited in their inter-compatibility.

---

[16] In actual fact, more than one generic entity may be used to represent a number of specific entities.

4.13 Current Approach

Hierarchical ordering of design knowledge has been used in previous studies to establish successive levels of generalization and specialization. Within a hierarchy, high-level options exist that have implications upon lower-level options. Initially, it was considered that a hybrid KBES-GA approach could be applicable to the CBD domain. It was originally envisaged that a DS system might operate by first using KBES techniques to reduce the number of feasible design alternatives, and then might access a GA-based optimisation procedure to validate and optimise the remaining solutions for a chosen building type. However, it was recognized that unless the KBES employed only elimination rules to reduce the number of design alternatives, its reasoning process and any decision based upon it could be imperfect. The subsequent GA optimisation might only be able to produce sub-optimal solutions if components of the most appropriate design solutions were inadvertently excluded from the design domain by the application of inappropriate KBES rules. This idea was rejected in favour of using the capability of the GA to explore a search space broadly.

4.14 Design Variables Types

In section 3.6, chromosome structures were described that used different genes to represent different types of design parameter. A variety of design parameters exist in the CBD domain. In thist study, the GA is integral in configuring aspects of a structural building concept and is specifically required to generate and select between alternative topologies, geometries, proportions and sizes of structural members or subsystems, and to choose alternative components and subsystem types, their locations and orientations.

Many former studies that applied the GA in the structural domain used binary representation schemes in which single-bit genes were used as Boolean variables to indicate the status of members that could be included or excluded from a design solution. In a similar way, binary genes that comprised of multiple bits were used on occasion to represent one of several (i.e. more than two) mutually exclusive options that were permissible in a particular situation. An example of this kind of high-level decision-making was in the aesthetic bridge design study by Furuta et al [91], introduced in section 3.6.5, where one particular gene controlled the overall form of the two bridge towers. Different gene values signified unique structural shapes. In general, genes used to synthesize alternation support discrete choices between alternative topologies and forms.

Researchers investigating structural design problems using the GA have implemented standard section sizes within their design models, as would be required for full-scale construction. Steel members, such as UBs and UCs, represent naturally discrete design parameters since they are fabricated in standard sizes. It is convenient to model these elements using a mapping of binary values to unique section sizes so that genes control member variation by selecting standard sections from tables. Other components involve sizing parameters that appear to be continuous, such as the cross-section of a concrete slab, can also be effectively modelled using a discrete parameter range, due to tolerances imposed by design and construction. For example, the minimum thickness for a structural floor is stipulated by COPs and BRs, and casting a RC slab would normally involve a tolerance of no less than about 25-30mm to ensure adequate cover is provided. Consequently, the depth of a RC slab may be considered as having a discrete range with typical values of 150mm, 175mm, 200mm, and so on.

Geometric or shape variations can be handled in a similar way. Whilst in general variations such as in the position of a node in space are naturally continuous, (meaning any value between an upper and lower bound is acceptable), a discrete variable range can be appropriate for representing a specified bay spacing or storey height in a building model. Note that whereas steel component sizing normally references a look-up table of standard sections, geometric design parameters can use a mathematical relationship to increment or decrement a parameter value. For example, a three-bit gene could be used to implement linear variation in the floor-to-floor height in a building, in 100mm intervals, using an equation such as: -

$$h = 0.1B + 2.7 \qquad\qquad (4.1)$$

where,

h is the floor-to-floor height, in metres, and

B is the integer value of a binary gene.

Since binary schemes produce integer values, a complicated decoding scheme may be required in order to produce suitable real numbers. Either a scaling function may be applied, or else individual gene values may be mapped to discrete real numbers. Other difficulties encountered with binary number schemes concern the use of unsuitable intervals and the need to redundantly encode variables. These difficulties have been alleviated through real number encoding.

In a real-number encoded GA, a real number is maintained between actual bounds that always represents the true value of a design parameter. A tolerance can easily be set and maintained through rounding. As the set of real number includes the set of integers, real number encoding naturally supports the use of integer ranges. Rafiq [100] describes equivalent crossover and mutation operators for use with real number schemes, effective in supporting the detailed design of RC columns. Crossover is always applied at gene boundaries. (The equivalent in a binary-encoding scheme is to apply crossover before the first bit or after the last bit in the gene). Real number mutation involves replacement of an existing gene value with another valid value, selected at random. As mentioned before, whilst the binary encoding scheme permits incremental changes to be made to chromosomes (see Schema Theorem, Goldberg [63]), real number encoding has been reported as being both easier to implement and computationally more efficient in tests.

4.15 Design Modelling

Design modelling using the GA introduces three separate issues: encoding, interpretation and fitness evaluation. Encoding involves determining which design characteristics to manipulate directly through the use of chromosomes. Interpretation involves the provision of valid, supplementary information and / or methods necessary to enable the contents of a chromosome to be decoded and to be used to develop a solution consistent with the original design objectives. Fitness evaluation requires the determination and implementation of suitable criteria for design assessment.

4.15.1 Design Objectives and Fitness Functions

Cost is a predominant factor in building design. In 1983, Billington [8] described how limited funds challenged 19th Century designers to seek economical structural solutions that resisted load in a highly efficient manner whilst permitting efficient construction practice.

There have been many studies concerned with minimum-weight design (or, minimum-volume design[17]), using classical optimisation methods, which seldom realize least-cost designs, when construction and other factors are introduced. In 1993, Reddy et al [32] said that minimum-weight design was a more reasonable goal when designing in steelwork as the cost was usually in proportion to the amount of steel required and steel was expensive. Reddy et al [32] asserted that an approach based directly on minimizing component cost

---

[17] Weight is proportional to volume for isotropic and homogenous materials.

(rather than minimizing member size), was more appropriate for RC structures, and therefore was more acceptable as a general design criterion.

Building design involves many considerations - i.e. it represents a multi-criteria optimisation problem (MCOP) - where it is necessary to achieve a balance. This study has limited its objectives to reflect tangible cost benefits, rather than subjective benefits such as those arising from aesthetic appeal. Costs have been used in the present study as they provide a common ground for different *(non-commensurable)* criteria, representing a single objective. Cost-efficiency has long encouraged material economy and structural efficiency. Harty et al [39] showed that through manipulating weighting factors, a best-compromise solution as perceived by the designer could be achieved in different circumstances, using a KBES approach. Kousmousis et al [87] applied similar rationale for detailing continuous RC beams to consistently produce cost-efficient, near-optimal solutions with a GA.

Building design can be broadly considered to be a minimization-of-overall-cost or maximization-of-returns activity, subject to particular functional requirements and constraints relating to construction time, durability, maintenance, aesthetics, the environment, energy consumption, and other factors. Given the present popularity of D&B contacts awarded by competitive tender, and the need for accurate fixed-cost projections, a cost-based approach appeared to have particular relevance for study.

In 1991, Jenkins [80] said that it ought to be possible to create an objective function that expressed structural costs for use with a GA, if sufficient cost information was available, but also noted that the best solution may not necessarily be the cheapest one. Freedom afforded during shape, topology and sizing optimisation within a design model can lead to difficulties in finding optimal solutions. Rajan [69] suggested that undesirable configurations might be favourably repressed using representative cost information. Additional cost, incurred through the selection of structurally inefficient design options, can form the basis for naturally occurring penalty function, promoting the survival of highly efficient systems. A least-cost, rationalized approach to building concept generation has duly been pursued, and is introduced hereafter, using embedded design knowledge of the kind described previously. This study involves a SCOP in which fitness is based upon cost. Grierson et al [99] have subsequently taken this idea further by applying Pareto-optimisation techniques and using rank-based fitness to study the trade-off between building construction, maintenance and running costs, as well as other criteria.

4.15.2 Design Encoding and Interpetation

In order to study the application of the GA in the CBD domain, a sufficiently broad design model clearly is essential to permit structural building concepts to be generated and appraised. To help create this, a series of design models were actually produced during the current research programme. Each model became progressively more realistic of the domain and more complex in its evaluation of fitness, than the last. Models were adjusted in accordance with their success, and their perceived validity to real design situations. Appropriate design models were used to investigate ways in which a GA-based approach might provide greater versatility than previous generative design tools, especially those based on KBESs. Specifically, models that accommodated greater flexibility in geometric and topological variations and supporting the co-development of alternative structural systems were studied with interest.

5 Numerical Experiments with Floor Planning

5.1 Introduction

The form and function of a building are amongst many factors that affect the suitability of different structural systems. Early KBESs supported the use of alternative structural systems in a rigid and uncompromising way. For example, in the design of very tall buildings, overall building height was one criterion frequently used to select or reject different structural systems outright. Figure 5.1 uses pseudocode to show typical application of heuristic information.

```
If   ( Number of floors <= 40 )

        Frame type = Braced Frame

Else

        If   ( ( Number of floors > 40 ) and

              ( Number of floors <= 80 )     )

              Frame type = Tubular Frame

        Else

              If ( Number of floors > 80 )

                          :
```

Figure 5.1: Pseudocode fragment containing heuristic design rules.

Furthermore, KBESs often implemented structural design knowledge in the form of a hierarchy because this enabled undetermined parts of a design concept to be established using relevant inter-related design knowledge. Details such as overall structure height, building footprint size, number of stories and structural grid layout[1] were usually determined early on in the conceptual design process and were used to help resolve other design aspects. Some KBESs have required such details to be supplied as user input. As mentioned in chapter 2, it is often difficult to discern (and hence, specify) the most suitable structural grid or the most favourable construction material, with confidence, at the outset of the design process.

Early research undertaken by the author within the scope of the current research programme concerned tests to determine whether the GA could help improve DS in this regard. Particular consideration was given to the design of medium-rise, general-purpose

offices as these types of building support the use of various structural systems but are generally less amenable to rule-processing techniques.

5.2 Introduction to Floor Planning

Floor planning is a sub-task of conceptual design that requires the subdivision of functional space within a building, around which a suitable configuration of structural elements must be provided. The structural elements comprise the structural frame and floors. It is often the duty of the architect to produce a floor plan for the location of the structural walls and columns, and the task often involves partitioning rectangular areas of a building in plan after the overall footprint dimensions have been established. In general, architects are more closely involved with, and hence, have more experience of, functional design matters rather than the structural design, itself. However, the chosen grid dimensions have a significant influence upon the structural performance and cost of a design concept - indeed, the creation of a suitable floor plan is prerequisite to the provision of an efficient structural system. Therefore, a method of determining the most appropriate grid layouts based on structural and functional design considerations would be useful in its own right, as well as being an important facility of a integrated design system.

Floor planning is an inherently difficult task because of the large number of potential locations and orientations of objects and the many interdependencies between them, for which there is apparently no known direct method guaranteed to produce optimal, feasible solutions, according to Schmidt [117]. The columns that are needed to support the roof of the building and overlying floor systems must generally be positioned in lines in orthogonal directions as a buildability requirement, resulting in a regular grid like that shown in figure 5.2. Early KBESs like HI-RISE (see Maher et al [29]) and DOLMEN (see Harty [37]) did not support floor planning activity; instead they circumvented the problem by assuming that the dimensions of the structural grid had already been ascertained. Whilst this simplified system development allowing progress to be made in other areas, little or no support was given to the architect whose task it was to determine a grid that satisfied the requirement of the different disciplines.

Following on from early KBESs, Karakatsanis [40] and Jain et al [45] developed software tools to automate building floor planning called FLODER[2] and 'Floor Generator', respectively. These tools applied rule-processing techniques. At around the same time, other researchers, like Schmidt [117] and Balachandran [118], investigated how related

---

[1] usually located near the top of the structural design hierarchy.

activities could be supported using numerical methods. The study by Schmidt involved space partitioning within a mobile home, and that by Balachandran involved the subdivision of the private zone of residences, along one plan dimension only.

Figure 5.2: A building floor plan showing the location of structural grid lines and column positions, for a building with footprint measuring 56.0m by 24.0m. (All dimensions in metres).

The rule-based approaches and the numerical methods were both found to exhibit shortcomings. Classical non-linear programming (NLP) optimisation techniques like the Hooke and Jeeves Method and the Simplex Method (see Himmelblau [59]) operate by perturbing one variable at a time to produce incremental improvements. For partitioning problems, where the number of divisions to be made is itself variable, numerical optimisation techniques can be trapped into local, sub-optimal solutions. In a floor planning exercise, unless the starting point of the search process contains the optimum number of bays, these methods are incapable of converging upon an optimal layout. Schmidt [117] reported this to be a fundamental weakness of classical hill-climbing optimisation techniques, applied to solve partitioning problems.

In contrast, the alternative rule-based approaches used heuristic information to develop suitable grids. Techniques presumed the structural layout would follow those of similar, existing structures, and used this knowledge to develop a concept in detail. For example, Jain et al [45] assumed the presence of key features like a central core or atrium in the design, around which it was then possible to create a suitable column grid.

---

[2] from FLO(OR) D(ESIGN)ER.

Some systems automatically located columns at the edges of prescribed *column-free zones*; others applied reasoning such as: - "if the remaining span is too large, then subdivide it into two equal spans by adding an column or secondary beam at mid-span." Although the applicability of the knowledge used was controversial, these systems provided a useful insight into ways in which rule-processing systems could make use of specific design features.

The present study sought to apply the GA in place of NLP and rule-processing techniques in the hope that an alternative technique might be discovered which might impose fewer restrictions upon the design, and might have wider application as a consequence. It was hoped that a suitable technique, if forthcoming, would enable architects and engineers to manipulate floor plans more easily in order to produce highly suitable structural configurations.[3] As a important task in its own right, floor planning provided a suitable platform for commencing the study of the application and capabilities of the GA and did not require unduly complicated objective functions.

5.3 Floor Planning Design Criteria

The author investigated functional space optimisation within a rectangular area and according to simple heuristic. A rectangular area was used to represent the two-dimensional plan view of a cuboid block, a basic functional unit that is commonly repeated in many buildings of complex shape. In an application akin to mathematical partitioning and packing problems, an optimal arrangement of vertical load bearing column members, supporting an overlying floor was sought for a given building footprint. The rationale used to generate and appraise basic floor plan designs assumed the following: -

- The two dimensions that defined the building footprint were fixed.

- Columns needed to be arranged in lines in two orthogonal directions, X and Y, to form a grid. (X was defined as the direction parallel to the long side of the building and Y was the direction parallel to the short side of the building). In each direction on plan, the gridlines in the perpendicular direction divide the structure into a number of bays.

- Floor beams and floor slabs span over columns to form an overlying floor system. A range of spans may be set according to practical and economic limits.

5.4 Encoding the Floor Planning GA

As part of the research, several different ways of encoding a floor plan were studied in the search for a satisfactory technique. All of the representation schemes that were

---

[3] To provide appropriate decision support requires a semi-automatic (designer-led) search process.

investigated used binary encoding schemes, in the first instance. The first chromosome structure to be studied – "Method 1" - was devised to enable a small number of columns (e.g. four or six) to take up positions freely within the plan area. To make this possible, a pair of genes was assigned to each column, acting as X-direction and Y-direction co-ordinate pairs. By applying selection pressure and an appropriate fitness function, it was hoped that columns would align themselves into rows, forming a grid, during the course of a genetic experiment. Fitness functions were developed, based on the design rationale given above, that awarded a fitness score based on the regularity of the grid. Despite many efforts to create a suitable fitness expression, the GA consistently failed to achieve the desired layout. It was discovered that columns tended to align themselves into single rows, rather than producing a uniform layout, and once so aligned, were unable to take up any other position. The randomness of the layouts made fitness appraisal, based upon regularity, difficult, whilst the likelihood of useful patterns appearing by chance alone was too remote for it to be effective on a scale of use in floor planning applications.

In section 3.4, it was reported that the effectiveness of the GA in addressing a particular problem was dependent upon how that problem was formulated. A task can be made easier or more challenging according to whether it is expressed in an open or highly-constrained manner. The intention in allowing structural grids to be generated with some degree of flexibility was valid. However, allowing each individual column to take up its own position (ignoring the buildability preconditions), made structural grid generation more difficult that it needed to be. An alternative approach was conceived which reformulated the problem into one involving the positioning of appropriate structural grid lines at unique locations throughout the floor plan (which, after all, was a fundamental requirement). This method was "Method 2". A design model was created using a binary chromosome in which each individual bit acted as a Boolean variable to indicate the presence or absence of a unique grid line. Initially, a design model was formulated that supported the generation of column grid lines at any whole metre interval in the two orthogonal directions on plan. To achieve this, the chromosome was required to contain the same number of bits as given by the total distance in metres that results from adding the two dimensions of the building footprint together. Bits with values of one indicated the existence of grid lines at specific locations. Conversely, bits with values of zero indicated the absence of grid lines at specific locations. A random population was used to initiate the search. In order to generate floor plans for the building shown in figure 5.3, whose footprint dimensions measured 20m by 12m, a 32-bit chromosome was required. To

represent the structural grid shown in figure 5.3, the chromosome would have the following form: -

$$\{\underline{1}\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,\underline{1}\quad\underline{1}\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,\underline{1}\}$$

For clarity, the chromosome shown above has been divided into the two segments that were used to locate grid lines perpendicular to X and Y, respectively and dummy bits, denoted with an underscore character above, have been included to indicate the presence of permanent grid lines at the edges of the plan area. The presence of a **one** in the left-hand segment (here, the first 20 bits) indicated a grid line running perpendicular to X and located at a distance from the vertical baseline proportional to the position of that bit in the chromosome segment. Similarly, the presence of a **one** in the right hand segment (the last 12 bits) indicated the existence of a grid line running perpendicular to Y at a distance from the horizontal baseline as determined by its position in the string. This representation had several obvious advantages. Decoding the chromosome was a simple matter of counting bit positions. Also, the sum of bay widths in each dimension of the floor plan remained unchanged no matter how the floor plan happened to be subdivided. (This might not necessarily be the case if each bay dimension became a design variable). Experiments were conducted with the following sets of GA parameters: $PopSize$ = 50 / 100, $NoGens$ = 50 / 100, $Prob_{cross}$ = 0.80, $Prob_{Mut}$ = 0.02.
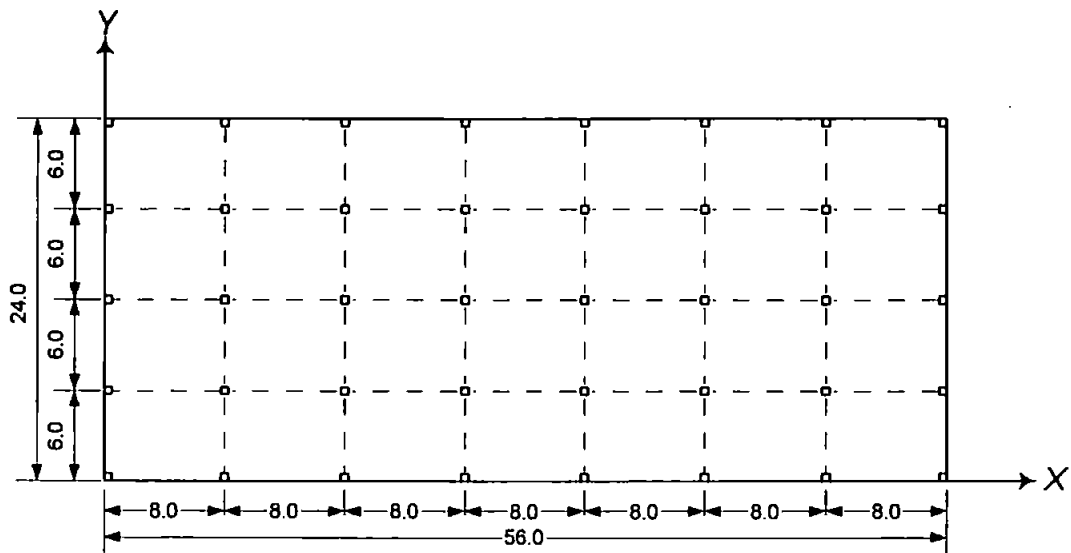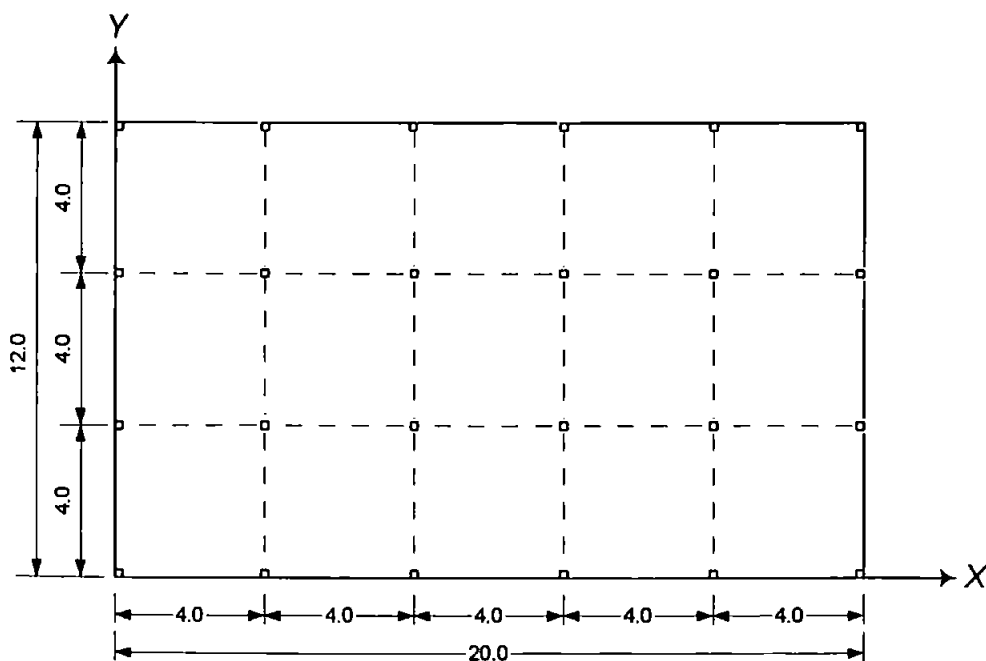

Figure 5.3: A building floor plan showing the location of structural grid lines
and column positions, for a building with footprint measuring 20.0m by 12.0m.
(All dimensions in metres).

5.5 Floor Planning Fitness Functions

To avoid additional complexity, rather than introducing actual structural components into the design formulation straight away, design criteria and constraints were simulated using a set of mathematical equations. A fitness function was developed to represent the dual intentions of maximising the amount of unobstructed floor space (available for let), by minimising the number of bays, m and n, that are partitioned in each direction of the plan layout by the structural gridlines. The objective function accommodated a variable number of bays in the design problem and awarded highest fitness values to chromosomes that used the fewest gridlines to divide the floor plan into the largest permissible bays. It was formulated first as a minimisation problem and transformed into a maximisation problem as follows: -

$$Max(A) = B - Min(C) \tag{5.1}$$

where,

A is the fitness of a design layout,

B is a large positive constant, for example 10^{12}.

C is a cost function.

The cost function, C, was defined as follows: -

$$C = D + P \tag{5.2}$$

where,

C is the cost,

D is the distribution factor,

P is a penalty factor.

The distribution factor, D, was based upon the position and number of column gridlines parallel to the X-direction and Y-direction, respectively, and was defined as follows: -

$$D = L + R \tag{5.3}$$

where,

L is a grid line length factor,

R is a regularity factor.

The grid line length term, L, was based upon the number of gridlines required by a given layout and was defined as follows: -

$$L = (n-1).\sum_{i=1}^{m} x_i^{\alpha} + (m-1).\sum_{j=1}^{n} y_j^{\alpha} \qquad (5.4)$$

where,

m is the number of bays in the X direction,

n is the number of bays in the Y direction,

x_i is the width of the ith bay in the X direction,

y_j is the width of the jth bay in the Y direction,

α is a constant positive, e.g. 1.0, 1.5, 2.0.

The regularity term, R, reflected the regularity of a given layout using the sum of the standard deviation of x_i and y_i and was defined as follows: -

$$R = \beta.\left(\sqrt{\frac{\sum_{i=1}^{m} x_i}{x_{av}}} + \sqrt{\frac{\sum_{j=1}^{n} y_j}{y_{av}}} \right) \qquad (5.5)$$

where,

β is a constant, used as a scaling factor.

Setting $\alpha = 2$ in equation 5.4 provided an incentive for increasing the number of bays in X and Y, and for them to adopt even spacing. This term reflects a general reduction in cost associated with providing structural components of shorter, uniform span. However, it can be seen that in order to generate more bays in X or in Y, more grid lines must also be provided, which constitute more structural elements, thereby increasing the cost. The factored standard deviation of x_i and y_j were incorporated into the fitness function as a regularity term, R. It encouraged the generation of regular grids and allowed a regular, but densely-packed structural grid to receive higher fitness than a random irregular grid, but lower fitness than a well-spaced regular grid. Using $\beta = 10$ was found to create a sufficient fitness variation.

In equation 5.2, the penalty term P was required to discourage the evolution of building layouts that contained one or more bays of uneconomic dimensions. Bays spaced at a distance of 4-8m apart were considered capable of providing a realistic span for a notional, overlying concrete floor system, carried over the column grid. Bays with dimensions in this range incurred no penalty. Layouts that contained bays at spacings less than the minimum limit incurred a natural cost penalty, from having to provide an excessive amount of structural components. No additional penalty was necessary. However, bays

spaced too far apart were penalised by a proportionate amount to simulate the exponential rise in cost of using oversized structural members.

5.6 A New Representation

Whilst providing a concise representation of a floor plan layout, mapping column positions to bit positions had significant disadvantages. Schema Theory suggests that convergence is assisted by high-fitness building blocks of short defining length within the chromosome (see Goldberg [63]). In the current representation scheme there was no distinct correlation of genes and specific design parameters. Instead, the significance of individual bits was based on their position in the chromosome in relation to neighbouring bits. Chromosomes that evolved with a relatively small number of ones separated by a relatively large number of zeros produced good layouts. The distance separating **ones** in the chromosome represented a linear variation. In a standard binary gene the value is derived from the order of each bit. (That is to say, a binary number has a most significant bit at the left-hand end and a least significant bit at the other). Single bit inversion, as performed by the mutation operator, had a dramatic effect. A **zero** that becomes a **one** signified the introduction of a new grid line, and a **one** that became a **zero** indicated the removal of a column grid line. This sometimes resulted in an excessively long span being generated, substantially degrading the overall fitness of a layout. Reasonable performance was achieved for a small footprint like that shown in figure 5.3. However, a large building footprint, for example 50m by 30m, required a significantly longer chromosome, after characteristic rapid improvement in the first few generations, convergence was found to be more gradual.

Various modifications were considered for improving convergence, which included varying the mutation rate for **ones** and **zeros**, and replacing standard bit inversion, with an operator that caused two adjacent bits to be juxtaposed. However, these modifications were dropped in favour another representation, "Method 3", that had began to show good results. In this new approach, bay widths became the design parameters manipulated by the GA. In the new scheme, whilst bay widths were absolute dimensions, the encoding required adjustment to maintain correct footprint dimensions, so in effect was determined in proportion to actual distances of the overall length of the floor plan. This allowed the overall footprint size to remain unchanged, whilst the actual number of (non-empty) bays could change freely. Layouts were decoded by calculating the fraction of the corresponding footprint dimension represented by each gene value. The dimension

obtained was rounded to the nearest 0.2m, which was treated as a practical tolerance for construction[4]. The decoded layout was evaluated for fitness.

In this approach, a maximum number of bays was determined from the floor plan dimensions provided initially, based upon a minimum practical grid size. Chromosomes were then allocated as many genes as there would be bays in this imaginary, dense grid. Each gene contained the same number of bits and used the same variable range to generate floor plans. It was found that sufficient accuracy was afforded by using six-bit genes, which each offered 64 individual values. Using a larger number of bits per gene could increase precision, if required. Assuming that the minimum practical grid dimension was 4m, meant that a chromosome used to manipulate floor plans for the building footprint in figure 5.3 would need five genes to represent the bays in the X-direction (20m ÷ 4m) and three genes to represent the bays in the Y-direction (12m ÷ 4m). In total, there would be eight, six-bit genes, creating a 48-bit chromosome. Hence, a building footprint, 50m square, would require 12 bays in each direction. In total this amounts to 24, six-bit genes, giving an overall chromosome length of 144 bits.

Genes that had a value of **zero**, which were produced when all six of its constituent alleles (bits) were **zero**, signified two different grid lines had converged into one, making the intermediary bay disappear, and causing the actual number of bays present to be decremented by one. In general, the actual number of bays remaining in X and Y could be determined from the number of non-zero genes contained in the corresponding X and Y segment of the chromosome. As with previous experiments, the initial population was randomly seeded, producing at first a variety of irregular layouts. Designs converged towards regular layouts. For example, for a given footprint length of 36m, the chromosome evolved to represent the following regular configurations: -

4 bays at 9.0m centres.

5 bays at 7.2m centres.

6 bays at 6m centres.

Mating could involve chromosome having not only the similar number of bays with different dimensions, but also chromosomes with different numbers of bays. Several refinements to the Simple GA were found to improve the rate of convergence and its reliability at finding a global optimum configuration. Two-point crossover was found to be

---

[4] A carry-over factor was required to maintain the correct footprint size when applying rounding.

much more effective than one-point crossover in achieving convergence rapidly, and for selection RSS was used in preference to the WRW method. Tournament Preselection and Elitism also were applied. As a further enhancement, the crossover operator was modified to recognise parent chromosomes that had similar genetic material and selected cross-sites that could result in useful design variations. To make this easier, genes within each chromosome segment were partially sorted so that any zero-value genes representing non-existent bays appeared last. This did not affect the interpretation of the layout, although it was, by its nature, a problem-specific enhancement. The equations took into account the fact that: -

- a regular layout is preferable to an irregular layout,
- as bay spacing increases, greater versatility is provided,
- as spans increase, so too do costs.

Figure 5.4, figure 5.5 and figure 5.6 and used to compare the results of floor planning using Method 2 (the mapping method) and Method 3 (the proportionate spans method). Figure 5.4 shows the results of a number of runs produced using Method 2. Figure 5.5 shows results produced by a series of runs using Method 3. By comparing these graphs it can be seen that Method 3 has the following advantages over Method 2: -

- A higher-fitness solution tends to be produced at intialisation of the GA (Generation 0). This is not a coincidence, but rather the outcome of using a more suitable encoding scheme,

- After rapid initial convergence, Method 3 appears to be better at continuing to improve upon the layout. Comparing generations 25 to 49, Method 2 can be seen to have converged to a good, sub-optimal solutions by generation 25 whereas, Method 3 shows marked, continuous improvement up to generation 49. Series 2 achieves the optimum layout, fitness = 182.0, by generation 46.

Figure 5.6 shows the average results of each set of tests using the two methods. Method 3 performs better. Figure 5.7 shows the best floorplan produced during at various times during a typical run, using Method 3 (see also Mathews et al [119,120]).

5.7 Related Issues

The study of floor planning activity using the GA introduced several issues. Convergence is found to vary dramatically according to the crossover method, applied. This is not shown here, but can be seen in Mathews et al [119,120]. The objective function was very simple, and there is a delicate balance between reducing structural cost and increasing floor

space. It was also found that adapting the objective function to produce the cheapest, non-penalised layout, could produce different size grids. A dense-grid was one possibility.
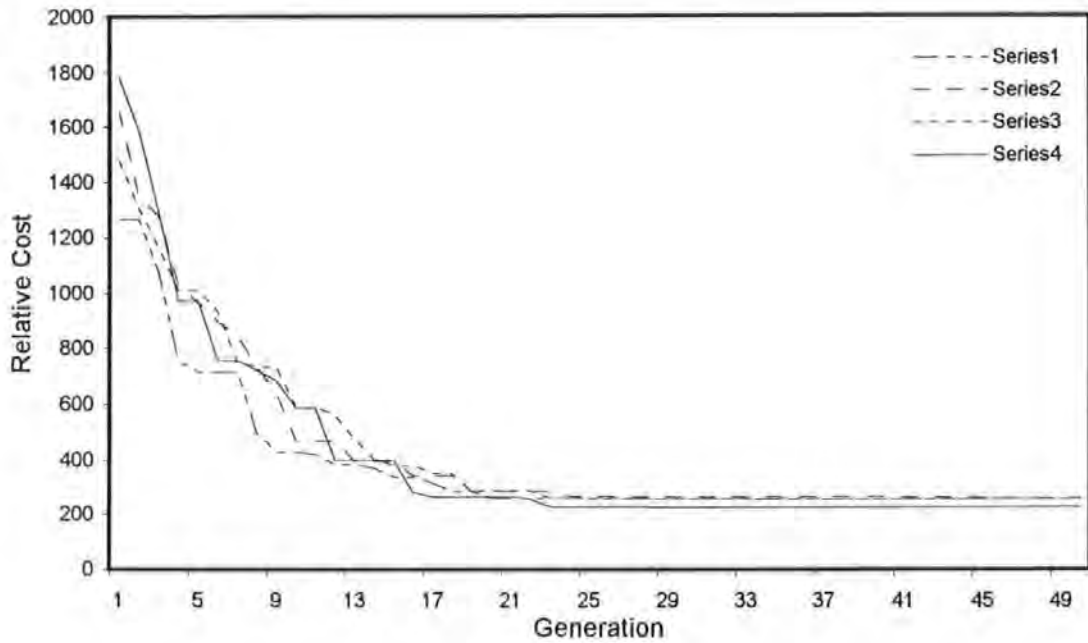
Figure 5.4: Floor planning for 30m x30m building footprint
using Method 2: mapping bits to column grid positions.
PopSize = 100, *NoGens* = 50, *Prob$_{Cross}$* = 0.8, *Prob$_{Mut}$* = 0.02, *CrossMethod* = 2-point,
SelMethod = RSS, *Elitism* = On, *TournPresel* = On.
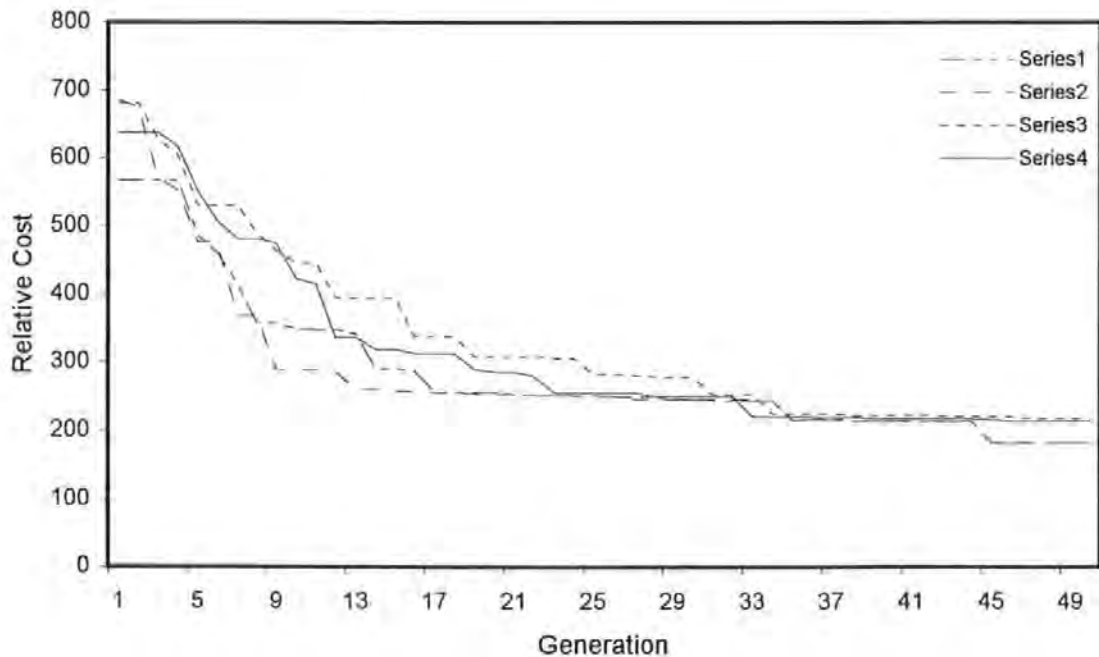
Figure 5.5: Floor planning for 30m x30m building footprint
Using Method 3: encoded bay widths.
PopSize =100, *NoGens* = 50, *Prob$_{Cross}$* = 0.8, *Prob$_{Mut}$* =0.02, *CrossMethod* = 2-point,
SelMethod = RSS, *Elitism* = On, *TournPresel* = On.

Another consideration that arose was how the GA might accommodate variable floor plan dimensions, in order to be adaptable to any situation requiring the layout of a rectilinear building. This can be achieved simply using a batch file. To support designers with no experience of the GA, it can be more convenient to allow parameters particular to the problem to be configured through a shell process, used to invoke the GA.

A third, general issue concerned the mode of operation of the GA. The GA supports a complex problem formulation – that permits greater variability to be expressed – and also can search for highly satisfactory solutions. Although the GA is an algorithm mainly applied without user intervention, it was considered useful to make a user aware of the evolution of highly suitable concepts. The fitnesses assigned to each individual layout were intended to provided a relative measure of suitability, based upon actual design criteria. Individual fitness values did not have especial significance. A method was devised to display the graphical image of best floor plan, produced by the GA. A further modification was then made so that the best floor plan could be seen to evolve, during the execution of the algorithm. It was particularly helpful to apply Elitism, so that every visible change represented improvement in the fitness. Again, no figures are presented but the design progression can clearly be seen if the best grid layouts are presented side by side, at 10 generation intervals. For a large floor plan, for example 50m by 50m, the convergence plots show clear steps as the number of bays is suddenly reduced.
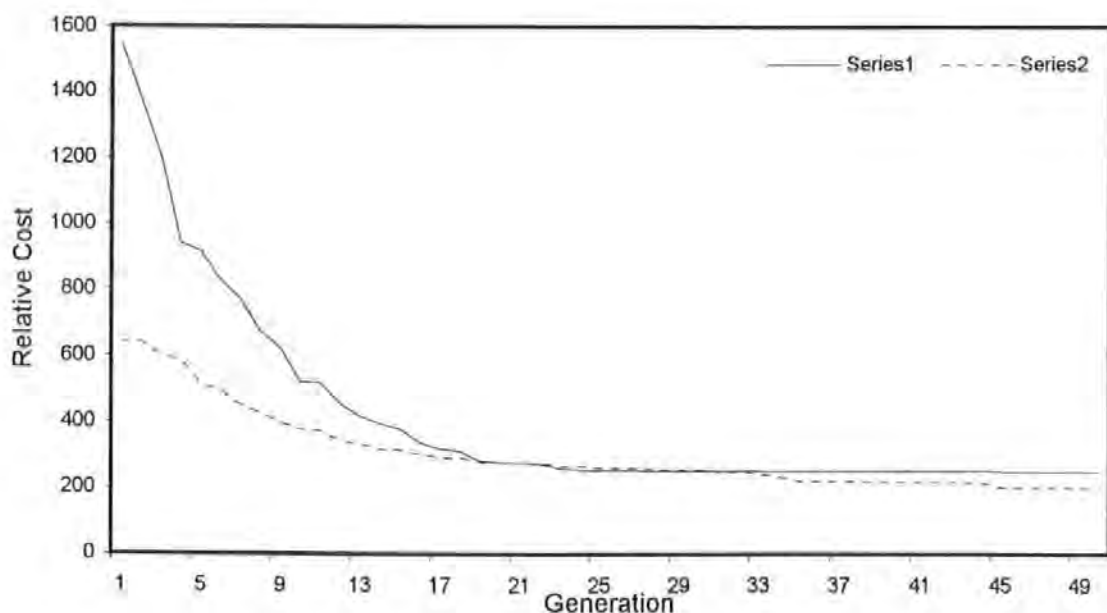
Figure 5.6: Averaged results showing convergence to optimal floor plan
arrangement for a building footprint measuring 30m by 30m.
Series 1 shows averaged results from using Method 2 (see figure 5.4).
Series 2 shows averaged results using Method 3 (see figure 5.5).

Gen 1
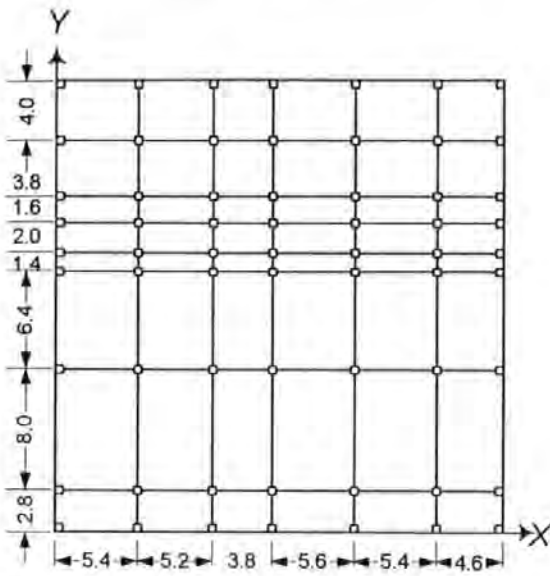
Figure 5.7a

Gen 10
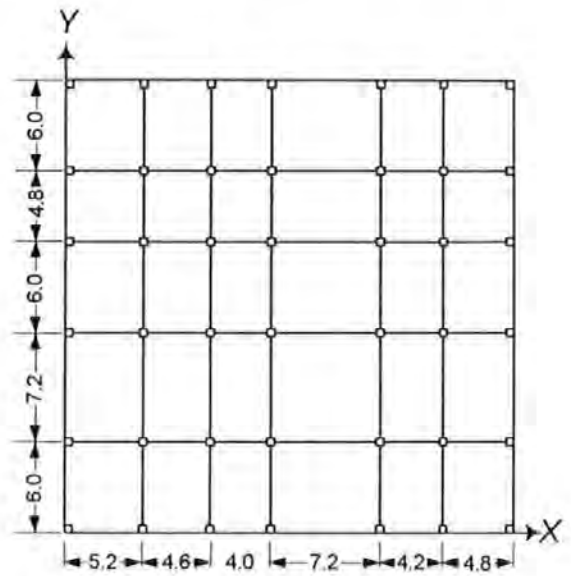
Figure 5.7b

Gen 30
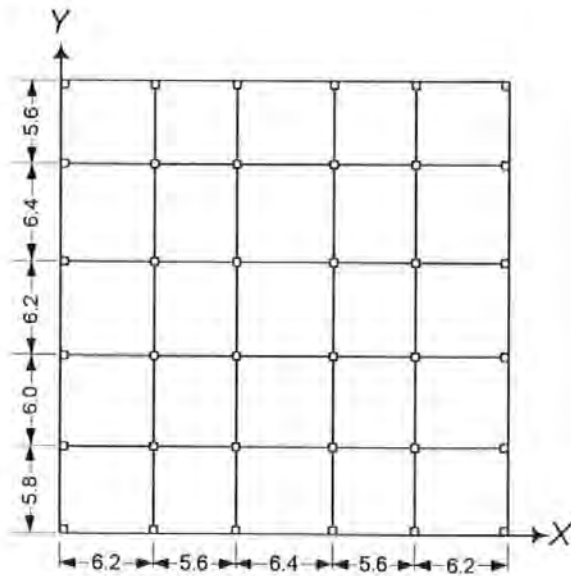
Figure 5.7c

Gen 50
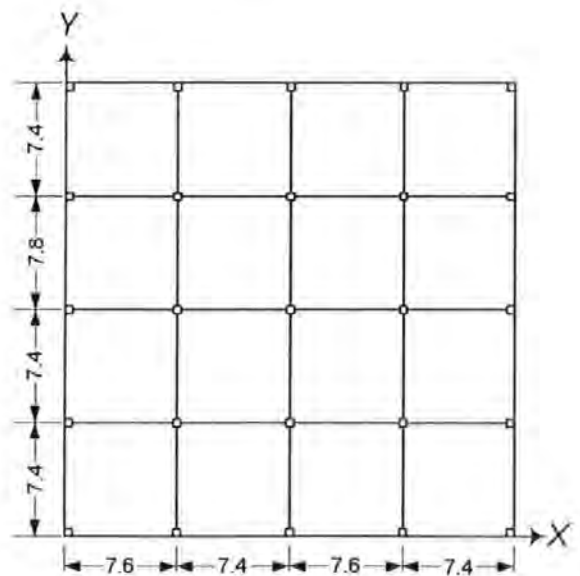
Figure 5.7d

Figure5.7(a)-(d): Example of best floorplan produced at initialisation of the GA and at generations 10, 30 and 50 showing convergence to optimal structural grid for 30m x30m building footprint. (All dimensions in metres).

6 Design System Modelling

6.1 Introduction

This chapter and the next chapter address specific issues relating to the application of the GA in the CBD domain, and introduce specific techniques intended to offer support to the designer. This chapter addresses the representation of domain knowledge, necessary for creating a general building design model. It also presents techniques that permit the manipulation of structural and architectural design aspects. The next chapter describes techniques associated with exploiting the power of the GA more effectively, which includes considering the role of the designer in the design process.

6.2 Building a Design Model

The complexity of a design model can be considered to be a function of its breadth and depth. The breadth of a design model is determined in part by the number of alternative components and design systems that are supported; the depth relates to the detail in which various components and design methods are implemented. The complexity of design models affects the amount of supplementary design knowledge that must be collated and assimilated in order to generate concepts. Complex design models generally require significant manual effort to build, not only in respect of the amount of design knowledge that needs to be synthesized, but also through the need to find a suitable representation scheme to support a substantial volume of information. Complexity also has a direct bearing on the size of chromosomes, the computational effort required to evaluate solutions and, importantly, the computational efficiency of the GA.

6.3 Concept Generation

As mentioned before, KBESs have used a knowledge hierarchy to help develop design concepts. Normally, the major structural dimensions and grid are located at the top of this hierarchy, and must be determined ahead of other aspects. Once resolved, an assessment of feasible three-dimensional frame types is then possible. The choice of three-dimensional frame can be based on alternative, two-dimensional structural subsystems that provide vertical and horizontal load resistance, separately. Element sizing may follow once appropriate subsystems have been decided.

A variety of structural systems are available for constructing medium-rise buildings. Some structural systems are fundamentally very similar to one another. In practice, certain systems are used widely whilst others are only used infrequently. A design model was

sought that could represent the diversity of options that are available in satisfying design criteria. A study was made to determine which options represented the most generic and most popular structural systems used in office buildings.

Figure 6.1 shows the main alternative systems for providing gravity load resistance, in the form of a hierarchy. The diagram differentiates between systems that are compatible with buildings having concrete, steel and timber frames. Items that appear in boxes in the hierarchy represent systems most commonly used in medium-rise, multi-storey buildings. The popularity of steel and RC over timber for the building frame material is highlighted. Masonry is not considered to be a financially viable structural material[1]. Floor systems that use steel plates and grating are impractical for other reasons, such as privacy and noise generation.

Figure 6.1 shows that the construction of an in situ floor slab itself offers additional choices, in terms of the slab profile and the way that the slab is designed to resist applied loading. Because considerable time and effort is needed to build a design model and because of the need to address other aspects of the research, it was considered appropriate to support in situ floor construction in a generic way, in this study. The RC Manual offers conservative estimates for slabs that are designed to resist load in two orthogonal directions. This influenced the decision to make this type of floor representative of all types of in situ floor construction. Similarly, where a variety of proprietary and prefabricated systems were workable in steel and concrete frame buildings, it was considered appropriate to choose one or more to act as generic design alternatives in each case. Figure 6.2 shows the most popular and diverse systems extracted from figure 6.1 and selected for inclusion in the current design model. Note that certain floor systems are compatible only with certain types of structural frame. For example, in situ floor construction (using formwork) is highly uncommon in steel frame structures. A more compatible floor system for steel frame buildings is composite steel decking.

Figure 6.3 shows lateral load resisting systems. For the purpose of conceptual design, a simplified way of providing lateral load resistance was sought. The RC Manual and the Steel Manual suggest than in order to support the rapid production of design concepts, lateral resistance may be considered as being provided independently by the appropriate placement of shear walls, and that this approach still affords economic design. Figure 6.4 shows how lateral load resistance was greatly assisted by this assumption.

---

[1] A stonework / brickwork cladding systems can be used for the building envelope.
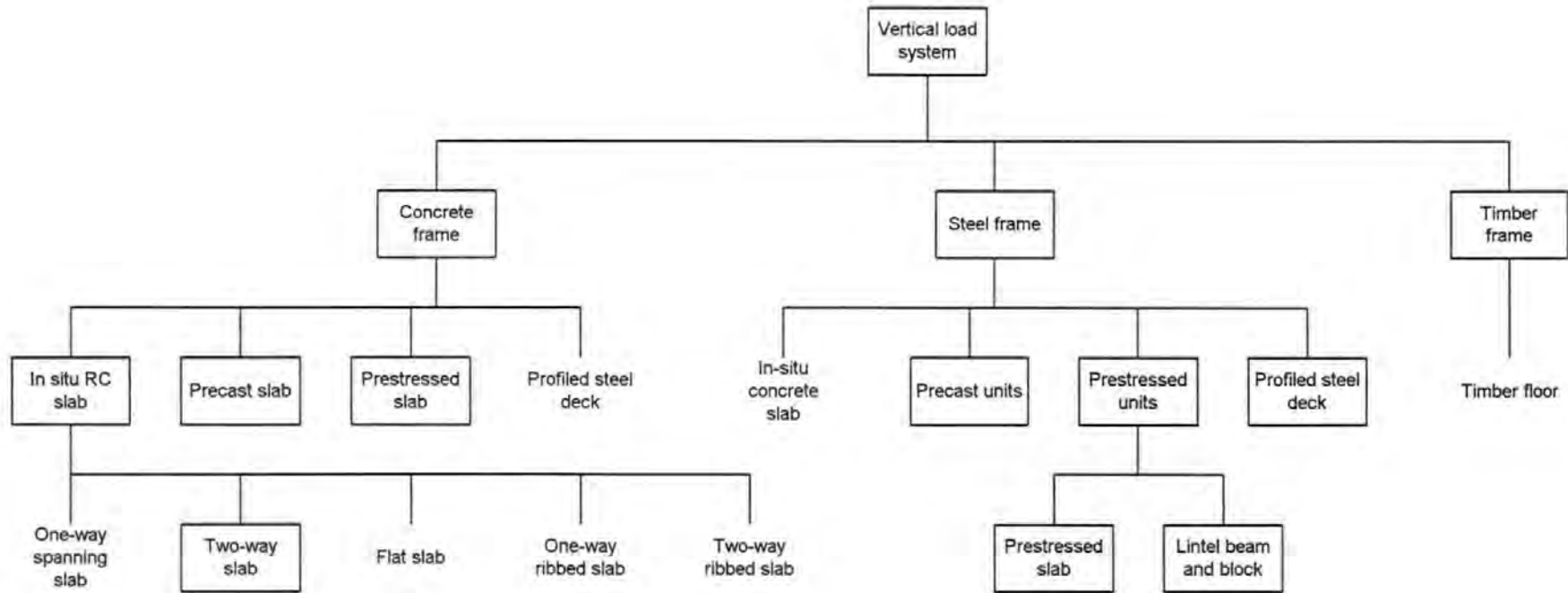
Figure 6.1: A detailed hierarchical model of vertical load resisting systems.
(Boxed items indicate common / generic types of structural system, chosen for modelling).
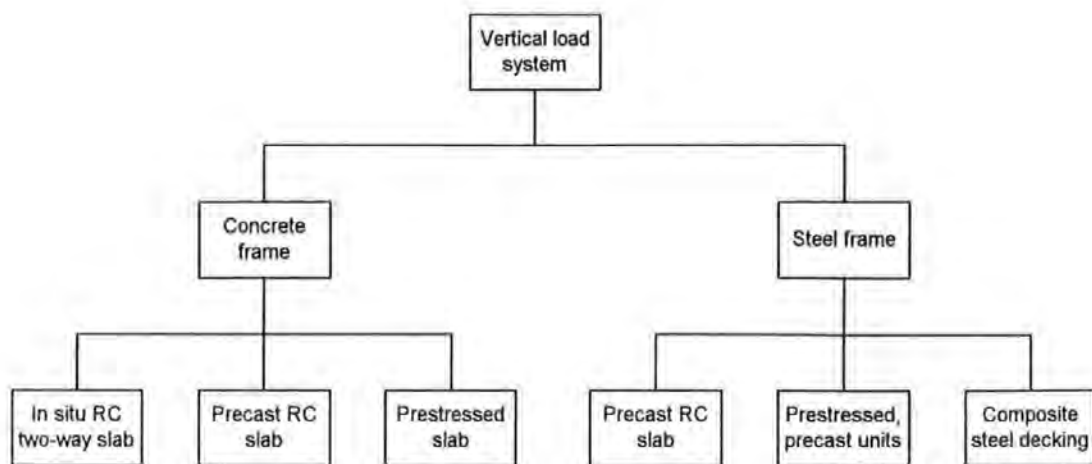
Figure 6.2: A simplified hierarchical model of generic types of vertical load resisting systems in office buildings.
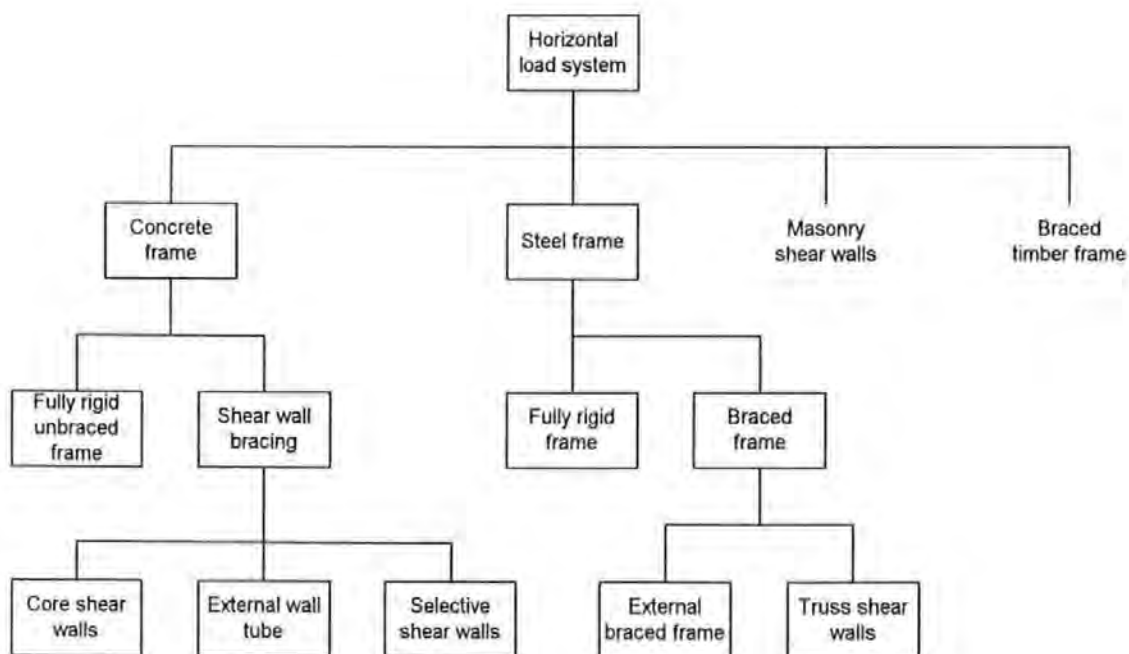
Figure 6.3: A detailed hierarchical model of horizontal load resisting systems.

6.4 Encoding A Building Design Model

A chromosome structure was required that was capable of supporting alternative structural systems as well as permitting the selection of different building dimensions, including the structural grid. Some preliminary experiments were performed that extended floor-planning chromosomes to incorporate an extra gene, identifying a type of floor system.
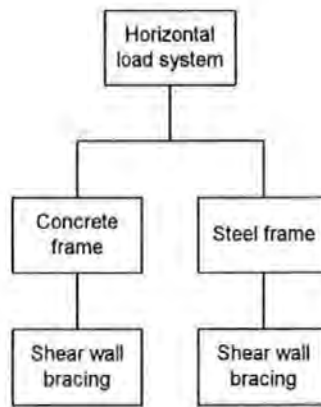
Figure 6.4: A simplified hierarchical model of horizontal load resisting systems.

Exisiting genes governing the topology of the structural grid were unchanged. The fitness function was modified to synthesize idealized grid dimensions for hypothetical concrete frame and steel frame structures. For each frame type a range of grid sizes deemed to be economical was created, any a penlaty was incurreed by any generated that exceeded the permissible range. (For steel framed buildings, a grid spacing of 8m or more is not unusual, whereas for RC-framed structure, a column grid of 6-7m is typical). Evaluation of each individual structural layout was based upon the currently active floor system, as indicated by the value of the newly inserted gene. As a result, existing genes that defined the layout now became dependent upon the floor system type for determining whether or not their values contributed to the production of highly satisfactory concepts.

Conceptual design requires techniques that are capable of enabling alternation between alternative structual system. The fact that different discrete solutions may exist for different floor systems presents a fundamental problem to the normal reproductive processes of the GA. In a standard GA application, crossover strives to combine bits from moderately fit genes to produce higher fitness genes. Crossover also attempts to unite different, beneficial genes from different parent chromosomes, to cause improvement. In attempting to support alternative structural systems simultaneously, genes belonging to different discrete design solutions are mixed that are not necessarily compatible with one another. This was noted to reduce performance. Convergence is likely to occur quicker when compatible parent chromosomes are combined and less likely otherwise.

Obviously, the GA could be used simply to attempt to generate the most viable structural configuration for any one particular method of construction, for any quantifiable criteria. (e.g. the most economic structural configuration, where the criteria is based on cost).

However, it is of far greater value to an architect or engineer involved in CBD to have a tool that can potentially generate and evaluate suitable configurations and subsystems from amongst various permissible construction methods in parallel, and draw attention to the most promising ones. This is especially true because there is usually very little time available to consider the alternatives before decisions begin to be made, and choices are committed. It is also important because, whilst there is a large number of systems and subsystems available to the designers, he or she generally works with familiar and available information – that in memory, and that recorded in other projects. The structural engineer is pragmatic and does not seek to use different solutions without good reason to do so. Applying familiar methods and materials is less risky, and past application is like employing heuristic knowledge of local optima. Proprietary systems may be received with healthy scepticism – particularly in respect of hidden disadvantages like design complexity and constructability - and may take a long period of advertising and promotion to be accepted. It is well known that systems such as prestressing are generally not used because of perceived added complexity in design and construction, as compared with RC.

For these reasons a means of supporting different structural systems in parallel was sought.

6.5 Applying the Structured GA

Design compatibility was only made possible by applying the SGA, a special variant on the Simple GA, in which compatibility is represented in the genetic hierarchy. The operation and applicability of the SGA is introduced at length, in this section.

The SGA was briefly mentioned in section 3.6.5 as having application in combinatorial optimisation problems where it was necessary to support alternative concepts. Implementation of the SGA is very similar to the Simple GA, created by Goldberg [63], with the exception that the SGA uses a chromosome structure that maintains alternative components simultaneously, enabling them to be included or excluded from the design solution at different times. This has enabled the SGA to be used in studies in which combinatorial optimisation is significant. At the Plymouth Engineering Design Centre (PEDC), applications of the SGA included the creation and appraisal of design concepts for a hydropower system and the optimisation of digital fir filters in the field of electronics, by Bullock et al [121].

The SGA introduces the notion of different types of gene within a common chromosome. Existing applications have used genes to represent the value of various design parameters.

For example, Grierson et al [84] used genes to identify UC sections for column members in a plane building frame. This type of gene shall be referred to as a *parameter gene*.

The SGA uses another type of gene, referred to as a *switch gene*. Switch genes are so called because they act as switches to activate or deactivate different segments of a chromosome. The value of a switch gene determines active segments of the chromosome. The contents of the active part(s) of the chromosome then determine the current design solution. By the same token, a switch gene also identifies inactive segments of the chromosome that lie dormant and which have no bearing at a given instant upon the current solution. Switch genes are susceptible to crossover and mutation operators, in the same way as ordinary parameter genes, used for manipulating parameter values. Indeed, crossover and mutation are critical to enable the positions of switch genes to change, so as to bring about the substitution of certain active components that make up a solution with others, and hence introduce variety into the design process.

Crossover and mutation apply to the entire content of the chromosome, and not only to those parts that are active. A shortcoming of the SGA is that for complex domains, the chromsome structure rapidly becomes large and therein relatively inefficient, since most of the material carried along is redundant. There is also always the possibility that crossover and mutation may lead to disturbance of good designs, prematurely.

The number of design alternatives to be supported determines the form of a switch gene. To support two design alternatives, a switch gene can be though of as acting in a similar manner to a standard Boolean parameter gene. The important difference is, however, that in decoding the chromosome, the value of the switch implicitly determines which additional parts of the chromosome are significant at that moment, and which are to be ignored.

In standard binary representation schemes, representing a discrete design variable that permits N discrete values is straightforward if N is one of the values given by $N = 2^n$, where n is the set of positive integers. Many studies have purposely represented a range of variable values as a base-2 multiple (i.e using 1, 2, 4, 8, ... bits to represent 2, 4, 8, 16, ... discrete permissible values). However, if N has some other value it can be more difficult to support the various alternatives. To handle three alternatives (to create a three-way switch), for example, a minimum binary gene length of two bits is required. This yields four discrete values: "00", "01", "10" and "11", and creates a redundancy problem. There

are several possible ways to handle redundancy. One possibility is to allow two solutions to represent the same alternative, but this can create an undesirable bias. Alternatively, it may be possible to contrive a means of ensuring that one of the four gene values is never possible, so that only three options ever exist. One way could involve randomly replacing a disallowed value with an allowed value, selected at random. The author has never attempted to create such an esoteric technique.

The preferred approach that avoids redundancy and bias when using a binary encoding scheme and that works in general for any reasonably small number of alternatives requires a switch gene to possess as many bits (alleles) as there are design options that need to be supported. In this approach, each bit position corresponds to one particular alternative, and every switch gene is initialized and subsequently manipulated in such a way that one of its constituent bits is always on or active (has value one), and the remaining bits are always off or inactive (have value zero). The active bit in the switch genes corresponds to the currently active option.[2] In this way, a switch gene used to control three design alternatives would require three bits and would always take one of the following forms: -

$$\{ \quad 001 \quad \}, \quad \{ \quad 010 \quad \}, \quad \{ \quad 100 \quad \}$$

A gene stored in a single byte can support up to eight 'switchable' options. Two bytes would permit up to 16 options. A special routine has been used to generate the switch gene value. The length of the switch gene, in bits, is set equal to the number of alternatives that require 'switching'. The index of the switch bit that is initially active can be easily generated using a random number, in the appropriate range. Modified crossover and mutation operators avoid invalid switch gene settings by preventing switch genes from having too many or too few active bits. The number of crossover locations are reduced in a binary switch gene. Crossover is not permitted to occur within bits that make up a switch gene, but only at the gene boundaries. In other words, the nearest valid cross-sites are immediately before the first bit or immediately after the last bit in the switch gene. The mutation operator is adapted so that it applies to a switch gene in its entirety, i.e. the whole switch gene is modified and the result is a rotation of the bits, at random, producing any other acceptable pattern of bits. The rotation operator (or bit-shift operator) can be thought of as moving the single active bit with value one to another valid position to its left or right in the gene.

---

[2] It may be obseved that the value of the genes follow principle similar to Gray encoding.

Switch genes support the creation of a genetic hierarchy, essential for providing alternation between compatible structural systems. High-level switches can activate lower-level switches, which represent options that become available through the outcome of an earlier decision. Ultimately, different chromosome segments containing parameter genes are activated or deactivated through switching.

In conceptual design genetic experiments, switch genes and parameter genes co-exist in the chromomsome but are handled differently. The modified crossover and mutation operators described for binary-encoded switch genes resemble the standard operators used for real-encoded parameter genes. (That is to say, crossover occurrs at gene boundaries and mutation involves parameter value replacement). The implementation of switch genes is simplified if real number encoding is applied. In fact, a real-encoded switch gene that is used to determine which part(s) of a chromosome become(s) active works in the same way as a real-encoded parameter gene, used to select a discrete design component from a list of candidates. Real number encoding enables all genes, whether parameter genes or switch genes, to be represented and handled in a consistent manner (using similar crossover and mutation operators).

6.6 Selecting Design Parameters

The SGA switching mechanism was used to accommodate alternative variations in geometry and member sizing associated with individual floor systems. The SGA was implemented within the framework of a design system environment called DPRO. In the system, some conceptual design aspects were directly determined by decoding 'live' parts of the chromosome whilst other aspects were not, and were instead determined according to which components and structural systems had been selected for inclusion in the concept, earlier in the design process. Dependent design details were obtained by applying standard design methods. Main beam and secondary beams are examples. Notably, secondary beam design was optional and implicit in achieving certain spans with certain types of floor system, as designated by the contents of the SGA chromosome. (Beams themselves were always designed according to standard design procedures).

The design aspects that were to become independent required consideration be given to their encoding / decoding, and those which were to become dependent were determined using relevant design calculations and predetermined data. At times during the creation of a design model it was appropriate to support a range of values, as defined by a minumum value, maximum value and increment. Other aspects required a discrete data to be

represented; for example PC components sizes appear in manufacturer's section catalogues.

Table 6.1 presents the different conceptual design variables contained within, and determined directly from, the chromosome. These included the building footprint dimensions and the structural grid dimensions. The building footprint dimensions represent continuous variables. Genes were required to provide variation in the major building dimensions to allow the form of the building to vary. This study confined itself to rectilinear buildings. In order to model the two building footprint dimensions, an encoding scheme was used that supported a range of values from 15m to 100m, at 5m intervals.

Design Parameter	Type	Default Range	Default Interval and Number of Bits
X / Y Footprint dimension	Continuous*	15-100m	5m (16)
X / Y Grid dimension (PS)	Continuous*	3.5-14.0m	1.5m (8)
X / Y Grid dimension (non-PS)	Continuous*	4.0m-11.0m	1.0m (8)
In Situ RC floor depth	Continuous*	0.10-0.50m	0.04m (8)
Composite deck span	Discrete[t]	2.4m or 3.0m	0.6m (2)
Composite deck concrete type	Discrete[t]	NWC or LWC	N/A. (2)

[t]Manufacturer sizes shown.
*Discretized for use.

Table 6.1: Design parmeters that undergo SGA chromosomal encoding / decoding, shown with default ranges and variable type.

The number of floors in the structure was a dependent variable. It was calculated from the total amount of floor space required (as normally specified in a design brief) and the amount provided per floor by a particular floor plan. Similarly, overall building height was calculated within the DPRO system from the number of stories required and using an estimate of floor-to-floor height[3]. As such, neither aspects required encoding.

It was necessary to encode structural grid layouts. It was desirable to permit the number of bays in the structure to vary, although it was assumed that a regular grid would always be generated which meant that parallel bays were equally spaced. Clearly, calculation of the width of each bay in either plan direction becomes a trivial matter when the footprint dimensions and the number of equally spaced bays in the same direction is known. Note

---

[3] Storey height calculation took account of the depth of a floor system and a clear height to ceiling provision.

that the presence or absence of secondary beams, and variation in their orientation, provided a form of topological variation and had a direct influence on the structural grid size.

Previous investigation of structural grids as part of the floor planning exercises described in chapter 5 revealed that the permissible span range of different types of structural floor systems heavily influences final bay spacing. As such, the structural grid dimension was discretized, in a similar manner to the building footprint dimension to whole-metre or half-metre intervals for each of the different floor systems supported. Later in the design process, these decoded values were intelligently adjusted to produce equal spacing in the structural grid layout, as necessary. For example, if the length of the building is 20m and there were three bays, then the concept produced has *three bays at 6.7m spacing*.

Most viable structural designs adopt economic or practical operating ranges. It was necessary to determine operating ranges for all variables and design parameters used by the DPRO system to generate building concepts. DPRO supported greater variation than is generally encountered in real structures and as such, necessitated certain additional considerations when modelling variable ranges. For some variables, the normal operating range was extended beyond the notional upper and lower bounds deemed economic and practical. This was necessary to support the required geometric variations. It meant that, for example, deep RC beams could be created for large spans, and conversely, adequate prestressed components were available for inclusion in layouts having extremely short spans. (That neither situation would be structurally efficient did not matter for this would be highlighted through the application of selection pressure based on fitness, by the GA).

Similarly, design procedures accommodated very large RC column sections that were capable of providing massive axial load and moment resistance. Again, this was not to support the generation of the most suitable concepts (which naturally involve more realistic loads and sections), but to support the other possible configurations, that do not. For these reasons, the component size ranges shown in table 6.2 at first glance appears to include abnormally large or small sections.

In contrast to extending ranges, buildability consideration meant that certain ranges were reduced . BS8110, Table 3.27, Clause 3.12.5.3 indicates that for in situ RC columns, that the amount of reinforment in the section should be within the range 0.4-6%, whilst

practical experience has found 1-4% to a more realistic range. This was the default range used.

System	Type	Parameter	Type	Permissible Values / Limits
Floor	RC	% Reinf'ment	Continuous	> 0.13%
	PC Panel	Span	Discrete[t]	5.0, 6.2, 7.7, 8.8, 9.3m
		Depth	Discrete[t]	155, 225, 300, 400, 500mm
	PS Panel	Span	Discrete[t]	3, 6, 7.5, 9.5, 12, 13, 14, 15m
		Depth	Discrete[t]	150, 200, 250, 300, 350, 400, 450mm
Beam	RC	Min Section	Continuous[*]	0.175m x 0.125m
		Max Secition	Continuous[*]	0.900m x 0.450m
		% Reinf'ment	Continuous	0.13- 4%
	UB	Min Section	Discrete[t]	127x76x13.0kg/m
		Max Section	Discrete[t]	914x419x388.0kg/m
Column	RC	Min Section	Continuous[*]	0.20m x 0.20mm
		Max Secition	Continuous[*]	2.50m x 2.50mm
		% Reinf'ment	Continuous	1 - 4%
	UC	Min Section	Discrete[t]	152x152x23.0kg/m
		Max Section	Discrete[t]	356x406x634kg/m

[t]Manufacturer sizes shown.
[*]Discretized for use.

Table 6.2: Non-encoded (calculated) design parameters associated with various structural floor and frame options, shown with permissible values.

In the course of creating the final design model, in situ RC components and precast concrete components were the first floor systems to be modelled. At this intermediate stage of development, the author published details of the approach being undertaken and its aims, see Mathews et al [120]. These details included a description of economic span ranges for the in situ RC and precast steel components that were considered. In its final state, the system considers not only these systems, but also composite steel decking and prestress concrete floor options.

Certain components are fabricated in discrete, absolute lengths. Composite steel decking is a good example of a structural floor system that is often manufactured in a very limited number of short spans (see table 6.1). In practice, it is necessary to introduce secondary beams in order to enable a structural grid to achieve satisfactory dimensions, i.e. to create adequate clear space. Secondary beams were implemented primarily to support short-span

systems such as composite deck construction, but were also made available elsewhere to provide greater diversity in concept generation.

The amount of calculation necessary to determine loads and suitable component sizes varied according to the type of floor system. Selection of a pre-fabricated section, from tabulated data, reduced the amount of calculation in some cases. An unfactored imposed load of $3.5kN/m^2$, typical for office buildings, was initially used to design a structural floor. Note that the system enabled the floor load to be altered, at run-time. Following the design of the floor slab, other components were designed using standard methods given in COPs. These require details such as spans, applied loads and bending moments to be determined. To include beam and column systems in the design solution did not require the SGA chromosome structure to be extended. Instead, factored dead load and live load values from the slab were used to calculate the total design load transferred to beams. The beam loads were used to design columns and foundations in an approximate way, in turn.

Beam and column component design used either RC or steel members according to whether the building frame was concrete or steel. For steel framed buildings, UB and UC sections were used. The full range of standard sections were made avaiable to the system for concept development. For concrete-framed buildings, it was assumed that by following in situ design methods, the size and cost of components could be approximated reasonably well. The RC Manual provided a rapid design technique based on *simple design* using stocky columns. Although full design methods accommodate more slender member design, this assumption gives a reasonable approximation to the true size of members and is certainly adequate for the purposes of concept generation. The ranges of non-encoded design parameters used for floor slabs, beam and column systems are shown in table 6.2.

The SGA enabled different segments of the chromosome to be assigned to individual structural systems, and permitted any supplementary variables to be included there, also. Table 6.1 and Figure 6.5 show the structure of the SGA supporting the different structural systems identified in figure 6.2. The SGA supports different grid dimensions, considered practical for each individual construction method being implemented (Notably long-span PS floor systems). Figure 6.5 shows that slab depth was treated as a design parameter for in situ RC floor construction, enabling slabs to be generated with different section depths and containing different amounts of reinforcement. For composite steel decking, it was possible and appropriate to model normal weight concrete (NWC) and light weight concrete (LWC), to provide a more realistic degree of design variation.

87
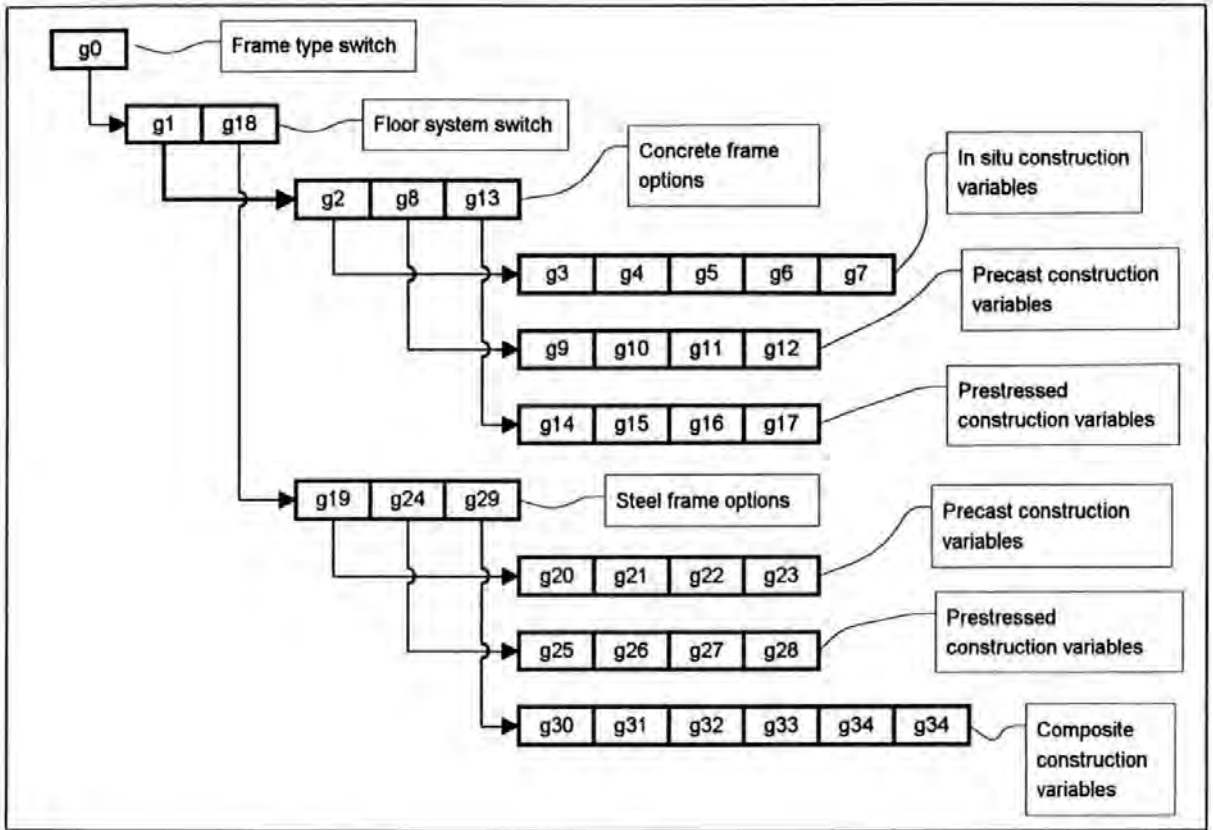
Key

Gene	Description	Gene	Description	Gene	Description
0	Frame type switch	12	Building y dimension	24	Prestressed floor option
1	Concrete frame switch	13	Prestressed floor option	25	Grid x dimension
2	In situ floor option	14	Grid x dimension	26	Grid y dimension
3	Grid x dimension	15	Grid y dimension	27	Building x dimension
4	Grid y dimension	16	Building x dimension	28	Building y dimension
5	Building x dimension	17	Building y dimension	29	Composite deck option
6	Building y dimension	18	Steel frame switch	30	Grid x dimension
7	In situ slab depth	19	Precast floor option	31	Grid y dimension
8	Precast floor option	20	Grid x dimension	32	Building x dimension
9	Grid x dimension	21	Grid y dimension	33	Building y dimension
10	Grid y dimension	22	Building x dimension	34	Concrete type for deck
11	Building x dimension	23	Building y dimension	35	Steel sheet span

Figure 6.5: Structured GA used to model alternative structural systems
with key to indicate function of genes.

6.7 Supplementary Design Parameters

The calculation of a cost-based fitness value was very straightforward once the necessary design procedures had been completed. The flowchart in figure 6.6 shows the basic design development and appraisal procedure that was followed. Certain structural and other parameter values were required to develop a design solution. One, the floor imposed load, was mentioned above. Others included the concrete grade, steel yield strength, limit on column reinforcement and the ideal amount of floor space to be provided. Initially, these values were hard coded but subsequently were allowed to be modified in an interactive manner, within the DPRO system. Since fitness was linked to costs, unit cost information was extracted from sources such as Spon's [122] and Glenigan's [123]. These price books list unit costs of large component, which have been adjusted for average fixing time and connection detailing.
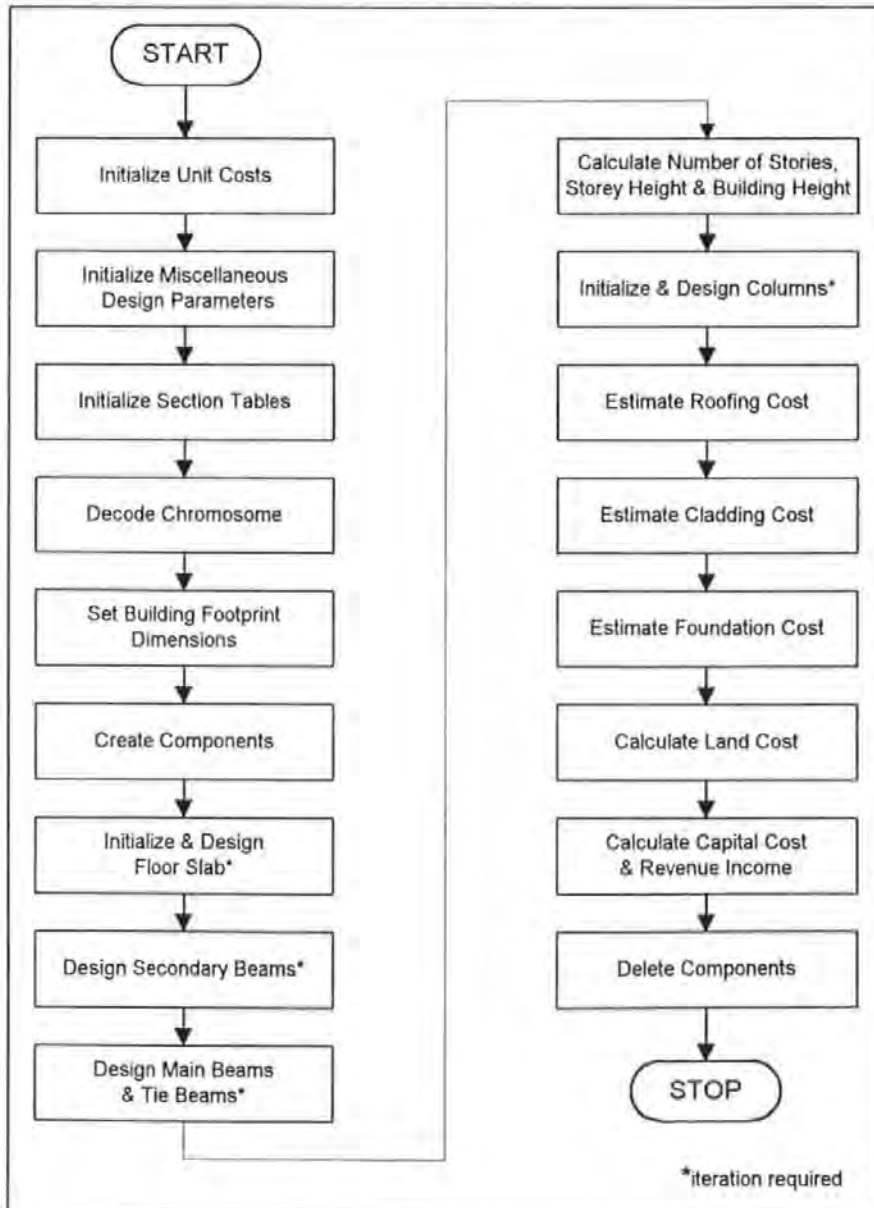
Figure 6.6: Stages in concept generation / fitness appraisal.

6.8 Calculation of Fitness

Unit cost data was supplied as input to the system and was used in approximating the structural and other costs. Once the structural layout and frame for a given alternative design was generated, the associated structural costs could be calculated in a relatively straightforword manner. Then, the costs associated with providing roofing, cladding and foundations, and the cost of purchasing the land on which the structure is to be built were estimated. Although these estimates were very broad, they were intended to produce realistic design variation. Chapter 7 and chapter 8 describe techniques used to support fluctuations in unit price and discuss the derivation of cost data in greater detail.

Cladding cost was a function of the total external surface area of the building, less the roof. The cost of the roof and the purchase of land were both proportionate to the size of the building footprint. Land purchase cost varies significantly between geographic locations. It is common for suitable sites for building to be in short supply in densely-populated urban areas, and this is reflected in a higher price than in less populated locations. The influence of land cost upon conceptual design is addressed in a parametric study in a forthcoming chapter (see section 8.9). Cladding cost, roofing cost and land cost were calculated simply by multiplying the appropriate area, in m^2, by the unit cost, in $£/m^2$, so it is not necessary to present the equations used.

Foundation cost was a function of applied load, and increased in a linear manner. In practice foundation types include pad, strip, raft and piled foundations. In this study, the approximate design of a pad foundation beneath each column is considered necessary for all design concepts. It was assumed that a foundation depth of 0.5m provided a bearing capacity of $200kN/m^2$, and that depth of the pad foundation would increase proportionately to the ratio of (applied load at the foundation / foundation bearing capacity). A foundation unit cost, U_f, that reflected the cost of excavation, design and construction was a pre-supplied, input design parameter, in $£/m^3$. The costs of foundations were calculated using the following equation: -

$$C_f = N.U_f \left(\left(\frac{P}{200.0} \right) 0.5 \right)$$
(6.1)

where,

C_f is the total cost of foundations, in £,

P is the load applied at the foundation, in kN, and

N is the number of pad foundations (= number of columns over the building footprint).

The capital cost of the building was determined by summing all component costs. The income revenue was determined according to the net amount of useable floor space, for which a rent could be charged. Using a projected service life, the profit associated with a particular concept could be estimated.

6.9 Simplifying Design Knowledge Processing

A large amount of structural design knowledge is encapsulated within design standards and its form does not make it readily amenable for use by multi-point search techniques like the GA. Standard design practice requires the satisfaction of numerous empirically-derived equations. Design clauses require particular care to implement in software. Design standards processing is a computationally intensive task since standards incorporate many checks and safety requirements. It can be difficult to represent sets of equations concisely without loss of significant details. For example in RC slab and beam design, there are a numerous of addendums, exceptions and special cases.

Prefabricated elements have standard dimensions, and it is not always possible to directly compare individual components that form part of alternative construction techniques, like-for-like. Nevertheless, an approach was sought that permitted comparisons of alternatives at a broad level. In situ components provide greatest versatility in design since they usually permit alternative section depths and reinforcement content to be provided. Several enhancements were applied to simplify complex design relationships so that they could be implemented in a more concise way and so that the computational effort that was required to generate solutions might be reduced.

6.9.1 Rationalisation and Interpolation

The RC Manual offers advice for estimating the reinforcement requirement of RC slabs, beams and columns. For RC beams it was assumed that steel reinforcing bars would be required in the longitudinal direction to resist flexure; in addition, shear link were provided in the vertical plane. As an approximation, shear links were provided at 250mm centres using 12mm diameter bars.[4] The volume of the links was estimated, and was based on the cross-sectional area of the section.

For the main compression and tension reinforcement, the amount of steel required was determined by graphical interpolation of a series of specially-created design charts. The relationship between beam span, loaded width and amount of reinforcement was

investigated in order to develop this set of charts. The charts were produced using TK-SOLVER[5], a parametric design tool capable of symbolic and numerical rule-processing, list-solving and featuring graphical capabilities. The list-solving capability was employed to determine a set of designs.[6] Figure 6.7a and figure 6.8a show the relationships between the total weight of longitudinal steel reinforcement (used in tension and compression) required in a RC T-beam and a RC rectangular beam against beam span. The graphs are disjoint in places where the RC sections change. The graphs indicate that it is possible to approximate the reinforecement curve to a straight line. Figure 6.7b and figure 6.8b show how the required amount of reinforcing steel varies with loaded width (the effective width of a superimposed slab, carried by the beam), in an approximately linear manner, for a T-beam and rectangular beam, respectively. Figures 6.7a and figure 6.8a were used to estimate the quantity of reinforcement required for either shape RC beam. The values were then modified according to the actual loaded width as given in figure 6.7a and figure 6.8b, respectively.

Where steel beams were required, UBs with suitable section properties were selected from tables to resist the calculated bending moment. Note that the functions required to implement RC beam design and T-beam design exist in DPRO but at present, the system has been limited, on purpose, to always generate rectangular beams, as opposed to T-beam to avoid undue complexity in comparing design concepts.

6.9.2 Memory versus Recalculation

Another enhancement that enabled design solutions to be generated more efficiently, involved a compromise between the amount of memory that is exclusively used by the program and the amount of processing time used for re-calculation. Some design initialisation only needs to be performed once, at start of a genetic experiment. Some design routines need to be peformed at the start of each run, and others, at the start of each evaluation cycle, but outside of a looping process, to avoid unnecessary re-calculation.

For the design of RC members, determining the best section size, the amount of reinforcement and optimal distribution of reinforcement itself represents a detailed design problem. It was possible to simplify the task for RC columns by using a pre-processing routine that determined the most cost-effective section for any applied load that fell within

---

[4] At the detailed design stage the distribution of shear reinforcement can be optimised across the span. This bar size is easier to fix than smaller bars.

[5] TK-SOLVER is produced by ESDU International Plc., 27 Corsham St., London, N1 6UA.

[6] An element of design optimisation was involved at this stage.
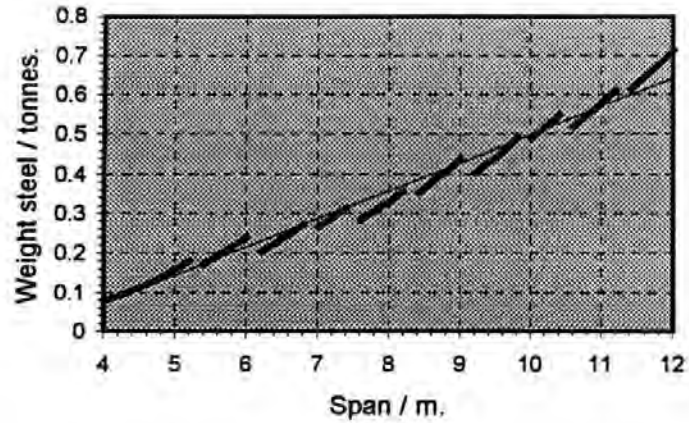
Figure 6.7a: T-beam weight of steel bars vs. beam span based on 6m loaded width (84kN/m run).
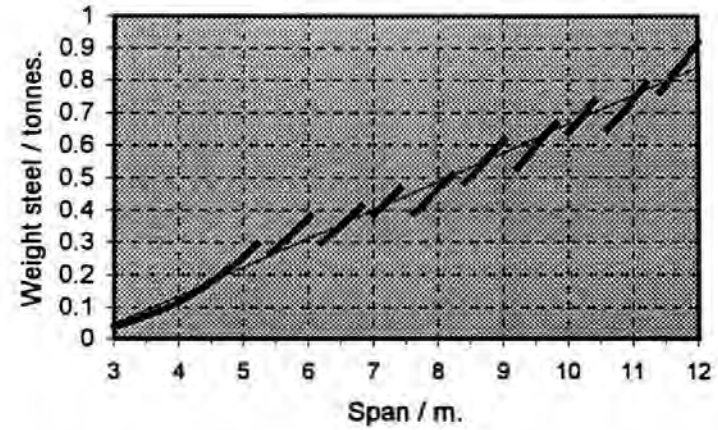
Figure 6.8a: Rectangular beam weight of steel bars vs. beam span based on 6m loaded width (84kN/m run).
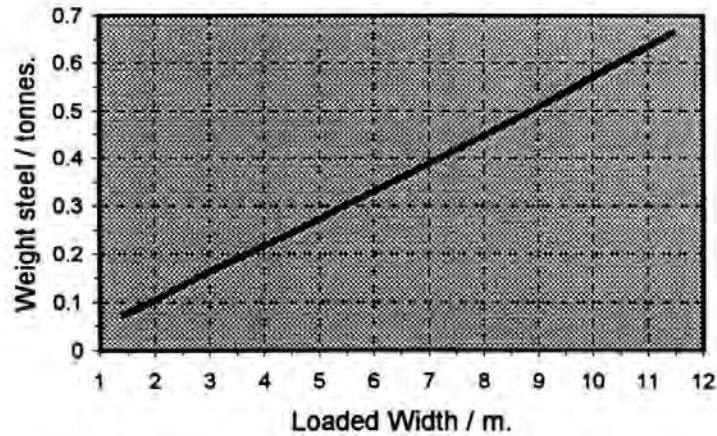
Figure 6.7b: T-beam corrections for various loaded widths.
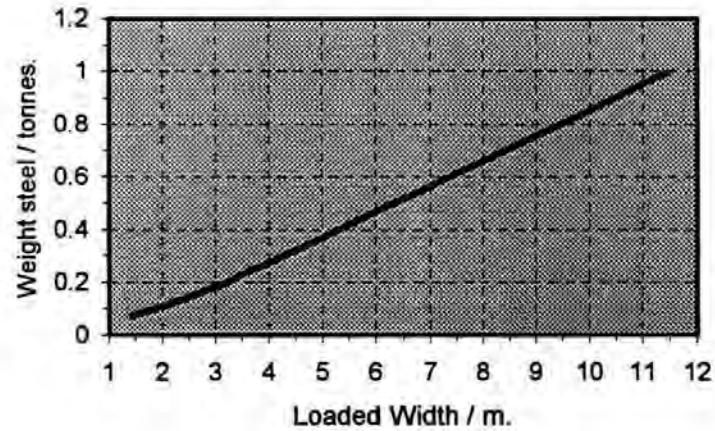
Figure 6.8b: Rectangular beam corrections for various loaded widths.

a series of load bands. The first load band was 0-250kN and the last was 110-125MN. For the range of stocky columns modelled, the reinforcement requirement was always found to be close to the lower limit.

6.9.3 Handling Complexity

A significant amount of programming effort was actually involved between decoding design parameters and developing a design concept, based upon standard methods of structural design. When incorporating a larger design domain, the chromosome becomes more complex. Consequently, the processes associated with initialising, manipulating, encoding and decoding it become longer and more complex as well. The SGA has the shortcoming that as the design domain widens and the objective function grows in size, a larger part of the chromosome is redundant at any given time, yet is still carried over into successive generations.

Whereas the floor plan GA supported a concise objective function, containing about 100 lines of program code, the building concept generation used around 5000 lines of program code to develop a design solution. It was estimated that this amount of code was at least matched, if not exceeded, the amount of code (and the number of individual code modules) required to implement the GA, itself. Clearly, it represents a significant part of the program. Discussion therefore proceeds by reporting attempts to make the lengthy process of concept development and fitness evaluation more manageable, using OOP techniques.

6.10 Application of Object-Oriented Programming

The OOP paradigm is a general approach to building software. All programs can be classified as one of two kinds - those that apply OOP techniques and those that do not. Whereas conventional programs execute a sequence of explicit instructions in a linear manner, OOP is based on the concept of self-contained code units called *classes* that encapsulate relevant data and functions. OOP techniques were used to good advantage in various ways in this project. OOP is particularly powerful as it allows programs to be built and maintained in a modular way, and permits efficient use of memory resources to maximise program execution speed. OOP permits different program functionality to be handled separately, via different groups of classes. Programs can gain greater flexibility and in principle, the intrinsic modularity of OOP methodology helps to make programs re-usable, supporting further development. [7]

Through an appropriate class structure, large programs remain manageable. For example, OOP can be used to segregate program code used for design knowledge representation from information processing techniques. OOP methodology was used extensively to create the present system in which the implementation of the GA constituted one fundamental part and the management of design data was another. The following section provides an overview of OOP methods to show how such techniques assisted in design modelling. OOP techniques are discussed again later with respect to supporting the implementation of the GA.

It was mentioned above that OOP uses programming units called classes. In fact, classes define how parts of a program work; a program itself is constructed modularly from instances of specific classes that are called *objects*. Objects own variables (strictly called *data members*) and member functions (or, *methods*). Data members represent attributes that belong to an object and member functions define the behaviour of the object. Once a class has been defined, it can be used to create as many objects of that class as are required. In order to distinguish classes from other text when describing software architecture, it is conventional for class names to begin with a capital 'C' character (which stands for class), e.g. 'CBeam'. When referring to objects that are instances of a specific class here in discussion, the object names shall appear in bold italic type, e.g. *Beam1*, *Beam2*.

Certain symbolic knowledge engineering languages have previously introduced the concept of a *knowledge frame* for representing design data. Within this context, the frame usually represented a specific design component. Frames had slots for storing design attributes. Slots were filled with design knowledge relating to the component to which they belonged. Empty slots could be filled through the application of relevant design knowledge, through a process called *instantiation*. Objects permit a similar approach to symbolic design knowledge using procedural languages[8]. Objects are more versatile, however, because they are self-sufficient and can be brought into existence and removed from existence at any time during the execution of a program, in a dynamic manner[9].

OOP embodies several programming concepts that are useful when creating a design system model including *encapsulation*, *inheritance* and *polymorphism* and these are briefly explained as follows: -

---

[7] The MS Foundation Classes, or MFC, are general-purpose classes that are made available to programmers to assist in the creation of new application software.

[8] Note frame-based knowledge representation is a specific application of object technology; OOP can be applied to all aspects of programming.

- Encapsulation is a technique that prevents data and functionality from being modified or from being used inappropriately. Essentially, data and functions are only made available to those objects that need to use them. Normally, an object has unrestricted access to its own data and methods, but needs permission in order to be able to modify data or to call methods belonging to other objects that exist outside of its own scope. Likewise, an object's own data and functions are automatically protected from being accessed inappropriately by other objects. The class definition of a generic beam object, CBeam, would typically have member variable as shown in table 6.3. Furthermore, a class can contain instances of other classes. This is a very important capability as it enables any class to make available a copy of its own data structures and methods to other classes that can benefit from using them.[10]

Variable name	Data type	Description
m_Name	Cstring	Unique beam idenfier, e.g. "UB356x127x39.0"
m_SlfWt_kg_m	Double	Beam self-weight, in kg/m
m_Breadth_mm	"	Beam breadth, in mm
m_Depth_mm	"	Beam depth, in mm
m_Span_m	"	Beam span, in m
m_LoadedWidth_m	"	Beam loaded width, in m
m_UDLoad_kN_m	"	Uniformly distributed load on beam, in kN/m*
m_PtLoad_kN	"	Central point load on beam, in kN*
m_MaxBM_kNm	"	Maximum bending moment in beam, in kNm

*may be zero if no load of this type applied.

Table 6.3: Typical attributes of a generic CBeam class.

- Inheritance refers to the ability to create a new class using the structure of one or more classes that already exist. This has powerful implications for creating design models, as for example, it allows specific design components to be based upon the attributes and behaviour of generic ones. For example, the generic CBeam class was used to provide basic common behaviour for a RC beam class, CConcBeam, and a steel beam class, CSteelBeam. The concrete beam class possessed additional attributes including the concrete volume and amount of reinforcement. Likewise, the steel beam class had additional attributes relating specifically to steelwork.

---

[9] using techniques involving dynamic memory allocation and de-allocation.

- Polymorphism is a term that refers to the ability of different objects to react independently to a general type of instruction. Whilst this is another very powerful programming concept, the benefits are harder to visualise. One way in which polymorphism is useful is in allowing a collection of objects that represent different design components – like slabs, beams, columns, pad foundations, amongst others – to perform common actions, via consistent terminology. For example, every component requires *design* and has an associated *cost*. Polymorphism enables individual functionality to be performed using functions having standard names like Design() and Cost(). The benefit is not merely in the convenience of the syntax during programming, but in the fact that design components can be handled in a holistic way.

6.10.1 Design Knowledge Class Structure

Design knowledge was made manageable using classes and by employing the techniques described previously. One overall class was used to implement concept development and fitness appraisal. This class was called CBuildingFitnessFn. From this class, calls were made to specific component classes in turn to determine particular design aspects. The stages involved in concept development and (cost-based) fitness evaluation were given in Figure 6.9. After initialising classes representing specific components contained within the design solution, functions specific to each class were then called to determine aspects of the design. In this way, the steps performed by the main fitness function class could be implemented concisely through a number of calls to different, self-contained objects.

Different design components were implemented as classes to modularise the design model. Figure 6.9 shows various components inheriting general behaviour from a CComponent class. Through its class definition, a design component was given an awareness of having access to a Design() method (used to resolve previously un-instantiated details) and similarly a Cost() method, to cost the component. Specific floor systems, beams and column types were derived from more general objects.

Other important classsses related to the CComponent-derived classes, but not shown, include CMaterial and, CSectionTable, from which were also derived the following: -
CPrecastFloorSectionTable, CPrestressFloorSectionTable, CUnivBeamSectionTable, CRCColSectionTable, CRCColLoadTable, CUnivColSection. It is hoped that the names convey their purpose sufficiently.

---

[10] While multiple instances of a class result in the creation of multiple data structures, only one instance of the member functions are maintained.
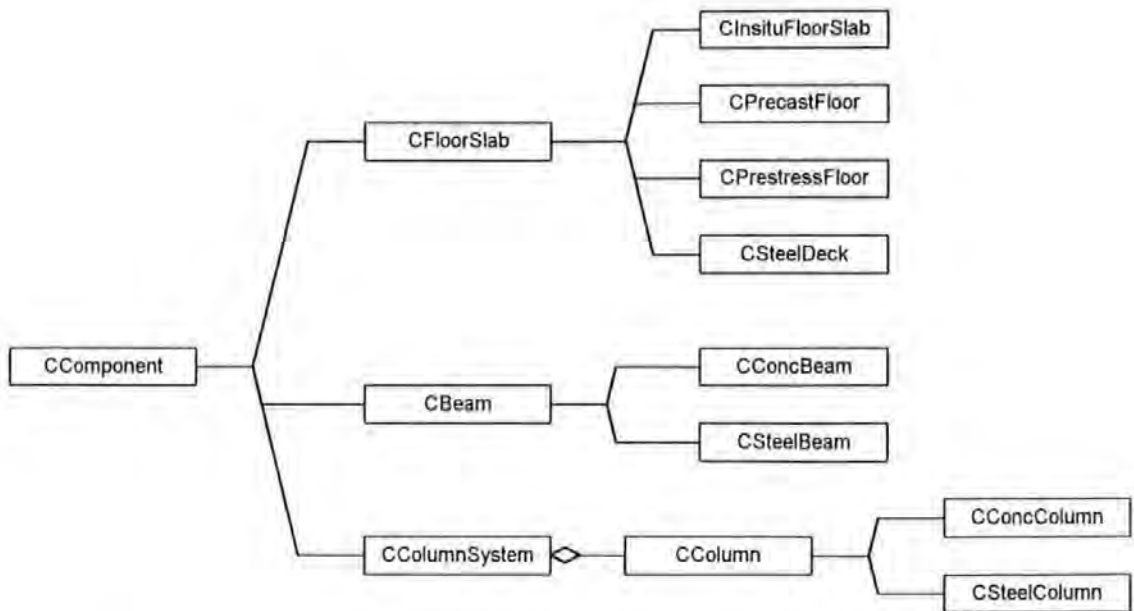
Figure 6.9: Class diagram showing classes used for concept generation / fitness appraisal. (Shown derived from CComponent base class).

Notably, figure 6.9 shows that the entire column system was handled as an object in its own right (CColumnSystem). The column system was used to determine the design of individual columns at different floor stages, and managed relevant design knowledge such as the accumulated axial load. Another important use of encapsulation was the idea that an object derived from CComponent was permitted to contain one of more instances of a class called CMaterial. This class was useful for calculating quantities of material and associated costs, necessary to create composite design components. Given an amount of material per item and unit cost, the CMaterial class calculated the amounts and costs associated with creating components of a given type, per floor and per building. For example, the concrete column class, CConcColumn, contained four instances of CMaterial to calculate individually amounts (and costs) of concrete, reinforcing bars, shear links and formwork required. The overall cost of a column was then easily determined from the cost of each constituent element.

6.10.2 Artificial Neural Networks Revisited

A further possibility that was not explored in this study, but which has formed the basis of subsequent work, has been the application of ANNs in conjuction with the GA to support the development of building concepts. After training and testing, ANNs can replace lengthy formal design procedures as well as being useful in identifying patterns between design specifications (input data) and existing solutions (output data) where the intermediate processes are ill-defined. Discrepencies between optimal (calculated) and neurally-determined design parameter values of up to 10% can be tolerated for conceptual

design, and many ANN studies show agreement in result, with maximum differences equivalent to only 1-2%.

Since GAs naturally support discontinous domains and ANNs can work with partial data, the two fields of AI would appear to complement one another. ANNs can be applied to detailed design problems, to determine specific design components. Jenkins [124] has highlighted the potential of ANNs to solve large-scale structural design problems containing geometric, topological and member sizing without the need for expensive re-analysis, to support activities in conceptual design. Rafiq et al [125,126] adapted the floor plan application of the GA, described in Chapter 5, to use an ANN. Here, the GA was used to generate geometric variation, with the dimension of the floor plan being passed on to a ANN. The ANN was trained for dimensional variation and variation in other design parameters like the design load, so as to be able to recommend a most suitable structural beam from a listed or recognized candidates. Overall fitness was based on the quantity of material that was required in order to produce a structural frame with the dimensions necessary to satisfy alternative layouts. Tests suggest that ANNs are highly suitable to this very kind of application – member selection – and are able to reduce the objective function size and computational re-analysis effort, considerably.

7 Human-Computer Interaction

7.1 Introduction

In previous chapters, it was mentioned that building design has to take account of a variety of different factors. Building designers are often required to exercise their own judgement, whilst adhering to formal design guidelines, in order to produce suitable design solutions. Although IT serves many purposes well, supporting CBD effectively has proved to be challenging. Smith [54] said that effort should focus upon providing an appropriate level of support and should seek ways of empowering designers in their various duties that do not inhibit their creative freedom. Collective research suggests that, in order to be effective, it can be important to: -

- synthesize multidisciplinary knowledge,
- support the full range of different options that are normally available to the designer (rather than to support only a limited subset of options),
- accommodate factors that constitute and necessitate design variations using suitable software techniques (rather than to ignore them outright),
- recognize that software created for the explicit purpose of assisting human designers carries with it various other implications.

At the conceptual stage of design, versatility has always been significant in determining the real value to designers of novel software techniques. Earlier chapters highlighted that, in addition to offering timely and beneficial support for specific tasks, the ability to accommodate design variation and user interaction in an appropriate manner were very important for achieving a satisfactory level of support.

Rzevski [127] recognized that secondary functionality was effective in complementing AI techniques in certain applications, and distinguished *intelligent machine systems* (IMSs) from *intelligent decision support systems* (IDSSs). Rzevski defined IMSs as software programs that were capable of satisfying some need through autonomous action alone, using built-in knowledge of a specific activity. In contrast, IDSSs were more advanced, both in their application and in their design, and were purposely developed to advise and support users through appropriate man-machine interfaces. Gero et al [128] maintained that HCI capabilities, present in IDSSs, were necessary to enable AI techniques to be exploited more effectively and to create practical software tools in fields such as building design.

DS embraces a variety of different techniques. Significant progress in support of CBD activities has been achieved by using more appropriate design process models and by providing better functionality within design systems. Jointly, progress in both areas has enabled design methods, criteria and constraints to be handled in a more open and flexible manner. Modifications to the internal software architecture and to the external user interface of KBESs were shown to influence flexibility. GUIs, auxiliary data files, OOP and *real-time* (or, *event-driven*) software represent specific technologies and techniques with the potential to enhance an underlying paradigm – which includes the GA - in order to provide effective DS.

Early EAS literature presented many techniques concerned with improving the robustness (i.e. the accuracy[1], reliability[2], and computational efficiency) of the GA in specific problem domains using alternative representation schemes and other modifications. In section 3.4, Rajan [69] noted that a separate and ongoing goal of optimisation research was to handle a wider class of problem. With this statement, Rajan respected growing efforts which have been concerned with making the GA more versatile, either by providing a broader search capacity, or else in some other way making the GA more amenable to a general type of problem.

The initial floor planning application of the GA revealed two promising avenues for further research. The first consideration has been continued study of automatic concept generation through the implementation of more comprehensive design models, as described in the previous chapter. The other focus, however, has been an investigation into the practicality of a systematic and interactive approach to conceptual design development, as opposed to the study of unrelated numerical experiments. This chapter addresses this second focus.

7.2 Supporting Variation in Building Design

It is recognised that certain aspects of design are more difficult to support in a flexible manner, than others are. This research set out to investigate the scope for flexibility permitted by the GA in supporting CBD. Particular consideration was given as to how the inherent versatility that the GA affords in the formulation and manipulation of design tasks (in relation to other techniques) might be used to advantage to support common variations in design criteria. In studying the broad application of the GA, ways of helping the designer to identify potentially good solutions were considered. Research addressed how aspects such as design configuration, process control and post-processing support might be

---

[1] Accuracy means the ability to consistently generate optimal or near-optimal solutions.

integrated beneficially within the framework of a GA-oriented DS system. This chapter describes an approach aimed at creating a versatile design tool.

7.2.1 Specific Design Requirements

Extraneous factors that do not relate directly to structural or functional aspects of design must also be taken into account to develop satisfactory solutions. Some constitute requirements and others represent preferences. Hard constraints represent inflexible requirements, whilst soft constraints are those that can be accommodated with some degree of flexibility. Some general constraints that apply in the building domain were introduced in chapter 4. Preference can relate to aesthetic quality based on considerations such as building form or choice of materials. Additionally, to the architect, there may be a preference to use a particular planning module, enabling open space to be suitably subdivided into rooms or workgroups. In structural terms, uniformity of member sizes and loading is generally desirable. Preferences are often linked to individual component costs, which frequently change.

A systematic approach was studied, which aimed to give the designer the means to examine the effect that different constraint and parameter variations had upon the concepts that were generated. However, whilst providing the flexibility to generate alternative buildings concepts is welcome, the extent of flexibility permitted in different circumstances varies. For instance, it is common for zoning regulations and lack of space to restrict the overall form that a building may assume. Often a functional need requires the provision of a minimum clear span, and shortage of space may dictate one or more building dimensions. In section 4.13, a hybrid KBES-GA approach to CBD was considered, but never pursued. The ability to dynamically configure a search space to reflect varying constraints and criteria was considered advantageous and was explored, instead.

As such, a flexible design system model was sought that not only supported the generation of alternative design solutions, but which also supported manual adjustment of the domain to reflect constraints imposed in specific circumstances. Techniques were studied that permitted the design domain to be modified in such a way as to limit or avoid the generation of concepts containing features that were, for various reasons, considered to be unfavourable or inappropriate. Techniques that permitted the range of any independent design parameter to be manipulated to reflect constraint variations were included in this

---

[2] Reliability means the ability to consistently produce good results, independent of the start condition.

study. In this way, the designer was able to concentrate investigation in specific regions of interest.

Naturally, restricting search to subsets of the design domain without good reason may yield sub-optimal solutions. This has happenned in practice where designers have decided that a structure should adopt a particular form and material, with insufficient consideration given to the implications upon other design apsects, or other design alternatives. Structural steelwork has been very fashionable in the last decade, due to aesthetic reasons and the expectation that only steel can achieve large, clear spans. Many warehouses and supermarkets have standardised designs, enabling groundwork to begin whilst designers complete the superstructure detailing and whilst structural elements are pre-ordered, fabricated and delivered to site. An aim of the Cost Model Study, undertaken by Goodchild [112], was to provide designers with a realistic comparison of total cost of steel-framed and concrete-framed buildings of similar size and function. The study showed that the obvious advantages of using structural steelwork relating to aesthetic quality, construction schedule, and large clear spans must be weighted against higher cladding and fire protection costs and loss of space by bracing members and poorer versatility for routing building services, amongst other matters.

In this research, the designer was regarded as being more competent at refining the search space than rigid rule-based knowledge. This approach, described in detail hereafter, was intended to demonstrate a new, flexible approach to co-operative design development using the GA.

7.3 Computational Advances

In section 2.3, it was stated that microcomputers gained wider application as a result of continuous improvements in performance-to-cost ratio. Concurrent advances in software and hardware were presented as *enabling technologies* that helped to further the use of IT in supporting design processes (as well as other fields) by making it easier to create more effective programs. Powerful yet affordable hardware has also led to graphical display technology and graphical OS software[3] becoming standard on modern microcomputers like the PC. Standard user interface components are an integral part of modern graphical OSs, like the current versions of MS Windows for PC. In this respect, the GUI has become synonymous with the graphical OS. Modern high-level programming languages, used to create programs that run on modern OS platforms, possess specific functionality to support

---

[3] From which graphical input devices like the mouse, joystick and trackball have followed.

standard GUI design and other associated technologies like OOP, supporting real-time systems. [4] MS Visual C++ is one example of a development environment containing such features. [5]

Norman [129] emphasized the importance of HCI for delivering effective support. Modern software has helped in this regard. Whilst ever faster computer processors are developed and allow greater processing to be achieved in the same time span, graphical OSs and GUIs have other benefits. Graphical interfaces help to simplify tasks and make the behaviour of programs more intuitive, and as such constitute ways in which software, itself has become more empowering. The capability to allow the user to control the order in which actions are executed is critical to the effectiveness of certain software, to achieve a certain goal, like word processors. Furthermore, some applications are designed for a wide range of users in mind, and support their varied objectives and capabilities. AutoCAD® is good example of a popular software application with a wide range of roles, being used both in academic studies and in professional practice.

7.3.1 Software Advances

OOP and GUIs represent specific enabling technologies. Both have been studied in conjunction with AI techniques to support engineering design and other fields. As mentioned in section 6.10, OOP techniques have allowed programs to be created in a more versatile manner[6]. GUI development tools and techniques have helped to make programs more *user-centred*, by enabling information to be clearly presented and through supporting user interaction in a logical and uncomplicated manner. With respect to the CBD process, GUIs allow designers to carry out parameteric studies to explore possible design solutions and to gain a better understanding of the process.

Interfaces that provide consistent appearance and behaviour across different applications help to overcome cognitive barriers. Discussion proceeds by addressing a number of techniques. Appropriate external interfaces to aid concept design development are introduced, founded upon appropriate internal data structures for representing and

---

[4] For example Windows functionality is provided through an Application Programming Interface and MS Foundation Classes. Necessary but low-level activities such as support for miscellaneous external hardware is handled indirectly via device drivers in, OSs such as MS Windows. Re-use of software components, techniques for linking data between applications and multitasking capabilities are amongst other newly derived benefits of modern OSs.
[5] MS Visual Basic and Inprise Delphi are amongst other popular programming languages, for which specialist toolkits and add-on controls have also become widely available.
[6] OOP is used extensively to create unified product models, described in section 4.4.

processing design information effectively. The topics described next are closely inter-related.

7.3.2 Graphical User Interfaces

The study of appropriate GUIs borders research and development. Appropriate GUIs have been shown to enhance support for design processes. Combining powerful functionality and straightforward interaction can reduce the need for users to undertake costly, formal training, and makes software more accessible. Well-designed software can improve efficiency and can reduce the likelihood of errors that arise through confusion or misuse. The creation of a satisfactory interface often requires careful consideration. AI research has grown to embrace user interface design and has been greatly assisted in this direction by programming languages that now support visual design methods and OOP techniques.

Visual programming tools and OOP enable existing program code, required in order to implement the default behaviour of standard components, to be re-used in specific applications without the need to rewrite it. Although the interface must be linked to internal data and tailored to the way that a particular application responds (which can involve over-riding, replacing or extending default behaviour), this approach nevertheless enables programming effort to concentrate on the basic functionality. For the prototype design tool created during this research, the implementation of the GA and suitable fitness functions were critical parts.

Standard components of a GUI include multiple output windows, dialog boxes, menu bars, tool bars and status bars. Current versions of commercial software exhibit GUIs with features such as these. For example, the LUSAS© FEA application suite[7] contains graphical pre-processing functionality that delivers a highly effective way[8] of creating / submitting a design model, for analysis. Simple controls provide support for basic actions. Radio buttons, check boxes, list boxes, static text and edit fields, and spin buttons represent a set of common controls that are used to present information clearly, to make valid selections and to execute valid actions at appropriate times. Functionality that becomes invalid through specific choices or actions can be hidden or appear disabled. Advanced controls such as the tree control, slider button, progress bar, data grid, and tab card have specific functionality to support special types of operation, involving the presentation, selection, and organisation of different kinds of information. Notably, common GUI components can be very simple in operation and yet when combined with one another and

---

[7] LUSAS is produced by FEA Ltd., Forge House, 66 High St., Kingston-Upon-Thames, Surrey, KT1 1HN.

modified in various ways, create powerful tools for managing design data at a high level. Figure 7.1 is a screen shot of DPRO, the GA-based, DSS created during this study. Many of the aforementioned user interface components listed above have been used. Those such as windows, menus, icon toolbars, push buttons and edit fields can be seen in the illustration.

In a DSS, the capability to easily configure, compare and revise design concepts has special significance. These ideas are taken up again later.

In general, major benefits of GUIs include: -
- making information and processes easier to understand and access,[9]
- making knowledge resources and design options easy to select and adjust,[10]
- guiding users by means of various supplementary help facilities,
- indicating the state of progress during lengthy, automatic procedures,
- maintaining user control at all times, which includes allowing the user to decide how, what and when information is to be acted upon.

7.3.3 Auxiliary Data Files

The earliest KBESs were developed for new domains by re-using an existing inference engine with different domain knowledge. Later, KBES shells were created that enabled domain knowledge to be supplied and maintained by an end user. KBES researchers like Adeli et al [51] identified that it was advantageous to separate the parts of a system that processed information in a predetermined manner from the parts that constituted the domain knowledge itself. Domain knowledge held in external data files can be easily updated or amended[11], and can avoid the need to modify programs directly. External data files allow various kinds of information to be stored, retrieved and altered, and this capability is not restricted only to handling domain knowledge. General uses include: -
- maintaining user settings within a design system,
- providing unique input to a process, for example, a design specification,
- maintaining an accessible database of design information including components, section properties and cost data, to be used during a computational process,
- creating a persistent record of the output of a generative design process, including a log of actions and a description of design details,

---

[8] i.e. more convenient, quicker, more intuitive, less prone to error and easier to rectify.
[9] Note that the external model of design information may differ significantly from its internal representation. Multimedia and related techniques can also help to convey ideas.
[10] which may involve the implementation of external data files.
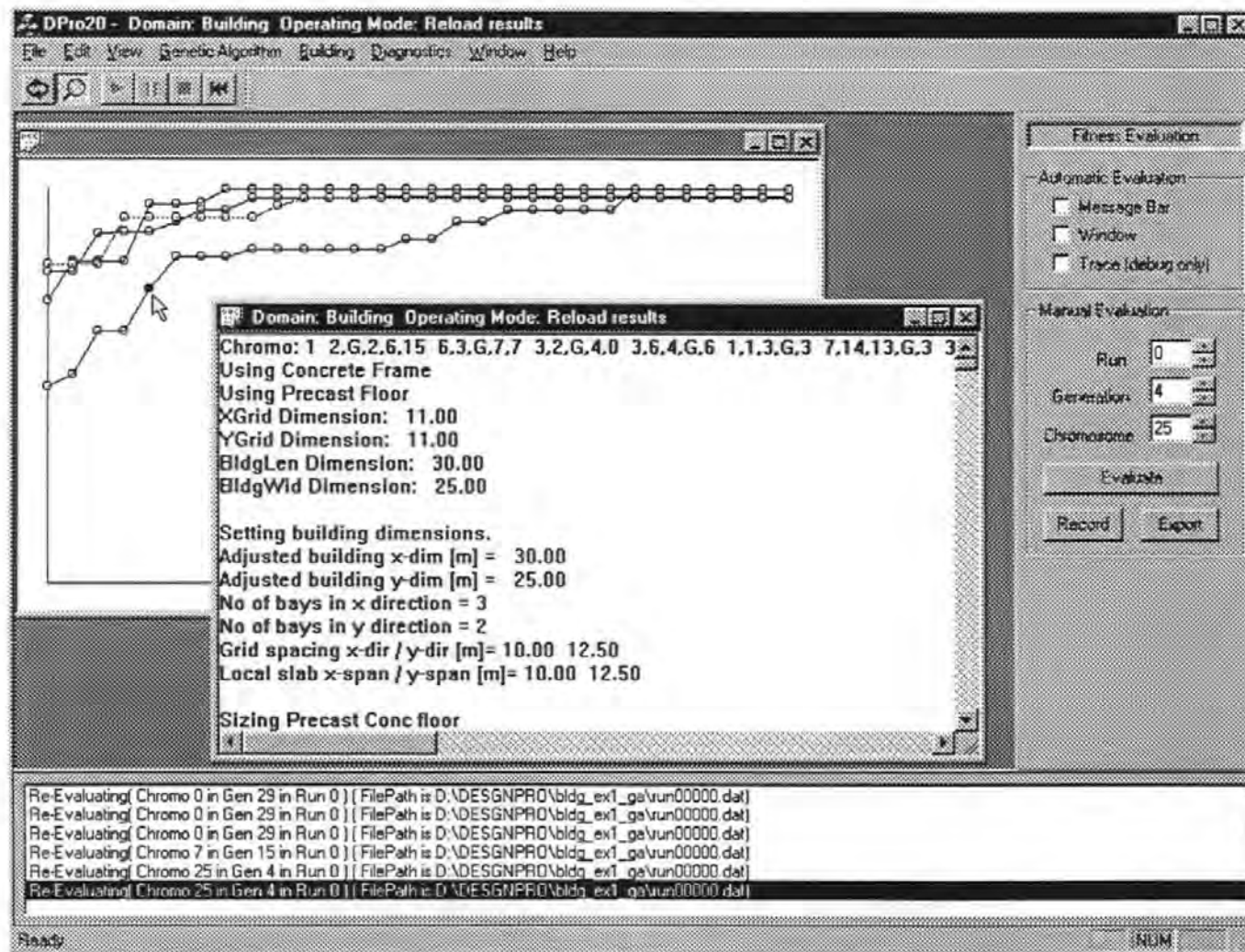
Figure 7.1: A view of DPRO, the GA-based DSS.

- interfacing with other software programs using a common data format.

7.4 Object-Oriented Programming Application

The previous chapter introduced OOP and mentioned how such an approach supported the creation of a complex design model. This chapter continues by describing in detail how the GA was implemented in a similar manner using OOP techniques, to provide enhanced versatility.

A class called CGeneticExperiment became the basic framework of the algorithm. To begin a genetic experiment, a *GeneticExperiment* object was constructed from this class definition. Simple data types were used to store the standard parameters associated with a GA, such as *PopSize*, *Prob$_{Cross}$* and *Prob$_{Mut}$* within the main *GeneticExperiment* class. Owing to program size and flexibility considerations, it became appropriate to delegate certain aspects of GA functionality to other self-contained objects. (Note the fact that the GA was implemented through a number of interacting objects did not affect its basic operation). In the last chapter, classes used to develop and evaluate a conceptual design from the information stored in the chromosome were described. Figure 7.2 shows additional classes that were created to implement a random number generator, a genetic mating pool and control data guiding the GA, all of which could be accessed from the main genetic experiment class.

Every genetic experiment involves the initialisation and subsequent manipulation of a population of design chromosomes, where every chromosome is comprised of a number of genes. The OOP paradigm permitted chromosomes and genes to also be implemented as objects. The fact that many objects were self-contained and were given self-regulating behaviour, by means of appropriate data structures and member functions, was very important in creating a versatile design tool. For example, each chromosome had access to an evaluation function, and possessed a data member to store the returned fitness value, amongst other properties. Similarly, each gene had one data member to indicate its current value or form, in the current encoding scheme and a member function to implement mutation in an appropriate manner. Upon initialisation, the *GeneticExperiment* object created, initialised and/or reset[12], the random number generator, the chromosome array (that constitutes the design population), the mating pool and the statistical control objects. Thereafter, objects could communicate with one another but managed themselves – for

---

[11] Manual or automatic updating is equally possible.
[12] as appropriate.

example, the random number generator object was responsible for automatically creating a new batch of random number when the current batch was exhausted.
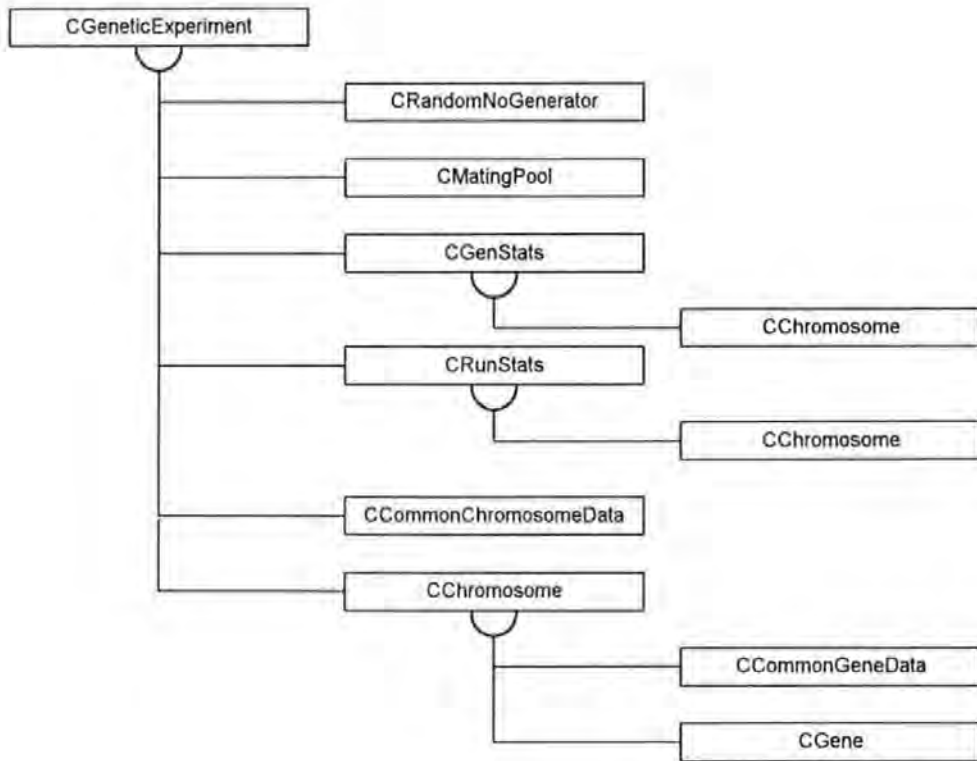
Figure 7.2: Classes diagram showing classes used to implement the GA.

7.4.1 GA Versatility Issues

Attempts were made to implement the GA in such a manner as to offer some degree of flexibility in different design situations. This involved separating the algorithm proper from the design task to which it was applied. Instead of hard coding a complete design problem, as is usual, external data files were used as a medium for supplying a partial design specification to the GA, that could be adjusted to suit different situations. Data relating to the chromosomal details was supplied in a design configuration file. General system settings were stored separately in an initialisation file. The configuration file was read during the execution of a genetic experiment to initialize a design problem and was accessible at other times to update the design specification. Its contents are described, shortly hereafter. Note that samples of all of the data files used for input or output with DPRO appear in Appendix A – F.

Given basic information about the design domain, it was possible to allocate the number of chromosomes in the population and the number of genes in each chromosome respectively,

in a dynamic manner and so avoid hard coding. The chromosome array was created dynamically by supplying the population size (*Popsize*) as an input parameter. For chromosomes containing a fixed number of genes (i.e. of fixed length), each chromosome could be constructed dynamically as an array of self-contained genes by specifying the number of genes that were required (*NoGenes*). Thus, the entire design population was represented by the zero-based chromosome array of the form: -

Chromosome[0], Chromosome[1], ... Chromosome[Popsize -1]

where, every chromosome contained a zero-based gene array of the form: -

Gene[0], Gene[1], ... Gene[NoGenes -1].

Implementation was slightly complicated by the fact that a pair of classes was used to describe the complete behaviour of any individual chromosome or gene. The classes CCommonChromosomeData and CCommonGeneData provided access to shared data and shared functionality, whilst the classes CChromosome and CGene contained independent information. Data that was unique to a particular gene - for example it's value - was stored in the class CGene directly, whilst the CCommonGeneData class was used to hold information common to genes representing the same parameter, i.e. common to *Gene[n]*, where $0 \leqslant n < NoGenes$. These classes are shown in figure 7.2.

Design variation is characterized by differences in the overall form of chromosomes, arising from individual differences in the value of constituent genes. In the last chapter, SGA switch genes that supported alternation were distinguished from standard parameter genes, used widely for modelling variations in size, shape, topology or other aspects of a design concept. Whilst in a binary encoding scheme the length of each gene (as given by the number of constituent bits) varies according to need (and may be capable of adaptation during a genetic experiment), under normal circumstances, the genotype-phenotype mapping remains static. In other words, irrespective of the number of chromosomes contained within the population, the first gene always refers to a certain design feature, the second gene refers to another different feature, and so on for every other gene. Using binary encoding, one attribute that is common to any *Gene[n]* in the population and which must be established in order to initialise the genetic experiment is the gene length (i.e. the number of bits or alleles required to encode a design variable). Supposing that the first,

second and third genes in the chromosome require four, three and two bits respectively, this can be specified using the CCommonGeneData class as follows: -

CommonGeneData[0] • Number of bits = 4

CommonGeneData[1] • Number of bits = 2

CommonGeneData[2] • Number of bits = 3

(Note that for a GA using a real-encoding scheme, it is the range of variable values and not the number of bits that needs to be stored). Given that the first gene in a chromosome represents a particular design parameter using four bits, then the first gene of the first two chromosomes in a population might typically take the form "0101" and "1011" respectively, and can be stored in the CGene class directly as follows: -

Chromosome[0] • *Gene[0]* • Value = 5 $(0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)$

Chromosome[1] • *Gene[0]* • Value = 11 $(1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)$

Used in combination, the CGene and CCommonGeneData classes described gene details concisely. For example, if a genetic experiment required a population of 500 chromosomes and each chromosome contained 20 genes, there would need to be 10000 instances of CGene (i.e. 10000 genes with 10000 independent values) but there would only ever be 20 instances of CCommonGeneData. This approach provided a convenient way of handling genes. Although chromosomes did not necessitate the same considerations as genes, their functionality was purposely implemented via a pair of matching classes, CCommonChromosomeData and CChromosome, for consistency.

It was possible to build upon these four classes to provide greater versatility in the handling of chromosomes and genes. These internal objects were specifically designed for use with the GA configuration data file. GUI tools were created to enable design configuration at a high level, and are described, hereafter. The classes used to implement gene and chromosome behaviour included member functions that enabled details to be saved to file, restored from file, configured manually, verified automatically and manipulated as appropriate to support the implementation of the GA operators.

Consideration was given as to how to provide flexibility in defining the genotype-phenotype mapping externally in order to relate the chromosomal structure to actual design parameters. Using the CCommonGeneData class, each gene was given a unique attribute

that indicated its purpose, i.e. identifying the design parameter to which it related. This attribute was referenced in order to decode a chromosome. The CCommonGeneData class contained data members that enabled a gene to be given a name and to contain a short description of its function.

7.5 Supporting Alternation and Aggregation

It was necessary to provide further information in the design configuration file and to expand the definitions of the gene classes accordingly, to support the different types of genes found in a SGA genetic hierarchy, and to be able to extend versatility in other ways. The CCommonGeneData class was allocated a data member identifying the type of gene to which it related. As different construction options require different numbers of design parameters, it was convenient to introduce a new, third type of gene called a *group gene* to assist in modelling the genetic hierarchy. The purpose of a group gene was solely to provide aggregation for a number of design parameters belonging to a particular structural system. As such, a group gene was not variable in the same sense as a switch gene or parameter gene, but simply controlled a fixed set of variable-value parameter genes associated with a particular construction option. By using group genes, switching of alternative structural systems was always executed via a single, high level gene.

The structure of the genetic hierarchy was described to the GA in terms of a series of parent-child relationships, rather than being hard coded. Genes were associated with one another through their static index in the chromosome. Switch genes were the parents of group genes or other switch genes. Group genes were both parents to parameter genes, and children of switch genes, simultaneously. The CCommonGeneData class contained attributes such as "Parent Gene ID No", and "List of Child Gene ID Nos" describing these inter-relationships. According to the type of gene, these data items were used in different ways to initialise and implement the GA.

Figure 6.5 in section 6.6 shows how design domain options relate to genes. Beneath the root gene are frame type switches, below which are compatible floor type groups and then finally the parameters associated with each individual floor type option (in situ RC, precast concrete, precast concrete, composite steel decking).

For a switch gene, the "List of Child Gene IDs" data member identified those group genes representing alternative structural solutions. It was possible to calculate the number of bits

required for a switch gene from such a list, and so dynamically configure the GA to support a variable number of design alternatives. More details follow.

For a group gene, the list of child genes indicated the set of parameter genes that were active when the group gene itself was active, or conversely, inactive when the group gene was inactive. Depending upon how design knowledge was defined, group genes could either form a permanent part of the design solution, or were activated and deactivated through a switch gene located at a higher level in the hierarchy. During the execution of a genetic experiment, only one group gene would be active beneath a switch, and its neighbours would remain inactive.

For parameter genes, it was possible to define the relationship between the gene value and the actual design parameter value in the design configuration file. In order to do this, it was necessary to specify the encoding scheme to be used along with other details that indicated whether the gene value was based on a discrete set of values or a fixed range. Where a range was specified, the minimum value, maximum value and the interval were used to determine the size of a gene.

7.6 Manipulating the Design Domain

This section describes techniques that supported the modification of the design configuration file, for altering the design domain, prior to executing the GA. Configuring the design domain was supported at two levels – by selecting permissible types of structural system and by modifying the values of design parameters associated with particular structural systems. GUI tools were used to demonstrate how the changes could be implemented at a high-level, efficiently, thereby avoiding errors and not requiring prior knowledge of the underlying workings of the program.

7.6.1 Selecting Structural Systems

Figure 7.3 shows a dialog box that was created specially, to configure the design domain. The left-hand side of this dialog box contained a tree control, which was used to present the independent genes that appear in the chromosome and defining the design domain. The tree control provides basic functionality for manipulating the nodes and branches in a hierarchy. The name of each gene was displayed in the hierarchy at the appropriate point. The tree control depicts the parent-child relationships between gene in the SGA hierarchy.
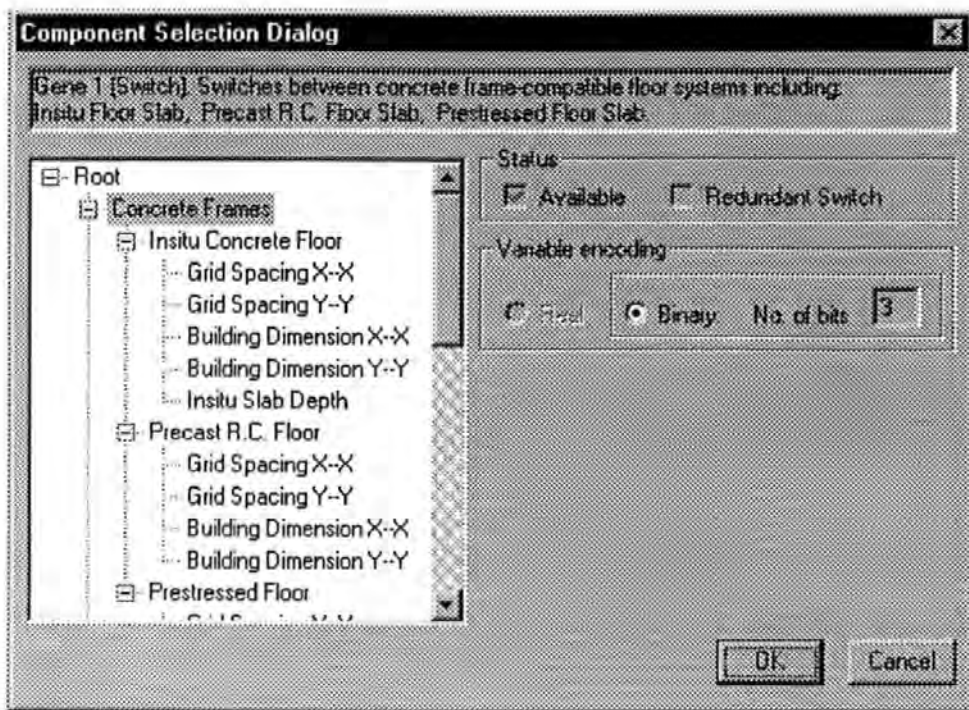
Figure 7.3: Component Selection dialog box showing Switch Gene options.
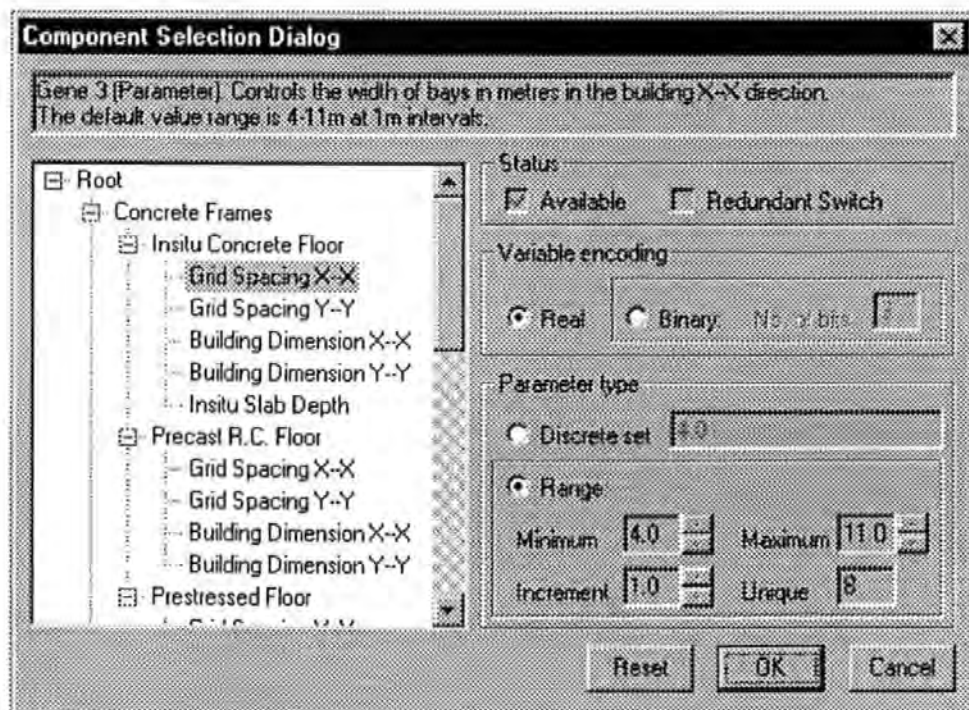
Figure 7.4: Component Selection dialog box showing Parameter Gene options.

The tree control supports several actions, including scrolling, the selection of an item, and allows nodes to be expanded and collapsed. Pham et al [130] pioneered the visual representation of a design domain, in studies involving the GA. In the present study, by linking the default functionality of the tree control to the structure of design knowledge, it was possible to effect changes to the design domain through graphical manipulation. It was possible to permit the use of certain construction options and to prohibit others. Group genes or switch genes that appeared in a collapsed state were used to represent unavailable design options. Conversely, an expanded group gene or expanded switch gene represented an available option.

The CCommonGeneData class contained an attribute that was used to indicate the status of each gene in the chromomsome, in terms of whether it was currently included (permitted) or excluded (prohibited) from the design domain. This attribute was updated to reflect changes that were made to the graphical hierarchy view. Underlying code was created to ensure that as certain design options became available or unavailable, the status of all genes associated with the same options were recursively updated via the status attribute. In essence, the tree control was used as a tool to manipulate the chromosome structure to match specific design requirements. In this way, the actual design domain effectively grew or shrank according to the parts of the chromosome that were enabled. Note that whilst different structural alternatives shown in the expanded state were permissible, their actual inclusion in a conceptual design solution was still governed by the status of switch genes, contolled either manually or by the GA process.

Switch genes maintained a static list of all of the construction options that were supported (and the genes to which they corresponded), and a separate dynamic list of the options that became permissible. The content of active list was updated after manually configuration, and allowed a GA to initialise switch genes in a dynamic manner, according to the number of options supported. By increasing or reducing the options available from which to generate a design concept, switch genes required greater or fewer switch settings, respectively. It was possible to make a switch gene redundant by permitting only one valid alternative. Code was created to handle this situation.

7.6.2 Selecting Variable Ranges

Highlighting a node in the design hierarchy in the tree control, caused relevant details to be displayed in fields above it and to the right-hand side of the dialog box. One field contained a brief description of the gene. Selecting a group gene or switch gene yielded a

general description of structural alternative, as shown in figure 7.3, whereas selecting a particular parameter gene resulted in the specific implementation details, including its genetic encoding to be displayed, as shown in figure 7.4. It was possible to interactively select different genes and to alter their details, using a combination of actions which involved clicking with a mouse and editing values in the fields provided, to alter the design domain.

Figure 7.4 shows that it was possible to specify how a particular design parameter was encoded and, via radio button selection, to specify whether a design parameter had a fixed range or would take specific discrete values. Through the GUI, it was possible to indicate that any parameter was permitted to only adopt a subset of the full range of normally available values, and in the extreme situation, to specify that a constant value should be used. The code to support this degree of configuration is lengthy. The form of the configuration file, called **config.ini**, used to store this information is shown in Appendix A.

7.7 *Modifying Parameter Values*

The parameters associated with producing a concept fall into several categories – those connected specifically with the operation of the GA, those affecting how a design concept was developed based on the chromosomal details, and those that affected the determination of fitness. Each set of parameters were handled through separate dialog boxes.

7.7.1 GA Control Parameters

Figure 7.5a – c shows various pages of information, contained within a *GA Options* dialog box, used to configure the GA. On one page, edit boxes were used to select values for GA control parameters like *PopSize*, *NoGens*, *Prob$_{Mut}$* and *Prob$_{Cross}$*. Another page permitted GA options to be configured. RW and RSS selection methods were supported. One-point, two-point methods were supported. Uniform crossover is yet to be added, and presently appears disabled. The source of random numbers was configurable. Seeded random numbers provided as an alternative to using random numbers generated using the system clock, allowing genetic experiments to be repeatable, for validation and testing purposes.[13] Specific modifications to the GA, including Elitism and Tournament Preselection, were controlled via check boxes.

---

[13] This capability was found to be particularly useful in testing the system.
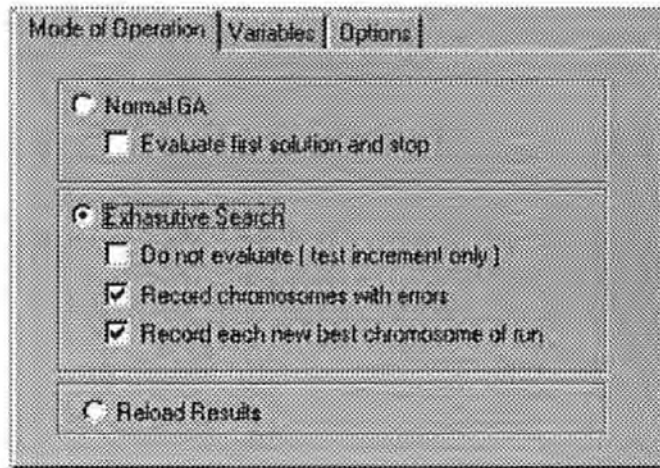
Figure 7.5a
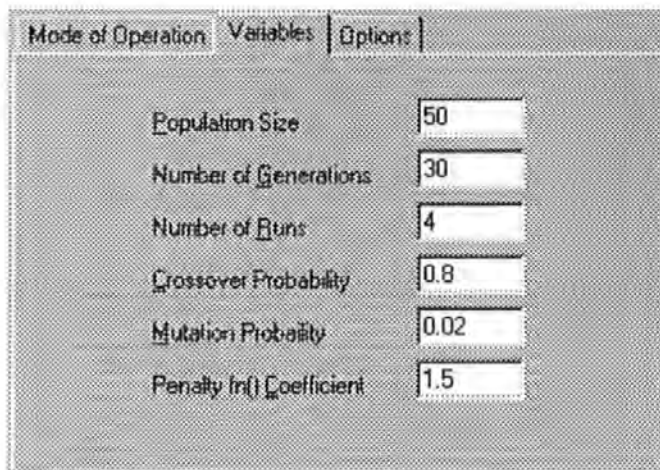
Figure 7.5b
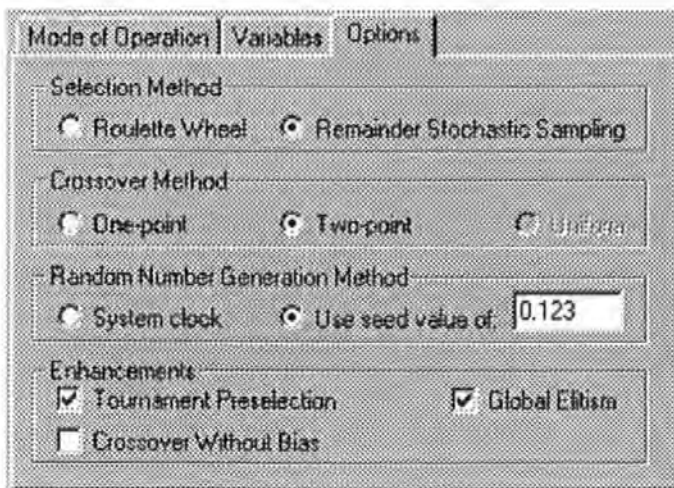
Figure 7.5c

Figure 7.5(a)-(c): GA Settings dialog box showing separate pages for specifying mode of operation, GA control parameters, refinements and other options.

7.7.2 Cost Parameters

(Note that where references are made to costs in program output the unit is given as GBP which is a conventional notation in business used for Pounds Sterling).[14]

Figures 7.6(a)-(e) show various pages of cost data, contained within a *Cost Options* dialog box, that enable the costs associated with different structural design alternatives to be manipulated. Individual pages were provided for components and materials associated with construction requiring involving structural steelwork, in situ RC concrete, precast RC concrete and prestressed concrete. A separate page was used to configure unit costs associated with providing roofing, external cladding and foundations and the purchase of land. The perceived revenue income that the building could be expected to generate, is also shown, in $£/m^2/year$.

The unit costs data used in this study were developed from standard price books and was supplemented with information supplied by a firm of Chartered Quantity Surveyors, with whom the School of Civil and Structural Engineering has had contact. Cost values were selected to represent the total cost associated with a particular structural system or component, based on cost price, cost of storage and cost of labour needed in fabrication, fixing and finishing. Note that whilst the cost data applied in this study was intended to be as realistic as possible at the time the system was designed, the costs can easily be modified to respond to fluctuations arising from inflation, material shortages, bulk purchasing and other reasons.

Cost data is applied in specific design case studies in the following chapter, where its significance is reiterated.

7.7.3 Miscellaneous Design Parameters

Figure 7.7 shows a *Miscellaneous Options* dialog box that presents miscellaneous structural and other design parameters, introduced in chapter 6 and required to developing conceptual structural design, flexibly.

7.8 Current Approach to Support User Interaction

The GUI tools and mode of operation of the GA design system serve important functions. The interface allows the user to conduct a parametric study for deeper understanding of the process(es) involved. Note, this relates in part to enable the architect to appreciate

Figure 7.6a

Figure 7.6b

Figure 7.6c

Figure 7.6d

Figure 7.6e

Figure 7.6(a)-(e): Cost Information dialog box pages for different structural systems.
The last page permits the configuration of roofing, cladding, foundations
and land purchase unit costs, and perceived revenue income.

119

structural engineering implications. Parametric study can also supplement experience in a specific deiscipline; the structural engineer can use a parametric study to confirm that a viable design solution is efficient.
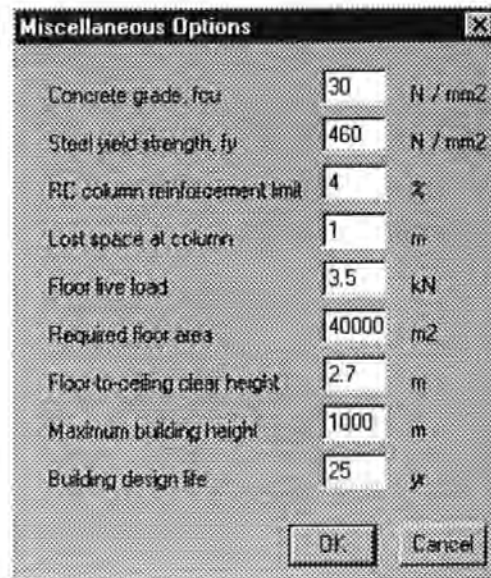
Figure 7.7: Miscellaneous Options dialog box showing structural design parameters and other design parameters.

Flexibility offered by the system allows users to switch on or off every branch of the design hierarchy to investigate different design solutions quickly and easily. Furthermore, it is possible to investigate the effect of changes of one parameter on one or more whole design concepts. This can be achieved, for example, by changing the cost of any design aspect, like in situ concrete or formwork. As a consequence the system has the potential not only for knowledge representation and processing, but also for supporting the creation of new knowledge.

With regard to the development of the University of Stathclyde ISDS, described in section 2.2.3.4, McLeod et al [131] recommended that design systems should be capable of operating in 'practitioner mode' in which the designer interactively controls the process. GUIs that contain graphical controls, and OOP mechanisms used to implement them, have empowered users even where automatic processing is involved. Accessibility raises user confidence. As well as offering greater interaction, from a user's perspective GUIs can help make computerised 'closed black box' processes more transparent. This is particularly relevant given the unconventional and stochastic nature of EAS methods such as the GA, as compared with more familiar software, like CAd programs, for example.

---

[14] It literally stands for Great Britain Pound.

The kind of support that it is appropriate to provide during the operation of a GA depends greatly on the philosophy behind how it is being used and by whom. It is one thing to show that the GA performs well in specific test problems and another to be able to demonstrate that the technology can be transferred into a viable, general-purpose tool for a building designer to use. The role of the GA in relation to other DS processes must be considered. The duration of a genetic experiment is another factor that must be considered. It is sometimes appropriate to sacrifice computational efficiency for versatility (i.e. a versatile algorithm may be marginally slower than a dedicated, hard-coded one.). The fact that the GA performs better or worse that other techniques is not the main focus. Efforts taken to produce an optimal solution should be commensurate with the real benefit that it bestows. In this research, the benefits of versatility far outweight computaional speed differences.

Already, it has been shown how the designer can be involved in the configuration of a design problem. The same consideration was extended to the execution of the GA. Within a DS system, it was possible to initiate the GA upon command. The designer is not necessarily a passive observer in the search process but an active one, capable of interpreting results and adjusting them for practical gain. Various ways were considered in which the designer might co-operatively produce suitable concepts. This starts with consideration of basic means of providing feedback to the designer to monitor the performance of the GA and progressed to consider ways of enhancing this feedback to suit the designer and to permit interaction.

7.8.1 Non-Interactive Support

The type of information that is useful includes how far the GA has progressed (i.e. how close it is to completion), and how well it performs. Whilst the optimal solution is not known *a priori*, displaying details such as current generation number together with fitness values, progress can be monitored. Floor planning experiments showed that information needed to presented to concisely to give useful feedback to a designer in real-time, during the execution of a GA. Specifically, the use of a convergence graph or visual representation of the best-solution-yet gave a clear impression of performance and incremental changes (see Figure 7.1). It is usual to investigate individual design solutions after the algorithm has terminated.

7.8.2 Real-time Interaction

Programs that are able to respond immediately help to make users feel in control.[15] As its name suggests, the phrase real-time software applies to programs that are required to (and hence, designed to) respond in real-time to users' needs. Whilst MS Windows programs support interaction, GAs are generally automatic processses. Assuming a constant hardware specification, *ChromoLen, PopSize*, fitness function complexity and the stopping condition all influence the computational effort, and hence the time, that is required to complete a genetic experiment. This raises the question of how user control may be provided appropriately, i.e. without significant performance degradation.

OSs like MS Windows 98 and Windows NT4 support multithreaded programs in which multiple processes can be executed simultaneously. It is possible to implement a standard genetic experiment and at the same time provide external control. A genetic experiment can be initialised and terminated as a separate process from within the main design system, that controls it. The capability to pause, re-start, or reset the GA was implemented. An approach in which the designer takes control of the search / optimisation process infers that it might be beneficial to extend, shorten, terminate or repeat a genetic experiment based on current progress, and allow design variables to be modified. Icons were used to create a simple toolbar for performing these actions.

7.8.3 Output

A number of output views were supported. A small status bar displayed scrolling text messages to indicate progress textually. A window containing a convergence graph, was used to show progress. Upon completion of each generation, the fittest design was plotted. The scale of the graph was automatically recalculated to emphasize relative changes in fitness between successive generations. A second, scrollable output window was used to show the current best design concept. A radio button enabled the user to select whether to explain in detail how a concept had been generated, or whether to summarise the most important features of the succeeding concept.

7.8.4 Modes of Operation

The fact that the GA concept generation requires extensive design modelling to generate alternative concepts also supports the evaluation of individual concepts, local search and exhaustive search, within a specified sub-domains. Figure 7.8 shows the stages in these three modes of operation. Since any design could be developed from the chromosomal
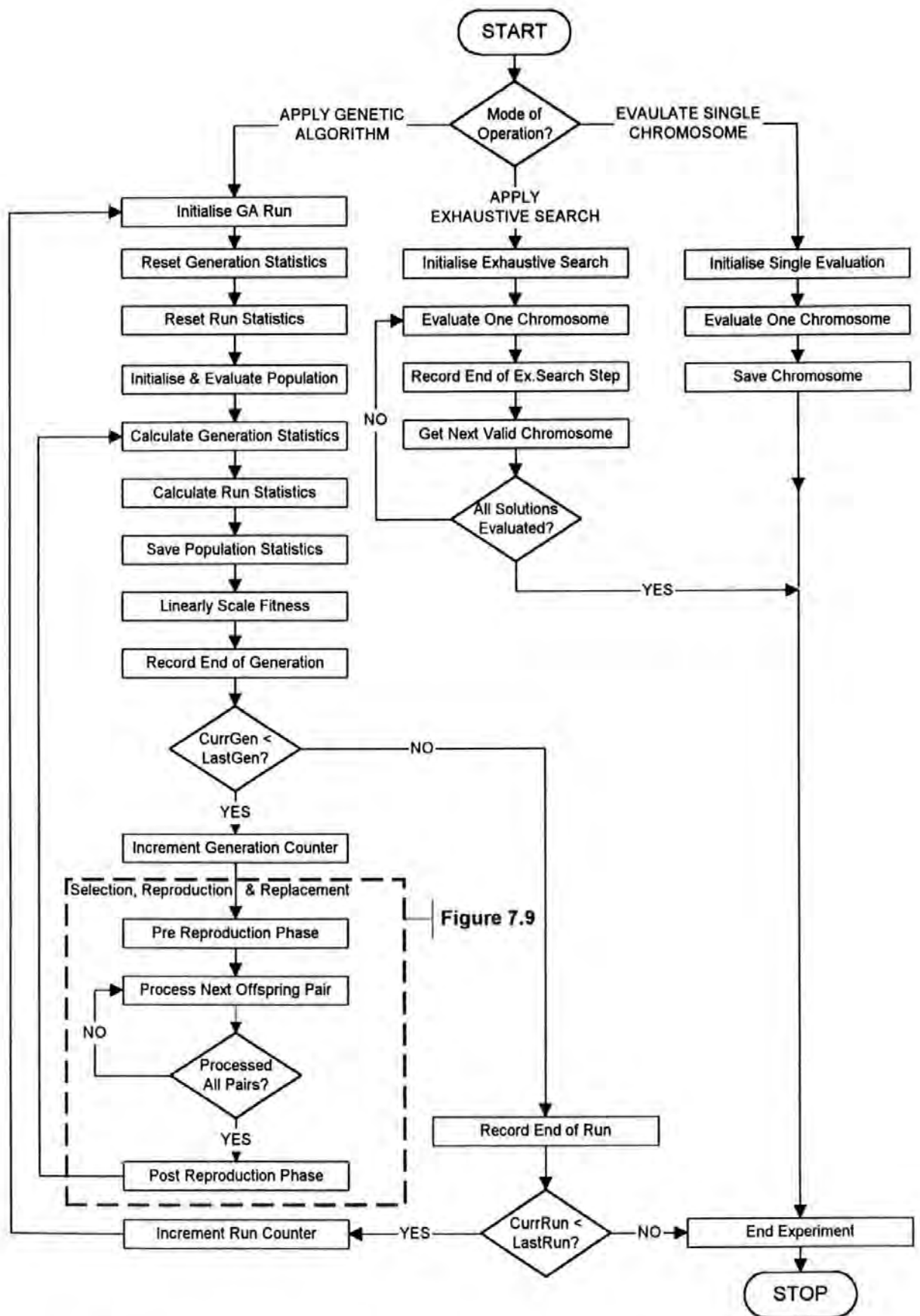
Figure 7.8: Flowchart showing steps in the GA, exhaustive search and single chromosome evaluation process.

details and given design parameters, this was all that it was necessary to store to be able to recreate a design. The details of each run were stored in file, called **runXXXXX.dat** where **XXXXX** represents the run number, for example **run00001dat, run00002.dat** and so on. Appendix F shows an example of a run file. The system was modified to support different modes of operation, including a review mode, so that instead of invoking the GA, any individual concept could be re-loaded and re-examined.

Figure 7.9 shows details of the steps involved in reproduction, shown boxed in figure 7.8. One noticeable change in the implementation sequence from that of a Simple GA is that, crossover, mutation and replacement is performed for one pair of chromomsomes at a time, rather than crossover being applied to the whole population, followed by whole-population mutation and so on. A technique was created to allow offspring chromosomes to re-enter the population directly. This avoided processes in which the design population is copied / overwritten in its entirety, with the new population of designs replacing the old one. OOP techniques were employed to safeguard parent chromosomes required for reproduction from being overwritten prematurely.[16]

Three factors have contributed to the external appearance, internal architecture and capabilites of the design system. Firstly, as a general rule, software systems should be designed with the user in mind. Second, there are many opportunities for errors to be inadvertently introduced into a complex design model. Thirdly, some functionality was specifically created to assist during development and testing phases. It was considered a useful capability to be able to report the actions of the GA for validation purposes, and yet to avoid major time delays.

Since stages in the GA and fitness function were extensive, they posed a logistical problem to test. Controls were provided to allow the GA to reach a point of interest, to be studied in detail, and at that point, for execution to pause. Execution could be resterminated, manually. The GA supported the capability to repeat the same genetic experiment (with different starting populations) for a given number of runs; a feature useful both in trials and real experiments. Using interactive controls, it was possible to allow the GA to advance a certain number of runs, generations or steps and then to pause itself. Diagnostic details were accessible and could be switched on or off. An unusual feature was the ability for a user to introduce a delay in the GA so that it could be run at such a speed as to allow the

---

[15] Actually, a reasonably short time delay qualifies as interactive behaviour.

tester to validate each step, (i.e. providing interactive debugging / checking), without there being the need to have to constantly start and stop the process.
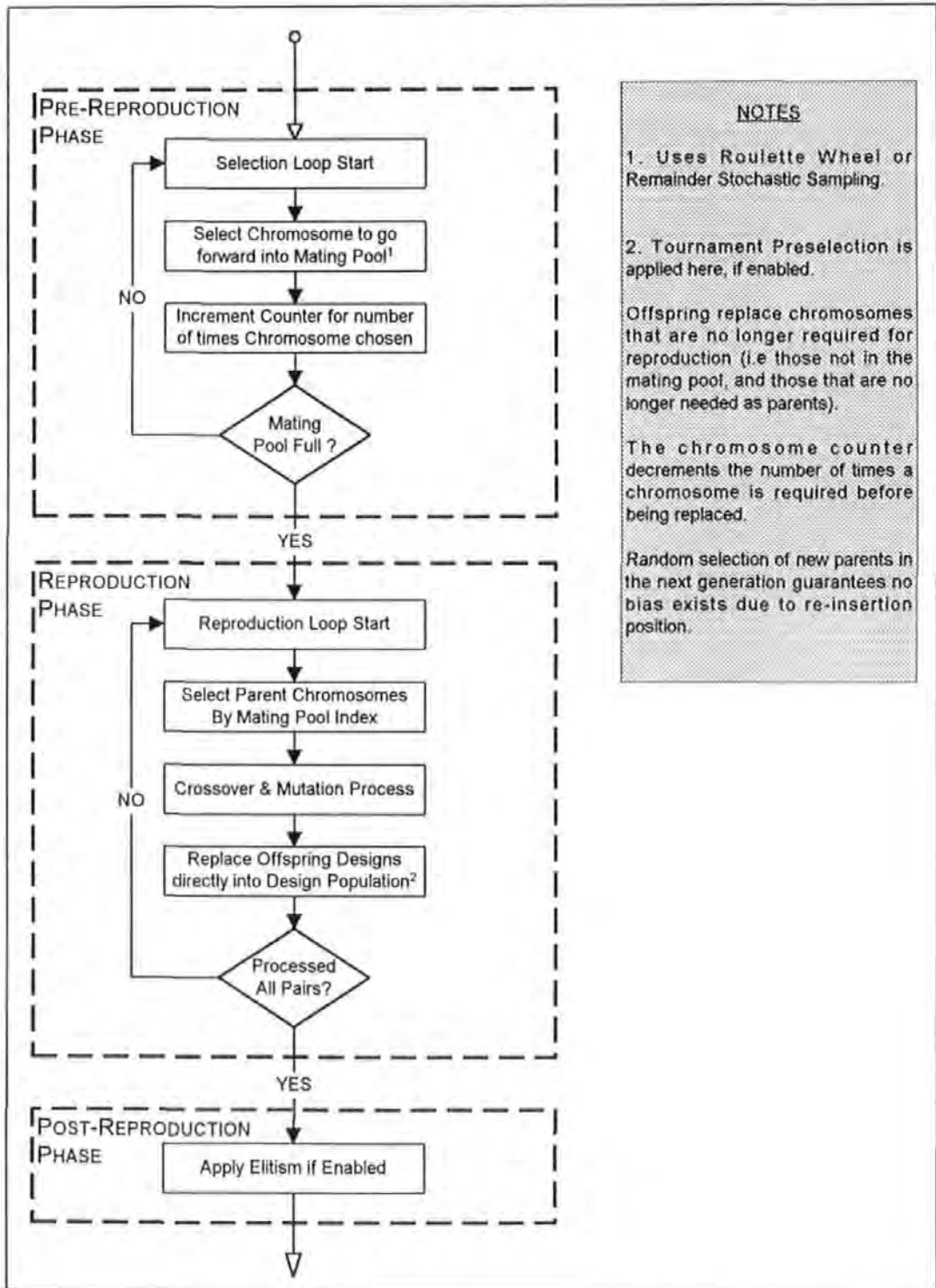
Figure 7.9: Reproduction details.

---

[16] Another enhancement was implemented in crossover and mutation operations. Integer data types were manipulated directly via bit-shift and logical operators to avoid character string conversion. This reduced the number of separate steps in the GA and helped make it faster.

A side-effect or by-product of the way in which the system was implemented, was that it became possible to evaluate an individual design concept without employing the GA. This is a useful capability in its own right. The modes of execution of the design system can be summarised as follows: -

- Apply the genetic experiment in the standard manner.
- Apply ES to a specify domain and/or design variable ranges.
- Evaluate an individual design concept.
- Reload and/or review current or previously saved results.

Although ES is slow and computationally expensive compared to the GA, it can be viable in the following circumstances: -

- when a design space that is normally extensive is much reduced because of case-specific constraints,
- when the designer expressly wishes to examine the options that lie within a narrow sub-domain of a larger design space,
- when it is necessary to obtain an absolute optimum design that can be used to help assess the accuracy, reliability and robustness of the GA,[17] and
- during fitness-function development and testing, for ensuring that all combinations and paths by which various design-specific routines may be called never produce unexpected (unhandled) results that might affect the genetic experiment.[18]

7.8.5 Revising a Design

The GA can be used to draw attention to regions of search worthy of further investigation. In some circumstances, this may be may be sufficient to enable a designer to recognise the nearest practical configurations to which the GA alludes based on experience and to refine a conceptual design solution, accordingly. The convergence graph view was modified to provide post-processing support by making it possible to examine solutions using cursor picking. By clicking the mouse of the graph, the corresponding chromosome was retrieved and re-evaluated, and the details of the concept were shown in the second window. This facility was also made available during a genetic experiment while the GA was paused.

A separate capability enabled multiple models to easily be configured without relying on the file management provided through the OS. The ability to re-load, re-display

---

[17] In particular, where the optimum solution cannot be derived by other means at the outset.
[18] Note whilst some concepts generated are feasible, others will naturally be infeasible, and this is not problematic if handled appropriately.

convergence plots and re-evaluate design solutions stored as chromomsomes from previous GA runs was of benefit. Figure 7.1 shows three push buttons on a side panel of the interface. The EVALUATE button supported manual review of design concepts. By clicking The RECORD button, the user is able to save the full details of a particular design concept to a data file, called **details.dat**, an example of which is shown in Appendix C. Similarly clicking the EXPORT button created an intermediate data file containing the major dimension of the design concept. This file, called **design.dat**, acted as the input to a program that created the geometry data needed to visualise a design. An example of the **design.dat** file is shown in Appendix D. The geometry builder program creates a geometry data file, called **geometry.dat**, which contained the data for visual presentation. An example of the **geometry.dat** file is shown in Appendix E.

7.9 Graphics and Visualisation for Post-Optimality Support

An important goal of a DSS is to permit design information relevant to a task to be clearly communicated to, and be understood by, the designer. Buildings contain a large amount of spatial information and in supporting CBD, this project also considered issues relating to the conveyance of design information in an effective manner. The amount of design information generated grows with the level of detail in the design model. It can be easier to present information to designers using a combination of graphical and textual techniques than using text alone.

Sisk et al [101] demonstrate some of the possibilities for enhancing basic features using graphics including: -

- Using a graphical image to make the input parts of a design easier, or to provide a means of verificiation. A graphical image of a building floor plan would be useful to allow a designer to manipulate locations of fixed features, such as a service core or atrium quickly, easily and confidently.

- Using a graphical image to provide supplementary help information, for example, to show the user how different systems appear, how they are constructed and how they relate to other parts of the design concept.

- To summarize results – like a BoQ – or design proforma, in a readable manner.

Various parties commented on the significance of emerging graphical techniques for supporting designers. Virtual reality systems and photo-realistic graphics have attracted considerable interest. This type of software is costly, and as Plant [132] recognised, whilst holding exciting possibilities for the future, without additional specific knowledge it tends

to be of limited use for supporting engineering aspects of design directly. Immediate application in architecture presentation and pre-planning is fast being supplemented with intelligent design tools based on OOP techniques, see Russell [133]. Whilst it is becoming increasingly possible to transfer parts of design models between draughting, analysis and modelling software, general-purpose commercial systems tend to limit the support available for non-standard activities and such systems are rarely designed to allow a user to modify them for their own needs. AutoCAD® is an exception, but even then it permits only limited customisation of its interface and user-defined functions. Plant [132] asserted that other task-orientated capabilities of computers with less visual appeal continued to support engineering design activities more effectively. Taffs [18] concurred that the quality of computer graphics required for structural engineering in general need only be of a standard sufficient for providing an adequate image, for the purpose of design development.[19]

This project developed a proprietary graphics module to study the usefulness it affords in presenting design concepts.[20] The program filters the design information generated from the GA search and converts it to 3D graphical data for displaying the building form. It takes a different approach to conventional CAd software, used to prepare detail design drawings, and visualisation software, used principally to assess different spatial layouts containing functional features. A wire-frame model offering two-dimensional (2D) and three-dimensional (3D) views was created to give an impression of the building. The 2D view shows the building in plan and in section. The 3D view can show the building from any angle. Figure 7.10 and figure 7.11 shows the visualisation module containing building concepts displayed in 2D and rendered in 3D, respectively.

Consideration was given to convenient and efficient means of manipulating a building form, and for presenting pertinent design details. Clearly many possibilities exist that are beyond the scope of this project. One capability that was achieved was in associating structural components visually with their corresponding conceptual design details. Using interactive graphical selection (cursor picking) it was possible to call up the details of parts of the design solution, including component quantities and cost estimates. This provided information to the user, about the details of the components that made up a design concept, in a convenient manner.

---

[19] In this regard, Taffs noted certain programs, executed on older hardware, were helpful to designers before being upgraded to use the latest GUI enhancements found in the later OSs.
[20] Note the usefulness for presentation purposes was also a consideration.
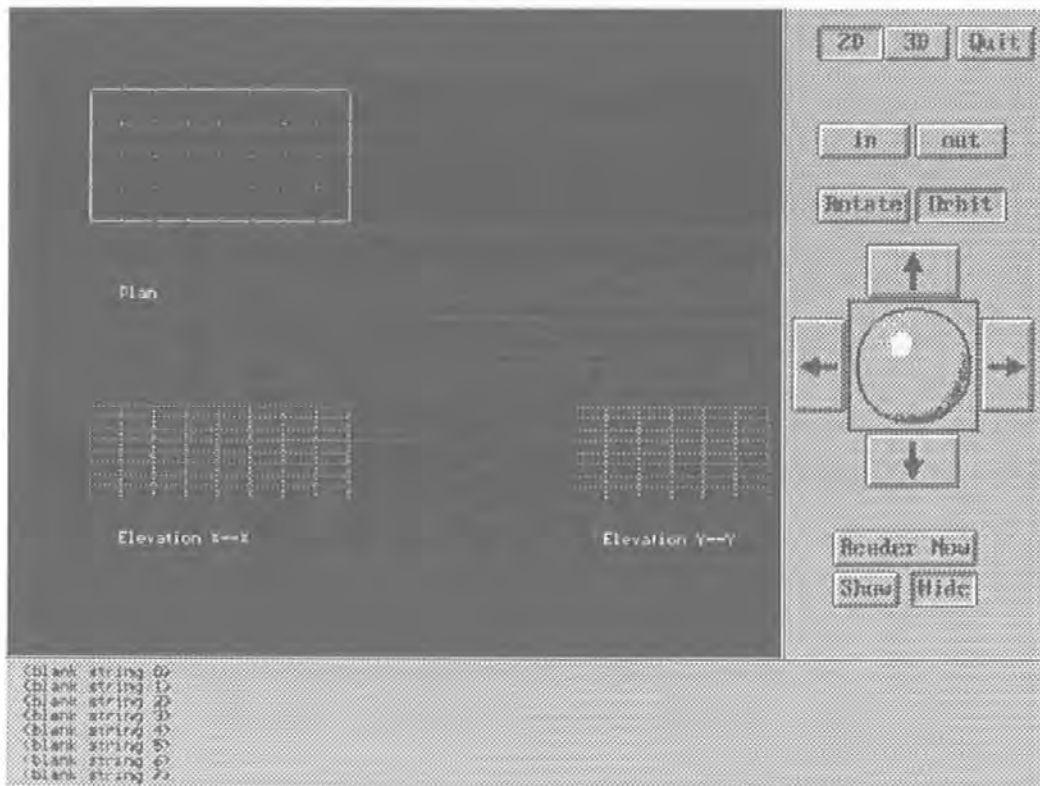
Figure 7.10: The visualisation module showing 2-D views of building concept.
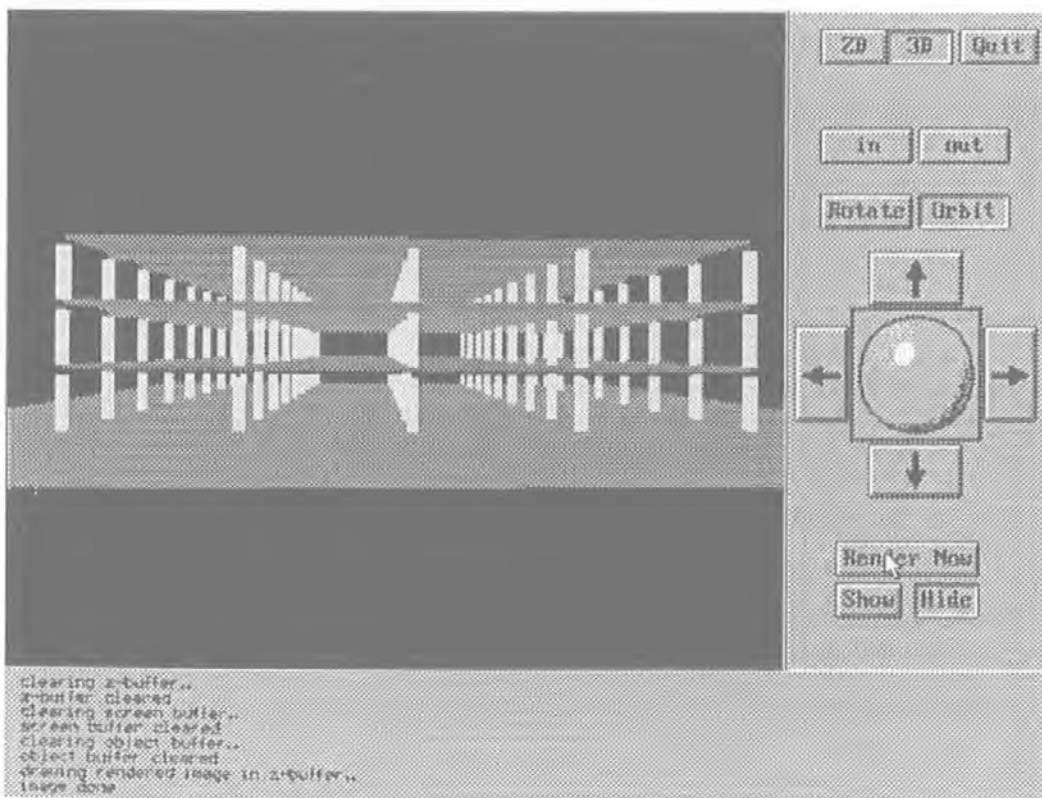
Figure 7.11: The visualisation module showing a 3-D rendered view of a building concept.

(As an aside, note the genetic experiment actually used several objects based on the CChromosome class, not shown in figure 7.2. In addition to the design population of CChromosome's, two additional copies were used to store offspring and one further chromosome was used to implement Elitism. It was also possible to maintain details of the best chromosome produced during successive generations of the genetic experiment using additional copies held in separate data structures).

8 Examples, Capabilites and Discussion

8.1 Introduction

At the start of CBD activity, the designer may welcome any advice and suggestions that can help to identify useful design alternatives. In attempting to offer support, early KBES tools adopted the *top-down* approach to design. In this approach, important, high-level knowledge having implications upon lower-level options was synthesized in order to refine a large search space. Making the designer more aware of favourable construction options and potential conflicts early on, were amongst benefits reported of the top-down approach.

The opposite of the top-down approach to design is the *bottom-up* approach. The bottom-up approach reflects current practice. It describes the situation where designers specify their requirements in order to satisfy the design brief. Normally, the design brief contains conditions that support the use of certain dimensions, structural systems and materials, and precludes or discourages others[1]. It becomes the duty of the design team to seek highly satisfactory design solutions, containing or adopting any fixed or prescribed aspect[2]. The need for software techniques to accommodate such activities in a flexible manner has consolidated research efforts in recent years.

Where choices exist, it can be beneficial to assess the implications before making a decision. In selecting a floor plan, for example, an architect could be greatly assisted in knowing its implications upon the structural system. This chapter demonstrates the capabilities of the design system based upon the GA to address both top-down and bottom-up approaches to design. It also aims to demonstrate the usefulness of the system as a parametric design tool, made possible through an efficiency search strategy. In the system, conceptual design tasks are formulated as genetic experiments using several data sets. The design domain is one aspect that is described in the data set. The data set also includes control parameters, design parameters and unit cost information. The control parameters govern the operation of the GA. The structural design parameters are used to develop appropriate design solutions from the content of chromosomes. Cost information is used as the principal fitness indicator. Each of these aspects has been made configurable, using suitable interface tools and external data files. In combination, these factors present huge scope for variation.

---

[1] There are usually also supplementary factors that promote certain viable systems and to reject others, outright.
[2] i.e. compatible with those same features.

8.2 A Default Design Scenario

There is no such thing as a typical design situation. Nevertheless, it is convenient to use a default test case as a datum against which comparisons may be drawn. Information pertaining to such a default scenario is presented in full in the **settings.ini** file listed in Appendix B. Briefly, the default design scenario represents a large, speculative office development - a building is required that can provide 40000m^2 of occupancy space, for typical office loading (3.5kN/m^2), with land costs set at £5000/m^2. The structure is expected to generate an annual revenue income of £80 per square metre of <u>net</u> lettable floor space, (i.e. £80/m^2/yr). It has a minimum design /service life of 25 years. The problem is formulated as a maximisation-of-profit[3], and profit is determined by subtracting the capital cost of the structure from the total revenue income. The capital cost represents the structural frame and foundations. Unit costs for different structural components, land purchase and the perceived revenue are required as input, indicative of real design considerations. Default values are given in Appendix B.

The default design scenario permits all construction alternatives to be used for developing concepts. All component sizes are valid. In addition, there is no rigid restriction upon any internal or external building dimension, i.e. upon footprint size, grid configuration or the height of the structure. However, a poor structural grid will significantly increase the amount of lost (non-lettable) space near columns. It was considered that utility could be affected up to one half of a metre away from a column, and a reduction in the net floor space available was applied accordingly. The floor space requirment was intentionally set to a high value so as to enable variation to be applied in generating different design concepts. (A lower requirement would provide fewer geometric alternatives). It is recognised that this amount of space can generate high-rise structures with a relatively small footprint, which are uncommon in the United Kingdom. (Canary Wharf, in London's Docklands is one example that fits this description). Subsequent design examples introduce a lower space requirement, in line with the majority of modest office developments.

Lateral load resisting systems were beyond the scope of this study but are an important design consdieration, especially for tall structures. For the concepts presented, it has been assumed that shear wall bracing or structural cores would be incorporated to provide adequate lateral resistance subject to wind loading and vibration.

In the default scenario, binary encoding, RSS selection, two-point crossover, elitism and tournament preselection were all employed. $Prob_{Cross}$ and $Prob_{Mut}$ are set at 0.80 and 0.02, respectively. The population contained 50 chromosomes, a run is terminated after 30 generations, and a genetic experiment used four runs to attempt to show consistently or otherwise in the results obtained.

8.2.1 Validity of Cost Functions

In the current research, costs functions have been created to provide a relative measure of the fitness (or, suitability) of different layouts and structural systems. Although the unit costs are based upon actual costs, the project has necessitated a more coarse approximation than is usual in the consideration of major items. Apart from this research work, cost modelling has been the subject of much study. Like construction sceduling, cost modelling is a field studied in its own right. Many studies concerned cost modelling assess direct construction costs. Few studies incorporate cost information for generative design purposes to the extent attempted in this research.

Actual construction costs may vary considerably from those used in these case studies; and as such the values that have been chosen should not be taken as typical. The present approach has listed it sources of cost data; in practice this could be supplemented using data from existing building projects. The DPRO systems accommodates variable costs. The current research considered the capital cost of the structure, including the cost of the structural frame, building envelope, foundations and cost of land purchase. Expected revenue income and building service life were introduced in a very general way to differentiate the profitability of different design concepts.

To produce a more accurate model, the costs associated with design consultancy, site investigation, maintenance and running costs would need to be studied in order to obtain more realistic life-cycle costs. Furthermore, detailed cost modelling should take int o account borrowing, repayment and interest rates. Whilst this was beyond the scope of the present study, the incorporatation of such consideations would appear to be simple and unlikely to significantly alter the effectiveness of the overall approach presented, based upon the GA.

---

[3] or, equally requiring minimum expenditure, where the building is not in the commercial sector.

8.3 Example 1 – Unconstrained Design

The default design scenario describes a top-down design task. There are no external factors influencing the best structural system; in other words, there is no preference, no system is excluded for any reason. The situation represents a city-centre site, where a large plot is available at a premium.

Figure 8.1 shows the convergence plots of the best (most profitable) solutions produced for a series of runs. The graph highlight several points. Rapid convergence can be seen to be taking place within the first 10 generations, as is characteristic of the GA. By generation 30, two of the four runs identify a common solution. The best solutions produced in each of the four runs have absolute fitness values of 72086.73, 72086.73, 71954.23 and 71917.88.
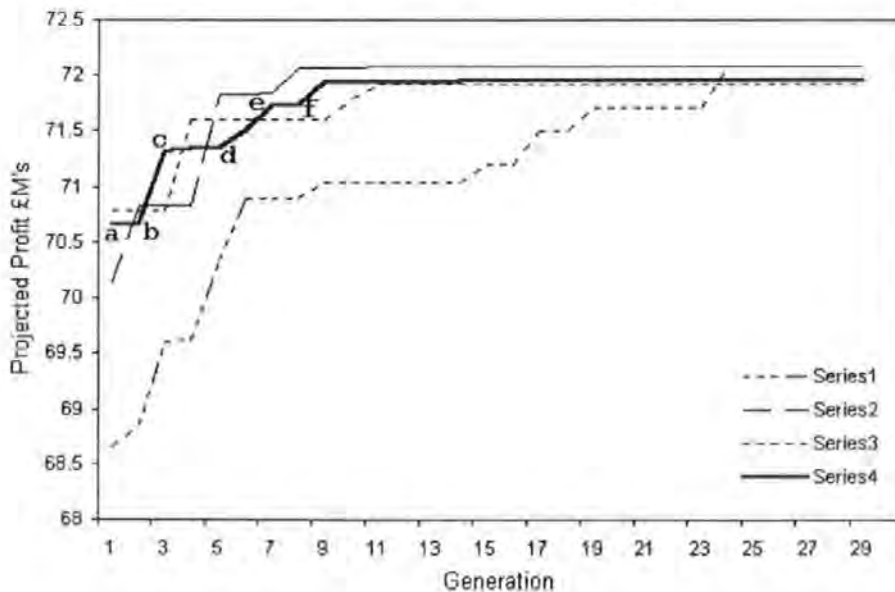
Figure 8.1: Convergence plot, Example 1.

ES was used to determine the optimal solution. It was found to be 72086.73, as achieved twice in the test. In one run the optimal solution was reached by generation 11, in the other by generation 24. There are 262144 different chromomsomes. Table 8.1 shows how this number is calculated. The exhaustive search process took nearly 6½ hrs to examine 262144 designs (as shown in table 8.1), whereas each run of the GA performed 1500 evaluations and took less than two minutes. One GA run is equivalent to evaluating 0.6% of the entire design space and four runs represents a search of only 2.3% of the entire design space. Clearly the GA appears to be efficient.

Frame type	Floor type	Combinations		
RC	IS, RC	$2^{(3+3+4+4+3)} =$	$2^{17} =$	131072
	PCC	$2^{(3+3+4+4)} =$	$2^{14} =$	16384
	PSC	"	"	"
Steel	PCC	"	"	"
	PSC	"	"	"
	CSD	$2^{(3+3+4+4+1+1)} =$	$2^{16} =$	65536

Σ 262144

Table 8.1: Derivation of number of chromosome combinations, from parameters associated with alternative structural frame / floor systems.

The optimal design solutions was as follows: -

Frame type: **steel.**

Floor system: **composite steel decking, 3m spans.**

Dimensions: **35m x 20m footprint, 165m tall.**

Grid: **4 bays at 8.75m by 3 bays at 6.67m.**

Floors: **58.**

Two secondary beams are introduced per panel (bay), parallel to the short span direction to compensate the short span of the steel deck floor. Steel columns are designed to be spliced at four-storey stages. It can be seen that the bays in the optimal solution are of practical dimensions. Column details for the given maximum profit design solution are shown in Table 8.2.

Figure 8.1 points (a) through (f), indicating fitness (profit) of design concepts produced from generation 1 to generation 6 for one run show the optimum design evolving and improving. Table 8.3 gives details of the design progression. Points (d), (e) and (f) clearly show different structural systems being identified through the SGA switch mechanism. Design progression provide insight into the problem. In the system the user (designer) is able to 'pick' points off of the convergence graphs, making it easy to perform an examination of a convergece plot.

Note, the profit figures are only a relative measure. They are abnormally high because the only outgoings considered have been the acquisition of land, superstructure cost, foundation cost, external cladding, roofing, all in an approximate manner. In practice, other important aspects of design, such as the cost of building services, finishing, furniture and external work (landscaping, car parking), rent, running costs and maintenance costs would need to be studied to obtain a more realistic profit. It should be noted however, that the Cost Model Study by Goodchild [112] showed that the opportunities for economic design are mainly in the structural design, and to a lesser extent, in the use and choice of fire casing and cladding systems. Relatively expensive aspects, like mechanical and electrical services (like air conditioning units and lifts) showed surprisingly little cost variation across buildings of similar type / dimensions. The cost involved in providing the features considered in the study – namely, the structural frame and cladding – appear to generate realistic variation in costs by realistic amounts shown in the convergence plot of figure 8.1.

Floors	Column Stage	UC Section [m] x [m] x [kg/m]	Axial load MN	Estimated cost of all columns / £ K
56 to Roof	15	0.152 x 0.152 x 23.000	0.38	1.9
52 to 56	14	0.204 x 0.206 x 52.000	1.85	8.6
48 to 52	13	0.209 x 0.222 x 86.000	3.32	14.2
44 to 48	12	0.368 x 0.356 x 129.000	4.79	21.3
40 to 44	11	0.265 x 0.289 x 167.0	6.26	27.6
36 to 40	10	0.314 x 0.340 x 198.0	7.73	32.7
32 to 36	9	0.395 x 0.381 x 235.0	9.21	38.8
28 to 32	8	0.322 x 0.365 x 283.0	10.7	46.7
24 to 28	7	0.403 x 0.406 x 340.0	12.1	56.1
20 to 24	6	0.407 x 0.419 x 393.0	13.6	64.8
16 to 20	5	"	15.1	"
12 to 16	4	0.412 x 0.437 x 467.0	16.6	77.1
8 to 12	3	"	18.0	"
4 to 8	2	0.418 x 0.456 x 551.0	19.5	90.9
Ground to 4	1	"	21.0	"

Table 8.2: Column details for the given maximum-profit design solution.

Generation	1 (a)	2 (b)	3 (c)	4 (d)	5 (e)	6 (f)
Frame type	Steel	Steel	Steel	Steel	Concrete	Steel
Floor type	Composite Deck	Composite Deck	Composite Deck	Composite Deck	Precast panels	Composite Deck
Footprint	45mx40m	60mx30m	55mx30m	55mx30m	30mx25m	55mx25m
Grid, x	8 at 5.63m	9 at 6.67m	5 at 11m	5 at 11m	3 at 10m	5 at 11m
Grid, y	4 at 10m	3 at 10m	4 at 7.5m	4 at 7.5m	2 at 12.5m	3 at 8.33m
Deck type & span	NWC, 2.4m	NWC, 3m	NWC, 3m	NWC, 3m	n/a	LWC, 2.4m
Bldg. Ht.	66m	66m	71m	71m	155m	86m
No. Stories	23	23	25	25	54	30
Lettable floor space	40188m^2	40316m^2	40350m^2	40356m^2	38560m^2	40417m^2
Profit, £K	68,657	68,851	69,600	69,621	70,342	70,900

Table 8.3 Best-of-run designs, showing design progression during first six generations of a genetic experiment.

Figure 8.2 shows the structural grid for the optimum design. Figure 8.3 shows beam details at the 20th floor.

8.4 Example 2 – Semi-Constrained Design

In the second example, RC is specified as the primary construction material to be used for the structural frame. To reflect this change, the design domain is modified so that all construction options compatible with steel-frame structures are removed from consideration. (Cladding and foundations are given higher values. The cost of precast concrete is amended). As highlighted in the previous chapter, a suitable interface enables a user who is not conversant with the mechanics of GA to easily effect this change. Disabling steel frame options leaves 163840 unique chromosomes. Figure 8.4 shows the results produced by a series of runs. The same best solution, having fitness 68648.14, was produced by all four runs. Exhaustive search confirms this to be the optimum solution, for the given design parameters.
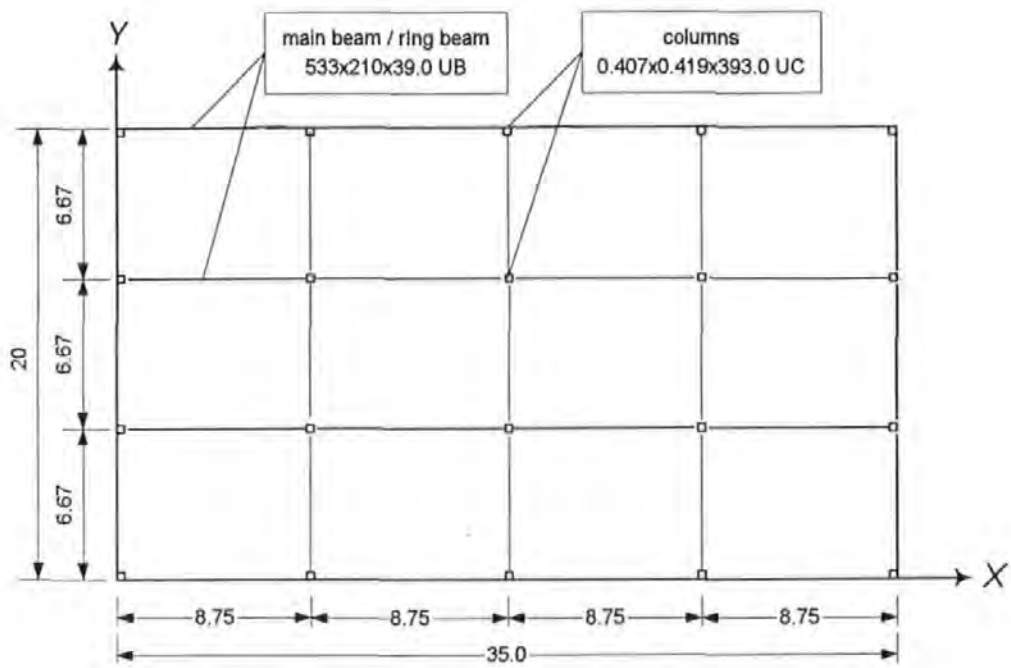
Figure 8.2: Structural grid layout, floor 20, from Example 1, solution by GA.
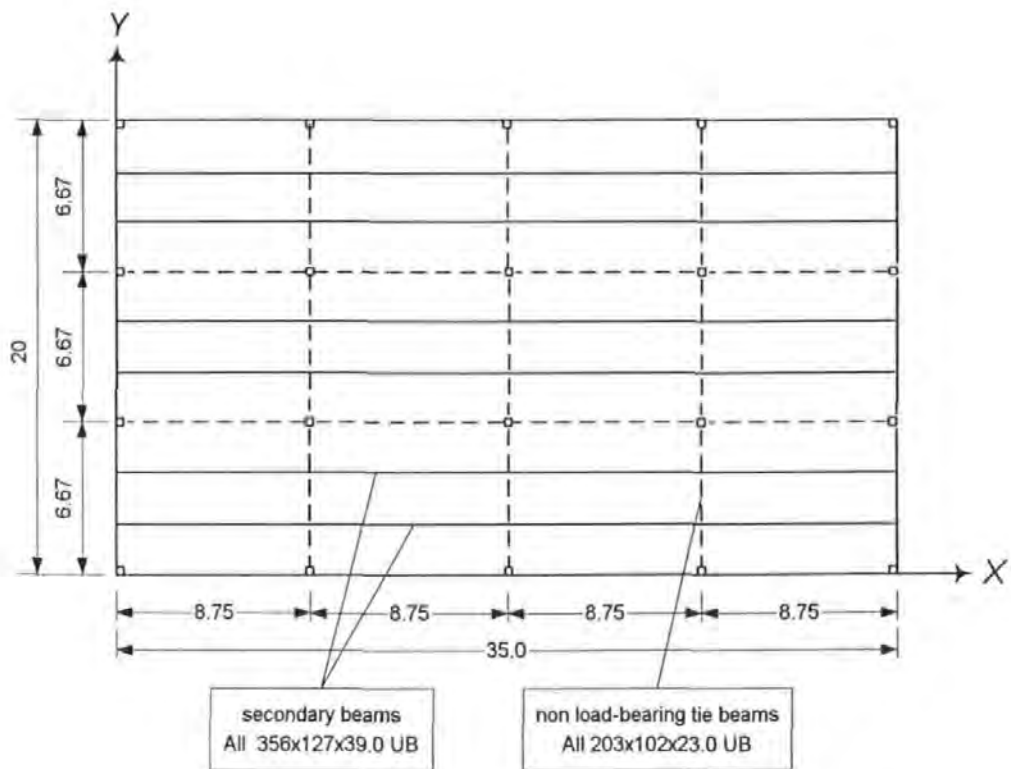
Figure 8.3: Floor system detail showing beams, floor 20,
from Example 1, solution by GA.

The optimal design solutions was as follows: -

     Frame type: **RC.**

     Floor system: **precast concrete panels.**

     Dimensions: **35m x 30m footprint, 112m tall.**

     Grid: **3 bays at 11.7m (equally spaced) by 3 bays at 10.0m.**

     Floors: **39.**

One secondary beam is used in this solution, so that the precast concrete slab design span is 5m. Notably the footprint is slightly larger than Example 1. This suggests the presence of a trade-off between high land cost and significant amount of lost space due to large columns, required to support a taller structure. Note the columns are RC, not UC sections. Once again, a practical grid is created. Concrete columns are spliced at two-stage intervals. Figure 8.4 shows convergence plots for Example 2. Figure 8.5 shows the structural grid layout.
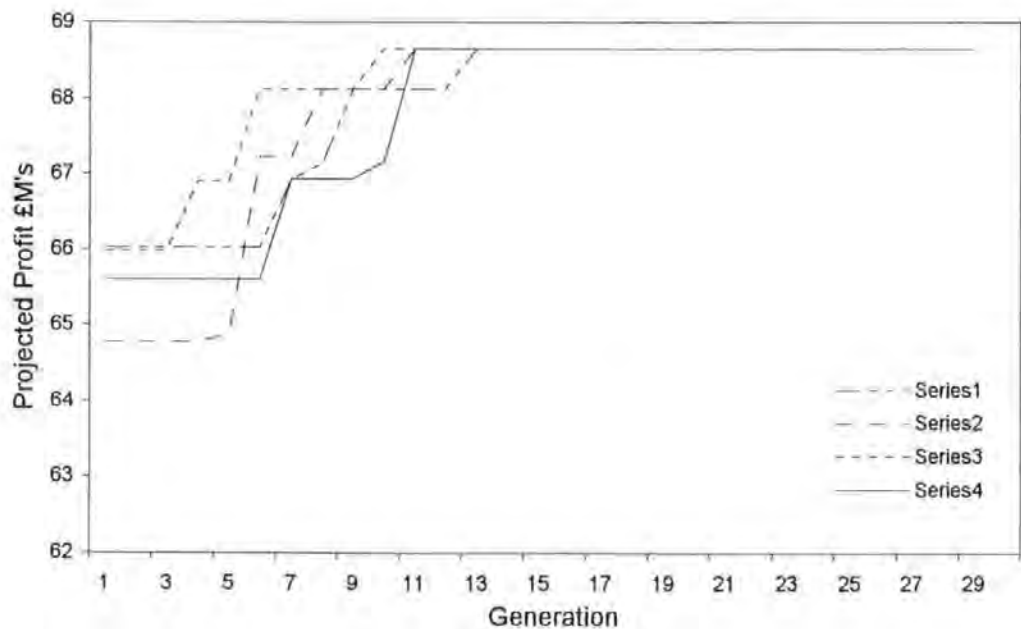
Figure 8.4: Convergence plot, Example 2.
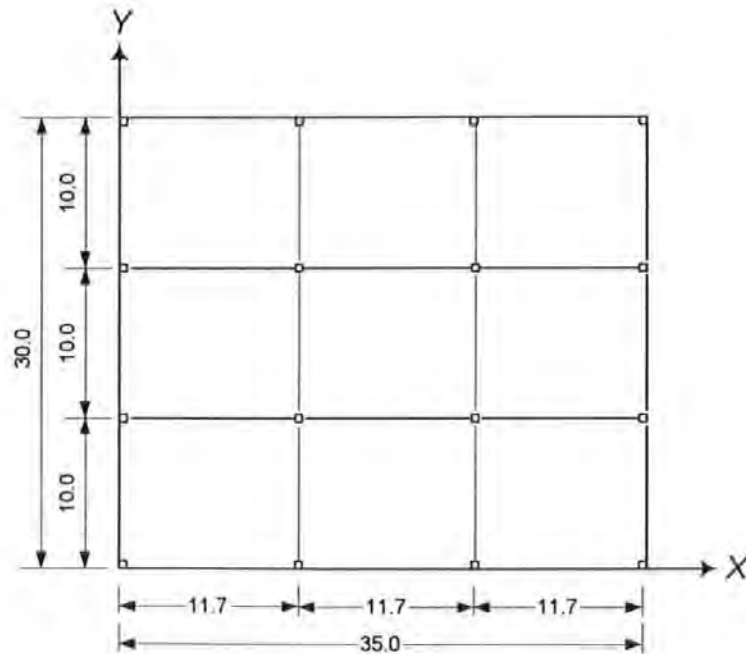
Figure 8.5: Structural grid layout, ground floor, from Example 2, solution by GA.

8.5 Example 3 – Fixed Structural System

In the third example, there is a preference for using a particular structural system, so the task is reduced further to one of finding the optimal form and layout of the structure. To offer variation, a building using in situ RC for the structural floor as well as frame is specified. Again this change is implemented using the visual design hierarchy, as shown in Figure 8.6. There are 131072 chromosomes. Land cost is £2500/m^2.
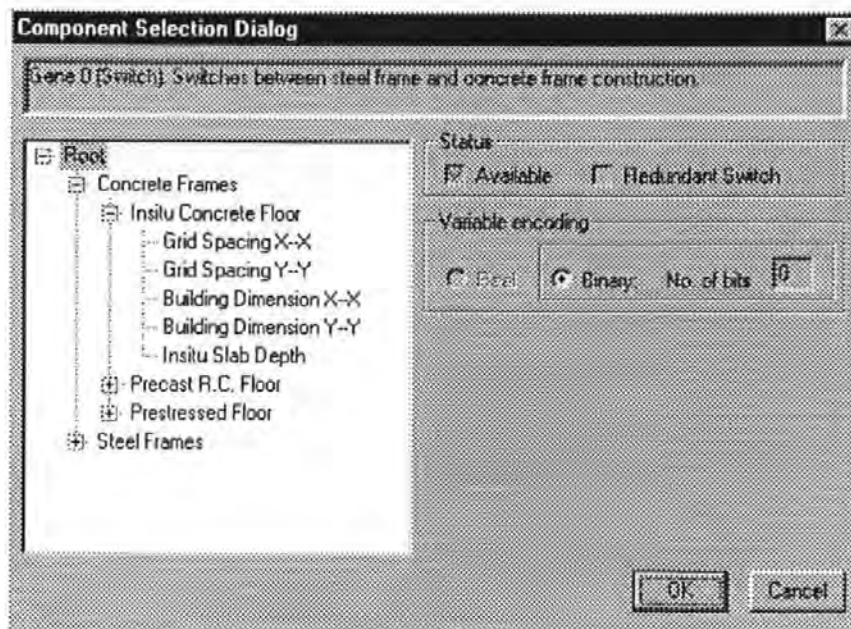
Figure 8.6: Component Selection dialog box updated to reflect the decision to only investigate concepts that use a RC frame with an in situ RC floor.

140

The optimal design solution was as follows: -

Frame type:	**RC (fixed by user).**
Floor system:	**in situ slab (fixed by user).**
Dimensions:	**55m x 60m footprint, 39m tall.**
Grid:	**8 bays at 6.8m (equally spaced) by 5 bays at 12.0m.**
Floors:	**13.**

Figure 8.7 shows convergence plots for Example 3. Figure 8.8 shows the structural grid.
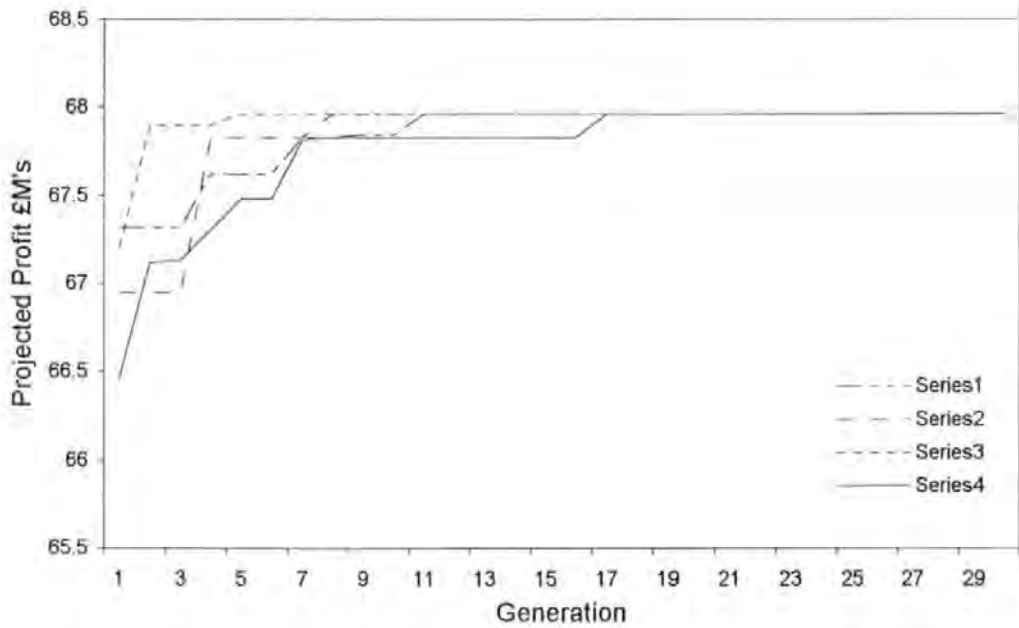
Figure 8.7: Convergence plot, Example 3 (In situ concepts only).
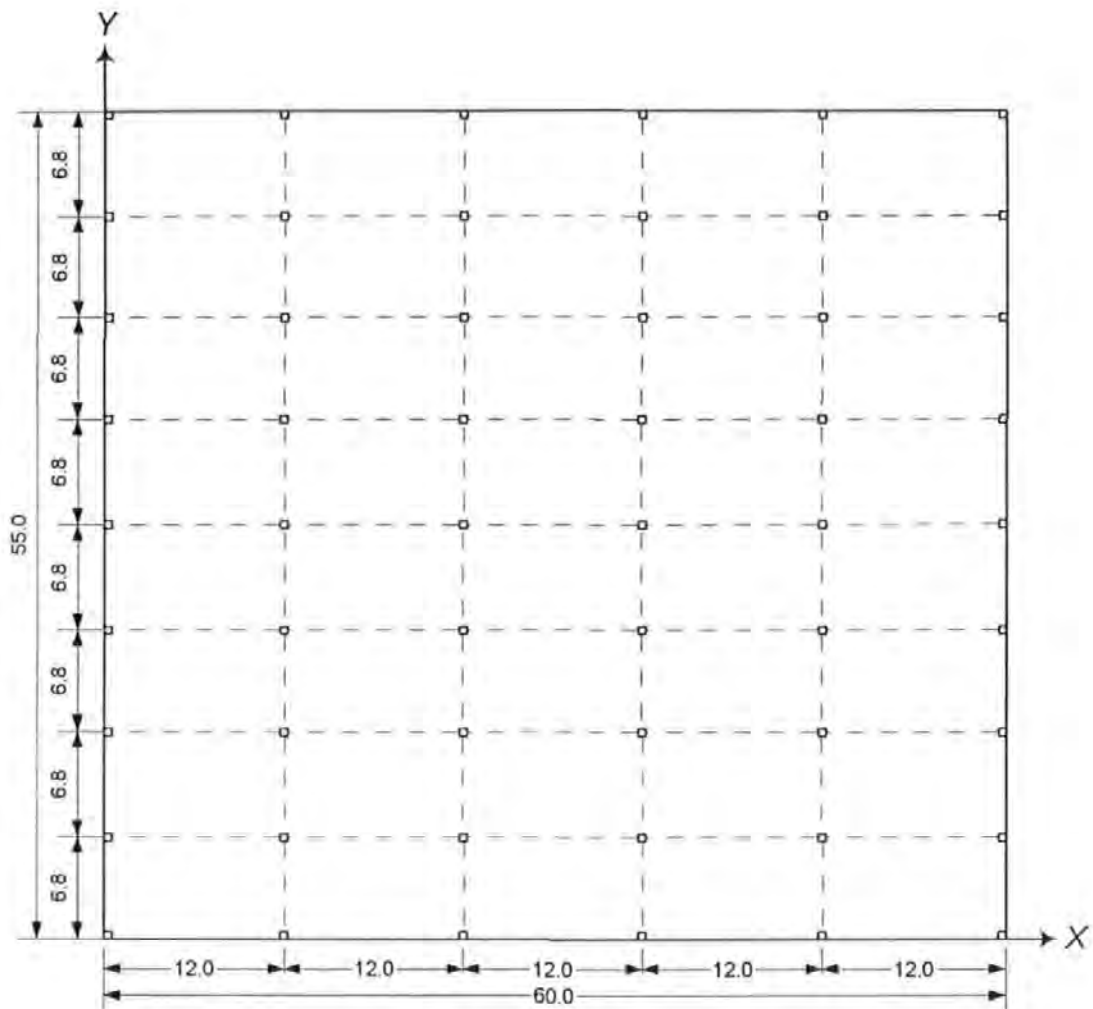
Figure 8.8: Structural grid layout, ground floor, from Example 3, solution by GA.

8.6 Example 4 – Fixed Footprint

The fourth example demonstrates bottom-up design. In practice, architects often limit the amount of space provided per floor of a building to about 4000-5000m^2. In this example, the building footprint is hypothetically fixed due to a restricted site. This situation is more typical of an inner-city development that the first example. The site may be located between other buildings, aligned in a row, such that it may conditional in obtaining planning permission that the structure should provide continuity in form. Here, the footprint is taken to be 40m by 20m. It is assumed that 8500m^2 of office space is ideally sought, which is considerably less than the figure used in previous examples. It should be apparent that feasible concepts should contain somewhere in the range of 11 to 15 floors. This is typical of a medium-rise building.

Land cost is unchanged from the previous example but becomes irrelevant in this situation, since its effect up the footprint has been overridden.

The optimal design solutions was as follows:-

Frame type: **steel.**

Floor system: **precast concrete panels.**

Dimensions: **40m x 20m footprint (fixed), 29m tall.**

Grid: **4 bays at 10.0m (equally spaced) by 2 bays at 10.0m.**

Floors: **10.**

8.7 Example 5 – Fixed Footprint, Theoretical Case

Another common situation is that where BRs limit building height. For example, new buildings in the Borough of Westminster, London are not permitted to exceed the height of St Paul's Cathedral and obscure the view of this important, historical landmark. There have been reported cases where feasibility studies have highlighted that the restriction upon the space that can be made available affect the viability of a project, and calls for a special solution to make it work. For example, the Bishopsgate development, described by Whitelaw [114], and mentioned in section 4.10.

Consider the siutation where certain combination of parameters, the details of which are not important, generated a conceptual design solution with an 11m by 4m grid and using prestressed concrete slabs. Note prestressed concrete has an economic range of about 6-14m, but shorter or longer spans can be constructed, less efficiently. In practice, a grid generated at a 3m or 4m interval would hinder functionality. The structural engineer would typically modify this solution to use secondary beam system, removing alternate grid lines to produce 6m bays – a much more practical dimension. In the DPRO system it is possible to avoid this solution. Table 6.1 and figure 7.6c (the PS page of the *Cost Options* dialog box) show that PS panels are available to suit spans from 3-15m. For GA encoding, prestressed concrete floor slabs were given a default range of 3.5m to 14m, at 1.5m increment, conveniently creating 8 individuals and ideal for 3-bit binary encoding. To change this range to avoid the generation of impractical 3.5m spans, the range can be changed to generate larger spans only – i.e. the set of spans: "5.0m, 6.5m, 8.0m, 9.5m, 11.0m, 12.5m, 14.0m". The seven permissible values are recognised as not being an exact base-2 multiple (2, 4, 8, 16, etc…) so DPRO automatically changes the encoding scheme to real-encoded variables. [Naturally, the original parameter range could have been

selected to be real encoded, in which case no change would be required for a reduced (non base-2 multiple) number of permissible values.]

8.8 Example 6 - Fixed Footprint and Grid

Fixing the footprint and the grid constrains the design greatly. The GA becomes superfluous to the task. There are only 16 different chromomsomes to evaluate. Performing constrained ES reveals that the most profitable design, having an absolute fitness value 5315.45 (£1000's), is as follows: -

Frame type: **steel.**
Floor system: **precast concrete panels.**
Dimensions: **60m x 30m footprint (fixed by user), 9m tall.**
Grid: **12 bays at 5m (equally spaced) by 4 bays at 7.5m.**
Floors: **3.**

8.9 Example 7 – Parametric Studies: Variation in Land Cost

As mentioned earlier, there are 262144 different chromosomes that produce design solutions (though not all are necessarily feasible or unique). In combination with variations in unit costs and structural design parameters there is massive scope for parametric study, that cannot be covered here. Instead parametric study is demonstrated for one variable aspect of design only – variation in land cost.

Some clients, like the Crown, government, and local authorities, may develop on their own property. In this case, land cost may become a secondary consideration. The purchase of land can be removed from consideration in the design system by setting the land cost value to zero, in the *Cost Options* dialog box.

Land purchase is necessary and the costs can vary greatly. At one extreme, urban areas where land is in short supply create enormous costs. Manhattan Island, New York and Hong Kong are prime examples. At the other extreme, land may be plentiful and can made available at generous rates to encourage business growth. An abundance of space and the relaxation of certain BRs is more likely away from city centres, and might be encountered in a green-field or brown-field business park.

We consider the effect of presenting the same building specification, with no shape or structural constraints, with variation in the cost of land purchase. Three cases are examined, Case I, Case II and Case III, in which land cost is set at £1000/m^2, £2500/m^2 and £5000/m^2, respectively. Optimal cost design solutions are shown in table 8.4: -

	Case I	Case II	Case III
Frame type	Steel	Steel	Steel
Floor type	PC panels	PC panels	Composite Deck
Length, m	80m	45m	30m
Width, m	70m	40m	35m
Grid x, m	8 bays at 10m	4 bays at 11.25m	4 bays at 7.5m
Grid y, m	7 bays at 10m	4 bays at 10m	4 bays at 8.75m
Main Bm	686x254x170UB	838x292x194UB	533x210x92UB
Sec. Bm.	610x229x101UB	610x229x113UB	356x171x45UB
Bldg Ht.	23m	66m	111m
No. of floors	8	23	39
Actual Profit	79132	69725	69725
Optimum Profit	80336	72859.36	"

Table 8.4: Details of concepts providing 40000m^2 of lettable space, generated for land costs of £1000/m^2, £2500/m^2 and £5000/m^2.

Figures 8.9(a) – (c) show the convergence plots produced for Case I, Case II and Case III. Figures 8.10(a) –(c) show the visualisation of the best design produced for each case.
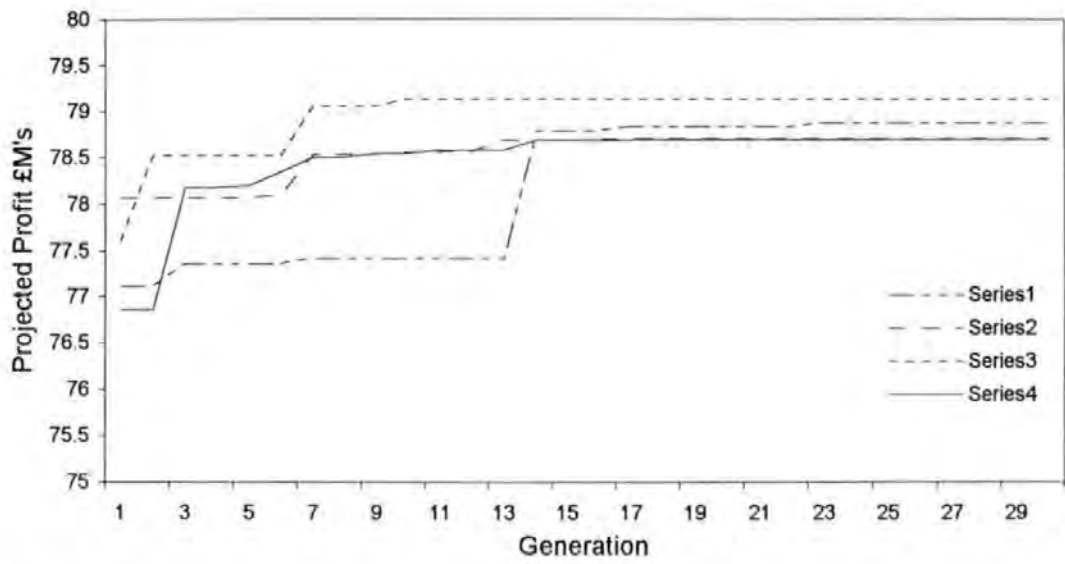
Figure 8.9a
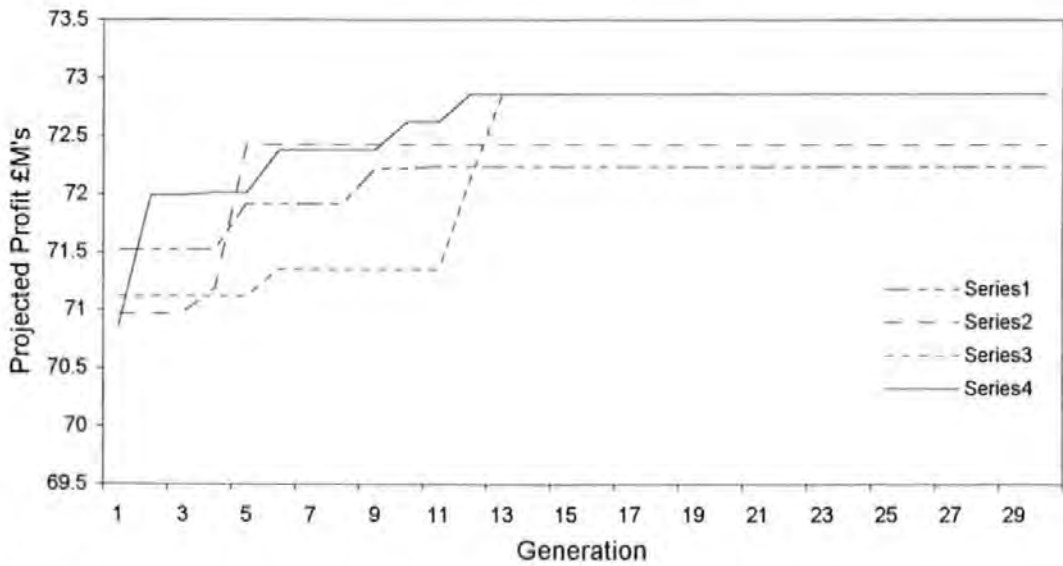
Figure 8.9b
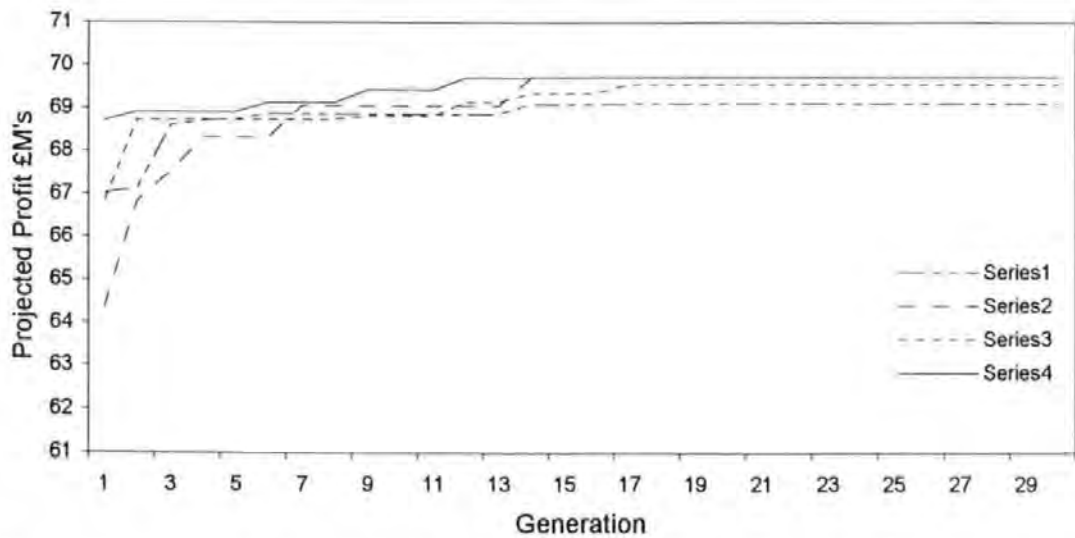
Figure 8.9c　　　　Figure 8-9(a) – (c): Convergence Plots, Example 7.
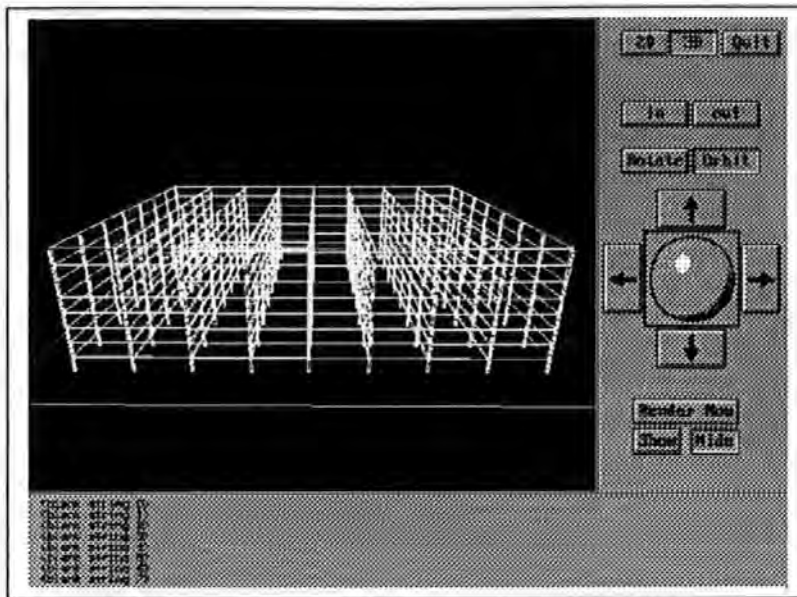
Figure 8.10a
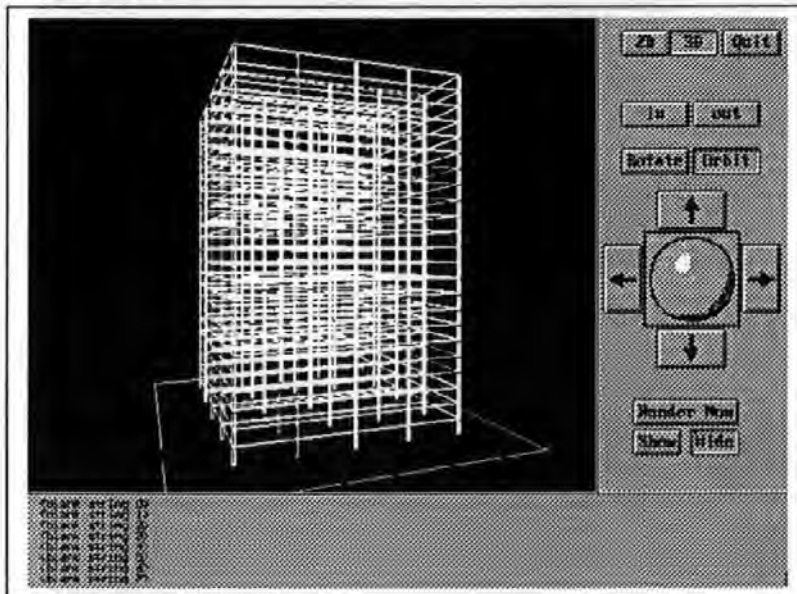
Figure 8.10b
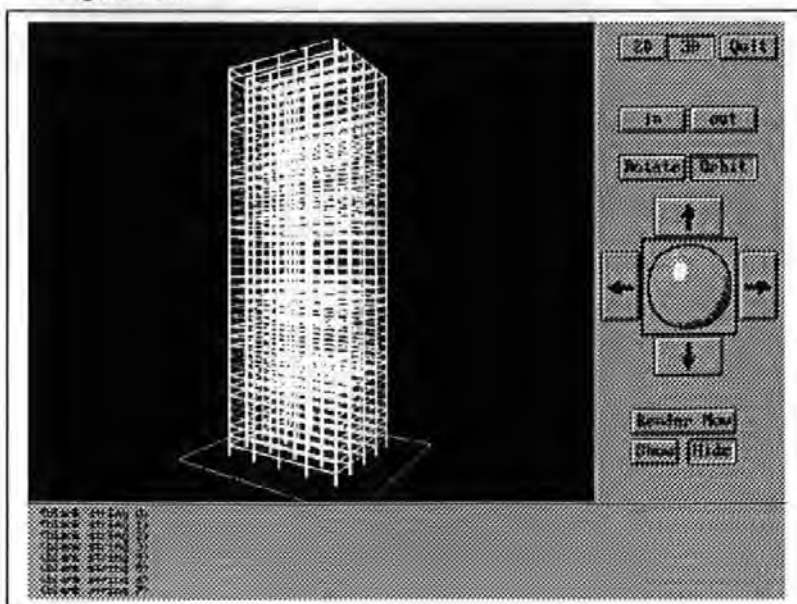
Figure 8.10c

Figure 8.10(a)-(c): Visualisation of best concept for variations in land cost.

147

8.10 Discussion

The GA has highlighted some interesting matters concerning fitness functions, fitness evaluation, and parametric study. A description of current research methods and findings were published in Mathews et al [134,135] and Rafiq et al [136]. Grierson et al [99] was inspired by some of these ideas and has pursued studies involving Pareto optimisation. Notably, however, the GA employed by Grierson et al [99] is simple in other regards. It applies straight heuristic style information to award zero fitness to buildings over prescribed heights that attempt to use certain structural system, for which an absolute limit has been applied in the compare. In DPRO poor designs are not prevented outright but are rejected very quickly in the normal course of a genetic experiment, instead. Sisk et al [101] also acknowledge the author's research.

8.10.1 Applying Stochastic Search Techniques Efficiently

The GA uses an efficient encoding to match the problem to a design specificiation. Earlier in this chapter a default design scenario was presented. One part of the source data was GA control parameters. The use of fixed parameters for *PopSize*, *Prob_Cross* and *Prob_Mut* demonstrates the robustness of the GA for the variety of tests, performed. Genetic experiments were also performed that used longer runs and and large population; the system performs confortably with *PopSize* of 200 / 300 chromosomes.

8.10.2 Parametric Study

It has been shown that the DPRO system accommodates variability and can in turn produce variation for example, for similar buildings in different locations. Parametric study offer the opportunities for knowledge creation and to investigate specific structural behaviour.

8.10.3 Fitness Evalutation and Computational Effort

- Different alternative systems can have different numbers of associated variables, meaning that the number of permutation for each type is not equal. Another way of putting this is that some systems offer greater variation, e.g. RC, through grade of concrete, reinforcement content and distribution.

- Whilst efforts were made to reduce the number of infeasible designs, some existed. Consequently, there is not necessarily an even balance between the number of viable solutions using an alternative system, even if the potential number of solutions of each type is equal. It should be noted that the constraints and structural design parmeters determine the number of infeasible designs; consider for example the combined effect

148

of a design brief that requires a structural floor to withstand a heavy imposed load, as for example in a library, and functionality requires a clear spans.

- Ignoring infeasible designs, computational effort required for design evaluation varies mirror the manual efforts as would be required to create different building forms. For a low-rise building, say one or two storeys high, there may be only be one column section. However, a medium rise building of say, 14 storeys, may contain seven column stages. The computation effort also varies according to the type of structural systems, based on complexity and the amount and type of processing involved (straight calculation, referencing a look-up table, iteration). In tests, it was possible to evaluate between 5 and 30 designs.

8.10.4 Miscellaneous

Using OOP technology it has been possible to create – and therein perform a genetic experiment containing a chromosome with mixed encoding schemes. Since each gene or allele looks after itself, using OOP principles, safe operation is assured. Notably in a studying Pareto-optimal building concept design using the GA, Grierson et al [99] reported that at the early part of a genetic experiment, standard crossover at bit level is beneficial, and that later, it has been found better to swap or replace existing gene values directly, i.e. as a whole entity, rather than to perform inter-gene crossover. The relevance of real-encoded crossover and mutation is highlighted.

In the examples presented in this chapter, $Prob_{Cross}$ = 0.80. Notably, Grierson et al [74,99] advocate the use of near-100% crossover rate in research studies. This seems to relate particularly to the desire to effect maximum diversity in order to create diverse Pareto-optimal designs.

9 Conclusions

9.1 Summary and Conclusions

This section reviews the current approach towards supporting designers at the conceptual stage. Conclusions arising specifically from the the research are offered.

9.1.1 Conceptual Design Aspects

The research had a clear practical application. This study sought the opportunity to provide DS at the conceptual stage of building design, where decisions have a significant effect upon the successful outcome of the project, using the GA. Specifically, opportunities to efficiently generate, appraise and convey to designers the relative merits of different structural design concepts, comprised of different materials and subsystems and taking various forms, for practical advantage were studied. The purpose of this project has been to assess the potential of the GA in assisting those members of the design team, involved at the outset of design, in producing efficient design concepts in an integrated manner. To this end, the project has drawn upon findings of relevant KBES and GA research and has combined this knowledge with new techniques relating to the CBD task.

In particular, the intention has been to encourage collaborative design by the architect and structural engineer using the electronic computer as a medium. During the course of research, the GA was first applied to floor planning activity, and later, to the generation of structural design concepts. The application of appropriate representation schemes, fitness functions and refinements enabled the GA to be applied with much success. However, it is important to note that the value of the techniques described herein is derived as much from the freedom afforded to the designer, as to the efficiency of the search. In this regard, there is a trade-off between specific, hard-coded, 'black box' processes and versatile, general purpose, semi-transparent processes. Design is a process that necessitates human involvement, and requires due consideration. Gero [137] said: -

"In conceptual designing the designer works with his experiences, his knowledge and his conception of what is in front of him – the situation – in order to determine what may be described more formally as, the variables that go to contribute to the function, behaviour and structure of the resulting design. The particular behaviour and structure variables are not chosen *a priori* but are produced in response to the various situations as they are encountered by the designer. What the designer has done previously, both prior to this design and during the current process of designing affects how the designer views the situation and what memories he constructs and brings to bear on the current situation."

Detail design has a fixed definition but a flexible solution. Conceptual design requires flexibility in the problem definition and in solution. There does not exist a single and universally applicable method to solve a general class of problems; rather a rational complementary approach based on a collection of ideas and techniques. The GA is better, in terms of its performance, and more suitable, in terms of its flexibility and speed, at some types of problems than others. As Bedford [90] noted, few problems are "uncomputable"; however, the best solution, and indeed the best approach to obtaining that solution, is seldom obvious.

Consideration has been given to offering effective support, rather than automation. Optimisation is one specific activity in a broader design process; others include the communication of requirements, solutions, analyses, and suitable information representations that support these processes. Numerous aspects of design processes may be supported using a collection of different techniques, either independently or in complementary manner. It is possible to generate various acceptable design solutions and vast quantities of associated information.

The application of EAS techniques and the computational expense should be justified. The DPRO system is adaptable to different design situations. The system provides versatility in the examination of alternative design concepts. By seeking efficient design representations and combining HCI, intelligent search manipulation has been demonstrated. Flexibility and control are required to support manual decision-making, using high-level knowledge, as opposed to automated decision-making. HCI maintains the fluidity of the creative process.

9.1.2 Review of the Current Approach

A building design model was created and verified. It was used to produce geometry, topology and structural component variations. Variables were encoded in a SGA chromosome. The SGA was used to study many alternative design permutations, efficiently. A study was made to determine which elements would best represent the diverse range of options that are available in practice. The model demonstrates broad application and encapsulated major structural alternatives. Buildings were rectilinear. Column grids were orthogonal. All floors were considered as having the same function, with similar spatial requirements and similar imposed load. Fitness was based on the single criteria of cost, by representing other criteria as constraints.

The model contained independent and dependent design parameters, where the latter are related to independent aspects. Component member sizing adopted standard, lower bound design methods and proprietary methods in order to select suitable section with adequate capacity. Ultimate limit state, serviceability limit state and buildability criteria were incorporated into the design development process. Design development and cost calculations were performed in the required sequence following the standard load path from a structural floor system, through beams, columns to foundations and from there into the ground.

Alternative structural systems were supported using the SGA. GUI tools and HCI techniques were created allowing modification to be made to the domain as necessary. Some parameters were constant for a particular design specification, and were used in the fitness function, were external to the chromosome. Imposed floor load, component sizes, material costs, footprint sizes, structure height could be configured. Using GUI controls, aspects such as the total amount of floor space required, the imposed load and the design life required could be set interactively, demonstrating applicability in different circumstances. Cost data was maintained independently for various components. Unit cost values combined material, plant and labour, and could be adjusted.

The scale of design system necessitated OOP. As a result, DPRO is now highly extensible.

9.1.3 Specific Findings of Research

The thesis offers guidance for further work through collated related information, discussion and the new techniques presented. Chapter 4 mentioned some important considerations for design modelling, including the use of generic components and conservative estimates to help simplify design details. The need to ensure compatibility is addressed. Chapter 6 introduced the structured GA, which was adopted for its capacity to support alternation and design compatibility. Specific techniques that enabled the GA to be implemented in a flexible manner using OOP, GUI and HCI concepts were described.

Some interesting side effects were discovered during this study. Notably, singular design evaluation and constrained ES can be useful. The creation of a design model can lead to realizations that are either unknown or taken for granted. The very process of bringing together knowledge as a collective resource demonstrates a capability for software to assimilate more knowledge that is humanly possible. Complex inter-relationship can be revealed through parametric study. The notion that deep knowledge may be revealed is

associated with the idea of using the microcomputer to think about, learn and understand better the relationships that exist within and between different disciplines that may not be obvious[1]. For example energy analysis, glazing and perceived cost relationships can be explored more easily. (Pareto optimisation, mentioned below, has enabled this). The ability of the GA to handle a design specification and to explore a domain efficiently rather than to follow a predetermined path has great potential and holds the possibility of the discovery of new knowledge. The analogy exists between the genetic building blocks and physical building blocks.

9.2 Future Directions

This section describes improvement and further development to functionality of the DPRO system, discusses opportunities to extend the domain to include variation in othe aspects of conceptual design, and mentions possible future research directions and complementary advances being made in related fields.

9.2.1 General Improvements and Further Development

The DPRO tool developed by this work is novel and demonstrational. It necessarily has limitations and applies some assumptions that would reduce its practicability without further development. This should not be seen to reflect the applicability of the methods as described. It should be clear that the system is straightforward enough to enable an architect to use it to appraise design concepts. The robustness of the GA control parameters was demonstrated through examples in Chapter 8. Notably many aspects can be overcome by adding detail in the design model and greater flexibility in its manipulation. The GA is not only limited to intelligent guessing like KBES tools, but can assimilate complex relationships. The following are suggested development that could help in testing, using and gathering results from the system: -

- The DPRO system would need to be totally seamless and stable for use by a third-party. Range checking, exception handling, clear presentation of design data including rounding of values, seamless integration of modules and greater help facilites relate to this point.

- Integration of visualisation and GA modules could show continuous design evolution in real-time. Fast microprocessors (1GHz and above) could make this viable, soon.

- The facility to create a new design domain or to expand an existing one to incorporate a new structural system (i.e. visually adding nodes to a hierarchy, or graphically

---

[1] Also known as inter-disciplinary and intra-disciplinary knowledge.

manipulating a design and specifying ranges), is a possibility. A more practical approach would be to enable a designer to evaluate a set of potential designs, make manual modifications to suit his / her individual requirements and apply the GA with the constraints specified by the designer to produce a new optimum design, thus developing designs in a cooperative maner. Ideally, the fitness function could be manipulated, and not only the parameters and constraints that are applied.

- In relation to the last point, it could be useful to be able to suspend execution and re-commence it later, and allow some other form of interim activity, like a structural analysis of the current, best-solution. This could be controlled interactively. This way the designer can develop ideas as they come to mind, perhaps by reconfiguring the domain, because execution has ended. There are additional reasons why it may be useful to temporarily exit an application, or close down the computer.

- The ability to record the duration of the GA and ES processes would be useful.

- The ability to set up and run genetic experiments as a batch process. This could allow the effectiveness of runs that use different genetic control parameters to be compared, for example.

- Whilst the exhaustive search is already limited to the domain shown in the *Component Hierarchy* dialog box, diagnostic tools to test particular solution could be helpful.

- Hill-climbing techniques could be added.

9.2.2 Extending the Design System Domain

There is enormous potential to extend the system. These include: -

- Exploring applicability beyond open-plan offices. Functional optimisation and the integration of building services could be attempted by considering layout and space planning. Construction cost involves the superstructure, architectural features, finishes and provision of services. Optimal cable routing is one potential application, particularly in relation to vertical openings in the structures. Functional optimisation has previously been used in architectural using diagrams called *graphs* to determine which design aspects should be adjacent. Medium-rise office / commercial structures are increasingly open-plan to attract a range of tenants. Certain other types of structure like hospitals, cinemas and leisure centres have specific functions, loads and inter-spatial requirements.

- The present system takes a pragmatic approach and assumes that a rectangular structure is the most cost-effective. Structural shapes can be extended to incorporate non-orthogonal geometries and irregular floor plans. This could be more supported using a

suitable GUI interface, perhaps using a commercial CAd system. Although buildings are orthogonal and rectilinear, some concepts permit closer agreement with design rationale. For example, whilst a hexagonal structure would be likely to produce some complication in the construction detailing, a L-shape structure is essentially two coincident (adjacent) rectangular parts. Potential situation that could warrant such a design could be those that require an unusual site plan or demonstrate energy-efficiency through greater daylight provision.

- Notably, all concepts developed in this study follow a predefined design "formula". Incorporating non-standard feature such as an atrium, foyer, basement or mezzanine floor could also be explored. Functional and structural consideration could produce a structure where additional columns are used at lower storeys in a multi-storey building.

- Costs could be expanded to take into account not only of intial capital costs, but life cycle costs, that recur, including maintenance and running costs.

- Related to the building design process, emerging fields of application include financial planning, construction scheduling, facilities management and resource allocation. Specifically, these applications aim to assist contractors, accountants, clients and other member of the design team to develop appropriate strategies. Construction planning and construction management could be combined with the estimation capabilities of the system, to provide a further level of integration.

- The range of components in the system has been restricted to common structural types and could be extended to offer greater variety. Tubular steel and inclined members are some possibilities. These options provide greater choice and creativity design. The computer is a powerful tool for extending designer awareness to new areas. For example, timber can be viable for buildings up to five stories high, has a long life span and is highly aesthetic, yet is under-used in the UK for low-rise buildings, mainly because of a lack of specialist design / construction skills.

- This thesis describes established practice as rationale. The rationale presented herein in derived from structural considerations. Integration at a wider level is likely to expand the rationale. For example, the implications of using suspended baskets to carry electrical services, and panelised cladding to replace a traditional stick system, maximising the use of prefabriaction, for economic design is an area that remains to be explored. Often this kind of advice is imparted through case studies, such as the Bishopsgate development, see Whitelaw [114]. KBESs are also a good source of information for carrying forward the research to address aspect such as formwork detail (see Koo et al [138]), transportation systems (see Cagdas et al [139]), foundation selection (see Kim et al [140]) and building envelope design.

9.2.3 Future Research Directions

The author has been greatly encouraged by others that have shown interest in the work described herein, and have taken the research forward and in new directions. This study formulated a multicriteria optimisation problem as a single criteria optimisation. Park et al [141] presented multicriteria optimisation using the GA for Pareto – optimisation, capable of producing set of designs for a designer to examine, and has developed models that highlight other interesting relationship. For example, he mentions how the relationship between cladding and window light affects the quality and perceived value of a structure, and explores the relationship between capital cost, maintenance cost and running cost. In conflicting multi-criteria problems, a compromise solution is usually necessary. One dimension of the compromise may involve the element of risk, e.g. in lost time or structural safety. Exploring a domain may reveal that a slightly sub-optimal solution carries fewer risks, perhaps by enabling design variation at a late stage, or placing less reliance upon most uncontrollable aspects (labour, weather). This approach is worthy of much consideration. For example, if the fitness landscape is flat, many equally-economic solution can be investigated.

Sisk et al [101] apply considerable experience in applying knowledge engineering techniques in the field of bridge design to CBD using GAs. In more theoretical research, Maher et al [55,56] has used CBR for pattern recognition, for creative design.

Rafiq et al [100,126] has continued to investigate how ANNs and GAs may be integrated, with to the GA generating concepts and the ANN used to develop design solutions, and helping to manage copious design knowledge effectively. The ANN approach offers great potential to very quickly identify high fitness solution, whilst reducing the computational burden on the GA and the length of SGA chromosomes. Research at the University of Plymouth PEDC by Parmee [67] has shown that the SGA can be surpassed in complex problem by an even more powerful technique involving a hybrid EAS called the GA-ANT algorithm.

The interest in the GA applied to the CBD domain, and its reported success from independent research groups attests its potential. The fact that the benefits of these ongoing studies can be combined shows great deal of promise. Where relevant, techniques to enable partially constrained design spaces to be searched, as described herein, can be useful. From earlier chapters hopefully it is clear that the power of the GA can be reinforced using other complementary techniques, whilst in general the benefits of design

software can be greater when it becomes possible to support the entire building design process in software, and provide seamless integration.

Also intelligent CAD and visualization tools could become more significant in widening interactive design development. Techniques such as the use of "4-D" CAD, where realistic virtual design model are overlayed with pertinent design information, like stress and cost contours, could represent a revolution in the way that conceptual design is approached. There is a significant role of graphical and visualization tools for interactively exploring design models. There may also be a role for using natural-language parsing and rule-reduction techniques to describes and encode complex relationships (see Hudson et al.[94], Balachandran [118]). Whilst the present study uses cost-based fitness functions, factors that are harder to quantify – and which may not be used in automatic appraisal (selection pressure) – including aspects such as aesthetic appeal and provision of natural lighting could be examined. Semi-automatic GAs where the user steers the selection could be possible.

9.2.4 Complementary Advances

Research aside, the publication and dissemination of design standards in electronic format by British Standards Organization, British Steel Corporation, the SCI, the RCC, and proprietary manufacturers seems to reflect a policy of promoting awareness and openness. Efforts are being made via the Internet and other media to promote collaborative research and to disseminate program source code and results. As electronic information becomes more common, the ability for design tools to treat a new type of structural system as an add-in, that can be easily updated, becomes more realistic.

In 1995, Taffs [18] predicted that component suppliers may soon be obliged to make cost information avaiable electronically. He described the need for open-standards and neutral data files to facilitate information exchange. He said that an external market-oriented influence might be required to instigate change in the commercial sector. Ritchie [142] said that in the light of technological advance: -

"there is no reason to suppose that we cannot make economy and efficiency subservient, without denying their crucial importance in the design process and eventual artefact."

Using the GA and co-operative techniques, the dream and the reality move ever closer.

References

CHAPTER TWO

[1] Dasgupta D. & McGregor D., 1991
"A Structured Genetic Algorithm", Research Report IKBS-2-91, University of Strathclyde, UK.

[2] Rafiq M.Y. & Mathews J.D., 1996
"An Integrated Approach to Structural Design of Buildings using Genetic Algorithms" in
Proc. 2nd World Congress of Integrated Design and Process Technology, IDPT-Vol.3,
Esat I.I., Veniali, F., Rasty J., Gransburg, D.D. & Ertas, A. (eds.), Dec., pp.84-90.

[3] Arora J.S., 1997
"Guide to Structural Optimization" in Advances in Structural Optimization, Frangapol D.P. & Cheng F.Y.
(eds.), ASCE.

[4] Bedard C. & Gowri, 1990
"Automating the Building Design Process with KBES", Journal of Computing in Civil Engineering, Vol.4,
No.2, April, pp.69-83.

[5] Howie W., 1994
"An Art Apart", New Civil Engineer, 17/11/1994, pp.23-4.

[6] Blockley D.I., 1980
"The Nature of Structural Design and Safety", Ellis Horwood Ltd., Chicester, UK.

[7] Lawson B.R., 1990
"How Designers Think – The Design Process Demystified", Butterworth Architecture.

[8] Billington, D.P., 1983
"The Tower and the Bridge: The New Art of Structural Design", Basic Book Inc Pubs.

[9] The Business Round Table Ltd., 1994
"Controlling the Upwards Spiral", in Building and Civil Engineering Research Focus, No. 21, Apr. 1995.
The Business Round Table Ltd., 18 Devonshire St., London, W1N 2AU.

[10] Moore D, 1995
"Buildability: A Problem of Managing the Transfer of Construction Process Knowledge",
in Campus Construction, CIOB, Vol.3, No.2, July.

[11] Russell H., 1995
"The Helicon Accord", New Civil Engineer, 23/02/1995, pp.18-20.

[12] Thompson K., 1996
"Integrated Software for Design and Management Projects" in Information Technology Strategy for Civil
and Structural Engineering Design, MacLeod I. (ed.) University of Strathclyde.

[13] CIOB, 1988
"Code of Estimating Practice, Supplement No.2, Design and Build", The Chartered Institute of Building.

[14] Turk Z., 1996
"Construction on the Internet: Hype and Reality" in Information Technology Strategy for Civil and Structural
Engineering Design, MacLeod I. (ed.), University of Strathclyde.

[15] RIBA, 1980
"Handbook of Architectural Practice and Management", RIBA Publications Ltd., 4th edition.

[16] Maver T.W., 1970
"Appraisal in the Building Design Process", in Emerging Methods in Environmental Design and Planning,
Moore G.T. (ed.), MIT Press, Cambridge, Massachusetts, pp.195-202.

[17] IStructE, 1985
"Manual for the Design of Reinforced Concrete Building Structures", Institution of Structural Engineers, London.

[18] Taffs D.H., 1995
"Computing for Consulting Engineers – An International Dimension", Sixth International Conference on Civil and Structural Engineering Computing, Cambridge, CIVIL-COMP Press, Edinburgh, UK, pp.

[19] Deiman E.P. & Plat H.T., 1993
"Cost Information in Succeeding Stages of the Design Process", in Advanced Technologies, Behesti M.R. & Zreik K. (eds.), Elsevier Science, pp.327-41.

[20] Evbuomwan N.F.O. & Anumba C.J., 1996
"Towards a Concurrent Engineering Model for Design-And-Build Projects"
in Struct. Eng., Vol.74, No.5, (5 March), pp.73-8.

[21] Taffs D.H., 1994
"The Structural Engineer and IT", in Struct. Eng., Vol.72, No.9, pp.146-8

[22] Grierson D.E., 1998
"Information Technology in Civil and Structural Engineering Design in the Twentieth Century", J. Comp. and Struct., No.67, Elsevier Science, pp.291-8.

[23] Koumousis V.K., Georgiou P.G. & Hernández S., 1993
"ADS Expert" in Optimization of Structural Systems and Applications (Computer Aided Optimum Design of Structures III), Hernández S. & Brebbia C.A. (eds.) Comp Mechs Pubs Southampton and Elsevier Applied Science, pp. 271-84.

[24] Bedard C. & Ravi M., 1989
"Preliminary Planning and Design of Multistory Buildings", in Computer Aided Optimum Design of Structures: Applications, Proc. 1st Int. Conf., Jun 89, Computational Mechanics, Springer-Verlag, pp.201-10.

[25] Adeli H. & Balasubramanyam K.V., 1988
"A Coupled Expert System for the Optimum Design of Bridges", in Proc.IEEE, New York, N.Y., 114-9.

[26] Jackson P., 1986
"Introduction to Expert Systems", Addison-Wesley Publishing.

[27] Bennett J., Creary L, Englemore R. & Melosh R., 1978
"SACON: A Knowledge-Based Consultant for Structural Analysis", Technical Report STAN-CS-78-699, Stanford University, September.

[28] AISC, 1978
"Specification for the Design Fabrication and Erection of Structural Steel for Buildings (SPECON)", American Institute of Steel Construction, Chicago.

[29] Maher M.L. & Fenves S.J., 1985
"HI-RISE: An Expert System for Preliminary Structural Design of High-Rise Buildings", in Proceedings of the IFIP Conference on Knowledge Engineering in Computer-Aided Design, Budapest, pp.125-46.

[30] Kumar B. & Topping B.H.V., 1988
"An Intergrated Rule-Based System for Industrial Building Design" (INDEX) in Micro Computer Knowledge Based Expert Systems in Civil Engineering, Adeli H. (ed.), 1993, Symposium of Conference Papers, 10-11th May 1988, Vol. 8, No. 1, American Society of Civil Engineers, pp.53-72.

[31] Lane V.P. & Loucopoulas P., 1985
"A Knowledge Based System as a Mechanism for Optimization of Conceptual Design in Civil Engineering Projects", in Optimisation in Computer Aided Design, Gero J.S. (ed.), Elsevier, Netherlands, pp.1-11.

[32] Reddy R.R., Gupta A. & Singh R.P., 1993
"Expert Systems for Optimum Design of Concrete Structures", J. Comput. Civ. Eng., Vol.7, No.2, April, pp.146-61.

[33] Kumar B.& Topping B.H.V.,1988
"Issues in the Development of a Knowledge-Based System for the Detailed Design of Structures" in Artificial Intelligence in Engineering Design, Gero J. (ed.), Computational Mechanics Publications, Southampton, UK, pp.295-314.

[34] Kumar B., 1995
"Knowledge Processing for Structural Design: Topics in Engineering Vol.25", Computational Mechanics Publications, Southampton, UK.

[35] Ades N., 1991
"Computer Aided Techniques for Structural Engineering Design", Ph.D. Thesis, University of Strathclyde.

[36] Sriram D., 1986
"DESTINY: A Model for Integrated Structural Design", Journal of Artificial Intelligence in Engineering, Vol. 1, No.2, pp.109-116.

[37] Harty N., 1987
"An Aid to Preliminary Design" in Knowledge Based Expert Systems in Engineering Planning and Design, Sriram D. & Adey R. (eds.), Computational Mechanics Publications, Boston, USA.

[38] Sriram D., 1987
"ALL-RISE: A Case Study in Constraint-Based Design", Artif. Intell. Eng., Vol. 2, No.4, pp.96-203.

[39] Harty N.& Danaher M., 1994
"A Knowledge-Based Approach to Preliminary Design of Buildings", Proc. Instit. Civ. Engrs, No.104, pp.135-44.

[40] Karakatsanis A., 1985
"FLODER: A Floor Designer Expert System", Master's Thesis, Dept. of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA.

[41] Smith D.F., 1986
"LOCATOR: A Knowledge-Based Lateral System Locator for High Rise Buildings", Master's Thesis, Dept. of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA.

[42] Pack Y. & Adeli H., 1988
"SDL: An Environment for Building Integrated Structual Design Expert Systems, in Microcomputer Knowledge-Based Expert Systems in Civil Engineering, Adeli H. (ed.), pub. ASCE, pp.40-52.

[43] Rafiq M.Y. & MacLeod I.A., 1988
"Automatic Structural Component Definition from a Spatial Geometry Model", Eng. Struct., Vol. 10, January, pp.37-40.

[44] Sussock H., 1982
"GOAL: General Outline Appraisal of Layouts - User Manual", ABACUS, University of Strathclyde.

[45] Jain D., Krawinkler H., Law K.H. & Luth G.P., 1991
"A Formal Approach to Automating Conceptual Structural Design, Part 2: Application to Floor Framing Generation", J. Eng. Comput. No.7, pp.91-107.

[46] Carrara G., Kalay Y.E. & Novembri G., 1994
"Knowledge-Based Computational Support for Architectural Design", in Automation in Construction, No.3, pp.157-75.

[47] Najafi A.A., 1997
"An Integrated Generative Computer-Aided System for Architectural and Structural Design", Ph.D. Thesis, Department of Architectural and Building Science, University of Strathclyde, Glagow, UK

[48] Lawson B.R., 1972
"Problem Solving in Architectural Design", Ph.D. Thesis, University of Aston, Birmingham.

[49] Maher M.L. & Zhao F., 1993
"Dynamic Associations for Creative Engineering Design" in Modelling Creativity and Knowledge-Based Creative Design, Gero J.S. & Maher M.L. (eds.), Lawrence Erlbaum Publishers, New Jersey & London, pp.320-351.

[50] Sandgren E., 1995
"Multiple-Objective, Shape Optimal Design via Genetic Optimisation", in Structural Optimisation –
Computer Aided Optimum Design of Structures VI (OPTI '95), Hernandez S., Al-Sayed M. & Brebbia C.A.
(eds.), Computational Mechanics Publishers, pp.3-10.

[51] Adeli H. & Chen Y.S., 1989
"Structuring Knowledge and Data Bases in Expert Systems for Integrated Structural Design", Microcomput.
Civ. Eng. IV, pp.175-204.

[52] Woodbury R.F., 1993
"A Genetic Approach to Creative Design", in Modelling Creativity and Knowledge-Based Creative Design",
Gero J.S. & Maher M.L. (eds.), Lawrence Erlbaum Publishers, New Jersey & London, pp.320-351.

[53] Sabouni A.R & Al-Mourad O.M., 1997
"Quantitative Knowledge Based Approach for Preliminary Design of Tall Buildings", in Artif. Intell. in
Eng.II (Elsevier Science), Vol.11, No.2, pp 143-54.

[54] Smith I.F.C., 1996
"Interactive Design – Time To Bite the Bullet", in Information Processing in Civil and Structural
Engineering Design, Civil-Comp Press, Edinburgh, UK, pp.171-5.

[55] Maher M.L., Balachandran M.B. & Zhang D.M., 1995
"Case-Based Reasoning in Design", Lawrence Erlbaum Associates Inc.

[56] Maher M.L. & Pu P., 1995
"Issues and Applications of Case Based Reasoning to Design", Lawrence Erlbaum Associates Inc.

[57] Gero J., 1996
"Design Tools That Learn: A Possible CAD Future", in Information Processing in Civil and Structural
Engineering Design, Civil-Comp Press, Edinburgh, UK, pp.171-5.

[58] Rafiq M.Y., Bugmann G. & Easterbrook D.J., 1999
"Building Concept Generation Using Genetic Algorithms Integrated with Neural Networks", Borkowski, A.
(ed.), Proc. 6th EG-SEA-AI Workshop, Warsaw, Poland.

CHAPTER THREE

[59] Himmelblau D.M., 1972
"Applied Non-linear Programming", McGraw Hill.

[60] Rechenberg I., 1965
"Cybernetic Solution Path of an Experimental Problem", Royal Aircraft Establishment, Library Translation
No.1122, Farnborough.

[61] Rechenberg I., 1973
"Evolutionsstrategie. Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.
Problemata." Stuttgart, Frommann-Holzboog.

[62] Holland J., 1975
"Adaptation in Nature and Artificial Systems", University of Michigan Press, Ann Arbor, MI.

[63] Goldberg D.E., 1989
"Genetic Algorithms in Search, Optimisation and Machine Learning", Addison-Wesley, Reading.

[64] Davis L. (ed.), 1991
"The Handbook of Genetic Algorithms", Van-Nostrand Reinhold, New York.

[65] De Jong K., Arciszewski T. & Vyas H., 1999
"An Overview of Evolutionary Computation and its Application to Engineering Design" in
"Artificial Intelligence in Engineering", Borkowski, A. (Ed.), Proc. 6th EG-SEA-AI Workshop, Warsaw,
Poland.

[66] Goldberg D.E., 1991
"Genetic Algorithms as a Computational Theory of Conceptual Design" in Applications of Artificial Intelligence in Engineering VI, Rzevski G. & Adey R.A.(eds.), Elsevier Applied Science / Computational Mechanics Publications, Southampton & Boston.

[67] Parmee I.C., 1996
"The Development of a Dual-Agent Strategy for Efficient Search Across Whole System Engineering Design Hierarchies", Proc. 4th Int. Conf. on Parallel Problem Solving from Nature, Berlin, Germany.

[68] Dawkins R. 1986
"The Blind Watchmaker", Longman Scientific and Technical publications.

[69] Rajan S.D., 1995
"Sizing, Shape and Topology Design Optimization of Trusses Using Genetic Algorithm", J. Struct. Eng., October, pp.1480-7.

[70] Suh J.Y. & Van Guch D., 1987
"Incorporating Heuristic Information into Genetic Search",
Proc. 2nd Int Conf. in Genetic Algorithms and Their Applications, pp.100-7.

[71] Goldberg D.E., 1986
"A Tale of Two Problems: Broad and Efficient Optimization using Genetic Algorithms", Proc. 1986 Summer Computer Simulation Conference, pp. 44-8.

[72] Jenkins, W.M, 1994
"A Space Condensation Heuristic for Combinatorial Optimization", in Advances in Structural Optimization, Proc. AI CIVIL-COMP '94, Topping B.H.V.& Papadrakakis M. (eds.), Vol. P, pp.215-24.

[73] Camp C., Pezeshk S. & Cao G., 1997
"Design of Framed Structures Using a Genetic Algorithms", in Advances in Structural Optimization, Frangapol D.P. & Cheng F.Y. (eds.), ASCE.

[74] Grierson D.E. & Park K.W., 1997
"Optimal Conceptual Topological Design", in Advances in Structural Optimization, Frangapol D.P. & Cheng F.Y. (eds.), ASCE.

[75] Leithe J.P.B. & Topping B.H.V., 1995
"Improved Genetic Operators for Structural Engineering Design" in Proceedings of the 4th Int. Conf. on the Application of Artificial Intelligence to Civil and Structural Engineering (AI CIVIL-COMP '95), pp.

[76] Dunsmore A.J.T., 1992
"The Search for Singularities in Truss Design Using the Genetic Algorithm", M.Sc.Dissertation, Heriot-Watt University, Edinburgh, UK.

[77] Schmit L.A., 1960
"Structural Design by Systematic Synthesis", in Proc. Second National Conf. on Electronic Computation, ASCE, pp.105-32.

[78] Goldberg D.E. & Samtani M.P., 1987
"Engineering Optimization via the Genetic Algorithm", in Proc. 9th Conf. on Electronic Computation, ASCE.

[79] Schwefel H.P., 1989
"" in Proc. of the Int. Conf. on Computer Aided Optimum Design of Structures, Brebbia C.A. & Hernández S. (eds.), pp218-9.

[80] Jenkins W.M., 1991
"Towards Structural Optimization via the Genetic Algorithm", in J. Comput. and Struct., Vol.40, No.5, pp.1321-27.

[81] Jenkins W.M., 1991
"Structural Optimisation with the Genetic Algorithm", in J. Struct. Eng., Vol.69, No.25, pp.418-22.

[82] Rajeev S. & Krishnamoorthy C.S., 1992
"Discrete Optimization of Structures Using Genetic Algorithms", J. Struct. Eng., Vol.118, No.5, pp.418-22.

[83] Cai J. & Thireauf G., 1995
"Solution of Mixed-Discrete Structural Optimisation with Evolution Strategy" in
" in OPTI IV : Proc. of the 4th Int. Conf. on Computer Aided Optimum Design of Structures, S. Hernández,
M. Al-Sayed, C. A. Brebbia, (eds.), pp.19-26.

[84] Grierson D.E. & Pak W.H., 1993
"Optimal Sizing, Geometrical and Topological Design Using a Genetic Algorithm", Structural Optimisation,
Springer-Verlag, Vol.6, pp115-159.

[85] Sugimoto H. & Bianli L., 1997
"Fully-Stressed Design of Framed Structures With Discrete Variables and Application of Genetic
Algorithms", in Advances in Structural Optimization, Frangapol D.P. & Cheng F.Y. (eds.), ASCE.

[86] Kousmousis V.K. & Georgiou P.G., 1994
"Genetic Algorithms in Discrete Optimization of Steel Truss Roofs", J. Comput. Civ. Eng., ASCE, Vol.8,
No.3, pp.309-325.

[87] Kousmousis V.K.& Arsenis J., 1994
"Genetic Algorithms in a Multi-Criterion Optimal Detailing of Reinforced Concrete Members" in 2nd
International Conference on Computational Structures Technology, Topping B.H.V. & Papadrakakis M.
(eds.), Civil-Comp Press, Edinburgh, Vol.P, Part 8, pp.233-240.

[88] Rafiq M.Y., 1995
"Genetic Algorithms in Optimum Design, Capacity Check and Detailing of Reinforced Concrete Columns"
in Structural Optimisation - Computer Aided Optimum Design of Structures VI (OPTI '95), Hernández S.,
Al-Sayed M. & Brebbia C.A. (eds.), Computational Mechanics Publishers, pp.161-9.

[89] Lucas W.K. & Roddis W.M.K., 1996
"Constraint-Based Genetic Algorithm Optimization Applied to Reinforced Concrete Design",
in Information Processing in Civil and Structural Engineering Design, Civil-Comp Press, Edinburgh, UK,
pp.171-5.

[90] Bedford M., 1996
"Gene Machines", Computer Shopper, No.98, pp.549-55.

[91] Parmee I.C. & Bullock G.N., 1993
"Evolutionary Techniques and their Application to Engineering Design", in Advanced Technologies, Behesti
M.R. & Zreik K. (eds.), Elsevier Science, pp.32-42.

[92] Furuta H., Morimoto H., Tonegawa T. & Watanabe E., 1997
"Application of Genetic Algorithms to Light-Up Design of Bridge" in Advances in Structural Optimization,
Frangapol D.P. & Cheng F.Y. (eds.), ASCE.

[93] Hills W. & Barlow M., 1994
"The Application of Simulated Annealing with a Knowledge-Based Layout Design System", in Proc. of 1st
Int. Conf. on Adaptive Computing in Engineering Design and Control, Parmee I.C.(ed.), University of
Plymouth, pp.122-7

[94] Hudson M. & Parmee I.C., 1995
"The Application of Genetic Algorithms to Conceptual Design", Plymouth Engineering Design Centre
(PEDC), University of Plymouth, UK.

[95] Mfinanga J.S., Stockel C.T. & Deakins E., 1991
"The Role of Simulation in Mechanical Engineering Design", Proc. Summer Computer Simulation Conf.,
Baltimore, Maryland, June, pp.22-4

[96] Deb K., 1990
"Optimal Design of a Class of Welded Structures via Genetic Algorithms", in Proc. 34th
AIAA/ASCE/ASME/AHS SDM Conf., ASCE, New York, N.Y., pp.544-54.

[97] Hajela P. & Lee E., 1993
"Genetic Algorithms in Topological Design of Grillage Structures", in Proc. IUTAM Sympos. on Discrete
Struct. Systems, IUTAM, Zakapane, Poland.

[98] Balling R., Day K. & Taber J., 1999
"City Planning Using a Multiobjective Genetic Algorithm and an Interactive Pareto Set Scanner", Proc. Optimization and Control in Civil and Structural Engineering, Civil-Comp Press, pp35-9.

[99] Grierson D.E. & Khajehpour S., 1999
"Multi-Criteria Conceptual Design of Office Buildings Using Adaptive Search", EG-SEA-AI, Warsaw, pp.51-74.

[100] Rafiq M.Y., 2000
"A Design Support Tool For Optimum Building Concept Generation Using a Structured Genetic Algorithm", Int. J. Comp. Intgrtd. Des. & Constrct., *to appear*.

[101] Sisk G.M., Moore C. & Miles J., 1999
"A Decision-Support System for the Conceptual Design of Building Structures Using a Genetic Algorithm", EG-SEA-AI, Warsaw, pp.175-88.

CHAPTER FOUR

[102] Mylius A., 1999
"Erotic Gherkin's High Tech Design (Swiss Re building)", New Civil Engineer, 16 Sept., p.25.

[103] Augenbroe G., 1995
"Design Systems in a Computer Integrated Manufacturing (CIM) Context", in Integrated Construction Information, Brandon P. & Betts M. (eds.), E&FN Spon, London, pp.194-212.

[104] Thorpe A., Baldwin A.N. & Lewis T., 1993
"Data Exchange Formats for Construction Management Information" in Informing Technologies for Construction, Civil Engineering and Transport, Powell J.A. & Day R. (eds.), Brunel University with SERC (now EPSRC), pp.212-9.

[105] Leal D., 1992
"STEP FEA Project Meeting Report 3-7 Feb1992", Ref. R0007, NAFEMS, Birniehill, East Kilbride, Glasgow, G75 0QU.

[106] Stansfield K. (ed.), 2000
Struct. Engr. J., supplement, Vol. 78, No.4, p.12.

[107] IStructE, 1985
"Manual for the Design of Steel Building Structures", IStructE, London.

[108] Reynolds C.E. & Steedman J.C., 1988
"Reinforced Concrete Designers' Handbook", 10th ed., E&FN Spon, London.

[109] Steel Construction Institute, 1989
"Steel Designers' Manual", 4th revised ed., BSP Professional Books, Oxford.

[110] SETO Ltd., 1995
"Aspects of Cladding", IStructE Special Report, 11 Upper Belgrave St, London SW1X 8BH.

[111] Goodchild C.H., 1997
"Economic Concrete Frame Elements", British Cement Association, Pub.No. 97.358, produced on behalf of the RCC.

[112] Goodchild C.H., 1993
"Cost Model Study – a report on the comparitive costs of concrete and steel framed office buildings", Pub. British Cement Association, Century House, Telford Ave., Crowthorne, Berks RG11 6YS, Tel: 0344-762676, ISBN 0-7210-1469-0.

[113] British Steel Corp., 1993
"Steel or Concrete – The Economics of Commercial Buildings", pub. British Steel Corp., Sections, Plates and Commercial Steels Division, P.O.Box 24, Steel House, Redcar, Cleveland, TS10 5QU.

[114] Whitelaw J.(ed.), 1999
"Breaking the Mould (Project 10-Bishopsgate)", Demonstration Projects Year One, New Civ. Engr. Supplement, Nov, pp.18-9.

[115] Reid E., 1984
"Understanding Buildings – A Multidisciplinary Approach", Construction Press, London.

[116] Jones V., 1984
"Neufert Architects' Data – Handbook of Building Types", 2nd ed., Collins.

CHAPTER FIVE

[117] Schmidt G., 1988
"Microcomputer Aided Design for Architects and Designers", John Wiley and Sons, Chichester.

[118] Balachandran M., 1993
"Knowledge-Based Optimum Design: Topics in Engineering Vol. 10", Computational Mechanics Publications, Southampton.

[119] Mathews J.D. & Rafiq M.Y., 1994
"Adaptive Search for Decision Support in the Preliminary Design of Structural Systems", in ACEDC '94: Proc. 1st Int. Conf. Adapt. Comput. for Eng. Des. Ctrl., Parmee I.C. (ed.)., pp.169-75.

[120] Mathews J.D. & Rafiq M.Y., 1995
"Adaptive Search to Assist in the Conceptual Design of Concrete Buildings", in AI CIVIL-COMP '95: Proc. 4th Int. Conf. Applic. AI to Civ. Struct. Eng^{g}., Topping B.H.V. (ed.), Vol. F., pp.179-87.

CHAPTER SIX

[121] Bullock G.N.B., Denham M.J., Parmee I.C. and Wade J.G., 1995
"Developments in the Use of the Genetic Algorithm in Engineering Design" in Design Studies, Vol.16, No.4, October, pp.507-25.

[122] E. & F.N.Spon. (pub.), 2000
"Spon's Architects' and Builders' Price Book", 125th ed., Davis, Langdon & Everest CQS (eds.).

[123] Glenigan Cost Information (pub.)., 2000
"Griffith's Complete Building Price Book", 46th ed., Denley King Partnership CQS (eds.).

[124] Jenkins W.M., 1995
"Neural Network-Based Approximations for Structural Analysis", in Proc. 4th Int. Conf. Applic. A.I. Civ. Struct. Eng^{g}., Vol. F : Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineers, Topping, B.H.V.T. (ed.), Civil-Comp Press, pp.25-35.

[125] Rafiq M.Y. & Williams C., 1998
"An Investigation into the Integration of Neural Networks with the Structured Genetic Algorithm to Aid Conceptual Design", in Artificial Intelligence in Structural Engineering – Information Technology for Design, Collaboration, Maintenance and Monitoring, (Lecture Notes in AI 1454), Smith I. (ed.), Springer, pp.295-307.

[126] Rafiq M.Y., Bugmann G. & Easterbrook D.J., 2000
"Artificial Neural Networks to Aid Conceptual Design" in J. IStructE, Vol.78, No.3, 1st Feb.

CHAPTER SEVEN

[127] Rzevski G., 1995
"Artificial Intelligence in Engineering: Past, Present and Future", in Applications of Artificial Intelligence in Engineering X, Rzevski G., Adey, R.A. & Tasso C. (eds.), Computational Mechanics Publications, pp.3-16.

[128] Gero J.S. & Maher M.L. (eds.), 1993
"Modelling Creativity and Knowledge-Based Creative Design", Lawrence Erlbaum Associates Publishers, New Jersey and London, pp.1-6.

[129] Norman D.A., 1986
"Cognitive Engineering" in "User Centred System Design", Norman D.A. & Draper S.W. (eds.), Lawrence Erlbaum Associates Publishers, New Jersey and London, pp.31-61.

[130] Pham D.T. & Yang Y., 1993
"A Genetic Algorithm Based Preliminary Design System", Proc. Instn. Mech. Engrs., Vol.207, Part D, Section J: Automotive Eng., pp.127-33.

[131] MacLeod I.A. & Rafiq M.Y., 1988
"Integrated Computer Aided Design of Buildings", The Structural Engineer, Vol.66, No.20, pp.325-30.

[132] Plant A., 1995
"Stimulus Paper: A Pragmatic Approach to VR Tools for Telecoms Engineering", in Virtual Reality and Rapid Prototyping, from Proc. Information Technology Awareness in Engineering, Powell J.A. (ed.), University of Salford / EPSRC / DRAL (now CLRC), p.125

[133] Russell L., 1996
"Reflex Action", NCE IT Supplement, 12/26 Dec., pp.VI-VIII.

CHAPTER EIGHT

[134] Mathews J.D., Rafiq M.Y. & Bullock G.N., 1996
"A Prototype for a Conceptual Structural Building Design System Using the Genetic Algorithm", in ACEDC '96: Proc. 2nd Int. Conf. Adapt. Comput. for Eng. Des. Ctrl.,
Parmee I.C. (ed.), pp.287-90.

[135] Mathews J.D. & Rafiq M.Y., 1996
"Integration and Optimisation in Conceptual Building Design Using Genetic Algorithms"
in "Knowledge-Based Systems in Structural Engineering: Current Developments and Future Trends ", Anumba C.J. (ed.)., IStructE seminar 11/09, pp.63-73.

[136] Rafiq M.Y. & Mathews J.D., 1998
"An Integrated Approach to Structural Design of Buildings Using a Structured Genetic Algorithm" in J. Intgrtd. Des. & Proc. Tech., Vol. 2., No.3, pp.20-31.

CHAPTER NINE

[137] Gero J.S., 1998
"Conceptual Design as a Sequence of Situated Acts", in Artificial Intelligence in Structural Engineering – Lecture Notes in AI 1454, Soringer-Verlag, pp.165-77.

[138] Koo T.K., Tiong R.L.K., Wong T.F. & Tay M.C., 1992
"Design and Development of an Intelligent System for Formwork Selection",
Proc. Instn Civ Engrs Structs & Bldgs, Vol. 94 Feb, Paper 9755, pp.73-91.

[139] Cagdas G. & Cankaya N., 1993
"Designing Vertical Circulation Systems in High-rise Buildings: An Expert System" in Advanced Technologies, Behesti M.R. & Zreik K. (eds.), Elsevier Science, pp.335-40.

[140] Kim, P.C., Soh, C.K. & Broms B.B., 1990
"A Knowledge-Based Approach to Foundation Design", J. Comp. Aid. Engrg., Vol.7, No.6, Dec., pp.165-172.

[141] Park K.W. & Grierson D.E., 1999
"Pareto-Optimal Conceptual Design of the Structural Layout of Buildings Using a Multicriteria Genetic Algorithm", J. Comp. Aid. Civ. & Infrstrct. Eng^{g}., Vol. 14, pp.163-170.

[142] Richie I., 1995
"Collaboration", Struct Engr. J., Vol.73, No.11, pp.191-2.

Appendix A – Sample CONFIG.INI file

CONFIG.INI is used to manipulate and describe design domain relationships. The data shown was used in Example 1 in section 8.3. The length of this file precludes it from being shown in its entirety. Here, only half of the data, relating to concrete construction options (Gene 0 – Gene 17), is presented.

```
[General]
NoOfGenes=36
IsMaximization=1

[Gene000]
IDNo=0
Name=Root
IsBinaryEncoded=1
Description1=Switches between steel frame and concrete frame construction.
Description2=
Description3=
Description4=
GeneType=Switch
IsAvailable=1
UsedInSoln=1
HasAParent=0
ParentGene=
NoChildren=2
ChildGenes=001,018
NoActiveChildren=2
ActiveChildGenes=001,018
DefaultSwitchValue=1

[Gene001]
IDNo=1
Name=Concrete Frames
IsBinaryEncoded=1
Description1=Switches between concrete frame-compatible floor systems including:
Description2=Insitu Floor Slab,  Precast R.C. Floor Slab,  Prestressed Floor Slab.
Description3=
Description4=
GeneType=Switch
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=0
NoChildren=3
ChildGenes=002,008,013
NoActiveChildren=3
ActiveChildGenes=002,008,013
DefaultSwitchValue=1

[Gene002]
IDNo=2
Name=Insitu Concrete Floor
IsBinaryEncoded=1
Description1=Groups together parameters associated with Insitu Floor Slab construction.
Description2=
Description3=
Description4=
GeneType=Group
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=1
NoChildren=5
ChildGenes=003,004,005,006,007
NoActiveChildren=5
ActiveChildGenes=003,004,005,006,007

[Gene003]
IDNo=3
Name=Grid Spacing X--X
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building X--X direction.
Description2=The default value range is 4-11m at 1m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=2
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=1
AllowRange=1
DefaultParameterType=Range
```

ParameterType=Range
NoOfDiscreteValues=1
DefaultDiscrete=3.0,6.0,9.0
Discrete=4.0
NoOfRangeValues=
DefaultRange=
Range=4.0,11.0,1.0

[Gene004]
IDNo=4
Name=Grid Spacing Y--Y
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building Y--Y direction.
Description2=The default value range is 4-11m at 1m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=2
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Discrete
ParameterType=Range
NoOfDiscreteValues=1
DefaultDiscrete=3.0,6.0,9.0
Discrete=8.0
NoOfRangeValues=
DefaultRange=4.0,11.0,1.0
Range=4.0,11.0,1.0

[Gene005]
IDNo=5
Name=Building Dimension X--X
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the X--X direction in metres.
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=2
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=1
DefaultDiscrete=
Discrete=15.0
NoOfRangeValues=
DefaultRange=4.0,11.0,1.0
Range=15.0,90.0,5.0

[Gene006]
IDNo=6
Name=Building Dimension Y--Y
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the Y--Y direction in metres.
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=2
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=1
Discrete=15.0
NoOfRangeValues=
DefaultRange=15.0,90.0,5.0
Range=15.0,90.0,5.0

[Gene007]
IDNo=7
Name=Insitu Slab Depth

168

```
IsBinaryEncoded=1
Description1=Controls the depth of the insitu concrete slab in metres.
Description2=The default value range is 100-380mm in 40mm intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=2
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=1
Discrete=100.0
NoOfRangeValues=
DefaultRange=15.0,90.0,5.0
Range=100.0,380.0,40.0

[Gene008]
IDNo=8
Name=Precast R.C. Floor
IsBinaryEncoded=1
Description1=Groups together parameters associated with Precast R.C. Floor Slab construction.
Description2=
Description3=
Description4=
GeneType=Group
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=1
NoChildren=4
ChildGenes=009,010,011,012
NoActiveChildren=4
ActiveChildGenes=009,010,011,012

[Gene009]
IDNo=9
Name=Grid Spacing X--X
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building X--X direction.
Description2=The default value range is 4-11m at 1m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=8
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=100.0,380.0,40.0
Range=4.0,11.0,1.0

[Gene010]
IDNo=10
Name=Grid Spacing Y--Y
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building Y--Y direction.
Description2=The default value range is 4-11m at 1m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=8
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=4.0,11.0,1.0
```

169

Range=4.0,11.0,1.0

[Gene011]
IDNo=11
Name=Building Dimension X--X
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the X--X direction in metres.
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=8
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=7
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=4.0,11.0,1.0
Range=15.0,90.0,5.0

[Gene012]
IDNo=12
Name=Building Dimension Y--Y
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the Y--Y direction in metres.
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=8
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=2
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=15.0,90.0,5.0
Range=15.0,90.0,5.0

[Gene013]
IDNo=13
Name=Prestressed Floor
IsBinaryEncoded=1
Description1=Groups together parameters associated with Prestressed Floor Slab construction.
Description2=
Description3=
Description4=
GeneType=Group
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=1
NoChildren=4
ChildGenes=014,015,016,017
NoActiveChildren=4
ActiveChildGenes=014,015,016,017

[Gene014]
IDNo=14
Name=Grid Spacing X--X
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building X--X direction.
Description2=The default value range is 3.5-14.0m at 1.5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=13
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=7
AllowRange=1
DefaultParameterType=Range

170

```
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=15.0,90.0,5.0
Range=3.5,14.0,1.5


[Gene015]
IDNo=15
Name=Grid Spacing Y--Y
IsBinaryEncoded=1
Description1=Controls the width of bays in metres in the building Y--Y direction.
Description2=The default value range is 3.5-14.0m at 1.5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=13
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=7
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=3.5,14.0,1.5
Range=3.5,14.0,1.5


[Gene016]
IDNo=16
Name=Building Dimension X--X
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the X--X direction in metres.
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=13
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=3.5,14.0,1.5
Range=15.0,90.0,5.0


[Gene017]
IDNo=17
Name=Building Dimension Y--Y
IsBinaryEncoded=1
Description1=Controls the overall building dimension in the Y--Y direction in metres
Description2=The default value range is 15m-90m at 5m intervals.
Description3=
Description4=
GeneType=Parameter
IsAvailable=1
UsedInSoln=1
HasAParent=1
ParentGene=13
NoChildren=0
ChildGenes=
NoActiveChildren=0
ActiveChildGenes=
DefaultParameterValue=0
AllowRange=1
DefaultParameterType=Range
ParameterType=Range
NoOfDiscreteValues=0
Discrete=
NoOfRangeValues=
DefaultRange=15.0,90.0,5.0
Range=15.0,90.0,5.0
```

Appendix B – Sample SETTINGS.INI file

SETTINGS.INI stores control parameters, unit cost data, structural design parameters and other miscellaneous design parameters used by the GA. The data shown was used in Example 1 in section 8.3. (Shown complete).

```
[GA Domain]
Domain=BuildingDomain

[GA Mode]
Mode=GA
Test Exhaustive Search Steps=1
Save Errors From Exhaustive Search=0
Save Best Chromosome From Exhaustive Search=1

[GA Variables]
Population Size=50
Number of Generations=30
Number of Runs=3
Crossover Probability=0.800
Mutation Probability=0.020
Penalty Function Coefficient= 1.50

[GA Options]
Selection Method=1
Crossover Method=1
Random Number Generation Method=0
Apply Tournament Preselection=1
Apply Global Elitism=1
Apply Correction for Bias at Crossover=0
Random Seed Value=0.1230

[Unit Cost Options]
Normal weight concrete=60.00
Light weight concrete=75.00
Reinforcing Steel=780.00
Formwork=10.00
Profiled Steel Decking=15.00
Universal Beam=780.00
Universal Column=780.00
PC Hollow to 5.01m span=10.00
PC Hollow to 6.21m span=22.00
PC Hollow to 7.71m span=24.00
PC Hollow to 8.81m span=26.00
PC Hollow to 9.31m span=28.00
PC Solid to 5.01m span=20.00
PC Solid to 6.21m span=11.00
PC Solid to 7.71m span=12.00
PC Solid to 8.81m span=13.00
PC Solid to 9.31m span=14.00
PS Hollowcore to 3m span= 30.40
PS Hollowcore to 6m span= 30.80
PS Hollowcore to 7.5m span= 31.70
PS Hollowcore to 9.5m span= 35.90
PS Hollowcore to 12m span= 40.00
PS Hollowcore to 13m span= 42.10
PS Hollowcore to 14m span= 45.70
PS Hollowcore to 15m span= 46.40
Roof=10.00
Cladding=15.00
Foundations=100.00
Land=5000.00
Rentable Value= 80.00
Ignore revenue from excess space=0

[Miscellaneous Options]
Concrete grade=30.00
Steel yield strength=460.00
Column percentage reinforcement limit= 4.00
Lost rentable space at column= 1.00
Live load= 3.50
Required floor area=40000.00
Maximum height= 999.00
Clear height=2.7
Building life=25
```

Appendix C – Sample DETAILS.DAT file

DETAILS.DAT contains the details of a conceptual design. The data was used in Example 1 in section 8.3. (Shown complete).

```
Chromo: 2  2,G,5,3,13  14,1,G,3,4  0,15,G,7,6  13,2,4,G,I  6,7,11,G,5  6,13,2,G,6  4,4,1,0,1

Interpreting Design
=======================
Using Steel Frame
Using Composite Steel Deck Floor
XGrid Dimension:   10.00
YGrid Dimension:    8.00
BldgLen Dimension:   35.00
BldgWid Dimension:   20.00
DeckConcTypeCode: 0
DeckSpan: 3000.00

Setting Building Dimensions
============================
Adjusted building x-dim [m] =   35.00
Adjusted building y-dim [m] =   20.00
No of bays in x direction = 4
No of bays in y direction = 3
Grid spacing x-dir / y-dir [m]=  8.75   6.67
Local slab x-span / y-span [m]=  6.67   8.75

Sizing Composite Deck Floor
============================
Using sheets with max span [m]=  3.00 requiring 2 secondary beams per panel.
Designing secondary beams parallel to short slab side.
Ultimate load [kN/m2]= ( 1.4 * 2.40 + 1.6 * 3.50 ) =   8.96
Beam loaded width [m]= 2.92 gives beam load per metre run (kN/m)= 26.13
Beam length (span)[m]= 6.67 gives Max BM (at beam midspan) [kNm]= 145.19

Sizing Steel Secondary Beam
============================
Z-permissible [cm3]=  527.89      Z-actual [cm3]=  572.00
Using an 356x127x39.0kg/m beam

Designing Main Beam
====================
Designing main beams parallel to long slab side.
Estimating the point load from a Concrete Secondary Beam on a Concrete Main Beam beam.
Sec Beam UDL Per Metre [kN/m]=  26.13
Sec Beam Wt Per Metre [kN/m]=   0.39
Main Beam Point Load [kN]=  176.82
Loaded width [m]=   6.67
Max BM  from point load [kNm]= 515.73

Sizing Steel Main Beam
======================
Z-permissible [cm3]= 1875.20      Z-actual [cm3]= 2080.00
Using an 533x210x92.0kg/m beam

Sizing a Tie beam
==================
Links columns in transverse direction. Non load bearing. (minimum size)
Using an 203x102x23.0kg/m beam

Sizing an Edge beam
===================
As main beam
Using an 533x210x92.0kg/m beam

Calculating Storey Heights
===========================
Building Height [m] =  165.30
Storey Height   [m] =    2.85
Number of floors    = 58
Floor space required [m2] = 40000.00
Gross area-one floor [m2] =   700.00
Gross area-total     [m2] = 40600.00

Estimating Quantities and Costs for Composite Deck floor
==========================================================
NWC in Deck          Quantities   [m3]: Gv/It/Sy/Bd =    0.10    5.83     70.00    4060.00
NWC in Deck          Costs       [GBP]: Gv/It/Sy/Bd =    6.00  350.00   4200.00  243600.00
Steel in Deck        Quantities   [m2]: Gv/It/Sy/Bd =    1.00   58.33    700.00   40600.00
Steel in Deck        Costs       [GBP]: Gv/It/Sy/Bd =   15.00  875.00  10500.00  609000.00
Composite Deck floor        (No of units):  Cm/Sy/Bd =            1       12        696
Composite Deck floor  Costs     [GBP]:      Cm/Sy/Bd =      1225.00  14700.00  852600.00

Estimating Quantities and Costs for Steel Secondary beam
Steel Secondary beam     [No of units]:  Cm/Sy/Bd =            1       24       1392
Steel Secondary beam  Costs     [GBP]:   Cm/Sy/Bd =       195.00   4680.00  271440.00

Estimating Quantities and Costs for Steel Main beam
Steel Main beam          [No of units]:  Cm/Sy/Bd =            1       16        928
```

```
Steel Main beam        Costs    [GBP]:    Cm/Sy/Bd =              603.75   9660.00 560280.00

Estimating Quantities and Costs for Steel Tie beam
Steel Tie beam          [No of units]:   Cm/Sy/Bd =                  1        9      522
Steel Tie beam         Costs    [GBP]:    Cm/Sy/Bd =              115.00   1035.00  60030.00

Estimating Quantities and Costs for Steel Edge beam
Steel Edge beam         [No of units]:   Cm/Sy/Bd =                  1        6      348
Steel Edge beam        Costs    [GBP]:    Cm/Sy/Bd =              603.75   3622.50 210105.00

Calculating column loads.
Main Beam Self Wt/m [kN/m]=   0.92
Effective length of main beam carried by an internal column [m]=  6.67 +   8.75 =   15.42
Main beam Self Wt [kN]=   14.18
Load on Main Beams [kN]=  353.64
Axial Load on Column [kN]=  367.83
Sizing columns of a 58 storey building
No of columns in the x-direction/y-direction/per Storey = 5   4   20

Column 1 of 15 (runs from floor 56 to the Roof (floor 58))
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.152 x  0.152 x 23.000kg/m.  Axial load [kN]=     378
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       40
Steel column           Costs    [GBP]:   Cm/Sy/St =               47.44   948.75   1897.50

Column 2 of 15 (runs from floor 52 to floor 56)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.204 x  0.206 x 52.000kg/m.  Axial load [kN]=    1849
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              107.25  2145.00   8580.00

Column 3 of 15 (runs from floor 48 to floor 52)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.209 x  0.222 x 86.000kg/m.  Axial load [kN]=    3320
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              177.38  3547.50  14190.00

Column 4 of 15 (runs from floor 44 to floor 48)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.368 x  0.356 x 129.000kg/m.  Axial load [kN]=    4792
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              266.06  5321.25  21285.00

Column 5 of 15 (runs from floor 40 to floor 44)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.265 x  0.289 x 167.000kg/m.  Axial load [kN]=    6263
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              344.44  6888.75  27555.00

Column 6 of 15 (runs from floor 36 to floor 40)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.314 x  0.340 x 198.000kg/m.  Axial load [kN]=    7734
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              408.38  8167.50  32670.00

Column 7 of 15 (runs from floor 32 to floor 36)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.395 x  0.381 x 235.000kg/m.  Axial load [kN]=    9206
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              484.69  9693.75  38775.00

Column 8 of 15 (runs from floor 28 to floor 32)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.322 x  0.365 x 283.000kg/m.  Axial load [kN]=   10677
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              583.69 11673.75  46695.00

Column 9 of 15 (runs from floor 24 to floor 28)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.403 x  0.406 x 340.000kg/m.  Axial load [kN]=   12148
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              701.25 14025.00  56100.00

Column 10 of 15 (runs from floor 20 to floor 24)
Sizing Steel column
Column dims [m]x[m]x[kg]:  0.407 x  0.419 x 393.000kg/m.  Axial load [kN]=   13620
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
Steel column           Costs    [GBP]:   Cm/Sy/St =              810.56 16211.25  64845.00
Column 11 of 15 (runs from floor 16 to floor 20)

Sizing Steel column
Column dims [m]x[m]x[kg]:  0.407 x  0.419 x 393.000kg/m.  Axial load [kN]=   15091
Estimating Quantities and Costs for Steel column
Steel column            [No of units]:   Cm/Sy/St =                  1       20       80
```

```
Steel column           Costs      [GBP]:    Cm/Sy/St =            810.56  16211.25  64845.00
Column 12 of 15 (runs from floor 12 to floor 16)


Sizing Steel column
Column dims [m]x[m]x[kg]:  0.412 x  0.437 x 467.000kg/m.  Axial load [kN]=   16562
Estimating Quantities and Costs for Steel column
Steel column               [No of units]:    Cm/Sy/St =             1      20       80
Steel column           Costs      [GBP]:    Cm/Sy/St =            963.19  19263.75  77055.00
Column 13 of 15 (runs from floor 8 to floor 12)


Sizing Steel column
Column dims [m]x[m]x[kg]:  0.412 x  0.437 x 467.000kg/m.  Axial load [kN]=   18034
Estimating Quantities and Costs for Steel column
Steel column               [No of units]:    Cm/Sy/St =             1      20       80
Steel column           Costs      [GBP]:    Cm/Sy/St =            963.19  19263.75  77055.00
Column 14 of 15 (runs from floor 4 to floor 8)


Sizing Steel column
Column dims [m]x[m]x[kg]:  0.418 x  0.456 x 551.000kg/m.  Axial load [kN]=   19505
Estimating Quantities and Costs for Steel column
Steel column               [No of units]:    Cm/Sy/St =             1      20       80
Steel column           Costs      [GBP]:    Cm/Sy/St =           1136.44  22728.75  90915.00
Column 15 of 15 (runs from  Ground Floor (floor 0) to floor 4)


Sizing Steel column
Column dims [m]x[m]x[kg]:  0.418 x  0.456 x 551.000kg/m.  Axial load [kN]=   20976
Estimating Quantities and Costs for Steel column
Steel column               [No of units]:    Cm/Sy/St =             1      20       80
Steel column           Costs      [GBP]:    Cm/Sy/St =           1136.44  22728.75  90915.00
Total Column Cost [GBP]= 713378
Calculating total frame costs.
Floor Slab/Main beam/Secondary beam/Tie beam/Edge beam/Columns/Frame [GBP]=
852600  560280  271440   60030  210105  713377   2667832


Roofing
=======
Roof Area [m2]=  700.00
Unit cost [GBP_m2]=   10.00
Roof cost [GBP]= 7000.00


Cladding
========
Cladding Area [m2]= 17864.00
Unit cost [GBP_m2]=   15.00
Cladding cost [GBP]= 267960.00


Foundations
===========
Ultimate Load from column [kN]= 20976.18
Unit cost [GBP_m3]=  100.00
Pad footing area [m2]=  104.88
Cost of one pad footing [GBP]= 5244.05
No. of pad footings = 20
Foundation cost [GBP]= 104880.92


Land
====
Area of land [m2]=  700.00
Unit cost [GBP_m2]= 5000.00
Land costs [GBP]= 3500000.00


Calculating Capital Cost
========================
Costs: Frame/Roof/Cladding/Foundations/Land/Total [GBP]=
2667832    7000  267960  104880  3500000   6547673


Calculating Returns
===================
Floor space required [m2]  = 40000.00
Gross area           [m2]  = 40600.00
Nett lettable area   [m2]  = 39317.20
Revenue-making area  [m2]  = 39317.20
Rent [GBP_m2yr]            =    80.00
Building life [yr]         =    25
Annual revenue [GBP_yr]    = 3145376.10
Income revenue [GBP]       = 78634402.49
Profit [GBP]               = 72086729.08
```

175

Appendix D – Sample DESIGN.DAT file

DESIGN.DAT contains the main design details of a conceptual design, used to create a geometrical model (see GEOMETRY.DAT). The data was one produced in Example 1 in section 8.3. (Shown complete).

```
[XDim]
35.00
[YDim]
20.00
[Storey Height]
2.85
[NoFloors]
58
[NoXBays]
4
[NoYBays]
3
[SlabXSpan]
8.75
[SlabYSpan]
6.67
[Slab Depth]
0.1
[AtriumXDim]
0.0
[AtriumYDim]
0.0
```

Appendix E – Sample GEOMETRY.DAT file

GEOMETRY.DAT contains the geometrical model used to create a 2D or 3D view of the design concept. It was created from a DESIGN.DAT file (see Appendix D). Here, the data shown was produced for Example 7(a) in section 8.9. The length of this file precludes it from being shown in its entirety.

```
#
b
1
255 0 0
641
[1=Slab]
      6
      8    48
    0.00      0.00      0.00
    0.00     70.00      0.00
   80.00      0.00      0.00
   80.00     70.00      0.00
    0.00      0.00     -0.20
    0.00     70.00     -0.20
   80.00      0.00     -0.20
   80.00     70.00     -0.20
       6      8      4     -1
       4      2      6     -1
       6      5      7     -1
       7      8      6     -1
       7      3      4     -1
       4      8      7     -1
       1      3      7     -1
       7      5      1     -1
       5      6      2     -1
       2      1      5     -1
       3      1      2     -1
       2      4      3     -1
[2=MainBeam]
      3
      8    48
    9.85      0.00     -0.20
    9.85     70.00     -0.20
   10.15      0.00     -0.20
   10.15     70.00     -0.20
    9.85      0.00     -0.60
    9.85     70.00     -0.60
   10.15      0.00     -0.60
   10.15     70.00     -0.60
       6      8      4     -1
       4      2      6     -1
       6      5      7     -1
       7      8      6     -1
       7      3      4     -1
       4      8      7     -1
       1      3      7     -1
       7      5      1     -1
       5      6      2     -1
       2      1      5     -1
       3      1      2     -1
       2      4      3     -1
[1=MainBeam]
      3
      8    48
   19.85      0.00     -0.20
   19.85     70.00     -0.20
   20.15      0.00     -0.20
   20.15     70.00     -0.20
   19.85      0.00     -0.60
   19.85     70.00     -0.60
   20.15      0.00     -0.60
   20.15     70.00     -0.60
       6      8      4     -1
       4      2      6     -1
       6      5      7     -1
       7      8      6     -1
       7      3      4     -1
       4      8      7     -1
       1      3      7     -1
       7      5      1     -1
       5      6      2     -1
       2      1      5     -1
       3      1      2     -1
       2      4      3     -1
[4=MainBeam]
      3
      8    48
   29.85      0.00     -0.20
   29.85     70.00     -0.20
   30.15      0.00     -0.20
```

```
   30.15     70.00      -0.20
   29.85      0.00      -0.60
   29.85     70.00      -0.60
   30.15      0.00      -0.60
   30.15     70.00      -0.60
      6     8     4    -1
      4     2     6    -1
      6     5     7    -1
      7     8     6    -1
      7     3     4    -1
      4     8     7    -1
      1     3     7    -1
      7     5     1    -1
      5     6     2    -1
      2     1     5    -1
      3     1     2    -1
      2     4     3    -1
[5=MainBeam]
   3
   8     48
   39.85      0.00      -0.20
   39.85     70.00      -0.20
   40.15      0.00      -0.20
   40.15     70.00      -0.20
   39.85      0.00      -0.60
   39.85     70.00      -0.60
   40.15      0.00      -0.60
   40.15     70.00      -0.60
      6     8     4    -1
      4     2     6    -1
      6     5     7    -1
      7     8     6    -1
      7     3     4    -1
      4     8     7    -1
      1     3     7    -1
      7     5     1    -1
      5     6     2    -1
      2     1     5    -1
      3     1     2    -1
      2     4     3    -1
[6=MainBeam]
   3
   8     48
   49.85      0.00      -0.20
   49.85     70.00      -0.20
   50.15      0.00      -0.20
   50.15     70.00      -0.20
   49.85      0.00      -0.60
   49.85     70.00      -0.60
   50.15      0.00      -0.60
   50.15     70.00      -0.60
      6     8     4    -1
      4     2     6    -1
      6     5     7    -1
      7     8     6    -1
      7     3     4    -1
      4     8     7    -1
      1     3     7    -1
      7     5     1    -1
      5     6     2    -1
      2     1     5    -1
      3     1     2    -1
      2     4     3    -1
[7=MainBeam]
   3
   8     48
   59.85      0.00      -0.20
   59.85     70.00      -0.20
   60.15      0.00      -0.20
   60.15     70.00      -0.20
   59.85      0.00      -0.60
   59.85     70.00      -0.60
   60.15      0.00      -0.60
   60.15     70.00      -0.60
      6     8     4    -1
      4     2     6    -1
      6     5     7    -1
      7     8     6    -1
      7     3     4    -1
      4     8     7    -1
      1     3     7    -1
      7     5     1    -1
      5     6     2    -1
      2     1     5    -1
      3     1     2    -1
      2     4     3    -1
[8=MainBeam]
   3
   8     48
   69.85      0.00      -0.20
   69.85     70.00      -0.20
   70.15      0.00      -0.20
   70.15     70.00      -0.20
   69.85      0.00      -0.60
```

```
  69.85    70.00    -0.60
  70.15     0.00    -0.60
  70.15    70.00    -0.60
      6    8       4    -1
      4    2       6    -1
      6    5       7    -1
      7    8       6    -1
      7    3       4    -1
      4    8       7    -1
      1    3       7    -1
      7    5       1    -1
      5    6       2    -1
      2    1       5    -1
      3    1       2    -1
      2    4       3    -1
[9=Column]
   14
    8    48
  -0.20   -0.20    -0.60
  -0.20    0.20    -0.60
   0.20   -0.20    -0.60
   0.20    0.20    -0.60
  -0.20   -0.20    -3.30
  -0.20    0.20    -3.30
   0.20   -0.20    -3.30
   0.20    0.20    -3.30
      6    8       4    -1
      4    2       6    -1
      6    5       7    -1
      7    8       6    -1
      7    3       4    -1
      4    8       7    -1
      1    3       7    -1
      7    5       1    -1
      5    6       2    -1
      2    1       5    -1
      3    1       2    -1
      2    4       3    -1
```

Appendix F – Sample RUNxxxxx.DAT file

The RUNxxxxx.DAT files store design concepts using their original chromomses, during a GA run. A new file is created for each run. Data is shown for RUN00001.DAT, produced during the first of four runs for Example 1 in section 8.3. The length of this file precludes it from being shown in its entirety. Here, only data corresponding to the first generation of the run is shown. This data is re-read/re-loaded in order to review/re-evaluate design concepts, interactively (Requires corresponding CONFIG.INI and SETTINGS.INI files).

```
Generation 000000 000021    68657.93
000000      2      2G.....    0     4     3     0    2G.....    7    0    15    7G.....      0    3
7    12      2G.....   7     6     2     5G.....    6    3    13    7G.....    1     3    10    1
1    1      55219.65
000001      2      2G.....    5     3    12     6    5G.....    0    6     4    15G.....     4    6
1    8      4G.....   6     6    13    15G.....    6    5    5     2G.....    7     4    12    9
1    1      54531.30
000002      1      1G.....    1     6     0    14    2G.....    2    7    13    12G.....     0    7
2    8      1G.....   0     5     7    10G.....    5    0    4     1G.....    1     1    2    14
0    1      64278.74
000003      1      1G.....    0     7     8    13    4G.....    5    4     6    13G.....     0    2
2    4      4G.....   3     2    15    12G.....    2    7    4     14G.....    5     0    3    3
0    0      53445.49
000004      1      4G.....    6     3     0    11    6G.....    6    6    12    10G.....     6    6
10    4      4G.....   6     3     6     1G.....    0    7    9     3G.....    4     3    8    10
1    1      64022.76
000005      1      4G.....    6     6    12    15    2G.....    5    7     7    0G.....      0    6
13    12      2G.....   2     0     4     4G.....    4    4    5     8G.....    0     3    15    0
1    0      45484.95
000006      1      1G.....    7     3    12     9    7G.....    7    7    13    8G.....      5    0
1    7      2G.....   4     0     1     2G.....    3    3    14    12G.....    3     6    14    5
1    0      49775.96
000007      1      2G.....    5     0    14     5    7G.....    6    3     5    13G.....     1    7
2    13      1G.....   5     4     1     4G.....    4    4    0     9G.....    0     7    6    14
1    0      60718.54
000008      2      1G.....    4     4     0     6    5G.....    7    2     5    12G.....     0    0
12    9      4G.....   3     2    12     7G.....    3    0    13    8G.....    3     6    14    2
0    1      64822.91
000009      2      4G.....    3     0     8    13    1G.....    1    4    15    10G.....     2    6
9    13      2G.....   5     4     2     2G.....    0    2    14    15G.....    6     5    11    8
0    0      42355.17
000010      1      2G.....    3     2    11     8    1G.....    3    7     0    2G.....      7    1
9    6      1G.....   1     6    11    12G.....    1    3    2     13G.....    2     1    8    3
0    0       0.00
000011      2      4G.....    1     3    13    11    1G.....    1    1    10    6G.....      7    6
5    0      2G.....   0     0    10    10G.....    1    4    3     14G.....    4     3    4    14
0    1      64827.76
000012      1      1G.....    1     2    10     3    1G.....    6    4    12    1G.....      1    1
1    10      2G.....   1     2     5    14G.....    6    4    11    5G.....    7     4    9    14
1    1      61420.64
000013      2      2G.....    4     3     9     4    5G.....    5    5     6    7G.....      6    5
12    13      2G.....   6     6    15     0G.....    6    3    1     14G.....    2     6    8    12
1    0       0.00
000014      1      2G.....    0     1     4     3    3G.....    2    1     2    12G.....     2    2
5    0      2G.....   4     6     0     0G.....    7    6    2     7G.....    0     0    4    6
0    0      64630.24
000015      2      2G.....    3     4     9     1    4G.....    2    5     6    15G.....     7    6
14    0      1G.....   2     4    12     4G.....    4    1    9     5G.....    4     0    0    9
1    0      65130.14
000016      2      1G.....    4     6     2    13    4G.....    2    2     0    2G.....      7    0
11    7      4G.....   1     6    14     0G.....    0    5    13    15G.....    5     5    12    9
0    1      54609.69
000017      2      1G.....    7     5     3     0    7G.....    0    3     3    12G.....     3    5
12    3      4G.....   2     4    10    14G.....    2    3    2     11G.....    4     3    2    14
1    0      64604.37
000018      2      2G.....    7     0     4    11    2G.....    2    4     9    8G.....      4    1
15    10      2G.....   4     0     6     4G.....    3    6    15    3G.....    7     1    5    13
0    1       0.00
000019      2      2G.....    4     7    10     2    2G.....    5    0    10    6G.....      1    1
6    6      2G.....   3     7    13     7G.....    0    2    8     1G.....    3     2    13    0
0    0      67603.01
000020      1      2G.....    1     6     2     5    3G.....    6    4     3    1G.....      4    2
15    2      1G.....   2     6    13    10G.....    2    2    6     5G.....    0     1    6    11
1    0      63777.97
000021      2      2G.....    1     6     8    13    7G.....    1    1     3    2G.....      4    0
3    2      4G.....   6     1     1     3G.....    6    0    4     2G.....    2     7    6    5
0    0      68657.93
000022      1      1G.....    4     2     1     7    3G.....    0    6     5    5G.....      1    6
9    12      1G.....   0     2     0     1G.....    6    0    7     6G.....    7     5    15    13
1    0      62538.83
000023      1      4G.....    3     7    14    15    6G.....    4    3     0    12G.....     4    0
15    7      4G.....   5     7     4    10G.....    0    4    8     3G.....    7     4    11    10
0    0      58209.05
000024      2      4G.....    4     6     9    11    0G.....    6    1     0    7G.....      1    0
6    1      1G.....   4     2     7     5G.....    7    2    6     10G.....    3     4    4    10
0    0      64359.84
```

180

```
000025      2    4G.....    2    3    8    9   4G.....    5    7    4   2G.....    3    3
14    10    4G.....  1    6   12   6G.....    1    0    5   4G.....    1    5    9    3
0     1    68361.59
000026      1    2G.....    7    7   14    0   5G.....    0    6    5  15G.....    5    3
1     1    4G.....  7    0    6   12G.....    7    0    6   5G.....    0    4    1    6
0     0    61461.62
000027      2    2G.....    7    4    4    7   7G.....    2    6   12  13G.....    0    3
15    7    1G.....  5    0    4   11G.....    4    1   11   12G.....   0    5    6    5
1     0    64967.64
000028      1    2G.....    2    7    8   13   2G.....    4    5   14   3G.....    6    6
11    3    4G.....  2    4    3   5G.....    0    6   12   1G.....    2    3    2   15
0     0    61311.92
000029      1    2G.....    5    6    8   15   2G.....    5    5   11  13G.....    0    6
14    9    4G.....  2    5    6   9G.....    3    3    3   2G.....    1    5   11    7
0     1    55305.45
000030      2    2G.....    6    7    1    2   0G.....    7    0    7   6G.....    5    1
10    13   2G.....  4    4   14   10G.,...    6    5    7   8G.....    6    5   10   11
1     0        0.00
000031      2    2G.....    4    2    4    1   7G.....    1    0   15   3G.....    5    5
12    13   4G.....  0    5    5   8G.....    0    0   11   0G.....    5    1   15    7
1     0    53835.75
000032      1    4G.....    6    0    3   10   7G.....    2    6    1   9G.....    6    2
5     9    4G.....  0    3   13   10G.....   5    1    0   11G.....   1    0    9    7
0     1    62921.66
000033      1    2G.....    2    0    6    6   3G.....    3    1    1  12G.....    4    6
10    13   4G.....  7    0   13   10G.....   4    3    6   0G.....    7    4    9   12
1     1    64757.99
000034      1    2G.....    1    5    4    0   7G.....    4    4   11  12G.....    7    4
3     0    1G.....  6    0   11   9G.....    3    6   11   4G.....    3    3    6    8
1     0    51470.70
000035      1    4G.....    6    3    5   12   7G.....    4    4    8   1G.....    7    2
3     3    4G.....  5    1    9   7G.....    1    0    0   6G.....    2    2   12   11
0     1    66853.46
000036      1    2G.....    7    7   10   13   5G.....    2    0    4   9G.....    5    3
4     5    2G.....  5    2   14   1G.....    5    5    2   3G.....    2    0   13   13
1     1    60258.60
000037      1    2G.....    4    2   15    6   3G.....    7    6   13   1G.....    2    1
7     12   2G.....  7    6    2   3G.....    3    7   14   13G.....   3    3   12    9
0     0    67005.30
000038      1    4G.....    3    1    3    3   3G.....    0    4   13   7G.....    7    1
10    10   4G.....  6    2   12   13G.....   2    1   13   9G.....    1    2    2    6
0     1    57158.63
000039      1    2G.....    3    0    5    2   1G.....    4    6   15  12G.....    4    0
5     13   1G.....  0    2   11   1G.....    2    2    7   3G.....    1    7    3    6
1     0    41850.35
000040      1    4G.....    1    4    0   14   6G.....    6    6    6  11G.....    3    4
10    8    1G.....  6    4    6   3G.....    0    5    8   2G.....    6    0    4   14
0     0    61478.59
000041      1    1G.....    5    3    3    4   7G.....    3    6   13   6G.....    3    7
7     0    2G.....  6    2    7   1G.....    5    2    6   6G.....    5    5    2   12
0     1    63008.50
000042      2    2G.....    0    7    0    3   4G.....    1    5   13   3G.....    4    0
5     9    4G.....  2    1    8   0G.....    7    3   10   3G.....    3    7    3   12
0     0    65370.84
000043      2    4G.....    5    7    9   14   3G.....    4    6    3   5G.....    5    6
7     1    1G.....  5    1   15   11G.....   3    3    6   4G.....    0    0   10    1
0     0    52098.28
000044      2    1G.....    2    3    8    9   5G.....    5    1    7   2G.....    0    1
12    8    2G.....  0    2    2   6G.....    2    1    6   4G.....    1    4    8    4
0     0    68052.92
000045      1    4G.....    2    3    0    4   3G.....    7    7    8   1G.....    2    0
9     6    2G.....  7    0    1   7G.....    4    3   10   5G.....    7    0   11   11
0     0    58596.26
000046      1    4G.....    5    5    5   12   0G.....    5    2    3   9G.....    7    2
10    13   4G.....  3    0    6   2G.....    7    5   13   3G.....    7    3    0   10
0     0    51806.90
000047      1    4G.....    4    5   15    2   0G.....    2    5    8  13G.....    6    5
14    9    4G.....  4    3    7   7G.....    0    0   13   2G.....    0    2    1    1
1     0    51092.99
000048      2    4G.....    7    6    5   11   1G.....    2    0   15   5G.....    7    5
14    1    4G.....  6    5    2   9G.....    2    7   15   13G.....   4    4   14   12
0     0    53033.08
000049      2    4G.....    2    5    3   14   1G.....    6    6   11  15G.....    6    2
13    2    1G.....  0    7    9   8G.....    3    2   11   7G.....    2    7    6    2
1     0    63823.65
Generation 000001 000001  68851.73
000000      1    2G.....    0    1    4    3   3G.....    2    1    2  12G.....    2    2
5     0    2G.....  4    6    0   0G.....    7    6    2   7G.....    0    0    4    6
0     0    64630.24
000001      2    4G.....    2    3    8    8   5G.,...    0    6    4  15G.....    4    6
0     8    4G.....  6    6   14   6G.....    1    0    5   4G.....    3    5    9    3
1     1    68851.73
000002      2    2G.....    5    3   12    3   4G.....    5    7    4   2G.....    3    3
14    10   4G.....  1    6   13   15G.....   6    5    5   2G.....    7    4   12    1
1     1    68634.00
```

181

Appendix G - Features and Benefits of the DPRO GA-based design tool

- It is possible to manipulate GA control parameters, unit cost data, structural design parameters and design criteria.

- Multiple modes of operation are provided. Constrained exhaustive search mode enables true optima to be determined.

- Data management tools enable individual design domains to be easily updated.

- Real-time control is provided, allowing processed to be activated and halted in a convenient manner. The GA can be slowed down and execution can be programmed to stop on certain conditions, or at certain stages.

- Graphical views provide real-time information, including details of the best design found so far, and a convergence plot. Diagnostic information can be turned on or off.

- Post-processing support includes allowing designs produced earlier during the course of the genetic experiment to be reviewed / stored to file if required. An output file can be created that enables a design to be visualised. The results of an earlier experiment are saved and can be reloaded.

- Binary and real encoding is supported.

- The GA permits extension of the system to other fields that use a hierarchical or heterachical domain knowledge.