A NEW UNIFORM ORDER-BASED CROSSOVER OPERATOR FOR GENETIC ALGORITHM APPLICATIONS TO MULTI-COMPONENT COMBINATORIAL OPTIMIZATION PROBLEMS

by

FUNDA SİVRİKAYA-ŞERİFOĞLU

B.S. in I.E., Boğaziçi University, 1989

M.S. in E.E.S. Stanford University, 1990

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Doctor

of

Philosophy

Boğaziçi University

1997

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my dissertation supervisor Prof. Dr. Gündüz Ulusoy for introducing me to genetic algorithms, for providing invaluable guidance and support throughout this work, and for sharing many reflections on various complex adaptive systems in general and on life in particular.

I am grateful to all of the members of my thesis committee for the time they devoted to read and discuss my dissertation and for their comments and suggestions.

I would like to thank Assoc. Prof. Linet Özdamar and Assoc. Prof. Ümit Bilge for providing test problems to be used in this work and for useful comments and discussions. Assoc. Prof. Linet Özdamar also provided the code of the heuristic LCBA.

I am very grateful to three women for the invaluable encouragement they provided especially at the start of this work: Assoc. Prof. Füsun Ülengin, Assoc. Prof. Tülin Yazgaç and my mother, Med. Dr. Nebahat Sivrikaya.

Med. Dr. Nebahat Sivrikaya has been and continues to be a role model for me with the quality of her teaching and the generosity of her motherhood.

This work is partially supported by Fahir İlkel Doktora Bursu, Boğaziçi University Foundation.

Finally, my special thanks go to Şerifoğlu Ağaç San. A.Ş. for making their resources available to me whenever I needed them, and to Büke and Mehmet Şerifoğlu for tolerating my busy schedule.

A NEW UNIFORM ORDER-BASED CROSSOVER OPERATOR FOR GENETIC ALGORITHM APPLICATIONS TO MULTI-COMPONENT COMBINATORIAL OPTIMIZATION PROBLEMS

FUNDA SİVRİKAYA-ŞERİFOĞLU

Industrial Engineering, Ph.D. Thesis, 1997

Dissertation Supervisor: Prof. Dr. Gündüz ULUSOY

Keywords: Crossover operators, genetic algorithms, combinatorial optimization

In this thesis, a new uniform order-based crossover operator to be used in genetic algorithm applications to combinatorial optimization problems is presented. There are three pure types of combinatorial problems: assignment, sequencing and selection problems. Most real world combinatorial problems involve components from these three types at the same time. The new operator is applicable to multi-component combinatorial optimization problems which involve one sequencing and one or more selection components. Examples are in abundance and include communications network design problems, machine and project scheduling problems. The operator is called the multi-component uniform order-based crossover or MCUOX. MCUOX processes ordering and value information concurrently. It works within a natural and direct representation of the problems. It is both general and powerful by being able to effectively search all the components of the problem. It enhances building block propagation, never violates precedence constraints given the parents are precedence feasible, and is able to deal with additional selection components without any significant overhead. The application and effectiveness of MCUOX is illustrated at the hand of three well known difficult problems: simultaneous scheduling of machines and automated guided vehicles, resource constrained project scheduling problem with discounted cash flows, and parallel machine scheduling with earliness and tardiness penalties.

GENETİK ALGORİTMALARIN ÇOK-BİLEŞENLİ KOMBİNATORYAL PROBLEMLERE UYGULANMASI İÇİN YENİ BİR DÜZGÜN, SIRALAMA TEMELLİ ÇAPRAZLAMA OPERATÖRÜ

FUNDA SİVRİKAYA-ŞERİFOĞLU

Endüstri Mühendisliği, Doktora Tezi, 1997

Tez Danışmanı: Prof. Dr. Gündüz ULUSOY

Anahtar Kelimeler: çaprazlama operatörü, genetik algoritma, kombinatoryal optimizasyon

Bu tezde, genetik algoritmaların kombinatoryal problemlere uygulanmasında kullanılmak üzere yeni bir düzgün, sıralama temelli çaprazlama operatörü sunulmaktadır. Pratikteki kombinatoryal problemlerin çoğu; atama, sıralama ve seçme problemlerinden oluşan üç temel tip kombinatoryal problemin iki veya üçünü aynı anda içerir. Yeni operatör, bir sıralama ve bir ya da birden fazla seçme bileşeni olan kombinatoryal problemler için geliştirilmiştir. Bu tür problemlerin komünikasyon ağları dizaynı, makina ve proje çizelgelemesi gibi pek çok örneği vardır. Yeni operatöre çok-bileşenli, düzgün, sıralama-temelli çaprazlama (MCUOX) adı verilmiştir. MCUOX sıralama ve değer bilgilerini aynı anda işleyebilen, problemin tüm bileşenlerini etkili arayabilen genel bir operatördür. Yapı taşlarının yeni nesillere taşınmasını sağlar ve öncelik kısıtlarını ihlal etmez. Seçme problemlerinin sayısı, fazladan hesap yüküne sebep olmadan artırılabilir. MCUOX operatörünün uygulanırlığı ve etkenliği, iyi bilinen şu üç zor problem üzerinde gösterilmiştir: makinalar ile otomatik güdümlü araçların eşzamanlı çizelgelenmesi, erken ve geç bitirme cezaları içeren paralel makina çizelgelemesi ve kaynak kısıtları altında iskontalanmış nakit akışlı proje çizelgelemesi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Integration is the term describing the global trend in the closing decades of twentieth century. The collapse of political borders and the diminishing importance of geographical distances as a result of fast and reliable information tools and networks are two concrete facets of this phenomenon. In the most general sense, integration implies that 'systems' - whether they be political, financial, geographical, or scientific - are not confined to disjointly operate within their strictly defined boundaries any more, and that interactions among them are becoming the rule rather than the exception. For example, there is no more a 'black' communist political system at one extreme and a 'white' liberalist political system at the other; but several tones of 'grey' political systems between the two.

In the context of sciences, interactions among systems occur in the form of exchanges and cross-fertilizations of ideas. Even those branches which, a couple of decades ago, had been considered to be totally different in character, are now borrowing from and lending to each other many ideas. This leads to the development of new approaches and the rise of new approaches in turn stimulates further interactions.

Genetic algorithms (GAs), the central theme of this work, stem from such interactions between biological and computer sciences. With the aim of creating an artificial adaptive system, they are designed as a high level simulation of evolution. Although it was not an objective when they were first introduced, GAs have and continue to celebrate rapidly growing interest as search algorithms. This is due to two facts: First, there is the need for new and robust optimization techniques to handle the large number of problems which defy traditional optimization techniques; and second, GAs are powerful search algorithms. Their power lies in their ability to adapt to different environments and to work effectively and efficiently in complex and time-varying search spaces where traditional techniques fail.

In reality, problems of practical interest are rarely as friendly as traditional search methods assume them to be. They represent search spaces which might be discontinuous,

nonconvex, multimodal and/or stochastic. GAs are successfully applied to such complex problems. The main premise in their employment as search algorithms is that they provide a robust search scheme that resolves the dilemma between an algorithm's generality and its power.

The deficiency of traditional methods in complex problems gave rise to the development of various new approaches which operate in the interface of operations research and artificial intelligence (Glover and Greenberg, 1989; Reeves, 1993). Together with *genetic algorithms* (Holland, 1975; Goldberg, 1989); *artificial neural networks* (McCulloch and Pitts, 1943; Rosenblatt, 1962; Tank and Hopfield, 1986), *evolutionary strategies* (ESs) (Rechenberg, 1973; Schwefel, 1975; Schwefel, 1977) and *evolutionary programming* (EP) (L.Fogel, 1962; L.Fogel, 1963; L.Fogel, Owens and Walsh, 1966) are biologically motivated approaches. GAs, ESs and EP form the three well-defined paradigms of a rapidly developing field called *evolutionary computation* (D.Fogel, 1995). The extension of GAs into the area of computer programs gave rise to the birth of a new field called *genetic programming* (Koza, 1992). *Simulated annealing* (Kirkpatrick,Gelatt and Vecchi, 1983; Cerny, 1985) is an approach based on the principles of physical sciences while *tabu search* (Glover, 1977 ; Glover, 1986; and Glover, 1989) is based on the general principles of problem solving.

1.1. Genetic Algorithms

Genetic algorithms have arisen from the works of John Holland on complex adaptive systems from 1960s onwards (Holland, 1975; Holland, 1992a). His students contributed much to the research on GAs and to their popularity as search algorithms (Goldberg, 1989a; Mitchell; 1996). Today, GAs as heuristic optimization algorithms attract many researchers from a wide range of research areas.

GAs operate on a coding of the problem and do not make use of any problem specific information other than the payoff (objective function) values of individual solutions. The

solutions are represented as finite-length strings over some finite alphabet and are denoted as *chromosomes* or *individuals* in parallel to biological systems. Chromosomes used in GA applications are simple *haploid* (single strand) strings as opposed to more complicated *polyploid* structures found in biological individuals. Chromosomes are made up of *genes* which generally encode parameters of the optimization problem. Each gene is located at a particular *locus* on the chromosome, and the values it takes are called *alleles*. The search is accomplished by a population of such individuals and the transition rules are probabilistic. The following four characteristics represent the differences separating GAs from more conventional optimization techniques (p. 7 in Goldberg, 1989a): (a) direct manipulation of codings of the parameter set, (b) search from a population of points, (c) blind search via sampling, and (d) search using stochastic operators.

Genetic algorithms emulate the *natural selection* as phrased by the "survival of the fittest" principle. Populations of individuals (chromosomes) are evolved over generations to eventually find an individual which performs satisfactorily in the given problem environment as measured by a fitness function. At each generation, individuals are chosen from the current population using a fitness-based selection scheme such that above-average individuals have a higher chance of being selected. The selected individuals undergo genetic operators such as crossover and mutation to yield offspring which totally or partially replace the old generation. The next generation is thus formed and the process continues until a termination criterion is satisfied.

The total replacement of the population by the offspring generated is called the *generational replacement*, and the corresponding GA is called a generational GA. The offspring may also replace only a portion of the original population which is called the *steady-state replacement*. In that case, the offspring replace their parents or some other members of the population selected by a certain criteria (usually based on performance) as soon as they are generated and participate in the next round of parent selection. As a compromise between these two replacement processes, a parameter called *generation gap rate, G,* is specified which gives the proportion of the parent population to be kept in tact into the offspring population. Another strategy that is employed especially when a GA is used in optimization is *elitism*

which transfers one or more of the very best performers of each generation in tact into the next generation.

Figure 1.1 below reproduces the canonical genetic algorithm from De Jong (1993). If the alphabet used in encoding is binary, the selection scheme is the so-called roulette-wheel selection, the replacement scheme is generational and the genetic operators are the reproduction, one-point crossover and bitwise mutation operators, then the resulting GA is referred to as the simple GA or SGA.

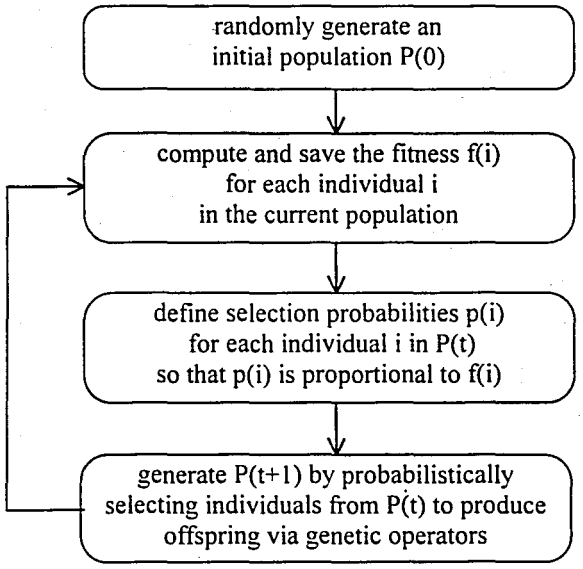


FIGURE 1.1. A canonical genetic algorithm (from De Jong,1993)

GA's effectiveness in complex search spaces is due to the exploitation of coding similarities of high performance strings. Similarities at certain string positions are described by similarity templates which are called *schemata*. Geometrically, schemata represent hyperplanes in the search space. For the binary encoding, they are defined over the binary alphabet extended with a {*} which is a "don't care" symbol matching either 0 or 1. To illustrate, the schema H=10** describes four strings of length four, which are 1000, 1010 ,

1001, and 1011. The order of a schema is the number of fixed (i.e. non-*) positions in the schema and is denoted by $o(.)$. The defining length is given by the distance between the first and the last fixed positions in the string, and is denoted by $\delta(.)$. Thus, for H=10**, $o(H)$=2 and $\delta(H)$=1.

When evaluating a population of *popsize* strings of length l, the GA is implicitly estimating the average fitnesses of all the schemata present in the population (and there are between 2^l and $popsize*2^l$ of them depending on the population diversity). This fact is named *implicit parallelism* by John Holland.

The power of the GA is believed to be due to this implicit search of schemata where

- increasingly more samples are allocated to short, low-order, above average schemata, or the so-called *building blocks* (as formalized by the *Schema Theorem*), and
- short, low-order, highly fit schemata are combined to produce fitter, higher-order schemata (as formalized by the *Hypothesis of Building Blocks*).

The implicit processing of schemata is analog to a well-known sequential decision process from statistical decision theory, namely the k-armed bandit problem. An optimal strategy for such a sequential decision process involving uncertainty must maintain a balance between exploitation of the so far accumulated knowledge and continued exploration for additional knowledge about solution characteristics. This is what the GA is thought to do implicitly. Reproduction through fitness based selection mechanisms and crossover accomplish exploitation of useful schemata. Randomized crossover and mutation together with initial random population sample the search space and accomplish exploration for better solutions.

Many extensions of and modifications to the SGA have been developed by GA researchers over the years. Alphabets of higher cardinalities, problem specific representation schemes and varying-length chromosomes are employed. As an improvement to the traditional one-point crossover, k-point and uniform crossover operators are developed. Various selection schemes like tournament selection and ranking are developed. Some researchers hybridize

GAs with local search algorithms to improve performance on specific problem instances. A huge amount of research is conducted on the application of GAs to the ordering type problems, where binary alphabet, *k*-point crossover, and bit mutation fails to produce any meaningful solutions. For that problem domain; permutation representation, many ordering crossover operators and swap mutation operators are developed.

The theory and the mechanics of how GAs work has been the subject of many research efforts. Mitchell (1996) provides some review of and references to recent discussions on that subject. Foundations of Genetic Algorithms (FOGA), Parallel Problem Solving from Nature (PPSN) and International Conference on Genetic Algorithms (ICGA) proceedings are valuable sources of most recent works.

There are two good introductory books on GAs (Goldberg, 1989a; and Mitchell, 1996). Holland (1992b) gives a very good account of the basic mechanisms of GAs with an emphasis on their evolutionary aspects. Current work of John Holland is on the mechanisms of rapid improvements in complex adaptive systems (Holland, 1995). One of the recent surveys on GAs is by Srinivas and Patnaik (1994). Booker, Goldberg and Holland (1989) review the definition, theory and application of classifier systems which are machine learning systems that incorporate GAs. There are many commercial implementations of GAs as search algorithms, and Filho, Treleaven, and Alippi (1994) give a review of such GA programming environments. The electronic list GA-Digest is both a discussion forum and a source of latest information on current research efforts and events (http:/www.aic.nrl.navy.mil).

1.2. Combinatorial Optimization

Combinatorial optimization is the mathematical study dealing with the "arrangement, grouping, ordering, or selection of discrete objects, usually finite in number" (Lawler, 1976). In the last decades, developing heuristics for combinatorial problems has been a very fast

growing area of research. The reason that heuristics play an important role in this field is simple: Many (practically important) problems for which efficient exact algorithms neither exist nor is expected to be developed in a reasonable future are of a combinatorial type.

Three pure types of combinatorial problems can be listed: *Assignment* problems, *sequencing* problems, and *selection* problems (Mueller-Merbach, 1981). In assignment problems, there exist two (or more) sets of discrete objects where pairs (or higher dimensional n-tuples) of single elements of each set have to be formed. Examples of two dimensional assignment problems are the linear assignment problem, the transportation problem, and the quadratic assignment problem.

Sequencing problems refer to only one set of discrete objects which have to be ordered in a sequence. The most popular sequencing problem is the travelling salesman problem. Other examples include the Chinese postman problem and the shortest path problem.

In selection problems, one set of discrete objects is given from which a subset has to be chosen. There are many examples for selection problems including the spanning tree problem, the edge (also node and set) covering problem, the node and set packing problems and the knapsack problem.

Most real world combinatorial problems include components from these three types at the same time; i.e. they are what will be called here multi-component combinatorial optimization problems (MCCOP). In this work, multi-component combinatorial optimization problems which involve one sequencing component and one or more selection components (MCCOP_SS) will be addressed. This is a wide class of problems and examples are numerous: As the name implies, any combinatorial problem with a set of objects to be sequenced and one or more sets of alternatives for each object is in this class. For example most of the machine scheduling problems are here.

1.3. Overview of the Dissertation

This dissertation presents a new uniform order-based crossover operator for MCCOP_SS. Some of the genetic algorithm approaches to MCCOP_SS have represented the problem as a one dimensional sequencing problem, others accounted for all the dimensions of the problem within the representation. In the first case, one of the several available order-based crossover operators is employed for the sequencing dimension and the rest of the search is accomplished by local hillclimbers. In the latter case, either several crossover operators (one for each dimension) are employed or a problem specific crossover operator is designed for each instance of MCCOP_SS. The operator presented here supplies the advantages of both of these approaches. It encompasses all the dimensions of the problem in one operator and it is a general tool applicable to any problem which can be cast in the form of a MCCOP_SS.

Chapter 2 formally defines MCCOP_SS and presents and discusses the new operator. It also presents the GA used in the applications. In Chapter 3 to Chapter 5, the effectiveness of the new operator is illustrated on three difficult combinatorial optimization problems with sequencing and selection components. Chapter 6 concludes.

2. THE NEW OPERATOR MCUOX AND THE GENERIC GENETIC ALGORITHM WITH MCUOX, GA-MCUOX

In this chapter, the generic multi-component combinatorial optimization problem with sequencing and selection components will be formally defined. The new crossover operator MCUOX will be presented along with all the other components of the genetic algorithm that will be used in the numerical study. The genetic algorithm with the new crossover operator MCUOX will hereforth be referred to as GA-MCUOX.

2.1. The Generic Multi-Component Combinatorial Optimization Problem with Sequencing and Selection Components

The generic multi-component combinatorial optimization problem with sequencing and selection components (MCCOP_SS) is defined as follows: Assume that there are N objects (e.g. N jobs in a flowshop problem) to be sequenced. There may or may not exist constraints regarding the sequencing of all or some of the objects. Examples for such constraints are precedence relations among jobs and/or among operations of a job.

Associated with each object there is one or more selection problems. The selection problem is such that from among a set of alternatives (e.g. several means of manufacturing or transportation, different rates of accomplishment, etc.), one alternative has to be selected. The selection problem (i.e. the set of alternatives) may be the same for all objects or be particular to each object. There can be more than one set of alternatives. The set(s) may have arbitrarily large cardinality.

The emphasis in this dissertation is on the instances of MCCOP_SS in machine and project scheduling; but as already noted in Chapter 1, MCCOP_SS encompasses many

problems from a variety of fields including scheduling, circuit design, line balancing and vehicle routing with capacity constraints.

One example instance of MCCOP-SS which will be discussed in detail in the following chapters is the resource constrained project scheduling problem. In this problem, a set of activities each with a set of modes of accomplishment is given. The problem is to schedule the activities such that a certain performance criterion is optimized. Here, the sequencing problem is to sequence the activities in the order they will be considered for scheduling; and the selection problem associated with each activity is to choose one mode from among the alternative modes of accomplishment for that activity.

Likewise, in parallel machine scheduling problems which will be discussed in detail in Chapter 5 below, the sequencing problem is to sequence a set of jobs in the order they will be considered for scheduling on a set of parallel machines. The selection problem associated with the jobs in this case is to choose one of the machines for their processing.

Traditionally, the GA approaches to MCCOP-SS instances have treated the individual components of sequencing and selection separately. The main reason for this has been the availability of various genetic operators for the corresponding *ordering* and *value* information separately and the difficulties involved in considering them concurrently. Recently, approaches to such problems try to optimize these two dimensions together. One interesting application is that of Fennel. Underbrink and Williams (1994) where the researchers used GAs to develop schedules at Boeing for different domains including assembly of military and commercial aircraft, weapon systems and space vehicles. The problem attacked is a scheduling and resource allocation problem which is represented on a two-chromosome scheme one giving the job sequence and the other the selected resource for the corresponding jobs.

Another real world application is provided by Cox *et al.* (1996) who employed GAs at COX California PCS to design cost-effective networks for carrying personal communication services traffic. The problem is represented on a two-part array (chromosome), the first part being a permutation of nodes in the network and the second a binary vector that indicates whether the corresponding node is to be a hub node or not.

Drechsler *et al.* (1996a) employ GAs for the construction of small and highly testable OKFDD-Circuits. OKFDD is the short name used for *Ordered Kronecker Functional Decision Diagram.* In the problem they attack, the aim is to find an ordering of a set of Boolean variables that are associated with a set of nodes and to select for each variable one of three possible decomposition types.

The problems attacked in these example applications are all instances of MCCOP-SS. More detail about the operators that these approaches employ is provided below in Section 2.4.4.

2.2. Chromosome Representation

The choice of a representation scheme is the foremost decision a GA researcher has to take. This choice is not simple since it is intermingled with both the choice of how much information is to be represented on the chromosomes and the choice of genetic operators. Appropriate choice and/or design of a representation scheme and of genetic operators is the most important step towards success.

A representation scheme which is a high level abstraction of the problem is referred to as an *indirect representation* scheme; whereas one which incorporates all facets of the problem is called a *direct representation* scheme. In the context of job shop scheduling for example, an indirect representation scheme gives rise to chromosomes that incorporate only the sequence of jobs. Thus the GA addresses only a small portion of the search space. The advantage of using an indirect scheme is obvious: there is a wealth of genetic operators available for use in sequencing problems. The price paid is that one needs to develop and/or employ routines which perform the rest of the search. Usually such routines perform a local search on the sequence specified on the chromosome and this may hinder the discovery of global optimum solutions. Moreover, incorporation of such routines introduces additional

noise. A good example for the use of indirect schemes is the work of Holsapple *et al.* (1993) who develop a hybrid scheduling algorithm that generates static schedules in flexible manufacturing contexts. The job sequences generated by the GA are fed into a filter beam search scheme which generates associated schedules.

In a direct representation scheme, a chromosome represents all the information relevant to the problem. This means that the search is accomplished entirely by the GA, and the chances of finding global optima are bigger. But more complex genetic operators and/or repair functions for validating illegal offspring are needed. In the context of scheduling, within a direct representation scheme, the schedule itself becomes the chromosome. Bruns (1993) gives a good example of a GA application with a direct representation in a general production scheduling environment. In his representation, a chromosome contains the temporal assignments of all operations of orders to the machines to be used.

There are many other schemes falling in between these two extremes of strictly direct and indirect representation schemes. The work of Fang, *et al.* (1993), for example, involves a chromosome representing all the information relevant to the scheduling problem but the explicit time settings. A simple schedule builder generates the schedule according to the ordering of genes on the chromosome. Bruns (1993) provides a list of and examples for several of such intermediary schemes used in the context of scheduling.

The expectation, in general, is that the more of the characteristics of the solution are represented on the chromosome, the larger is the proportion of the search space addressed by the GA and the larger the chance to locate the global optimum solutions becomes. Empirical work has, indeed, shown the superiority of direct schemes over indirect ones. Uckun, *et al.* (1993) study examples of these schemes in their schedule optimizer prototype VSOP in the context of job shop scheduling. Their conclusion is that the more problem specific information is included in the representation, the better is the performance.

All the above mentioned schemes are set up at the start of the runs and do not change from then on. There are some other interesting approaches regarding the representation schemes. Lee, *et al.* (1993), for example, use an adaptive representation scheme for solving

two related problems, that of lot sizing and sequencing, concurrently. The chromosome gives both lot sizes and sequences, and itself undergoes an evolution. The so called messy GAs (mGAs) have another interesting representation scheme (Goldberg, Korb, and Deb, 1989; Goldberg *et al.*, 1993): Chromosomes are variable-length strings which may be over- or underspecified with respect to the problem being solved.

Whichever representation scheme is used, the majority of GA applications employ a haploid chromosome structure where the chromosome is a one-dimensional array of genes. Each locus of the chromosome accommodates one gene, and each gene encodes a particular trait of the solution (e.g. a parameter related to a parameter optimization problem). The diploid chromosome structures (with two strands of genes) and the dominance relationships (to decide on the phenotypic outcome) are often used where the environment is non-stationary. Ng and Wong (1993) report a recent study on diploid structures.

The chromosome representation employed for MCCOP_SS in this work is more complex than the simple haploid chromosome structure with one trait per locus. Here, each locus accommodates more than one trait of the solution: an object and the selections associated with it. So, there are more than one gene at each locus. If there is one selection problem associated with each object, then there are two genes per locus: one encoding the object, the other the selection made for that object. Additional genes are to be employed to encode selections associated with additional selection problems. For the sake of simplicity, in the chromosomal structure employed in this work, the contents of each locus will be treated as a single gene. This is practical as the contents of each locus collectively form a body of information for that particular locus and do not have connections to other loci. And it is also a popular approach followed by many researchers.

Thus the chromosome representation employed for MCCOP-SS can be described as follows: There are N genes; one per object. Each gene consists of the identification of the object it is associated with plus the alternative selected for each selection problem associated with that object.

This representation scheme is more compact as opposed to schemes which represent the two dimensions on two separate chromosomes like the ones employed in the applications discussed in Section 2.1.

The scheme is a direct scheme for the generic MCCOP_SS since all facets of the problem are represented on the chromosomes. It is also in line with the principles suggested by Goldberg (1989a, p.80) for good encodings in that it uses the smallest alphabet which permits a natural expression of the problem. For the sequencing problem, the alphabet used is $\{1,..,N\}$, and the maximal alphabet used for the selection components is $\{1,.., \max_j S_j\}$ where S_j is the number of alternatives in the selection problem associated with object j.

As a small example for the chromosomal representation, consider a problem with 5 objects to be sequenced. Let them be identified by integer numbers from 1 to 5. No constraints are given regarding the ordering of the objects. Assume that there is one selection problem associated with each object, where only one alternative is to be chosen from the respective alternative sets. For the first, third and fourth objects, the alternative sets have two elements each. For the second and fifth objects, there are three and four alternatives respectively. Figure 2.1 below outlines the problem and illustrates one possible chromosome for this problem.

The example chromosome [3-2 1-2 4-1 2-3 5-2] indicates that the objects are to be sequenced in the order 3, 1, 4, 2, and 5. The first gene consists of the object-selection pair 3-2. It indicates that for object 3, the second alternative from its set of alternatives is selected. Similarly, for objects 1 and 5, the second alternatives from their respective sets are chosen, while for object 4 and 2 the selection resulted in the first and third alternatives of the respective sets.

objects	1	2	3	4	5
set of alternatives I	1 2	1 2 3	1 2	1 2	1 2 3 4
⋮	⋮	⋮	⋮	⋮	⋮

a) Selection problems associated with each object; for each object, one alternative is to be selected from the given sets

locus	1	2	3	4	5
object sequence	3	1	4	2	5
alternative selection	2	2	1	3	2

b) An example chromosome sequencing object-selection pairs

[3-2 1-2 4-1 2-3 5-2]

c) The internal representation of the example chromosome

FIGURE 2.1 A small example for chromosome representation with N=5

2.3. Schemata

The schemata relevant to the selection component of MCCOP_SS are *allelic schemata* (or *a*-schemata as referred to by Goldberg) which are introduced by Holland (1975) and which define all possible allele (value) similarities. For ordering problems Goldberg introduced another type of schemata called *ordering schemata* (or *o*-schemata) which define all similarities in the ordering of values (Goldberg and Lingle, 1985; Goldberg, 1989a). A position on an *o*-schema is either fixed or is marked with an exclamation point (!). The exclamation points on the schema indicate any of the possible orderings over the remaining unmentioned values of objects. For example, the *o*-schema H=! ! 3 5 ! with order $o(H)$=2 and defining length $\delta(H)$=1 defines the subset of all orderings that have object 3 and 5 in the third and fourth positions on the chromosome respectively. This schema describes the subset of (N-

$o(H)$)!=(5-2)!=3! orderings of the remaining symbols {1,2,4}. Goldberg (1989a) defined various extensions of o-schemata and noted that the schema theorem operates on both a-schemata, o-schemata and their combinations. This is an important footing for MCCOP_SS defined in this work, since its representation incorporates both types of schemata.

2.4. The New Operator: Multi-Component Uniform Order-Based Crossover - MCUOX

The new crossover operator is designed in response to the need for a general and powerful operator to be used in GA applications to multi-component combinatorial optimization problems with sequencing and selection components (MCCOP_SS). While there are several crossover operators for pure sequencing problems and a wealth of crossover operators for parameter selection problems, there is no crossover operator for problems involving both dimensions.

This lack of availability has lead many GA researchers to treat MCCOP_SS instances as pure sequencing problems to take advantage of the availability of various sequencing crossover operators. In doing this, they partitioned the search between GAs and some other routines. Researchers who wanted to address all the dimensions of the problem within the GA approach resorted to various means such as applying a different crossover operator for each dimension or designing a problem specific crossover operator for each application.

MCUOX fills a gap in GA research by providing a general and yet powerful operator applicable to all the instances of MCCOP_SS which itself encompasses many real world problems.

It is also in line with the quest for robustness in genetic search by being able to efficiently apply to a breadth of problems. Robustness is usually sacrificed to obtain peak performance on a given problem. But as the founders of the GAs state ".. Nature is unconcerned with achieving peak performance on any given problem. Natural genetics has

evolved so the same search procedures could function under many different kinds of environmental conditions. This breadth combined with relative -if not peak- efficiency defines the primary theme of genetic search: robustness" (Goldberg, 1989b). This is because ".. there is a wide range of problems .. where fine tuning will only yield a very local, temporary improvement, requiring a whole new fine tuning each time the problem evolves. [In such complex problems] a robust algorithm is of great help" (Holland, 1997).

2.4.1. How Does MCUOX Operate?

The way MCUOX operates is as follows. MCUOX constructs one offspring from two parent chromosomes. The construction proceeds gene by gene and is mainly based on the sequencing component of the problem.

Starting from the first genes on the parents and on the child, iteratively, one of the parents is chosen randomly and its next unconsidered object becomes the next object on the child. If the selected alternative for that object is the same on both parents, then that selection is also made on the child; if not, one of the selections of the parents is chosen randomly.

If there are more than one selection component in the problem, the last part is repeated once for each such component. Figure 2.2 provides a more formal account of the algorithm and Figure 2.3 presents an example application with N=4.

As already noted, MCUOX generates one offspring from two parents. But it is perfectly possible to generate two or more offspring using the same two non-identical parents. The choice for one offspring as opposed to the traditional number of two is based on the fact that this way twice as many offspring of different origins can be generated. This in turn provides a more through mixing and juxtaposition of building blocks present in the parent population.

START

- With the first position on the parents and the child.

LOOP

- Choose one of the parents randomly.
- Find the first object on the chosen parent which is not assigned to the child yet. That object becomes the next object on the child.
- If the selections made for that object are the same on both parents, then make this same selection on the child too. If not, choose one of the selections of the parents randomly and record it on the child.
- Repeat the last part for each selection problem involved.
- If all the genes of the child chromosome are completed, STOP; else proceed to the next gene on the child and repeat LOOP.

FIGURE 2.2. Algorithm of MCUOX

iteration	result of parent selection		child
0			C: [- - - -]
1	P1	P2	C: [3-1 - - -]
2	P2	no need	C: [3-1 1-2 - -]
3	P2	P1	C: [3-1 1-2 2-3 -]
4	P1	P1	C: [3-1 1-2 2-3 4-1]

FIGURE 2.3. MCUOX: Example application

Parent 1 (P1) is [3-2 1-2 4-1 2-3] and parent 2 (P2) is [1-2 3-1 2-1 4-2]

2.4.2. How Does MCUOX Process Schemata?

Crossover operator is an indispensable operator for a GA provided that it is capable of putting together the partial solutions present in the population. More formally, crossover operator must be able to recombine short, low-order, highly fit schemata, the so-called

building blocks, into larger and more fit ones. How does MCUOX process schemata, in particular building blocks?

Schema survival under any crossover operator, including MCUOX, improves as the order and/or the defining length of the schema decrease. Thus a schema like H_1=*AB*..* is more likely to survive crossover than a schema like H_2=*A*..*B*C, where A,B, and C correspond to fixed values. The following formulae for the survival probability under MCUOX for order-1 and order-2 schemata illustrate this point:

$$P_S(A^*..^*) = \begin{cases} 1.00 & \text{P2} \in A^*..^* \\ 0.50 & \text{P2} \in B^*..^* \end{cases}, \tag{2.1}$$

$$P_S(AB^*..^*) = \begin{cases} 1.00 & \text{P2} \in AB^*..^* \\ 0.50 & \text{P2} \in AC^*..^* \text{ or } BC^*..^* \\ 0.25 & \text{P2} \in CA^*..^*,\ CB^*..^* \text{ or } CD^*..^* \end{cases}, \tag{2.2}$$

where $P_S(H)$ denotes the survival probability of H under MCUOX, and P2 stands for 'parent 2' and denotes the string mated with a string that is an instance of the schema H under consideration. In this case, H=A*..*.

In case of MCUOX, the survival also improves as the fixed positions are located towards the ends of the chromosome. That is, H_3=A*..* is more likely to survive MCUOX than H_4=*..*A*..*, and similarly, H_5=*..*A has a bigger chance to survive than H_6=*..*A*..*. For example, if the defining value in the order-1 schema A*..* is located in position 2 rather than in position 1, the following survival probability distribution results:

$$P_S(*A*..*) = \begin{cases} 1.00 & P2 \in *A*..* \text{ and P2 has the same allele at locus 1} \\ 0.50 & P2 \in *A*..* \text{ and P2 has a different allele at locus 1} \\ 0.50 & P2 \in *B*..* \text{ and P2 has the same allele at locus 1} \\ 0.25 & P2 \in *B*..* \text{ and P2 has a different allele at locus 1} \end{cases} \quad (2.3)$$

This means that in addition to propagating short, low order schemata better than the longer ones, MCUOX propagates schemata with defined positions located towards the ends of the undefined region better than others. Thus at the start of a run, schemata with the first few positions defined are processed, propagated and recombined at a higher rate than others. And once the relatively more fit ones of these schemata, i.e. the building blocks, start to take important portions of the population according to the survival of the fittest principle through the fitness-based selection mechanism, schemata with the next few positions defined will come to the stage; and this process will continue a couple of positions at a time until the middle parts are also fixed and the population has largely converged. This means that a *double-sided convergence of genes* towards the middle parts of the schemata is expected to occur with a GA employing MCUOX. Before illustrating this aspect with an example, one point needs to be emphasized.

The property of double-sided convergence of genes is especially important for problems where the ordering implicitly or explicitly corresponds to a priority list of objects (such as jobs, activities, etc.) competing for the usage of a common resource (such as time, money, machine center, etc.). The usage of the resources starts with the first objects in the ordering and goes on towards the end of it. The fixation of the first couple of positions at the start of the sequence determines the room left for the remaining objects and hence the quality of the resulting solution. Thus in such problems, schema processing under MCUOX is in accordance with the nature of the problem.

To illustrate this phenomenon of double-sided schema growth under MCUOX, a small experiment with N=10 is carried out, where a small instance of a resource constrained project

scheduling problem with discounted cash flows (RCPSPDCF) is subjected to GA-MCUOX and the development of schemata is kept track of.

RCPSPDCF is defined in detail in Chapter 4 below. In the problem treated here, a project with 10 activities is given. Associated with each activity there is a set of operating modes which give the duration of the activity as a discrete, decreasing function in the level of usage of a specific renewable resource. There is a unit cost of resource usage and the total cost incurred for a specific activity is assumed to occur at its start time. A lump sum payment is obtained at the completion of the project. The aim is to sequence activities subject to some precedence constraints, and for each activity, to select one of its possible modes so that the net present value (NPV) of the resulting schedule is maximized. Precedence diagram and input data are given in Figure 2.4 and Table 2.1 respectively. Cost per unit of the resource used is 3, the availability of the resource is 2 units per period, the lump sum payment is taken to be 500, project due date is set at 100, and the discount rate is 0.005.

GA-MCUOX is run on the example problem with a small population of 25 chromosomes for 100 generations. Table 2.2 gives the frequencies of observed genes at several generations and illustrates the convergence of best chromosomes. For each case, the dominant schema in the corresponding generation is also outlined along with the best chromosome of that generation and its objective function value. The question mark (?) is used to define gene similarities (of activity-mode selections) in the schemata. To save space, details of gene frequencies for generations 50 and 100 are left out.

At the start of the run, in generation 0, there isn't any dominant schema as is to be expected and desired. The population is well mixed with many building blocks present; i.e. for each loci almost all values for each gene are represented that are possible given the precedence relations in Figure 2.4. For example, all of the four possible order-one schemata defining the first position are present; and although 1-1,?,..,? and 2-1,?,..,? occupy a larger portion of the population as opposed to 1-2,?,..,? and 2-2,?,..,?, there is no clear winner. Yet already at generation 6, two of these four schemata are ruled out, and from among the remaining two,

2-1,?,..,? presents itself as the dominant schema defining the first position as it is being represented by the 21/24'th of the population of chromosomes.

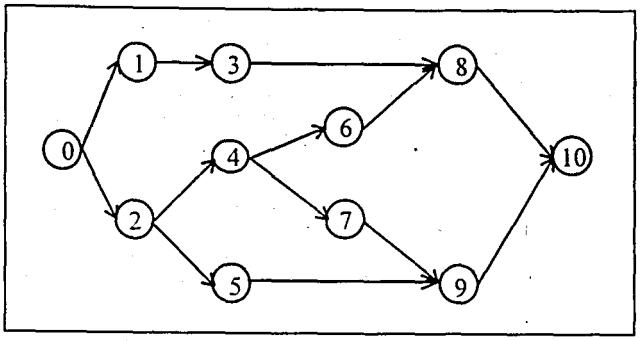


FIGURE 2.4. Project network for the example problem with N=10 with activity-on-node representation

TABLE 2.1. Input data for the RCPSPDCF example problem

Activity	Resource Requirement	Duration
1	2 1	2 5
2	3 3	5 6
3	3 1	8 13
4	3 1	7 12
5	2 1	6 10
6	2 1	4 9
7	2 1	3 7
8	2 1	3 6
9	3 1	1 3
10	2 1	3 4

TABLE 2.2. Convergence of genes in the example run of GA-MCUOX

Generation	Locus	Observed genes (and their frequencies respectively)
0	1	1-1, 1-2, 2-1, 2-2 (9), (2), (10), (4)
	2	1-1, 1-2, 2-1, 2-2, 3-1, 3-2, 4-1, 4-2, 5-1, 5-2 (1), (3), (3), (2), (5), (1), (4), (3), (1), (2)
	9	8-1, 8-2, 9-1, 9-2 (4), (8), (10), (3)
	10	10-1, 10-2 (12), (13)
=> Dominant schema of generation 0: ?, ?,.., ?, ? Best chromosome in generation 0: [2-1, 1-1, 3-2, 5-2, 4-2, 7-2, 6-1, 9-2, 8-2, 10-1] with z=261.25		
6	1	2-1, 2-2 (21), (4)
	2	1-1, 1-2, 4-1, 4-2 (6), (10), (4), (5)
	9	8-1, 9-1, 9-2 (4), (14), (7)
	10	10-1, 10-2 (8), (17)
=> Dominant schema of generation 6: 2-1, ?, .., ?, 9-1, 10-2 (8 instances) Best chromosome in generation 6: [2-1, 1-1, 4-2, 5-2, 3-2, 6-1, 8-2, 7-1, 9-1, 10-2] with z=272.09		

TABLE 2.2. (Convergence of genes in the example run of GA-MCUOX) continued.

Generation	Locus	Observed genes (and their frequencies respectively)
12	1	2-1, 2-2 (24), (1)
	2	1-1, 1-2, 4-2 (16), (5), (4)
	9	8-2, 9-1, 9-2 (1), (23), (1)
	10	10-1, 10-2 (9), (16)
=> Dominant schema of generation 12: 2-1, 1-1, ?,.., ?, 9-1, 10-2 (9 instances) Best chromosome of generation 12: [2-1, 4-2, 1-1, 5-2, 3-2, 6-1, 7-1, 8-2, 9-2, 10-2] with z=274.11		
Dominant schema of generation 50: 2-1, 1-1, 4-2, ?,.., ?, 6-1, 8-1, 9-2, 10-2 (15 instances) Best chromosome of generation 50: [2-1, 1-1, 4-2, 5-2, 3-2, 7-1, 6-1, 8-1, 9-2, 10-2] with z=274.27		
Dominant schema of generation 100: 2-1, 1-1, 4-2, 5-2, 3-2, 7-1, 6-1, 8-1, 9-2, 10-2 (13 instances) Best chromosome of generation 100: [2-1, 1-1, 4-2, 5-2, 3-2, 7-1, 6-1, 8-1, 9-2, 10-2] with z=274.27		

The dominant schema in generation 6 with the largest number of converged genes is 2-1, ?, .., ?, 9-1, 10-2, which has eight instances in the population. The schema with the next highest number of instances is schema 2-1, ?, .., 9-1, 10-1 with four instances. The schema 2-1, ?, .., ?, 9-1, 10-2 is the combination of the winners of the competition among order-1

schemata defining the first, ninth and tenth positions respectively. This illustrates that MCUOX operates in line with the *Hypothesis of Building Blocks* and is able to combine building blocks effectively to produce fitter, longer schemata.

The run detailed in Table 2.2 has ended up with a schedule with z=NPV=274.27. A chromosome corresponding to this schedule first appears in generation 18 and has the form [2-1, 1-1, 4-2, 3-2, 5-2, 6-1, 7-1, 8-1, 9-2, 10-2]. An alternative solution appears at generation 30 which is also the solution reported in Table 2.2, namely [2-1, 1-1, 4-2, 3-2, 5-2, 7-1, 6-1, 8-1, 9-2, 10-2].

It may be conjectured that the double-sided convergence is actually a result of precedence constraints rather than a property of MCUOX. In other words, the fixation of positions necessarily proceeds from both ends towards the middle parts given the precedence relations for two reasons: First, the precedence relations dictate a smaller number of alternative values for the two ends and thus the population more quickly settles down in one of those few alternative values, and second, as those ends are fixed, the candidate sets for inner positions get smaller increasing the convergence rate.

The precedence relations indeed catalyze the double-sided growth of schemata in this example. But the above conjecture is not true in general for the following reasons: (1) Convergence is not a once-and-for-all process: Until generation 16, the schema ?,.., ?, 9-1, ? has been the dominant schema containing also the best so far chromosome until generation 12. At generation 16, the schema ?,.., ?, 9-2, ? takes over and becomes the dominant schema until the end of the run. Thus there is something more than the pressure of precedence relations working here. (2) The same phenomenon of double-sided convergence is observed by the author of this work both in other scheduling applications discussed in the next chapters and in flowshop scheduling applications where there is no precedence relations whatsoever.

The claim discussed is important though as it points to another area where the double-sided convergence of genes under MCUOX may be very useful, namely in problems involving precedence relations. MCUOX operates in line with the problem nature and this enhances the convergence to near-optima.

To verify this reasoning, the same GA is run on the same problem both with another crossover operator and with no crossover operator at all. The other crossover operator is the partially mapped crossover (PMX) developed by Goldberg and Lingle (1985) which is one of the most popular sequencing operators. PMX produces two offspring from two parent chromosomes. Parent strings are aligned and two crossing sites are chosen uniformly at random along the strings. The so-called *matching section* inbetween these crossing sites is used to accomplish positionwise exchanges on each parent chromosome separately. The following small example illustrates the process better (p.171 in Goldberg, 1989a). Parents are given as follows

P1= 9 8 4 | 5 6 7 | 1 3 2 10

P2= 8 7 1 | 2 3 10 | 9 5 4 6

with crossing sites chosen after the third and the sixth loci. The matching section used as a mapping between the two strings gives rise to exchanges of places of objects 5 and 2, of objects 6 and 3 , and of objects 7 and 10 on the parent strings separately. The offspring thus produced are

C1= 9 8 4 | 2 3 10 | 1 6 5 7

C2= 8 10 1 | 5 6 7 | 9 2 4 3.

Following the suggestion of Goldberg and Lingle, the operator is applied to the MCCOP_SS instance under investigation by tagging on the selection components to the sequenced activities and thus treating an activity-mode selection pair as one unit. One difficulty encountered is that the application of PMX causes precedence violations. A repair algorithm is developed which, starting from the first activity, checks whether there are any predecessors of the current activity in the rest of the sequence. If there are one or more such predecessors, the activity is leapfrogged right after the last one in the sequence. The process is repeated with the next activity in the sequence until the end of the sequence is reached.

GA-MCUOX, GA-PMX and the mutation-only-GA are replicated three times on the example RCPSPDCF problem. Table 2.3 presents the average of the best solution values of the three replications and their standard deviations. As is evident from this small experiment,

recombination through crossover plays an important role in identifying near-optimal solutions. The extra power of MCUOX, which it gains by operating in accordance with the problem nature, is reflected in its much better performance as compared to PMX.

A question remains as to whether the solution found by GA-MCUOX with z=274.27 is really a good one. To answer this question, GA-MCUOX is run on this same example problem 10 times with a population size of 250 for 750 generations. The population size is sufficiently large for an example problem with N=10, and it can safely be expected that the solution found by this GA is a near-optimal one. The average of the best solution values of these 10 replications is 274.27 and the standard deviation is 0.00.

TABLE 2.3. Comparison of the effectiveness of GA-MCUOX with that of GA-PMX and mutation-only-GA on the 10 activity RCPSPDCF example

	GA-MCUOX	GA-PMX	mutation-only-GA
average	274.27	272.52	266.77
st.Dev.	0.00	1.75	6.99

Owing mostly to the large population size and also to mutation and crossover operators, the final populations of these runs exhibit a rich diversity. Population worst values range from 212.92 to 231.95 in these 10 runs. A similar observation regarding the diversity is made within the final populations of the previous smaller runs with the population size of 25. All these suggest that GA-MCUOX is able to maintain enough diversity throughout the run which is important to avoid premature convergence.

2.4.3. Properties of MCUOX

The following list of properties summarizes the points made in the discussions so far and states a few more.

Property 1: MCUOX considers ordering and value concurrently.

Property 2: MCUOX is a general and powerful tool accomplishing search in the ordering and value dimensions separately.

Property 3: MCUOX operates within a natural and direct representation for the generic MCCOP-SS.

Many sequencing problems involve precedence constraints implicitly or explicitly stated in the problem definition. Depending on the nature of the problem, the constraints, and the crossover operator, they may be handled in several ways within the GA ranging from a simple check in the evaluation routine to complex repair algorithms. One of the most important characteristics of MCUOX is that it never violates precedence constraints. The next property states this important characteristic.

Property 4: MCUOX never violates precedence constraints. Given that the parents are precedence feasible, so will be the child.

This claim can easily be proven as follows.

Proof: The first object on the child is the first object from either parent. Given that the parents are precedence feasible, this first object is one without any predecessors. At any iteration of the algorithm, the next object on the child becomes the first yet unconsidered object on the randomly chosen parent. This object has all its predecessors already assigned to the child since otherwise the chosen parent would be infeasible. □

Another important characteristic of MCUOX is that additional selection problems are easily processed by MCUOX at virtually no cost. In approaches where separate operators are utilized for each dimension of the problem, each additional component gives rise to an additional set of operators. In case of MCUOX, only the if statement in the main loop of the algorithm (see Figure 2.2) is repeated for each such selection problem.

Property 5: MCUOX is able to effectively and efficiently deal with additional selection components.

For example, a second (a third and any other higher number of selection components) can be very easily added on the example problem of Figure 2.1 by appending the new selected alternative to the one for the first selection for each object. The chromosome then becomes [3-2,1 1-2,1 4-1,1 2-3,1 5-2,1], if for all objects the first alternatives in their second selection problems are selected respectively.

2.4.4. Convenience Provided by MCUOX

So far the effectiveness of MCUOX has been emphasized. In this section, the advantage of using MCUOX will be illustrated via example GA applications.

The first example GA is developed by Drechsler *et al.* (1996a) for the construction of small and highly testable OKFDD-Circuits. OKFDD is the short name used for *Ordered Kronecker Functional Decision Diagram* which is an ordered graph-based representation of a Boolean function over a set of n Boolean variables where each variable is associated with one of a set of n nodes, and one of three specific decompositions is to be carried out in each node. The aim is to find an ordering of the Boolean variables and to select for each variable one of the three possible decomposition types so that the size of the resulting circuit is minimized and its testability is maximized. The problem attacked is thus an instance of MCCOP_SS with an ordering dimension and one selection component.

The researchers represent the chromosomes in two arrays; one being for the sequence of Boolean variables, the other for the selections. Separate crossover and mutation operators are applied to these two arrays. These operators are presented in Table 2.4. The operator MUT , is the name given by the researchers to a special exchange operator, and 2*(operator) denotes that the same (operator) is applied twice to the selected array.

As is illustrated in Table 2.4, altogether four crossover operators are utilized to search the two dimensions of the problem. MCUOX is able to do the job of these four operators by itself and hence provides a more compact and convenient alternative.

Drechsler *et al.* test their GA on some benchmark problems from literature and obtain up to 60 per cent improvement as compared to the previously obtained best results. It is hard to claim ad hoc that a GA with MCUOX would be able to obtain similar or better results, but one thing is clear: ".. computation times can drop down significantly with MCUOX since it will replace a bunch of crossover operators that are used separately for each of the two arrays." (Drechsler, 1996b).

TABLE 2.4. Crossover and mutation operators used in the circuit design example

Ordering Operators	Selection Operators
PMX	one-point crossover 2 * one-point-crossover uniform crossover
MUT	bit mutation
2 * MUT	2 * bit mutation
neighbour exchange	neighbour exchange

Another application is provided by Fennel, Underbrink and Williams (1994) who employ GAs to develop schedules for final assembly of military and commercial aircraft, weapons systems and space vehicles at Boeing. The problem consists of finding a sequence of

jobs and selecting a resource for each job in the sequence. The genetic representation is distributed onto two chromosomes one for each dimension. For the sequencing component, which is represented as a permutation, the popular PMX and swap mutation operators are utilized, while for the resource-allocation component, which is represented as a binary array, uniform crossover (UX) and bit mutation operators are employed. UX simply chooses for each position of the offspring chromosome the value recorded in the respective position on one of the parent chromosomes. In this application, the precedence constraints are dealt with explicitly within the schedule builder, and repair algorithms are developed to handle violations of resource and temporal constraints.

The work of Cox *et al.* (1997) is on another important practical problem. They develop a GA to design networks for carrying personal communications services (PCS) traffic. Again the chromosome has two parts, the first one representing a permutation of nodes in the network and the second one denoting whether the corresponding node is a hub node or not. Order-based crossover and scramble sublist mutation operators of Davis (1991) are utilized for the permutations, and uniform crossover and bit mutation operators are utilized for the binary array. Order-based crossover takes a randomly chosen subset of the ordering from one parent and fills in the gaps by ordering the missing nodes in the order they appear on the other parent.

All these applications attack instances of MCCOP-SS. To effectively search the individual dimensions, they employ separate crossover operators. MCUOX, on the other hand, is able to recombine two (or more if there are more than one selection problems) dimensions concurrently and provides an important advantage especially in large problems.

2.5. Selection and Mutation Operators

The selection scheme employed in GA-MCUOX is the well-known *proportionate selection* scheme, also called the *roulette wheel selection* scheme (Goldberg and Segrest, 1987;

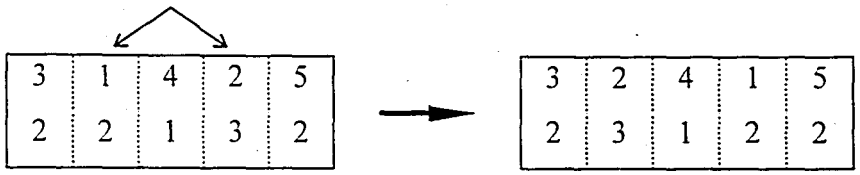
Goldberg, 1989). There are various other selection techniques a review of which can be found in Goldberg (1989). Roulette wheel selection scheme has been chosen here due to its popularity and simplicity. Shortly, in this selection scheme, each individual in the population is thought to occupy a slot on a hypothetical roulette wheel in proportion to its relative fitness. This relative fitness determines also the individual's selection probability. Each time an individual is to be selected, a random number is generated and probabilities of individuals are summed up beginning from a fixed individual until the sum just equals or exceeds the random number generated. The individual last added is the one which is selected to reproduce.

Reproduction and recombination processes can be designed to be two separate phases or in a combined fashion. In the former case, individuals are selected in the reproduction phase and exact copies are stored in a temporary mating pool. In the following recombination phase, crossover and mutation are applied to randomly selected pairs of individuals from this mating pool. In the latter case, these two phases are combined into one by applying recombination operators instantly to each pair selected. This eliminates the need for a temporary mating pool. Both of these approaches are employed in this work. The application of the former approach is illustrated in Chapter 3 and of the latter combined approach in Chapters 4 and 5. In all the applications, the generational replacement strategy is employed; i.e. the offspring replace the totality of the parent generation. Also the elitist strategy is used to ensure the protection of the best individuals encountered in the search.

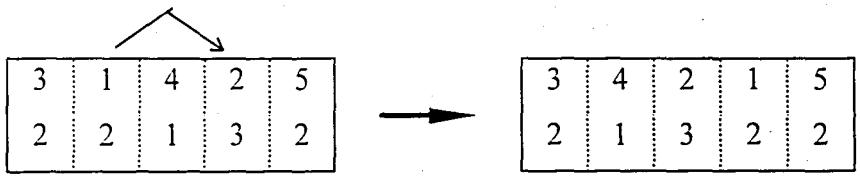
The primary function of mutation operators in the GA is that they ensure that each and every corner of the search space is reachable. To ensure this for MCCOP_SS, two types of mutation operators are needed, one for the sequencing problem and the other for the selection problem(s).

The most widely used mutation operator for sequencing problems is the *swap* mutation. This operator randomly chooses two positions on the chromosome and changes their contents. Another mutation operator for sequences is the *reposition* mutation which randomly chooses one locus and repositions its content into another randomly chosen locus. The selection problem is represented by independent genes taking on integer values and the

mutation operator employed for the selection problem is the popular *bit mutation*. The bit mutation operator chooses one of the possible selection alternatives for the object at the locus where bit mutation is to apply, and reassigns the chosen selection to that object. This may result in the same selection for the object in question and aims to prevent the loss of any good assignments. Figure 2.5 below illustrates the operation of these mutation operators on the same example chromosome of Figure 2.1.

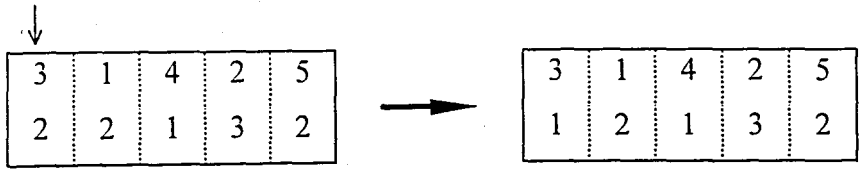


Swap mutation



Reposition mutation

(a) Swap and reposition mutation operators act on the ordering of object-selection pairs



(b) Bit mutation operates on the selection component

FIGURE 2.5. The operations of the mutation operators on the example chromosome.

2.6. Evaluation Routines

The evaluation routines evaluate the chromosomes to determine how good they perform in a given environment. Ideally, this is the only job of an evaluation routine. Yet if the encoding is such that some portion of the problem is not represented on the chromosomes, then this left out part is searched for by the evaluation routine.

It has been a common conclusion of many GA researchers including the author of this work that the evaluation routines (schedulers for scheduling problems, for example) and the repair algorithms be rather simple than complex and clever. This is because the GA is a black-box optimizer using the relative goodness of the chromosomes as the only feedback in deciding about their fate. This feedback is distorted if a smart evaluation and/or repair algorithm reorganizes material represented on the chromosome to construct a wonderful solution from a poorly performing individual.

In Chapters 3 to 5 below, GA-MCUOX will be applied to three problems from three different fields of scheduling research. In the context of scheduling, the chromosome represents all the information related to the scheduling problem except the explicit time settings. The time of production for each object (job, activity, operation) is given implicitly by the ordering of the object on the chromosome. This allows the use of simple schedulers which build the schedule associated with each chromosome obeying the ordering on the chromosome and following a simple dispatching rule. In one of the applications for example, nondelay scheduling rule is employed. This rule, in general, does not guarantee optimality. Still it provides very good results in many cases, and it is one of the most popular rules employed both in shop floors and in theoretical studies.

Also, in the context of scheduling, the representation scheme is not strictly direct anymore as the dimension of time is not included explicitly on the chromosome; but this does not cause a degradation in performance as time is implicitly given by the ordering of objects (e.g. jobs) on the chromosome, and simple and effective schedulers can generate good schedules using the ordering information.

2.7. Parameter Finetuning

The process of finetuning the parameters of the GA for good performance in specific environments is the most time-consuming step in any application. This is so because there are no rules to guide the researchers in the choice of parameters, and each problem represents another environment with a different fitness landscape so that previous experience with a different problem is of little use.

The first serious experimentation regarding the choice of parameters for the GA is done by Grefenstette (1986). Grefenstette attempted to determine the optimal control parameters for efficient GAs via a meta-level GA. The test bed he used consisted of the well-known five test functions of De Jong (1975). Experiments were performed both to optimize online performance and offline performance. The results demonstrated that the performance of a GA is a nonlinear function of control variables. And they also indicated that while it was possible to optimize GA control parameters, very good performance could be obtained with a range of values for these parameters.

In recent GA-literature, there are various approaches for parameter finetuning which can be classified mainly in three groups: *trial and error* approaches, use of *experimental designs*, and use of *meta-GAs*. The trial and error approach, as its name implies, consists of trying different operators and various sets of values for the parameters until a specific set is found which performs nicely in the test environment. This approach has been and continues to be used frequently, and in many application papers the finetuning process is summarized within a couple of sentences sounding like "We have experimented with various parameters and found that ...". The popularity is due to two reasons: it takes shorter than the other two approaches, and the GA is a robust algorithm showing a good performance within a range of values for its parameters. Once it has been found that a high crossover rate is essential, a choice between a rate of 0.80 and 0.85 is usually somewhat arbitrary.

The use of an experimental design is the least popular approach since it takes too long to accomplish a proper experiment. Experimenting with five parameters each with five different values already amounts to running 5^5=3125 GAs. Usually both the number of parameters and the number of levels are much higher, and this gives rise to days of computer run times. An example experiment with 13 months of run time is accomplished by Schaffer *et al.* (1989).

Use of meta-GAs for the finetuning process is both structured and relatively cheaper. The idea is that if a GA is able to effectively search for good parameter values in optimization problems, there is no reason why it can't be used to search for good performing values for its own parameters in specific applications.

Chromosomes in the meta-GA incorporate the parameters of the GA to be fine-tuned. Each gene on the chromosome is associated with one of the GA parameters. Chromosomes are initially generated randomly by selecting a value for each parameter from its range of values. They are then subjected to the evolutionary process using genetic operators over generations. The genetic operators used in the meta-GA in this work are the roulette wheel selection, two-point crossover and bit mutation operators.

Evaluating a chromosome is usually the most time consuming step in any GA application. With meta-GAs, this is readily true. Evaluation of a chromosome in a meta-GA is accomplished by running a GA with the parameters specified in the chromosome as many times as specified by the parameter denoting the number of replications. Usually, the average of the best solution values obtained in these replications is employed to determine the fitness of the chromosome in question. After all chromosomes are evaluated in this way, reproduction and recombination phases take place, and the process continues until a user-specified termination condition is fulfilled.

In the following chapters, an extensive trial and error approach is employed in one of the applications, while meta-GAs are used to fine-tune the parameters of GA-MCUOX in the other two applications.

Another alternative approach for parameter finetuning which hasn't been considered in GA-research so far is the use of evolutionary strategies (Rechenberg, 1973; Schwefel, 1975; Schwefel, 1977). This method might prove itself useful especially if the parameters to be finetuned are real-valued like the probabilities of crossover and mutation.

2.8. Concluding Remarks

GA-MCUOX developed here is one of many possible GAs employing MCUOX as the crossover operator. Other GA-MCUOX algorithms can be developed with different selection and/or mutation operators, different replacement strategies, adaptive parameters and so on. The aim here has not been to develop the best GA-MCUOX but to show the applicability and effectiveness of the new operator.

Run times reported in the applications in the following chapters include everything from chromosome processing to input and output operations. Also, no claim is made here that the most efficient coding has been utilized.

3. SIMULTANEOUS SCHEDULING OF MACHINES AND AUTOMATED GUIDED VEHICLES

In this chapter, a study accomplished by Ulusoy, Sivrikaya-Şerifoğlu and Bilge (1997) will be discussed in detail. The study applies GA-MCUOX developed as part of this dissertation to address the problem of concurrent scheduling of machines and a number of identical automated guided vehicles (AGVs) in a flexible manufacturing system (FMS). Automated Guided Vehicles Systems (AGVS) are advanced material handling techniques which are increasingly applied in computer integrated manufacturing (CIM) environments. Integrating the scheduling of AGVs into the overall scheduling activity is important as it will improve the performance of the FMS as a result of the coordination of the machine and the material handling system during the machine scheduling phase.

The problem of simultaneous scheduling of machines and automated guided vehicles (SSMAGV) is defined in the following general form:

Given a particular FMS environment and a set of jobs, determine the starting and completion times of operations for each job and the trips between work stations together with the vehicle assignment according to some system performance measure. The objective in this study is the minimization of makespan denoted by C_{max}. The makespan is defined as the time interval between the pick up of the first part from the load/unload station to the finishing time of the last operation processed. The number of AGVs is assumed to be two.

A literature review on SSMAGV and a detailed list of assumptions about the FMS can be found in the paper. Briefly, the assumptions state that all the times are known and deterministic, that all AGVs have the same speed and load carrying characteristics, that all vehicles start from a load/unload station (L/U) which serves as a distribution and collection center and return there after accomplishing all their assignments, that buffer capacities are infinite, and finally that both the operations and the trips are nonpreemptive.

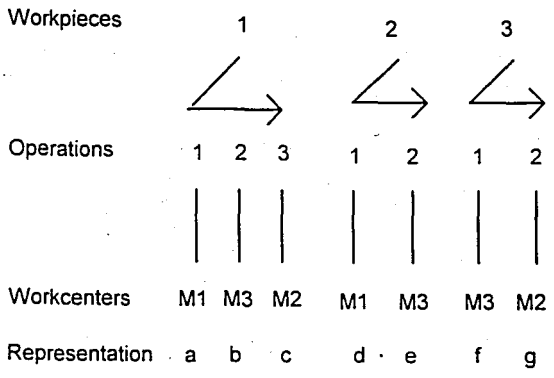
3.1. GA-MCUOX for Simultaneous Scheduling of Machines and Automated Guided Vehicles

In this application, objects to be sequenced are operations. For each operation, one of the AGVs is to be selected. Thus chromosomes represent the two dimensions of the search space: operation sequencing and AGV selection. The third dimension, time, is implicitly given by the ordering of operations on the chromosomes; the operations are assumed to be scheduled at their earliest starting time resulting in a nondelay schedule. Each operation denotes the processing of a specific part family on a specific machine center, and is expressed by an ASCII character starting from lower case letters.

A small example from Ulusoy, Sivrikaya-Şerifoğlu, and Bilge (1997) is reproduced in Figure 3.1. In this example, there are 3 workcenters, 3 workpieces and 2 AGVs. The first workpiece has 3 operations and the other 2 workpieces have 2 operations each. The workcenters where each operation will be performed are as indicated. The travel times and processing times are given in Figure 3.1(b) and (c). In the example chromosome [d2,f1,a1,e2,b2,g1,c2], AGV-1 is to carry operations *f*, *a*, and *g*; and AGV-2 the operations *d*, *e*, *b*, and *c* , in the given order on the chromosome. For example, workpiece 3 will be carried by AGV-1 from the L/U station to M3 for its first operation, *f* , to be performed. Similarly, workpiece 1 will be carried by AGV-2 from M1 to M3 for operation *b* to be performed. The Gantt chart of the schedule associated with this chromosome is provided in Figure 3.1(e). On this Gantt chart, the numerals (1) and (2) in front of each operation refer to the AGV that is assigned to carry the part involved.

3.1.1. Evaluation of Chromosomes

The makespan of the schedule associated with a chromosome is computed by building the nondelay schedule represented by the ordering of operations on that chromosome. The nondelay scheduling rule is chosen because of its relative success over



(a) Job set

machine	L/U	M1	M2	M3
L/U	0	4	8	10
M1	18	0	4	6
M2	20	14	0	8
M3	12	8	6	0

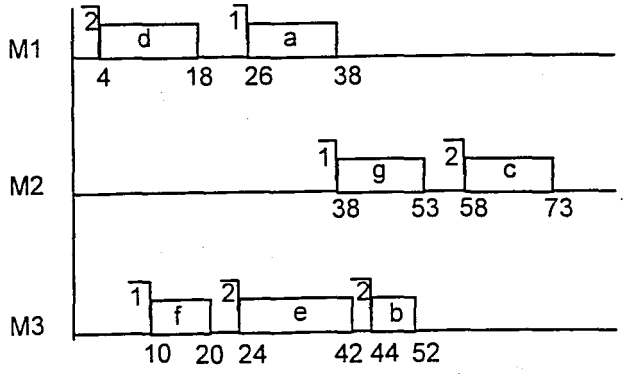
(b) Travel time (t) matrix

P1			P2			P3		
op	p	mach	op	p	mach	op	p	mach
a	12	M1	d	14	M1	f	10	M3
b	8	M3	e	18	M3	g	15	M2
c	15	M2						

(c) Processing Times

[d2,f1,a1,e2,b2,g1,c2]

(d) An example chromosome



(e) The Gantt chart for the schedule associated with [d2,f1,a1,e2,b2,g1,c2]

FIGURE 3.1. Example problem

active scheduling strategy for this particular problem as reported by Bilge and Ulusoy (1995). Parts are processed at a workcenter following the FCFS rule. When two parts arrive at the same workcenter at the same time, then SPT rule is used to break the tie in the ordering of these parts for processing.

Deadlock situations arise when AGVs have to wait for each other since the predecessors of the currently assigned operations are both not considered yet. As the number of chromosomes representing such situations was not high, the researchers decided to eliminate these chromosomes by penalizing them with a very high makespan value and hence a very low fitness value.

3.1.2. Operators

For the problem of simultaneous scheduling of machines and AGVs, MCUOX is rephrased as follows: Starting from the first *operations* on the parents, iteratively, one of the parents is chosen randomly and its next unconsidered *operation* becomes the next *operation* on the child. If the *AGV selected* for that *operation* is the same on both parents, then that selection is also made on the child; if not, one of the *AGV selections* of the parents is chosen randomly.

For the sequence of operations, the swap mutation is utilized, and for AGV selections, the bit mutation is employed. A simple repair function is developed to complement the operation swap mutation operator which may cause violations of the precedence relations. Figure 3.2 below presents the main part of the repair function which is repeatedly applied to the chromosome until it is precedence feasible.

The random solution generator constructs a chromosome gene by gene. The generator keeps a set of eligible operations to be scheduled next. An operation is said to be eligible once all its predecessors are assigned. Starting from the first position on the child chromosome, at each iteration, one of the operations in the eligible set is chosen randomly

and put onto the next position on the child. Then, one of the AGVs is randomly chosen to complete the gene. The eligible set is updated and the process continues until the child is completed.

```
Find positions of the operations which violate the precedence relations;
If the distance inbetween is smaller than half the chromosome length
    then
            swap violating operations
    else
            choose one of the operations randomly; take it out and reinsert it
            right before/after the other one depending on the precedence
            relations.
```

FIGURE 3.2. Repair function

In this application, reproduction and recombination processes are separated and a mating pool is formed explicitly. A user-specified rate, called mating pool formation rate and denoted by C, gives the proportion of the population that will be selected to form a mating pool from where parent solutions are chosen to undergo recombination. Figure 3.3 illustrates the pool formation process in the GA. There are three strategies for the selection of individuals for both generation gap and mating pool formation: i) fitness-based; ii) elitist, where only the best solution is preserved and rest is chosen fitness-based; iii) elitist, where all the best (*popsize***G*) or (*popsize* **C*) solutions are preserved with *popsize* being the population size and *G* being the generation gap formation rate.

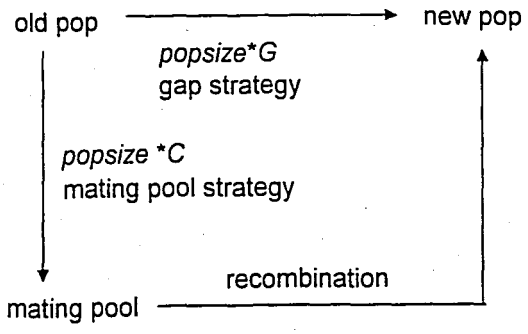


FIGURE 3.3. Pool formation structure

3.1.3. Finetuning of the Parameters of GA-MCUOX

In this study, a trial and error approach has been used to finetune the parameters of GA-MCUOX and to evaluate various search strategies. A test bed comprised of 20 of the generated test problems is utilized in the experiments. The following ten parameters are investigated regarding their effects on the performance of the GA: population size, number of generations, crossover, bitwise mutation and operation swap mutation rates, generation gap and mating pool formation rates and strategies, and the form of the fitness function. For population size, various levels above 50 are tried. Number of generations is varied between 30 and 400; crossover rate between 0.50 and 1.00; bitwise mutation rate between 0.001 and 0.03; and operation swap mutation rate between 0.00 and 0.50. Several levels between 0 and 1 are tried for the generation gap formation rate G and mating pool formation rate C. Also, all three strategies are investigated for both generation gap formation and the mating pool formation processes. Two different fitness functions are tried; namely, $f(x)=1/z(x)$ and $f(x)=1/z^2(x)$.

The experimentation has shown the viability of a well established and widely used search strategy for off-line optimization of difficult problems as suggested by Goldberg (1989c): Given a fixed number of evaluations, running smaller sized GAs many times on the same problem is better than running a larger GA for a few times. Each random restart yields a different sampling of the search space; and over many replications, the probability of locating the global optima increases.

The parameters are fixed as follows: GA-MCUOX is replicated 20 times for each problem. For all sets of problems, each replication of the GA is run for 100 generations. The objective function of each new chromosome is compared with the lower bound for the problem under consideration. If they are equal, then, since an optimal solution is found, the runs for that replication are terminated before completing all 100 generations. The population size is selected to be 140. For all the problems, mating pool formation rate C and generation gap G are taken to be 1 and 0.04, respectively. Mating pool formation strategy is fitness-based, while gap generation strategy is elitist where all the best chromosomes ($popsize*G$) are preserved from one generation to the next. Crossover rate

is selected to be 0.8. The rate of operation swap mutation per chromosome is 0.4 and the bitwise AGV mutation rate is 0.01. The fitness function $f(x)=1/z^2(x)$ has been more effective in sorting out better individuals, therefore it has been utilized in the GA runs.

One important characteristic of the problem is the multimodality of its fitness landscape. Furthermore, there are a large number of representatives for each (local) optimum. This is also due to the fact that makespan is an aggregate measure of performance, and for each value of makespan there may be several schedules having that value as the makespan. The crossover rate, mutation rates and the fitness function do provide a balance between exploration for new solutions and the exploitation of present solutions in this landscape. High crossover rate and mutation rates enhance the exploration while the power function in the fitness formula empowers the bias towards better solutions in the selection process. MCUOX helps in this exploitation process while high mutation rates help to escape the local optima. Also the explicit mating pool formation enhances the selection pressure by subjecting individuals to a two-step elimination process so that really best individuals obtain the chance to reproduce.

3.2. Numerical Study

3.2.1. Generation of Test Problems

GA-MCUOX has been tested on 180 different problems generated. Each problem is a combination of a layout and a job which consists of a set of parts to be processed. There assumed to be three different layouts and four different job sets. The layouts are displayed in Figures 3.4, 3.5, and 3.6 where the travel distances in meters are indicated on each arc. The travel times are calculated from the distances by assuming an average

speed of 40 m/min for the AGVs. The loading and unloading times of the parts onto and from the AGVs are taken to be 0.5 min each.

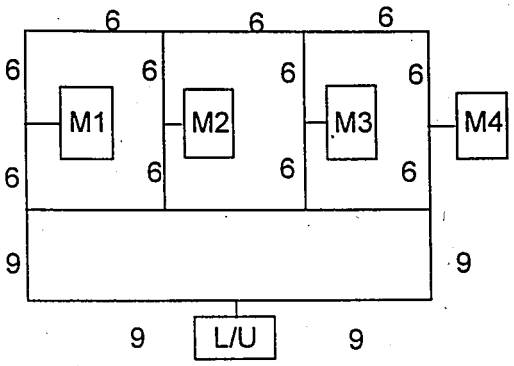


FIGURE 3.4. Layout 1 (not to scale).

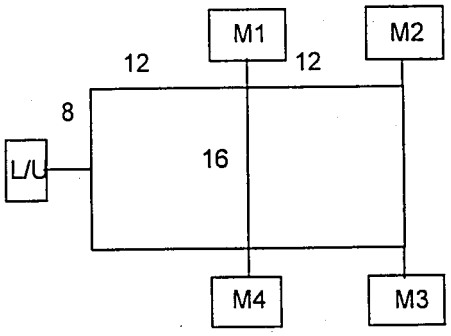


FIGURE 3.5. Layout 2 (not to scale).

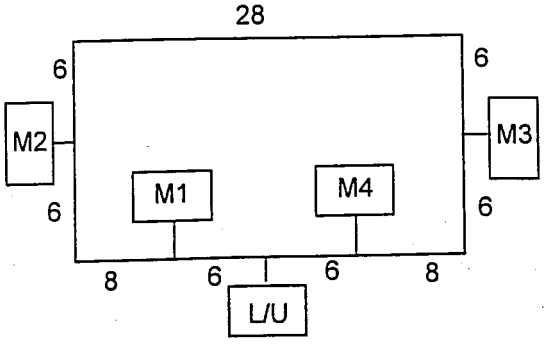


FIGURE 3.6. Layout 3 (not to scale).

Two different part pools are generated. It is assumed that there is a strict linear precedence relation between operations of a part. Thus in the following, parts are specified by the respective part routings which is in turn defined by the sequence of the machines visited by the parts.

The first part pool : {4–2–3}, {3–1–4}, {2–4–3},{2–1–3}, {3–4–2–1}, {4–2–1}, {1–2–4}, {3–2–4}, {1–3–2}, {1–2–3–4}, {4–3–1}.

The second part pool : {3–4}, {1–2–3–4}, {1–2–4}, {1–2}, {1–3–4}, {1–3}, {1–2–3}, {2–3}, {2–4}, {1–4}, {2–3–4}.

Using these part pools, four job sets are generated; the first two using the first part pool, and the last two using the second part pool. The second and fourth job sets consist of jobs with a larger number of parts and thus a larger total number of operations compared to the jobs in the first and third job sets. Each job set is applied to all three layouts.

The first job set . This job set contains 30 jobs . The jobs consist of 6–10 parts randomly selected with replacement from the first part pool.

The second job set . There are 5 jobs in this set. The jobs consist of 15–20 parts randomly selected with replacement from the first part pool.

The third job set . This job set contains 20 jobs. The jobs consist of 6–10 parts randomly selected with replacement from the second part pool.

The fourth job set . There are 5 jobs in this set. The jobs consist of 15–20 parts randomly selected with replacement from the second part pool.

The processing times for the operations are randomly generated as integers from the interval [3,15] minutes. An important characteristic of the problems is the relative magnitude of the travel times and the processing times represented as the ratio $(\bar{t}/\bar{p})$ where $\bar{t}$ is the average of all the nonzero entries in the travel time matrix and $\bar{p}$ is the average of all processing times. The problems are generated with a $(\bar{t}/\bar{p})$ ratio in the range [0.05, 0.20] in order to reflect the practice into the test problems.

3.2.2. Computation of a Lower Bound

To compute a lower bound for SSMAGV, it has been assumed that an AGV is always available when needed and that there is no idle time inbetween operations so that on each machine all the operations are processed one after the other. The only idle time on a machine is the time span necessary for the arrival of the first workpiece to be processed on that machine. This time span includes the travel times and the nondelayed processing of the predecessor operations. If there is an operation within the set of operations to be processed on a machine which is the first operation of its part, then only the travel time from the L/U unit to that particular machine need to be accounted for. If not, then, from among alternative choices for the first operation, the one is chosen which gives rise to smallest idle time. LB is then taken to be the maximum of all LBs for completion times on all machines. Equations (3.1) to (3.4) detail the formulation of the lower bound.

$$PT_j = \begin{cases} L + t\left[L/U, M(j)\right] + U, & \text{if } P(j) = \varnothing \\ \sum_{k \in P(j)} \left(L + t\left[M(pred(k)), M(k)\right] + p_k + U\right) + & \\ \qquad L + t\left[M(pred(j)), M(j)\right] + U, & \text{o/w} \end{cases} \tag{3.1}$$

$$IT_{1_i} = \min_{j \in O_{M_i}} \left\{PT_j\right\}, \tag{3.2}$$

$$C_{M_i} = IT_{1_i} + \sum_{j \in O_{M_i}} p_j, \tag{3.3}$$

$$LB = \min_i \left\{C_{M_i}^{1}\right\}, \tag{3.4}$$

where

$P(j)$ set of predecessors of operation j

O_{M_i} set of operations to be processed on machine i

$pred(j)$ predecessor of operation j; the first operations of each part are preceded by a dummy operation d where $M(d)=L/U$

$M(j)$ machine index upon which operation j is performed

L, U loading and unloading times

PT_j time needed for the processing and transportation of all predecessors of operation j

IT_{1_i} the first idle time on machine i

C_{M_i} completion time of the last operation processed on machine i.

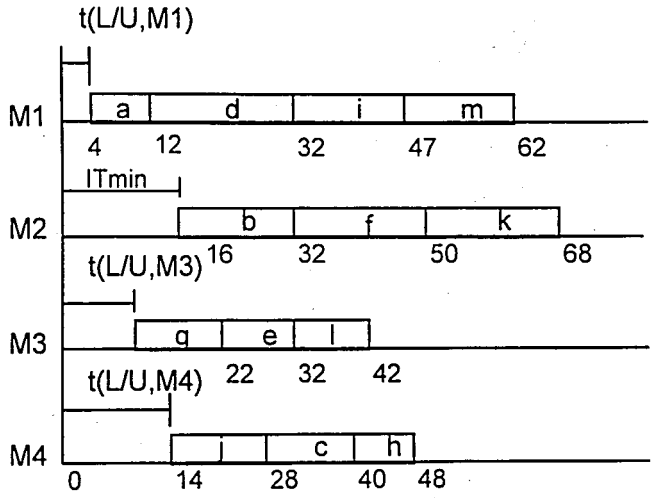
A small illustrative example is taken from Ulusoy, Sivrikaya-Şerifoğlu and Bilge (1997) and reproduced in Figure 3.7 below. In the example problem, the processing of parts P1 and P2 start on M1; P3 and P5 start on M3 and P4 starts on M4. Therefore these machines can start processing as soon as an AGV transfers a part from the L/U station to the respective machine. This means, a lower bound on the completion times for M1, M3 and M4 can be computed as the sum of the processing times of the operations which are to be processed on these machines and the transfer time from the L/U station. They are calculated and are shown in Figure 3.7 (a). Since no part has its first operation processed on M2, a lower bound on M2's completion time is determined by considering all the operations to be performed on M2 (namely, $b, f,$ and k) one by one as the first operation to be processed by M2. Starting with operation b leads to an idle time of t[L/U,M1]+p(a)+t[M1,M2]=4+8+4=16, which is the processing time of the predecessor operation a plus the transfers. Similarly, starting with the operation f gives rise to an idle time of t[L/U,M1]+p(d)+t[M1,M3]+p(e) +t[M3,M2]=46 and with operation k to an idle time of t[L/U,M4]+p(j)+t[M4,M2]=40. The minimum of these is 16 for operation b (denoted as IT_{min} in Figure 3.7), thus b is processed first on M2. The lower bound on the completion time of M2 becomes the sum of the processing times of b, k, and f plus IT_{min}. Lower bound LB is determined as the maximum of the lower bounds on the completion times of all the machines. Here, it corresponds to that of M2 and has a value of 68.

P1			P2			P3			P4			P5		
op	p	mach	op	p	mach	op	p	mach	op	p	mach	op	p	mach
a	8	M1	d	20	M1	g	12	M3	j	14	M4	l	10	M3
b	16	M2	e	10	M3	h	8	M4	k	18	M2	m	15	M1
c	12	M4	f	18	M2	i	15	M1						

(a) Parts and processing times

machine	L/U	M1	M2	M3	M4
L/U	0	4	8	10	14
M1	18	0	4	6	10
M2	20	14	0	8	6
M3	12	8	6	0	6
M4	14	14	12	6	0

(b) Travel time (t) matrix



(c) The Gantt chart for the lower bound

FIGURE 3.7. Example problem for the computation of the lower bound

3.2.3. Numerical Results on the Set of 180 Test Problems

Table 3.1 below provides a summary of the results on the randomly generated test problems presented extensively in Ulusoy, Sivrikaya-Şerifoğlu and Bilge (1997). The entries in the table are the average percent deviations of the best solutions provided by GA-

MCUOX from the lower bounds averaged over 30, 5, 20 and 5 problems for job sets 1, 2, 3 and 4 respectively over the three layouts. The average deviation varies between 0.00 percent and 3.71 percent among the twelve problem sets. The average deviation over all problems is calculated to be 2.53 percent. In 108 out of 180 problems (60 percent), the GA solution coincides with the LB implying that the solution is optimal.

TABLE 3.1. Summary of results on the 180 test problems

* average percent deviation of the best GA-MCUOX solution from the lower bound

Job Set	Layout	Avg%Dev*
1	1	3.02
	2	3.71
	3	3.32
2	1	0.00
	2	0.00
	3	0.00
3	1	2.64
	2	2.59
	3	1.45
4	1	1.37
	2	1.62
	3	1.00

An important observation made by the researchers is that compared to the magnitude of the mean values of solutions obtained by GA-MCUOX over 20 replications, the standard deviation values are extremely small. For problems with a nonzero standard deviation the coefficient of variation varies in the range [0.10, 4.28]. For 61 out of 180 problems the standard deviation is zero. Furthermore, the alternative solutions found by GA-MCUOX are not all the same. In fact, the final populations of the 20 replications for

these problems contain various distinctly different solutions with the same C_{max} value. Thus there are many alternative optima and GA-MCUOX is capable of finding them.

Another observation made is that the mean values of solutions are relatively close to each other over the different layouts. Results of t–test confirm the observation for all problem sets at a significance level of 0.005. A major reason for this result is proposed to be the relatively low $(\bar{t}/\bar{p})$ ratio.

In this application, GA-MCUOX is implemented on an IBM-compatible personal computer with a 486 microprocessor of 60 MHz. Run times are in the range of 1.50 minutes to 3.75 minutes per replication per problem.

3.2.4. Comparison of GA-MCUOX with a Time Window Approach

GA-MCUOX is also tested on another set of 82 problems which were used to test a time window approach by Bilge and Ulusoy (1995) on the same problem. The problems are grouped into two groups; one with relatively low $(\bar{t}/\bar{p})$ ratio and one with relatively high $(\bar{t}/\bar{p})$ ratio. In 59 percent of all the problems GA-MCUOX outperforms the time window approach where the reverse is true only in 6 percent of the problems. GA-MCUOX performs up to 7.19 percent better than the time window approach in the first group and up to 6.54 percent better in the second group. Average improvement of GA-MCUOX over the time window approach increases from 1.16 percent in the first group of problems to 2.16 percent in the second group.

Another observation made within this set of experiments confirms the expectation that the performance of the lower bound should deteriorate with increasing $(\bar{t}/\bar{p})$ ratio because of the assumption of zero travelling times. The deviation of best solutions from

the lower bound is 5.30 percent within the first group of problems whereas it is 26.68 percent within the second group.

3.3. Conclusion

In this chapter, a study accomplished by Ulusoy, Sivrikaya-Şerifoğlu and Bilge (1997) is discussed which employs GA-MCUOX developed as part of this dissertation to attack the problem of the simultaneous scheduling of the machines and a number of identical AGVs in an FMS. Results of the extensive experiments and comparisons with the lower bound suggest that GA-MCUOX performs well on the problem. In 60 percent of the 180 randomly generated test problems, GA-MCUOX finds a global optimum solution. The average deviation from the lower bound over all the problems is found to be 2.53 percent. GA-MCUOX is also compared with a heuristic approach developed for the same problem on a test bed of 82 problems from literature. GA-MCUOX outperforms the heuristic in 59 percent of the problems whereas the reverse is true in only 6 percent of the problems.

3.4. Post-Analysis of the Representation in this GA-MCUOX Application

An interesting observation regarding the representation in this application is that the assumption of identical AGVs introduces some redundancy in representation. This may happen in general when the selection problem involves alternatives which are identical to each other. In this particular problem, the selection problem involves two AGVs which are identical. This means that (number of AGVs)! chromosomes represent the same information. For example, AGV 1 could have been identified by 2 rather than 1 and AGV 2 by 1, and nothing would change since the two AGVs are identical. For small cardinalities of alternative sets, this redundancy is only minor and constitutes no problem for GA performance.

When the selection problem is common to all objects in addition to involving identical alternatives, the generic multi-component problem MCCOP-SS becomes a complex grouping problem: In addition to grouping objects within a given number of groups, the objects in each group are to be sequenced. In the current application, for example, the selection problem for each object is the same and involves the choice of one AGV from among two identical ones. Thus the problem becomes a complex grouping problem where operations are to be assigned to one of the AGVs, and they are to be sequenced in the order they should be transferred to their corresponding machine center by the AGV they are assigned to.

One can now go a step further and claim that the redundancy in terms of GA representation is even bigger in such a case since any mixing of the members of different groups on the chromosome without changing their relative intra-group orderings would yield the same schedule; and hence all those chromosomes are the 'same'. For example, [d2,f1,a1,e2,b2,g1,c2], [f1,d2,a1,e2,b2,g1,c2], [f1,a1,d2,e2,b2,g1,c2], ... describe the same schedule and could be simply written as [f1,a1,g1,d2,e2,b2,c2] or [f2,a2,g2,d1,e1,b1,c1].

But this claim is not totally true; these chromosomes are not really all the same. The reason is that the two chromosomes (and any two chromosomes with the same relative intra-group ordering but different mixing of genes) respond totally differently to the

operations of the genetic operators; i.e. they produce different offspring when subjected to the genetic operators. This means that each one of the chromosomes jumps to a different portion of the search space when subjected to genetic operators. As a simple illustration, suppose that a bit flipping mutation is to apply to the second gene of the first two example chromosomes. The resulting chromosomes are [d2,f2,a1,e2,b2,g1,c2] and [f1,d1,a1,e2,b2,g1,c2], respectively, and they are in no way the same.

This observation is quite interesting as it presents a potential advantage of redundancy in representation which has traditionally been associated with any loss in performance.

3.4.1 Performance of GA-MCUOX on a Pure Grouping Problem

Motivated by the above mentioned performance of GA-MCUOX in the presence of some redundancy in representation, some additional experiments are carried out to further investigate its behavior in the presence of larger amounts of redundancy. For this purpose, a pure grouping problem, i.e. a problem where a set of objects are to be grouped into a given number of groups, is attacked. Redundancy in the representation employed in this work is immense in such a problem: Given a grouping of the objects, any ordering of objects within the groups represents the same information.

The problem chosen is the equal piles problem cited in Falkenauer (1995), where 34 objects with given weights are to be grouped in 10 groups such that the total weight in each group is equal. Falkenauer refers to Jones and Beltramo (1991), who compare nine GAs on this instance differing in encoding and operators. The best of those nine GAs has been able to find a total deviation of 2 units of group weights from the desired level. The mean value of the best results of 30 replications has been 171. Falkenauer tests the grouping GA he has developed on this problem and identifies several optima with 0 total deviation.

GA-MCUOX is tested on the same problem using the same population size and number of evaluations as Falkenauer's grouping GA. Within the same number of replications of 30, a best solution of 2.00 has been found. The mean value of the best results of 30 replications is 33.20 and the standard deviation is 12.74.

These results suggest that GA-MCUOX is able to perform satisfactorily in the presence of redundancy. They also indicate that MCUOX is able to recombine useful partial solutions even if they are represented in various different forms.

4. RESOURCE CONSTRAINED PROJECT SCHEDULING WITH DISCOUNTED CASH FLOWS

The resource constrained project scheduling problem (RCPSP) is the problem of scheduling activities under resource and precedence restrictions with the objective of minimizing the makespan. When the problem involves more than one project each with a given due date, other time related performance measures are employed like the mean completion time and mean lateness (tardiness). RCPSP is proven to be NP-hard (Blazewicz *et al.*, 1983).

The resources are classified based on their availability (Shtub, Bard and Globerson; 1994): Resources which are available at the same level every time period such as a fixed workforce are called renewable resources. Resources which come in a lump sum at the beginning of the project and are used up over time such as computer time are called nonrenewable or depletable resources. There are some other resources which are available in limited quantities every time period; in addition, their total availability throughout the project is also constrained. Such resources like cash are called doubly constrained resources.

Until recently, the majority of research on RCPSP assumed that the duration of an activitiy is fixed with a given resource usage. In more realistic formulations, an activity's duration might be crashed by assigning it extra units of resources at additional cost. The duration-resource consumption pairs are called the modes of the activity. These modes represent a time-cost tradeoff problem for the activity, and their presence increases the problem complexity.

An important extension of RCPSP is the resource constrained project scheduling problem with discounted cash flows (RCPSPDCF). Here, cost considerations come into the picture. Cash in- and out-flows are associated with the completions of activities and the time

value of money is taken into consideration. The objective becomes the maximization of the net present value (NPV) of the cash flows.

Özdamar and Ulusoy (1995) provide a recent survey on project scheduling problems with limited resources. In this survey, a broad range of research efforts are covered including single- and multi-project models with discrete and continuous time-resource functions, with time related and cost based objectives, and with various types of constraints (ranging from precedence only to precedence and doubly constrained renewable- and nonrenewable-resource constraints). Both optimization and heuristic approaches are discussed.

As pointed out in that survey, minimization of makespan and maximization of NPV are the two mostly emphasized objectives in the field. Models incorporating NPV criteria usually assume a project due date and/or a large positive cash flow at the end of the project, since otherwise activities involving negative cash flows would be delayed indefinitely to maximize NPV (Elmaghraby and Herroelen, 1990). Under these assumptions, the two criteria of makespan minimization and NPV maximization support each other. RCPSPDCF attacked here incorporates both of these assumptions.

4.1. Problem Definition and Literature Survey

The definition of RCPSPDCF attacked in this chapter is as follows: A single project is given with a set of activities and a set of renewable resources. Precedence constraints are given on the sequencing of activities. There are resource constraints on the availability of renewable resources in each time period. Associated with each activity, there is a set of modes representing time-cost tradeoffs. Associated with each resource, there is a unit consumption cost. Cash outflows (CF_j^-) occur at the start of each activity j, and a single lump sum payment (LSP) is received at the completion of the project. The activities are to be scheduled such that

the makespan of the project (C_{max}) does not exceed a given due date (DD) and the NPV of all the cash flows as given by the following formula is maximized:

$$NPV = \sum_{j} CF_j^- (1+r)^{-ST_j} + LSP(1+r)^{-C_{\max}}, \qquad (4.1)$$

where r is the discount rate, ST_j is the start time of activity j, and CF_j^- are computed as follows: First, the total cost of resource usage per unit time are computed by multiplying the unit cost of each resource by the number of units required by activity j and by summing the products over all resources. Then the sum is multiplied by the duration of activity j.

An extensive review of research attacking this problem is given in the above cited survey of Özdamar and Ulusoy (1995). Two recent algorithms for this problem are the iterative algorithm of Li and Willis (1992) and the so called local constraint based analysis (LCBA) of Özdamar and Ulusoy (1994).

Li and Willis (1992) discuss the difference in the resource profiles generated by forward and backward scheduling techniques. The forward technique schedules the activities as early as possible whereas the backward technique schedules the activities as late as possible. Thus the backward technique delays the cash outflows associated with the activities and decrease the time between these outflows and the payments for the progress (or completion) of the project. This decreases the interest payments on the funds borrowed and increases NPV of the project. Yet the backward technique is not without flaws. Li and Willis discuss the problems associated with backward scheduling in detail. Some of these problems are that the backward technique may produce a project duration longer than when the forward technique is used (which means that the savings from financing costs are offset by the fixed costs due to longer project duration); that scheduling activities as late as possible makes them all critical activities and a delay in starting any activity affects the whole project; and that backward scheduling technique often fails to produce any feasible schedule at all. Li and Willis propose

an iterative algorithm which tries to incorporate the merits of both the forward and backward scheduling and thereby produce projects which are both short and cheap.

Local constraint based analysis (LCBA) of Özdamar and Ulusoy (1994) aims to achieve this same result. As the name implies, this technique analyses resource conflicts locally on small segments of the project. At every scheduling time point, through an activity selection process, a set of schedulable activities are identified from which a group of activities are chosen as the highest priority group in the allocation of scarce resources. The delay of activities from this group extends the project duration. The same selection process also identifies another set of activities to be discarded from the set of schedulable activities. After the activities in the highest priority group are all scheduled, activities not discarded from the schedulable set are considered for scheduling.

Ulusoy and Özdamar (1995) test the performance of LCBA against a couple of good decision rules and the algorithm of Li and Willis. LCBA and the other decision rules are first used as single pass heuristics and then are embedded into an iterative scheme. The results demonstrate that all decision rules benefit from being embedded in an iterative algorithm; LCBA's performance, in particular, is three-folded by the iterative algorithm. The results of the comparisons show that LCBA performs far better than the other approaches including the algorithm of Li and Willis. Therefore GA-MCUOX will be compared against LCBA in this application.

The results also indicate that there is an evident relationship between the improvements in the objectives of makespan minimization and NPV maximization, i.e. when one improves the other improves too. This mutual support between the two criteria has also been demonstrated by Smith-Daniels and Aquilano (1987).

4.2. GA-MCUOX for Resource Constrained Project Scheduling with Discounted Cash Flows

The RCPSPDCF as defined above is an example for MCCOP_SS. Here, the objects to be sequenced are activities. The selection problem associated with each activity is the selection of one of its possible modes. The extra difficulty here is that there are precedence relations constraining the sequencing of activities.

4.2.1. Representation and Evaluation of Chromosomes

The chromosome in this context decodes into an ordering of accomplishment for the activities together with their selected modes. This representation applies to both RCPSP and RCPSPDCF. Only a change in the objective function is needed to switch from one to the other.

As a small illustrative example, suppose that five activities are given within the network presented in Figure 4.1. There is only one renewable resource with a given availability and a unit utilization cost. Input data are listed in Table 4.1.

The example chromosome [3-2 1-2 4-1 2-3 5-2] from Figure 2.1 can be taken to represent a potential solution for this problem. This chromosome would sequence the activities in the order 3, 1, 4, 2, and 5. This is a feasible sequence in terms of the precedence relations. The activities 3, 1 and 5 are to be accomplished in their second modes; activity 4 in its first mode; and activity 2 in its third mode.

To schedule the activities in the sequence on the chromosomes, a forward scheduling scheme is preferred due to the failures of the backward scheduling scheme as summarized above and discussed in length by Li and Willis (1992). A simple scheduler schedules the activities in the given order on the chromosome by keeping track of the resource availabilities.

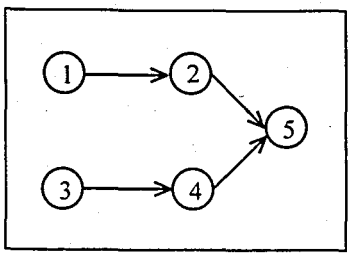


FIGURE 4.1. Project network for the RCPSP example problem with activity-on-node representation

TABLE 4.1. Input data for the RCPSPDCF example problem

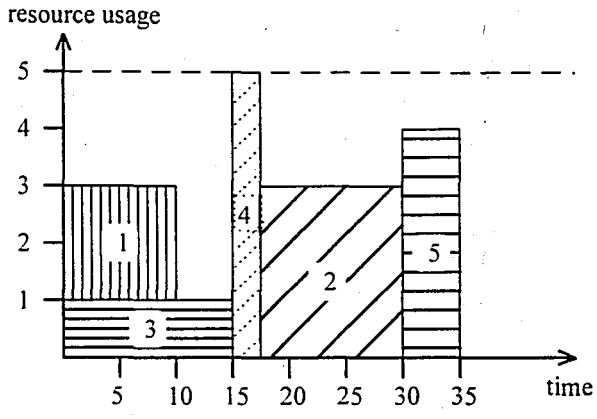
Activity	Mode	Duration	Resource Requirement
1	1	3	5
	2	10	2
2	1	2	5
	2	4	4
	3	13	3
3	1	6	4
	2	15	1
4	1	2	5
	2	9	1
5	1	1	5
	2	5	4
	3	17	1

Once the activities are scheduled, NPV of the project can be found from the resulting cash flow diagram and the formula (4.1). Fitness of the chromosome is then taken to be equal to the NPV value, i.e.

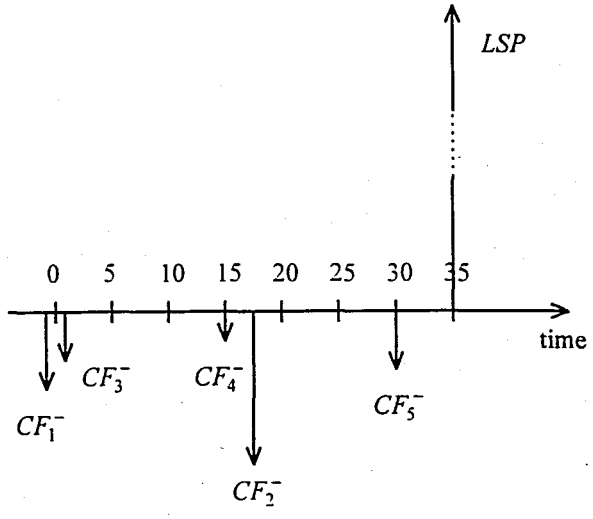
$$f(x)=z(x), \tag{4.2}$$

where z(x) is the objective function value (that is, NPV of the associated schedule).

Assuming a unit cost of 10 and an availability of 5 units per period for the single resource, a discount rate of 0.005 and an LSP of 1750, the example chromosome will give rise to the resource profile and the cash flow diagram given in Figure 4.2 (a) and (b).



(a) Resource profile



$CF_1^- = -200$, $CF_2^- = -390$, $CF_3^- = -150$, $CF_4^- = -100$ and $CF_5^- = -160 \Rightarrow NPV = +530.84$

(b) Cash flow diagram

FIGURE 4.2. Evaluation of the example chromosome [3-2 1-2 4-1 2-3 5-2]

It should be noted that the scheduler is not a simple nondelay scheduler. A nondelay scheduler would schedule activities at the earliest possible start time which is allowed by the precedence and resource constraints. The scheduler here keeps track of events which are the completion times of one or more activities. The next activity on the chromosome is scheduled at the latest event time where its scheduling does not violate precedence and resource constraints. This scheduling scheme is in line with the objective of NPV maximization as the cash outflows are delayed without causing any harm to the other parts of the schedule. The effect of this scheme is best illustrated with the best solution obtained in the RCPSPDCF example with N=10 of Section 2.4.2. The corresponding chromosome is [2-1, 1-1, 4-2, 3-2, 5-2, 7-1, 6-1, 8-1, 9-2, 10-2] and the associated resource profile is given in Figure 4.3 below.

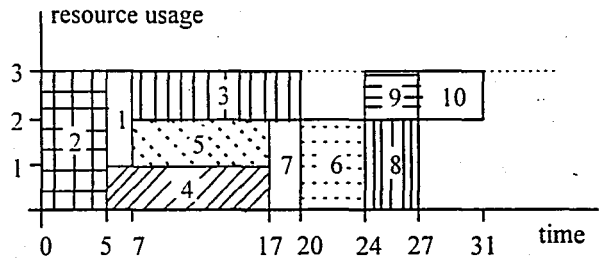


FIGURE 4.3. Resource profile for the example chromosome
[2-1, 1-1, 4-2, 3-2, 5-2, 7-1, 6-1, 8-1, 9-2, 10-2]

In Figure 4.3, events like completion of activity 2, of activity 1, of activities 4 and 5, etc. take place at time points 5, 7, 17, etc. Activity 9 is scheduled to start at time point 24 although it could perfectly be scheduled at time point 20. This delays the cash outflow associated with activity 9 and results in an increase in the NPV of the overall schedule. Although the scheduling scheme resulted in latest-start-time scheduling in this example, this may not be the case in general. If the duration within the first mode of activity 8 were four time units instead of three, a slack time of one unit would remain after activity 9 would be finished.

4.2.2. Operators

In the context of RCPSP, MCUOX can be rephrased as follows: Starting from the first *activities* on the parents, iteratively, one of the parents is chosen randomly and its next unconsidered *activity* becomes the next *activity* on the child. If the *mode* selected for that *activity* is the same on both parents, then that *mode* is also selected for the *activity* on the child; if not, one of the *mode* selections of the parents is chosen randomly.

One of the most important advantages of MCUOX is that it does not violate the precedence constraints. This property proves to be very useful in this application as there are precedence constraints on the sequencing of activities; and the fact that the application of MCUOX will not violate them eliminates the need for repair functions as a complement to the crossover operator in generating feasible offspring.

There are two mutation operators, one for each dimension of the search: The reposition mutation operator is employed for activities and the bit mutation operator for mode selections.

The application of the reposition mutation may cause precedence violations. A simple repair function complements the reposition mutation operator in generating feasible mutants. Suppose that the gene at locus *pos1* is repositioned into another locus *pos2*. The simple repair algorithm checks precedence violations for the two cases of *pos1*<*pos2* and *pos1*>*pos2*. When *pos1*<*pos2*, the activity now in locus *pos2* may have flown past some successor(s) on the way from *pos1* to *pos2*. If this is the case, the repositioned gene is taken out of *pos2* and put into the location right before the first successor in the interval from *pos1* to *pos2*-1. When *pos1*>*pos2*, the activity now in locus *pos2* may have flown past some predecessor(s) on the way from *pos1* to *pos2*. If this is the case, the repositioned gene is leap-frogged right after its last predecessor in the interval from *pos2*+1 to *pos1*. If the first successor (last predecessor) in the interval is located next door to the original position *pos1*, this repair routine will reverse the effect of the reposition-mutation, and the chromosome will be back in its original configuration that it had before the application of the reposition-mutation. This in turn means

that the effective mutation rate will be lower than the specified rate $pm_{reposin}$, and hence a relatively higher rate is expected to result in the fine tuning process.

Here the reposition mutation operator is chosen as opposed to the more popular swap mutation for the following reason: Swap mutation operator repositions two activities which means that it can give rise to two violations of the precedence relations and hence to two repairs. As repair means reorganization of the material on the chromosome, its use should be avoided whenever possible.

The chromosomes for the initial population are generated randomly by a solution generator. To this end, an eligible set of activities is kept by the generator which is initially formed by the activities with no predecessors. At each step of the generation, one of the activities from the eligible set is chosen and put into the next free position on the chromosome. One of the possible modes of that activity is selected randomly to complete the gene. The eligible set is updated by deleting the selected activity and adding activities predecessors of which are all located on the chromosome. The process iterates until the chromosome is completed.

4.2.3. Parameter Finetuning

A meta-level GA has been designed and employed for the optimization of the parameters of the GA- MCUOX for RCPSPDCF. In this study, seven parameters are subjected to the fine-tuning process via the meta-GA. These parameters and the range of values investigated are listed in Table 4.2 below in the order in which they appear on the chromosomes. One chromosome of the meta-GA might be for example [50, 160, 8, 0.20, 0.30, 0.00, 3] describing a GA with a population size of 50 and number of generations of 160 to be replicated 8 times. The probability of crossover, reposition mutation and bit mutation are 0.20, 0.30 and 0.00 respectively. G=3/*popsize*, so at each generation the best three chromosomes are

transferred in tact into the next generation. From the table it follows that the meta-GA is to search through a space of more than 2^{28} possible parameter sets.

TABLE 4.2. Parameters and their range of values fine-tuned in the meta-GA

Parameter	Identifier	Values
population size	*popsize*	20,30,40,..,150
number of generations	*nGen*	10,20,30,..,250
number of replications	*nRep*	5,8,10,12,15,18,20,22,25,28,30
probability of crossover	*pc*	0, .10, .20, .30, .40, .45, .50, .55, .60, ..,1
probability of bit mutation	pm_{bit}	0, .05, .10, .15, .20, .., .50, .60, .70, ..,1
probability of reposition mutation	$pm_{reposln}$	0, .05, .10, .15, .20, .30, .40, ..,1
generation gap	*G*	1/*popsize*,..,20/*popsize*

To evaluate a chromosome in the meta-GA, the GA with the parameters specified in the chromosome is run as many times as specified by the *nRep* parameter. The payoff value is then taken to be the average of best solution (NPV) values of all these replications. Fitness of the chromosome *X*, *F(X)*, equals the payoff value, i.e.

$$F(X) = \bar{Z}(X), \tag{4.3}$$

where $\bar{Z}(X)$ is the average of the best solution values obtained using the parameters in X to run GA-MCUOX for *nRep* replications.

The problem with the maximal sum of number of activities and number of resources is chosen from the test suite to be used in the fine-tuning study. This problem has 53 activities and 3 resources. The discount rate is taken to be 0.005. The parameters of the meta-GA are as follows: population size is 51, probability of crossover is 0.60, and probability of bit mutation is 0.125.

The meta-GA is run on this problem for 60 generations. The resulting best chromosome has yielded the following GA parameters: *popsize*=140, *nGen*=240, *nRep*=8, pc=0.65, pm_{bit} =0.05, $pm_{reposln}$=0.50, and G=19/140=0.14.

4.3. Numerical Study

GA-MCUOX is compared against the iterative version of LCBA on a set of 93 problems from literature (Özdamar and Ulusoy, 1994). These problems involve up to 60 activities, 5 renewable resources and 3 modes of accomplishment per activity.

The algorithms are run on the test suite, and for each problem both the maximum NPV value and the minimum C_{max} value are recorded. In case of LCBA, the iterative runs result in a set of solutions, some of which are efficient while some are not. For each problem, from among the efficient solutions, the one with the maximum NPV value is chosen as the solution provided by LCBA.

Two comparative analyses are carried out on LCBA and GA-MCUOX using these run results: The average percentage of improvement in NPV values and the average percentage of improvement in C_{max} values brought by GA-MCUOX are computed and are subjected to t-tests. Table 4.3 displays the results.

The results indicate that GA-MCUOX yields an average improvement of 18.37 percent in NPV values as compared to LCBA. This is an important improvement at a significance

level of 0.0005. In C_{max} values, GA-MCUOX yields an average improvement of 0.97 percent which is significant at a level of 0.01. Here, the C_{max} values reported for GA-MCUOX may not be the best values that GA-MCUOX can provide since neither a record of best C_{max} values are kept during the runs nor from among alternative chromosomes having the maximum NPV the ones are selected as best solutions that have the minimum C_{max} value. The incorporation of such details could cause a rise in the average percentage improvement in C_{max} values.

To test the hypothesis that GA-MCUOX would perform even better in minimizing makespan if this criterion is dealt with explicitly in it, GA-MCUOX is run on the same test suite with the same parameters as before, the only difference being that the fitness of a chromosome x in population P_t is computed this time by:

$$f(x)=z_{max}(P_t)-z(x), \qquad (4.4)$$

where $z_{max}(P_t)$ is the maximum makespan value observed in the current population P_t and $z(x)$ is the makespan value associated with x, i.e. $z(x)=C_{max}(x)$. The resulting GA-MCUOX yields an average percent improvement of 5.35 which is significant at a level of 0.0005 (see the last row in Table 4.3).

This fact that GA-MCUOX with the same set of parameters performs better than a domain specific heuristic in two different environments is an illustration of its robustness.

In this application, GA-MCUOX is implemented on an IBM-compatible personal computer with a pentium microprocessor of 100 MHz. Run times range from 1.00 to 1.50 minutes per replication for all the problems except for a few bigger problems. For those bigger problems, run times are in the range of 2.00 to 3.50 minutes per replication per problem.

TABLE 4.3. Results of the comparisons between GA-MCUOX and LCBA

(* indicates that f(x)=NPV(x) in GA-MCUOX;
^{s} indicates that f(x)= $z_{max}(P_t)$-z(x), where z(x)=C_{max}(x).)

	avg % impr	t-test
Comparison of NPV values *	18.37	6.17
Comparison of C_{max} values *	0.97	2.37
Comparison of C_{max} values ^{s}	5.35	7.51

4.4. Conclusion and Extensions

In this chapter, GA-MCUOX is used to attack the resource constrained project scheduling problems. It is shown that GA-MCUOX is applicable to both RCPSP and RCPSPDCF with a minor change in the definition of the fitness function.

GA-MCUOX proves to be a robust algorithm which outperforms one of the best problem specific heuristics in both RCPSP and RCPSPDCF.

Incorporation of nonrenewable resources will certainly make the problem more interesting; but this necessitates the addition of a repair routine to check for and restore the feasibility on each chromosome regarding the use of the nonrenewable resource within its availability limits. The repair routine is not likely to be a simple one in this case as it has to incorporate some heuristics to redistribute the usage of the nonrenewable resource among the activities. This in turn can lead to important modifications on the chromosome in terms of both mode selections and activity sequences.

Another interesting extension could be the incorporation of progress payments into RCPSPDCF. If this incorporation can be formulated as an additional selection problem, GA-MCUOX can be utilized as designed here without any modifications but the addition of a third dimension to the chromosome structure.

5. PARALLEL MACHINE SCHEDULING WITH EARLINESS AND TARDINESS PENALTIES

5.1. Problem Definition and Literature Survey

In the general parallel machine scheduling problem (PMSP), a set of independent jobs is to be scheduled on a number of parallel machines (processors) to meet an objective related to machine completion times. The most common objective is the minimization of the makespan.

The problem considered here is the scheduling of a set of independent jobs with given due dates on parallel machines so that total weighted earliness and tardiness penalties are minimized (PMSP_E/T). This objective is in compliance with the current trends towards just in time production strategy in which both early and tardy completion of jobs are undesirable since the former gives rise to inventory carrying costs and the latter to loss of customer goodwill and to tardiness penalties.

Baker and Scudder (1990) and Cheng and Gupta (1989) provide surveys on parallel machine scheduling with earliness and tardiness penalties. In the latter survey, due dates are considered as decision variables. As pointed out in the survey of Baker and Scudder, most of the models incorporating earliness and tardiness penalties (E/T models) are single-machine models involving a common due date for all jobs. The results indicate that when the due date is common for all jobs, the optimum schedule is V-shaped with no inserted idle time between the jobs. One job completes at the due date and one starts at the due date. Jobs that complete before the due date are in LPT sequence and jobs that complete after the due date are in SPT sequence.

Models with distinct due dates for each job are rather rare; and all of them are single machine models. When the due dates are distinct for each job, the optimum schedule may

contain idle times. Garey, Tarjan, and Wilfong (1988) prove that the one-processor scheduling problem with symmetric earliness and tardiness penalties is NP-complete. That is why studies with several simplifying assumptions (M=1, ready times for all jobs are zero, setup times are included in the processing times, and no idle time is inserted between the jobs) could attempt only small-sized problems involving maximally 25 jobs (Abdul-Razaq and Potts, 1988) and (Ow and Morton, 1989). More recent research has both relaxed the no-idle-time assumption and attempted single machine problems with up to 40 jobs (Yano and Kim, 1991). But still, the assumption that all jobs are available at the beginning of the scheduling period remains as an important deviation from reality.

When jobs are allowed to have also distinct arrival times, the problem becomes very complicated even for the single machine case. Nandkeolyar *et al.* (1993) propose a modular approach for the single machine case with equal weights for earliness and tardiness. Sridharan and Zhou (1996) develop a decision theory based procedure for a more general version with unequal earliness and tardiness weights. In their formulation, jobs have distinct arrival and processing times, due dates, and possibly unequal weights for earliness and tardiness.

The problem attacked here is similar but more difficult:

N independent jobs are given which have distinct arrival and processing times, and due dates. M machines are given which are one of two types, Type I and Type II. Machines belonging to the same type are identical whereas machines belonging to different types are uniform, i.e. they are similar but have different rates of processing. The setups are sequence dependent. Weights for earliness and tardiness penalties are common to all jobs; and they may well be unequal. The objective to minimize is given by

$$z = w_E \sum_{j \in J} E_j + w_T \sum_{j \in J} T_j \,. \tag{5.1}$$

where J denotes the set of jobs, w_E and w_T are the common earliness and tardiness weights respectively, $E_j = max\{0, d_j - C_j\}$ is the earliness and $T_j = max\{0, C_j - d_j\}$ is the tardiness of job j respectively with C_j being the completion time of job j. The notation used throughout this chapter is given in Figure 5.1 below.

5.2. GA-MCUOX for Parallel Machine Scheduling with Earliness and Tardiness Penalties

5.2.1. Representation and Operators

The problem is another instance of the generic MCCOP_SS. Here jobs are to be sequenced, and for each job, one of the machines is to be selected for processing. A small example chromosome is given by [4-1, 2-1, 5-2, 1-1, 3-2] for a problem with N=5 and M=2. Here, jobs 4, 2, and 1 are to be processed by machine 1 in the given order, and jobs 5 and 3 are to be processed by machine 2 in the given order.

For this problem, MCUOX is rephrased as follows: Starting from the first *jobs* on the parents, iteratively, one of the parents is chosen randomly and its next unconsidered *job* becomes the next *job* on the child. If the *machine assignment* for that *job* is the same on both parents, then that *assignment* is also made on the child; if not, one of the *machine assignments* of the parents is chosen randomly.

Two mutation operators are utilized: The swap mutation and the bit mutation. Fitness of chromosome x is given by

$$f(x) = z_{max}(P_t) - z(x), \qquad (5.2)$$

where z(x) is the objective function value associated with x and $z_{max}(P_t)$ is the maximum objective value observed in the current population P_t.

A random solution generator initializes the chromosomes at the very first generation. A chromosome is generated gene by gene by randomly selecting a job for each gene from among jobs not yet scheduled. To complete the gene, one of the *M* machines selected. The process is repeated until all *N* genes are completed.

N	number of jobs
M	number of machines
J	set of jobs, $J=\{1,..,N\}$
K	set of machines, $K=\{1,..,M\}$
rt_j	ready time of job j
d_j	due date of job j
p_j^k	processing time of job j on machine k
a_{ij}^k	sequence dependent setup time when job i precedes job j on machine k
p_j^{II} (a_{ij}^{II})	processing time (setup time) on a Type II machine
$\bar{p}^{II}$ $(\bar{a}^{II})$	average of processing (setup) times on the Type II machine
P_{max}	maximum processing time
A_{max}	maximum setup time
R_{max}	maximum ready time
w_E (w_T)	common weight for earliness (tardiness) penalties
$\lceil x \rceil$	greatest integer not larger than x.

FIGURE 5.1. The notation used in Chapter 5

5.2.2. Scheduling rules

The evaluation of the chromosomes, i.e. the generation of a schedule out of the sequence given on the chromosome and the determination of z(x) in the above formula, is a challenging issue in this problem. Three different scheduling rules are examined. One of these rules is the well-known *nondelay* scheduling rule. Given a job sequence to be scheduled on a machine, this rule simply schedules each job at its earliest start time. The other two rules are developed to reflect the shift in emphasis in the objective as w_E increases from 0 to 1.

A *forward-pass* scheduling rule is developed which schedules the jobs in a given sequence one by one, starting from the first job, so as to meet their due dates whenever possible. A job can be scheduled to complete at its due date if the preceding job completes before the implied start time of the job in question and there is enough time for the setup. If i precedes j in the sequence on machine k, then the implied start time of j, say $ST_j^{'}$, is given by $ST_j^{'} = d_j - pt_j^k$, and job j can complete at its due date if $ST_i + p_i^k + a_{ij}^k \leq ST_j^{'}$, where ST_i is the scheduled start time of job i. If this is not the case then job j is necessarily tardy.

The third scheduling rule is a *backward-forward-pass* scheduling rule which operates on the same principle as the forward-pass scheduler in that it tries to schedule jobs one by one to complete at their due dates whenever possible, but this time starting from the last job in the sequence. A job can be scheduled to complete at its due date if the succeeding job starts after the due date of the job in question and there is enough time for the setup inbetween. That is , if i precedes j in the sequence on machine k, then i can complete at its due date if $ST_j \geq d_i + a_{ij}^k$. Otherwise, job i is necessarily early. If job i is early, it needs to be checked whether it is scheduled earlier than its ready time. If so, job i is shifted right on the time axis until it starts at its ready time. All succeeding jobs are also checked one by one for possible interference of their processing with the processing of the immediate predecessor as a result of the shift in the starttime of that predecessor. If this is the case, the job in question is also shifted right by an

amount equal to the interference. The process continues until there is no interference in the next job to be considered. Figure 5.2 illustrates this right shifting process. For the purposes of illustration, setup times are ignored in Figure 5.2.

One important observation regarding the forward and backward-forward rules is that they are myopic. The first few jobs from the start (in case of the forward-pass rule and from the end in case of the backward-forward-pass rule) of the sequence can be scheduled to complete at their due dates while it becomes less and less probable that this happens for later (earlier) jobs in the sequence. This might well lead to suboptimal schedules since deliberately scheduling a job in the sequence a little early (tardy) even when it is perfectly possible to meet its due date may yield a much needed time span for the rest of the sequence to decrease the tardiness (earliness) values of some of the latter jobs. While this is especially important when the weights are distinct for each job, it is very hard to incorporate such a lookahead flavour into the scheduling rules. Instead, an extended genetic representation scheme is developed which incorporates that flavour by the aid of deliberate right and left shifts.

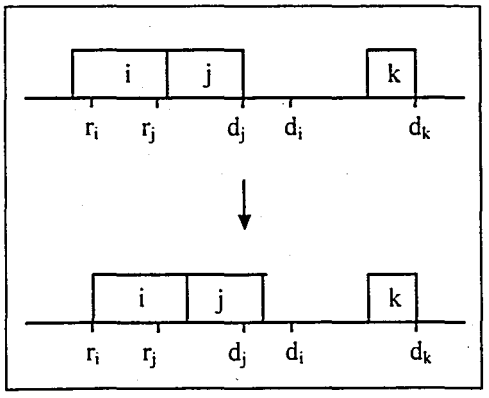


FIGURE 5.2. Illustration of the repair of ready time violations in the backward-forward scheduler by a right shift.

The new representation scheme has the same two dimensions of job sequencing and machine selection as before. A third dimension added to the gene indicates whether the job associated with the particular gene is to be scheduled at its ready time (or nondelay if this is not possible) or at its best start time so that it completes at its duedate. Assuming that, in this third dimension, a 0 indicates that the job in the corresponding locus is to be scheduled at its ready time (or nondelay) and a 1 indicates that the job is to be scheduled at its best start time, the example chromosome from Section 5.2.1 can be augmented to accomodate this third dimension as in Figure 5.3 below.

In this new scheme, an idle time might appear between two consecutive jobs one of which is scheduled at its ready time and the other at its best start time. If this is the case, the earliness of the job scheduled at its ready time can be decreased by shifting it to the right until its completion hits its own due date or the start time of the setup before the succeeding job. So, once all the jobs are scheduled according to the instructions carried by the third alleles, a forward swap is needed for possible right shifts.

locus	1	2	3	4	5
job sequence	4	2	5	1	3
machine selection	1	1	2	1	2
dispatch rule selection	0	1	1	0	1

b) An example chromosome with two selection components

[4-1,0 2-1,1 5-2,1 1-1,0 3-2,1]

c) The internal representation of the example chromosome

FIGURE 5.3. Example chromosome with three dimensions per locus

These four scheduling schemes are compared on the basis of their average performance on a test bed of five problems with N=50, M=4 and A_{max}=8. The GA used in the experiments has the following parameters: *popsize*=45, number of generations=300, pc=0.65, pm_{bit}=0.005, pm_{swap}=0.15, and generation gap G=0.06. The experiments are carried out for two cases where earliness penalties are more heavily weighted and visa versa. Two values for w_E are used to exemplify these cases, namely w_E=0.25 and w_E=0.75. The GA is replicated 10 times for each problem and each scheduling scheme.

Tables 5.1 and 5.2 represent average best solutions and their standard deviations for each problem. From the tables it follows that if earliness is weighted more heavily than tardiness, the forward-pass scheduler performs quite nicely; whereas if tardiness is more heavily weighted, then the backward-forward pass scheduler is better. So these rules are utilized in the numerical study. The 3-dimensional representation scheme performs better when w_E=0.25 as compared to the case when w_E=0.75; and the opposite is true for the nondelay scheduling rule.

5.2.3. Parameter Finetuning

Meta-GAs are developed and used to fine tune the parameters of two GAs with the forward and backward-forward schedulers separately. These parameters and the range of values investigated are listed in Table 5.3 in the order in which they appear on the chromosomes.

From the table it follows that the meta-GA is to search through a space of more than 2^{24} possible parameter sets. The parameters of the meta-GA itself are as follows: population size is 51, probability of crossover is 0.65, and probability of bit mutation is 0.05. The problem used in the finetuning process is a problem with N=50, M=4 and A_{max}=8 which is denoted by 50481 in Tables 5.1 and 5.2.

TABLE 5.1. Average best solution values (standard deviations) of 10 replications for w_E=0.25. ('*' marks the best value for each problem)

PROBLEM	NONDELAY	FORWARD	BACKWARD	3-D
50481	242.47	518.03	161.47*	224.72
	(69.16)	(110.78)	(66.35)	(87.56)
50482	220.37	432.72	182.11*	312.55
	(74.01)	(278.55)	(78.17)	(46.73)
50483	266.77	545.51	162.27*	349.30
	(116.47)	(110.84)	(59.36)	(145.02)
50484	257.26	633.71	218.92*	321.63
	(87.82)	(118.85)	(103.82)	(118.64)
50485	245.29	568.30	187.44*	233.61
	(75.93)	(162.56)	(108.68)	(83.42)

TABLE 5.2. Average best solution values (standard deviations) of 10 replications for w_E=0.75. ('*' marks the best value for each problem)

PROBLEM	NONDELAY	FORWARD	BACKWARD	3-D
50481	287.87	119.07*	247.07	152.88
	(33.50)	(37.82)	(21.90)	(38.25)
50482	310.29	126.37*	256.80	196.18
	(28.28)	(61.46)	(51.87)	(48.64)
50483	289.05	163.36*	278.71	187.80
	(42.78)	(32.56)	(28.56)	(44.93)
50484	290.01	185.82	250.59	181.78*
	(40.41)	(44.58)	(25.33)	(61.95)
50485	329.87	182.54*	257.85	192.15
	(37.45)	(32.69)	(48.88)	(36.63)

The evaluation of a chromosome in the meta-GA is done by running GA-MCUOX with the parameters specified in the chromosome as many times as specified by the *nRep* parameter. The payoff value is then taken to be the average of best solution values of all replications. Fitness of the chromosome X is measured by how much it deviates from the worst performing chromosome in the current population, i.e.

$$F(X) = \bar{Z}_{max} - \bar{Z}(X), \tag{5.3}$$

where $\bar{Z}(X)$ is the average of the best solution values obtained using the parameters in X in *nRep* replications and $\bar{Z}_{max}$ is the maximum average best solution value observed in the current population.

TABLE 5.3. Parameters and their range of values fine-tuned in the meta-GAs

Parameter	Identifier	Values
population size	*popsize*	20,30,40,..,120
number of generations	*nGen*	20,40,60,..,400
number of replications	*nRep*	5,8,10,12,15,18,20
probability of crossover	*pc*	0, .2, .4, .5, .6, .65, .7, .75,..,1
probability of bit mutation	pm_{bit}	0, .01, .025, .05, .075, .1, .15, .2, .25, .3, .4, .5
probability of swap mutation	pm_{swap}	0, .1, .2, .3, ..,1
generation gap	*G*	0/*popsize*,..,10/*popsize*

The meta-GA to fine-tune the parameters of the GA with the forward-pass scheduler is run for 53 generations and the meta-GA for the backward-forward-pass scheduler for 45

generations. As a result of these meta-GA runs, the parameters for the GA to be used with the forward-pass scheduler (in the case of $w_E > w_T$) are fixed as follows: *popsize*=90, *nGen*=380, *nRep*=10, *pc*=0.65, pm_{bit} =0.01, pm_{swap}=0.40, and *G*=9/90=0.10. And the parameters for the GA to be used with the backward-forward-pass scheduler (in the case of $w_E < w_T$) are determined to be as follows: *popsize*=100, *nGen*=400, *nRep*=5, *pc*=0.40, pm_{bit} =0.01, pm_{swap}=0.60, and *G*=8/100=0.08.

5.3. Numerical Study

5.3.1. Generation of Test Problems

Since this is the first time that this problem is being attacked and thus there are no benchmark problems in the literature, problems are generated randomly. It has been assumed that machines belong to one of two different types which have the same characteristics except that they have different processing rates. Type II machines represent an older technology. The processing time of job *j* on a Type II machine is 10-20 percent larger than its processing time on a Type I machine. Similarly, setup times on a Type II machine are 20-40 percent larger than the corresponding setup times on a Type I machine.

Processing times on the Type I machine are U[4,20]. To generate the processing time of job *j* on the Type II machine, a multiplier is chosen randomly from [1.10,1.20] and is applied to the processing time of job *j* on the Type I machine.

Setup times on Type I machines are U[1,A_{max}] where two levels of A_{max} is utilized in this study. Again a multiplier chosen from [1.20,1.40] is employed to compute the setup times on the Type II machine. Ready times are U[0,R_{max}]. Here,

$$R_{\max} = \left\lceil \left(\bar{p}^{II} + \bar{a}^{II} \right) \left(\frac{N}{M} - 1 \right) \right\rceil . \tag{5.4}$$

Due date of job j is taken to be the sum of its ready time, processing time on the Type II machine, maximum time to setup a Type II machine for the processing of job j and a slack value. A number of preliminary experiments with N=50, M=4, and A_{max}=8 are carried out to find a suitable formula for slack. In particular,

$$slack = k * \left(\bar{p}^{II} + \bar{a}^{II} \right), \text{ and} \tag{5.5}$$

$$slack = k * \left(P_{\max} + A_{\max} \right) \tag{5.6}$$

with k=1, 2 are tried. In this study, P_{max} is 20. It has been sought that meaningful earliness and tardiness values result for each job so that the problems be not trivial. As a result of this experimentation, slack is chosen to be the sum of the mean values of processing and setup times on a Type II machine; i.e. $slack = \bar{p}^{II} + \bar{a}^{II}$.

Due dates are generated as follows:

$$d_j = r_j + \max_i a_{ij}^{II} + p_j^{II} + slack \ . \tag{5.7}$$

The test problems are generated using the following design:

N	20, 40, 60
M	2, 4
A_{max}	4, 8

It is assumed that half of the machines belong to Type I and the other half to Type II.

Twenty instances are generated for each combination of the design parameters. Each one is run for four different levels of weights for earliness penalties: 0.00, 0.25, 0.75, and 1.00. Thus the test bed is comprised of 960 problems.

5.3.2. Mathematical Programming Models

To provide a comparison basis for the performance of GA-MCUOX, it is attempted to optimally solve the problems in the test suite (in the hope that at least the ones with N=20 would be solvable). For this purpose, two mathematical models are developed which are presented in Appendix A and B. Both of the models are mixed integer programming models. GAMS has been utilized to solve the models.

The first model denoted by Model I involves $3N$ positive variables and $(N+1)^2M$ binary variables. The second model, Model II, involves $2N+NM(N-M+1)$ positive variables and $NM(N-M+1)$ binary variables.

5.3.3. Results

GAMS experiments on models with $N \leq 6$ has shown that Model I is more efficient than Model II as it can be solved in a smaller number of iterations as compared to Model II. The difference in the number of iterations can be an order of magnitude. This is to be expected since Model I contains a smaller number of binary variables and constraints as compared to Model II.

Except for the verification of this expected difference in efficiency between the two models, GAMS runs have not been very fruitful. Runtimes explode when N is increased from trivial to more meaningful levels let aside to the levels in the test suite. For example, GAMS is able to solve problems with N=6 or less within a reasonable time period (i.e. within a couple of hours); but when N is increased to 10, it takes prohibitively long to solve the models. With N=10, runs are cut off after 96 hours of run time, at which point GAMS has not even found a feasible integer solution.

GA-MCUOX is implemented on an IBM-compatible personal computer with a pentium microprocessor of 100 MHz. On the same set of problems with N=10, run times of GA-MCUOX are on the average 9 seconds for w_E=0.00, 17 seconds for w_E =0.25, 16 seconds for w_E =0.75 and 0 seconds for w_E=1.00 per one run of the GA on one problem.

When w_E =1.00 the optimum value of zero is found right away in the first generation. This is to be expected since the forward scheduler schedules jobs so that they are either on time or tardy and thus provides the optimum schedule for any sequence. When w_E =0.00, for the majority of problems, the runs result in an objective value of zero but this takes on the average longer: 6 seconds per replication on a problem with M=4 and 11.4 seconds per replication on a problem with M=2. In general, and as expected, problems with M=4 are easier to solve than problems with M=2.

When N is increased to 20, for example, the GAMS models also explode in the number of binary variables. GA-MCUOX, on the other hand, is not bothered much. Runtimes range on

the average from 0.9 seconds per replication to 4.2 minutes per replication over all levels of M, A_{max} and w_E. For N=40, the range of run times is 2 seconds to 8.3 minutes per replication, and for N=60 it is 3 seconds to 13.8 minutes per replication.

5.3.4. Comparing GA-MCUOX with GA-PMX

As the mathematical models have not been helpful in providing a comparison basis for the performance of GA-MCUOX, a GA with another crossover operator is developed. The crossover operator chosen is again the well-known partially mapped crossover (PMX) operator of Goldberg and Lingle (1985).This choice is not only due to the popularity of PMX but also due to the fact that it is the first and only sequencing operator that is presented to be able to "exchange both ordering and value information among different strings" (Goldberg and Lingle, 1985). The approach proposed it to tag on the value information to the corresponding ordering information and to apply PMX to this composition. This means that the object-selection pairs are to be treated as one single unit both in representation and during evolution.

GA-PMX is developed following this suggestion. Its parameters are finetuned for the two cases of w_E<0.50 and w_E>0.50 with the forward and backward-forward schedulers respectively using the same meta-GAs and the same test problem that were used for the finetuning of the parameters of GA-MCUOX (see Section 5.2.3). The resulting parameter sets are first surprising but they are also to be expected:

The parameters for GA-PMX to be used with the forward-pass scheduler (in the case of w_E>w_T) are determined to be as follows: *popsize*=110, *nGen*=400, *nRep*=18, *pc*=0.00, pm_{bit} =0.01, pm_{swap}=0.60, and G=9/110=0.08. And the parameters for GA-PMX to be used with the backward-forward-pass scheduler (in the case of w_E<w_T) are determined to be as follows: *popsize*=100, *nGen*=400, *nRep*=10, *pc*=0.00, pm_{bit} =0.01, pm_{swap}=0.60, and G=9/100=0.09.

The surprising part is that in both cases *pc* is evolved to the value of 0.00; in other words, the crossover operator PMX is turned off. The other parameters are more or less similar to their counterparts in GA-MCUOX the biggest difference being in the doubling of the number of replications. The results are to be expected as by treating the two dimensions of the problem as a single unit, PMX can not accomplish an effective search especially in the selection dimension. Overall, the attachment of the selection component to the ordering component hinders an effective search by PMX. PMX is easily replaced by a mutation operator, and GA-PMX becomes a mutation-only-GA.

GA-PMX with pc=0.00 (or the mutation-only-GA) is still run on the test suite generated (Section 5.3.1). This provides an opportunity to see the effectiveness of GA-MCUOX as compared to a separately fine-tuned mutation-only-GA on PMSP_E/T. Table 5.4 presents the average percent deviations of the best and average solution values found by GA-MCUOX from those found by GA-PMX together with the standard deviations and the t-test values. '*' indicates that some solution values are zero and the given figures are averages rather than percentages. The figures for w_E=1.00 are not listed as the optimum values are found by both algorithms in all problems. 'score' gives the number of problems among 20 for which GA-MCUOX result is better than that provided by the mutation-only-GA.

From Table 5.4 it follows that the mutation-only-GA performs better than GA-MCUOX for small sized problems with N=20, but it is outperformed by GA-MCUOX on larger problems with N=40 and N=60. Moreover, the performance of GA-MCUOX improves with increasing values for N and decreasing values for M. This means that the recombinative power of GA-MCUOX becomes increasingly important as problems increase in size and difficulty.

TABLE 5.4. Comparison of GA-MCUOX with mutation-only-GA.

				bestZ			avgZ		
N	w_E	M	A_{max}	score	avg%dv	t-test	score	avg%dv	t-test
20	0.00	2	4	8	-0.60*	0.28	11	-46.46	1.37
		2	8	2	-3.89*	2.71	12	-21.78	0.54
		4	4	3	-25.33*	3.17	8	-31.32	1.02
		4	8	2	-30.79*	2.54	11	-20.85	0.17
	0.25	2	4	8	-4.28	1.33	12	-12.78	0.88
		2	8	10	-0.94	0.82	10	-4.51	0.64
		4	4	5	-77.24	3.83	5	-18.77	2.50
		4	8	2	-126.73	5.11	8	-23.50	1.56
	0.75	2	4	8	0.12	0.65	7	-7.94	1.90
		2	8	12	2.20	1.27	11	-0.75	0.62
		4	4	3	-16.26	3.13	2	-34.66	7.46
		4	8	5	-38.76	3.47	0	-47.78	7.70
40	0.00	2	4	17	30.27	5.61	20	35.01	7.99
		2	8	18	34.64	5.27	20	41.84	10.66
		4	4	15	4.09*	1.05	11	-30.71	0.15
		4	8	15	7.90*	1.72	9	-32.20	0.67
	0.25	2	4	19	17.83	5.82	20	29.69	9.80
		2	8	18	20.22	5.14	20	31.40	6.28
		4	4	11	-3.18	0.69	11	-21.77	0.97
		4	8	12	4.06	1.27	10	-26.54	0.41
	0.75	2	4	18	13.79	6.08	20	20.55	11.04
		2	8	17	19.56	7.18	20	26.91	12.34
		4	4	12	3.12	1.62	5	-19.03	2.52
		4	8	12	3.64	1.39	9	-19.46	1.58

TABLE 5.4. (Comparison of GA-MCUOX with mutation-only-GA) continued.

N	w_E	M	A_{max}	bestZ score	bestZ avg%dv	bestZ t-test	avgZ score	avgZ avg%dv	avgZ t-test
60	0.00	2	4	20	50.35	12.13	20	49.36	15.48
		2	8	20	59.49	13.16	20	54.48	17.03
		4	4	18	40.84	7.00	19	46.26	7.68
		4	8	18	32.93	4.80	20	50.51	8.31
	0.25	2	4	20	45.14	16.83	20	46.97	24.23
		2	8	20	47.55	13.69	20	49.86	16.03
		4	4	19	27.26	6.03	20	40.31	10.77
		4	8	17	21.09	5.86	18	27.97	4.20
	0.75	2	4	20	39.46	18.12	20	36.39	27.67
		2	8	20	37.98	12.89	20	42.56	29.33
		4	4	18	17.69	5.36	20	24.21	9.89
		4	8	20	23.88	8.36	16	15.95	2.92

5.4. Conclusions

In this chapter, GA-MCUOX is applied to a very difficult parallel machine scheduling problem with earliness and tardiness penalties. The problem is attacked for the first time in literature and therefore benchmark problems are not available. Mathematical models are developed in an effort to optimally solve the test problems generated. But it turned out that the employed optimization package GAMS was not able to attack even very small problems of half the size of the smallest problems in the test suite. On the test suite generated, GA-MCUOX is compared to GA-PMX which turned into a specialized mutation-only-GA as a

result of the parameter finetuning process. The results suggest that the performance of GA-MCUOX improves drastically with increasing problem size and difficulty while its run times increase at a decreasing rate. This makes GA-MCUOX an attractive algorithm for applications of ever larger sizes.

6. CONCLUSION

In this dissertation, a new uniform order-based crossover operator is presented. The new operator, referred to as MCUOX, is a general and powerful tool which can be used in genetic algorithm (GA) applications to multi-component combinatorial optimization problems with sequencing and selection components (MCCOP_SS). Such multi-component formulations are important as they more accurately reflect the reality as opposed to traditional formulations considering only the sequencing component. Indeed, MCCOP_SS comprise a wide range of problems from various fields such as production scheduling, network and circuit design.

The important characteristic of MCCOP-SS from the view point of GA applications is that they incorporate ordering and value information at the same time. The general approach to such problems in GA-literature has been to employ a different set of operators for each dimension of the problem. Some applications have considered all the components simultaneously by the aid of problem specific operators which usually also incorporate local hill climbing heuristics. Either approach is problem dependent, and the set of operators designed or employed in one application is not likely to be applicable in a different problem environment. This in turn is against the quest for robustness in genetic search.

MCUOX provides a resolution to this dilemma by being both applicable to all instances of MCCOP-SS and capable of recombining partial solutions to obtain optimum or near-optimum solutions.

An interesting characteristic of MCUOX is that it propagates schemata with defined positions located towards either or both ends of the chromosome better than others. This gives rise to a double-sided convergence of alleles.

In this work, applications of a generic GA with MCUOX denoted by GA-MCUOX on the following three different and inherently difficult instances of

MCCOP-SS are discussed in detail: simultaneous scheduling of machines and automated guided vehicles, resource constrained project scheduling with discounted cash flows and parallel machine scheduling with earliness and tardiness penalties. On all these problems, GA-MCUOX performed well. It has been able to locate optima or near-optima within reasonable run times and has outperformed problem specific heuristic procedures from literature.

In one of these applications, namely in the problem of simultaneous scheduling of machines and automated guided vehicles (AGVs), there is a relatively small amount of redundancy present in the representation. This redundancy is caused by the fact that the alternatives in the selection problem, i.e. the AGVs, are identical. This means that the same information is represented on two chromosomes with the same ordering of operations but with the identification numbers of the AGVs exchanged. Despite the redundancy, GA-MCUOX has performed very well on this problem; and further experimentation with a pure grouping problem from literature has shown that GA-MCUOX is able to locate near-optimum solutions even in the presence of an immense amount of redundancy in representation.

In summary, the new operator MCUOX

- fills a gap in GA-research by processing ordering and value information concurrently;
- is general as it is applicable to every instance of MCCOP_SS;
- operates within a natural and direct chromosomal representation of MCCOP_SS;
- is compact, aesthetic and convenient in that it encompasses all the dimensions of the problem in one operator and eliminates the need for several crossover operators;
- is able to search every dimension of the problem separately;
- respects precedence relations and eliminates the need for repair algorithms;

- is able to handle additional selection components at virtually no cost;
- effectively recombines partial solutions presented in the population;
- enhances building block formation and propagation especially in problems where the ordering has a temporal meaning.

Concurrent consideration of ordering and value information in genetic operators is an untouched area of research. Yet the combination of ordering and value is much more interesting than either component alone both from a theoretical and a practical point of view. Future research can concentrate on this topic.

Another topic for future research is the multi-parent and/or multi-offspring implementations of MCUOX. Such implementations should better be considered in conjunction with large population sizes which counterbalance the exploitative pressures caused by multi-parent/offspring implementations.

Finally, an interesting topic to further investigate comprises the potential advantages of the presence of redundancy in the representation scheme.

APPENDIX A - MATHEMATICAL MODEL I

In the first mathematical model (Model I) for PMSP_E/T, there are three positive variables and two binary variables. They are defined as follows:

ST_j: start time of job j

E_j: earliness of job j

T_j: tardiness of job j

δ_{jk}: $\begin{cases} 1 & \text{if job j is processed on machine k} \\ 0 & \text{o / w} \end{cases}$

σ_{ijk}: $\begin{cases} 1 & \text{if job i precedes job j on machine k} \\ 0 & \text{o / w} \end{cases}$

The model is given below. Job 0 is defined to be a dummy job with zero ready and processing times and due date. *BIG* is a big number.

$$\text{min} \quad z = w_E \sum_{j \in J} E_j + w_T \sum_{j \in J} T_j + BIG * ST_0 \tag{A.0}$$

$$\text{st} \quad E_j \geq d_j - \sum_{k \in K} \delta_{jk} p_j^k - ST_j \quad , \quad j \in J \tag{A.1}$$

$$T_j \geq ST_j + \sum_{k \in K} \delta_{jk} p_j^k - d_j \quad , \quad j \in J \tag{A.2}$$

$$ST_j \geq \sigma_{ijk} \left(ST_i + p_i^k + a_{ij}^k \right) \quad , \quad i, j \in J;\ i \neq j;\ k \in K \tag{A.3}$$

$$ST_j \geq rt_j \quad , \quad j \in J \tag{A.4}$$

$$\delta_{jk} = \sum_{\substack{i \in J + \{0\} \\ i \neq j}} \sigma_{ijk} \quad , \quad j \in J,\ k \in K \tag{A.5}$$

$$\delta_{ik} \geq \sum_{\substack{j \in J \\ j \neq i}} \sigma_{ijk} \quad , \quad i \in J + \{0\},\ k \in K \tag{A.6}$$

$$\sum_{k \in K} \delta_{jk} = 1 \quad , \quad j \in J \tag{A.7}$$

$$\sum_{\substack{j \in J \\ j \neq i}} \sum_{k \in K} \sigma_{ijk} \leq 1 \quad , \quad i \in J \tag{A.8}$$

$$\sum_{j \in J} \sigma_{0jk} = 1 \quad , \quad i \in J,\ k \in K \tag{A.9}$$

$$ST_j,\ E_j,\ T_j,\ Y_{ijk} \geq 0 \quad , \quad j \in J \tag{A.10}$$

$$\delta_{jk},\ \sigma_{ijk} = \{0,1\} \quad , \quad i \in J + \{0\},\ j \in J,\ i \neq j,\ k \in K \tag{A.11}$$

Constraints (A.1) and (A.2) define earliness and tardiness of job j. Constraints (A.3) assure that jobs do not start before the preceding job is completed and the required setup is accomplished while (A.4) assures that no job starts before its ready time. Constraints (A.5) and (A.6) are needed to build the relations between the selection and sequencing components. (A.7) , (A.8) and (A.9) are the familiar assignment constraints. Before the model can be solved by an MIP solver in GAMS, the nonlinear term in (A.3) needs to be linearized. For this purpose, a new variable Y_{ijk} replaces the term $\sigma_{ijk}*ST_i$ in constraints (A.3), and the following three constraints are added to the model:

$$Y_{ijk} - BIG * \sigma_{ijk} \leq 0 \quad , \quad i \in J + \{0\}, j \in J, i \neq j,\ k \in K \tag{A.12}$$

$$Y_{ijk} - ST_i \leq 0 \quad , \quad i \in J + \{0\}, j \in J,\ i \neq j,\ k \in K \tag{A.13}$$

$$ST_i - Y_{ijk} + BIG * \sigma_{ijk} \leq BIG \quad , \quad i \in J + \{0\}, j \in J,\ i \neq j,\ k \in K \tag{A.14}$$

APPENDIX B - MATHEMATICAL MODEL II

The second model (Model II) for PMSP_E/T involves a different formulation of the problem. The variables are:

ST_{jrk}: start time of job j if it is the r'th job on machine k

E_j: earliness of job j

T_j: tardiness of job j

γ_{jrk}: $\begin{cases} 1 & \text{if job j is r' th on machine k} \\ 0 & \text{o / w} \end{cases}$

In this formulation, r is the index indicating the order of job j on machine k, and it takes values in $R=\{1,...,|R|\}$, where $|R|=N-M+1$. The model is as follows:

$$\text{min} \quad z = w_E \sum_{j \in J} E_j + w_T \sum_{j \in J} T_j \tag{B.0}$$

$$\text{st} \quad E_j \geq d_j - \sum_{k \in K} \left(\sum_{r \in R} \gamma_{jrk} \right) p_j^k - \sum_{r \in R} \sum_{k \in K} ST_{jrk} \quad , \quad j \in J \tag{B.1}$$

$$T_j \geq \sum_{r \in R} \sum_{k \in K} ST_{jrk} + \sum_{k \in K} \left(\sum_{r \in R} \gamma_{jrk} \right) p_j^k - d_j \quad , \quad j \in J \tag{B.2}$$

$$ST_{jrk} \geq \gamma_{jrk} \left(ST_{i(r-1)k} + \gamma_{i(r-1)k} (p_i^k + a_{ij}^k) \right), \quad i, j \in J;\ i \neq j;\ r \in R,\ r \geq 2,\ k \in K \tag{B.3}$$

$$ST_{j1k} \geq \gamma_{j1k} a_{0j}^k \quad , \quad j \in J,\ k \in K \tag{B.4}$$

$$\sum_r \sum_k \gamma_{jrk} = 1, \qquad j \in J \tag{B.5}$$

$$\sum_j \gamma_{j1k} = 1, \qquad k \in K \tag{B.6}$$

$$\sum_J \gamma_{jrk} \le \sum_J \gamma_{j(r-1)k}, \qquad r \in R,\ r \ge 2,\ k \in K \tag{B.7}$$

$$ST_{jrk} - \mathrm{UB}_{ST}\gamma_{jrk} \le 0, \quad j \in J,\ r \in R,\ k \in K \tag{B.8}$$

$$ST_{jrk} - \mathrm{rt}_j\gamma_{jrk} \ge 0, \quad j \in J,\ r \in R,\ k \in K \tag{B.9}$$

$$ST_{jrk},\ E_j,\ T_j \ge 0, \quad j \in J \tag{B.10}$$

$$\gamma_{jrk} = \{0,1\}, \qquad j \in J,\ r \in R,\ k \in K \tag{B.11}$$

UB_{ST} is an upper bound on the ST_{jrk}. The constraints (B.1), (B.2) and (B.3) are similar to the first three constraints of the first model. Constraints (B.4) consider first jobs in the sequence on each machine. Constraints (B.5) and (B.6) are assignment constraints, while (B.7) are needed to assure that the positions on each machine are filled starting from the first one. Constraints (B.8) and (B.9) establish the relations between the variables ST_{jrk} and γ_{jrk} so that one of them can take a value greater than zero if and only if the other one does also.

Two nonlinear terms are present in constraints (B.3). The first one, namely $\gamma_{jrk}ST_{i(r-1)k}$, is replaced by X_{ijrk}; and the second one, $\gamma_{jrk}\gamma_{i(r-1)k}$, is replaced by Y_{ijrk}. The following two sets of three constraints are added to the model:

$$X_{ijrk} - UB_{ST}\gamma_{jrk} \le 0, \qquad i,j \in J,\ i \ne j,\ r \in R,\ k \in K \tag{B.12}$$

$$X_{ijrk} - ST_{i(r-1)k} \le 0, \qquad i,j \in J,\ i \ne j,\ r \in R,\ r \ge 2,\ k \in K \tag{B.13}$$

$$ST_{i(r-1)k} - X_{ijrk} + UB_{ST}\gamma_{jrk} \le UB_{ST}, \quad i,j \in J,\ i \ne j,\ r \in R,\ r \ge 2,\ k \in K \tag{B.14}$$

$$Y_{ijrk} - \gamma_{jrk} \le 0, \qquad i,j \in J, i \ne j,\ r \in R,\ k \in K \tag{B.15}$$

$$Y_{ijrk} - \gamma_{i(r-1)k} \le 0, \qquad i,j \in J, i \ne j,\ r \in R,\ r \ge 2,\ k \in K \tag{B.16}$$

$$\gamma_{jrk} + \gamma_{i(r-1)k} - Y_{ijrk} \le 1, \quad i,j \in J, i \ne j,\ r \in R,\ r \ge 2,\ k \in K \tag{B.17}$$

REFERENCES

Abdul-Razaq, T.S., and C.N. Potts, "Dynamic programming state-space relaxation for single machine scheduling", *Journal of the Operational Research Society*, Vol.39, pp.141-152, 1988.

Baker, K.R., and G.D. Scudder, "Sequencing with earliness and tardiness penalties: A review", *Operations Research*, Vol.38, pp.22-36, 1990.

Bilge, Ü. and G. Ulusoy, "A time window approach to simultaneous scheduling of machines and material handling system in an FMS", *Operations Research* , Vol.43, pp.1058–1070, 1995.

Blazewicz, J., J.K. Lenstra, and A.H.G. Rinnoy Kan, "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol.5, pp.11-24, 1983.

Booker, L.B., D.E. Goldberg, and J.H. Holland, "Classifier systems and genetic algorithms", *Artificial Intelligence*, pp.235-282, 1989.

Bruns, R., "Direct chromosome representation and advanced genetic operators for production scheduling", *Proceedings of Fifth International Conference on Genetic Algorithms*, S. Forest (ed.), Morgan Kaufmann, Fairfax, Virginia, pp.352–359, 1993.

Cerny, V., "Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, Vol. 45, pp. 41-52, 1985.

Cheng, T.C.E, and M.C. Gupta, ""Survey of scheduling research involving due date determination decisions", *European Journal of Operational Research*, Vol.38, pp.156-166, 1989.

Cox, Jr., L.A., L. Davis, L.L. Lu, D. Orvosh, X. Sun, and D. Sirovica, "Reducing costs of backhaul networks for PCS networks using genetic algorithms", *Journal of Heuristics*, Vol.2, pp.201-216, 1996.

Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

De Jong, K.A., "An analysis of the behavior of a class of genetic adaptive systems", Ph.D. dissertation, University of Michigan, *Dissertation Abstracts International*, 36(10), 5140B. University Microfilms No. 76-9381, 1975.

De Jong, K.A., "Genetic algorithms are NOT function optimizers", in L.D. Whitley (ed.), *Foundations of Genetic Algorithms-2*, pp.5-18, Morgan Kaufmann, San Mateo, CA, 1993.

Drechsler, R., B. Becker, and N. Göckel, "A genetic algorithm for the construction of small and highly testable OKFDD-circuits", *Proceedings of the First Genetic Programming Conference*, July 28-31, Stanford University, pp.473-478, 1996a.

Drechsler, R., personal communication, 1996b.

Elmaghraby, S.E., and W.S. Herroelen, "The scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, Vol.49, pp.35-40, 1990.

Falkenauer, E., "Solving equal piles with the grouping genetic algorithm", *Proceedings of the Sixth International Conference on Genetic Algorithms,* Eshelman, L.J. (ed.), pp. 492-497, San Mateo, CA: Morgan Kaufmann, 1995.

Fang, H–L. , P. Ross, D. Corne, "A promising GA approach to job–shop scheduling, rescheduling and open–shop scheduling problems", *Proceedings of Fifth International Conference on Genetic Algorithms*, S. Forest (ed.), Morgan Kaufmann, Fairfax, Virginia, pp.375–382, 1993.

Fennel, T.R., A.J. Underbrink, Jr., and G.P.W. Williams, Jr., "Scheduling with genetic algorithms", in *Proceedings of the Third International Symposium on AI, Robotics and Automation for Space, i-SAIRAS 94*, 1994.

Filho, J.L.R., P.C. Treleaven and C. Alippi, "Genetic-algorithm programming environments", *IEEE Computer*, pp.28-43, 1994.

Fogel D.B., *Evolutionary Computation, Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.

Fogel, L.J., "Autonomous automata", *Industrial Research*, Vol. 4, pp. 14-19, 1962.

Fogel, L.J., *Biotechnology: Concepts and Applications*, Prentice Hall , Englewood Cliffs, NJ, 1963.

Fogel, L.J., A.J. Owens and M.J. Walsh , *Artificial Intelligence through Simulated Evolution*, John Wiley, NY, 1966.

Garey, M.R., R.E. Tarjan, and G.T. Wilfong, "One-processor scheduling with symmmetric earliness and tardiness penalties", *Mathematics of Operations Research*, Vol.13, pp.330-348, 1988.

Glover, F., "Heuristics for integer programming using surrogate constraints", *Decision Sciences*, Vol. 8, pp. 156-166, 1977.

Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers and OR*, Vol. 13, pp. 533-549, 1986.

Glover, F., "Tabu search - part 1", *ORSA Journal on Computing*, Vol. 1, pp. 190-206, 1989.

Glover, F. and H. J. Greenberg, "New approaches for heuristic search: A bilateral linkage with artificial intelligence", *European Journal of Operations Research*, Vol.39, pp.119-130, 1989.

Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, 1989a.

Goldberg, D.E., "Zen and the art of genetic algorithms", *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), Morgan Kaufmann, pp.80-85, 1989b.

Goldberg, D.E., "Sizing populations for serial and parallel genetic algorithms", *Proceedings of the Third International Conference on Genetic Algorithms* , D.J. Schaffer (ed.), pp.70–79, Morgan Kaufmann, Fairfax, Virginia, 1989c.

Goldberg, D.E., K.Deb, H. Kargupta, and G. Harik, "Rapid, accurate optimization of difficult problems using fast messy genetic algorthms", *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forest (ed.), Morgan Kaufmann, Fairfax, Virginia, pp.56-64, 1993.

Goldberg, D.E., B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results", *Complex Systems*, Vol.3, pp.493-530, 1989.

Goldberg, D.E., and R.L. Lingle, "Alleles, loci, and the traveling salesman problem", *Proceedings of the First International Conference on Genetic Algorithms*, Erlbaum, pp.154-159, 1985.

Goldberg, D.E., and P. Segrest, "Finite Markov chain analysis of genetic algorithms", *Proceeding of the Second International Conference on Genetic Algorithms*, pp.1-8, Cambridge, Massachusetts, 1987.

Grefenstette, J.J., "Optimization of control parameters for genetic algorithms", *IEEE Transactions on Systems Man and Cybernetics*, Vol. SMC-16, pp. 122-128, 1986.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, 2nd edition, The MIT Press, 1992a.

Holland, J.H., "Genetic Algorithms", *Scientific American*, pp.44-50, July 1992b.

Holland, J.H., *Hidden Order; How Adaptation Builds Complexity*, Addison-Wesley, 1995.

Holland, J.H., personal communication, 1997.

Holsapple, G.W., V.S. Jacop, R. Pakath, and J.S. Zaveri, "A genetics based hybrid scheduler for generating static schedules in flexible manufacturing contexts", *IEEE Trans. Systems Man and Cybernetics*, Vol. SMC–23, pp. 953–972, 1993.

Jones, D.R. and M.A. Beltramo, " Solving partitioning problems with genetic algorithms", *Proceedings of the Fourth International Conference on Genetic Algorithms,* Belew, R.K. and Booker, L.B. (eds), pp. 442-449, San Mateo, CA: Morgan Kaufmann, 1991.

Kirkpatrick, S., C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing", *Science*, Vol. 220, pp. 671-680, 1983.

Koza, J.R., *Genetic Programming: On The Programming of Computers By Means of Natural Selection*, The MIT Press, Cambridge, MA, 1992.

Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Holt, Reinhart and Winston, New York, 1976.

Lee, I., R. Sikora, and M.J. Shaw, "Joint lot sizing and sequencing with GAs for scheduling: Evolving the chromosome structure", *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forest (ed.), Morgan Kaufmann, Fairfax, Virginia, pp.383–389, 1993.

Li, K.Y., and R.J. Willis, "An iterative scheduling technique for resource-constrained project scheduling", *European Journal of Operational Research*, Vol.56, pp.370-379, 1992.

McCulloch, W.S., and W. Pitts, "A logical calculus of ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.

Mitchell, M., *An Introduction to Genetic Algorithms*, The MIT Press, 1996.

Mueller-Merbach, H., "Heuristics and their design: A survey", *European Journal of Operational Research* , Vol. 8, pp. 1-23, 1981.

Nandkeolyar, U., M.U. Ahmed, and P.S. Sundararaghavan, "Dynamic single machine weighted absolute deviation problem: Predictive heuristics and evaluation", *International Research of Production Research*, Vol.31, pp.1453-1466, 1993.

Ng, K.P., and K.C. Wong, "A new diploid scheme and dominance change mechanism for non-stationary function optimization", *Proceedings of the Sixth International Conference on Genetic Algorithms*, L.J. Eshelman (ed.), Morgan Kaufmann, Pittsburgh, Pennsylvania, pp.383–389, 1995.

Ow, P.S., and T.E. Morton, "The single machine early/tardy problem", *Managment Science*, Vol.35, pp.177-191, 1989.

Özdamar, L., and G.Ulusoy, "A local constraint based analysis approach to project scheduling under general resource constraints", *European Journal of Operational Research*, Vol.79, pp.287-298, 1994.

Özdamar, L., and G.Ulusoy, "A survey on the resource-constrained project scheduling problem", *IIE Transactions*, Vol.27, pp.574-586, 1995.

Rechenberg, I., *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

Reeves, C.R., *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley and Sons, New York, 1993.

Rosenblatt, F., *Principles of Neurodynamics*, Spartan, Washington, D.C, 1962.

Schaffer, J.D., R.A. Caruana, L.J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization", Proceedings of the Third International Conference on Genetic Algoirthms, J.D. Schaffer (ed.), Morgan Kaufmann, 1989.

Schwefel, H.-P., *Evolutionsstrategie und Numerische Optimierung*, Dissertation, Technische Universitaet Berlin, May 1975.

Schwefel, H.-P., *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Vol. 26 of Interdisciplinary Systems Research, Birkhaeuser, Basel, 1977.

Shtub, A., J.F. Bard and S. Globerson, *Project Management; Engineering, Technology and Implementation*, Prentice Hall, Englewood Cliffs, NJ, 1994.

Sirinivas, M., and L.M. Patnaik, "Genetic algorithms: A survey", *IEEE Computer*, pp .17-26, 1994.

Smith-Daniels, D.E., and N.J. Aquilano, "Using a late start resource constrained project schedule to improve project net present value", *Decision Sciences*, Vol.18, pp.617-630, 1987.

Sridharan, V. and Z. Zhou, "A decision theory based scheduling procedure for single machine weighted earliness and tardiness problem", *European Journal of Operational Research*, Vol.94, pp.292-301, 1996.

Tank, D.W., and J.J. Hopfield, "Simple optimization networks: An A/D converter and a linear programming circuit", *IEEE Transactions on Circuits and Systems*, Vol. 33, pp. 533-541, 1986.

Uckun, S., S. Bagchi, K. Kawamura, and Y. Miyabe, "Managing genetic search in job shop scheduling", IEEE Expert, pp.15–24, 1993.

Ulusoy, G., and L. Özdamar, "A heuristic scheduling algorithm for improving the duration and net present value of a project", *International Journal of Operations & Production Management*, Vol.15, pp.89-98, 1995.

Ulusoy, G., F. Sivrikaya-Şerifoğlu and Ü. Bilge, "A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles", *Computers and Operations Research*, Vol.24, No.4, pp.335-351, 1997.

Yano, C.A., and Y. Kim, "Algorithm for a class of single machine weighted tardiness and earliness problem", *European Journal of Opeartional Research*, Vol.52, pp.167-178, 1991.