Tạp chí Tin học và Điều khiển học, T.21, S.2 (2005), 143–150

CƠ CẤU LỰA CHỌN THÍCH NGHI TOÁN TỬ LAI GHÉP TRONG GIẢI THUẬT DI TRUYỀN MÃ HOÁ SỐ THỰC

NGUYỄN THANH THỦY[1], VŨ MẠNH XUÂN[2]

[1]*Khoa Công nghệ thông tin, ĐH Bách khoa HN*
[2]*Khoa Công nghệ thông tin, ĐH Thái Nguyên*

Abstract. Self-adaptation is an essential feature of the natural evolution. Among the evolutionary methods, self-adaptation properties have been explored with evolutionary strategies, evolutionary programming and genetic algorithms. In this paper, we introduce an adaptive mechanism of selecting crossover operators in real-code genetic algorithms. Exprimentations have showed the efficiency of the proposed approach.

Tóm tắt. Tự thích nghi là một đặc trưng chủ yếu của tiến hóa tự nhiên. Trong các phương pháp tiến hóa, các tính chất tự thích nghi đã được khảo sát trong chiến lược tiến hóa, chương trình tiến hóa và giải thuật di truyền. Trong bài báo này, chúng tôi giới thiệu một cơ cấu lựa chọn thích nghi toán tử lai ghép trong giải thuật di truyền mã hóa số thực. Các thực nghiệm đã chứng tỏ hiệu quả của cách tiếp cận này.

1. ĐẶT VẤN ĐỀ

Tính toán tiến hóa (Evolutionary Computations - EC) bao gồm giải thuật di truyền (Genetic Algorithm - GA), chương trình di truyền (Evolutionary programming - EP) và chiến lược tiến hóa (Evolution Strategies - ES) dựa trên nền tảng tiến hóa tự nhiên đó cũng là các phương pháp tự nhiên nhằm giải quyết bài toán tối ưu và tìm kiếm. Mục tiêu cơ bản của EC là cơ cấu tính toán nhằm tạo ra sự tiến hóa của quần thể gồm nhiều cá thể với mục đích quần thể sau "tốt hơn" quần thể trước. Các toán tử sử dụng trong EC bao gồm: lai ghép (crossover), đột biến (mutation) và chọn lọc (selection). Các toán tử này kết hợp với nhau trong một mô hình tiến hóa và được điều khiển bởi một vài tham số như kích cỡ quần thể, xác suất lai ghép, xác suất đột biến...

Tính toán tiến hóa thích nghi (Adaptive Evolutionary Computations) đã được nhiều tác giả ([2, 3, 6, 7, 9]) nghiên cứu, chủ yếu nhằm điều chỉnh các tham số tác động đến quá trình tiến hóa ngay trong khi thực hiện chương trình, làm tăng tốc độ tìm kiếm. Peter J. Angeline [6] đã phân tích các tính toán tiến hóa thích nghi và chia ra 3 lớp chính: thích nghi lớp quần thể, thích nghi lớp cá thể và thích nghi lớp thành phần. Các tác giả [2, 4, 6, 9] cũng quan tâm tới vấn đề tự thích nghi (self-adaptive) trong tính toán tiến hóa nói chung và giải thuật di truyền nói riêng.

Bài báo này đề xuất một cơ chế lựa chọn thích nghi toán tử lai ghép trong thuật toán di truyền mã hóa số thực bằng cách chọn toán tử có tiềm năng hơn để tiến hành tiến hóa. Tư

tưởng chính ở đây là: Dựa trên hai toán tử lai ghép đã có, ban đầu một toán tử được chọn để thực hiện một cách ngẫu nhiên. Sau đó, xác suất để lựa chọn toán tử được tính dựa trên tỷ lệ áp dụng thành công của toán tử đó (khái niệm thành công ở đây được hiểu là sau lần tạo sinh, xuất hiện cá thể con có độ thích nghi cao hơn cha mẹ chúng). Cơ chế này có thể sử dụng với số các toán tử đề xuất nhiều hơn 2, trong quá trình thực hiện chương trình, toán tử nào có tiềm năng hơn sẽ có cơ hội được chọn áp dụng nhiều hơn và như vậy khả năng hội tụ nhanh hơn.

Bài báo được cấu trúc như sau: Phần 1 giới thiệu cơ sở chung về tính toán tiến hóa thích nghi. Phần 2 trình bày sơ lược các toán tử lai ghép sẽ được sử dụng trong thử nghiệm và một mô hình tiến hóa (MGG - Minimal Generation Gap). Phần 3 đưa ra cơ chế lựa chọn thích nghi toán tử lai ghép được đề xuất. Phần 4 nêu các kết quả thử nghiệm và Phần 5 là kết luận.

2. TÍNH TOÁN TIẾN HÓA TỰ THÍCH NGHI
(SELF-ADAPTIVE EVOLUTIONARY COMPUTATIONS)

Một cách hình thức như trong [6], có thể xem bài toán chuẩn trong tính toán tiến hóa EC là một hàm $F : V \to R$, trong đó V là không gian tìm kiếm, F được gọi là hàm thích nghi chứa đựng các thông tin đặc trưng cho bài toán. Mỗi vòng lặp của tính toán tiến hóa có thể định nghĩa bởi công thức:

$$v_{i+1} = \beta(v_i, F(v_i)),$$

trong đó $\beta : V \times R \to V$ là một hàm tạo ra một vectơ mới, nghĩa là một quần thể mới v_{i+1} từ quần thể cũ v_i, β thường được xác định bởi dạng tính toán đang sử dụng, cách chọn quần thể mới từ quần thể cũ và các con mới sinh ra cùng với các toán tử điều khiển chúng.

Tính toán tiến hóa thích nghi có thể định nghĩa là một hàm

$$v_{i+1} = \beta_\delta(v_i, F(v_i), \delta_i),$$

trong đó $\beta_\delta : V \times R \times \delta \to V$, hàm này có bổ sung một đối số δ được gọi là tham số thích nghi hay tham số chiến lược. Các tham số thích nghi này điều khiển quá trình tiến hóa theo hướng tạo ra thế hệ mới có các con tốt hơn cha mẹ chúng.

Có hai dạng quy tắc cập nhật các tham số thích nghi là quy tắc cập nhật vô điều kiện và quy tắc cập nhật theo kinh nghiệm. Quy tắc cập nhật vô điều kiện dựa trên tính toán thống kê trên tập các lần tạo sinh và các quần thể. Quy tắc cập nhật theo kinh nghiệm chỉ rõ một hàm đột biến cho các tham số thích nghi và quá trình cạnh tranh vốn có trong EC để quyết định xem việc thay đổi các tham số mang tính tích cực. Một tính toán tiến hóa biến đổi cả các tham số thích nghi được gọi là tự thích nghi.

Tự thích nghi là một trong những đặc trưng chủ yếu của tiến hóa tự nhiên. Khi mô phỏng hiện tượng này trong tính toán tiến hóa có thể phân làm ba lớp cơ bản tùy thuộc vào các tham số thích nghi ([6]): kỹ thuật thích nghi lớp quần thể (population-level) chỉnh lý các tham số chung nhất cho toàn bộ quần thể chẳng hạn như tần số lai ghép, phương pháp thích nghi lớp cá thể (individual-level adaptive) hiệu chỉnh một cá thể trong quần thể tương tự như phép đột biến và thích nghi lớp thành phần (component-level) biến đổi các thành phần

trong một cá thể một cách độc lập. Tất cả các lớp này đều nhằm một mục đích điều chỉnh một cách năng động các giá trị của các tham số chiến lược theo hướng tạo được thế hệ mới chứa các con cháu có độ thích nghi cao hơn.

Bài báo này đề xuất một cơ chế lựa chọn thích nghi không phải theo hướng điều chỉnh các tham số của giải thuật di truyền. Với một bài toán tối ưu, có thể thiết lập hai hay một vài toán tử lai ghép, trong quá trình tiến hóa sẽ quyết định chọn sử dụng toán tử này hay toán tử kia tùy thuộc vào hiệu quả của nó.

3. MỘT SỐ TOÁN TỬ LAI GHÉP TRONG GA MÃ HÓA SỐ THỰC

Xét bài toán tối ưu $\min f(x_1, x_2, ..., x_n)$ trên miền D thuộc không gian R^n.

Trong các bài toán dạng này, một cách tự nhiên thường sử dụng GA mã hóa số thực, mỗi cá thể được biểu thị bởi một vectơ thực trong R^n (tất nhiên phải thuộc miền xác định D). Mỗi quần thể kích cỡ m (có m cá thể) có thể biểu diễn như một ma trận thực cấp $m \times n$. Với GA mã hóa số thực, có nhiều dạng toán tử lai ghép khác nhau đã được giới thiệu. Trong bài này chúng tôi chỉ tập trung trình bày một vài dạng của toán tử lai ghép trong GA mã hóa số thực được sử dụng trong các thử nghiệm.

Các toán tử lai ghép nêu dưới đây đã được trình bày trong [2, 8, 10].

3.1. Lai số học

Với hai cá thể cha mẹ $p_1 = (x_1, x_2, ..., x_n)$ và $p_2 = (y_1, y_2, ..., y_n)$ các cá thể con được sinh ra như sau:

$$c_1 = \lambda p_1 + (1 - \lambda)p_2; \quad c_2 = (1 - \lambda)p_1 + \lambda p_2.$$

3.2. Lai đơn giản

Phép lai này tương tự như lai một điểm của GA kinh điển. Với một vị trí k chọn ngẫu nhiên $(1 < k < n)$, các cá thể con được sinh ra như sau:

$$c_1 = (x_1, ..., x_k, y_{k+1}, ..., y_n); \quad c_2 = (y_1, ..., y_k, x_{k+1}, ..., x_n).$$

3.3. Lai ghép mặt nạ

Phép lai này khởi tạo một vectơ ngẫu nhiên $r = (r_1, r_2, ..., r_n)$ trong đó các r_i chỉ là 0 hay 1. Sau đó cá thể con được sinh ra như sau:

$c_1 = (z_1, z_2, ..., z_n)$ trong đó $z_i = x_i$ nếu $r_i = 1$ và $z_i = y_i$ nếu $r_i = 0$.

$c_2 = (u_1, u_2, ..., u_n)$ trong đó $u_i = x_i$ nếu $r_i = 0$ và $u_i = y_i$ nếu $r_i = 1$.

3.4. Lai ghép BLX-α

Phép lai này chỉ tạo một cá thể con từ 2 cá thể cha mẹ. Mỗi thành phần z_i của cá thể con được chọn theo phân phối ngẫu nhiên đều trong khoảng $[\min(x_i, y_i) - I.\alpha, \max(x_i, y_i) + I.\alpha]$, trong đó $I = \max(x_i, y_i) - \min(x_i, y_i)$.

Tham số α thường được chọn là 0,5. Khi đó toán tử này còn được gọi là BLX-0.5.

Mô hình tiến hóa MGG (Minimal Generation Gap)

Mô hình MGG [3] được giới thiệu bởi Satoh năm 1996 và được sử dụng như một mô hình tạo sinh luân phiên. Thuật toán chi tiết được mô tả như sau:

B1. Khởi tạo quần thể: Tạo lập quần thể khởi tạo bởi các vectơ số thực một cách ngẫu nhiên.

B2. Chọn để tạo sinh: Chọn ngẫu nhiên một cặp cha mẹ từ quần thể.

B3. Tạo sinh: Sinh ra các cá thể mới sử dụng một toán tử lai ghép với cặp cha mẹ đã chọn.

B4. Chọn lọc để tái sinh: Chọn 1 cá thể tốt nhất từ tập các cá thể vừa tạo được cùng với cha mẹ chúng. Một cá thể khác được chọn theo cách dùng bánh xe Roulette. Hai cá thể này sẽ thay thế cặp cha mẹ đã chọn ban đầu.

B5. Lặp lại từ B2 đến B4 cho đến khi thoả điều kiện dừng.

4. CƠ CHẾ CHỌN LỰA THÍCH NGHI TOÁN TỬ LAI GHÉP

Trong phần này chúng tôi đề xuất một cơ chế lựa chọn thích nghi bằng cách chọn toán tử lai ghép thích hợp. Để tiện việc trình bày, chúng tôi xem xét trường hợp sử dụng hai toán tử lai ghép. Với các toán tử lai ghép đã được thiết kế trước, tại mỗi lần lặp, thuật toán sẽ quyết định sử dụng toán tử nào phụ thuộc vào tỷ lệ áp dụng thành công của toán tử đó. Ký hiệu LG1 và LG2 là 2 toán tử lai ghép, $P_{\text{LG1}}^{\text{apply}}$ và $P_{\text{LG2}}^{\text{apply}}$ là xác suất áp dụng LG1 hay LG2, các giá trị này được tính bởi:

$$P_{\text{LG1}}^{\text{apply}} = \frac{P_{\text{LG1}}^{\text{success}}}{P_{\text{LG1}}^{\text{success}} + P_{\text{LG2}}^{\text{success}}}, \qquad (*)$$

$$P_{\text{LG2}}^{\text{apply}} = 1 - P_{\text{LG1}}^{\text{apply}}, \qquad (**)$$

trong đó $P_{\text{LG1}}^{\text{success}}$ và $P_{\text{LG2}}^{\text{success}}$ là tỷ lệ để mỗi toán tử lai ghép tạo được con tốt hơn cha mẹ trong lần áp dụng cuối cùng của LG1 hay LG2 tương ứng.

Thuật toán được mô tả như sau:

Input: Quần thể có m cá thể được khởi tạo ngẫu nhiên.

Output: Cá thể có độ thích nghi (theo giá trị hàm mục tiêu F) cao nhất.

Algorithm:

B1. Tạo sinh quần thể ban đầu

Tạo ngẫu nhiên m vectơ thực (mỗi vectơ ứng với một cá thể).

Khởi tạo $P_{\text{LG1}}^{\text{apply}}$ bởi người sử dụng. Đặt $P_{\text{LG2}}^{\text{apply}} = 1 - P_{\text{LG1}}^{\text{apply}}$.

B2. Khởi tạo các tham số

Các tham số N_1, N_2, TC_1 và TC_2 khởi tạo bằng 0. (N_1, N_2 đếm số lần thực hiện toán tử LG1 hay LG2. TC_1 và TC_2 đếm số lần LG1 hay LG2 tạo được con tốt hơn cha mẹ).

B3. Tạo sinh

1. Chọn cá thể cha mẹ:

Chọn 1 cặp cá thể Parent1 và Parent2 ngẫu nhiên từ quần thể hiện tại.

2. Sinh con:

+ Chọn LG1 hay LG2 theo xác suất $P_{\text{LG1}}^{\text{apply}}$ và $P_{\text{LG2}}^{\text{apply}}$. Ký hiệu toán tử được chọn là LG.

+ Áp dụng toán tử LG trên Parent1 và Parent2 tạo nên hai con C1 và C2.

+ $N_i = N_i + 1$ (với N_i tương ứng LG).

3. Chọn cá thể tồn tại:

Chọn 2 cá thể từ quần thể chứa cả cha mẹ và các con của chúng như sau: cá thể thứ nhất là cá thể có độ thích nghi cao nhất, cá thể thứ hai được chọn từ các cá thể còn lại sử dụng bánh xe Roulette.

Thay thế hai cha mẹ bởi 2 cá thể vừa chọn. Nếu có con tốt hơn cả 2 cha mẹ thì $\mathrm{TC}_i = \mathrm{TC}_i + 1$ (TC_i tương ứng với LG).

B4. Tính lại xác suất chọn toán tử $P_{\mathrm{LG1}}^{\mathrm{apply}}, P_{\mathrm{LG2}}^{\mathrm{apply}}$, theo (*) và (**) trong đó

$$P_{\mathrm{LG1}}^{\mathrm{success}} = \mathrm{TC}_1/N_1; \quad P_{\mathrm{LG2}}^{\mathrm{success}} = \mathrm{TC}_2/N_2.$$

B5. Lặp lại từ B2 đến B4 khi điều kiện dừng chưa thỏa.

Cá thể có giá trị hàm mục tiêu tốt nhất trong quần thể cuối cùng là cá thể cần tìm.

5. KẾT QUẢ THỬ NGHIỆM

Các chương trình thử nghiệm được viết trên phần mềm Matlab. Các bài toán chọn để minh hoạ đều có số chiều $n = 30$; kích cỡ quần thể là $m = 50$. Mỗi quần thể là một ma trận thực gồm 50 dòng, 30 cột. Mỗi cá thể được biểu diễn là một vectơ dòng. Quần thể khởi tạo là một ma trận cấp 50×30 được sinh ngẫu nhiên. Để tiện khảo sát, chỉ có một loại toán tử lai ghép được sử dụng (không dùng toán tử đột biến).

Các bảng sau nêu kết quả thu được sau 5 lần chạy độc lập (mỗi dòng phản ánh kết quả 1 lần chạy), mỗi lần chạy thực hiện lặp 1000 lần.

Cột 2 cho giá trị hàm mục tiêu của cá thể tốt nhất trong quần thể khởi tạo. Các cột 3 và 4 cho giá trị hàm mục tiêu của cá thể tốt nhất trong quần thể cuối cùng thu được sau khi lặp sử dụng một loại toán tử lai ghép tương ứng. Cột cuối cùng cho giá trị hàm mục tiêu của cá thể tốt nhất khi sử dụng cơ chế chọn thích nghi hai toán tử lai ghép theo thuật toán đã nêu trên. Các giá trị được làm tròn đến 5 chữ số lẻ.

5.1. Cực tiểu hàm Rastrigin

$$\min f(x_1, x_2, ..., x_n) = 10.n + \sum_{i=1}^{n}(x_i^2 - 10.\cos(2.\pi.x_i))$$

với $n = 30$; kích cỡ quần thể $m = 50$; $-5{,}12 \leqslant x_i \leqslant 5{,}12$ với $i = 1, ..., 30$.

Lần chạy	Cá thể tốt nhất khởi tạo	Cá thể tốt nhất khi lai BLX-0,5	Cá thể tốt nhất khi lai đơn giản	Cá thể tốt nhất theo cơ chế chọn thích nghi
1	244,86370	102,78037	121,00012	55,80548
2	261,03475	92,92947	153,39344	47,41601
3	267,70168	108,31135	102,99350	50,37644
4	291,86649	122,07861	108,33097	53,40808
5	280,47529	106,69162	133,78436	34,91191

5.2. Cực tiểu hàm

$$\min f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} i.(x_i^2 + 10.(1 - \cos(2.\pi.x_i)))$$

với $n = 30$; kích cỡ quần thể $m = 50$; $-2 \leqslant x_i \leqslant 2$ với $i = 1, ..., 30$.

Lần chạy	Cá thể tốt nhất khởi tạo	Cá thể tốt nhất khi lai BLX-0,5	Cá thể tốt nhất khi lai mặt nạ	Cá thể tốt nhất theo cơ chế chọn thích nghi
1	3924,78608	520,87936	578,16316	69,64405
2	3488,71266	419,63400	178,41935	93,49087
3	3325,51171	817,84562	467,60317	118,01287
4	3705,48158	473,12391	394,97221	168,45532
5	4030,72816	428,37519	535,78499	135,01246

5.3. Cực tiểu hàm Elipcoit

$$\min f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} i.x_i^2$$

với $n = 30$; kích cỡ quần thể $m = 50$; $-5 \leqslant x_i \leqslant 5$ với $i = 1, ..., 30$.

Lần chạy	Cá thể tốt nhất khởi tạo	Cá thể tốt nhất khi lai BLX-0,5	Cá thể tốt nhất khi lai mặt nạ	Cá thể tốt nhất theo cơ chế chọn thích nghi
1	2392,71041	415,87447	97,47571	53,25859
2	2367,29037	496,768754	105,57320	76,07677
3	2536,57566	352,12613	154,99772	77,93205
4	2195,43508	463,54587	78,11627	37,24473
5	2232,75074	451,21011	167,94205	87,47584

5.4. Cực tiểu hàm Schwefel

$$\min f(x_1, x_2, ..., x_n) = 418{,}9828 \times n + \sum_{i=1}^{n} (x_i \times \sin(\sqrt{|x_i|}))$$

với $n = 30$; kích cỡ quần thể $m = 50$; $-500 \leqslant x_i \leqslant 500$ với $i = 1, ..., 30$.

Lần chạy	Cá thể tốt nhất khởi tạo	Cá thể tốt nhất khi lai BLX-0,5	Cá thể tốt nhất khi lai mặt nạ	Cá thể tốt nhất theo cơ chế chọn thích nghi
1	10500,48774	2787,67522	2069,80611	270,14869
2	10285,84499	5715,07773	2075,52312	27,92845
3	9937,92722	7252,29452	2581,90905	92,30387
4	10815,51780	5161,55513	2482,93591	8,60257
5	10630,67775	6420,70504	1710,45171	63,64511

5.5. Cực tiểu hàm

$$\min f(x_1, x_2, ..., x_n) = -20 \times \exp(-0{,}2) \times \sqrt{\frac{1}{2}\sum_{i=1}^{n} x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e.$$

với $n = 30$; kích cỡ quần thể $m = 50$; $-32 \leqslant x_i \leqslant 32$ với $i = 1, ..., 30$.

Lần chạy	Cá thể tốt nhất khởi tạo	Cá thể tốt nhất khi lai BLX-0,5	Cá thể tốt nhất khi lai mặt nạ	Cá thể tốt nhất theo cơ chế chọn thích nghi
1	20,70876	10,14688	12,76219	8,87978
2	20,83165	10,69340	11,02621	8,37523
3	20,68829	11,20207	11,54916	8,46823
4	20,71883	10,11810	11,80349	8,27243
5	20,78156	9,56505	13,69364	7,68781

Căn cứ các kết quả thử nghiệm trên ta thấy rõ ràng cơ chế lựa chọn thích nghi toán tử lai ghép cho kết quả tốt hơn (xem cột 5 trong các bảng) việc sử dụng chỉ một loại toán tử với cùng số lần lặp (xem các cột 3, 4). Cá biệt có những lần thử cho kết quả rất tốt (hàm Schwefel). Ngoài ra nếu tăng số lần lặp đối với việc sử dụng một toán tử lai ghép sẽ có hiện tượng dừng sớm, không tìm được cá thể con có giá trị hàm mục tiêu tốt hơn thế hệ trước. Song nếu thực hiện cơ chế lựa chọn thích nghi như trên, khả năng tìm được cá thể tốt hơn được kéo dài hơn. Các tác giả cũng đã thử nghiệm theo hướng thực hiện hai toán tử lai ghép kế tiếp nhau trong quá trình lặp, song không thu được kết quả tốt như thuật toán nêu trên.

Tuy nhiên, việc chọn 2 toán tử nào để kết hợp với nhau chủ yếu dựa vào kinh nghiệm, chưa có cơ sở toán học chỉ rõ những loại toán tử nào kết hợp với nhau thì hiệu quả tốt. Khi thử nghiệm, nếu kết hợp ngẫu nhiên 2 toán tử nào đó sẽ có những trường hợp không tốt hơn việc chỉ dùng 1 toán tử.

6. KẾT LUẬN

Bài báo này đã giới thiệu và khảo sát một cơ chế chọn lựa thích nghi trong việc sử dụng toán tử lai ghép để giải các bài toán tối ưu số bằng thuật toán di truyền mã hóa số thực. Kết quả thực nghiệm đã chỉ ra thuật toán này cho kết quả tốt hơn cách làm truyền thống là chỉ sử dụng một loại toán tử lai ghép trong một bài toán. Ngoài ra trong GA không chỉ có một loại toán tử lai ghép được thực hiện mà còn sử dụng cả toán tử đột biến. Vấn đề kết hợp các toán tử lai ghép theo phương thức này với toán tử đột biến cũng cần được nghiên cứu nhằm tăng tốc độ tìm kiếm và kết quả tốt hơn.

TÀI LIỆU THAM KHẢO

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison - Wesley, 1989.

[2] Kalyanmoy Deb, Hans-Georg Beyer, *Shelf-Adaptive Genetic Algorithms with Simulated Binary Crossover*, 1999.

[3] Kalyanmoy Deb, Dhiraj Joshi, Ashish Anand, *Real-Coded Evolutionary Algorithms with Parent-Centric Recombination,* 2001.

[4] Ko-Hsin Liang, Xin Yao, Charles S. Newton, *Adapting Selff-adaptive Parameters in Evolutionary Algorithms,* 1999.

[5] Osamu Takahashi, Hajime Kita, Shigenobu Kobayashi, *A Real-Coded Genetic Algorithm using Distance Dependent Alternation Model for Complex Function Optimization*, 1999.

[6] Peter J. Angeline, *Adaptive and Self-Adaptive Evolutionary Computations*, 1996.

[7] Shengxiang Yang, *Adaptive Crossover in Genetic Algorithms Usinh Statistics Mechanism*, 2002.

[8] Ulrich Bodenhofer, *Genetic Algorithms: Theory and Applications*, Lecture Notes Third Edition-Winter 2003/2004.

[9] William M. Spears, *Adapting Crossover in a Genetic Algorithm*, 1991.

[10] Nguyễn Đình Thúc, *Lập trình tiến hoá*, NXB Giáo dục, 2001.

Nhận bài ngày 13 - 5 - 2005