

Short Paper

A New Efficient Encoding Mode of Genetic Algorithms for the Generalized Plant Allocation Problem

YOUNG-CHANG HOU AND YING-HUA CHANG

Department of Information Management

National Central University

Chungli, 320 Taiwan

This study proposes a novel, efficient means of encoding genetic algorithms to solve the generalized plant allocation problem. The problem relates to allocating products across plants to minimize a total cost function. The proposed encoding method can reduce the search space of solutions more efficiently than the penalty encoding method does. The new encoding method thus exhibits higher performance. It need involve only a few more generations to yield sufficiently good solutions when the number of plants is increased. The penalty encoding method, however, requires many more generations to yield the same solutions. Additionally, a new simultaneous crossover and mutation operation is proposed to enable the new method of encoding chromosomes to run correctly following standard genetic algorithm procedures. In addition to the mathematical certification, the performance of this approach is evaluated using some test problems of various sizes. Solutions obtained by this approach are always efficient.

Keywords: genetic algorithms, efficient encoding scheme, plant allocation, combinatorial optimization problems, penalty encoding

1. INTRODUCTION

The generalized resource allocation problem is an NP-hard combinatorial optimization problem which is concerned with allocating limited resources among activities so that the return from the activities is maximized [22-24, 28]. Some real-world examples show the wide applicability of resource allocation problems. These include allocating advertising budgets across sales territories [22, 27], distributing search efforts [6, 26], selecting an optimal portfolio [17, 39], optimally allocating samples in stratified sampling [33, 40], production planning [4, 44], the resource distribution problem [18], and others. Production planning concerns some issues, one of which is the plant allocation problem. The plant allocation problem is to allocate products among several plants such that the total cost is minimized, subject to the plant capacity constraints. The plant allocation problem has been investigated of length by mathematical programming [16, 29, 30] and dynamic programming [3, 15, 35], but has rarely been solved using genetic algo-

Received November 4, 2002; revised February 14, 2003; accepted May 15, 2003.
Communicated by Hsu-Chun Yen.

rithms. Hence, in this paper, we proposed a new encoding mode of genetic algorithms to solve the plant allocation problem efficiently.

Genetic algorithms are probabilistic search algorithms which mimic biological evolution to produce gradually better offspring solutions. Each solution to a given problem is encoded by a string of bits that represents an individual in a population. Genetic algorithms assign a fitness value to every individual according to the quality of the solution it represents. A fitter individual in the population is better able to survive into the next generation. Every generation must pass through three main genetic operations, which are selection, crossover and mutation. This evolution cycle is repeated until a satisfactory solution is obtained. Genetic algorithms have been applied successfully to combinatorial optimization problems such as the Set Partitioning Problem [12], Traveling Salesman Problem [7, 25], Timetable Problem [5], Quadratic Assignment Problem [1, 34, 38, 41], Three-Matching Problem [31], Job Shop Scheduling Problem [8, 9, 14, 42, 43], Airline Crew Scheduling Problem [36], the Mapping Problem [10, 11], and others.

The penalty encoding method is perhaps the most popular approach used in genetic algorithms for constrained optimization problems, because of its simplicity and ease of implementation [13, 19, 32]. However, using this conventional encoding method to solve plant allocation problems may waste much time and lead to inefficient performance. This study elucidates a novel means of encoding genetic algorithms to solve the generalized plant allocation problem. This method can greatly reduce the search space and thus outperform the penalty encoding method. A new crossover and mutation operation is proposed to enable the new method of encoding chromosomes to run correctly with the standard genetic algorithm procedures. The performance of this approach is mathematically certified and evaluated for some test problems of various sizes. Solutions obtained by this approach are always efficient.

2. GENERALIZED PLANT ALLOCATION PROBLEMS

The generalized plant allocation problem is concerned with allocating products among plants to minimize the cost of such plants. This paper considers the generalized plant allocation problem as follows. Q units of a product are allocated across T plants so as to minimize the total cost from each allocation of x_t units of a product to the t -th plant. Therefore, the generalized plant allocation problem can be modeled as

$$\begin{aligned} \min Z(x_1, x_2, \dots, x_T) &= \sum_{t=1}^T f_t(x_t) \\ \text{subject to } \sum_{t=1}^T x_t &= Q, 0 \leq x_t \leq C_t \end{aligned}$$

where $f_t(x_t)$ is the nondecreasing cost function for allocating x_t units of a product to the t -th plant, and where C_t is the capacity of plant t with $C_t \leq Q$. Q is the total amount of the product that needs to be allocated. If the product is divisible, x_t is a continuous variable that can take any nonnegative real value. If the product is discrete, such as persons,

processors, trucks, etc., x_i is discrete and can take nonnegative integer values. This paper focuses only on the case in which x_i is a discrete variable [24].

Cook's theorem [37] states that a problem A is NP-hard if there exists another NP-hard problem B such that B can be polynomially reducible to A. A well-known NP-hard problem is the knapsack problem, which can be polynomially reducible to the plant allocation problem. Thus, the generalized plant allocation problem can be proven to be NP-hard. This fact strongly implies that no algorithm for the generalized plant allocation problem can be solved within polynomial time. Genetic algorithms have a very simple architecture. Their evolutionary process can powerfully search for solutions, using the operations of evolution. Such a search scheme is very likely to find a globally optimal solution with a high probability [20]. These advantages make genetic algorithms suitable for solving the generalized plant allocation problem.

3. GENETIC ALGORITHMS

Holland (1975) proposed the theory of genetic algorithms, in which a genetic algorithm is a search algorithm based on genetic evolution [20]. The theory relies on the contest for survival of natural organisms. Genetic algorithms express each organism's chromosomes by strings. The fitness value of a chromosome decides whether a chromosome survives. Chromosomes can crossover and mutate randomly to produce the subsequent generation. This evolution continues until satisfactory chromosomes are generated. Holland's (1975) research described the basic algorithm, called the simple genetic algorithm (SGA), as follows.

```

SGA()
{
  Randomly set the initial population
  Calculate the fitness of the chromosomes
  While (objective not achieved)
  {
    Selection
    Crossover and mutation
    Calculate the fitness of the chromosomes
  }
}

```

The genetic algorithm especially suits situations with a large, nonlinear, complex and noisy searching space. Deterministic optimization methods and greedy heuristic techniques do not effectively solve problems with such features. Genetic algorithms support a very simple architecture and the corresponding evolutionary process facilitates powerful searches. The algorithms can explore various regions by using multiple chromosomes of the population and iteration characteristics. Random searching, called implicated parallel processing, is very likely to find a globally optimum solution. These advantages have promoted the use of genetic algorithms.

4. THE GENETIC ALGORITHM FOR GENERALIZED PLANT ALLOCATION PROBLEMS

This section describes a new method for encoding the generalized plant allocation problem. Some mathematical certifications are presented to explain how this new encoding method can really reduce the search space to a range narrower than that for the general penalty encoding method. A new corresponding crossover and mutation operation is also defined on the new encoding method.

4.1 Penalty Encoding for Generalized Plant Allocation Problems

In the simple genetic algorithm, solutions can be encoded by strings of bits. Consider T plants, A_1, A_2, \dots, A_T . Q is the total quantity of a product. Then, a chromosome can be encoded as illustrated in Fig. 1.

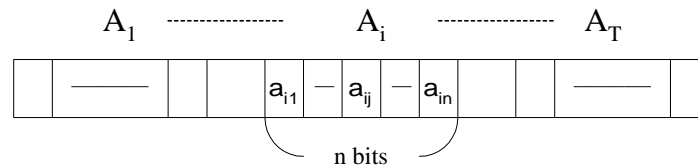


Fig. 1. Chromosome encoding method.

The allocated units of a product are encoded for each plant by a segment with n bits, where n is $\lceil \log_2 Q \rceil$. The value of a_{ij} is 1 or 0. The chromosome's length is nT , so that the searching space is 2^{nT} . Each plant, i , can be allotted $x_i = \sum_{j=1}^n a_{ij} 2^{j-1}$ products and the total units allocated to all plants must equal Q . Therefore, $x_1 + x_2 + \dots + x_T = Q$. For each segment of chromosome, x_i must not exceed the minimum value of the total available units of a product Q and each plant's capacity. For generalized plant allocation problems, three main constraints affect the fitness function. These are that the total quantity of products allocated by every plant must equal Q , and that no plant may be allotted more than Q units and its capacity. These constraints can be applied to evaluate the fitness of chromosomes.

4.2 New Encoding for Generalized Plant Allocation Problems

A new encoding method for generalized plant allocation problems is offered to enhance the performance of the evolutionary algorithm and yield superior offspring solutions. Consider T plants, A_1, A_2, \dots, A_T . Q is the total quantity of products or the percentage of products' quantity. $T-1$ bits with value 1 and Q bits with value 0 can be used by the chromosome as shown in Fig. 2.

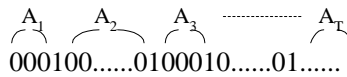


Fig. 2. New encoding method.

The bits with value 0 represent the products allocated to the plant. Bits with value 1 are grouped together as a unit and are distinguished from the bit with value 0. The i -th group of 0-bits is assigned to the i -th plant. The total number of 0-bits in the i -th group, x_i , is the quantity of products or the percentage of products' quantity allocated to plant A_i . Thus, the total number of 0-bits in all the chromosomes is Q , representing the total quantity of products. A 0-bit stands one unit or percent of products. Bit 1 is a delimiter that divides the 0-bits (products) into different groups (plants). Hence, $T - 1$ 1-bits are required to separate the 0-bits since T plants are considered. Consequently, the total length of a chromosome is $Q + T - 1$. This new encoding method yields a search space of C_{T-1}^{Q+T-1} which is just the combinatorial problem of placing $T - 1$ 1-bits into $Q + T - 1$ numbered boxes.

Evaluation

The new method directly codes the constraint $\sum_{i=1}^T x_i = Q$ into the chromosome.

That is, it satisfies the requirement that the total number of allocated units across all plants equals Q , and also satisfies $x_i \geq 0$. Now, simply counting the number of 0-bits of each plant determines the fitness value of each chromosome. That is by counting the number of 0-bits of segment t , it can find the cost of allocating x_t products to the t -th plant. If x_t exceeds the capacity of plant t , then it must reallocate the amount of product until it satisfies $x_t \leq C_t$. This method can enhance the performance of genetic algorithms in searching for good solutions. This encoding method can be shown by mathematical proof and numerical computation to be able to greatly reduce the cost of searching for good solutions in the later sections.

Selection

Each chromosome is assigned a fitness, which determines the probability of its survival in the following generation. A larger fitness value corresponds to a higher chance of survival.

Crossover

A crossover procedure must ensure that the chromosome does not violate the constraints after crossover. This new encoding method directly codes the constraint $\sum_{i=1}^T x_i = Q$ into the chromosome, and ensures that $T - 1$ bits of value 1 and Q bits of value 0 are always present in each chromosome. The method employs a two-point crossover operation. After randomly selecting two chromosomes, it can arbitrarily select two crossover points. The middle segment of these two chromosomes may not be simply exchanged as it is in the typical crossover operation, since doing so may violate the constraint that each chromosome can include only $T - 1$ bits with value 1 and Q bits with

value 0. A new scheme is required to handle this situation in the middle segments. The positions of bits with value 1 in the middle segment of both chromosomes are first recorded. Then, some bit positions are randomly selected from them for chromosome 1 and assigned value 1. The chosen bit number depends on the number of 1-bits originally in chromosome 1. Chromosome 2 assigns value 1 to the remaining bit locations that originally had a value of 1 for the two chromosomes. After crossover, two chromosomes still satisfy the requirement of having $T - 1$ 1-bits in the chromosome. But, if it happens that the chromosome violates any plant's capacity, suppose that it happened in segment A_i , then it can give the surplus amount to neighbor segment $A_{i+1 \bmod T}$ until the chromosome satisfies the capacity constraint. Fig. 3 gives a simple example.

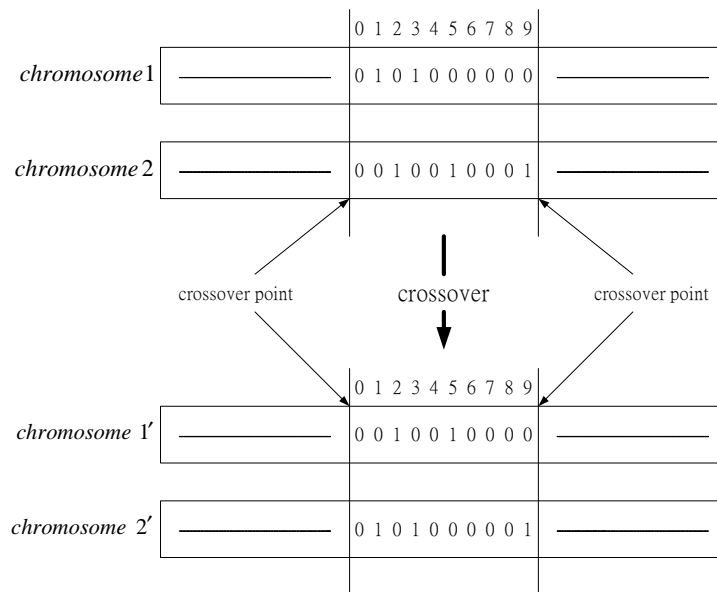


Fig. 3. An example of crossover.

Before crossover, chromosome 1 has a value of 1 at bit locations 1 and 3; chromosome 2 has a value of 1, at bit locations 2, 5 and 9, and zero is assigned to other bit locations between the two crossover points. During crossover, two out of five (1, 2, 3, 5, 9) bit locations can be chosen for chromosome 1, such as 2 and 5, and a value of 1 is assigned to these positions. The remaining bit locations 1, 3 and 9 are assigned a value of 1 to chromosome 2. The complete crossover procedure generates two new chromosomes, chromosome 1' and chromosome 2'.

Mutation

The mutation operation is performed with a small probability after crossover. Two adjacent bits, one of which takes the value 1 and the other takes the value 0, are first randomly selected and then exchanged. But, if it happens that the chromosome violate plant i 's capacity, it can exchange the difference amount with neighbor plant $i + 1$ or $i - 1$

(which depend on whether the adjacent bits are 01 or 10). The new chromosome is formed and still conforms to the constraints. Fig. 4 shows a simple example.

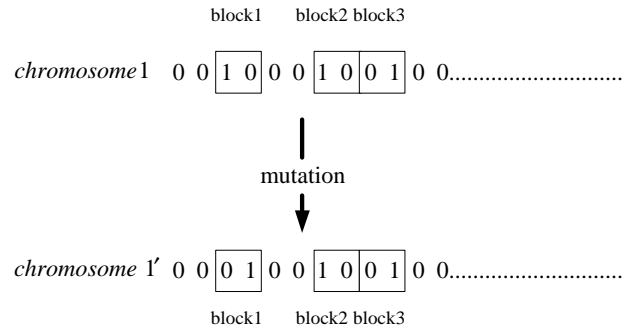


Fig. 4. Example of mutation.

4.3 Mathematical Certification

This section presents a mathematical certification to elucidate how the new encoding method can reduce the search space of genetic algorithms more than the penalty encoding method can. The computational results are presented in the following section. The mathematical certification follows.

Consider T plants. The quantity of products is Q . If the value of Q is not large, such as less than 100, then the number of 0-bits can be used to indicate the quantity of products. But, when the value of Q is large, we use the number of 0-bits to represent the percentage of the products' quantity. The new encoding method uses $T - 1$ bits with value 1 and Q bits with value 0 in the combination. The number of each group's 0 bits must be a nonnegative integer. Therefore, the size of the solution space generated by the new encoding method is C_{T-1}^{T+Q-1} .

The penalty encoding method must use $\lceil \log_2 Q \rceil$ bits to represent the allocated units of each plant. T plants correspond to a length of the chromosome of $\lceil \log_2 Q \rceil T$. The size of the solution space generated by the penalty encoding method is $2^{\lceil \log_2 Q \rceil T}$.

Proposition 1 $C_{T-1}^{T+Q-1} < 2^{\lceil \log_2 Q \rceil T}$ for $T \geq 1, Q > 2$ and $T, Q \in \mathbb{Z}^+$

Proof:

Basis Step. The Basis step is in this case obtained by setting $T = 1$.

\therefore the left side of the inequality is $C_{1-1}^{1+Q-1} = C_0^Q = 1$
 and the right side of the inequality is $2^{\lceil \log_2 Q \rceil 1} = 2^{\lceil \log_2 Q \rceil} \geq Q$,

So, $C_0^Q < 2^{\lceil \log_2 Q \rceil}$ is true.

Inductive Step. Assume that the statement is true for $T = n$ for $n > 1$.

That is, $C_{n-1}^{n+Q-1} < 2^{\lceil \log_2 Q \rceil n}$.

Now, with $T = n + 1$, the left side of the inequality is

$$C_{(n+1)-1}^{(n+1)+Q-1} = C_n^{n+Q} = \frac{(n+Q)!}{Q!n!} = \frac{(n+Q)(n+Q-1)!}{nQ!(n-1)!} = \frac{(n+Q)}{n} C_{n-1}^{n+Q-1}$$

$\therefore \frac{n+Q}{n}$ decreases when n increases.

$$\therefore \text{Max}\left(\frac{n+Q}{n}\right) = \frac{2+Q}{2} \text{ when } n = 2$$

Also, $\therefore n \geq 2$,

$$\therefore \frac{n+Q}{n} \leq 1 + \frac{Q}{2} < Q, \text{ as } Q > 2 \text{ and } Q \in \mathbb{Z}^+$$

Then, the left size of the inequality is

$$\begin{aligned} C_{(n+1)-1}^{(n+1)+Q-1} &= \frac{(n+Q)}{n} C_{n-1}^{n+Q-1} \\ &< Q 2^{\lceil \log_2 Q \rceil n} = 2^{\log_2 Q} 2^{\lceil \log_2 Q \rceil n} \\ &\leq 2^{\lceil \log_2 Q \rceil} 2^{\lceil \log_2 Q \rceil n} = 2^{\lceil \log_2 Q \rceil (n+1)} \end{aligned}$$

By the principle of mathematical induction, $C_{T-1}^{T+Q-1} < 2^{\lceil \log_2 Q \rceil T}$ is always true for $T \geq 1$, $Q > 2$ and $T, Q \in \mathbb{Z}^+$.

Proposition 1 shows that the search space generated by the new encoding method is smaller than that generated by the penalty encoding method. As the number of plants increases, the new encoding method only gradually expands the search space. However, the penalty encoding method may expand the search space quickly. A comparison of the trend of the two functions in Fig. 5 justifies the above claims under the condition of $Q = 100$.

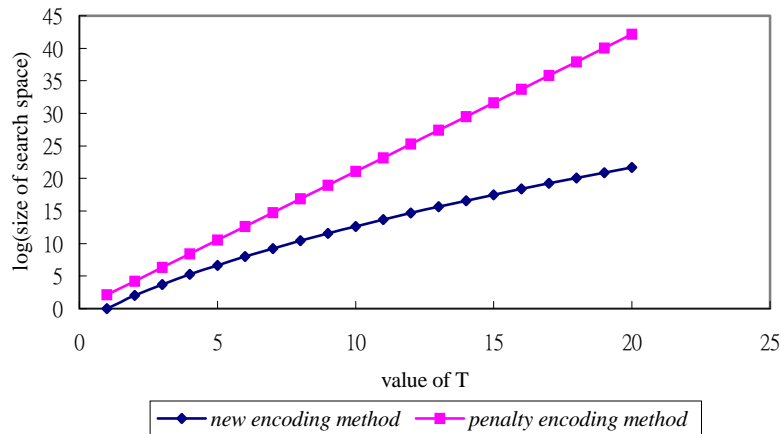


Fig. 5. Graph of two search spaces (under the condition of $Q = 100$.)

Fig. 5 clearly shows that the new encoding method can efficiently reduce the solution space, whereas the penalty encoding method cannot. On the other hand, as the number of resources increases, our result is still much better than the penalty encoding method. Fig. 6 displays the outcome under the condition of $T = 8$, $Q = 3$ to 100. The plots for other values of T are similar, but the vertical distance between the new encoding's line and the penalty encoding's line are different. When T increases, the vertical distance between the two lines also increases.

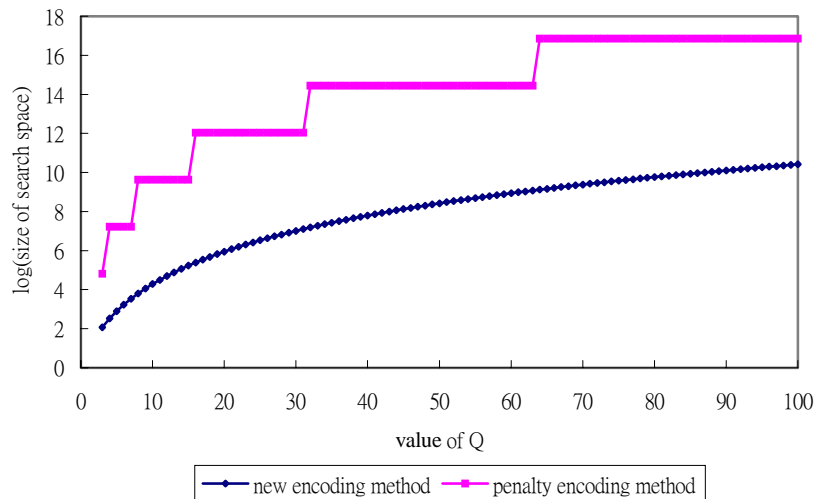


Fig. 6. Graph of two search spaces (under the condition of $T = 8$.)

4.4 Comparison of New Encoding and Penalty Encoding

In this section, we compare the variation in the size of search space between new encoding and penalty encoding. Fig. 7 exhibits the size difference of search space between these two encoding methods under the conditions of $Q = 3$ to 100 and $T = 2, 4, 6, 20$. When the difference of Q and T is small, as in the left side of this figure, we find that the variation of search space is about 10^{10} . In the right side of this figure that represents the difference between Q and T is relative larger, the divergence of search space is about 10^{20} . Hence, the new method can reduce the search space greatly in most conditions. So that Q must be very larger than T is not necessary for our new method to get superior performance than the penalty method.

We also compare the variation of the chromosome's length between new encoding and penalty encoding. The results are given in Table 1. The value of Q is in the first column, the value of T is in the top row, and the remaining values are the differences in chromosome lengths for the two encoding methods. If the difference is negative, then it indicates that the chromosome's length for the new encoding method is longer than for the penalty encoding method. The shaded part indicates that the new encoding method has longer chromosome length than the penalty encoding method, having covered an area about 8% of all conditions ($Q = 3$ to 100, $T = 1$ to 100). The new method is better for

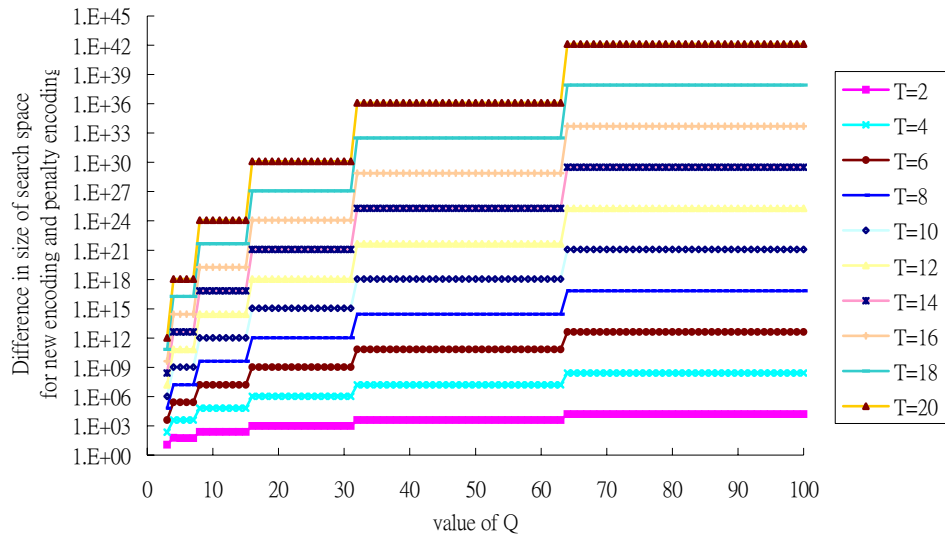


Fig. 7. Comparison graph for the variation of search space.

Table 1. Comparison of the variation of chromosome length.

Q \ T	1	5	10	15	20	40	60	80	100
3	-1	3	8	13	18	38	58	78	98
20	-15	1	21	41	61	141	221	301	381
40	-34	-14	11	36	61	161	261	361	461
60	-54	-34	-9	16	41	141	241	341	441
80	-73	-49	-19	11	41	161	281	401	521
100	-93	-69	-39	-9	21	141	261	381	501

92% of all the conditions. As the conditions of Q is 100 and T is between 1 to 100, there are just 16 conditions which the new encoding's chromosome length is longer than the penalty encoding's. Although there are some times that our chromosome length is a little bit longer, the performance of the new encoding method is much better than the penalty encoding method, as our method can reduce the search space significantly.

5. COMPUTATIONAL RESULTS

The algorithm was tested on five problems using eight to twelve plants to demonstrate that the new encoding method outperforms the penalty encoding method for generalized plant allocation problems. These test problems are expanded from examples in

earlier research, such as Lai and Li (1999) and Hussein and Abo-Sinna (1995), and characterized by i ($i = 8$ to 12) plants and with Q is 100, and in which one 0-bit represents 1%. Table 2 presents a simple example illustrating the test problem, where P_i represents plant i , the first column represents the percentage x_j of products, and the corresponding value C_{ij} between P_i and the resource number x_j denotes the cost for plant i produce x_j percent of products, which increases discretely and nonlinearly. The main objective is to minimize the sum of the costs from each allocation of x_j percentages of products to the i -th plant.

Table 2. Configuration of the test problem.

Percentage of Products	P ₁	P ₂P _i
0	xx	xxxx.....
1	xx	xxxx.....
2	xx	xxxx.....
:			:
:			:
:			:
:			:
x_j	xx	xxC _{ij}
:			:
:			:
:			:
100	xx	xxxx.....

The algorithm was coded in C and run on an IBM compatible PC. The computational procedure performed ten trials of the test program for each of the five problems. The average generation number and time cost of ten trials was listed for each problem. The population size was 100, the crossover rate was 1 and the mutation rate was set to 0.1 per chromosome. The programs employ a two-point crossover, roulette-wheel selection and terminated when the same fitness value was obtained over 50000 generations. The performance of the programs was judged by the number of evolutionary generations required to produce a sufficiently good solution. And the computation time was also supported to do the comparison. Better programs required fewer generations and less computation time.

Table 3 compares the new encoding method, the penalty encoding method and integer programming for the test problems. The left half of Table 3 (a) shows the computational results for the new encoding method and the right half presents for the penalty encoding method. #8 means that the number of plants is eight, and so on. The "Generation" column specifies how many generations of the algorithms are needed to obtain a relative minimum cost for the test problems. The last three rows show the goal value of the problem, the generation and the time cost in which the goal is attained. Table 3 (b) displays the results from using integer programming. According to the schema theorem and the building block hypothesis, smooth, unimodal problems, noisy multimodal problems and

Table 3. Computational results for test problems.

(a) Results of evolutionary methods

Cost Generation	The new encoding method					The penalty encoding method				
	#8	#9	#10	#11	#12	#8	#9	#10	#11	#12
1700	1058	1118	1192	1315	1472	1489	5044	6153	6685	7866
2000		1102	1176	1284	1437	1488	1538	6040	6500	7781
2500			1138	1258	1383	1444	1514	6038	6477	7669
3500				1198	1324	1325	1218	5799	6313	7344
4000					1262	1290	1208	5758	6280	7219
10000						1058	1102	4241	4255	5079
50000								1138	1198	1439
65000										1262
Goal	1058	1102	1138	1198	1262	1058	1102	1138	1198	1262
Cov.Gen*	1700	2000	2400	3100	3900	6000	10000	45000	50000	65000
Time Cost (Unit:sec)	14.5	19.6	26.6	38.4	51.5	31.2	75.0	391.5	490.0	715.0

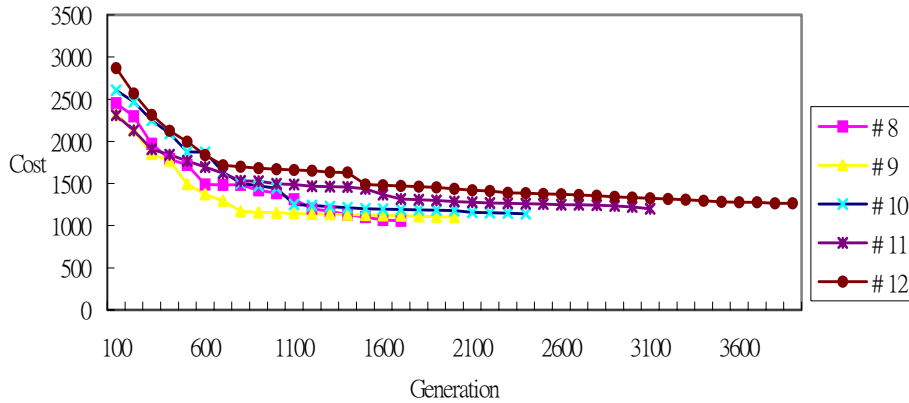
* Cov.Gen is the simplified form of "Convergent Generation"

(b) Results of traditional method

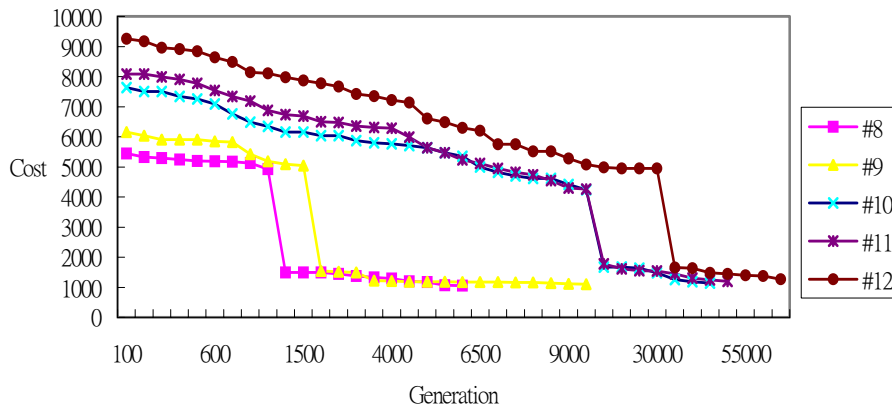
Integer programming				
#8	#9	#10	#11	#12
Integer programming just can get feasible solutions. The corresponding feasible solution is listed in the next row.				
1492	1638	1669	1808	1892

combinatorial optimization problems have all been successfully tackled using virtually the reproduction-crossover-mutation genetic algorithm [20]. The goal solutions can be thought of as sufficiently good solutions.

Table 3 shows that the new method requires fewer generations and computation time to produce goal solutions. Table 3 enables the trend line for the number of generations required by the genetic algorithm to be drawn, versus the number of plants. The gradient of Fig. 8 reveals that when the number of plants is increases, the number of necessary generations for the new encoding method grows much slowly than for the penalty encoding method. It corresponds to the search space expanding only slightly for the



(a) New encoding method.



(b) Penalty encoding method.

Fig. 8. Trend line from Table 3.

new encoding method when the number of plants increases, while the penalty encoding method may greatly expand the search space. The LINGO package was used to solve these test problems by integer programming. For these cases, LINGO can get feasible solutions only (Table 3 (b)) because the test problems are all highly complex. The computational results of this section demonstrate that the new encoding method improves the performance of genetic algorithms by reducing their search space.

6. CONCLUSIONS

The generalized plant allocation problem is an NP-hard, combinatorial optimization problem suitable for solving by genetic algorithms. The general penalty encoding method may perform inefficiently when used to solve plant allocation problems. This study pro-

posed a new encoding method that can not only greatly reduce the search space but also improve the performance of genetic algorithms. A new corresponding crossover and mutation operation enables the new method of encoding chromosomes to run correctly following standard genetic algorithm procedures.

A mathematical justification is provided to show that the new encoding method really can reduce the search space of genetic algorithms and improves their performance. Comparing the experiment results of our new encoding method and the penalty encoding method have shown that reducing the search space can certainly improve the performance of genetic algorithms. Hence, solutions obtained by our new approach are always efficient. Future research will apply this efficient encoding method to other real world problems.

REFERENCES

1. R. K. Ahuja, J. B. Orlin, and A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem," *Computers and Operations Research*, Vol. 27, 2000, pp. 917-934.
2. H. Al-Tabtabai and A. P. Alex, "An evolutionary approach to the capital budgeting of construction projects," *Cost Engineering*, Vol. 40, 1998, pp. 28-34.
3. R. E. Bellman and S. A. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, 1962.
4. G. H. Bitran and A. C. Hax, "Dissagregation and resource allocation using convex knapsack problems with bounded variables," *Management Science*, Vol. 27, 1981, pp. 431-441.
5. E. K. Burk and J. P. Newall, "A multistage evolutionary algorithm for the timetable problem," *IEEE Transactions on Evolutionary Computation*, Vol. 3, 1999, pp. 64-74.
6. A. Charnes and W. W. Cooper, "The theory of search: optimal distribution of effort," *Management Science*, Vol. 5, 1958, pp. 44-49.
7. S. Chatterjee, C. Carrera, and L. A. Lynch, "Genetic algorithms and traveling salesman problems," *European Journal of Operational Research*, Vol. 93, 1996, pp. 490-510.
8. R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms – I. representation," *Computers and Industrial Engineering*, Vol. 30, 1996, pp. 983-997.
9. R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies," *Computers and Industrial Engineering*, Vol. 36, 1999, pp. 343-364.
10. T. Chockalingam and S. Arunkumar, "Genetic algorithm based heuristics for the mapping problem," *Computers and Operations Research*, Vol. 22, 1995, pp. 55-64.
11. T. Chockalingam and S. Arunkumar, "Genetic algorithm based heuristics for the mapping problem," *Location Science*, Vol. 4, 1996, pp. 282-283.
12. P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalized assignment problem," *Computers Operations Research*, Vol. 24, 1997, pp. 17-23.
13. K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer*

- Methods in Applied Mechanics and Engineering*, Vol. 186, 2000, pp. 311-338.
14. C. Della, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Computers and Operations Research*, Vol. 22, 1995, pp. 15-24.
 15. S. A. Dreyfus and M. L. Averill, *The Art and Theory of Dynamic Programming*, Academic Press, New York, 1997.
 16. J. M. Einbu, "Extension of Luss-Gupta resource allocation algorithm by means of first order approximation," *Techniques*, Vol. 29, 1981, pp. 621-626.
 17. E. Elton, M. Gruber, and M. Padberg, "Simple criteria for optimal portfolio selection," *Journal of Finance*, Vol. 31, 1976, pp. 1341-1357.
 18. A. Federgruen and P. Zipkin, "Solution techniques for some allocation problems," *Mathematical Programming*, Vol. 25, 1983, pp. 13-24.
 19. M. Gen and R. Cheng, "A survey of penalty techniques in genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 804-809.
 20. D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley, 1989.
 21. J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.
 22. M. L. Hussein and M. A. Abo-Sinna, "A fuzzy dynamic approach to the multicriterion resource allocation problem," *Fuzzy Sets and Systems*, Vol. 69, 1995, pp. 115-124.
 23. T. Ibaraki, "Solving mathematical programming problems with fractional objective functions," in S. Schaible and W. T. Ziemba, eds., *Generalized Concavity in Optimization and Economics*, Academic Press, New York, 1981, pp. 441-472.
 24. T. Ibaraki and N. Katoh, *Resource Allocation Problems – Algorithmic Approaches*, The MIT Press, 1988.
 25. K. Katayama, H. Sakamoto, and H. Narihisa, "The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem," *Mathematical and Computer Modelling*, Vol. 31, 2000, pp. 197-203.
 26. B. O. Koopman, "The theory of search: part III, the optimum distribution of searching effort," *Operations Research*, Vol. 5, 1957, pp. 613-626.
 27. P. Kotler, *Marketing Decision Making: A Model Building Approach*, Holt, Rinehart and Winston, New York, 1976.
 28. K. K. Lai and L. Li, "A dynamic approach to multiple-objective resource allocation problem," *European Journal of Operational Research*, Vol. 117, 1999, pp. 293-309.
 29. S. M. Lee, L. J. Moore, and B. W. Taylor, *Management Science*, W. C. Brown, Dubuque, IA, 2nd ed., 1985.
 30. H. Luss and S. K. Gupta, "Allocation of effort resource among competing activities," *Operations Research*, Vol. 23, 1975, pp. 260-366.
 31. G. Magyar, M. Johnsson, and O. Nevalainen, "An adaptive hybrid genetic algorithm for the three-matching problem," *IEEE Transactions on Evolutionary Computation*, Vol. 4, 2000, pp. 135-146.
 32. Z. Michalewicz, D. Dasgupta, R. G. Le Riche, and M. Schoenauer, "Evolutionary algorithms for constrained engineering problems," *Computers and Industrial Engineering*, Vol. 30, 1996, pp. 851-870.
 33. J. Neyman, "On two different aspects of the representative method: the method of

- stratified sampling and the method of selection," *Journal of the Royal Statistical Society*, Vol. 97, 1934, pp. 558-606.
34. V. Nissen, "Solving the quadratic assignment problem with clues from native," *IEEE Transactions on Neural Networks*, Vol. 5, 1994, pp. 66-72.
 35. G. L. Numhauser, *Introduction to Dynamic Programming*, Wiley, New York, 1966.
 36. H. T. Ozdemir and C. K. Mohan, "Flight graph based genetic algorithm for crew scheduling in airlines," *Information Sciences*, Vol. 133, 2001, pp. 165-173.
 37. C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.
 38. V. Schnecke and O. Vornberger, "Hybrid genetic algorithms for constrained placement problems," *IEEE Transactions on Evolutionary Computation*, Vol. 1, 1997, pp. 266-277.
 39. W. F. Sharpe, "A simplified model for portfolio analysis," *Management Science*, Vol. 9, 1963, pp. 277-293.
 40. K. S. Srikantan, "A problem in optimum allocation," *Operations Research*, Vol. 18, 1963, pp. 265-273.
 41. D. M. Tate and A. E. Smith, "A genetic approach to the quadratic assignment problem," *Computers and Operations Research*, Vol. 22, 1995, pp. 73-83.
 42. L. Wang and D. Z. Zheng, "An effective hybrid optimization strategy for job-shop scheduling problems," *Computers and Operations Research*, Vol. 28, 2001, pp. 585-596.
 43. H. Zhou, Y. Feng, and L. Han, "The hybrid heuristic genetic algorithm for job shop scheduling," *Computers and Industrial Engineering*, Vol. 40, 2001, pp. 191-200.
 44. H. Ziegler, "Solving certain singly constrained convex optimization problems in production planning," *Operations Research Letters*, Vol. 1, 1982, pp. 246-252.

Young-Chang Hou (侯永昌) received the B.S. degree in Atmospheric Physics from National Central University, Taiwan, R.O.C. in 1972, the M.S. degree in Computer Applications from Asian Institute of Technology, Bangkok, Thailand, in 1983 and Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, R.O.C. in 1990. From 1976 to 1987, he was a senior engineer of Air Navigation and Weather Services, Civil Aeronautical Administration, R.O.C. where his work focused on the automation of weather services. Since 1987, he joined the faculty at the National Central University. Currently, he is a professor of Department of Information Management, National Central University. His research interests include information hiding and signal processing, fuzzy logic, genetic algorithm, and cryptography.

Ying-Hua Chang (張應華) received the Ph.D. degree in Management Information Systems from National Central University, Taiwan, R.O.C. His research interests include artificial intelligence, soft computing, operations research, decision analysis and information management.