

# Genetic Algorithms: Artificial Selection vs Natural Selection\*

Xiao-Bing Hu and Ezequiel Di Paolo  
Department of Informatics  
University of Sussex

Falmer, Brighton, BN1 9QH, UK  
[Xiaobing.Hu@sussex.ac.uk](mailto:Xiaobing.Hu@sussex.ac.uk), [ezequiel@sussex.ac.uk](mailto:ezequiel@sussex.ac.uk)

**Abstract** - Genetic Algorithms (GAs) are a stochastic searching and optimizing method inspired by the biological mechanism of natural selection and evolution. To improve the searching power of GAs for complicated problems, many deterministic measures, particularly, experience and/or expert knowledge-based heuristic rules, have been studied in the existing literature. This paper proposes a potentially more useful general methodology of integrating deterministic strategy with the original stochastic method of GAs. Apart from the mechanism of stochastic natural selection, this paper introduces a more deterministic method of artificial selection as a crucial step to designing the new GAs. An artificial selection gene database contributes a lot to the good performance of the new GA. The concept of artificial selection could even possibly prepare GAs for on-line/real-time implementations in dynamic complex systems. Two case studies – the travelling salesman problem and operation/route planning problem – are conducted and the results prove that the performance of the new GA is significantly improved.

*Index terms* - Genetic Algorithms, Natural Selection, Artificial Selection, Optimization.

## I. INTRODUCTION

Genetic Algorithms (GAs) have been widely used for numerical optimization, combinatorial optimization, classifier systems and many other engineering problems [1], [2]. Basically, a GA is a large-scale parallel stochastic searching and optimization method inspired by the biological mechanisms of Darwinian natural selection: given a population of chromosomes, environmental pressures result in natural selection (survival of the fittest) and hereby the fitness of the population may grow. Many components of such an evolutionary process in GAs are stochastic. Due to the stochastic nature of GAs, a common problem arises regarding the optimizing power: GAs cannot always guarantee finding the global optimum, even for a simplest problem, let alone complex systems. In order to enhance the searching efficiency and therefore improve the performance of GAs, many efforts have been made in the past decades (see [3], [4], [5], [6] and [7]) with varying results. There is a common sense that some deterministic measures, particularly, experience and/or expert knowledge based heuristic rules, should be integrated into

GAs, and a trade-off between these deterministic measures and the stochastic nature of GAs should be made. For instance, some research [3] proposes the use of on-line self-adaptive probabilities for crossover and mutation, while classification of chromosomes is another deterministic measure often taken to improve the searching efficiency of GAs [4]. Apart from such general deterministic methods to improve the performance of GAs, a more widely accepted strategy in many real implementations is to introduce problem-dependent heuristic rules [6], [7].

In this paper, we propose a new general methodology of integrating deterministic measures into the stochastic evolutionary process of GAs. Natural selection is the essential part of GAs and generates their basic searching power, but it also makes them an extremely time-consuming method, particularly when GAs are applied in complex systems. We notice that, in biological science, after people identify and understand the functions of certain genes in chromosomes, they can then artificially modify those genes to serve certain specific purposes which, otherwise under natural evolution, could take millions of years for those genes to mutate to the states which people artificially set. Motivated by this observation, here we introduce the concept of artificial selection into normal GAs, in order to improve the performance.

Let us suppose that a complex optimization problem can be divided into several relatively independent sub-problems. The solution to a sub-problem corresponds to certain genes in a chromosome. *Artificial Selection* (AS) is used to narrow down the search space of a specific sub-problem in advance. Those genes associated with the specific sub-problem are called *AS genes*. To better manage AS genes, a database needs to be set up before the new GA is implemented, and this database is called *AS gene database*. For the sake of convenience, we denote *Natural Selection* as NS hereafter. As will be proved later in this paper, the introduction of the AS gene database can significantly enhance the searching efficiency as well as improve solution quality when compared with NS-based GAs.

## II. ARTIFICIAL SELECTION BASED GENETIC ALGORITHM

As is well known, the mechanism of NS generates the basic search power of GAs. When a new chromosome is initialized, its genes are set randomly. When crossover or mutation is carried out, the choice of which chromosome or genes will be used for the evolutionary operation follows a

\* This work is supported by EPSRC Grant EP/C51632X/1 to E. di Paolo.

random law. When the value of a gene changes during the evolutionary operation, it changes randomly. After all these stochastic operations are carried out in a generation, chromosomes compete for a place in the next generation according to fitness. The stochastic nature of NS can make GAs a very time-consuming method.

In biological science, in order to alter a species to have certain desired new features, scientists study their genotypes and track down some certain genes which are involved in their representation. After they identify those genes in the chromosome, they then focus on the functions of those genes, in order to artificially modify them to serve certain specified purposes. Normally, this is also a trial-and error process. Scientists modify certain genes and then observe the representations in a generation of individuals; based on the results, they choose the best individuals, and modify them again; this process is repeated many times until the desired features are stable and fully present. This highly purpose-focused and purpose-driven selecting process by people is called *Artificial Selection* (AS). Compared with NS, which could take millions of years to evolve a species to a certain level, AS probably only needs a few years to achieve the same goal. During their study on genes, bio-scientists also set up genes databases, which enable them to more efficiently modify the genes of species.

Motivated by this observation, here we attempt to introduce the concept of AS into normal GAs, in order to improve the performance. If a complex optimization problem can be divided into several relatively independent sub-problems, then the AS-based GA has the potential to be applied. If the original complex problem can be transformed into several completely independent sub-problems, then it is easy to solve. Here we consider a more general situation: those resulted sub-problems are not completely independent, instead there are some coupling relationships between them. In other words, the optimal solution to each sub-problem does not necessarily lead to the optimal solution to the original complex problem. In order to optimize the original complex problem, the solution to each sub-problem actually depends on some certain states of other sub-problems. If, to optimize the original complex problem, the solution to a certain sub-problem can be uniquely determined when those relevant states in other sub-problems are fixed, and if the ranges where those relevant states in other sub-problems vary are known, then this sub-problem could be used for AS operation. To carried out the AS operation, an AS gene database needs to be set up where all solutions to the sub-problem are saved in association with those relevant states in other sub-problems varying in specified ranges. Therefore, when a GA is running, those genes related to this sub-problem do not vary randomly, but only within the AS gene database. Those genes which determine a solution to this sub-problem link together in a chromosome of the original complex problem, and therefore form a *gene serial* related to this sub-problem. Consequently, the AS gene database for the sub-problem records those gene serials which will lead to the optimal solution to the original complex problem under different circumstances.

For example, the original route planning problem in Fig.1 can be divided into two sub-problems, Clearly the sub-problem 2 can be easily solved by dynamical programming, and two solutions to the sub-problem 2 are saved in the AS gene database for constructing the solution for the original problem. Since the sub-problem 1 is still a NP complete problem, it is normally not suitable for AS operation unless its solution space is very small (of course in the sub-problem 1 in Fig.1 is still very simple for demonstrational purposes). When the AS-based GA is applied to the original problem given in Fig.1, those genes related to the sub-problem 1 vary randomly. When a solution to sub-problem 1 is determined by those associated genes, then one needs to choose from the only two records in the AS gene database to form a chromosome for the original problem. From the example given in Fig.1, one can feel that the AS-based GA could achieve better performance than normal GAs. The analyses on the size of solution space in the AS-based GA in the following section will support this intuition.

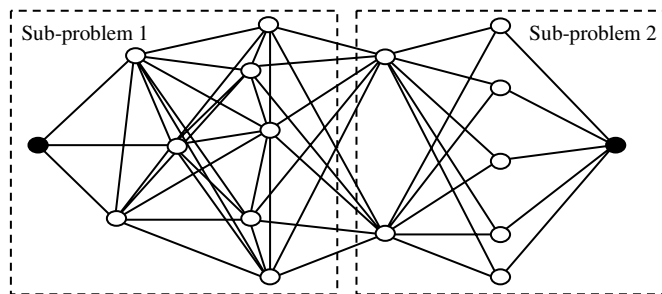


Fig. 1 Sub-problems and AS operation

Fig.2 gives the flow chart which integrates above AS-related steps into a common practice of GAs. Clearly, identification of those gene serials which are suitable for AS-related operations and setup of AS gene database are two crucial steps in the proposed AS-based GA. The flow chart in Fig.3 further explains the details of these two steps. From Fig.3, one can see that whether a sub-problem is suitable for AS-related operations depends on how complex it is and how heavily it couples with other sub-problems. If there is no simple or efficient method to find optimal solutions to the sub-problem, e.g., as in the sub-problem 1 in Fig.1 unless its solution space is small enough, then we think it is too complex. Otherwise, like the sub-problem 2 in Fig.1, we go on to classify its input environment. The input environment of a sub-problem is determined by solutions to other sub-problems. Each solution combination of other sub-problems is a potential case in the input environment, and has influence on solving the concerned sub-problem in terms of optimizing the original complex problem. Usually, we do not need to study every possible solution combination of other sub-problems, because some solution combinations have exactly the same influence on the concerned sub-problem. According to the influence, we can classify the input environment. For example, there are many optional routes which go through the network associated with the sub-problem 1 in Fig.1, but they link to the sub-

problem 2 through only two nodes, which means the input environment for the sub-problem 2 can be classified into two categories. Under each category of inputs, we can get an optimal solution to the sub-problem 2. No matter which route we choose to go through the network of sub-problem 1, the associated optimal route through the entire network is guaranteed by one of the 2 optimal solutions to the sub-problem 2. After classifying the input environment of the concerned sub-problem, we can have a feeling of how heavily the sub-problem is coupled with other sub-problems: the more categories of inputs, the more heavily coupled. Fig.4 illustrates some typical cases when classifying input environment for a sub-problem, where Case (a) and Case (b) are heavily coupled cases, while Case (c) and Case (d) are suitable for AS-related operations. One can see that Case (a) and Case (d) are two extreme cases and Case (d) implies the concerned sub-problem is completely independent of others.

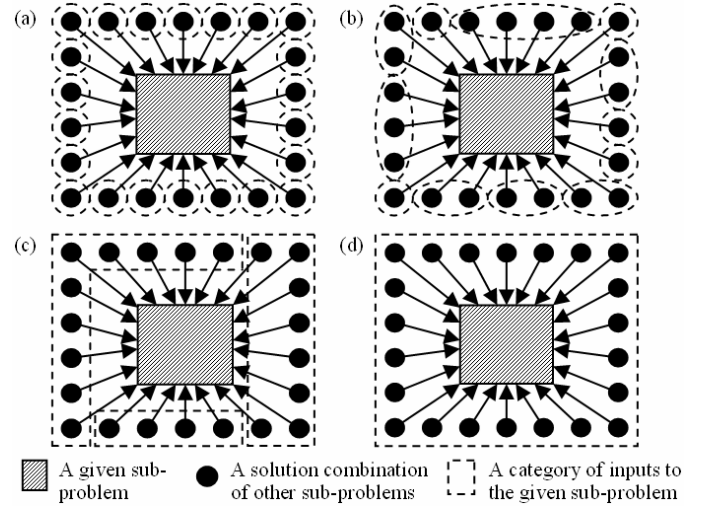


Fig. 4 Classify the input environment of a given sub-problem

### III. FURTHER ANALYSES

In this section, we will take a deeper look into the computational efficiency of AS-based GA proposed in this paper. For the sake of simplicity, we assume that a complex problem has only one sub-problem which is suitable for AS-related operations. Suppose the structure of chromosomes is as that given in Fig.5, where a chromosome is composed of  $L$  genes, i.e., the length of chromosome is  $L$ ,  $g_i$  is the  $i$ th gene in the chromosome, and has  $m_i$  discrete values to choose

$$g_i \in \{g_{i,1}, \dots, g_{i,m_i}\}, m_i \geq 1, i = 1, \dots, L. \quad (1)$$

Based on this chromosome structure, the size of the solution space for NS-based GAs,  $S_{NS}$ , can be calculated as

$$S_{NS} = \prod_{i=1}^L m_i. \quad (2)$$

For AS-based GAs, the size of solution space,  $S_{AS}$ , can be estimated by

$$S_{AS} = S_{ASDB} \prod_{i=1}^{a_1-1} m_i \prod_{j=a_n+1}^L m_j \quad (3)$$

where  $S_{ASDB}$  is the size of AS database, and normally one has

$$S_{ASDB} \ll \prod_{i=a_1}^{a_n} m_i. \quad (4)$$

For example, suppose that an AS gene serial in Fig.5 is determined by its first gene,  $g_{a_1}$ , and its last gene,  $g_{a_n}$ , which means once  $g_{a_1}$  and  $g_{a_n}$  are given, then all other genes in the AS gene serial can be uniquely determined in order to optimize the concerned problem. In this case, one can calculate the size of AS database as

$$S_{ASDB} = m_{a_1} m_{a_n}, \quad (5)$$

and then the size of solution space for AS-based GAs is

$$S_{AS} = \prod_{i=1}^{a_1} m_i \prod_{j=a_n}^L m_j, \quad (6)$$

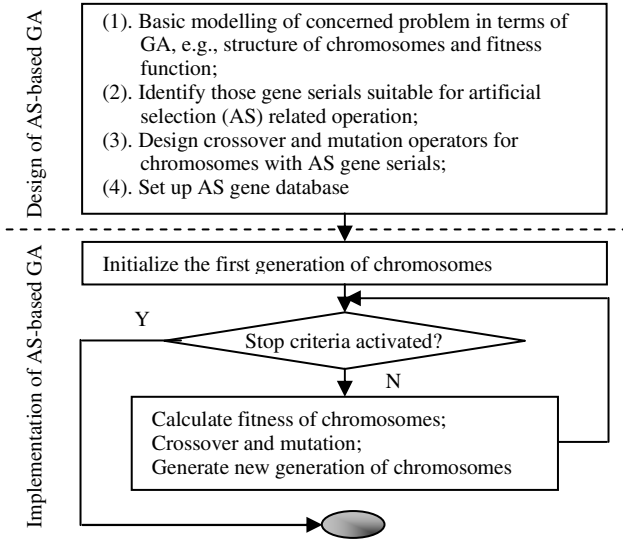


Fig. 2 Flow chart of AS-based genetic algorithm

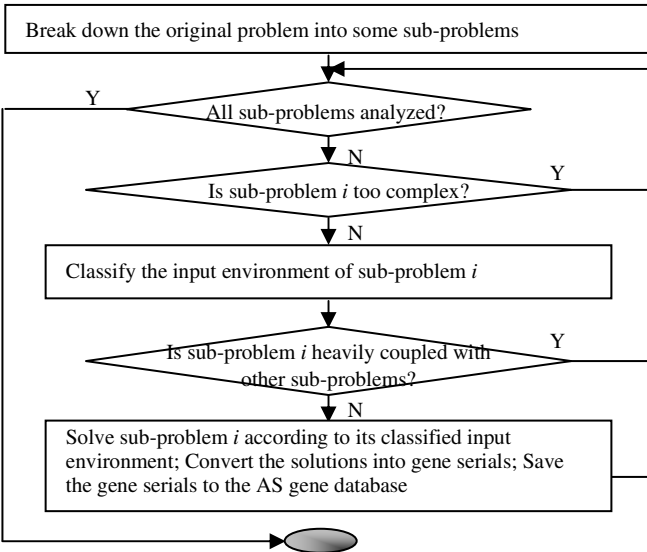


Fig. 3 How to identify AS genes and set up AS gene database

which is  $\prod_{i=a_i+1}^{a_n-1} m_i$  time smaller than  $S_{NS}$ , the size of solution space for NS-based GAs.

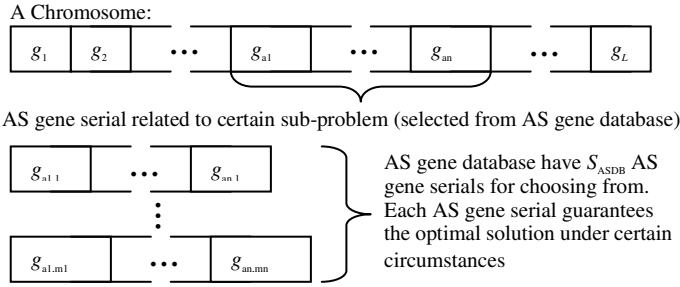


Fig. 5 Chromosome and AS gene database

Suppose that every gene in the chromosome has the same number of discrete values to choose, i.e.,  $m_i=m$ ,  $i=1, \dots, L$ . Then one has

$$S_{NS} = m^L, S_{ASDB} = m^2, S_{AS} = m^{L-n+2}, \quad (7)$$

which means  $S_{NS}$  is  $m^{n-2}$  times larger than  $S_{AS}$ .

Clearly, the advantages resulting from the usage of AS gene database suggest an improvement in terms of computational efficiency. As will be proved by simulation tests later, both solution quality and speed of GAs are significant improved by introducing the concept of AS.

As for the set up of AS database, as discussed before, AS gene serials are related to some relatively more independent and much simpler sub-problems, which can therefore be isolated and solved easily and independently in advance. For instance, these simpler sub-problems could possibly be efficiently solved by some deterministic optimizing algorithms, such as linear programming and dynamical programming; or because of their small solution spaces, their optimum solutions can even be found in a short period of time by using stochastic search algorithms or even exhaustive search. Once AS gene serials are identified offline, the calculation of the AS database can even be put into the online optimization procedure of GA.

Of course, the advantages of this new method largely depend on the possibility of dividing complex problems into useful and relatively isolated sub-problems. If the identified AS gene database leads to a relatively larger proportion of good solutions or good paths towards an optimum in the reduced search space, one could expect a much higher computational efficiency. For some complex situations this may not be the case.

#### IV. CASE STUDIES

To assess the performance, the new proposed algorithm is applied to two NP complete problems: route planning problem (RPP) and travelling salesman problem (TSP). A normal GA, i.e., an NS-based GA is also applied to solve these two problem for comparative purposes. For the sake of distinguishing, hereafter, the AS-based GA is denoted as ASGA, while the NS-based GA as NSGA.

#### A. Route planning problem

The Route Planning Problem (RPP) has a wide engineering background [6], [8]. Consider the route planning problem whose route network is given in Fig.6. On average, a route linking the starting node and the end node is composed of 10 nodes, and each node has on average 10 links to other nodes.

For NSGA, the size of solution space is estimated as

$$S_{NS} = 10^{10}. \quad (8)$$

For ASGA, we identify there is one sub-problem suitable for AS-related operations. The route network associated with this sub-problem has five levels: L1 to L5 as indicated in Fig.6. AS gene serials are constructed based on these five levels, and then the AS database is set up. For each gene serial, it starts with a node in L1 and ends with a node in L5, and represents the shortest link between these two nodes through the five levels. Then one has

$$S_{ASDB} = 4 \times 5 = 20. \quad (9)$$

Excluding the 5 nodes in an AS gene serial, there are on average another 5 genes left in a chromosome, which means

$$S_{AS} = S_{ASDB} 10^5 = 2 \times 10^6. \quad (10)$$

Therefore, one can see that the solution space of NSGA is about 5000 times larger than that of ASGA. The AS gene database can be easily established by employing dynamic programming.

Table I gives some simulation results. For both NSGA and ASGA, each has a population of 300 chromosomes, and the maximum evolutionary procedure is 200 generations long. Each GA is run 20 times, and then the average is taken. The sign “ $\geq$ ” in Table I means sometimes the NSGA ends up in a local optimum after 200 generations, while the ASGA finds the global optimum in all 20 runs. From Table I, one can conclude that the ASGA is more efficient than the NSGA in terms of both solution quality and speed. Fig.6 also gives examples of optimized routes under each GA.

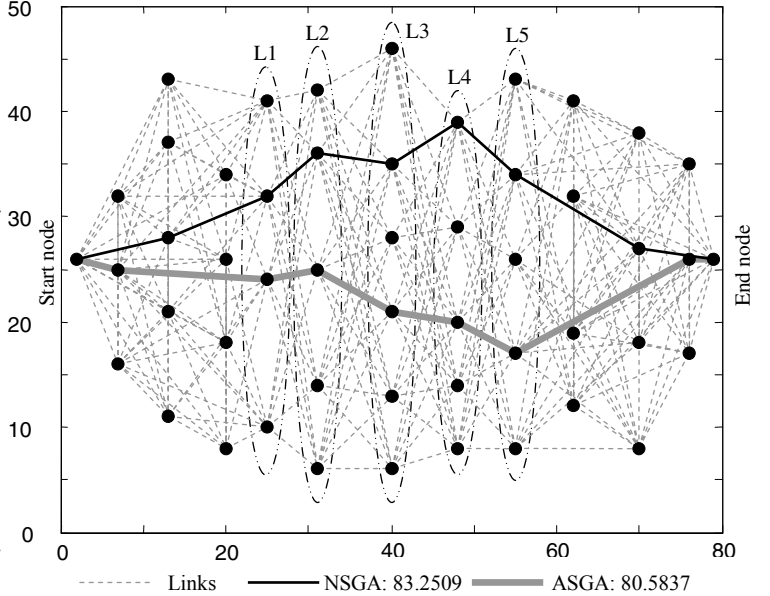


Fig. 6 Optimized routes under NSGA and ASGA

TABLE I. RESULTS OF ROUTE PLANNING PROBLEM

|      | Length of shortest route ever found | In which generation the shortest route is found |
|------|-------------------------------------|---|
| NSGA | 82.8493                             | $\geq 117$                                      |
| ASGA | 80.5837                             | 12  |

B. Travelling salesman problem

Many sequencing and scheduling problems can be converted into the Travelling Salesman Problem (TSP) [9], [10]. Consider a TSP with 50 cities to visit, and the distribution of cities is given in Fig.7. Since each city can link to any other city, the size of solution space for NSGA is

$$S_{NS} = 49!. \quad (11)$$

For the ASGA, we identify 4 groups of cities, G1 to G4, which cluster together in each group and are relatively far from other cities or groups, as indicated in Fig.7. Based on these 4 groups of cities, we construct 4 sub-problems which are suitable for AS related operations. In each sub-problem, randomly choosing any two cities in the group as starting point and end point, we need to find the shortest route which links all other cities in the group. In the chromosome, each group relates to an AS gene serial and then has its own AS gene database to save optimal serials. The sizes of AS gene database associated with G1, G2, G3 and G4 are respectively

$$S_{ASGA1}=5!, S_{ASGA2}=5!, S_{ASGA3}=8!, S_{ASGA4}=7!. \quad (12)$$

Suppose the start city is outside of these 4 AS groups (theoretically, the choice of start city should not affect the solution quality of TSP, so we can always choose a city outside of these 4 AS groups as the start city to generate a chromosome/solution). Since there are 25 cities not included in these 4 AS groups, one has

$$S_{AS} = 28 \times 8! \times 7! \times (5!)^2. \quad (13)$$

Therefore, the solution space of the NSGA is  $6.8179 \times 10^{20}$  times larger than that of the ASGA.

Table II gives some simulation results. For both the NSGA and ASGA, the population is of 300 chromosomes, and the maximum evolutionary procedure is 500 generations long. Each GA is run 20 times, and then the average is taken. Since we do not know what the global optimum in this TSP is, we just analyze in Table II how many generations are required for either GA to find a route which has a length of less than 300. From Table II, we can come to the same conclusion as we got from Table I that the ASGS has higher solution quality and faster speed. Fig.7 also gives examples of optimized routes under the NSGA and the ASGA.

TABLE II. RESULTS OF TSP

|      | Length of shortest route ever found | How generations for finding a route shorter than 300 |
|------|-------------------------------------|--|
| NSGA | 315.74                              | $\geq 389$   |
| ASGA | 289.15                              | $\geq 172$   |

V. CONCLUSIONS

This paper proposes a general methodology of improving the performance of GAs, where the concept of deterministic artificial selection is introduced to work together with the

mechanism of stochastic natural selection. Based on the concept of artificial selection, the new GA proposed in this paper is designed with an artificial selection gene database, which efficiently reduces the size of solution space and therefore greatly improves the optimization power. The artificial selection based GA may have a good potential for applications to dynamic complicated problems in real time. The advantages of the new GA are clearly illustrated by two case studies.

REFERENCES

- [1] E. Goldberg, *Genetic Algorithms in Search Optimization & Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [2] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.
- [3] Y. Wu, Z.H.H. Shao and X.Y. Wu. "A new self-adaptive genetic algorithm and its application". *Control Theory and Applications*, vol.16, no.1, pp. 127-129, 1999.
- [4] P.C. Pendharkar and J.A. Rodger, "An empirical study of impact of crossover operators on the performance of non-binary genetic algorithm based neural approaches for classification", *Computers & Operations Research*, vol.31, no.4, pp. 481-498, 2004.
- [5] X.L. Chao, Z. Zheng, N. Fan and X.F. Wang. "A modified genetic algorithm by integrating neural network technology". *Pattern Recognition and Artificial Intelligence*, vol.12, no.4, pp. 486-492, 1999.
- [6] X.B. Hu, S.F. Wu and J. Jiang, "On-Line Free-Flight Path Optimization Based on Improved Genetic Algorithms", *Engineering Applications of Artificial Intelligence*, vol.17, no.8, pp. 897-907, 2004.
- [7] X.B. Hu and W.H. Chen, "Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling", *Engineering Applications of Artificial Intelligence*, vol.18, no.5, pp. 633-642, 2005.
- [8] P.K.K. Loh and W.J. Hsu, "Fault-tolerant routing for complete Josephus Cubes", *Parallel Computing*, vol.30, no.9-10, pp. 1151-1167, 2004.
- [9] In-Chan I.-C. Choi, Seong-In S.-I. Kim, Hak-Soo H.-S. Kim, "A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem", *Computers & Operations Research*, vol.30, no.5, pp. 773-786, 2003.
- [10] L. Bianco, P. Dell'Olmo and S. Giordani, "Scheduling models and algorithms for TMA traffic management" in: Bianco, L., P. Dell'Olmo and A.R. Odoni (Eds.), *Modelling and Simulation in Air Traffic Management*. Springer Verlag, pp.139-167, 1997.

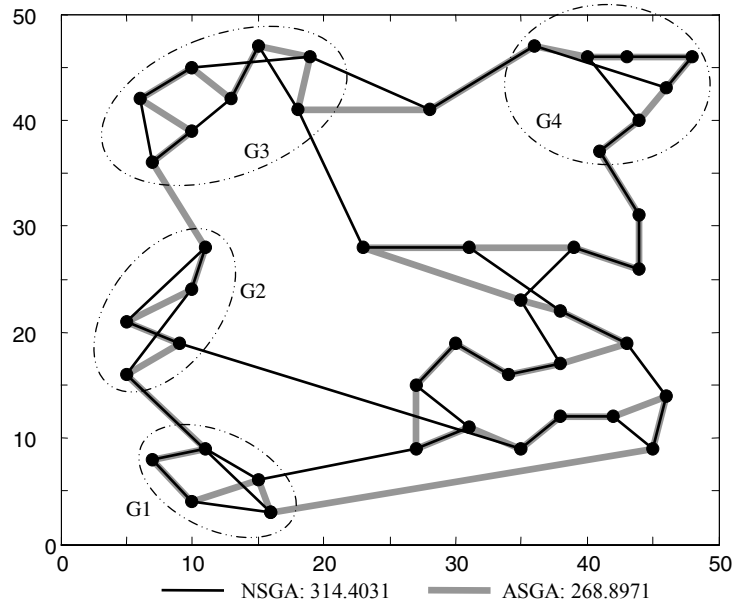


Fig. 7 Optimized routes under NSGA and ASGA