Tạp chí Tin học và Điều khiển học, T. 16, S. 3 (2000), 16–22

ỨNG DỤNG THUẬT TOÁN TIẾN HÓA GIẢI BÀI TOÁN TỐI ƯU ĐA MỤC TIÊU CÓ RÀNG BUỘC

VŨ NGỌC PHÀN

Abstract. In the last few years, many attentions have been paid to multi-objective optimization problems. The reason of this phenomenon is the increasingly importance of optimization to many branches of industrialized society. On the other hand the development and confirmation of evolutionary algorithms have reached to a new stage. The target of the present paper is to show the possibility of solving multi-objective optimization problems by evolutionary algorithms.

1. KHÁI QUÁT

Tối ưu đa mục tiêu được nghiên cứu nhiều trong thập kỷ 70 và đã được coi là một phần không thể thiếu được trong lý thuyết tối ưu và ứng dụng [4,5]. Nhưng lúc bấy giờ việc thực hiện bài toán trên máy tính gặp nhiều khó khăn. Do dung lượng bộ nhớ và tốc độ các máy tính đã được cải thiện đáng kể trong những năm gần đây, bài toán tối ưu đa mục tiêu ngày càng được quan tâm trong nhiều lĩnh vực như năng lượng, giao thông vận tải, xây dựng, công nghiệp điện tử, kinh tế, dịch vụ v.v.. Thí dụ, trên một địa bàn nào đó, người ta muốn vừa tăng cường phát triển công nghiệp vừa đẩy mạnh dịch vụ du lịch. Một cơ sở sản xuất, với số vốn đầu tư cố định, vừa tăng cường áp dụng tự động hóa để nâng cao chất lượng sản phẩm, vừa thu hút nhiều lao động để giải quyết vấn đề công ăn việc làm cho xã hội. Trong xu thế hội nhập quốc tế, lợi ích của nhiều quốc gia cùng phải được coi trọng và phát triển như nhau. Sau một thời gian im lặng, tối ưu đa mục tiêu lại thu hút sự chú ý của xã hội nói chung và của giới chuyên môn nói riêng [2,6].

Nhiều cách giải bài toán tối ưu đa mục tiêu đã được nêu trong [6]. Như đã biết, thực ra bài toán tối ưu đa mục tiêu không có lời giải tối ưu theo nghĩa đen của từ này. Nói chung không thể cùng một lúc tất cả các mục tiêu đạt giá trị tối ưu, chưa kể chúng có thể đối kháng nhau, nghĩa là mục tiêu này càng tốt lên bao nhiêu thì mục tiêu kia càng xấu đi bấy nhiêu. Lời giải thỏa hiệp hay lời giải hiệu quả vẫn là ý tưởng phù hợp thực tế nhất. Một câu hỏi xuất hiện ở đây là, có bao nhiêu lời giải hiệu quả và làm thế nào tìm ra chúng? Thông thường các hàm mục tiêu là các hàm phi tuyến theo tham số quyết định nên câu hỏi trên hầu như không thể trả lời được. Với độ phức tạp của bài toán, các phương pháp gradient hoặc biến phân đều gặp những trở ngại lớn. Đây cũng là lý do làm cho bài toán tối ưu đa mục tiêu một thời gian dài không có ứng dụng.

Với sự ra đời của các thuật toán tiến hóa, cùng với việc nhìn nhận vấn đề một cách thực dụng hơn, bài toán tối ưu đa mục tiêu đã trở nên đầy hấp dẫn. Chúng ta chấp nhận với nhau những quan điểm cơ bản sau đây:

- Không cần chứng minh sự tồn tại cúa lời giải tối ưu, không cần giả thiết các hàm mục tiêu có khả vi hay không khả vi, thậm chí không cần biết không gian khảo sát là lồi hay lõm. Tóm lại ta không cần nhiều thông tin tiên nghiệm về các hàm mục tiêu. Những thiếu hụt thông tin này sẽ được khắc phục bởi thuật toán tiến hóa.
- Sử dụng thuật toán tiến hóa để tìm lời giải hiệu quả. Theo thuật toán, thế hệ sau sinh ra không thể xấu hơn thế hệ trước nên dù thế nào đi nữa vẫn tìm được lời giải cho bài toán. Thuật toán tiến hóa không kết thúc khi đã tìm được một lời giải mà sẽ tìm tiếp các lời giải khác nhờ quá trình lai ghép và đột biến.
- Không cần đưa ra một tiêu chuẩn để kết thúc quá trình tìm kiếm. Quá trình tìm kiếm sẽ dừng lại khi người có thẩm quyền quyết định đã chọn ra một lời giải trong số những lời giải hiệu quả đã tìm thấy.

Phần 2 sẽ mô tả thuật toán tiến hóa đã được cải tiến cho phù hợp với bài toán tối đa mục tiêu. Trong Phần 3, bài toán tối ưu đa mục tiêu sẽ được trình bày tóm lược và đưa ra những kỹ thuật cần thiết để có thể áp dụng thuật toán tiến hóa. Đặc biệt việc xử lý điều kiện ràng buộc được nghiên cứu kỹ và một dạng hàm phạt được đưa ra nhằm khắc phục nhược điểm của các dạng hàm phạt đã biết.

2. THUẬT TOÁN TIẾN HÓA

2.1. Các thuật toán tiến hóa

Thuật toán tiến hóa là khái niệm dùng để chỉ những thuật toán tìm kiếm và tối ưu hóa dựa trên nguyên lý tiến hóa tự nhiên. Xin kể ra đây một số thuật toán tiến hóa đã được công bố.

- Quy hoạch tiến hóa EP do D. B. Pogel đề xuất. Có thể diễn tả EP đơn giản như sau: Cho một lớp các phương pháp khả dĩ giải quyết được một hay nhiều phần của một vấn đề. Dựa vào quy luật tiến hóa, tìm một phương pháp liên hợp đủ khả năng giải quyết trọn vẹn vấn đề đó.

- Chiến lược tiến hóa do T. Baeck, F. H. Hofmeister và H. P. Schwefel đề xuất. Thuật toán này cho phép từ một số chiến lược ban đầu, tạo ra những chiến lược mới phù hợp với môi trường thực tế một cách tốt nhất.

- Thuật toán di truyền do D. E. Goldberg đề xuất, được L. Davis và Z. Michalevicz phát triển. Đây là một phương pháp tìm kiếm ngẫu nhiên mở rộng [1, 3]. Thuật toán này đã được sử dụng vào việc thiết kế bộ điều khiển tối ưu theo tiêu chuẩn H_∞ [7].

Các thuật toán trên hình thành dựa vào quan niệm cho rằng, quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu. Quan niệm này có thể xem như một tiên đề đúng, không chứng minh được, nhưng phù hợp với thực tế khách quan. Quá trình tiến hóa thể hiện tính tối ưu ở chỗ, thế hệ sau bao giờ cũng tốt hơn (phát triển hơn, hoàn thiện hơn) thế hệ trước. Tiến hóa tự nhiên được duy trì nhờ hai quá trình cơ bản: sinh sản và chọn lọc tự nhiên. Xuyên suốt sự tiến hóa tự nhiên, các thế hệ mới luôn luôn được sinh ra để bổ sung thay thế thế hệ cũ. Cá thể nào phát triển hơn, thích ứng hơn với môi trường sẽ tồn tại. Cá thể nào không thích ứng được với môi trường sẽ bị đào thải. Sự thay đổi môi trường là động lực thúc đẩy quá trình tiến hóa. Ngược lại, tiến hóa góp phần làm thay đổi môi trường.

Các thế hệ mới sinh ra trong quá trình tiến hóa nhờ sự lai ghép ở thế hệ bố mẹ. Một cá thể mới có thể mang những tính trạng của bố mẹ (di truyền), cũng có thể mang những tính trạng hoàn toàn mới (đột biến). Di truyền và đột biến là hai cơ chế có vai trò quan trọng như nhau trong tiến hóa, mặc dù hiện tượng đột biến xảy ra với xác suất nhỏ hơn nhiều so với hiện tượng di truyền. Các thuật toán tiến hóa tuy có những điểm khác nhau nhưng đều mô phỏng bốn quá trình cơ bản: lai ghép, đột biến, sinh sản và chọn lọc tự nhiên.

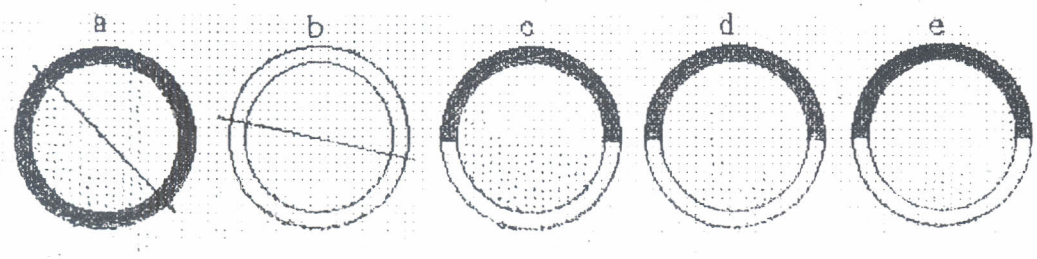
Quá trình lai ghép

Quá trình lai ghép là quá trình hình thành nhiễm sắc thể mới trên cơ sở các nhiễm sắc thể cha mẹ, bằng cách ghép một hay nhiều đoạn alen của hai nhiễm sắc thể cha mẹ với nhau. Quá trình lai ghép xảy ra với xác suất P_c, có thể mô phỏng như sau:

- Chọn ngẫu nhiên hai cá thể bất kỳ của quần thể. Giả sử gien di truyền của bố mẹ đều bao gồm m alen.
- Tạo một số ngẫu nhiên trong khoảng từ 1 đến $m-1$ (điểm phân chia chuỗi alen). Giả sử điểm đó chia chuỗi m alen thành hai nhóm m_1 và m_2. Hai chuỗi alen mới sẽ là $m_{11} + m_{22}$ và $m_{21} + m_{12}$.
- Đưa hai cá thể mới này vào quần thể để tham gia các quá trình tiến hóa tiếp theo.

Cách mô phỏng quá trình lai ghép trên đã được sử dụng trong [7]. Đối với bài toán tối ưu đa mục tiêu, cách lai ghép này cần phải được mở rộng. Thay vì chọn một cặp bố mẹ để lai ghép, ta sẽ chọn ra hai nhóm có cùng số cá thể như nhau. Cho các nhóm alen này khép kín thành vòng tròn.

Với mỗi vòng tròn ta tạo ra một nhát cắt ngẫu nhiên (hình 1.a, 1.b). Lấy nửa vòng tròn này ghép với nửa vòng tròn kia (hình 1.c, 1.d). Quá trình cắt ghép các vòng tròn alen có thể diễn ra một số lần. Cuối cùng từng vòng tròn lại được cắt một cách ngẫu nhiên thành những đoạn alen có độ dài ban đầu (hình 1.e).



Hình 1

Quá trình đột biến

Đột biến là hiện tượng cá thể con mang một hay nhiều tính trạng không có trong mã di truyền của bố mẹ. Quá trình đột biến xảy ra với xác suất P_m nhỏ hơn rất nhiều so với xác suất lai ghép P_c. Quá trình đột biến có thể mô phỏng như sau:

- Chọn ngẫu nhiên một cá thể bất kỳ của quần thể.
- Tạo một số ngẫu nhiên k trong khoảng từ 1 đến m, $1 \leq k \leq m$.
- Thay đổi alen thứ k và trả cá thể vào quần thể để tham gia quá trình tiến hóa tiếp theo.

Quá trình đột biến cũng có thể xảy ra ở nhiều vị trí của chuỗi alen.

Quá trình sinh sản và chọn lọc tự nhiên

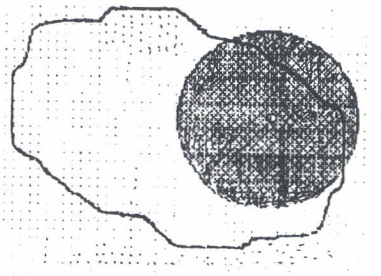
Quá trình sinh sản là quá trình trong đó các cá thể được sao chép trên cơ sở giá trị fitness của nó. Quá trình này có thể mô phỏng như sau:

- Tính giá trị fitness của từng cá thể, lập bảng cộng dồn các giá trị fitness (theo số thứ tự gán cho các cá thể). Giả sử quần thể có n cá thể. Ký hiệu giá trị fitness của cá thể thứ i là F_i, tổng dồn thứ i là F_{ti}, tổng các giá trị fitness trên cả quần thể là F_m.
- Tạo một số ngẫu nhiên F trong khoảng từ 0 đến F_m.
- Chọn cá thể thứ k đầu tiên thỏa mãn điều kiện $K \geq F_{tk}$ đưa vào thế hệ mới của quần thể. Cũng có thể chọn tất cả các cá thể có F_k bằng phần nguyên của $F(F_k/F_m)$.

2.2. Rút gọn miền tìm kiếm

Khi quá trình tìm kiếm đang hội tụ dần đến một điểm tối ưu cục bộ, quá trình đột biến có thể sinh ra các cá thể cho giá trị fitness nhỏ hơn so với cá thể hiện tại trong nhiều thế hệ liên tiếp, đôi khi mất hẳn khả năng trở lại điểm tối ưu cục bộ đã sắp đạt đến ở các thế hệ trước. Để khắc phục tình trạng này, ta sử dụng phương pháp rút gọn miền tìm kiếm. Khi thấy quá trình tìm kiếm có xu thế dần đến điểm tối ưu cục bộ (giá trị fitness tăng chậm), quá trình tiến hóa sẽ chỉ xảy ra với quần thể gồm những cá thể lân cận (xem hình 2).

Trong hình 2, giả sử điểm đen là điểm tối ưu cục bộ, điểm trắng là điểm hiện thời của quá trình tìm kiếm. Quá trình tiến hóa khi đó chỉ cho diễn ra với các cá thể nằm trong đường tròn tâm là điểm trắng và bán kính là r. Khi đã đạt đến điểm tối ưu cục bộ (độ tăng giá trị fitness nhỏ hơn một số ε cho trước), vòng tròn hạn chế sẽ được phá vỡ. Quá trình tiến hóa lại được tiến hành trên toàn cục.



Hình 2

3. TỐI ƯU ĐA MỤC TIÊU

3.1. Mô tả bài toán

Gọi $x \in \Omega$ là biến quyết định, Ω là không gian Oclit m chiều. Cho n hàm mục tiêu $f_1(x), f_2(x), ..., f_n(x)$, là các hàm phi tuyến phụ thuộc x. Cho k ràng buộc $g_i(x) \leq a_i$ $(i = 1, 2, ..., k)$. Bài toán chúng ta quan tâm có dạng sau:

$$\begin{aligned} &\max_{x \in \Omega} f(x) \\ &\text{với } g_i(x) \leq a_i \,. \end{aligned} \tag{1}$$

Ở đây $f(x) = (f_1(x), f_2(x), ..., f_n(x))$ là hàm vectơ n chiều. Như trên đã nói, ta chỉ quan tâm đến lời giải hiệu quả của (3.1). Chúng ta nhắc lại những khái niệm sau đây:

Định nghĩa 1. Một vectơ giá trị $u = (u_1, u_2, ..., u_n)$ được gọi là tốt hơn vectơ $v = (v_1, v_2, ..., v_n)$ khi và chỉ khi

$$\forall i \in \{1, 2, ..., n\} : v_i \leq u_i \text{ và } \exists i \in \{1, 2, ..., n\} : v_i < u_i \,.$$

Định nghĩa 2. Một lời giải $x_v \in \Omega$ được gọi là lời giải tối ưu của bài toán (1) khi và chỉ khi không tồn tại một $x_u \in \Omega$ sao cho $u = f(x_u) = (u_1, u_2, ..., u_n)$ tốt hơn $v = f(x_v) = (v_1, v_2, ..., v_n)$.

Như đã nói ở phần trên, các Định nghĩa 1 và 2 làm cho bài toán tối ưu đa mục tiêu trở nên quá chặt chẽ và khó áp dụng. Việc chứng minh không tồn tại một $x_u \in \Omega$ theo Định nghĩ2 cho đa số các trường hợp thực tế là một việc khó khăn. Thay vì Định nghĩa 2, chúng ta đưa ra một định nghĩa mềm dẻo hơn. Ta đã biết, sự tiến hóa tự nhiên là một quá trình liên tục, thế hệ sau tốt hơn thế hệ trước. Không ai đặt câu hỏi bao giờ quá trình tiến hóa kết thúc và thế hệ cuối cùng là thế nào. Rõ ràng trong thực tế ta phải sống và làm việc với thế hệ con cháu, mặc dù biết rằng thế hệ con cháu của con cháu sẽ phát triển hơn.

Định nghĩa 3. Giả sử với $x_v \in \Omega$ ta có $v = f(x_v) = (v_1, v_2, ..., v_n)$. Lời giả $x_u \in \Omega$ gọi là lời giải tốt hơn x_v nếu $u = f(x_u) = (u_1, u_2, ..., u_n)$ tốt hơn v.

Dựa vào Định nghĩa 3, dùng thuật toán tiến hóa để tìm một lời giải tốt hơn lời giải đã có. Ta không chứng minh sự tồn tại của lời giải tối ưu và không quan tâm nhiều đến việc ta có tìm được nó hay không. Tuy nhiên, có thể có nhiều lời giải tốt như nhau. Điều này làm cho người quyết định khó xử khi phải chọn ra một lời giải để áp dụng. Để thoát khỏi tình trạng này, chúng ta đưa ra nguyên tắc **top N.**

Nguyên tắc top N. Trong quần thể gồm m cá thể, ta chọn ra N cá thể có fitness cao nhất. Từ N cá thể này, người có thẩm quyền (người ra bài toán) sẽ tự chọn lấy một cá thể tốt nhất theo chủ quan của mình.

Như vậy, tối ưu đa mục tiêu gắn với quyết định. Như đã biết, quá trình quyết định nào cũng kèm theo rủi ro. Người quyết định chứ không phải người làm tối ưu hóa gánh chịu rủi ro đó. Đó chính là qui luật khách quan của nền kinh tế thị trường đang là xu thế tiến hóa của thế giới ngày nay.

Người ra bài toán có thể chọn một cá thể từ N cá thể có fitness cao nhất theo ba cách như trình bày sau đây.

- Cho trước mỗi mục tiêu một trọng số (đưa bài toán đa mục tiêu về bài toán một mục tiêu). Chọn cá thể có giá trị fitness cao nhất trong N ứng cử viên tìm được theo hàm mục tiêu mới (Priori Articulation of Preferences).
- Chọn một lời giải trong N lời giải theo kinh nghiệm chuyên gia và những chủ định riêng mà người giải bài toán tối ưu không biết trước (Posteriori Articulation of Preferences).
- Người quyết định và người giải bài toán luôn luôn trao đổi với nhau. Người giải bài toán dựa vào những thông tin này để tác động vào quá trình tiến hóa nhằm tạo ra những cá thể đúng yêu cầu của người ra quyết định (Progressive Articulation of Preferences).

3.2. Xử lý điều kiện ràng buộc

Những lời giải tìm được theo Định nghĩa 3 được gọi là lời giải khả thi (feasible solution) nếu nó thỏa mãn điều kiện ràng buộc, gọi là lời giải không khả thi (infeasible solution) nếu nó không thỏa mãn điều kiện ràng buộc. Để giải quyết bài toán tối ưu đa mục tiêu có rằng buộc, người ta thường sử dụng các phương pháp sau:

- Hạn chế x trong một miền con $\Omega^+ \subset \Omega$ sao cho với $x \in \Omega^+$ các điều kiện ràng buộc mặc nhiên được thỏa mãn. Cách này rất đơn giản và tiện lợi nhưng chỉ thực hiện được trong trường hợp miền Ω^+ dễ xác định, chẳng hạn khi x bị chặn bởi cận trên và cận dưới. Trong biểu thức (1), nếu $g_i(x)$ là các hàm phi tuyến thì việc tìm Ω^+ nói chung không dễ dàng.
- Trong quá trình tiến hóa, loại bỏ ngay một cá thể $x \in \Omega$ nào đó vừa sinh ra khi nó không thỏa mãn điều kiện ràng buộc trước khi thử xem nó có phải là lời giải tốt hơn không. Cách này là cách tốt nhất đảm bảo điều kiện ràng buộc luôn được thỏa mãn. Tuy nhiên nó làm cho khả năng tiến hóa bị giảm đi. Bởi vì, cũng như trong thế giới tự nhiên, một cá thể lúc này không phù hợp với môi trường có thể lại phù hợp rất tốt khi môi trường thay đổi. Hơn nữa, bố mẹ thành đạt chắc gì con cái cũng thành đạt và ngược lại. Một cá thể vi phạm điều kiện ràng buộc, lai ghép với một cá thể khác có thể sinh ra một cá thể vừa thỏa mãn điều kiện ràng buộc vừa đạt tính tối ưu.
- Xấp xỉ lời giải không khả thi bằng một lời giải khả thi ở lân cận gần nhất. Về mặt logic, phương pháp này xem ra có lý. Nó tránh được hiện tượng khủng hoảng, không tìm được lời giải. Trong trường hợp vì một lý do ngẫu nhiên nào đó, các cá thể mới sinh ta đều không thỏa mãn điều kiện ràng buộc thì vẫn tìm được một cá thể khả thi. Nhưng khi đi tìm lân cận tốt nhất ta lại phải giải một bài toán tối ưu khác, chưa chắc đã đơn giản hơn bài toán ban đầu.
- Đưa vào hàm mục tiêu một hàm phụ trợ gọi là hàm phạt. Hàm phạt được xây dựng sao cho giá trị của nó tương ứng với mức độ vi phạm điều kiện ràng buộc. Cách này có khả năng khắc phục nhược điểm của cách 2. Khi một cá thể mới sinh ra không thỏa mãn điều kiện ràng buộc, ta không loại bỏ nó ngay mà chỉ "phạt" nó, vẫn cho nó một cơ hội tham gia quá trình tiến hóa. Để hạn chế ảnh hưởng của nó đến quá trình tiến hóa, hàm phạt giảm giá trị fitness của cá thể này. Nếu nó sinh ra các thế hệ con cháu tốt hơn thì thế hệ con cháu của nó sẽ tồn tại. Còn bản thân nó sau một thời gian sẽ bị quá trình chọn lọc tự nhiên đào thải.

3.3. Phương pháp xây dựng hàm phạt

Như trên vừa nói, mục đích đưa ra hàm phạt là làm thay đổi giá trị fitness của các cá thể vi phạm điều kiện ràng buộc. Giả sử ta chọn hàm fitness là $\lambda(x)$. Hàm phạt sẽ đưa ra là $\xi(x)$. Hàm fitness bây giờ sẽ có dạng

$$\varphi(x) = \lambda(x) + \xi(x).\mu. \tag{2}$$

Trong biểu thức (2), $\mu = 0$ khi x thỏa mãn các điều kiện ràng buộc, $\mu = 1$ khi x không thỏa mãn các điều kiện ràng buộc. Hàm $\xi(x)$ là một hàm tỷ lệ với mức độ vi phạm điều kiện ràng buộc. Việc xây dựng hàm $\xi(x)$ là một việc quan trọng cũng tương tự như việc chọn hình phạt trong đời sống xã hội. Nếu hình phạt quá nhẹ, các cá thể vi phạm có thể chèn ép các cá thể khác trong quá trình tiến hóa. Nếu hình phạt quá nặng, cơ may "làm lại cuộc đời" đối với cá thể này quá mong manh. Dưới đây là một số cách xây dựng hàm phạt đã được nêu trong [6]:

- $\xi(x) = (v/2)^\sigma$, trong đó v là số các điều kiện bị vi phạm, $v \in \{1, 2, \ldots, k\}$, σ là hệ số nghiêm khắc (severity factor).
- $\xi(x) = \delta(\tau)\phi(x) = \delta(x)[\phi_0 + \tau j]$, trong đó $\delta(x)$ là độ đo mức độ vi phạm, $\phi(\tau)$ là hệ số nghiêm khắc, ϕ_0 là giá trị ban đầu, τ chỉ số của thế hệ tiến hóa, j số bước tìm kiếm đã thực hiện.
- $\xi(x) = (C\tau)^\alpha \delta^\beta(x)$, trong đó C là hằng số, τ chỉ số của thế hệ tiến hóa, $\delta(x)$ là độ đo mức độ vi phạm, α và β là các hệ số thể hiện tính nghiêm khắc.
- $\xi(x) = \lambda(x)\tau^\gamma$, trong đó τ là chỉ số của thế hệ tiến hóa, γ là một số dương nằm trong khoảng $[0{,}5, 1]$.
- $\xi(x) = \alpha(\tau)\delta(x) + \beta(\tau)$, trong đó $\alpha(\tau)$ và $\beta(\tau)$ là các hàm đơn điệu tăng, τ là chỉ số của thế hệ tiến hóa.

Cách xây dựng hàm phạt đầu tiên đơn giản nhưng chưa thực sự là độ đo mức vi phạm điều kiện ràng buộc. Thí dụ, một cá thể chỉ vi phạm một điều kiện ràng buộc nhưng rất nặng, nghĩa là khoảng cách của nó tới Ω^+ rất lớn. Trong khi đó một cá thể khác vi phạm nhiều điều kiện nhưng khoảng cách của nó tới Ω^+ lại rất nhỏ. Cách xây dựng hàm phạt từ thứ hai đến thứ năm có sử dụng chỉ số của thế hệ tiến hóa. Giá trị hàm phạt tăng dần theo các thế hệ tiến hóa làm cho những cá thể vi phạm điều kiện ràng buộc bị loại bỏ nhanh chóng hơn. Tuy nhiên, các cách xây dựng hàm phạt này không sử dụng một thông tin quan trọng. Đó chính là số lượng các cá thể vi phạm điều kiện ràng buộc tại một thời điểm xác định của quá trình tiến hóa. Rõ ràng, nếu trong quần thể có quá nhiều cá thể vi phạm điều kiện ràng buộc thì sẽ có nguy cơ quá trình tiến hóa bị khủng hoảng. Nghĩa là số lượng các cá thể vi phạm điều kiện ràng buộc tiếp tục tăng lên, nhiều bước tiến hóa không sinh ra được các cá thể tốt, quá trình tìm kiếm không hội tụ. Để khắc phục những thiếu sót vừa phân tích trên đây, chúng tôi đề xuất cách xây dựng hàm phạt như sau.

$$\xi(x) = \gamma.e^{\frac{\tau\eta}{\varepsilon}} \sum_{i=1}^{k} \Phi(g_i(x) - a_i). \tag{3}$$

Trong biểu thức (3), γ là một số dương, τ là chỉ số của thế hệ tiến hóa, η là số lượng cá thể vi phạm điều kiện ràng buộc ở thế hệ thứ τ, ε là hệ số nâng đỡ, $\phi(t) = 0$ nếu $t \leq 0$ và $\phi(t) = t$ nếu $t > 0$. Giá trị hàm phạt trong biểu thức (3) tăng theo hàm mũ của số cá thể vi phạm điều kiện ràng buộc và sự kéo dài qua các thế hệ của nó. Hàm $\phi(.)$ trong biểu thức (3) cho phép quan tâm tới các điều kiện ràng buộc bị vi phạm như thế nào và bản thân nó chứa đựng thông tin có bao nhiêu điều kiện bị vi phạm.

3.4. Kết quả mô phỏng

Cho các hàm mục tiêu diễn tả bởi các công thức (4) và (5). Điều kiện ràng buộc cho bởi công thức (6). Chọn $\gamma = 1{,}15$ và $\varepsilon = 1{,}95$

$$f_1(x, y) = 0{,}5 - \frac{\sin^2\sqrt{x^2 + y^2} - 0{,}5}{\left(1 + 0{,}01(x^2 + y^2)\right)^2}, \tag{4}$$

$$f_2(x, y) = 1 - (x - 0{,}3)^2 - (y - 0{,}3)^2, \tag{5}$$

$$g(x, y) = x + y - 0{,}25 \leq 0. \tag{6}$$

Sau 14 bước tiến hóa, chọn được 3 cá thể có độ đo fitness lớn nhất là

$$(x, y)_1 = (0,103, 0,115),$$
$$(x, y)_2 = (0,192, 0,057),$$
$$(x, y)_3 = (0,061, 0,183).$$

Để đạt được kết quả tương tự, đối với các dạng hàm phạt khác phải cần nhiều hơn 25 bước tiến hóa.

4. KẾT LUẬN

Tối ưu đa mục tiêu là bài toán có tầm quan trọng to lớn đối với nhiều lĩnh vực khác nhau. Tuy nhiên cần thấy rằng, giải bài toán tối ưu đa mục tiêu không phải là công việc của riêng người lập trình. Một phần cơ bản phải do chính người đặt bài toán giải quyết. Đó là việc chọn ra một lời giải từ một lớp các lời giải khả thi. Việc đưa ra khái niệm tối ưu mềm dẻo theo gương của quá trình tiến hóa tự nhiên có thể làm cho bài toán tối ưu đa mục tiêu đỡ phức tạp và có ý nghĩa thực tế hơn.

Thuật toán tiến hóa không chỉ làm thay đổi quan niệm về tính tối ưu mà còn là một công cụ thích hợp để giải quyết bài toán tối ưu đa mục tiêu. Nhờ thuật toán này việc xử lý các điều kiện ràng buộc trở nên đơn giản hơn. Trong bài này, tác giả đã đưa ra được một dạng hàm phạt để làm giảm giá trị fitness của lời giải không khả thi. Dạng hàm phạt mới này khắc phục nhược điểm của các dạng hàm phạt trước đây đã không tận dụng thông tin về số lượng các cá thể vi phạm điều kiện ràng buộc. Kết quả mô phỏng cho thấy, nhờ tận dụng thông tin này quá trình tìm kiếm đã hội tụ nhanh hơn.

TÀI LIỆU THAM KHẢO

[1] Chin-Teng Lin, C. S. George Lee, *Neural Fuzzy Sytems*, Prentice-Hall International, Inc., 1996.

[2] C. M. Fonseca, P. Flemming, Multi-objective optimization and multiple constraint handling with evolutionary algorithms, Part I: a unified formulation, *IEEE Trans. On System, Man and Cybernetics* **38** (1) (1998) 26-47.

[3] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Reading, MA, Addition-Wesley.

[4] Ignizio J. P., *Linear Programming in Single- and Multiple-Objective Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[5] Rao S. S., *Optimization: Theory and Applications*, Wiley Eastern Ltd. New Dehli, 1977.

[6] V. Petridis, S. Kazarlis, A. Bakirtzis, Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problem, *IEEE Trans. On System, Man and Cybernetics* **28** (5) (1998).

[7] Vũ Ngọc Phàn, H_∞-optimal Controller design using genetic algorithms, *Tạp chí Tin học và Điều khiển học* **15** (3) (1999).

Nhận bài ngày 8 - 10 - 1999
Nhận lại sau khi sửa ngày 14 - 2 - 2000

Viện Công nghệ thông tin