DOI 10.1007/s10115-003-0082-0

Knowledge and Information Systems (2003) 5: 503–511

Knowledge and Information Systems

A Heuristic Genetic Algorithm for Solving Resource Allocation Problems

Zne-Jung Lee[1], Shun-Feng Su[1], Chou-Yuan Lee[1], Yao-Shan Hung[2]

[1]Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan

[2]Chung-Shan Institute of Science and Technology, Taiwan

Abstract. In the paper, a heuristic genetic algorithm for solving resource allocation problems is proposed. The resource allocation problems are to allocate resources to activities so that the fitness becomes as optimal as possible. The objective of this paper is to develop an efficient algorithm to solve resource allocation problems encountered in practice. Various genetic algorithms are studied and a heuristic genetic algorithm is proposed to ameliorate the rate of convergence for resource allocation problems. Simulation results show that the proposed algorithm gives the best performance.

Keywords: Genetic algorithm; Heuristics; Optimization problems; Resource allocation problems

1. Introduction

Resource allocation problems are optimization problems with constraints. They have been found in many fields such as load distribution, computer scheduling and production planning (Ibarraki and Katohi, 1988). It is difficult to solve this class of problems directly (Sahni and Gonzales, 1976; Hammer et al., 1984; Lloyd and Witsenhausen, 1986). Various methods for solving resource allocation problems have been reported in the literature (Ibarraki and Katohi, 1988; Bjorndal et al., 1995). Those methods usually use graph search approaches and result in exponential computational complexities. Recently, genetic algorithms have been widely used as search algorithms in various applications and have demonstrated satisfactory performance. Genetic algorithms are based on the mechanism of natural selection to search for a solution that optimizes a given fitness function. Vast genetic algorithms have been proposed in the literature (Olsen, 1994; Tate and Smith, 1995; Bäck et al., 1997) and shown superior performance over other

Received 22 August 2000
Revised 27 October 2001
Accepted 24 May 2002

methods. One naturally expects that genetic algorithms can be used in solving resource allocation problems. However, if not carefully designed, genetic algorithms may have a very slow convergence rate for resource allocation problems (Lee et al., 2000). In this paper, a heuristic genetic algorithm (HGA) for solving resource allocation problems is proposed. HGA is based on genetic algorithms with the use of problem-specific knowledge, and is employed to ameliorate the rate of convergence for resource allocation problems.

This paper is organized as follows. In Section 2, the considered resource allocation problems are formulated. In Section 3, various genetic algorithms are introduced and an HGA is proposed to improve the search efficiency. Thereafter, those genetic algorithms are employed to solve resource allocation problems and their results are reported in Section 4. Finally, Section 5 concludes the paper.

2. Formulation of Resource Allocation Problems

The resource allocation problems addressed in this paper are an optimization problem with constraints. It is to allocate resources to activities so that the assignment can lead to the best fitness value. The following two assumptions are made for resource allocation problems. The first one is that there are N resources and N activities, and all resources and activities must be one-to-one allocated. The second assumption is that the individual probability (K_{ij}) of the fulfill of the i-th activity when allocating the j-th resource to the i-th activity is known for all i and j. We consider the resource allocation problems as to minimize the fitness function:

$$F(X) = \sum_{i=1}^{N} COST(i) \left[\prod_{j=1}^{N} [1 - K_{ij}]^{X_{ij}} \right] \tag{1}$$

subject to the assumption that all resources and activities must be one-to-one allocated; that is,

$$\sum_{j=1}^{N} X_{ij} = 1, \quad i = 1, 2, \ldots, N, \text{ and}$$

$$\sum_{i=1}^{N} X_{ij} = 1, \quad j = 1, 2, \ldots, N \tag{2}$$

X_{ij} is a Boolean value for the allocation of the j-th resource to the i-th activity, where $X_{ij} = 1$ indicates that the j-th resource is allocated to the i-th activity and $COST(i)$ is the penalty of the i-th activity not being fulfilled by its assigned resource. It should be noted that equation (1) can be equivalently rewritten as some simple forms. However, since genetic algorithms do not use any auxiliary mathematical operations of the fitness function, we keep this form for better understanding. Besides, there are also other fitness functions for various resource allocation problems. Such fitness functions can also be adopted in our algorithm.

3. Various Genetic Algorithms

It has been shown that the resource allocation problems are 0-1 programming problems (Ibarraki and Katohi, 1988) and is difficult to solve directly when the numbers of

resources are large (Hammer et al., 1977). Various methods such as dynamic programming, separable convex objective functions, and graph theory have been employed to find the best solutions for this kind of problem (Ibarraki and Katohi, 1988; Bjorndal et al., 1995). Recently, genetic algorithms have often been employed. Fundamentally, genetic algorithms or, more generally, evolutionary algorithms (Bäck et al., 1997) are search algorithms, which adopt the mechanism of natural selection to search for the best solution from candidates. Genetic algorithms can handle almost any kinds of objective functions with linear or nonlinear constraints without any auxiliary mathematical operations, such as derivatives or matrix inverse. Tate and Smith (1995) employed genetic approaches to quadratic assignment problems and showed that genetic algorithms can outperform other methods without introducing extra computational overhead. Other genetic algorithms have also been employed in various applications and have shown that they are suitable for solving optimization problems (Olsen, 1994). However, if genetic operations are not carefully designed, the rate of convergence may be very low. In this study, we employ various genetic algorithms to solve resource allocation problems and their search performances are investigated. An HGA is also proposed to ameliorate the rate of convergence for resource allocation problems.

3.1. Genetic Algorithms with Penalty Functions

In a genetic algorithm, variables are represented as genes on a chromosome. Through natural selection and genetic operators, mutation and recombination, chromosomes with better fitness are found. Usually, a simple genetic algorithm (SGA) consists of four steps. The first step is to initialize a population by randomly selecting a set of input chromosomes (strings) encoding possible solutions. The second step is to evaluate the fitness of each chromosome in the population. The third step is to create new chromosomes by mating current chromosomes with the use of mutation and recombination operations. The fourth step is to evaluate the new chromosomes and to delete the chromosomes with the lowest fitness from the population. The general structure of an SGA (Goldberg, 1989; Gen and Cheng, 1997) is shown in the following, where $P(t)$ and $C(t)$ are parents and offspring in generation t.

Procedure: Simple Genetic Algorithm
Begin
$t \leftarrow 0$;
Initialize $P(t)$;
Evaluate $P(t)$;
While (not match the termination conditions) do
Recombine $P(t)$ to yield $C(t)$;
Evaluate $C(t)$;
Select $P(t+1)$ from $P(t)$ and $C(t)$ based on the fitness;
$t \leftarrow t+1$;
End
End

The SGA is a general problem solver for unconstrained optimization problems. When the problems in hand are constrained optimization problems, some approaches handle constraints by penalizing the fitness of illegal chromosomes. In this kind of approach,

a constrained problem is transformed into an unconstrained problem by introducing a penalty function to the fitness function. By degrading the fitness ranking of infeasible solutions, the penalty method yields more rapid optimization and avoids premature termination. Olsen (1994) has shown the effectiveness of using penalty functions. Since this resource allocation problem has two constraints, we can adopt penalty functions to solve this problem.

In this study, the chromosome $X = (x_{11} \ldots x_{1N} x_{21} \ldots x_{2N} x_{N1} \ldots x_{NN})$ is used for encoding solutions, where N is the number of resources to be allocated. When $x_{ij} = 1$, it represents that the j-th resource is allocated to the i-th activity. The traditional roulette wheel approach is employed in the selection of parent chromosomes from the population. For crossover operations, the one cut point crossover operation is used. The cut point is selected randomly. Mutation operators alter one or more genes with a pre-specified mutation probability.

The objective of the optimization problem is to find the optimal solution that can minimize the fitness $Eval(X)$ without considering the feasibility of possible candidates by introducing a penalty function into the fitness function. Thus, the evaluation fitness function (*Eval*) is defined as:

$$Eval(X) = F(X)Pe(X) \tag{3}$$

where $F(X)$ is the fitness function used to evaluate a set of allocation X, and $Pe(X)$ is the penalty function used to degrade infeasible allocation. The penalty function used in our study is:

$$Pe(X) = \max\left\{1, \left[\sum_{i=1}^{N} W_1(x_{i.}) + \sum_{j=1}^{N} W_1(x_{.j})\right]\right\} \tag{4}$$

where $x_{i.} = [x_{i1} x_{i2} \ldots x_{iN}]$, $x_{.j} = [x_{1j} x_{2j} \ldots x_{Nj}]$, and $W_1(x)$ is the feasibility measure of x and defined as $W_1(x) = |d(x, \mathbf{0}) - 1|$ with $d(a, b)$ as the Hamming distance of two binary vectors a and b; i.e., $d(a, b) = \sum_{i=1}^{n} |a_i - b_i|$. Here $0 = [0, 0, \ldots, 0]$. It is easy to verify that when x is feasible $W_1(x) = 0$ and $Pe(X) = 1$. When there is an infeasible $x_{i.}$ for some i-th (i.e. $W_1(x_{i.}) \geq 1$), there must also exist an $x_{.j}$ being infeasible for some j. In other words, when X is infeasible, then $Pe(X) > 1$. In our previous study (Lee et al., 2000), other penalty function has also been considered. Simulation results showed that genetic algorithms with a penalty function like equation (4) could have better performance.

3.2. Order Representation Approach (ORA) and Constrained Search

This representation uses N numerical genes and each gene with an integer number form 1 to N to represent activity–resource pairs. Take $N = 4$ as an example. With the binary gene representation, the chromosome is $X = (0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1)$. Due to the constraints of completeness and uniqueness required by the allocation (i.e., equation. (2)), the chromosome can be represented as (2 3 1 4), which represents a feasible solution where activity 1 is allocated to resource 2, activity 2 is allocated to resource 3 and so on. This type of representation is called the order representation. For such a representation, the partially mapped crossover (PMX) (Goldberg, 1989) and the inversion mutation (Gen and Cheng, 1997) operators are implemented. Such a genetic algorithm is simply referred to as the order representation approach (ORA).

The idea of the PMX operation is to generate mapping relations so that the offspring can be repaired accordingly and become feasible. In other words, instead of introducing a penalty function in the fitness function, this approach is to restrict the search in the feasible solutions. PMX has been shown to be effective in many applications (Goldberg, 1989). The PMX algorithm is stated as (Gen and Cheng, 1997); (Goldberg, 1989):

Step 1: Select substrings from parents at random.

Step 2: Exchange substrings between two parents to produce proto-offspring.

Step 3: Determine the mapping relationship from the exchanged substrings.

Step 4: Repair proto-offspring with the mapping relationship.

To see the procedure, consider two chromosomes:

$$A = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$
$$B = 4\ 5\ 6\ 9\ 1\ 2\ 7\ 3\ 8$$

First, two positions, e.g., the 3rd and the 6th positions, are selected at random to define a substring in a chromosome, and the defined substrings in those two chromosomes are then exchanged to generate the proto-offspring as:

$$A = 1\ 2\ \underline{6\ 9\ 1\ 2}\ 7\ 8\ 9$$
$$B = 4\ 5\ \underline{3\ 4\ 5\ 6}\ 7\ 3\ 8$$

Then the mapping relationship between those two substrings can be established as:

(A's genes) $2 \leftrightarrow 6 \leftrightarrow 3$ (B's genes)
$9 \leftrightarrow 4$
$1 \leftrightarrow 5$

Notice that the mapping relations $6 \leftrightarrow 3$ and $2 \leftrightarrow 6$ have been merged into $2 \leftrightarrow 6 \leftrightarrow 3$. Finally, the proto-offspring are repaired according to the above mapping lists. The resultant feasible offspring are:

$$A' = 5\ 3\ 6\ 9\ 1\ 2\ 7\ 8\ 4$$
$$B' = 9\ 1\ 3\ 4\ 5\ 6\ 7\ 2\ 8$$

The algorithm of the inversion mutation operation is (Gen and Cheng, 1997):

Step 1: Select two positions within a chromosome at random.

Step 2: Invert the substring between these two positions.

Consider a chromosome:

$$A = 1\ 2\ \underline{3}\ 4\ 5\ \underline{6}\ 7\ 8\ 9.$$

In a mutation process, the 3rd and the 6th positions are randomly selected. If the mutation operation is meant to be performed, then the offspring become:

$$A' = 1\ 2\ \underline{6}\ 5\ 4\ \underline{3}\ 7\ 8\ 9.$$

3.3. The Heuristic Genetic Algorithm

In the proposed HGA, a problem-specific heuristic procedure is embedded into the algorithm to improve the search performance. The procedure of the proposed algorithm is shown as follows where $P(t)$ and $D(t)$ are parents and offspring in generation t.

Procedure: The Proposed Algorithm (HGA)
Begin
$t \leftarrow 0$;
Initialize population $P(t)$;
While (not match the termination conditions) do
Perform crossover and mutation on $P(t)$ to yield $C(t)$;
Perform heuristic search on $C(t)$ to yield $D(t)$;
Evaluate $D(t)$;
Select $P(t+1)$ from $P(t)$ and $C(t)$ based on the fitness;
$t \leftarrow t+1$;
End
End

In the algorithm, the chromosome is represented as ORA. The PMX and the inversion mutation operators are also used. The difference between SGA and our HGA is the introduction of heuristic search as a local search procedure after new chromosomes are generated. It has been shown that a problem-specific heuristics procedure can improve the search efficiency (Kolen and Pesch, 1994; Merz and Freisleben, 2000). In our study, a heuristic search procedure is applied to improve the rate of convergence. In order to minimize the fitness function of equation (1), the algorithm locally modifies genes to be the activity–resource pairs with highest available $K_{ij}{}^{*}COST(i)$, and it continues to allocate resources to activities until a feasible solution is constructed.

Based on the above concept, offspring are locally modified with heuristics. The better the chromosome is, the more likely it is the optimal solution. Hence, the above local search procedure is conducted based on their fitness values, and the selected probability (P_i) for the i-th chromosome is defined as

$$P_i = \frac{e^{-F_i(X)}}{\sum_{i=1}^{u} e^{-F_i(X)}} \tag{5}$$

where $F_i(X)$ is the fitness of the i-th chromosome and u is the population size. If the chromosome is selected for performing heuristic search, a randomly generated number (i) between 1 and N is chosen as the starting gene in the chromosome. The chromosome is forward modified from the starting gene to the N-th gene or backward modified to the first gene according to a predefined value (q_1), where $0 \leq q_1 \leq 1$. The chromosome is forward modified if $r > q_1$ is satisfied, where r is randomly obtained from the unit uniform distribution. In forward modification, the genes from the first up to the $(i-1)$-th genes are fixed and the i-th gene's value is replaced by the unallocated activity j which has the available highest $K_{ij}{}^{*}COST(i)$. The replacement process is repeatedly performed up to the N-th gene. In backward modification, the allocations from the $(i+1)$-th up to the N-th gene are fixed and the i-th gene's value down to the first gene's value are consecutively replaced by the unallocated activity with their available highest $K_{ij}{}^{*}COST(i)$.

Table 1. The *COST* values and K_{ij} for $N = 4$

Activity	K_{ij}				*COST*
	Resource 1	Resource 2	Resource 3	Resource 4	
Activity 1	0.5	0.6	0.2	0.1	100
Activity 2	0.7	0.6	0.25	0.15	85
Activity 3	0.1	0.3	0.75	0.7	50
Activity 4	0.1	0.35	0.8	0.85	65

Table 2. The simulation results for $N = 4$. Results are averaged over 10 trials

Algorithms	Averaged fitness	Averaged converged generation	Percentage of convergence
GA with penalty function	98.46	N/A	60%
ORA	87.75	7.26	100%
HGA	87.75	1.2	100%

N/A, not available.

4. Simulation Results

Several scenarios are considered to test the performance of various genetic algorithms. The size of the initial population is selected as the same as the number of resources (N) considered. The simulation is conducted for 10 trials and the averaged results are reported. The parameters used are the crossover probability $P_c = 0.1$ and the mutation probability $P_m = 0.05$. The first example consists of four resources and activities, and the corresponding *COST*s of activities and K_{ij} are shown in Table 1. This is a very simple example used to demonstrate the effects of those considered algorithms. The simulation is conducted under the maximum generation number, max _gen = 100. Results from various genetic algorithms are listed in Table 2. The ORA and the HGA can 100% converge to the best solution, but the GA with a penalty function only has 60% of convergence. Since the GA with a penalty function cannot 100% converge to the best solution, the averaged converged generation is marked as not available (N/A).

Since the GA with a penalty function seems to have worse performance, we focused on the comparison between the ORA and the HGA. The scenario of $N = 6$ is used to compare the performance of ORA and HGA. HGA found the best solution with fitness 17.7883 after 4.7 generations in 0.611 s (Pentium 1 GHz processor), while ORA also found the best solution after 12.8 generations in 1.152 s. HGA may need more computing time than ORA does in one generation, but the overall search efficiency for HGA is still superior to that of ORA. Furthermore, several scenarios are considered to compare the performance of ORA, HGA, and the simulated annealing (SA) algorithm. The SA optimization approach is employed to search for the optimal solution by making successive random modifications for candidate solutions and is also widely used for solving combinatorial optimization problems (Lin et al., 1993; Van and Arts, 1992). It is analogy to the annealing process of solids, which is the process of healing up a solid to a high temperature and then cooling down gradually (Van

Table 3. Compare the best fitness of randomized scenarios obtained by various search algorithms. Results are averaged over 10 trials

	$N = 6$	$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$
ORA	17.7883	26.9452	61.9631	73.5337	108.9688	137.1167
SA	17.7883	26.9452	49.2844	54.8355	94.6685	102.6171
HGA	17.7883	26.9452	49.2844	54.8355	77.4033	81.4832

Laarhoven and Arts, 1992). In our comparison, we simply stopped those algorithms after a fixed time of running. Experiments were still run on PCs with a Pentium 1 GHz processor, and were stopped after one hour of running. The results are listed in Table 3. It is easy to see that the HGA can always have the best search efficiency among these algorithms.

5. Conclusion

In this paper, we have presented a heuristic genetic algorithm (HGA) for solving resource allocation problems. HGA includes domains specific knowledge in genetic algorithms for solving resource allocation problems. From simulations, it can be found that the proposed algorithm can improve search efficiency and shows superiority over existing search algorithms.

Acknowledgements. We thank anonymous reviewers for their very useful comments and suggestions. This work was supported in part by the National Science Council of Taiwan, ROC, under the grant NSC-89-2218-E-011-002.

References

Bäck T, Hammel U, Schwefel HP (1997) Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation 1(1): 1–17

Bjorndal AMH, Caprara A, Cowling PI et al (1995) Some thoughts on combinatorial optimisation. European Journal of Operational Research 83: 253–270

Gen Mitsuo and Cheng Runwei (1997) Genetic algorithms, engineering design. Wiley, New York, pp 1–2

Goldberg DA (1989) Genetic algorithms in search, optimization, and machine learning. Adision-Wesley, Reading, MA, pp 170–174

Hammer PL, Hansen P, Simeone B (1984) Roof duality, complementation and persistency in quadratic 0-1 optimization. Mathematical Programming 28: 121–155

Ibarraki Toshihide and Katohi Naoki (1988) Resource allocation problems. MIT Press, Cambridge, MA, pp 1–10

Kolen A, Pesch E (1994) Genetic local search in combinatorial optimization. Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science 48: 273–284

Lee ZJ, Lee CY, Su SF (2000) A fuzzy-genetic based decision aided system for naval weapon–target assignment problems. Proceedings of 2000 ROC automatic control conference 1: 163–168

Lin FT, Kao CY, Hsu CC (1993) Applying the genetic approach to simulated annealing in solving some NP-hard problems. IEEE Transactions on Systems, Man and Cybernetics 23(6): 1752–1767

Lloyd SP, Witsenhausen HS (1986) Weapon allocation is NP-complete. IEEE summer simulation conference, Reno, Nevada

Merz P, Freisleben B (2000) Fitness landscape analysis and memetic algorithms for quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4(4): 337–352

Olsen A (1994) Penalty functions and the knapsack problem. Proceedings of the first IEEE conference on evolutionary computation 6: 554–558

Sahni S, Gonzales T (1976) P-complete approximation problem. ACM Journal 23: 556–565

Tate DM, Smith AE (1995) A genetic approach to the quadratic assignment problem. Computers Ops. Res. 22(1): 73–83
Van Laarhoven PJM, Arts EHL (1992) Simulated annealing: theory and applications. Kluwer Academic, Dordrecht, pp 9–12

Correspondence and offprint requests to: Shun-Feng Su, Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan. Email: su@mig1.ee.ntust.edu.tw