

New evolutionary genetic algorithms for NP-complete combinatorial optimization problems

Fam Quang Bac, V. L. Perov

Mendeleev Institute of Chemical Engineering, ul. Vilisa Lasisa, d. 21/1, 123480 Moscow, Russian Federation

Received: 30 June 1992/Accepted: 15 February 1993

Abstract. Evolutionary genetic algorithms have been proposed to solve NP-complete combinatorial optimization problems. A new crossover operator based on group theory has been created. Computational processes motivated by proposed evolutionary genetic algorithms were described as stochastic processes, using population dynamics and interactive markovian chains. The proposed algorithms were used in solving flowshop problems and an asymmetric traveling salesman problem. The experimental results showed the superiority of new evolutionary algorithms in comparison with the standard genetic algorithm.

1 Introduction

This paper deals with optimization problems which belong to a large family of problems classified as NP-complete; that is, an exact solution requires a number of computational steps that grows faster than any finite power of some appropriate measure of the size of the problem. Many problems in science and engineering fields such as expert systems, operations research, flexible manufacturing systems, multiproduct batch chemical or food plants can be formulated as a traveling salesman problem (TSP) and scheduling problems that belong to the class of problems considered.

As an optimal solution for NP-complete problems cannot be efficiently found, a vast amount of time has been spent on studying approximate-solution algorithms whose running time is proportional to small powers of n .

A large number of papers have been devoted to NP-complete problems. A traditional approach of operational research including local and heuristic methods has been used (Lin and Kernighan 1973; Lawler et al. 1985). Kirkpatrick et al. (1983) used simulated anneal-

ing based on the Metropolis procedure (Metropolis et al. 1953) and annealing scheduling. Hopfield and Tank (1985) applied a neural network of totally interconnected analog neurons which relaxes toward the minimum of an energy function for finding a suboptimal solution for TSP. In recent years interest has risen in the application of evolutionary algorithms to combinatorial optimization problems (Holland 1975; Brady 1985; Grefenstette et al. 1985; Boseniuk et al. 1987; Fontana and Schuster 1987; Wang 1987; Fogel 1988; Banzhaf 1990).

This paper introduces algorithms based on the evolutionary approach and a new genetic crossover operator.

In Sect. 2 the generalized formulation of combinatorial optimization problems is defined and some concrete problems are considered. In Sect. 3 the Goldberg genetic algorithm (Goldberg and Lingle 1985) is analyzed. In Sect. 4 the new crossover operator based on the non-Abel group theory is created. In Sect. 5 the evolutionary procedures are proposed. The efficiency of the proposed algorithms is assessed in solving some TSP and scheduling problems in Sect. 6. Finally, in Sect. 7 some conclusions are drawn.

2 Combinatorial optimization problems

Many combinatorial optimization problems can be formulated as follows. Let $\pi = \{i_1, i_2, \dots, i_n\}$, some permutation from set $\{1, 2, \dots, n\}$; Ω , a space of feasible solutions (states); and $f(\pi)$, the optimality function (criterion). It is necessary to find π^* such that

$$\pi^* = \{i_1^*, i_2^*, \dots, i_n^*\} = \arg\{f(\pi) \rightarrow \min_{\pi \in \Omega}\} \quad (1)$$

It is important to note that the structure of Ω and $f(\pi)$ depends on the features of the problems considered. Next we consider TSP and some scheduling problems.

2.1 Asymmetric TSP (Lawler et al. 1985)

$$\pi^* = \{i_1^*, i_2^*, \dots, i_n^*\}$$

$$= \arg \left\{ f(\pi) = \theta_\phi + \sum_{q=1}^{n-1} \theta_{i_q i_{q+1}} + \theta_{i_n \phi} \rightarrow \min_{\pi \in \Omega} \right\}$$

$$\Omega \equiv S_n \quad \text{and} \quad |\Omega| = |S_n| = n! \quad (2)$$

where ϕ is the first and last city; i_q the number of cities through which the salesman travels in the q th order of the path π ; θ_{jk} , the distance between cities j and k ; and S_n the full set of all permutations from $\{1, 2, \dots, n\}$.

2.2 Single-machine scheduling problem (Lawler et al. 1989)

$$\Omega = \left\{ \begin{array}{l} T_{i_1} = t_{i_1} \\ T_{i_q} = T_{i_{(q-1)}} + \theta_{i_{(q-1)}, i_q} + t_{i_q} \end{array} \quad (q = \overline{2, n}) \right\} \quad (3)$$

$$\pi^* = \arg \left\{ f(\pi) = \sum_{j=1}^n \max(0, T_j - D_j) \rightarrow \min_{\pi \in \Omega} \right\} \quad (4)$$

where $\pi = \{i_1, i_2, \dots, i_q, \dots, i_n\}$, the sequence of jobs processed in the machine; t_i, D_i , the processing time and due date of the i th job, respectively; θ_{jk} , the setup time from the j th to the k th job; and T_{i_q} , the moment of processing completion of the i_q th job. The optimality criterion $f(\pi)$ is total tardiness. $|\Omega| = |S_n| = n!$

2.3 Flowshop problem (Lawler E. et al. 1989)

For the i_1 th job

$$T_{i_1, 1} = t_{i_1, 1}; \quad T_{i_1, j} = T_{i_1, (j-1)} + T_{i_1, j} \quad (j = \overline{2, m}) \quad (5)$$

For the i_q th job ($q = \overline{2, n}$)

$$T_{i_q, 1} = T_{i_{(q-1)}, 1} + \theta_{i_{(q-1)}, i_q} + t_{i_q, 1} \quad \text{for } j = \overline{2, m}$$

$$\text{Then } T_{i_q, j} = \max\{T_{i_q, (j-1)}; [T_{i_{(q-1)}, j} + \theta_{i_{(q-1)}, i_q}]\} + t_{i_q, j}$$

$$\pi^* = \{i_1^*, i_2^*, \dots, i_n^*\} = \arg\{f(\pi) = T_{i_n, m} \rightarrow \min_{\pi \in \Omega}\} \quad (6)$$

where n, m is the number of jobs and machines, respectively; $t_{i, j}$, the processing time of the i th job in the j th machine; θ_{ik}^j , the set up time from the i th to the k th job in the j th machine; and T_{i, j_q} , the moment of processing completion of the i_q th job in the j th machine. The optimality criterion $f(\pi)$ is makespan: maximum completion time. $|\Omega| = |S_n| = n!$ All problems considered above belong to the class of NP-complete problems according to Stirling's formula $n! = \sqrt{2\pi \cdot n} \cdot n^n \cdot e^{-n}$ and $n! > a^n$ with any given positive number a .

3 Genetic algorithms

The genetic algorithm is a "pseudorandom" search operator which provides the means to solve NP-complete problems. Holland (1975) developed the genetic algorithm and provided its theoretical foundation in *Adaptation in Natural and Artificial Systems*. Holland's

formulation was motivated by the observation that reproduction in conjunction with the pressure of natural selection has developed species remarkably well adapted to their environment. The crossover operator as the discovery mechanism played the central role in this algorithm.

Although there are many possible variants of the basic genetic algorithm (Liepins and Hilliard 1989), the fundamental underlying mechanism operates on a population of individuals, is relatively standard, and consists of three operations: (1) evaluation of individual fitness, (2) formulation of a gene pool, and (3) recombination and mutation. The individuals resulting from these three operations form the next generation's population.

Goldberg and Lingle (1985) suggested a crossover operator, the "partially mapped crossover (PMX)", which they believe will lead to an efficient solution of the TSP. PMX proceeds as follows.

Consider two tours A and B of the TSP:

A = 9 8 4 5 6 7 1 3 2 10

B = 8 7 1 2 3 10 9 5 4 6

If two random numbers, for example, 4 and 6, are chosen as the crossover points for these two genes, then each string is to be modified by the permutations (5 2), (6 3), and (7 10):

A = 9 8 4 | 5 6 7 | 1 3 2 10
B = 8 7 1 | 2 3 10 | 9 5 4 6

and the respective results become

A' = 9 8 4 2 3 10 1 6 5 7

B' = 8 10 1 5 6 7 9 2 4 3

Formally, the genetic algorithm is specified as follows (Liepins and Hilliard 1989):

1. Choose a desired population size N and initialize the starting population p .

2. Evaluate individuals according to the function $f(\pi)$.

3. If the stopping criterion is met, stop; if not, probabilistically select individuals (according to their fitness determined by the function evaluation) to populate a gene pool of size N .

4. So long as the size of the temporary population TP is less than N , randomly choose a parent from the gene pool. With probability p_1 , inject the parent into the temporary population. With probability $(1 - p_1)$, choose a coparent, cross the parents, and inject the resultant offspring into the temporary population.

5. With low probability p_2 , mutate randomly chosen individuals of TP . Set $P = TP$ and return to step 2.

We now present a more intelligent algorithm (GA ϕ):

1. Choose a desired population size $2N$ and randomly initialize the starting population $P = \{\pi_j \mid j = \overline{1, 2N}\}$; $r = 0$.

2. Compute $f(\pi_j) \forall j = \overline{1, 2N}$.

$$f^{\text{best}} = \min_{j=\overline{1, 2N}} f(\pi_j)$$

$$S(\pi_j) = [f^{\text{max}} - f(\pi_j)] / \sum_{k=1}^{2N} [f^{\text{max}} - f(\pi_k)]$$

for $(j = \overline{1, 2N})$ where $f^{\text{max}} = \max_{j=\overline{1, 2N}} f(\pi_j)$

3. Form the next generation's population P' .

- Randomly select π_j, π_k from P based on S_j, S_k
- Use the Goldberg crossover operator to obtain two offspring π'_j and π'_k . Place π'_j, π'_k in P' .
- Repeat steps a and b until $|P'| = 2N$.

4. With low probability p , mutate all individuals of P' .

5. Compute $f_{\min} = \min_{j=\overline{1, 2N}} f(\pi'_j)$.

6. If $f_{\min} > f^{\text{best}}$, then $r = r + 1$; if not, then $r = 0$.

7. If stopping criterion is met ($r > R$), stop; if not, set $P = P'$ and return to step 2.

4 New crossover operator

In this section we suggest the new crossover operator created by the non-Abel group theory.

Consider two permutations:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 9 & 8 & 4 & 5 & 6 & 7 & 1 & 3 & 2 & 10 \end{pmatrix};$$

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 8 & 7 & 1 & 2 & 3 & 10 & 9 & 5 & 4 & 6 \end{pmatrix}$$

According to the non-Abel group theory:

- If $A, B \in \Omega$ and $C = A \otimes B$, then $C \in \Omega$.
- $(A \otimes B) \otimes C = A \otimes (B \otimes C) = A \otimes B \otimes C$.
- $A \otimes B \neq B \otimes A$.
- $\exists e: e \otimes A = A \otimes e = A$.

$$\forall A, \exists A^{-1}: A \otimes A^{-1} = A^{-1} \otimes A = e.$$

Take A' and B' as offspring obtained from A and B :

$$A' = A \otimes B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 9 & 8 & 4 & 5 & 6 & 7 & 1 & 3 & 2 & 10 \end{pmatrix}$$

$$\otimes \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 8 & 7 & 1 & 2 & 3 & 10 & 9 & 5 & 4 & 6 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 5 & 2 & 3 & 10 & 9 & 8 & 1 & 7 & 6 \end{pmatrix}$$

$$B' = B \otimes A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 8 & 7 & 1 & 2 & 3 & 10 & 9 & 5 & 4 & 6 \end{pmatrix}$$

$$\otimes \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 9 & 8 & 4 & 5 & 6 & 7 & 1 & 3 & 2 & 10 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 1 & 9 & 8 & 4 & 10 & 2 & 6 & 5 & 7 \end{pmatrix}$$

When creating the crossover operator algorithmically, we use the following formulas:

If

$$\pi_k = [k(1), k(2), \dots, k(n)]; \quad \pi_j = [j(1), j(2), \dots, j(n)],$$

$$\text{then } \pi_{i1} = \pi_k \otimes \pi_j = [i_{(1)}^1, i_{(2)}^1, \dots, i_{(n)}^1]$$

$$\text{where } i_{(1)}^1 = j[k(1)]; \quad i_{(2)}^1 = j[k(2)], \dots; i_{(n)}^1 = j[k(n)]$$

$$\text{and } \pi_{i2} = \pi_j \otimes \pi_k = [i_{(1)}^2, i_{(2)}^2, \dots, i_{(n)}^2]$$

$$\text{where } i_{(1)}^2 = k[j(1)]; \quad i_{(2)}^2 = k[j(2)], \dots, i_{(n)}^2 = k[j(n)]$$

Lemma 1. For an arbitrarily chosen permutation A , it is able to find $n!$ different pairs of permutations X and Y , such that $X \otimes Y = A$.

Proof. It is easy to prove Lemma 1 if X is arbitrarily chosen. $X \in \Omega$ ($|\Omega| = n!$) and $Y = X^{-1} \otimes A$.

5 Evolutionary algorithms

5.1 Algorithm 1 (mutation and crossover)

1. Choose a desired population size $2N$ and randomly generate the starting population $P = \{\pi_j | j = \overline{1, 2N}\}$. Set $r = 0$.

2. Evaluate the objective function for each individual $f(\pi_j) | j = \overline{1, 2N}$ and best value $f^{\text{best}} = \min_{j=\overline{1, 2N}} f(\pi_j)$.

3. Mutate all individuals of the population using the mutation operator (randomly choose a number q and exchange places of adjacent genes i_q and i_{q+1}).

4. Repeat step 3 N_m times.

5. Crossover N randomly chosen pairs π_j, π_k from the last mutated population to form new population P' (by the Goldberg operator or the non-Abel operator).

6. Compute $f_{\min} = \min_{j=\overline{1, 2N}} f(\pi'_j)$

7. If $f_{\min} > f^{\text{best}}$, then $r = r + 1$; if not, then $r = 0$.

8. If stopping criterion is met ($r > R$), stop; if not, set $P = P'$ and return to step 2.

We will now consider mathematical models which describe stochastic processes on interactive markovian chains (markovian nets).

$x_i(t)$ denotes the number of individuals of the i th state at time t . $\sum_{i \in \Omega} x_i(t) = 2N$ is the total number of individuals of the evolved population.

5.1.1 The continuous model

$$\begin{aligned} \frac{dx_i(t)}{dt} &= \sum_{\substack{j \in \Omega_i \\ j \neq i}} D_{ji} * x_j(t) - \sum_{\substack{j \in \Omega_i \\ j \neq i}} D_{ij} * x_i(t) \\ &+ \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{kj}^i * x_k(t) * x_j(t) \\ &- \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{ij}^k * x_i(t) * x_j(t) \quad (\forall i \in \Omega) \\ &\times \sum_{i \in \Omega} x_i(t) = 2N \end{aligned}$$

where Ω_i is the local neighborhood of the i th state and D_{ij} , the probability of a mutation's generation of the j th state in the Ω_i .

$$D_{ij} = 1/|\Omega_i| \quad \forall j \neq i \quad \text{and} \quad j \in \Omega_i; \quad \sum_{j \in \Omega_i} D_{ij} = 1$$

If $\pi_k \otimes \tau_j = \pi_i$, $P_{kj}^i = 1$; if not, then $P_{kj}^i = 0$

5.1.2 The discrete model.

1. Mutation stage

$$\bar{x}_{(t)}^{\text{mutation}} = [D^{\text{trans}}]^{N_m} * \bar{x}(t)$$

$$D_{ij} = 1/|\Omega_i| \quad \forall j \neq i \quad \text{and} \quad j \in \Omega_i;$$

$$D_{ij} = 0 \quad \forall j \notin \Omega_i; \quad D_{ii} = 0;$$

and

$$\sum_{j \in \Omega} D_{ij} = 1$$

where N_m is the length of the markovian chains.

2. Crossover stage

$$x_i(t+1) = \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{kj}^i * x_k^{\text{mutation}} * x_j^{\text{mutation}}$$

$$\sum_{i \in \Omega} x_i(t) = \sum_{i \in \Omega} x_i(t+1) = 2N \quad \forall i \in \Omega$$

If $\pi_k \otimes \pi_j = \pi_i$, $P_{kj}^i = 1$; if not, then $P_{kj}^i = 0$.

5.2 Algorithm 2 (local search and crossover)

1. Choose a desired population size $2N$ and randomly generate the starting population $P = \{\pi_j, j = \overline{1, 2N}\}$. Set $r = 0$.

2. Carry out a parallel local search using the “greedy” algorithm method, starting from individuals of the population P and resulting in the new population of locally optimal individuals $P^{LS} = \{\pi_j^{LS}, j = \overline{1, 2N}\}$.

3. Evaluate the objective function for each individual $f(\pi_j^{LS})$, $j = \overline{1, 2N}$ and the best value $f^{\text{best}} = \min_{j = \overline{1, 2N}} f(\pi_j^{LS})$.

4. Crossover N randomly chosen pairs π_j^{LS} , π_k^{LS} from population P^{LS} to receive N pairs of offspring and to form new population P' (by the Goldberg operator or the non-Abel operator).

5. Repeat step 2 for the population P' and receive a population $P'^{LS} = \{\pi_j'^{LS}, j = \overline{1, 2N}\}$.

6. Compute $f_{\min} = \min_{j = \overline{1, 2N}} f(\pi_j'^{LS})$.

7. If $f_{\min} > f^{\text{best}}$, then $r = r + 1$; if not, then $r = 0$.

8. If the stopping criterion is met ($r > R$), stop; if not, set $P^{LS} = P'^{LS}$ and return to step 3.

Consider the following mathematical models for algorithm 2.

5.2.1 The continuous model.

$$\begin{aligned} \frac{dx_i(t)}{dt} = & \sum_{\substack{j \in \Omega_i \\ j \neq i}} D_{ji} * x_j(t) - \sum_{\substack{j \in \Omega_i \\ j \neq i}} D_{ij} * x_i(t) \\ & + \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{kj}^i * x_k(t) * x_j(t) \\ & - \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{ij}^k * x_i(t) * x_j(t) \quad (\forall i \in \Omega) \end{aligned}$$

and

$$\sum_{i \in \Omega} x_i(t) = 2N$$

If the i th state is the local minimum of the j th state, $D_{ji} = 1$; if not, then $D_{ji} = 0$.

$$\sum_{j \in \Omega} D_{ij} = 1$$

If $\pi_k \otimes \pi_j = \pi_i$, $P_{kj}^i = 1$; if not, then $P_{kj}^i = 0$.

5.2.2 The discrete model.

1. Local search stage

$$x_i^{LS}(t) = \sum_{j \in \Omega} D_{ji} * x_j(t) \quad (\forall i \in \Omega)$$

If the i th state is local minimum of the j th minimum of the j th state, $D_{ji} = 1$; if not, then $D_{ji} = 0$.

$$\sum_{j \in \Omega} D_{ij} = 1$$

2. Crossover stage

$$x_i(t+1) = \sum_{k \in \Omega} \sum_{j \in \Omega} \frac{1}{(2N)^2} * P_{kj}^i * x_k^{LS}(t) * x_j^{LS}(t)$$

$\forall i \in \Omega$

$$\sum_{i \in \Omega} x_i(t) = \sum_{i \in \Omega} x_i(t+1) = 2N$$

If $\pi_k \otimes \pi_j = \pi_i$, $P_{kj}^i = 1$; if not, then $P_{kj}^i = 0$.

6 Experimental results

We present some results of computer simulations of the proposed algorithms for the flowshop problem and for an asymmetric TSP. We denote algorithm 1;2 with the Goldberg operator PMX1; PMX2; and algorithm 1;2 with the non-Abel operator, NAbel1; NAbel2.

For the flowshop problem ($n = 12$; $m = 4$), processing times t_{ij} ($i = \overline{1, n}$; $j = \overline{1, m}$) were generated from a uniform distribution over the range [50–100]; setup times θ_{ik}^j ($i, k = \overline{1, n}$; $j = \overline{1, m}$) were generated from a uniform distribution over the range [10–150]. The results of the performance of proposed algorithms are presented in Table 1 under the condition that CPU times are approximately equivalent.

For asymmetric TSP ($n = 16$), distances θ_{jk} ($j, k = \overline{1, n}$) were generated from a uniform distribution over the range [0–100]. The results of the performance of proposed algorithms are presented in Table 2 under the condition that CPU times are approximately equivalent.

Thus, the experimental results show that in each case the performance of NAbel2 and PMX2 is superior to that of a standard genetic algorithm (GA ϕ). Our interpretation is that the population, which quickly evolved in GA ϕ , is occupied by the individuals of the small number of states. In other words, a diversity of the evolved population is quickly reduced and the evolutionary process, quickly finished. NAbel2 and PMX2

Table 1. Results of a performance of the algorithms for the flowshop problem ($n = 12$ jobs; $m = 4$ machines)

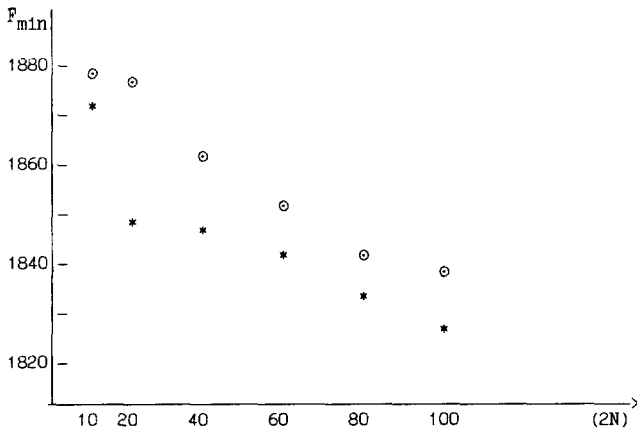
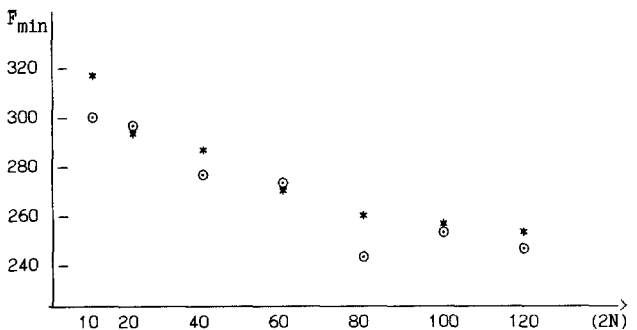
	GA ϕ $2 = 300; R = 10$	NAbel1 $2N = 40; N_m = 20; R = 10$	NAbel2 $2N = 20; R = 10$	PMX2 $2N = 20; R = 10$
$\bar{F}_{\min}$	$1.93769e + 03$	$1.90508e + 03$	$1.84833e + 03$	$1.87765e + 03$
SD	7.28	10.44	14.64	3.27
	$2N = 500; R = 10$	$2N = 40; N_m = 50; R = 10$	$2N = 40; R = 10$	$2N = 40; R = 10$
$\bar{F}_{\min}$	$1.91144e + 03$	$1.89645e + 03$	$1.84679e + 03$	$1.86116e + 03$
SD	4.17	8.47	5.46	11.76

$\bar{F}_{\min}$ is the average function value of the five trials

Table 2. Results of a performance of the algorithms for asymmetric TSP ($n = 16$ cities)

	GA ϕ $2N = 100; R = 5$	NAbel2 $2N = 2; R = 5$	PMX2 $2N = 6; R = 5$
$\bar{F}_{\min}$	$4.1096e + 02$	$3.4386e + 02$	$3.1125e + 02$
SD	12.21	28.15	11.44
	$2N = 200; R = 5$	$2N = 10; R = 5$	$2N = 10; R = 5$
$\bar{F}_{\min}$	$3.8995e + 02$	$3.1887e + 02$	$2.9724e + 02$
SD	13.94	12.45	11.45

$\bar{F}_{\min}$ is the average function value of the five trials

**Fig. 1.** Comparison of NAbel2 and PMX2 for various population sizes ($2N$) for flowshop problem. $\bar{F}_{\min}$ is the average function value of the five trials. $\odot$, Algorithm PMX2; *, algorithm NAbel2**Fig. 2.** Comparison of NAbel2 and PMX2 for various population sizes ($2N$) for asymmetric TSP. $\bar{F}_{\min}$ is the average function value of the five trials. $\odot$, Algorithm PMX2; *, algorithm NAbel2

perform better than NAbel1 because there is no selection in the latter.

In order to compare NAbel2 and PMX2 in more detail, we show how their efficiency depends on the population's size $2N$ for the flowshop problem (Fig. 1) and for asymmetric TSP (Fig. 2). In fact, NAbel2 performs better than PMX2 in solving the flowshop problem (Fig. 1). In solving the asymmetric TSP both NAbel2 and PMX2 perform approximately equally (Fig. 2).

7 Conclusions

We have shown that many problems can be formulated as NP-complete combinatorial optimization problems. We have created a new crossover operator based on group theory. Two new evolutionary algorithms have been suggested and described by stochastic models (interactive markovian chains). Experimental results showed a superiority of proposed algorithms compared with the standard genetic algorithm in solving a flowshop problem and an asymmetric TSP.

New theoretical research and practical applications of evolutionary genetic algorithms will be the focus of our future work.

References

- Banzhaf W (1990) The "molecular" traveling salesman. *Biol Cybern* 64:7-14
- Boseniuk T, Ebeling W, Engel A (1987) Boltzmann and Darwin strategies in complex optimization. *Phys Lett* 125A:307-310
- Brady RM (1985) Optimization strategies learned from natural evolution. *Nature* 317:804-806
- Fogel DB (1988) An evolutionary approach to the traveling salesman problem. *Biol Cybern* 60:139-144
- Fontana W, Schuster P (1987) A computer model of evolutionary optimization. *Biophys Chem* 26:123-147
- Goldberg DE, Lingle R (1985) Alleles, loci, and the traveling salesman problem. In: *Proceedings of the International Conference on Genetic Algorithms and Their Applications*. Carnegie Mellon University, Pittsburgh, ed. Grefenstette, Lawrence Erlbaum, Hillside, N.J. pp 154-159
- Grefenstette JJ, Gopal R, Rosmaita B, Van Gucht D (1985) Genetic algorithms for the traveling salesman problem. In: *Proceedings of the International Conference on Genetic Algorithms and Their Applications*. Carnegie Mellon University, Pittsburgh, ed. Grefenstette, Lawrence Erlbaum, Hillside, N.J. pp 160-168
- Holland JH (1975) *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor

- Hopfield JJ, Tank DW (1985) "Neural" computation of decisions in optimization problems. *Biol Cybern* 52:141–152
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220:671–680
- Lawler EL, Lenstra JK, Rinnooy Kan AHG (1985) *The traveling salesman problem*. Wiley, New York
- Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1989) Sequencing and scheduling. Algorithms and complexity. Report BS-R8909. Centre for Mathematics and Computer Science, Amsterdam
- Liepins GE, Hilliard MR (1989) Genetic algorithms. Foundations and applications. *Ann Oper Res* 21:31–58
- Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling salesman problem. *Oper Res* 21:498–516
- Metropolis N, Resenbluth AW, Resenbluth MN, Teller AH, Teller EJ (1953) Equations of state calculations by fast computing machines. *J Chem Phys* 21:1087–1091
- Wang Q (1987) Optimization by simulating molecular evolution. *Biol Cybern* 57:95–101