ANADOLU ÜNİVERSİTESİ BİLİM VE TEKNOLOJİ DERGİSİ
ANADOLU UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY
Cilt/Vol.:2 - Sayı/No: 1 : 37-49 (2001)

DERLEME/REWIEW

AN OVERVIEW OF GENETIC ALGORITHMS

Özgür YENİAY[1]

ABSTRACT

In this paper, general knowledge about Genetic Algorithms (GAs), which have received much attention the past few years because of the fact that they have been successfully applied in several different fields of study, is given. The basic principles of GAs, such as representation, selection and genetic operators are summarized and the fundamental theorem of GAs, the so-called schema theorem, is discussed. A simple example is given in order to explain how they work. Finally, a literature survey for the applications of GAs to Operational Research (OR) problems is also provided.

Key Words: Genetic Algorithms, Stochastic Search, Function Optimization, Operational Research.

GENETİK ALGORİTMALARA GENEL BİR BAKIŞ

ÖZ

Bu çalışmada, birçok farklı alanda başarıyla uygulanmalarından dolayı son yıllarda büyük ilgi gören Genetik Algoritmalar (GA) hakkında genel bilgi verilmektedir. GA'ların gösterim, seçme ve genetik operatörler gibi temelleri özetlenmekte ve şema teoremi olarak bilinen GA'ların temel teoremi incelenmektedir. GA'ların nasıl işlediğini açıklamak amacıyla basit bir örnek verilmektedir. Son olarak, Yöneylem Araştırması problemlerinde GA'ların uygulamalarına yönelik bir yayın taraması da sunulmaktadır.

Anahtar Kelimeler: Genetik Algoritmalar, Stokastik Arama, Fonksiyon Eniyileme, Yöneylem Araştırması.

1. INTRODUCTION

A GA may be described as a mechanism that mimics the genetic evolution of a species (Goldberg, 1989; Holland, 1992). The basic principles of GA were first proposed by John Holland at the University of Michigan in the late 1960s and early 1970s. Thereafter a series of literature and reports have become available. There have been a number of international conferences and also journals on the theory and applications of GAs. They have been used successfully in a wide variety of applications including mathematical function optimization, very large scale integration (VLSI) chip layout, parameter fitting, scheduling, manufacturing, clustering, machine learning, design problems etc. and are still finding increasing acceptance. Reeves (1997) states that the level of funding for GA-based projects by national governments and by organizations such as European Commission, far exceeds any funding given to the projects based on the methods such as simulated annealing and tabu search.

There are some differences between GA's and the traditional optimization techniques. They work using an encoding of the variables, rather than the variables themselves, and use probabilistic transition rules to move from one population of solutions to another rather than a single solution to another. The most important and interesting characteristics of genetic algorithms is that they require only the value of the objective function. That is, they do not use derivatives or other knowledge (Hadj-Aloune and Bean, 1997).

2. GENETIC ALGORITHMS

In nature, individuals in a population compete with each other for food, space and mates. Those individuals which are fitter with respect to their environ-

1 Hacettepe University, Faculty of Science, Deparment of Statistics 06532, Beytepe-Ankara.

Received: 16 May 2000; **Revised:** 18 December 2000; **Accepted:** 18 April 2001.

ment and more attractive for mates, survive and reproduce longer than weaker ones. Their characteristics, encoded in their genes, are transmitted to their offspring and tend to propagate into new generations. This natural phenomenon is called "survival of the fittest". In this way, on average, each generation consists of fitter individuals than parents. (Srinivas and Patnaik, 1994).

GAs simulate, in a rather simplified way, the processes outlined above to get better solutions for a problem. They work with a population of individuals, each representing a possible solution to a given problem. Each possible solution must be encoded in a binary or non-binary string format such as Gray code, floating point representation, sequence representation etc. (for details, see, Reeves, 1993; Janikow and Michalewicz). These strings are analogous to chromosomes in nature. Strings are also called individuals or solutions in GA literature. A position on a chromosome is called a locus. A set of one of more loci is called a gene and the possible values that a gene can have are known as alleles. Note that the GA literature is rich in terms borrowed from genetics. Holland used these terms in order to describe numerous steps of his new algorithm.

GAs typically work by iteratively generating and testing candidate solutions as given in Figure 1.

The simplest forms of GAs work according to the scheme shown in Figure 1.

Representation: The first step to develop a GA for an optimization problem is to represent it so that every solution for it is in the form of a string of bits, all of them consisting of the same number of element. Each of the strings represents a random point in the space of the possible solutions to a problem. There are several ways of encoding the parameters. Binary encoding (i.e. bit strings) is the most common way of encoding. If a parameter which can take values between a and b is mapped to a string of length n, the precision is,

$$\Pi = \frac{b - a}{2^n - 1} \qquad (1)$$

As seen from equation 1, the length of the string is a function of the required precision. The longer the string the better the precision. It is possible to find the original values of the parameters by decoding a string. In order to find the real value, x, corresponding to the binary string (b_n-1 b_n-2 ,..., b_0), the transformation given below is used:

$$x = a + \left(\sum_{i=0}^{n-1} 2^i b_i \right) \Pi \qquad (2)$$

where n is the number of the bits in the string. A string is created by joining the parameters together. Suppose that the problem is to maximize a function of three parameters, f (x,y,z). Each parameter might be represented by a 5-bit binary string. Therefore, the string contains three genes and consists of 15 binary numbers.

Binary strings are sufficiently general but they are not always the more natural or the more adequate representation. So, other types of coding such as Gray code, floating point representation, sequence representation etc. have been used in several studies.

Initial Population: Usually an initial population is created randomly. In some applications, the initial population is generated by using some other method. Unless a given region of the search space is known to contain the optimal solution, this initial population should be spread over enough of the search space to represent as wide a variety of solutions as possible.

The number of population strings (N) that are initially generated (this number remains constant throughout successive generations) is usually 40 to 250 or larger, depending on the size of the problem being solved.

Fitness value: A fitness function must be divided for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical fitness value, which is proportional to the ability or utility of the solution represented by that chromosome. It is this information that guides in the selection of the fitter solutions in each generation. The higher the fitness value of a solution, the higher its chances of survival and reproduction and the larger its representation in the subsequent generation.

If objective function is profit or utility function i.e. maximum value of f(x) is being searched, and f(x) is positive for all solutions, the problem is,

$$\text{"find } \hat{x} \text{ such that } f(\hat{x}) = \max_{x} f(x)\text{"}$$

For such problems, we can use f(x) as a fitness measure, where f(x) is the raw fitness.

In some problems f(x) can be negative for some solutions. Therefore, fitness proportionate selection can

1. initialize population (t)
2. determine fitness of population (t)
3. repeat until a stopping criterion is satisfied
   a) select parents from population (t)
   b) perform crossover on parents creating population (t+1)
   c) Perform mutation on population (t+1)
   d) determine fitness of population (t+1)

Figure 1. The Steps of A Typical Genetic Algorithm.

not be used ($f_{raw} < 0$ gives a negative selection probability). In order to avoid this, the transformation given below is used:

$$g(x) = \begin{cases} f(x) + C_{min} & \text{, if } f(x) + C_{min} > 0 \\ 0 & \text{,otherwise} \end{cases} \quad (3)$$

where C_{min} is the absolute value of one of the following:

- minimum value, f(x) can take (when known),
- minimum value of f(x) in the current / or last k generations,
- A function of variance of f(x) in the current population, e.g.

$$\text{Mean}[f(x)] - 2\sqrt{\text{Var}[f(x)]} \quad (4)$$

In some problems fitness function, f(x), is a cost or an error E(x) and the problem is,

$$\text{"find } \hat{x} \text{ such that } E(\hat{x}) = \min_{x} E(x)\text{"}$$

In this situation, f(x)= –E(x) is taken as a fitness function to be maximized and in order to avoid negative probabilities, equation 1 is used to avoid negative probabilities with fitness proportionate selection or transformation given below is used:

$$g(x) = \begin{cases} C_{max} - E(x) & \text{, if } C_{max} > E(x) \\ 0 & \text{,otherwise} \end{cases} \quad (5)$$

where C_{max}, is one of the following:

- maximum value, E(x) can take (when known),
- maximum value of E(x) in the current and/ or last k generations,
- a function of the variance of E(x) in the current population, e.g.

$$\text{Mean}[E(x)] + 2\sqrt{\text{Var}[E(x)]} \quad (6)$$

If all values of $f(x) \in [f_{min}, f_{max}]$ with $f_{min} >> 0$, and $f_{max} - f_{min} >> 0$, then fitness proportionate selection can lead to stagnation even at the beginning of a run. In this situation, $C_{min} = -f_{min}$ is taken and the equation 1 is used so that $f(x) \in [0, f_{max} - f_{min}]$.

Selection: Once all individuals in the population have been evaluated, their fitnesses are used as the basis for selection. Selection allows strings to be copied for possible inclusion in the next generation. The chance that a string will be copied is based on the string's fitness value, calculated from a fitness function. The object of the selection method is to give exponentially increasing trials to the fittest strings. There are several possibilities (see Blickle and Thiele, 1995), but here fitness proportionate selection, one of the simplest, will be explained here. After having found the fitness of each individual f_i in a given generation, a probability is assigned to each string as follows:

$$P_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \quad (7)$$

where f_i is fitness value of individual i , $\sum_{i=1}^{N} f_i$ is the total population fitness.

Finally, a cumulative probability is obtained for each string by adding up the fitnesses of the preceding population members;

$$c_i = \sum_{k=1}^{i} P_k \qquad i = 1,2,...,N \quad (8)$$

A random r, uniformly distributed in [0,1], is drawn N times and each time the i-th string is selected such that $c_{i-1} < r \leq c_i$. When $r<c_1$, the first string is selected. This process can be visualized as the spinning of a biased roulette wheel divided into N slots, each with a size proportional to the string's fitness (Tomassini, 1995).

This selection method is not without problems. One problem is that, after a while, since better individuals get more copies in successive generations, the differences in fitness between individuals become small which renders selection ineffective. In this case, the selection pressure need to reduce more often than they would under the normal fitness evaluation. Another problem is the possible existence of a super individual (solutions with fitness values significantly better than average). With fitness proportionate selection this individual get many copies in successive generations and rapidly come to dominate the population, thus causing premature convergence to a possibly local optimum. Scaling mechanisms and selection methods do not allocate trials proportionally to fitness (see next section).

Genetic Operators: Whatever selection method is used, it does not introduce any new solutions. It is here that two individuals selected in the previous step are allowed to produce offspring via the use of specific genetic operators such as crossover and mutation. The most commonly used crossover operators are one point crossover, two point crossover, multi point crossover and uniform crossover. In crossover, GA seeks to construct better solutions by combining the features of good existing ones.

One point Crossover: One point crossover is applied to two chromosomes (parents) and creates two new chromosomes (offspring) by selecting a random positi-

on along the string and splicing the section that appears before the selected position in the first string with the section that appears after a selection position in the second string and via versa (See Figure 2).

Crossover is not applied to all pairs of individuals. Pairs of individuals are selected to undergo crossover with crossover probability p_c. Typically, it is between 0.6 and 1.0. If crossover is not applied, the two selected strings are passed into the next generation unchanged (Beasley et al., 1993). It is clear that approximately p_cN strings undergo crossover in each generation. (Grefenstete, 1986).

Mutation: Selection and crossover alone can obviously generate a staggering amount of differing strings. However, depending on the initial population chosen, there may not be enough variety of strings to ensure the GA seeks the entire population space. The purpose of the mutation is to provide insurance against the irrevocable loss of genetic information and hence to maintain diversity within the population. For example, if every solution in the population has 0 as the value of a particular bit, then no amount of crossover will produce a solution with a 1 there instead.

Mutation is applied to each string after crossover. It randomly alters each bit with a small probability (see Figure 3). Typical values for p_m mutation probability range from 0.001 to 0.01. Approximately, p_mNn mutations occur per generation.

In Figure 3, the third bit of the string is being mutated.

Stopping Criteria: When cycle of evolution, selection, crossover, mutation and fitness evolution is iterated for many generations, the overall fitness of the population generally improves and the strings in the population represent improved solutions. The stopping criterion can be set by the number of evolution cycles, the amount of variation of individuals between different generations, or a predefined value of fitness.

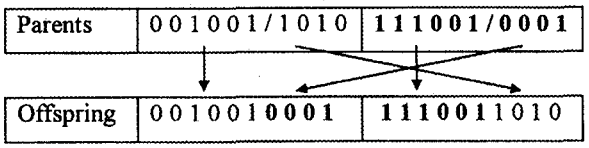

Figure 2. One Point Crossover.

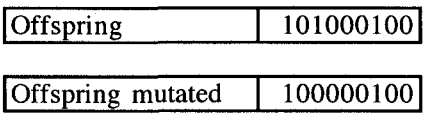

Figure 3. Mutation.

3. SELECTION OF GAs PARAMETERS

The effects of GAs parameters are very important on the performance of GAs. Therefore, these parameters, called control parameters, should be chosen correctly. There are 6 control parameters. These are population size, crossover probability, mutation probability, generation gap (G), selection strategy and scaling function.

Population size: One of the most important decisions is the population size determined by GA user. If the population size is too small, GA may converge to a local optimum, if it is too large it increases the cost per generation (e.g. run-time).

Crossover probability: The aim of crossover is to construct better chromosomes by combining the features of good existing chromosomes. Pair of chromosomes are chosen to undergo crossover operation with probability p_c. The increase of it causes the recombination of building blocks to rise but it also increases the disruption of good chromosomes.

Mutation probability: The purpose of mutation is to maintain genetic diversity in the population. Mutation occur at each bit in a chromosome with probability p_m. If the p_m increases, genetic search transform into a random search; but it also helps reintroduce the lost genetic material.

Generation gap: The fraction of new chromosomes at each generation is called the generation gap. It determinates how many chromosomes are selected for genetic operators (between 0 and 100 percent). A high value means that many chromosomes are replaced which causes faster convergence.

Selection strategy: There are several strategies to replace the old generation. In generational strategy, the chromosomes in the current population are completely replaced by the offspring. Since the best chromosome of the population may fail to reproduce offspring in the next generation, it is usually combined with elitist strategy. Elitist strategy consist of never replacing the best chromosomes in the population with inferior solution, so the best solution is always available for reproduction. However, in the steady state strategy, only a few chromosomes are replaced in each generation. Usually the worst chromosomes are replaced when new chromosomes are inserted in to the population.

Scaling Functions: There are several scaling methods. The most commonly used are given below:

- *Linear Scaling:* Linear scaling computes the scaled fitness value as

$$f' = af + b \qquad (9)$$

where is the fitness value, f′ is the scaled fitness value. Coefficients a and b are chosen each generation so that the average values of f and f′ are equal and so that the maximum value of f' is a specified multiple of the average. (Michalewicz, 1994)

- *Sigma Truncation:* In this method, the scaled fitness values are determined as follows:

$$f' = f - (\bar{f} - c\,\sigma) \qquad (10)$$

where $\bar{f}$ is the average fitness values of the population, σ is the standard deviation in the population, c is a small constant typically ranging from 1 to 3., Any negative result $f'<0$ is set to zero, in order to prevent negative values of f′.

- *Power Law Scaling:* In this method, the actual fitness value is taken to some specific power:

$$f' = f^k \qquad (11)$$

where k is, in general, problem dependent or even varying during the run (Man et al., 1996) .

4. WHY GAs WORK: SCHEMATA AND BUILDING BLOCKS

In this section, we will look into the standard genetic algorithm workings in a more detailed way, in order to see why GAs constitute an effective search procedure. Holland introduced the notation of schema to explain how GAs search for regions of high fitness. Schemata (plural of schema) are theoretical constructs used to explain the behavior of GAs and are not processed directly by the algorithm. A schema (or similarity template) is a string that describe similarity between certain sets of strings. Alphabet {0, 1, *} is used in order to define a schema,. For example schema H below is a template for a set of chromosomes having 0 in their first locus and 1 in their second and fourth locus.

H= 0 1 * 1 *

The * is a don't care symbol that matches both 0 and 1. A bit string x that matches the pattern of a schema, x, is said to be an instance of H. For example the string x given by

x= 0 1 1 1 0

is an instance of H above. The schema H also matches the following strings: (0 1 0 1 0), (0 1 0 1 1), (0 1 1 1 0) and (0 1 1 1 1).

The fitness of any bit string in the population gives some information about the average fitness of the 2^n different schemata of which it is an instance, so an explicit evaluation of a population of N individual strings is also an implicit evaluation of a much larger number of schemata. This is referred to as implicit parallelism.

In schemata, a 0 or 1 is referred to as a defined bit. The order of a schema H, denoted by o(H), is the number of non-* symbols in the schema. Its defining length, denoted by l(H), is the distance between the furthest non-* symbols in the schema. For the schema H= 0 1 * 1 *, l(H)=4-1=3 and o(H)=3. The order of a schema and its defining length are very important in the fundamental theorem of genetic algorithms: schema theorem. The theorem is as follows:

Suppose n is the length of a chromosome and H is a schema of the string, f(H,t) is the average fitness value of the instances of schema H at generation t, and N(H,t) is the number of those instances in the population, then the expected number of instances schema H at generation t+1 is,

$$N(H, t+1) \geq \left\{1 - \frac{p_c l(H)}{n-1} - p_m o(H)\right\} \frac{f(H, t)}{\bar{f}(t)} N(H, t) \qquad (12)$$

where $\bar{f}(t)$ is the average fitness value of the whole population at generation t (Mitchell, 1996; Odetayo,1995).

This result says that the number of short, low order, above average schemata grows exponentially in subsequent generations of a genetic algorithm. We can say that the schema theorem expressed a reduced view of GA. This above average, low order and short defining length schemata are called building blocks and play an important role in the theory. Holland propose that GAs work by identifying good building blocks and eventually combining these to get larger building blocks. This idea has become known as the building block hypothesis.

5. A SIMPLE EXAMPLE

In this section, in order to explain how GAs work and their power, optimization of a simple function of one variable will be presented. Although GAs are not limited to this domain, their workings are probably better understood in an optimization setting. The function is defined as

$$f(x) = 3x\mathrm{Sin}(2px) + 10 \qquad 0 \leq X \leq 3.$$

and is drown in Figure 4.

The problem is to find x* in the interval [0,3] which maximizes the function. The problem is not a mathematically hard one. It could be solved by hand or with a number of other established methods. It is purely of illustrative value. Note that, for comparison, it was solved by using Mathematica for Windows and x*=2.264 value which correspond f(x*)=16.7657 is found. For the optimization of the function, SGA (simple genetic algorithm) code has been used (see Goldberg (1989)). Let us examine the components of the genetic algorithm for solving the given problem in turn.

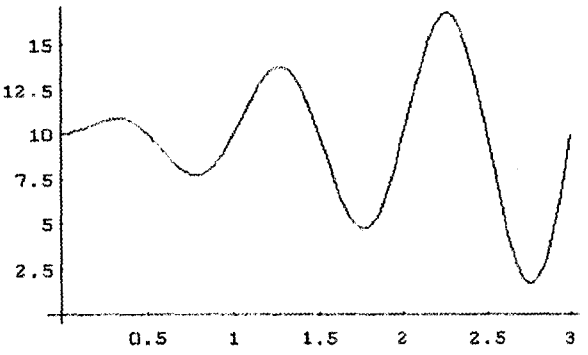

Figure 4. Graph of f(x) = 3xSin(2px)+10.

Representation: In order to represent real values of the variable x, binary string will be used. Thus, each individual is a value of the real variable x. Because of the fact that the length of the string depends on the required precision, the longer the string the better the precision. In this study, the length of a string is n=10 bits. The length of the string implies that the interval [0,3] should be devised into at least 1024 equal size ranges. Note that, our use of 10 bit strings is only for the illustrative purposes. In real applications, longer strings are needed. Thus the precision is,

$$\Pi = \frac{3 - 0}{2^{10} - 1} = 0.00293255$$

This means that GA will be able to sample points no less than 0.00293255 apart from each other. The strings 0 0 0 0 0 0 0 0 0 0 and 1 1 1 1 1 1 1 1 1 1 will represent 0 and 3 respectively. Any other 10 bit string will be mapped to an interior point.

Initial population: The initial population will be generated by 15 randomly chosen strings.

Genetic operators: SGA uses two classical genetic operators: mutation and crossover.

Parameters: The parameter values given below was used :

Population size: N=15

Probability of crossover: p_c=0.7

Probability of mutation: p_m=0.01.

Stopping criteria: SGA will be stopped at the end of 30 generations .

Experimental results: Initial population given in Table 1 was generated at random. In this Table, strings generated at random, their real values, their fitness values and their probability of being chosen is given.

SGA was run 30 generations by starting from the initial population given in Table 1. In Table 2, we provide the generation number for which we noted an improvement in the evaluation function, maximum and the average fitness values at those generations.

Table 1. The Initial Population Generated By SGA.

String number	String	x	f(x)	Pi
1	0 1 1 1 0 1 0 0 0 1	1.3636	13.0917	0.0760
2	1 1 0 0 0 1 0 0 0 1	2.3020	16.5401	0.0960
3	1 0 1 1 0 0 0 1 1 1	2.0850	13.1856	0.0766
4	1 0 1 1 0 0 0 1 1 1	2.0850	13.1856	0.0766
5	1 1 0 0 0 1 0 1 1 1	2.3196	16.3032	0.0946
6	0 1 1 0 0 1 0 0 0 0	1.1730	13.1154	0.0762
7	0 1 0 1 0 1 1 1 0 0	1.0205	10.3938	0.0604
8	0 1 0 1 0 1 1 1 0 0	1.0205	10.3938	0.0604
9	1 1 0 0 0 1 1 0 1 1	2.3314	16.0996	0.0935
10	1 0 1 1 0 0 0 1 1 1	2.0850	13.1856	0.0766
11	1 1 1 1 0 1 0 1 0 0	2.8739	3.8616	0.0224
12	0 0 0 1 1 0 0 0 0 0	0.2815	10.8281	0.0629
13	1 1 1 0 0 1 1 1 1 1	2.7185	2.0040	0.0116
14	0 0 0 0 0 0 0 0 1 0	0.0059	10.0006	0.0581
15	0 0 0 0 0 0 0 0 0 0	0.0000	10.0000	0.0581

Table 2. Generations For Which An Improvement In The Evaluation Function Is Noted and The Maximum and The Average Fitness Values At Those Generations.

Generation number	Maximum fitness value	Average fitness value
0	16.5401	10.8126
2	16.5715	9.5090
8	16.7568	10.3071
9	16.7619	8.6607
21	16.7658	11.4978
28	16.7668	9.0576

The best string found at the end of the run is 1 1 0 0 0 0 0 1 1 which corresponds to a value of x*=2.260997067, f(x*)=16.7668. Note that a single run was done with SGA. If several runs had been done, better solutions could have been found.

6. LITERATURE SURVEY

As GAs are general optimization method, this makes both kind and number of application increase. Therefore, it is clear that preparing a full reference list of GAs according to the all subjects will be hard and this list will be very long.

One of the popular application areas using GAs is OR. In this section, we give a literature survey for applications of GAs to OR problems. We restricted our literature search to published journal articles. Total of 131 journal articles were identified through the our literature search. These were published in 41 different journals. The classification of these articles based on application area is given in Table 3.

Table 3. Some Applications of GAs In OR.

Application	Reference
1. Assembly Line Balancing	Leu, Matheson and Rees (1994); Rubinovitz and Levitin (1995); Tsujimura, Gen and Kubota (1995); Kim, Kim and Kim (1996); Suresh, Vinod and Sahu (1996); Sabuncuoğlu, Erel and Tanyer (1999); Lee, Kahoo and Yin (2000); Kim, Kim and Kim (2000); Ponnambalam, Aravindan and Naidu (2000)
2. Assignment	Huntley and Brown (1991); Nissen (1992); Levitin and Rubinovitz (1993); Tate and Smith (1995); Ahuja, Orlin and Tiwari (2000)
3. Bin-Packing	Kroger (1995); Jakobs (1996); Reeves (1996)
4. Facility Layout	Tam (1992); Chan and Tansri (1994); Conway and Venkataramanan (1994); Suresh, Vinod and Sahu (1995); Del maire, Langevin and Riopel (1997); Islier (1998); Kochhar, Foster and Heragu (1998); Raja sekharan, Peters and Yang (1998); Tam and Chan (1998); Hamamoto, Yih and Salvendy (1999); Kochhar and Heragu (1999); Al-Hakim (2000)
5. Goal Programming	Gen, Ida, Lee and Kim (1997); Taguchi, Ida and Gen (1997)
6. Integer Programming	Yokota, Gen and Li (1996); Yokota, Gen, Li and Kim (1996)
7. Knapsack	Thiel and Voss (1994)
8. Linear	Wang (1997); Lee and Yang (1998); Gueyagueler and Guemrah (1999); Urdeneta, Gomez, Sorrentino, Flores and Programming Diaz (1999)
9. Nonlinear Programming	Yokota, Gen, Taguchi and Li (1995); Sakawa and Yauchi (1998)
10. Scheduling	
10.1. Job-Shop	Biegel and Davern (1990); Uckun, Bagchi, Kawamura and Miyabe (1993), Bierwirth (1995); Dagli and Sittisat hanchai (1995); Della, Tadei and Volta (1995); Domdorf and Pesch (1995) Gilkinson, Rabelo and Bush (1995); Cheng, Gen and Tsujimura (1996); Kumar and Srinivasan (1996); Maturana, Gu, Naumann and Norrie (1996); Lee, Piramuthu and Tsai (1997); Shi (1997); Candido, Khatar and Barcia (1998); Cheng, Gen and Tsujimura (1999); Ghedjati (1999); Sakawa and Mori (1999); Hajri, Liouane, Hammadi and Borne (2000); Qi, Burns and Harrison (2000); Sakawa and Kubota (2000); Wu and Zhao (2000)
10.2. Flow-Shop	Badami and Parks (1991); Ishibuchi, Yamamoto, Murata and Tanaka (1994); Chen, Vempati and Aljaber (1995); Reeves (1995); Murata, Ishibuchi and Tanaka (1996); Nagar, Heragu and Haddock (1996); Janiak and Portmann (1998); Onwubolu and Mutingi (1999); Nitin, Bagchi and Wagneur (2000)
10.3. Open-Shop	Liaw (2000)
10.4. Single Machine	Gupta, Gupta and Kumar (1993); Lee and Choi (1995); Lee and Kim (1995); Crauwels, Potts and Van Wassenhove (1996); Maimon and Broha (1998); Webster, Jog and Gupta (1998); Hussain and Sastry (1999); Liu and Tang (1999); Wang, Gen and Cheng (1999); Armentano and Mazzini (2000)
10.5. Parallel Machine	Cheng, Gen and Tozawa (1995); Kimms (1999); Min and Cheng (1999)
10.6. Maintenance	Chan, Fwa and Tan (1994); Kim, Nara and Gen (1994); Deris, Omatu, Ohta, Kutar and Samat (1999)
10.7. Process Planning	Vancza and Markus (1991); Awadh, Sepehri and Hawaleshka (1995); Yip-Hoi and Dutta (1996); Jain and ElMaraghy (1997); Morad and Zalzala (1999)
10.8. Others	Wren and Wren (1995); Levine (1996); Malmborg (1996); Mori and Tseng (1997); Ulusoy, Sivrikaya and Bilge (1997); Covalieri and Gaiardelli (1998); Moutaz, Michalewicz and Wilmot (1998); Easton and Man sour (1999); Lam, Lin, Sriskandrarajah and Yan (1999); Norman and Bean (1999); Todd and Sen(1999); Cai and Li (2000); Hoda, Viskvas and Ben (2000); Ip, Li, Man and Tang (2000); Jahangiran and Conray (2000); Li, Man, Tang, Kwong and Ip (2000)
11. Set Covering	Al-Sultan, Hussain and Nizami (1996); Beasley and Chu (1996)
12. Steiner Tree	Kapsalis, Rayward-Smith and Smith (1993); Esbensen (1995); Saltouros, Verentziotis, Markai, Theologou and Vanieris (2000)
13. System Reliability	Painton and Campbell (1995); Sasaki, Gen and Yamashiro (1995); Coit and Smith (1996 a,b); Gen and Cheng (1996); Dengiz, Altiparmak and Smith (1997); Altiparmak, Dengiz and Smith (2000)
14. Timetabling	Monfroglio (1996 a,b); Kragelund (1997)
15. Traveling Salesman	Jog, Jung and Gucht (1991), Homaifar, Guan and Liepins (1992); Lin, Delgado, Gause and Vassiliadis (1995); Potvin (1996); Chatterjee, Carrera and Lynch (1996); Qu and Sun (1999); Louis and Li (2000)
16. Vehicle Routing	Kopfer, Pankratz and Erkens (1994); Filipec, Sklec and Krajcar (2000)

Table 3 shows that there is a great deal of interest in the use of GAs to solve problems arising in the area of OR. According to the our literature search, job shop scheduling has the largest number of applications, followed by facility layout, assembly line balancing, flow shop scheduling, traveling salesman, system reliability and other problems. An examination of the publication source of these 131 articles reveals that Computers & Industrial Engineering, European Journal of Operational Research and Computers & Operations Research are the top three journals publishing the articles. Also, it is important that more than 30% of these 131 articles were published in the last two years. This is very important result for the future of GA applications to OR problems.

7. CONCLUSION

This paper presented an overview of all the technical aspects and basic principles of GAs. All the different steps of the algorithm were reviewed separately. We have also provided a literature survey of OR applications that have been successfully tackled using GAs. Our literature survey results have revealed that GAs have great acceptance in OR community. There is no doubt that GA applications to OR problems will be increasing in the future. The findings of this articles will hopefully be of value to students, researchers and practitioners in the field.

REFERENCES

Ahuja, R.K., Orlin, J.B. and Tiwari, A. (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic Generated Initial Populations and Multiple Evaluation Criteria, *Decision Sciences*, 25, 581-606.

Al-Hakim, L. (2000). On Solving Facility Layout Problems Using Genetic Algorithms, *International Journal of Production Research*, 38, 2573-2582.

Al-Sultan, K.S., Hussain, M.F. and Nizami, J.S. (1996). A Genetic Algorithm for the Set Covering Problem, *Journal of the Operational Research Society*, 47, 702-709.

Altıparmak, F., Dengiz, B. and Smith, A.E. (2000). An Evolutionary Approach for Reliability Optimization in Fixed Topology Computer Networks, *Yöneylem Araştırması Dergisi*, 12, 57-75.

Armentano, V.A. and Mazzini, R. (2000). A Genetic Algorithm for Scheduling on a Single Machine with Set-up Times and Due Dates, *Production Planning & Control*, 11, 713-720.

Awadh, B., Sepehri, N. and Hawaleshka, O. (1995). A Computer-Aided Process Planning Model Based on Genetic Algorithms, *Computers & Operations Research*, 22, 841-856.

Badami, V.S. and Parks, C.M. (1991). A Classifier Based Approach to Flow Shop Scheduling, *Computers & Industrial Engineering*, 21, 329-333.

Beasley, D., Bull, D.R. and Martin, R.R. (1993). An Overview of Genetic Algorithms: Part I-Fundamentals, *Univ. Compt.*, 15, 258-269.

Beasly, J.E. and Chu, P.C. (1996). A Genetic Algorithm for The Set Covering Problem, *European Journal of Operational Research*, 94, 392-404.

Biegel, J.E. and Davern, J.J. (1990). Genetic Algorithms and Job Shop Scheduling, *Computers & Industrial Engineering*, 19, 81-91.

Bierwirth, C. (1995). A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms, *OR Spektrum*, 17,87-92.

Blickle, T. and Thiele, L. (1995). A Comparison of Selection Schemes Used in Genetic Algorithms, *TIK-Report*, No:11, 2nd edition.

Cai, X. and Li, K.N. (2000). A Genetic Algorithm for Scheduling Staff of Mixed Skills Under Multi-Criteria, *European Journal of Operational Research*, 125, 359-369.

Candido, M.A.B., Khator, S.K. and Barcia, R.M. (1998). A Genetic Algorithm Based Procedure for More Realistic Job Shop Scheduling Problems, *International Journal of Production Research*, 36, 3437-3457.

Cavalieri, S. and Gaiardelli, P. (1998). Hybrid Genetic Algorithms for a Multiple-Objective Scheduling Problem, *Journal of Intelligent Manufacturing*, 9, 361-367.

Chan, K.C. and Tansri, H. (1994). A Study of Genetic Crossover Operations on The Facilities Layout Problem, *Computers & Industrial Engineering*, 26, 537-550.

Chan, W.T., Fwa, T.F. and Tan C.Y. (1994). Road Maintenance Planning Using Genetic Algorithms, *Transportation Engineering*, 120, 693-709.

Chatterjee, S., Carrera C. and Lynch, L.A. (1996). Genetic Algorithms and Traveling Salesman Problems, *European Journal of Operational Research*, 93, 490-510.

Chen, C.L., Vempati, V.S. and Aljaber, N. (1995). An Application of Genetic Algorithms for Flow Shop Problems, *European Journal of Operational Research*, 80, 389-396.

Cheng, R., Gen, M. and Tozowa, T. (1995). Minimax Earliness/Tardiness Scheduling in Identical Parallel Machine System Using Genetic Algorithms, *Computers & Industrial Engineering*, 29, 513-517.

Cheng, R., Gen, M. and Tsujimura, Y. (1996). A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms. Part I: Representation, *Computers & Industrial Engineering*, 30, 983-998.

Cheng, R., Gen, M. and Tsujimura, Y. (1999). A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms. Part II: Hybrid Genetic Search Strategies, *Computers & Industrial Engineering*, 36, 343-364.

Coit, D. and Smith, A.E. (1996a). Penalty Guided Genetic Search for Reliability Design Optimization, *International Journal of Computers and Industrial Engineering*, 30, 895-904.

Coit, D. and Smith, A.E. (1996b). Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm, *IEEE Transactions on Reliability*, 45, 254-260.

Conway, D.G. and Venkataramanan, M.A. (1994). Genetic Search and The Dynamic Facility Layout Problem, *Computers & Operations Research*, 21, 955-960.

Crauwels, H.A.J., Potts, C.N. and Van Wassenhave, L.N. (1996). Local Search Heuristics for Single-Machine Scheduling with Batching to Minimize the Number of Late Jobs, *European Journal of Operational Research*, 90,200-213.

Dağlı, C.H. and Sittisathanchai, S. (1995). Genetic Neuro-Scheduler: A New Approach for Job Shop Scheduling, *International Journal of Production Economics*, 41, 135-145.

Della, C.F., Tadei, R. and Volta, G. (1995). A Genetic Algorithm for The Job Shop Problem, *Computers & Operations Research*, 22, 15-24.

Delmaire, H., Langevin, A. and Riopel, D. (1997). Skeleton-Based Facility Layout Design Using Genetic Algorithms, *Annals of Operations Research*, 69, 85-104.

Dengiz, B., Altıparmak, F. and Smith, A.E. (1997). Efficient Optimization of All-Terminal Reliable Networks Using an Evolutionary Approach, *IEEE Transactions on Reliability*, 46, 18-26.

Deris, S., Omatu, S., Ohta, H., Kutar, L. and Somat, P.A. (1999). Ship Maintenance Scheduling by Genetic Algorithm and Constraint-Based Reasoning, *European Journal of Operational Research*, 112, 489-502.

Dorndorf, U. and Pesch, E. (1995). Evolution Based Learning in a Job Shop Scheduling Environment, *Computers & Operations Research*, 22, 25-40.

Easton, F.F. and Mansour, N. (1999). A Distributed Genetic Algorithm for Deterministic and Stochastic Labor Scheduling Problems, *European Journal of Operational Research*, 118, 505-523.

Esbensen, H. (1995). Computing Near Optimal Solutions to The Steiner Problem in a Graph Using a Genetic Algorithm, *Networks*, 26, 173-185.

Filipec, M., Sklec, D. and Krajcar, S. (2000). Genetic Algorithm Approach for Multiple Depot Capacitated Vehicle Routing Problem Solving with Heuristic Improvements, *International Journal of Modeling & Simulation*, 20, 320-328.

Gen, M. and Cheng R. (1996). Optimal Design of System Reliability Using Interval Programming and Genetic Algorithms, *Computers & Industrial Engineering*, 31, 237-240.

Gen, M., Ida, K., Lee, J. and Kim, J. (1997). Fuzzy Nonlinear Goal Programming Using Genetic Algorithm, *Computers & Industrial Engineering*, 33, 39-42.

Ghedjati, F. (1999). Genetic Algorithms for The Job-Shop Scheduling Problem with Unrelated Parallel Constraints: Heuristic Mixing Method Machines and Precedence, *Computers & Industrial Engineering*, 37, 39-42.

Gilkinson, J.C., Rabelo, L.C. and Bush, B.O. (1995). A Real-World Scheduling Problem Using Genetic Algorithms, *Computers & Industrial Engineering*, 29, 177-182.

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

Grefenstette, J.J. (1986). Optimization of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, *Man and Cybernetics*, 16, 122-128.

Gueyagueler, B. and Guemrah, F. (1999). Comparison of Genetic Algorithm with Linear Programming for the Optimization of an Underground Gas-Storage Field, *In Situ*, 23, 131-149.

Gupta, M., Gupta, Y. and Kumar, A. (1993). Minimizing Flow Time Variance in a Single Machine System Using Genetic Algorithm, *European Journal of Operational Research*, 70, 289-303.

Hadj-Alouane, A. and Bean, J.C. (1997). A Genetic Algorithm for the Multiple-Choice Integer Program, 45, 92-101.

Hajri, S., Liouane, N., Hammadi, S. and Borne, P. (2000). A Controlled Genetic Algorithm by Fuzzy Logic and Belief Functions for Job Shop Scheduling, *IEEE Transactions on Systems, Man and Cybernetics*, 30, 812-820.

Hamamato, S., Yih, Y. and Salvendy, G. (1999). Development and Validation of Genetic Algorithm-Based Facility Layout. A Case Study in the Pharmaceutical Industry, *International Journal of Production Research*, 37, 749-768.

Hoda, E., Viskvas, P. and Ben, A. (2000). Scheduling of Manufacturing Systems Under Dual-Resource Constraints Using Genetic Algorithms, *Journal of Manufacturing Systems*, 19, 186-201.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge.

Homaifar, A., Guan, S. and Liepins, G.E. (1992). Schema Analysis of the Traveling Salesman Problem Using Genetic Algorithms, *Complex Systems*, 6, 533-552.

Huntley, C.L. and Brown, D.E. (1991). Parallel Heuristics for Quadratic Assignment Problems, *Computers & Operations Research*, 18, 275-289.

Hussain, S.A. and Sastry, V.U.K. (1999). Application of Genetic Algorithm to Stochastic Single Machine Scheduling with Earliness and Tardiness Costs, International *Journal of Computer Mathematics*, 70, 383-392.

Ip, W.H., Li, Y., Man, K.F. and Tang, K.S. (2000). Multi-Product Planning and Scheduling Using Genetic Algorithm Approach, *Computers & Industrial Engineering*, 38, 283-296.

Ishibuchi, H., Yamamoto, N., Murata, T. and Tanaka, H. (1994). Genetic Algorithms and Neighborhood Search Algorithms for Fuzzy Flow Shop Scheduling Problems, *Fuzzy Sets and Systems*, 67, 81-100.

Islier, A.A. (1998). A Genetic Algorithm Approach for Multiple Criteria Facility Layout Design, *International Journal of Production Research*, 36, 1549-1569.

Jahangiran, M. and Conray, G.V. (2000). Intelligent Dynamic Scheduling System: The Application of Genetic Algorithms, *Integrated Manufacturing Systems*, 11, 247-257.

Jain, A.K. and Elmaraghy, H.A. (1997). Single Process Plan Scheduling with Genetic Algorithms, *Production Planning & Control*, 8, 363-376.

Jakobs, S. (1996). On Genetic Algorithms for the Packing of Polygons, *European Journal of Operational Research*, 88, 165-181.

Janiak, A. and Portmann, M.C. (1998). Genetic Algorithm for the Permutation Flow Shop Scheduling Problem with Linear Model of Operations, *Annals of Operations Research*, 83, 95-114.

Janikow, C. and Michalewicz, Z. (1991). *An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms, Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, 31-36.

Jog, P., Jung, Y.S. and Gucht, D.V. (1991). Parallel Genetic Algorithms Applied to the Traveling Salesman Problem. *SIAM Journal on Optimization*, 4, 515-529.

Kapsalis, A., Rayward-Smith, V.J. and Smith, G.D. (1993). Solving the Graphical Steiner Tree Problem Using Genetic Algorithms, *Journal of Operational Research Society*, 44, 397-406.

Kim, H., Nara, K. and Gen, M. (1994). A Method for Maintenance Scheduling Using GA Combined with SA, *Computers & Industrial Engineering*, 27, 477-480.

Kim, Y.K., Kim, Y. and Kim, Y.J. (2000). Two-Sided Assembly Line Balancing: A Genetic Algorithm Approach, *Production Planning & Control*, 11, 44-53.

Kim, Y.K., Kim, Y.J. and Kim, Y. (1996). Genetic Algorithms for Assembly Line Balancing with Various Objectives, *Computers & Industrial Engineering*, 30, 397-409.

Kimms, A. (1999). A Genetic Algorithm for Multi-Level Multi-Machine Lot Sizing and Scheduling, *Computers & Operations Research*, 26, 829-848.

Kochhar, J.S. and Heragu, S.S. (1999). Facility Layout Design in a Changing Environment, International *Journal of Production Research*, 37, 2429-2446.

Kochhar, J.S., Foster, B.T. and Heragu, S.S. (1998). HOPE: A Genetic Algorithm for the Unequal Area Facility Layout Problem, *Computers & Operations Research*, 25, 583-594.

Kopfer, H., Pankratz, G. and Erkens, E. (1994). Vehicle Routing by Genetic Algorithms, *OR Spektrum*, 16, 21-31.

Kragelund, L.V. (1997). Solving a Timetabling Problem Using Hybrid Genetic Algorithms, *Software: Practice & Experience*, 27, 1121-1134.

Kroger, B. (1995). Guillotineable Bin Packing: A Genetic Approach, European *Journal of Operational Research*, 84, 645-661.

Kumar, N.S.H. and Srinivasan, G. (1996). A Genetic Algorithm for Job Shop Scheduling – A Case Study, *Computers in Industry*, 31, 155-160.

Lam, F.S.C., Lin, B.C., Sriskandrarajah, C. And Yan, H. (1999). Scheduling to Minimize Product Design Time Using a Genetic Algorithm, *International Journal of Production Research*, 37, 1369-1386.

Lee, C.Y. and Choi, J.Y. (1995). A Genetic Algorithm for Job Sequencing Problems with Distinct Due Dates and General Early-Tardy Penalty Weights, *Computers & Operations Research*, 22, 857-869.

Lee, C.Y. and Kim, S.J. (1995). Parallel Genetic Algorithms for the Earliness-Tardiness Job Scheduling problem with General penalty Weights, *Computers & Industrial Engineering*, 28, 231-243.

Lee, C.Y., Piramuthu, S. and Tsai, Y.K. (1997). Job Shop Scheduling with a Genetic Algorithm and Machine Learning, *International Journal of Production Research*, 35, 1171-1191.

Lee, K.Y. and Yang, F.F. (1998). Optimal Reactive Power Planning Using Evolutionary Algorithms: A Comparative Study for Evolutionary Programming, Evolutionary Strategy, Genetic Algorithm and Linear Programming, *IEEE Transactions on Power Systems*, 13, 101-108.

Lee, S.G., Khoo, L.P. and Yin, X.F. (2000). Optimizing an Assembly Line Through Simulation Augmented by Genetic Algorithms, *The International Journal of Advanced Manufacturing Technology*, 16, 220-228.

Leu, Y.Y., Matheson, L.A. and Rees, L.P. (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic Generated Initial Populations and Multiple Evaluation Criteria, *Decision Sciences*, 25, 581-606.

Levine, D. (1996). Application of a Hybrid Genetic Algorithm to Airline Crew Scheduling, *Computers & Operations Research*, 23, 547-558.

Levitin, G. and Rubinovitz, J. (1993). Genetic Algorithms for Linear and Cyclic Assignment Problem, *Computers & Operations Research*, 20, 575-586.

Li, Y., Man, K.F., Tang, K.S., Kwong, S. and Ip, W.H. (2000). Genetic Algorithm to Production Planning and Scheduling Problems for Manufacturing Systems, *Production Planning & Control*, 5, 443-458.

Liaw, C.F. (2000). A Hybrid Genetic Algorithm for the Open Shop Scheduling Problem, *European Journal of Operational Research*, 124, 28-42.

Lin, W., Delgado, F.J.G., Gause, D.C. and Vassiliadis, S. (1995). Hybrid Newton-Raphson Genetic Algorithm for the Traveling Salesman Problem, *Cybernetics and Systems*, 26, 387-412.

Liu, J. And Tang, L. (1999). A Modified Genetic Algorithm for Single Machine Scheduling, *Computers & Indusrial Engineering*, 37, 43-46.

Louis, S.L and Li, G. (2000). Case Injected Genetic Algorithms for Traveling Salesman Problems, *Information Sciences*, 122, 201-226.

Maimon, O.Z. and Braha, D. (1998). A Genetic Algorithm Approach to Scheduling PCBs on a Single Machine, *International Journal of Production Research*, 36, 761-784.

Malmborg, C.J. (1996). A Genetic Algorithm for Level Based Vehicle Scheduling, *European Journal of Operational Research*, 93, 121-134.

Man, K.F., Tang, K.S. and Kwong, S. (1996). Genetic Algorithms: Concepts and Applications, *IEEE Transactions on Industrial Electronics*, 43, 519-533.

Maturana, F., Gu, P., Naumann, A. and Norrie, D.H. (1996). Object-Oriented Job-Shop Scheduling Using Genetic Algorithms, *Computers in Industry*, 32, 281-294.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd edition, Springer-Verlag, Berlin.

Min, L. and Cheng, W. (1999). A Genetic Algorithm for Minimizing the Makespan in the Case of Scheduling Identical Parallel Machines, *Artificial Intelligence in Engineering*, 13, 399-403.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, MIT Press, Massachusetts.

Monfroglio, A. (1996a). Hybrid Genetic Algorithms for Timetabling, International *Journal of Intelligent Systems*, 11, 477-524.

Monfroglio, A. (1996b). Timetabling Through Constrained Heuristic Search and Genetic Algorithms, *Software: Practice & Experience*, 26, 251-279.

Morad, N. and Zalzala, A. (1999). Genetic Algorithms in Integrated Process Planning and Scheduling , *Journal of Intelligent Manufacturing*, 10, 169-180.

Mori, M. And Tseng, C.C. (1997). A Genetic Algorithm for Multi-Mode Resource Constrained Project Scheduling Problem, *European Journal of Operational Research*, 100, 134-141.

Moutaz, K., Michalewicz, Z. and Wilmot, M. (1998). The Use of Genetic Algorithms to Solve Economic Lot size Scheduling Problem, *European Journal of Operational Research*, 110, 509-524.

Murata, T., Ishibuchi, H. and Tanaka, H. (1996). Genetic Algorithms for Flow Shop Scheduling Problems, *Computers & Industrial Engineering*, 30, 957-968.

Nagar, A., Heragu, S.S. and Haddock, J. (1996). A Combined Branch and Bound and Genetic Algorithm Based Approach for a Flowshop Scheduling Problem, *Annals of Operations Research*, 63, 397-414.

Nissen, V. (1992). Solving the Quadratic Assignment Problem with Clues from Nature, *IEEE Transactions on Neural Networks*, 5, 66-72.

Nitin, J., Bagchi, T.P. and Wagneur, E. (2000). Flow Shop Scheduling by Hybridized GA: Some New Results, *International Journal of Industrial Engineering*, 7, 213-223.

Norman, B.A. and Bean, J.C. (1999). A Genetic Algorithm Methodology for Complex Scheduling Problems, *Naval Research Logistics*, 46, 199-211.

Odetayo, M.O. (1995). Knowledge Acquisition and Adaptation: Genetic Approach, *Expert Systems*, 12, 3-13.

Onwubolu, G.C. and Mutingi, M. (1999). Genetic Algorithm for Minimizing Tardiness in Flow-Shop Scheduling, *Production Planning & Control*, 10, 462-471.

Painton, L. and Campball, J. (1995). Genetic Algorithms in Optimization of System Reliability, *IEEE Transactions on Reliability*, 44, 172-178.

Ponnambalam, S.G., Aravindan, P. and Naidu, G.M. (2000). A Multi-Objective Genetic Algorithm for Solving Assembly Line Balancing Problem, *The International Journal of Advanced Manufacturing Technology*, 16, 341-352.

Potvin, J.Y. (1996). Genetic Algorithms for the Traveling Salesman Problem, *Annals of Operations Research*, 63, 339-360.

Qi, J.G., Burns, G.R. and Harrison, D.K. (2000). The Application of Parallel Multipopulation Genetic Algorithms to Dynamic Job Shop Scheduling, *The International Journal of Advanced Manufacturing Technology*, 16, 609-615.

Qu, L. and Sun, R. (1999). A Synergetic Approach to Genetic Algorithms for Solving Traveling Salesman Problem, *Information Sciences*, 117, 267-283.

Rajasekharan, M., Peters, B.A. and Yang, T. (1998). A Genetic Algorithm for Facility Layout Design in Flexible Manufacturing Systems, *International Journal of Production Research*, 36, 95-110.

Reeves, C. (1997). Genetic Algorithms for Operations Researcher, *Journal on Computing*, 9, 231-250.

Reeves, C. (1993). *Modern Heuristic Methods for Combinatorial Problems*, Blackwell Scientific Publications, Oxford.

Reeves, C.R. (1995). A Genetic Algorithm for Flow Shop Sequencing, *Computers & Operations Research*, 22, 5-13.

Reeves, C.R. (1996). Hybrid Genetic Algorithms for Bin-Packing and Related Problems, *Annals of Operations Research*, 63, 371-396.

Rubinovitz, J. and Levitin, G. (1995). Genetic Algorithm for Assembly Line Balancing, *International Journal of Production Economics*, 41, 343-354.

Sabuncuoğlu, I., Erel, E. and Tanyer, M. (1999). Assembly Line Balancing Using Genetic Algorithms, *Journal of Intelligent Manufacturing*, 11, 295-310.

Sakawa, M. and Kubota, R. (2000). Fuzzy Programming for Multiobjective Job shop scheduling with Fuzzy Processing Time and Fuzzy Due Date Through Genetic Algorithms, *European Journal of Operational Research*, 120, 393-407.

Sakawa, M. and Mori, T. (1999). An efficient Genetic Algorithm for Job-shop Scheduling Problems with Fuzzy Processing Time and Fuzzy Due Date, *Computers & Industrial Engineering*, 36, 325-342.

Sakawa, M. and Yauchi, K. (1998). Coevolutionary Genetic Algorithms for Nonconvex Nonlinear Programming Problems, *Cybernetics and Systems*, 29, 885-910.

Saltouros, M.P., Verentziotis, E.A., Markai, M.E., Theologou, M.E. and Venieris, I.S. (2000). An Efficient Hybrid Genetic Algorithm for Finding Near-Optimal Steiner Trees: An Approach to Routing of Multipoint Connections, *International Journal of Computers & Applications*, 22, 159-165.

Sasaki, M., Gen, M. and Yamashiro, M. (1995). A Method for Solving Fuzzy de Novo Programming Problem by Genetic Algorithms, *Computers & Industrial Engineering*, 29, 507-511.

Shi, G. (1997). A Genetic Algorithm Applied to a Classic Job-Shop Scheduling Problem, *International Journal of Systems Science*, 28, 25-32.

Srinivas, M. and Patnaik, L.M. (1994). Genetic Algorithms: A Survey, *Computer*, 27, 17-26.

Suresh, G., Vinod, V.V. and Sahu, S. (1996). A Genetic Algorithm for Assembly Line Balancing, *Production Planning & Control*, 7, 38-46.

Suresh, G., Vinod, V.V. and Sahu, S. (1995). A Genetic Algorithm for Facility Layout, International *Journal of Production Research*, 33, 3411-3423.

Taguchi, T., Ida, K. and Gen, M. (1997). Method for Solving Nonlinear Goal Programming with Interval Coefficients Using Genetic Algorithm, *Computers & Industrial Engineering*, 33, 597-600.

Tam, K.Y. and Chan, S.K. (1998). Solving Facility Layout Problems with Geometric Constraints Using Parallel Genetic Algorithms : Experimentation and Findings, *International Journal of Production Research*, 36, 3253-3272.

Tam, K.Y. (1992). Genetic Algorithms, Function Optimization and Facility Layout Design, *European Journal of Operational Research*, 63, 322-346.

Tate, D.M. and Smith, A.E. (1995). A Genetic Approach to the Quadratic Assignment Problem, *Computers & Operations Research*, 22, 73-84.

Thiel, J. and Voss, S. (1994). Some Experiences on Solving Multiconstraint Zero-One Knapsack Problems with Genetic Algorithms, *INFOR*, 32, 226-242.

Todd, D. and Sen, P. (1999). Distributed Task Scheduling and Allocation Using Genetic Algorithms, *Computers & Industrial Engineering*, 37, 47-50.

Tomassini, M. (1995). A Survey of Genetic Algorithms, *Annual Reviews of Computational Physics*, 3, 87-118.

Tsujimura, Y., Gen, M. and Kubota, E. (1995). Solving Fuzzy Assembly-Line Balancing Problem with Genetic Algorithms, *Computers & Industrial Engineering*, 29, 543-547.

Uckun, S., Bagchi, S., Kawamura, K. and Miyabe, Y. (1993). Managing Genetic Search in Job Shop Scheduling, *IEEE Expert*, 8, 15-25.

Ulusoy, G., Sivrikaya, F. and Bilge, Ü. (1997). A Genetic Algorithm Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles, *Computers & Operations Research*, 24, 335-352.

Urdeneta, A.J., Gomez, J.F., Sorrentino, E., Flores, L. and Diaz, R. (1999). A Hybrid Genetic Algorithm for Optimal Reactive Power Planning Based Upon Successive Linear Programming, *IEEE Transactions on Power Systems*, 14, 1292-1298.

Vancza, J. and Markus, A. (1991). Genetic Algorithms in Process Planning, *Computers in Industry*, 17, 181-194.

Wang, D. (1997). An Inexact Approach for Linear Programming Problems with Fuzzy Objective and Resources, *Fuzzy Sets and Systems*, 89, 61-68.

Wang, D., Gen, M. and Cheng, R. (1999). Scheduling Grouped Jobs on Single Machine with Genetic Algorithm, *Computers & Industrial Engineering*, 36, 309-324.

Webster, S., Jog, P.D. and Gupta, A. (1998). A Genetic Algorithm for Scheduling Job Families on a Single Machine with Arbitrary Earliness/Tardiness Penalties and an Unrestricted Common Due Date, *International Journal of Production Research*, 36, 2543-2551.

Wren, A. and Wren D.O. (1995). A Genetic Algorithm for Public Transport Driver Scheduling, *Computers & Operations Research*, 22, 101-110.

Wu, Z. and Zhao, C. (2000). Genetic Algorithm Approach to Job Shop Scheduling and Its Use in Real-Time Cases, *International Journal of Computer Integrated Manufacturing*, 13, 422-429.

Yip-Hoi, D. and Dutta, D. (1996). A Genetic Algorithm Application for Sequencing Operations in Process Planning for Parallel Machining, *IEEE Transactions*, 28, 55-68.

Yokota, T., Gen, M. and Li, Y. (1996). Genetic Algorithm for Non-Linear Mixed Integer Programming Problems and Its Applications, *Computers & Industrial Engineering*, 30, 905-918.

Yokota, T., Gen, M., Li, Y. and Kim, C.E. (1996). A Genetic Algorithm for Interval Nonlinear Integer Programming Problem, *Computers & Industrial Engineering*, 31, 913-918.

Yokota, T., Gen, M., Taguchi, T. and Li, Y. (1995). A Method for Interval 0-1 Nonlinear Programming Problem Using a Genetic Algorithm , *Computers & Industrial Engineering*, 29, 531-536.

Özgür Yeniay, Has been a research assistant in the Department of Statistics, Hacettepe University, Beytepe, Ankara, since 1992. He received his MSc degree on the travelling salesman problems in 1994 and his PhD degree on the genetic algorithms in 1999, both from Hacettepe University. His main research interests are on the applications of modern heuristic methods such as genetic algorithms, tabu search and simulated annealing.