Tạp chí Tin học và Điều khiển học, T. 17, S. 3 (2001), 1–14

CÁC THUẬT TOÁN TIẾN HÓA VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN TỰ ĐỘNG

VŨ NGỌC PHÀN

Abstract. Evolutionary algorithms are of great attentions in the last decade. They are not only powerful search techniques for solving many conventional optimization problems but also being utilized in different artificial intelligent systems. The present paper intends to clarify the basic of evolutionary algorithms and their applicability to the issues of automatic control. In Section 2 the essence of general optimization problems is expressed. The fundamental of the theory of the natural evolution and that of the genetics will be shortly given in the third section. A combination of Section 2 and Section 3 indicates why and how the evolutionary algorithms have been developed. The ground stones of evolutionary algorithms, namely the parameter coding, the fitness and fitness scaling, the simulation of the reproduction and selection processes, the simulation of the crossover and the mutation, are clarified in Section 4 to 8. In the 9th section the handling of the equality and inequality constraints by assigning the fitness value based on the penalty function is discussed. The convergence of evolutionary algorithms can be improved by a search interval reduction as shown in Section 10. The 11^{th} section is dealing with the starting step of an evolutionary algorithm, i.e. the creation of the initial population. The applicability of the evolutionary algorithms on the field of automatic control will be described in Section 12.

Tóm tắt. Bài báo nhắc lại nội dung cơ bản của bài toán tối ưu hóa, trình bày vài nét sơ đẳng nhất về tế bào học và học thuyết tiến hóa tự nhiên. Tiếp theo, bài báo đề cập đến những nội dung như: mã hóa tham số tìm kiếm, xác định độ đo fit-nit, mô phỏng quá trình sinh sản, mô phỏng quá trình lai ghép và đột biến. Sau đó là cách xử lý điều kiện ràng buộc đẳng thức và bất đẳng thức của bài toán tối ưu bằng phương pháp hàm phạt. Cuối cùng, bài báo đề cập đến vấn đề tạo lập quần thể ban đầu và rút gọn miền tìm kiếm trong quá trình tiến hóa, vài nét về khả năng ứng dụng của các thuật toán tiến hóa.

1. MỞ ĐẦU

Trong nhiều năm lại đây, các thuật toán tiến hóa đã được phát triển và ứng dụng rất rộng rãi trong nhiều lĩnh vực mà ở đó tối ưu luôn là trung tâm của sự chú ý. Về mặt lý thuyết, các thuật toán tiến hóa không chứa đựng những khó khăn toán học lớn, chúng mang nặng tính heuristic. Hiệu quả của một thuật toán tiến hóa phụ thuộc nhiều vào bài toán cụ thể và kinh nghiệm của người giải quyết. Thực chất, các thuật toán tiến hóa là các thuật toán tìm kiếm ngẫu nhiên. Ưu điểm nổi bật của các thuật toán tiến hóa so với các thuật toán tìm kiếm thông thường là ở chỗ, nó cho phép giải các bài toán tối ưu khi hàm mục tiêu không thể diễn tả một cách tường minh và tham số tìm kiếm mang nhiều bản chất tự nhiên khác nhau. Để làm thí dụ hãy xét trò chơi trí tuệ thường được đưa vào các chương trình như *Đường lên đỉnh Ô-lim-pi-a* dành cho học sinh phổ thông. Cho hai nhóm đồ vật. Nhóm thứ nhất gồm máy tính, con dao và quả cầu lông. Nhóm thứ hai gồm bông hoa tươi, hòn đá, con ong chết và một ít rễ tơ hồng. Giả sử bây giờ có người đưa ra một lẵng hoa và yêu cầu hãy xếp nó vào nhóm nào cho hợp lý nhất (tối ưu nhất). Lời giải tối ưu nhất ở đây là xếp lẵng hoa vào nhóm thứ nhất: nhóm các đồ vật nhân tạo, vì nhóm thứ hai là nhóm các đồ vật tự nhiên. Trò chơi này cho đến nay chỉ có thể do con người tự suy luận và đưa ra lời giải. Nhờ các thuật toán tiến hóa, vấn đề này máy tính có thể giải bằng cách gán cho mỗi đồ vật các giá trị fit-nit khác nhau theo các góc độ khác nhau. Xác định sự chênh lệch giá trị fit-nit của cá thể mới (lẵng hoa) so với hai quần thể đã có. Cá thể mới sẽ được xếp vào quần thể với sự chênh lệch giá trị fit-nit nhỏ hơn.

Phần 2 của bài báo nhắc lại nội dung cơ bản của bài toán tối ưu hóa. Vài nét sơ đẳng nhất về

CV 14

tế bào học và học thuyết tiến hóa tự nhiên sẽ được nêu trong Phần 3 để làm rõ cơ sở lý luận của các thuật toán tiến hóa. Các phần tiếp theo đề cập đến những nội dung như: mã hóa tham số tìm kiếm, xác định độ đo fit-nit, mô phỏng quá trình sinh sản, mô phỏng quá trình lai ghép và đột biến. Phần 9 sẽ trình bày cách xử lý điều kiện ràng buộc đẳng thức và bất đẳng thức của bài toán tối ưu bằng phương pháp hàm phạt. Để tìm hiểu về tính hội tụ của các thuật toán tiến hóa, các phần 10 và 11 sẽ đề cập đến vấn đề tạo lập quần thể ban đầu và rút gọn miền tìm kiếm trong quá trình tiến hóa. Vài nét về khả năng ứng dụng của các thuật toán tiến hóa sẽ được trình bày trong Phần 12.

2. VẤN ĐỀ TỐI ƯU HÓA VÀ CÁC THUẬT TOÁN TÌM KIẾM

Khái niệm tối ưu hóa được dùng để chỉ quá trình nhận ra lời giải tốt nhất theo một qui ước nào đó. Để làm ví dụ, chúng ta hãy xét vấn đề điều khiển tối ưu. Giả sử một đối tượng điều khiển được mô tả bởi phương trình vi phân dạng

$$\dot{x} = \varphi(x, u, t), \quad x(t = 0) = x(0), \tag{2.1}$$

$$y = \psi(x, u, t), \tag{2.2}$$

trong đó $x(t)$ là vectơ trạng thái n chiều, $u(t)$ là vectơ điều khiển p chiều, $y(t)$ là vectơ đầu ra q chiều, t là biến thời gian, φ và ψ là các hàm vectơ có số chiều tương ứng. Mục tiêu của bài toán điều khiển tối ưu là tìm một chiến lược điều khiển $u(t)$ sao cho phiếm hàm

$$J = \frac{1}{T}\int_0^T (x^T R x + u^T Q u)dt \tag{23}$$

đạt giá trị nhỏ nhất.

Trong kỹ thuật và công nghệ chúng ta thường gặp rất nhiều vấn đề mà lời giải của nó là làm sao để cho một hay nhiều mục tiêu đạt giá trị cực đại hoặc cực tiểu. Về mặt toán học, vấn đề tối ưu hóa (cực đại hóa hoặc cực tiểu hóa) có thể diễn tả tổng quát như sau. Tìm $x = (x_1, x_2, \ldots, x_n)$ sao cho x tối ưu hóa $f(x)$ với các ràng buộc

$$g_i(x) \leq 0, \quad i = 1, 2, \ldots, k, \tag{2.4}$$

$$h_j(x) = 0, \quad j = 1, 2, \ldots, m, \tag{2.5}$$

trong đó x là vectơ tham số n chiều, $f(x)$ là hàm mục tiêu, $g_i(x)$ và $h_j(x)$ là các ràng buộc dạng bất đẳng thức và đẳng thức. Nếu $k = 0$ và $m = 0$, bài toán tối ưu là bài toán không có ràng buộc. Nếu $k \neq 0$ và/hoặc $m \neq 0$ thì vấn đề trên được gọi là bài toán tối ưu có ràng buộc. Dựa vào các đặc điểm của vectơ tham số cũng như của hàm mục tiêu, người ta đặt cho vấn đề tối ưu hóa những tên gọi khác nhau như: qui hoạch tuyến tính (linear programming), qui hoạch phi tuyến (non-linear programming), qui hoạch nguyên (integer programming), qui hoạch lồi (convex programming), qui hoạch lõm (concave programming), qui hoạch hình học (geometric programming), qui hoạch ngẫu nhiên (stochastic programming), qui hoạch mờ (fuzzy programming), qui hoạch động (dynamic programming)... Trong lịch sử khoa học, bài toán tối ưu đã được nghiên cứu từ thời Newton, Lagrange và Cauchy. Từ sau đại chiến thế giới thứ hai, bài toán tối ưu hóa thực sự trở thành mối quan tâm không chỉ của các nhà khoa học kỹ thuật mà của mọi lĩnh vực đời sống xã hội. Với tính chất đa dạng và phức tạp nhưng mang lại hiệu quả rất cao, nhiều phương pháp giải quyết vấn đề tối ưu hóa đã được đề xuất và không ngừng phát triển. Nhìn chung, chúng ta có thể chia các phương pháp này thành hai nhóm cơ bản. Nhóm thứ nhất bao gồm tất cả các phương pháp giải tích, còn được gọi là các phương pháp kinh điển (classical methods). Trong thời đại máy tính điện tử, các phương pháp này ít được ứng dụng thực tế, nhưng luôn được đưa vào giáo trình giảng dạy ở các trường đại học vì tính toán học chính xác của nó. Nhóm thứ hai bao gồm các phương pháp tìm kiếm (search methods). Trong nhóm này chúng ta có thể nhắc đến một số phương pháp như phương pháp tìm kiếm trực tiếp (direct search method), phương pháp tìm kiếm ngẫu nhiên (random search method), phương pháp quay tọa độ của Rosenbrock (method of rotating coordinates), phương pháp đơn hình (simplex method) để giải các bài toán không có ràng buộc, và các phương pháp như

phương pháp mặt cắt (cutting plane method), phương pháp hàm phạt nội (interior penalty function method), phương pháp hàm phạt ngoại (exterior penalty function method), phương pháp đa hình (complex method) để giải các bài toán có ràng buộc. Mặc dù các phương pháp tìm kiếm đã được cải tiến và đã góp phần giải quyết được rất nhiêu bài toán tối ưu hóa trong thực tế, song một số khó khăn mang tính nguyên tắc vẫn tồn tại.

Trước hết, là khó khăn liên quan đến điểm xuất phát của quá trình tìm kiếm (starting point). Nếu chọn điểm xuất phát không thích hợp thì quá trình hội tụ sẽ rất chậm, thậm chí không tìm được lời giải mong muốn. Hơn nữa, hầu hết các phương pháp đòi hỏi điểm xuất phát phải là một lời giải thỏa đáng (feasible solution) của bài toán tối ưu. Trên thực tế, việc tìm một lời giải thỏa đáng ban đầu (initial feasible solution) để làm điểm xuất phát khó khăn không kém gì chính bài toán đó.

Tồn tại thứ hai ở các thuật toán tìm kiếm thông thường là câu hỏi về tính toàn cục của lời giải. Khi quá trình tìm kiếm đã dừng lại ở một điểm tối ưu, không có thông tin nào cho biết liệu điểm này có phải là điểm tối ưu toàn cục (global optimum) hay chỉ là điểm tối ưu cục bộ (local optimum). Nếu nghi rằng đó chỉ là điểm tối ưu cục bộ thì cũng chẳng có con đường nào vượt ra khỏi điểm đó để có cơ may đi đến một điểm tốt hơn.

Rất may là các thuật toán tiến hóa (evolutionary algorithms), một công cụ tìm kiếm vạn năng, đã ra đời và khắc phục được những thiếu sót của các thuật toán tìm kiếm trước nó. Thuật toán tiến hóa không bắt đầu quá trình tìm kiếm từ một điểm xuất phát duy nhất. Trái lại, nó bắt đầu quá trình tìm kiếm từ một tập các điểm xuất phát, gọi là quần thể ban đầu (initial population), trong đó không nhất thiết mọi cá thể (individual) đều phải là một lời giải thỏa đáng. Hơn thế nữa, quá trình tìm kiếm phỏng theo quá trình tiến hóa (evolution process) cho phép thoát ra khỏi các điểm tối ưu cục bộ. Nói cách khác, quá trình tiến hóa có thể tiếp tục không phụ thuộc gì vào vị trí và thời điểm hiện tại của nó. Điều này cho ta cơ may tìm được lời giải hiệu quả hơn khi mà hàm mục tiêu có vô số điểm cực trị (thí dụ hàm Weierstrass bậc cao).

3. CƠ SỞ LÝ LUẬN CỦA THUẬT TOÁN TIẾN HÓA

Phỏng sinh học là một hành động vĩ đại táo bạo của loài người và đã có lịch sử từ rất lâu. Tầu ngầm phỏng theo hình dạng cá voi, máy trèo núi phỏng theo con cánh cam, tay máy phỏng theo cánh tay người v.v.. Người ta còn dự định xây dựng các máy tính phỏng theo bộ não của con người. Ý tưởng áp dụng toàn bộ quá trình tiến hóa tự nhiên vào các hệ thống nhân tạo (artificial systems) được bắt đầu từ công trình của Holland [9, 10] và tiếp tục phát triển bởi Goldberg [8] cũng như nhiều tác giả khác. Học thuyết tế bào, học thuyết đầu tiên đi sâu vào bản chất của sự sống, đã được xây dựng cách đây 150 năm và ngày càng hoàn thiện. Tế bào là hệ thống vật chất hoàn chỉnh mang những đặc tính của sự sống. Chất nguyên sinh của tế bào gồm tế bào chất và nhân. Protit của tế bào chất là vật chất biểu hiện các đặc tính của sự sống, nhưng sự tổng hợp các protit này lại được chương trình hóa bởi các phân tử ADN nằm trong nhân. Các đoạn riêng rẽ của ADN được gọi là gien. Phân tử ARN được tạo ra trên khuôn mẫu của gien, chui từ nhân ra tế bào chất làm nhiệm vụ điều khiển quá trình tổng hợp protit. Một trong những đặc tính của sự sống biểu hiện trên tế bào là khả năng tự phân chia để tạo ra các tế bào mới. Quá trình này xảy ra rất phức tạp và tuân theo những định luật hết sức nghiêm ngặt. Đó là các định luật như: định luật tính trội, định luật phân ly và bảo tồn các kiểu gien (genotype) và kiểu hình (phenotype), định luật di truyền kết hợp giới tính v.v.. Tuy đã có thể giải mã sơ đồ gien, nhưng cho đến nay loài người vẫn chưa hiểu đầy đủ những gì đã chi phối quá trình hình thành sự sống và còn rất nhiều vấn đề khác về sự sống cần

tiếp tục tranh luận. Mặc dù vậy, mọi người đều dễ dàng công nhận với nhau, sự sống là hình thức tồn tại vật chất cao nhất và sự tiến hóa theo nguyên lý chọn lọc tự nhiên là một quá trình tối ưu hoàn hảo nhất so với tất cả các quá trình tối ưu mà loài người tạo ra. Tiên đề trên là cơ sở khoa học của các thuật toán tiến hóa.

Trong sinh học, nói đến kiểu gien tức là nói đến tập hợp các gien riêng biệt và nói đến kiểu hình là nói đến những tính trạng biểu hiện ra bên ngoài. Kiểu hình là kết quả của kiểu gien và tác động của môi trường lên cơ thể sinh vật. Các thuật toán tiến hóa khác nhau được xây dựng xuất phát từ cách nhìn kiểu gien hoặc kiểu hình.

Các thuật toán xuất phát từ cách nhìn kiểu gien được gọi là *thuật toán di truyền* (genetic algorithm). Trong các thuật toán di truyền, miền tìm kiếm thường là các miền thuần nhất và không thay đổi bản chất trong suốt quá trình tiến hóa. Sự vận động từ lời giải hiện thời đến lời giải tối ưu là sự vận động nội tại. Các thuật toán tìm kiếm được xây dựng theo cách nhìn kiểu hình được gọi là *các thuật toán tiến hóa* (evolutionary algorithms). Trong các thuật toán tiến hóa, các cá thể được sinh ra phải chịu tác động của môi trường, thí dụ sự tiến hóa của virut [20]. Tuy nhiên cần lưu ý rằng, không có ranh giới rõ ràng giữa các thuật toán di truyền và các thuật toán tiến hóa. Thứ nhất, như trên kia đã nói, kiểu hình bị chi phối bởi kiểu gien. Các thuật toán tiến hóa sử dụng sự mô phỏng quá trình sinh sản, lai ghép và đột biến như các thuật toán di truyền. Nói cách khác, thuật toán di truyền là cơ sở của thuật toán tiến hóa. Thứ hai, các thuật toán di truyền đôi khi cũng diễn ra do tác động bên ngoài, thí dụ chủ quan của con người [15]. Vì lý do này, trong các phần sau sẽ chỉ dùng chung một khái niệm thuật toán tiến hóa.

Như trên đã nói, thuật tiến hóa là thuật toán tìm kiếm lời giải tối ưu dựa trên sự "bắt chước" quá trình tiến hóa tự nhiên. Về phương diện toán học, ta có thể coi thuật toán tiến hóa là phương pháp tìm kiếm ngẫu nhiên tổng quát. Tuy nhiên, thuật toán tiến hóa khác các phương pháp tìm kiếm thông thường ở mấy điểm sau:

- Thuật toán tiến hóa tiến hành quá trình tìm kiếm lời giải tối ưu trên một quần thể (population) và tìm đồng thời một lúc nhiều điểm cực trị có thể có. Do vậy sẽ hạn chế sự kết thúc quá trình tìm kiếm tại điểm cực trị cục bộ và tăng khả năng đạt đến điểm cực trị toàn cục.
- Thuật toán tiến hóa thao tác với các chuỗi a-len (allele) dùng để mã hóa tham số chứ không thao tác trực tiếp với các tham số.
- Thuật toán tiến hóa không sử dụng giá trị hàm mục tiêu mà sử dụng giá trị fit-nit của các cá thể trong quá trình tìm kiếm. Thuật toán tiến hóa cũng không nhất thiết cần đến giá trị đạo hàm của hàm mục tiêu hay các thông tin phụ khác.
- Các luật chuyển đổi thuật toán giữa các bước tìm kiếm là các luật ngẫu nhiên chứ không phải là các luật tiền định.

Cơ thể sống là một hệ thống đa chiều, tự tổ chức và tự ổn định, có khả năng tự thích nghi với những tác động đa dạng của môi trường xung quanh. Rõ ràng không thể mô phỏng đầy đủ quá trình tiến hóa. Việc sử dụng thuật toán di truyền mang nhiều tính heuristic và không chặt chẽ như các phương pháp toán học kinh điển. Tuy vậy, qua các công trình đã được công bố, một thuật toán di truyền thường bao gồm những công việc sau:

- Mã hóa tham số bằng các chuỗi a-len (trên máy tính là các chuỗi nhị phân) có độ dài thích hợp. Các chuỗi a-len này đóng vai trò như các tế bào sống tham gia vào các quá trình sinh sản, chọn lọc tự nhiên, chịu sự chi phối của các qui luật di truyền và đột biến.
- Biến đổi hàm mục tiêu về dạng thích hợp nếu cần thiết và tìm độ đo fit-nit (fitness meassure) làm cơ sở để tiến hành quá trình chọn lọc tự nhiên.
- Tạo lập một quần thể ban đầu với số lượng cá thể cần thiết để tham gia vào quá trình tiến hóa. Như trên đã nêu, các cá thể này không nhất thiết phải tương ứng với một lời giải thỏa đáng của bài toán tối ưu có ràng buộc.

- Mô phỏng quá trình sinh sản và chọn lọc tự nhiên thông qua việc sao chép các cá thể tốt và loại bỏ các cá thể xấu dựa trên độ đo fit-nit. Quá trình này cần phải thực hiện sao cho không phải mọi cá thể có giá trị fitnit nhỏ đều bị đào thải, nghĩa là không làm mất tính đa dạng của quần thể.
- Mô phỏng quá trình lai ghép, trong đó các cặp cá thể kết hợp với nhau để tạo ra các bộ gien mới (các cá thể mới). Các cá thể mới này hòa nhập vào cộng đồng để tham gia quá trình tiến hóa.
- Mô phỏng quá trình đột biến, trong đó một hay một số cá thể bị biến đổi một hay nhiều gien một cách ngẫu nhiên. Các cá thể bị biến đổi gien sẽ biến thành các cá thể mới, có thể tốt hơn hoặc xấu hơn theo độ đo fit-nit. Quá trình này xảy ra với xác suất nhỏ nhưng vô cùng quan trọng vì như đã biết, không có đột biến thì không có tiến hóa.

Các công việc trên, gọi là các toán tử gien trong các thuật toán di truyền (genetic operator), được thực hiện xen kẽ nhau theo một trình tự nào đó tùy thuộc vào vấn đề cụ thể. Đối với những vấn đề đơn giản, ba công việc đầu chỉ cần làm một lần, ba công việc sau được lặp lại cho đến khi tiêu chuẩn dừng thỏa mãn. Tuy nhiên, như sẽ trình bày trong các phần sau, đối với cặc vấn đề phức tạp, ba công việc đầu có thể phải thực hiện trong cả quá trình tìm kiếm lời giải tối ưu. Tiêu chuẩn dừng có thể chọn là một hoặc kết hợp các thông tin như sau:

- Số thế hệ tiến hóa đã vượt quá một số cho trước.
- Sự tiến hóa hầu như không diễn ra nữa. Nói cách khác, sự khác biệt giữa các cá thể trong quần thể qua nhiều thế hệ là không đáng kể.
- Giá trị fit-nit của các cá thể tốt nhất trong quần thể hầu như không tăng ở nhiều thế hệ tiến hóa nối tiếp nhau.
- Dựa vào ý kiến của chuyên gia về vấn đề đang quan tâm theo nguyên lý *top N* trong [26].

Sau đây chúng ta sẽ đi sâu vào những bước cụ thể của các thuật toán tiến hóa.

4. MÃ HÓA THAM SỐ BẰNG CÁC CHUỖI A-LEN

Như trên đã nêu, thuật toán tiến hóa không tiến hành quá trình tìm kiếm lời giải tối ưu trực tiếp trên các tham số, trái lại các tham số trước hết được mã hóa bởi các chuỗi a-len và trở thành các cá thể trong quần thể của quá trình tiến hóa. Sau quá trình tiến hóa, các cá thể có độ đo fit-nit lớn hơn sẽ được chọn ra và giá trị tham số tối ưu sẽ nhận được qua phép biến đổi ngược lại (phép giải mã). Phép mã hóa dễ hình dung nhất và dễ thể hiện trên máy tính nhất là phép mã hóa nhị phân (binary encoding). Để dễ hình dung, chúng ta xét thí dụ được nêu ở [26]. Tìm lời giải tối ưu chung của

$$f_1(x,y) = 0{,}5 - \frac{\sin^2\sqrt{x^2+y^2} - 0{,}5}{\left(1+0{,}01(x^2+y^2)\right)^2}, \tag{4.1}$$

$$f_2(x,y) = 1 - (x-0{,}3)^2 - (y-0{,}3)^2, \tag{4.2}$$

với ràng buộc

$$g(x,y) = x + y - 0{,}25 \le 0. \tag{4.3}$$

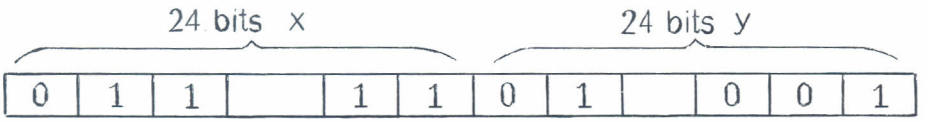
Ở đây có hai tham số là x và y, có thể nhận các giá trị trong khoảng $(-\infty, +\infty)$. Tuy nhiên trên thực tế tính toán người ta chỉ xét các giá trị của x và y trong khoảng $[-\alpha, +\alpha]$ với α là một số đủ lớn. Ta mã hóa x và y bằng các chuỗi a-len (trên máy tính là các chuỗi nhị phân). Giả sử dùng một chuỗi gồm 48 bits (6 bytes) trong đó 24 bits đầu mã hóa x và 24 bits sau mã hóa y. Chuỗi 48 bits này đóng vai trò là một cá thể của quá trình tiến hóa (hình 4.1). Với 24 bits nhị phân ta có thể diễn tả các số từ 0 đến $2^{24} - 1 = 16777215$. Trên thực tế ta chỉ muốn giới hạn miền tìm kiếm trong khoảng $[-20, +20]$. Khi đó x và y được xác định bởi

$$x = \frac{40}{16777215}\{x\} - 20, \quad y = \frac{40}{16777215}\{y\} - 20, \tag{4.4}$$

trong đó $\{x\}$ chỉ nội dung 24 bits đầu vào và $\{y\}$ chỉ nội dung 24 bits tiếp theo của chuỗi a-len. Một cách tổng quát, giả sử chọn n bits mã hóa mỗi tham số p_i, $i = 1, 2, \ldots, m$. Tham số p_i có cận dưới bằng a_i, cận trên bằng b_i. Giá trị thật của p_i được xác định bởi

$$p_i = a_i + \frac{b_i - a_i}{2^n - 1}\{p_i\}, \tag{4.5}$$

trong đó $\{p_i\}$ chỉ nội dung của n bits mã hóa p_i. Một chuỗi a-len khi đó sẽ có độ dài bằng $n.m$ bits. Trong một bài toán, không nhất thiết tất cả các tham số đều phải mã hóa bằng các chuỗi có độ dài bằng nhau.



Hình 4.1

Số lượng bits nhị phân sử dụng để mã hóa tham số đóng vai trò quan trọng. Như ta đã biết, ở các sinh vật bậc thấp, một phân tử ADN cũng đã chứa hàng ngàn gien. Còn ở các sinh vật bậc cao như người, một phân tử ADN chứa tới hàng triệu gien. Chuỗi các bit mã hóa càng dài thì việc mô phỏng quá trình tiến hóa càng có hiệu quả, càng sát thực với tự nhiên hơn. Số lượng bit mã hóa không đủ lớn sẽ gây ra hiện tượng hội tụ chậm trong hầu hết các trường hợp thực tế. Tuy nhiên, số bit càng lớn đòi hỏi dung lượng bộ nhớ và thời gian xử lý càng lớn. Khó khăn này gần giống như khó khăn trong việc thiết kế các bộ biến đổi A/D của các kỹ sư điện tử. Các thuật toán tiến hóa kinh điển thường dùng phương pháp mã hóa tĩnh, nghĩa là các tham số được mã hóa ngay từ đầu và không thay đổi trong suốt quá trình tìm kiếm. Để dung hòa giữa đòi hỏi về số lượng bit mã hóa chuỗi a-len và rút ngắn thời gian tính toán, người ta đã đưa ra một số giải pháp sau:

- Các tham số được diễn tả theo kiểu dấu phảy di động (floating point) bằng các trường [18].
- Thay đổi độ dài của các chuỗi a-len trong quá trình tiến hóa nhờ một cơ chế tự thích nghi (adaptation) [24].
- Dùng phương pháp điều khiển mờ để thay đổi độ dài chuỗi a-len trong quá trình tiến hóa [22].

Việc thay đổi độ dài chuỗi a-len khi mã hóa tham số và việc rút gọn miền tìm kiếm có quan hệ với nhau. Chúng ta sẽ xét vấn đề này kỹ hơn trong Phần 11.

5. XÂY DỰNG ĐỘ ĐO FIT-NIT

Thuật ngữ *độ đo fit-nit*, tiếng Anh gọi là *fitness measure*, dùng để chỉ sức mạnh của mỗi cá thể, khả năng thích ứng của cá thể với môi trường, mức độ biểu hiện các tính trạng tốt của cá thể trong quần thể. Thí dụ, một giống lúa cho năng suất cao hơn và có khả năng chống sâu bệnh tốt hơn, ta nói rằng giống lúa đó có độ đo fit-nit lớn hơn. Việc xây dựng độ đo fit-nit cũng quan trọng như việc mã hóa tham số, vì quá trình chọn lọc tự nhiên sẽ dựa vào độ đo fit-nit chứ không dựa trực tiếp vào giá trị của hàm mục tiêu. Độ đo bao giờ cũng là một số không âm trong khi hàm mục tiêu có thể nhận giá trị bất kỳ. Quá trình chọn lọc tự nhiên giữ lại các cá thể có giá trị fit-nit cao hơn. Nói cách khác, quá trình chọn lọc tự nhiên có xu hướng cực đại hóa giá trị fit-nit. Trong khi đó, bài toán tối ưu hóa có thể là bài toán cực đại hóa (maximization) hoặc cực tiểu hóa (minimization).

Qua phân tích trên đây ta thấy, việc xây dựng độ đo fit-nit được tiến hành như sau. Nếu vấn đề quan tâm là một bài toán cực tiểu hóa thì trước hết phải biến đổi nó thành bài toán cực đại hóa. Ta biết rằng, cực tiểu hóa và cực đại hóa là hai bài toán đối ngẫu. Việc chuyển đổi từ bài toán này sang bài toán kia không có gì khó khăn [1, 22]. Nếu bản thân hàm mục tiêu là một hàm không nhận giá trị âm, có thể sử dụng luôn nó làm độ đo fit-nit. Nếu hàm mục tiêu nhận giá trị âm, độ đo fit-nit được chọn là một hàm tuyến tính sao cho hàm này ánh xạ miền giá trị của hàm mục tiêu

vào một khoảng không âm. Thí dụ hàm mục tiêu $f(x)$ có miền giá trị là $[c_l, c_u]$, $c_l, c_u \in R$. Trong trường hợp này, độ đo fit-nit có thể chọn là $F(x) = f(x) + c_l$. Trong thực tế ứng dụng các thuật toán tiến hóa, thang đo fit-nit thường được căn chỉnh để tránh hiện tượng hội tụ sớm (premature convergence). Khi bắt đầu quá trình tiến hóa, nếu các cá thể với giá trị fit-nit cao chiếm đa số áp đảo trong quần thể, các cá thể với giá trị fit-nit thấp ít có cơ may tồn tại qua chọn lọc tự nhiên. Tính đa dạng của quần thể khi đó bị giảm, quá trình tiến hóa trở nên trì trệ. Để vượt qua tình trạng này, thang đo fit-nit cần phải căn chỉnh lại. Thủ tục căn chỉnh đơn giản nhất là thủ tục căn chỉnh tuyến tính (linear scaling). Gọi độ đo fit-nit ban đầu là F và độ đo fit-nit đã căn chỉnh là F', F_{tb} và F'_{tb} là giá trị fit-nit trung bình của quần thể. Quan hệ giữa F và F' được xác định bởi

$$F' = \alpha F + \beta, \tag{5.1}$$

trong đó α và β là các số được chọn sao cho

$$F'_{tb} = F_{tb} \quad \text{và} \quad F'_{\max} = \kappa F'_{tb}. \tag{5.2}$$

Trong biểu thức (5.2) κ là tỷ lệ hợp lý giữa giá trị fit-nit của cá thể tốt nhất so với giá trị fit-nit trung bình của quần thể.

6. MÔ PHỎNG QUÁ TRÌNH SINH SẢN

Sinh sản là khả năng đặc biệt của các cơ thể sống. Cha mẹ sinh ra con cái, thế hệ trước sinh ra thế hệ sau. Quần thể là nền tảng của tiến hóa, sinh sản tạo ra quần thể mới. Như ta đã biết, các hình thức sinh sản trong tự nhiên rất đa dạng và phong phú. Các sinh vật có mức độ tiến hóa càng cao thì quá trình sinh sản càng phức tạp. Trong tự nhiên, chúng ta không thể tách quá trình sinh sản ra khỏi các quá trình khác như quá trình lai ghép và đột biến. Quá trình sinh sản được mô phỏng trong các thuật toán tiến hóa đã công bố có thể xem như quá trình sinh sản vô tính. Nghĩa là, cá thể con sinh ra giống hoàn toàn cá thể mẹ. Việc sao chép cá thể mẹ thành cá thể con được định đoạt bởi sự chọn lọc tự nhiên. Quá trình chọn lọc tự nhiên được mô phỏng sao cho thế hệ mới có giá trị fit-nit trung bình lớn hơn thế hệ trước. Vì các cá thể con giống hoàn toàn các cá thể mẹ sinh ra nó, sự tăng giá trị fit-nit trung bình đồng nghĩa với sự có mặt nhiều hơn của các cá thể có giá trị fit-nit cao hơn. Quá trình sinh sản đơn giản này có thể diễn tả như sau. Giả sử quần thể hiện thời gồm N cá thể, có số thứ tự từ 1 đến N, với các giá trị fit-nit $F_1, F_2, \ldots, F_N$. Trước hết ta tính tổng giá trị fit-nit của quần thể

$$F = F_1 + F_2 + \ldots + F_N \tag{6.1}$$

và tỷ lệ đóng góp của mỗi cá thể vào tổng giá trị fit-nit

$$w_i = \frac{F_i}{F} \cdot 100\,(\%). \tag{6.2}$$

Để thế hệ sau có giá trị fit-nit trung bình lớn hơn thế hệ trước, cách hợp lý nhất là tạo ra một cơ chế sao cho cá thể thứ i sẽ có con với xác suất w_i. Cách đơn giản nhất xây dựng cơ chế này là làm một cái hộp đựng các quả cầu giống nhau, trên mỗi quả cầu ghi một số từ 1 đến N. Số lượng các quả cầu mang số i chia cho tổng số quả cầu trong hộp đúng bằng w_i. Nhắm mắt lại và thò tay vào hộp nhặt hú họa một quả cầu. Số ghi trên quả cầu này cho ta cá thể thứ nhất của quần thể thế hệ mới. Trả quả cầu vào hộp, trộn đều và lấy hú họa một quả cầu thứ hai để được cá thể thứ hai của quần thể mới. Lặp lại thí nghiệm cho đến khi thu được đủ số cá thể của quần thể mới. Nên nhớ rằng, quần thể mới có thể gồm đúng N cá thể, nhưng cũng có thể gồm nhiều hơn N cá thể.

Quá trình sinh sản cũng có thể mô phỏng theo kiểu lập bảng đấu loại, tương tự cách tổ chức thi đấu giải bóng đá dành cho chức vô địch thế giới (word cup), cụ thể như sau.

1) Chia ngẫu nhiên N cá thể thành K nhóm.
2) Chọn cá thể có giá trị fit-nit lớn nhất trong nhóm làm cá thể của thế hệ mới.
3) Lặp lại các bước 1) và 2) cho đến khi thu được đủ số lượng cá thể mong muốn của quần thể mới.

Chúng ta có thể thực hiện bước 1) theo cách chỉ lập một số ít nhóm (thí dụ 2 hoặc 3 nhóm) với số lượng thành viên mỗi nhóm bị hạn chế (thí dụ mỗi nhóm chỉ có 5 cá thể). Như vậy không phải mọi cá thể đều đứng trong một nhóm. Các cá thể được đưa vào nhóm theo thể thức bốc thăm (tung con xúc sắc N mặt).

Một cách khác để mô phỏng quá trình sinh sản qua chọn lọc tự nhiên đã được nêu trong [5], gồm các bước sau.

1) Đặt $V_1 = F_1$
2) Tính $V_2 = F_1 + F_2 = V_1 + F_2$.
3) Tính $V_i = V_{i-1} + F_i$ $(i = 3, 4, ..., N)$.
4) Tạo một số ngẫu nhiên trong khoảng từ 0 đến V_N. Giả sử giá trị nhận được là V_k.
5) Chọn cá thể đầu tiên có giá trị fit-nit lớn hơn hoặc bằng V_k để làm cá thể của thế hệ mới.
6) Lặp lại bước 4) và 5) cho đến khi thu được đủ số lượng cá thể mong muốn của quần thể mới.

Còn nhiều cách chọn lọc tự nhiên mà chúng ta có thể áp dụng. Cần chú ý là, nên cân đối giữa việc sinh sản các cá thể có giá trị fit-nit cao và tính đa dạng của quần thể. Điều này đặc biệt quan trọng khi giải các bài toán tối ưu với ràng buộc. Chúng ta sẽ xét kỹ vấn đề này trong Phần 9 của bài báo này.

7. MÔ PHỎNG QUÁ TRÌNH LAI GHÉP

Quá trình lai ghép (crossover) là quá trình hình thành nhiễm sắc thể mới trên cơ sở các nhIễm sắc thể bố mẹ, bằng cách ghép một hay nhiều đoạn a-len của hai nhiễm sắc thể bố mẹ với nhau. Quá trình lai ghép có thể mô phỏng như sau.

- Chọn ngẫu nhiên hai cá thể bất kỳ của quần thể. Giả sử nhiễm sắc thể của bố gồm m a-len và nhiễm sắc thể của mẹ gồm n a-len.
- Tạo một số nguyên ngẫu nhiên trong khoảng từ 1 đến $m - 1$ (điểm phân chia chuỗi a-len bố). Giả sử điểm đó chia chuỗi m a-len thành hai chuỗi nhỏ m_1 và m_2.
- Tạo một số nguyên ngẫu nhiên trong khoảng từ 1 đến $n - 1$ (điểm phân chia chuỗi a-len mẹ). Giả sử điểm đó chia chuỗi n a-len thành hai chuỗi nhỏ n_1 và n_2.
- Ghép các chuỗi a-len nhỏ với nhau để tạo ra các chuỗi a-len mới. Theo thí dụ trên các chuỗi a-len mới sẽ là $m_1 + n_1$ và $m_2 + n_2$ hoặc $m_1 + n_2$ và $m_2 + n_1$. Nếu m và n có độ dài bằng nhau và ta chỉ muốn tạo ra các chuỗi có độ dài không đổi thì chỉ có chuỗi $m_1 + n_2$ và $m_2 + n_1$ là cá thể của thế hệ mới.

Cách mô phỏng quá trình lai ghép trên có tên gọi là quá trình lai ghép một điểm (one-point crossover) và đã được sử dụng trong [25]. Nên nhớ rằng, một chuỗi a-len mã hóa cùng một lúc nhiều tham số. Chỉ chọn một điểm lai ghép ngẫu nhiên có khả năng làm cho nhiều đoạn a-len không thay đổi ở nhiều thế hệ tiếp theo. Tính đa dạng của quần thể bị hạn chế. Muốn làm tăng tính đa dạng phong phú của quần thể, nên sử dụng cách lai ghép nhiều điểm (multi-point crossover). Trong thí dụ đã nêu, giả sử tách chuỗi m a-len của bố thành k chuỗi nhỏ $m_1, m_2, ..., m_k$ và chuỗi n a-len của mẹ thành các chuỗi nhỏ $n_1, n_2, ..., n_k$. Cá thể mới được sinh ra là chuỗi lai ghép giữa các m_i và n_j, thí dụ $m_1 + n_k + m_2 + n_{k-1} + ... + m_r + n_{k-r+1}$ $(r = k/2)$. Cũng có thể thực hiện cách lai ghép như đã trình bày trong [26]. Ở đó, trước khi thực hiện quá trình lai ghép, ta tạo ra các đại cá thể (representative individual) bằng cách ghép nhiều chuỗi a-len nguyên thủy với nhau. Thực hiện quá trình lai ghép nhiều điểm với hai đại cá thể. Cuối cùng, dùng phương pháp tách ngẫu nhiên đại cá thể để sinh ra các cá thể con có độ dài nguyên thủy. Cách lai ghép này phức tạp và chỉ nên áp dụng cho trường hợp các tham số được mã hóa với số a-len bằng nhau.

Ở mỗi bước tiến hóa, quá trình lai ghép có thể được thực hiện nhiều lần. Lai ghép làm tăng tính đa dạng của quần thể. Tuy nhiên quá nhiều con lai trong quần thể sẽ làm cho tính hội tụ cục bộ bị giảm. Vì vậy, tùy từng bài toán cụ thể mà chọn tỷ lệ lai ghép. Tỷ lệ lai ghép được chọn là

0,65 cho thí dụ mô phỏng ở [3], 0,95 cho thí dụ mô phỏng ở [24] và được chọn là 0,33 cho các ứng dụng ở [25] và [26].

Quá trình chọn cặp bố mẹ và quá trình tìm điểm lai ghép là quá trình ngẫu nhiên có phân bố đều. Điều này cần thiết bởi nó đảm bảo khả năng cặp đôi của các cá thể là như nhau và khả năng lai ghép của mọi a-len là như nhau.

8. MÔ PHỎNG QUÁ TRÌNH ĐỘT BIẾN

Đột biến (mutation) là hiện tượng cá thể con mang một hay nhiều tính trạng không có trong mã di truyền của bố mẹ. Trong thế giới sinh vật, đột biến xảy ra với xác suất rất nhỏ nhưng lại chính là động lực của tiến hóa. Xác suất xảy ra hiện tượng đột biến được chọn bằng 0,008 cho thí dụ mô phỏng ở [3], được chọn bằng 0,001 cho thí dụ mô phỏng ở [24] và được chọn bằng 0,004 cho các ứng dụng ở [25] và [26]. Quá trình đột biến có thể mô phỏng như sau:

- Chọn ngẫu nhiên một cá thể bất kỳ của quần thể.
- Giả sử các cá thể đều có độ dài chuỗi a-len bằng m. Tạo một số nguyên ngẫu nhiên k trong khoảng từ 1 đến m.
- Thay đổi a-len thứ k (bit 1 đổi thành 0, bit 0 đổi thành 1).
- Trả cá thể vào quần thể tham gia quá trình tiến hóa tiếp theo.

Cũng như khi mô phỏng quá trình lai ghép, số ngẫu nhiên k phải được tạo ra bởi một quá trình ngẫu nhiên có phân bố đều. Phân bố đều đảm bảo khả năng đột biến xảy ra ở mọi a-len của chuỗi đều bằng nhau.

Hiện tượng đột biến trên đây là đột biến một a-len. Cũng có thể sử dụng đột biến nhiều a-len bằng cách lặp lại một số lần đột biến một a-len. Nên đột biến ở bao nhiêu a-len là tốt nhất? Cho đến nay chưa có công trình nào đưa ra được một định hướng có tính thuyết phục. Chúng ta có thể thực hiện quá trình đột biến nhiều a-len theo cách tìm số lần đột biến thông qua một quá trình ngẫu nhiên phụ có phân bố Poisson. Đó là quá trình với phân bố có dạng

$$p(x) = \frac{\lambda^x}{x!} e^{-\lambda}. \tag{8.1}$$

Sở dĩ ta chọn quá trình Poisson vì quá trình này có kỳ vọng và phương sai bằng nhau (bằng λ theo công thức 8.1). Trong lý thuyết phục vụ đám đông, quá trình Poisson mô tả rất tốt hiện tượng xuất hiện nhu cầu phục vụ. Ta có thể hình dung hiện tượng đột biến như là quá trình phục vụ sự tiến hóa, bởi vì đột biến là động lực của tiến hóa tự nhiên. Một thực tế nữa dễ nhận thấy là, chuỗi a-len càng dài thì số điểm đột biến cũng cần phải nhiều hơn. Vì vậy nên chọn λ trong công thức (8.1) tỷ lệ thuận với độ dài của chuỗi. Tuy nhiên theo kinh nghiệm mô phỏng, λ chỉ nên nhận giá trị không quá 2% số bit của một cá thể.

Thời điểm thực hiện quá trình đột biến có thể trước hoặc sau quá trình sinh sản cũng như trước hoặc sau quá trình lai ghép. Cách tốt nhất là tại một thời điểm ngẫu nhiên. Cho chạy chương trình tạo số ngẫu nhiên 0 hoặc 1 sau mỗi lần thực hiện quá trình sinh sản và lai ghép. Nếu kết quả là 1 sẽ cho thực hiện quá trình đột biến.

9. XỬ LÝ ĐIỀU KIỆN RÀNG BUỘC CỦA BÀI TOÁN TỐI ƯU

Như các phần trên đã nói, các thuật toán tiến hóa là các thuật toán tìm kiếm lời giải tối ưu trong miền tham số cho trước. Nói ngắn gọn, các thuật toán tiến hóa chỉ giải quyết bài toán có dạng chuẩn diễn tả như sau:

$$\textit{Tìm } x \in \Omega := \{x | \alpha \leq x \leq \beta\} \textit{ sao cho } x \textit{ cực đại hóa } f(x),$$

trong đó α và β là giới hạn dưới và giới hạn trên của tham số. Để giải quyết bài toán tối ưu dạng tổng quát với các ràng buộc cho bởi công thức (2.4) và (2.5), ta có thể làm như sau.

- Trong quá trình tiến hóa, kiểm tra xem một cá thể mới sinh ra có thỏa mãn điều kiện ràng buộc không trước khi xác định giá trị fit-nit của nó. Nếu nó không thỏa mãn điều kiện ràng buộc thì loại bỏ ngay. 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 lô-gic, 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 ra đều không thỏa mãn điều kiện ràng buộc thì vẫn tìm được một cá thể làm lời giải khả thi. Nhưng khi đi tìm lân cận tốt nhất ta lại phải giải một bài toán tối ưu phụ khác.
- Đưa vào hàm mục tiêu một hàm phụ trợ gọi là *hàm phạt* và chuyển bài toán tối ưu với ràng buộc thành bài toán tối ưu không có ràng buộc. 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 làm giảm giá trị fit-nit 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.

Có nhiều công trình nghiên cứu cách xây dựng hàm phạt cho bài toán tối ưu với ràng buộc khi sử dụng thuật toán tiến hóa [7, 21, 26]. Như trên đã nói, mục đích việc đưa ra hàm phạt là làm thay đổi giá trị fit-nit của các cá thể vi phạm điều kiện ràng buộc. Giả sử hàm fit-nit tương ứng với hàm mục tiêu $\lambda(x)$. Hàm phạt được chọn là $\xi(x)$. Hàm fit-nit bây giờ sẽ có dạng

$$\varphi(x) = \lambda(x) + \xi(x)\,\mu. \tag{9.1}$$

Trong biểu thức (9.1) $\mu = 0$ 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 [21].

- $\xi(x) = (\nu/2)^{\sigma}$, trong đó ν là số các điều kiện bị vi phạm, $\nu \in \{1, 2, \ldots, k\}$, σ là hệ số nghiêm khắc (severity factor).
- $\xi(x) = \delta(x).\phi(\tau) = \delta(x).[\phi_0 + \tau.j]$, 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, trong khi đó một cá thể khác vi phạm nhiều điều kiện nhưng chỉ ở mức độ nhẹ. 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 nêu trên, trong [26] có đề xuất cách xây dựng hàm phạt như sau:

$$\xi(x) = \gamma.e^{\frac{\tau.\eta}{\varepsilon}}.\sum_{i=1}^{k}\phi(g_i(x) - a_i). \tag{9.2}$$

Trong biểu thức (9.2), 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 (9.2) tăng theo hàm mũ của số lượng cá thể vi phạm điều kiện ràng buộc và sự kéo dài qua các thế hệ của nó. Hàm $\phi(.)$ trong biểu thức (9.2) 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.

10. VẤN ĐỀ HỘI TỤ CỦA CÁC THUẬT TOÁN TIẾN HÓA

Vấn đề hội tụ là vấn đề sống còn của các thuật toán lặp, nghĩa là, có tìm thấy lời giải sau một số hữu hạn các bước lặp hay không. Khi xây dựng các thuật toán lặp, câu hỏi đầu tiên phải trả lời là thuật toán đó có hội tụ không và hội tụ với tốc độ như thế nào. Việc chứng minh bằng lý thuyết tính hội tụ của một thuật toán lặp không phải lúc nào cũng thực hiện được. Đối với các kỹ sư, việc chứng minh càng khó khăn hơn. Như đã thấy rõ qua các phần trên, các thuật toán tiến hóa cũng là các thuật toán lặp và mang đặc tính heuristic. Tính hội tụ của nó chỉ được khẳng định qua thực tế ứng dụng. Tuy nhiên một câu hỏi có thể đặt ra là, những yếu tố nào ảnh hưởng đến tính hội tụ của các thuật toán tiến hóa. Qua thực tế mô phỏng chúng tôi đã gặp tình huống như sau: Khi quá trình tìm kiếm đang tiến đến điểm tối ưu toàn cục, quá trình lai ghép và đột biến có thể sinh ra các cá thể cho giá trị fit-nit nhỏ hơn so với giá trị fit-nit trung bình của quần thể. Nếu số lượng cá thể trong quần thể không đủ lớn, hiện tượng này có thể xảy ra trong nhiều thế hệ liên tiếp. Khi đó giá trị fit-nit của cá thể tốt nhất trong quần thể không tăng nữa tuy nó chưa phải là lời giải tối ưu. Để khắc phục tình trạng này, ta sử dụng phương pháp rút gọn miền tìm kiếm. Khi thấy quá trình tìm kiếm có xu hướng trì trệ (giá trị fit-nit cực đại hoặc giá trị fit-nit trung bình tăng chậm), ta tiến hành việc rút gọn miền tìm kiếm. Giả sử miền tìm kiếm hiện thời là $[\alpha, \beta]$. Giá trị tham số tương ứng với cá thể tốt nhất là $\alpha \leq \chi \leq \beta$. Chọn một số δ thích hợp, chẳng hạn chọn $\delta = 0{,}362\,(\beta - \alpha)$. Khoảng tìm kiếm mới sẽ là $[\chi - \delta, \chi + \delta]$. Cách làm này đã được sử dụng có hiệu quả trong [26].

Một hiện tượng khác xảy ra khi ứng dụng các thuật toán tiến hóa là hiện tượng hội tụ sớm (premature convergence). Streifel và cộng sự đã chỉ ra rằng, khi các chuỗi a-len có cấu trúc gần giống nhau thì hiện tượng hội tụ sớm xảy ra, làm mất khả năng đạt đến điểm tối ưu toàn cục [24]. Để dễ hình dung, ta xét trường hợp các tham số đã được mã hóa bằng chuỗi nhị phân. Khi đó sự khác biệt giữa hai chuỗi a-len sẽ được tính bằng số bit 1 của phép cộng modul 2 của hai chuỗi nhị phân. Thí dụ, hai chuỗi 01100010 và 10001110 có độ khác biệt cấu trúc bằng 5. Nếu sự khác biệt cấu trúc lớn nhất trong quần thể nhỏ hơn một giới hạn nào đó, cần tiến hành thay đổi miền tìm kiếm. Để dễ so sánh, ta lấy lại thí dụ đã nêu. Giả sử miền tìm kiếm hiện thời là $[\alpha, \beta]$. Trung điểm của miền tìm kiếm ở đây là $c = 0{,}5\,(\beta - \alpha)$. Giá trị tham số tương ứng với cá thể tốt nhất là $\alpha \leq \chi \leq \beta$. Biểu thức $\sigma = |\chi - c|$ được gọi là khoảng cách giữa điểm tốt nhất hiện thời và trung tâm miền tìm kiếm. Việc thay đổi miền tìm kiếm sẽ tiến hành theo luật IF ... THEN ... của lý thuyết mờ. Cụ thể như sau:

- IF σ rất nhỏ THEN miềm tìm kiếm co hẹp nhiều
- IF σ nhỏ THEN miềm tìm kiếm co vừa phải
- IF σ lớn THEN miềm tìm kiếm mở rộng vừa phải

- IF σ rất lớn THEN miền tìm kiếm mở rộng nhiều
- IF $\sigma \approx 0$ lớn THEN miền tìm kiếm không thay đổi

Phương pháp đơn hình và phương pháp đa hình cũng là các phương pháp dựa trên cơ sở rút gọn kết hợp với dịch chuyển miền tìm kiếm. Sự khác biệt giữa các phương pháp đơn hình và đa hình so với thuật toán tiến hóa có thể chỉ ra như sau. Ở phương pháp đơn hình và đa hình, trong quá trình dịch chuyển, miền tìm kiếm sẽ co dần và cuối cùng trở thành một điểm là điểm tối ưu. Ở các thuật toán tiến hóa, miền tìm kiếm không nhất thiết phải co nhỏ thành một điểm. Hơn nữa, miền tìm kiếm không phải co dần đều mà có thể khi co hẹp, khi mở rộng, tùy thuộc vào tình trạng của quần thể hiện thời.

Trong phần mở đầu chúng ta nói rằng, các thuật toán tiến hóa không đòi hỏi thông tin về gradient của hàm mục tiêu. Tuy nhiên, để tăng thêm tính hội tụ của các thuật toán tiến hóa, trong trường hợp có thể, cũng không nên bỏ qua những thông tin này. Cần nghiên cứu thêm khả năng sử dụng các thông tin về gradient vào việc hiệu chỉnh miền tìm kiếm. Rõ ràng việc mở rộng miền tìm kiếm về phía có gradient lớn hơn và rút bớt ở phía có gradient nhỏ hơn sẽ tốt hơn là mở rộng về cả hai phía của điểm trung tâm như hai cách làm trên kia.

11. CHỌN ĐỘ LỚN CỦA QUẦN THỂ VÀ TẠO LẬP QUẦN THỂ BAN ĐẦU

Độ lớn của quần thể (population size) đóng vai trò quan trọng trong các thuật toán tiến hóa. Thông thường một quần thể phải gồm hàng trăm cá thể. Đối với những vấn đề đơn giản như các thí dụ trong [3, 24, 25, 26], độ lớn của quần thể chọn bằng 100. Đối với những vấn đề phức tạp hơn, độ lớn của quần thể đòi hỏi phải gấp nhiều lần. Tuy nhiên, số cá thể của quần thể càng lớn thì dung lượng bộ nhớ cần thiết càng lớn và quá trình tính toán giá trị fit-nit cho cả quần thể càng lâu. Vì vậy, chọn độ lớn của quần thể phải phụ thuộc vào vấn đề đặt ra. Tốt hơn cả, nên áp dụng các thuật toán tiến hóa hai giai đoạn.

- *Giai đoạn 1 (giai đoạn tiến hóa thô)*
  - Dùng một số ít bit để mã hóa các tham số.
  - Tạo ngẫu nhiên một số cá thể để làm quần thể ban đầu.
  - Thực hiện quá trình tiến hóa đơn giản (lai ghép một điểm, đột biến một a-len, không biến đổi miềm tìm kiếm).
  - Dừng quá trình tiến hóa sau một số thế hệ.
- *Giai đoạn 2 (giai đoạn tiến hóa chính xác)*
  - Mã hóa lại các tham số bằng các chuỗi a-len có độ dài thỏa đáng.
  - Thực hiện quá trình tiến hóa với mức độ tinh xảo hơn (lai ghép nhiều điểm, đột biến nhiều a-len, hiệu chỉnh miền tìm kiếm).

Ở giai đoạn 1, do các cá thể lúc đầu được chọn ngẫu nhiên nên chúng rất khác nhau. Do đặc điểm của quá trình tiến hóa, thế hệ tiếp theo sẽ tốt hơn thế hệ trước. Vì số bit mã hóa ít và các toán tử tiến hóa đơn giản nên tốc độ tiến hóa nhanh. Tuy nhiên, vì số bit mã hóa ít nên không đủ chính xác để tìm thấy lời giải đích thực. Các toán tử tiến hóa tinh xảo hơn ở giai đoạn 2 sẽ hội tụ tới điểm tối ưu chính xác. Chúng ta cũng có thể tiến hành giai đoạn một theo cách khác, trong đó không sử dụng toán tử đột biến và quá trình lai ghép chỉ xảy ra với xác suất rất nhỏ. Cách làm này hàm chứa hiện tượng hội tụ sớm như đã trình bày ở phần trên. Ở giai đoạn tiến hóa thô, chúng ta không ngăn ngừa hiện tượng hội tụ sớm, trái lại còn tận dụng nó. Giai đoạn 1 sẽ được lặp lại một số lần. Khi quá trình tiến hóa thô hội tụ, ta chọn những cá thể có giá trị fit-nit lớn để tạo lập quần thể ban đầu cho quá trình tiến hóa ở giai đoạn 2.

12. ỨNG DỤNG CÁC THUẬT TOÁN TIẾN HÓA TRONG ĐIỀU KHIỂN TỰ ĐỘNG

Như phần đầu đã nói, phạm vi ứng dụng của các thuật toán tiến hóa rất rộng rãi, trong đó có lĩnh vực điều khiển tự động [2, 3, 4, 11 - 17, 25]. Các thuật toán tiến hóa có thể được áp dụng để giải

quyết các vấn đề sau đây:

- Giải bài toán nhận dạng tham số mô hình và bài toán rút gọn mô hình động học hệ thống [14, 27].
- Giải phương trình Riccati trong bài toán điều khiển tối ưu tuyến tính [17].
- Thiết kế các bộ điều khiển tối ưu, thí dụ điều khiển tối ưu theo tiêu chuẩn H_∞ [25].
- Điều khiển rô-bôt và điều khiển thông minh [3, 20].
- Xây dựng các hệ luật trong điều khiển mờ [4].
- Điều khiển các hệ phi tuyến [13].

Cần chú ý rằng, việc mô phỏng các thuật toán tiến hóa cho đến nay nhìn chung còn cần thời gian xử lý tương đối dài. Vì vậy việc cài đặt các thuật toán vào các cơ chế điều khiển có phản hồi hoặc điều khiển theo thời gian thực chưa thực hiện được, ngoại trừ các hệ có ứng dụng phương pháp xử lý song song hoặc xử lý mềm. Việc ứng dụng các thuật tiến hóa có những ưu điểm lớn sau đây:

- Tính bền vững của lời giải (solution robustness) được đảm bảo.
- Giải quyết được các bài toán điều khiển tối ưu phi tuyến một cách dễ dàng hơn vì các thuật toán tiến hóa không cần nhiều thông tin như các thuật toán kinh điển.
- Giải quyết được những bài toán điều khiển khi mô hình toán học có độ phức tạp hay độ bất định lớn.

Tính bền vững là một trong những vấn đề lớn của lý thuyết điều khiển hiện đại đang thu hút sự chú ý của các nhà khoa học. Thuật toán tiến hóa vận hành trên quần thể và cùng một lúc tìm nhiều điểm tối ưu. Lời giải được hình thành qua tiến hóa ngẫu nhiên khiến cho nó có bản chất bền vững. Hơn thế nữa, dựa vào kinh nghiệm chuyên gia, từ các cá thể với giá trị fit-nit lớn có thể chọn ra các cá thể thích hợp. Các cá thể đó có thể chỉ là các lời giải cận tối ưu (suboptimal) nhưng lại có tính bền vững rất lớn. Điều khiển các hệ thống phức tạp bao gồm nhiều phân hệ tương tác chéo với nhau (MIMO systems, large scale systems), cho đến nay vẫn là vấn đề nan giải. Đề xuất của Koza sử dụng các thuật toán tiến hóa với diễn tả đa hợp (S-expression [3]) mở ra một khả năng mới cho lớp vấn đề này. Có thể hy vọng rằng, trong tương lai gần, các hệ thống nhiều chiều và các hệ thống lớn không còn là nỗi lo ngại của các nhà điều khiển tự động. Đối với các hệ thống điều khiển thông minh (intelligent control systems) các thuật toán tiến hóa là công cụ đắc lực. Nhờ các thuật toán tiến hóa, việc xây dựng hệ luật cơ sở của các hệ điều khiển thông minh trở nên dễ dàng và hiệu quả hơn [4]. Việc tìm kiếm quĩ đạo tối ưu cho các rô-bốt tự hành, xây dựng qui trình vận hành tối ưu cho một hệ thống công nghệ v.v. đều có thể giải quyết nhờ các thuật toán tiến hóa.

13. KẾT LUẬN

Bài báo này đã trình bày những nội dung cơ bản nhất về các thuật toán tiến hóa và khả năng ứng dụng của nó. Do phải hạn chế số trang một bài báo nên nhiều chỗ mới chỉ nêu sơ lược, chưa diễn giải chi tiết. Tuy vậy, chúng tôi hy vọng bài báo này có thể giúp ích cho những ai quan tâm tới vấn đề tối ưu hóa nói chung và điều khiển tối ưu nói riêng.

Tác giả xin chân thành cảm ơn Ban biên tạp Tạp chí Tin học và Điều khiển học đã khích lệ và cho những gợi ý bổ ích về công trình nghiên cứu này.

TÀI LIỆU THAM KHẢO

[1] Beveridge S. G., Schechter R. S., *Optimization: Theory and Practice*, McGraw-Hill Book Company, NewYork, St. Louis, San Francisco, London, Sydney, Toronto, 1994.

[2] Bhuyan J. N., Raghavan V. V., Elayavalli V. K., Genetic Algorithm for Clustering with an Ordered Representation, *Proc 4th Int. Conf. Genetic Algorithms*, San Mateo, CA: Morgan Kaufman, 1991.

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

[4] Cordon O., Herrera F., A two-stage evolutionary process for designing TSK Fuzzy rule-based system, *IEEE Trans. on System, Man and Cyb.* **29** (6) (1999).

[5] Davis L., *Handbook of Genetic Algorithms*, New York: Van Nostral Reinhold, 1991.
[6] Fogel D. B., An introduction to simulated evelutionary optimization, *IEEE Trans. Neural Networks* **5** (1) (1994).
[7] Fonseca C. M., Flemming P., Mutli-objective optimization and multiple constraint handling with evolutionary Algorithms, Part I: A unified formulation, *IEEE Trans. On System, Mand Cybernetics* **38** (1) (1998) 26–47.
[8] Goldberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, New York: Addsion-Westley, 1989.
[9] Holland J. H., *Adaptation in Natural and Artificial Systems*, Ann. Arbor MI: Univ. of Michigan Press, 1975.
[10] Holland J. H., Genetic Algorithms and Classifier systems, Foundation and Future Directions, Gen. Alg. and Their Appl., *Proc. 2nd Int. Conf.*, Cambridge, 1987.
[11] Jones D. R., Beltrmo, Solving partitioning problems with genetic algorithms, *Proc. of 4th Int. Conf. Genetic Algorithms*, San Mateo. CA: Morgan Kaufman, 1991.
[12] Krishna K., Murty M. N., Genetic K-mean algorithm, *IEEE Trans. on Sys., Man and Cyber.* **29** (6) (1999).
[13] Krishnapumar K., Goldberg D. E., Control system optimization using genetic algorithms, *J. Guidance Control. Dyn.* **15** (1992).
[14] Kristinsson K., Dumont G. A., System identification and control using genetic algorithms, *IEEE Trans. on Syst., Man and Cyber.* **22** (1992).
[15] Lee J. Y., Cho S. B., Interactive Genetic Algorithm with Wavelet Coefficients for Emotional Image Retrieval, Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, Vol. 2, Singapore, New Jersey, London, Hong Kong, 1995.
[16] Levy-Vehel J., Optimization of Fractal Functions Using Genetic Algorithms,*INRIA Research Report*, No. 1941 (1993).
[17] Marco N. et al., A genetic algorithm compared with a Gradient-Based method for the solution of an Active-Control model Problem, *INRIA Research Report*, No. 2948 (1996).
[18] Michalewicz Z., Krawezyk J. B., A modified genetic algorithm for optimal control problems, *Compt. Math. Appl.* **23** (1992).
[19] Michalewics Z., *Genetic Alorithms + Data Structures = Evolution Programs*, New York: Springer Verlag, 1992.
[20] Nakaya N., Kanasugi A., Kondo K., A Reconfiguration Method of WSI Circuits Using Evolutionary Algorithm, Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, Vol. 2., Singapore, New Jersey, London, Hong Kong, 1995.
[21] Patridis V., Kazarlis S., Bakirtzis A., Varying fitness functions in genetic algorithm Constrained optimization, *IEEE Trans. on Sys., Man and Cybernetics* **28** (5) (1998).
[22] Rao S. S., *Optimization, Theory and Applications*, Wiley Eastern Ltd. New Dehli, 1997.
[23] Rudolph G., Convergence analysis of canonical genetic algorithms, *IEEE Trans. Neural Networks* **5** (1) (1994).
[24] Streifel R. J., Marks R. J., Choi J. J., Healey M., Dynamic fuzzy control of genetic algorithm parameter coding, *IEEE Trans. on Sys., Man and Cyb.* **29** (3) (1999).
[25] Vũ Ngọc Phàn, H_∞-optimal controller design using genetic algorithms, *Journal of Computer Science and Cybernetics* **15** (3) (1999).
[26] Vũ Ngọc Phàn, Ứng dụng thuật toán tiến hóa giải bài toán tối ưu đa mục tiêu, *Tạp chí Tin học và Điều khiển học* **16** (3) (2000) 17–22.
[27] Vũ Ngọc Phàn, Vũ Như Lân, Model Reduction for Robust Control by Using Genetic Algorithms (will be published).

Nhận bài ngày 4 tháng 12 năm 2000
Nhận bài sau khi sửa ngày 28 tháng 4 năm 2001

Viện Công nghệ thông tin