Evolutionary Computation on Multicriteria Production Process Planning Problem

Gengui Zhou Mitsuo Gen

Department of Industrial and Systems Engineering
Ashikaga Institute of Technology, Ashikaga 326, Japan
Email: {zhou gen}@genlab.ashitech.ac.jp

Abstract— **Production Process Planning (PPP) problem is abundant among manufacturing systems. In general the problem can be approached by the network analysis or dynamic programming. But in the case of the multicriteria Production Process Planning (mPPP for short) problem, it is difficult for those traditional optimization techniques to cope with. In this paper, a new Evolutionary Computation (EC) approach is developed to deal with the PPP problems with both single or multiple objective criteria. The proposed EC approach adopts a new simple state permutation encoding and combines with the neighborhood search technique in mutation operation to improve the evolutionary process in finding the optimal solution of the PPP problems. The numerical analysis shows that the proposed EC is both effective and efficient to the PPP problems.**

Keywords— **Production Process Planning, Multicriteria Optimization, Evolutionary Computation.**

I. Introduction

Production Process Planning (PPP) problem is abundant among manufacturing systems. The problem, in general, provides a detailed description of manufacturing capabilities and requirements for transforming a raw stock of materials into a completed product through multi-stage process. The PPP problem lends itself, in a natural manner, to optimization techniques in order to find the best production process plan amongst numerous alternatives given a certain criterion such as minimum cost, minimum time, maximum quality or with multiple these criteria. The implicit enumeration of all these alternatives can be formulated as network flow. In the case of single objective criterion, the problem is equivalent to solving the Shortest Path problem which can be efficiently dealt with by some algorithms such as Dijkstra's and Floyed's [11]. However, in the case of multiple objective criteria (*i.e.* the mPPP problem), some techniques based on Goal Programming were developed but only illustrated with small networks [14] [15].

Recently a Genetic Algorithm (GA) approach for the PPP problem has been reported in [1] where binary strings encoding was adopted for chromosome representation. The binary strings have been proved effective to some problems[5] [6][10], but they do not always work so well. In [1] a complicated modification has to be adopted for the genetic operation as the initial population or offspring generated by crossover and mutation operations may be illegal at most cases, which greatly affects the efficiency of the evolutionary process of the GA approach.

Evolutionary Computation (EC), developed on the base of GA, adopts natural coding such as float point or permutation naturally to represent the real-world problems and evolves them towards the optimal solution combined with the genetic operations [3][7][13]. This new approach has been widely and successfully applied in variety of research areas [8] [9] [12]. In this paper, a new encoding for the PPP or mPPP problem is developed. The new encoding directly encodes the chosen state at each stage by the means of permutation, which is only with the length of n - 1 for an n-stage PPP problem, therefore to save much memory and raise the computation efficiency. The merit of the new encoding is that any common genetic operations on this encoding will not generate illegal offspring so that it need not any modification both for the initial population and for the offspring generated in the evolutionary process. Moreover, It is easy to hybrid this new encoding with some heuristic mutation operation such as the neighborhood search technique to efficiently evolve towards the optimal solution. The experiment on several test problems shows that the proposed method is both efficient and effective to the PPP problems with single objective and multiple objectives.

The mPPP problem and its mathematical model are described in Section II. Section III gives out our new encoding for the mPPP problem. The whole EC approach is discussed in Section IV. In Section V the test problems on the new encoding and EC approach are analyzed and conclusion follows in Section VI.

II. mPPP PROBLEM DESCRIPTION

The PPP system usually consists of a series of machining operations, such as turning, drilling, grinding, finishing and so on, to transform a part into its final shape or product. The whole process can be divided into several stages. At each stage, there are a set of similar manufacturing operations. The PPP problem is to find the optimal process planning among all possible alternatives given a certain criteria such as minimum cost, minimum time, maximum quality or under multiple of these criteria which are defined on the operations to be chosen. As an example, Figure 1 shows a simple PPP problem by the means of network flow.

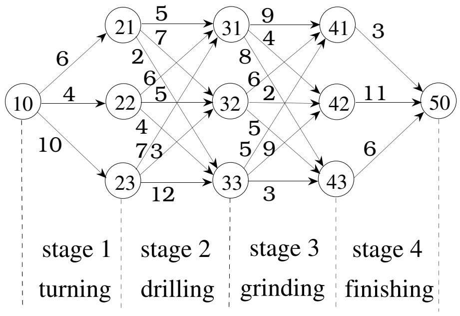

Figure 1: Flow network for a simple PPP problem

For an n-stage PPP problem, let s_k be some state at stage k, $D_k(s_k)$ be the set of possible states to be chosen at stage k, $k = 1, 2, \cdots, n$, and let x_k be the decision variable to determine which state to choose at stage k, obviously $x_k \in D_k(s_k)$, $k = 1, 2, \cdots, n$. Then the PPP problem can be formulated as follows:
PPP:

$$\min_{\substack{x_k \in D_k(s_k) \\ k=1,2,\cdots,n}} V(x_1, x_2, \cdots, x_n) = \sum_{k=1}^{n} v_k(s_k, x_k) \tag{1}$$

where $v_k(s_k, x_k)$ represents the criterion to determine x_k under state s_k at stage k, usually defined as real number such as cost, time or distance *etc.*

The problem (1) can be rewritten as an dynamic recurrence expression as Figure 1 shows that it can be approached by the method of the Shortest Path method or Dynamic Programming. However, in the case of multiple objective criteria, the problem (1) takes the following formulation:
mPPP:

$$\begin{array}{c} \min V_1(x_1, x_2, \cdots, x_n) = \sum_{k=1}^{n} \ v_k(s_k, x_k) \\ \min V_2(x_1, x_2, \cdots, x_n) = \sum_{k=1}^{n} \ v_k(s_k, x_k) \\ \cdots \\ \min V_p(x_1, x_2, \cdots, x_n) = \sum_{k=1}^{n} \ v_k(s_k, x_k) \\ \\ \text{s. t.} \quad x_k \in D_k(s_k), \quad k = 1, 2, \cdots, n \end{array} \tag{2}$$

where p is the number of the objectives. The problem (2) is denoted as the mPPP problem.

Obviously it is difficult to transform the problem (2) into its equivalent dynamic recurrence expression to be solved by the Shortest Path method or Dynamic Programming. As to small scale problem, the problem (2) can be approached by some traditional multiple criteria decision making techniques such as goal programming. However, if the problem scale increases, it becomes difficult to be dealt with even in the case of single objective criteria because of the rapid expansion of the number of the states to be considered, not mansion of the case of multiple objective criteria. So it is much necessary to develop new approach to solve this mPPP problem. In the following section we discuss our Evolutionary Computation approach.

III. State permutation encoding

When considering the network flow of problem (1) shown in Figure 1, it is intuitive to describe the process planning by indicating which node or state is chosen for a particular operation at each stage. If the node or state is chosen, then denoted as "1", if not, denoted as "0". In such way, the PPP solution can be encoded in a binary string format by concatenating all the set states of the stages as shown in Figure 2.

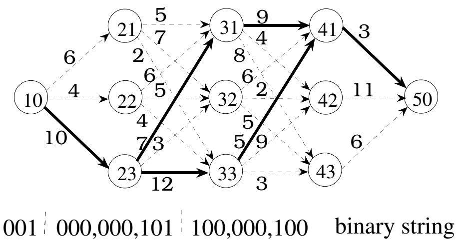

Figure 2: Binary string encoding for the PPP problem

As there is always only one state to be chosen at the last stage, it need not indicate in the encoding. Though this binary string encoding can be operated within a GA environment, it is inevitable that there will be more than two possible process plannings encoded simultaneously by one encoding or no possible process planning encoded in the initial population and the offspring generated in the evolutionary process. Therefore, there needs a more complicated modification for the GA operations involving both "Path Identifier" and "Multiple Path Corrector" [1].

Actually, as to the PPP problems shown in Figure 1, the alternative states at each stage can be expressed by a series of integers to indicate the node or state. If a state for an operation is chosen at some stage for the process planning, then its corresponding integer for that node or state can be assigned whereas the integer is within the number of possible states at that stage. Therefore, the PPP solution can be concisely encoded in a state permutation format by concatenating all the set states of the stages as shown in Figure 3.

This state permutation encoding is one-to-one mapping for the PPP problem, so it is easy to decode and evaluate. Compared with the binary string encoding, this state permutation encoding has three valuable advantages:

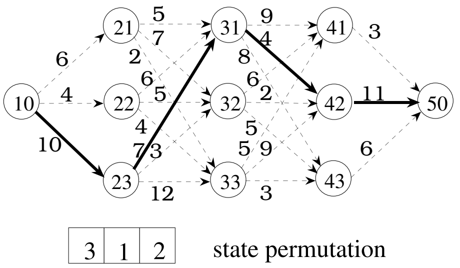

Figure 3: State permutation for the PPP problem

(1) the encoding length is only n - 1 for an n-stage PPP problem, which is to save much more computation memory while the problem scale gets larger;
(2) the encoding need not any modification in genetic operations such as crossover or mutation, which is also to raise the computation efficiency;
(3) the encoding is easy to be hybridized with some heuristic genetic operations to improve the evolutionary process.

IV. Evolutionary computation approach

A. Genetic Representation

As it has been well discussed in the previous section, the state permutation encoding is more suitable for the PPP problem. Those three advantages guarantee the state permutation encoding adaptable to the genetic operation in EC approach. As to the initial population for an n-stage PPP problem, each individual is a permutation with n-1 integers whereas the each integer is generated randomly within the number of all possible states at the corresponding stage.

B. Genetic Operation

Though any common genetic operation on the state permutation encoding will not result in illegal offspring, simply we only adopt mutation operation in the EC approach as the crossover operation has no great effect on the PPP problem, which will be demonstrated by the experiment analysis in Section V. On the other hand, for this state permutation encoding, it is easy to hybrid the neighborhood search technique in mutation operation to produce an improved offspring. Figure 4 shows an example for this mutation operation with neighborhood search technique, supposed that the mutated gene is at stage 3 and the number of possible states to be chosen is 4.

C. Evaluation

In the case of the PPP problem with single objective criterion, we can directly calculate the fitness value of each individual according to the objective function of the problem (1). However, as to the mPPP problem, we can

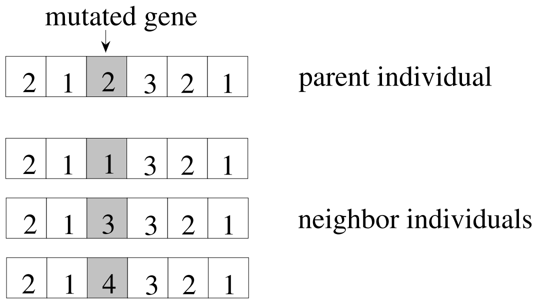

Figure 4: Mutation with neighborhood search

only calculate each objective value of the problem (2) but can not simply evaluate its fitness value as these multiple objectives usually conflict with each other in practice. In other words, we can not obtain the absolute optimal solution, but we can only get the Pareto optimal solutions. Here, we adopt some multiple criteria decision making techniques [4] to evaluate the fitness for the mPPP problem.

Definition 1: Given a set of feasible solution X for the problem (2), solution $\boldsymbol{x}' \in X$ is denoted as the **Pareto optimal solution** (or **nondominated solution**) for the problem (2) if and only if there is no any other solution $\boldsymbol{x} \in X$, satisfying the following conditions:

$$\begin{array}{ll} V_q(\boldsymbol{x}) < V_q(\boldsymbol{x}') & \text{for some } \; q \in \{1, 2, \cdots, p\} \\ V_k(\boldsymbol{x}) \leq V_k(\boldsymbol{x}') & \text{for all } \; k \neq q \end{array}$$

Definition 2: As to the problem (2), in optimizing each objective: $\min_{\boldsymbol{x} \in X} V_k(\boldsymbol{x}), k = 1, 2, \cdots, p$, there are respectively p optimal solution $\boldsymbol{x}^j, j = 1, 2, \cdots, p$, and corresponding objective value:

$$V_k^j = V_k(\boldsymbol{x}^j), \quad k = 1, 2, \cdots, p, \quad j = 1, 2, \cdots, p$$

In objective space, through these p points $V_k = (V_k^1, V_k^2, \cdots, V_k^p) \;\; (k = 1, 2, \cdots, p)$, there is a hyperplane which satisfies:

$$\sum_{k=1}^{p} \beta_k \; V_k = \alpha$$

where α and $\beta_k \;\; (k = 1, 2, \cdots, p)$ are the solution of the following equations:

$$\begin{array}{lcl} \sum_{k=1}^{p} \beta_k & = & 1 \\ \sum_{k=1}^{p} V_k^j \; \beta_k - \alpha & = & 0, \;\; j = 1, 2, \cdots, p \end{array} \tag{3}$$

This hyperplane is denoted as the **adaptive objective evaluation hyperplane** for the problem (2).

The above equations (3) have $(p+1)$ variables and $(p+1)$ linear equations. It is not difficult to verify that there is an unique solution if there does not exist absolute optimal solution for the problem (2).

Using this adaptive objective evaluation hyperplane, we can evaluate the fitness value of the problem (2) in the sense of multicriteria and enforce its Pareto optimal solutions to get close to the ideal point as much as possible. The evaluation process can be operated as follows:

procedure: evaluation for the mPPP

step 1: Decode all individuals and calculate their objective values in each objective.

step 2: Determine $\beta_k (k = 1, 2, \cdots, p)$ according to the Definition 2.

step 3: Determine the fitness value $\mathrm{eval}(\boldsymbol{x})$ of all individuals according to the following formula:

$$\mathrm{eval}(\boldsymbol{x}) = \sum_{k=1}^{p} \beta_k \; V_k \tag{4}$$

As to the mPPP problem with two objectives, the mechanism of evaluation in evolutionary process can be illustrated by Figure 5. The adaptive objective evaluation hyperplane divides the search space into two parts. One contains Pareto optimal solutions and the ideal point which is unreachable, and the other only contains dominated solutions. With the evolutionary process and under the pressure of selection operation, the adaptive objective evaluation hyperplane is updated again and again, and moved toward the ideal point to enforce all Pareto solutions get close to the ideal point as much as possible. Finally all Pareto solutions get to its Pareto frontier.

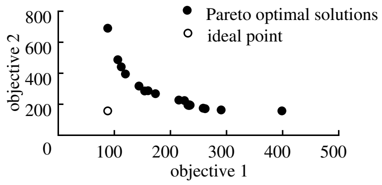

Figure 5: Illustration of the evaluation

D. Selection

Here we adopt the $(\mu + \lambda)$ - selection strategy [2]. But in order to avoid the premature convergence of the evolutionary process, our selection strategy only selects μ different best individuals from μ parents and λ offspring. If there are no μ different individuals available, the vacant pool of population is filled with renewal individuals generated randomly.

V. Numerical analysis

Firstly we make some numerical experiments on the PPP problem. The numerical example with single objective, presented by B. Awadh *et al.*, has been in detail studied in[1]. The problem comprises 7 stages, 24 nodes, 80 arcs and the total number of 1440 possible production process plannings. It is given out as shown in Figure 6

For this simpler example, by using the binary string and setting the population size as *pop_size*: 200, in the best case the optimal solution can be found at the 5th generation and on average the performance of the evolutionary process has a leveling off starting only after the 21th generation in 20 trials [1]. However, by using the state permutation encoding and the proposed EC approach, the optimal solution can be averagely obtained at the 4th generation for a smaller population size as *pop_size*: 5 and the 3rd generation for a larger population size as *pop_size*: 50. Since the state permutation encoding is much simpler than the binary string encoding for this problem, it really takes no much more effort in the evolutionary process to evolve to the optimal solution. Figure 7 clearly illustrates the evolutionary process of the state permutation encoding under the action of the proposed EC approach, which is plotted according to the average results in 20 trials.

As to the analysis for the setting of the genetic operation parameters, we figure out a larger scale PPP problem which consists of 15 stages, 89 nodes, 562 arcs on which the weights are integers generated randomly and uniformly distributed over $[1, 50]$. According to the experiment results, the crossover operation really has no much effect on the evolutionary process for the EC approach on this problem. If only crossover operation is adopted, no optimal solution can be obtained in 20 trials. As to the mutation operation, it provides with great probability to obtain the optimal solution whether the crossover operation is simultaneously adopted or not. Figure 8 clearly illustrates the sensitivity of crossover and mutation operations on the proposed EC approach and indicates that the mutation rate within 0.6 to 0.9 has the best mechanism for the evolutionary process.

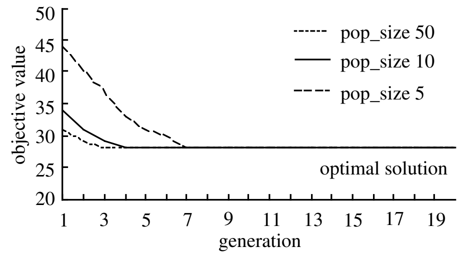

Figure 8: Illustration of the sensitivity on crossover and mutation

The experiment results show that in the EC approach on this PPP problem, the crossover operation (uniform crossover or cut-point crossover) does not bring about enough new genes for the evolutionary process to evolve towards the optimal solution. Only the mutation operation generates series of new genes necessary in the evolutionary process and guarantee to evolve to the optimal solution while being combined with the neighborhood search technique. Moreover, too low mutation rate is unable to make enough mutation on some genes for evolution; while too high mutation rate results in too much random perturbation on individuals so that the offspring have more chance to lose their resemblance to the parents and the algorithm loses the ability to learn from the history of the search. When the mutation rate is equal to $0.7 \sim 0.8$, the EC approach has the great probability to evolve to the

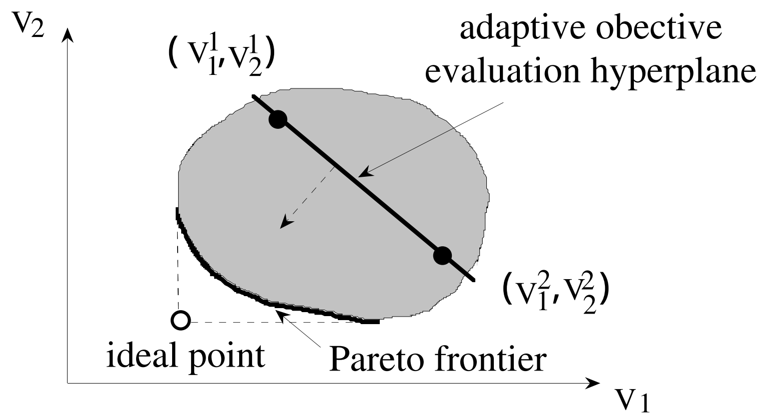

Figure 6: Illustration of the numerical example with 7 stages and 24 nodes

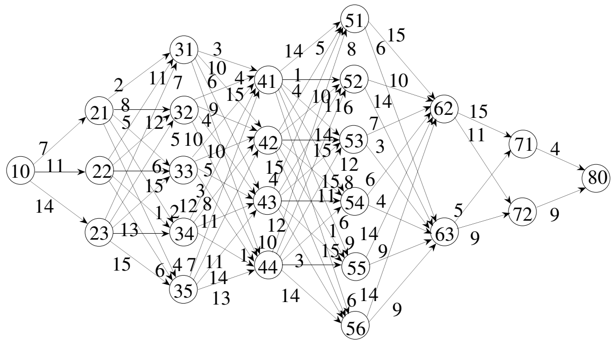

Figure 7: Illustration of the evolutionary process by the EC approach

optimal solution for this problem.

In order to further demonstrate the effectiveness and efficiency of the proposed EC approach on the PPP problem, we generate 10 different PPP problems which have the same scale as those in [1]. All results are given out in Table 1. In order to compare with two different EC approaches on a same problem, there are several factors which have to be taken into consideration. Besides the different setting of parameters such as crossover rate, mutation rate, population size, total generation and total runs, different soft and computational means are also very important for the computational comparison. However, compared with the experiment results in [1], Table 1 still clearly shows that our EC approach is both effective and efficient to the PPP problem.

Finally, let us discuss the mPPP problem. Based on the numerical analysis of the PPP problem with single objective, an larger scale mPPP problem with 15 stages and 89 nodes are designed. Two attributes are defined on each arc whose weights are integers generated randomly and uniformly distributed over $[1, 50]$ and $[1, 100]$ respectively. The ideal point is (91, 159) and other two extreme points (Pareto optimal solutions) are (91, 686) and (402,159). Because there are different attributes defined on each arc, it is impossible to get the ideal point as the optimal solution. We can only get the Pareto optimal solution of the mPPP problem. By using the adaptive objective evaluation hyperplance as the fitness function, the results by the proposed EC approach are plotted as shown in Figure 9.

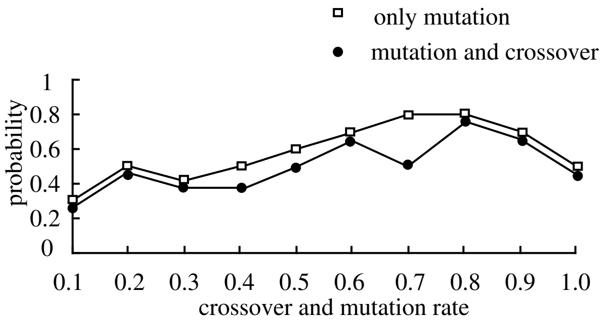

Figure 9: Illustration of the mPPP problem

Figure 9 clearly illustrates that the adaptive objective evaluation hyperplane plays an very important role in guiding the search towards the ideal point in the evolutionary process for the case of multiple objective criteria, and really enforces all Pareto optimal solutions got to the ideal point as close as possible and focused on the Pareto frontier near the ideal point. This is usually the results some decision makers hope in multicriteria decision making.

VI. Conclusion

Table 1: Comparison among different approaches to the PPP problems

No.	Number of stages	Number of nodes	CPU time OK^1 (sec.)	CPU time(sec.) BS^1 Min	Ave	CPU time(sec.) SP^2 Min	Ave	%
1	7	24	3.01		0.08	0.03	0.04	100
2	7	27	2.84		0.04	0.03	0.04	100
3	8	38	4.31		0.40	0.06	0.09	100
4	9	37	4.25	0.1	0.57	0.06	0.12	100
5	10	47	4.48	0.1	0.63	0.09	0.27	100
6	11	53	4.58	0.2	0.75	0.34	0.45	100
7	12	63	4.69	0.15	0.36	0.32	0.48	100
8	13	72	5.08	0.8	1.30	0.61	1.03	100
9	14	79	5.19	0.5	1.75	0.49	1.01	90
10	15	89	5.35	0.4	1.53	0.97	1.28	80

OK^1: Out-of-Kilter algorithm using SAS/OR program operated on SUN workstation[1];
BS^1: Binary String encoding by GAs operated on SUN workstation, pop_size: 200[1];
SP^2: State Permutation encoding by EC operated on EWS4800/360PX workstation, pop_size: 100;
Min: the minimal CPU time in all 20 runs; Ave: the average CPU time in all 20 runs;
%: the frequency to obtain the optimal solution in all 20 runs.

This paper presents a new approach to the PPP problem by using the EC technique. The proposed EC approach adopts a new state permutation encoding to make the evolutionary process even more simpler in finding the optimal solution, and combines with the neighborhood search technique to have the great probability to evolve to the optimal solution. The numerical analysis shows that the proposed EC approach is competitive to the traditional network analysis techniques in solving this kind of problem. Compared with the other genetic algorithms approach, the proposed EC approach is both effective and efficient to deal with this kind of problem with single or multiple objectives.

Acknowledgment

This research work was partially supported by the International Scientific Research Program (No. 07045032: 1995.4 – 1998.3) Grant-in-Aid for Scientific Research by the Ministry of Education, Science and Culture of the Japanese Government.

References

[1] B. Awadh, N. Sepehri and O. Hawaleshka, "A computer-aided process planning model based on genetic algorithms," *Computers & Ops. Res.*, Vol. 22, pp. 841-856, 1995.

[2] T. Bäck, "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms," *Proceedings of the First IEEE Conference on Evolutionary Computation*, D. Fogel (ed), IEEE Press, Florida, pp. 57-62, 1994.

[3] T. Bäck and H.-P. Schwefel, "Evolutionary computation: an overview," *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, D. Fogel (ed), IEEE Press, Nagoya, pp. 20-29, 1996.

[4] V. Chankong and Y.Y. Haimes, *Multiobjective Decision Making Theory and Methodology*, North-Holland, New York, 1983.

[5] R. Das and D.E. Goldberg, "Discrete-time parameter estimation with genetic algorithms," *Proc. of 19th Annual Pittsburgh Conference Modeling and Simulation*, 1988.

[6] K.A. DeJong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, Univ. of Michigan, Ann Arbor, 1975.

[7] D. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. on Neural Networks*, Vol.5 No.1, pp. 3-14, 1994.

[8] D. Fogel, *Evolution Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, 1995.

[9] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, 1997.

[10] J.J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. on Syst. Man & Cybernet*, Vol. 16, pp. 122-128, 1986.

[11] A.P. Jensen and J.W. Barnes, *Network Flow Programming*, John Wiley & Sons, New York, 1980.

[12] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*, 2nd ed., Springer Verlag, Berlin Heidelberg, 1994.

[13] Z. Michalewicz, "Evolutionary computation: practical issues," *Proceedings of 1996 IEEE International*

Conference on Evolutionary Computation, D. Fogel (ed), IEEE Press, pp. 30-39, 1996.

[14] N.G. Sancho, "A multi-objective routing problem," *Engng. Optimization*, Vol. 10, pp. 71-76, 1986.

[15] M. Sniedovich, "A multi-objective routing problem revisited," *Engng. Optimization*, Vol. 13, pp. 99-108, 1988.