Ts. **Nguyễn Đình Thúc** (chủ biên)

MK.PUB

NHÀ SÁCH MINH KHAI M.K.P.U.B

TRÍ TUỆ NHÂN TẠO

Lập trình tiến hóa

Cấu trúc dữ liệu + Thuật giải di truyền = Chương trình tiến hóa

- Thuật giải di truyền
- Tối ưu số
- Tối ưu tổ hợp

NHÀ XUẤT BẢN GIÁO DỤC

MỤC LỤC

MỞ ĐẦU : Cấu trúc dữ liệu + Thuật giải di truyền
= Chương trình tiến hoá 5

Phần 1 : Thuật Giải Di Truyền 11

Chương 1 : Thuật Giải Di Truyền - Các khái niệm cơ bản 13
1.1. Tối ưu hàm một biến 18
1.2. Thế tiến thoái lưỡng nan của tù nhân 24
1.3. Bài toán Người Du Lịch 28
1.4. Thuật giải leo đồi, mô phỏng việc luyện thép và di truyền 31
1.5. Kết luận 37

Chương 2 : Thuật giải di truyền - Cơ chế thực hiện 39

Chương 3 : Thuật giải di truyền - Nguyên lý hoạt động 61

Phần 2 : Tối Ưu Số 77

Chương 4 : Biểu diễn nhiễm sắc thể cho bài toán tối ưu số 79
4.1. Mô tả bài toán 82
4.2. Hai cài đặt thử nghiệm 83
4.3. Thử nghiệm 84
4.4. Hiệu quả về thời gian 90
4.5. Kết luận 91

CHƯƠNG 5 : Bàn thêm về phép đột biến không đồng bộ 93
5.1. Các trường hợp thử nghiệm 94
5.2. Chương trình tiến hóa giải bài toán tối ưu hóa số 97
5.3. Thử nghiệm và kết quả 100
5.4. Chương trình tiến hóa với các phương pháp khác 103
5.5. Kết luận 109

5/289-00 PIK08n0

Chương 6 : Xử lý ràng buộc....113
6.1. Bài toán qui hoạch phi tuyến trên không gian lồi....114
6.2. Tối ưu hàm phi tuyến....126
6.3. Các kỹ thuật khác....130
6.4. Các khả năng khác....135
6.5. GENOCOP III....140

Chương 7 : Bài toán vận tải....143
7.1. Bài toán vận tải tuyến tính....143
7.2. Bài toán vận tải phi tuyến....159

Phần 3 : Tối Ưu Tổ Hợp....179

Chương 8 : Bài toán người du lịch....181

Chương 9 : Các bài toán tối ưu tổ hợp khác....227
9.1. Bài toán lập lịch....227
9.2. Lập thời khoá biểu cho trường học....237
9.3. Phân hoạch đối tượng và đồ thị....239
9.4. Vạch lộ trình cho rôbô di chuyển....248
9.5. Lưu ý....261

PHỤ LỤC....269

Phụ lục 1 : Các chủ đề chọn lọc....271
1. Cơ chế tạo mẫu....272
2. Các đặc trưng của hàm....282
3. Thuật giải di truyền ánh xạ co....286
4. Thuật giải di truyền với kích thước quần thể thay đổi....294
5. Thuật giải di truyền, các ràng buộc và bài toán ba lô....305
6. Những ý kiến khác....318

Phụ lục 2 : Chiến lược tiến hóa và các phương pháp khác....325
1. Tiến hóa của các chiến lược tiến hóa....325
2. So sánh các chiến lược tiến hóa và các thuật giải di truyền 332
3. Tối ưu hóa hàm đa mục tiêu và đa kết quả....338
4. Những chương trình tiến hóa khác....344

Mở đầu

Cấu trúc dữ liệu + Thuật giải di truyền = Chương trình tiến hoá

Thuật ngữ *Chương trình tiến hoá* trong công thức trên là khái niệm dùng để chỉ các chương trình máy tính có sử dụng thuật toán tìm kiếm và tối ưu hóa dựa trên nguyên lý *tiến hóa tự nhiên*. Ta gọi chung các thuật toán như thế là *thuật toán tiến hóa*. Dưới đây là một số thuật toán tiến hóa đã được công bố.

- Qui 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 (số) phần của vấn đề. Dựa vào qui 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 dựa trên một số chiến lược ban đầu, tiến hóa để tạo ra những chiến lược mới phù hợp với môi trường thực tế một cách tốt nhất.
- Thuật giải di truyền do D.E. Goldberg đề xuất, được L. Davis và Z. Michalevicz phát triển. Đây là thuật toán tiến hóa chính chúng tôi đề cập trong cuốn sách này.

Thuật giải di truyền, cũng như các thuật toán tiến hóa nói chung, hình thành dựa trên 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 được 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 quá trình tiến hóa tự nhiên, các thế hệ mới 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 cũng tác động trở lại góp phần làm thay đổi môi trường.

Các cá thể mới sinh ra trong quá trình tiến hóa nhờ sự lai ghép ở thế hệ cha-mẹ. Một cá thể mới có thể mang những tính trạng của cha-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 trình tiến hóa, dù rằng 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 biệt, nhưng đều mô phỏng bốn quá trình cơ bản: *lai ghép, đột biến, sinh sản* và *chọn lọc tự nhiên*.

Quá trình lai ghép (phép lai)

Phép lai 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 gen của hai (hay nhiều) nhiễm sắc thể cha-mẹ với nhau. Phép lai xảy ra với xác suất p_c, có thể mô phỏng như sau:

- Chọn ngẫu nhiên hai (hay nhiều) cá thể bất kỳ trong quần thể. Giả sử các nhiễm sắc thể của cha-mẹ đều có m gen.
- Tạo một số ngẫu nhiên trong khoảng từ *1* đến *m-1* (ta gọi là điểm lai). Điểm lai chia các chuỗi cha-mẹ dài m thành hai nhóm chuỗi con dài m_1 và m_2. Hai chuỗi nhiễm sắc thể con 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.

Quá trình đột biến (phép đột biến)

Đột biến là hiện tượng cá thể con mang một (số) tính trạng không có trong mã di truyền của cha-mẹ. Phép đột biến xảy ra với xác suất p_m, nhỏ hơn rất nhiều so với xác suất lai p_c. Phép đột biến có thể mô phỏng như sau:

- Chọn ngẫu nhiên một cá thể bất kỳ cha-mẹ trong quần thể.
- Tạo một số ngẫu nhiên k trong khoảng từ *1* đến m, $1 \le k \le m$.
- Thay đổi gen thứ k và trả cá thể này về quần thể để tham gia quá trình tiến hóa tiếp theo.

Quá trình sinh sản và chọn lọc (phép tái sinh và phép chọn)

Phép tái sinh là quá trình trong đó các cá thể được sao chép trên cơ sở độ thích nghi của nó. Độ thích nghi là một hàm gán một giá trị thực cho các cá thể trong quần thể. Quá trình này có thể được mô phỏng như sau:

- Tính độ thích nghi của từng cá thể trong quần thể hiện hành, lập bảng cộng dồn các giá trị thích nghi (theo số thứ tự gán cho từng cá thể). Giả sử quần thể có n cá thể. Gọi độ thích nghi của cá thể thứ i là F_i, tổng dồn thứ i là F_{ti}, tổng độ thích nghi của toàn quần thể là F_m.
- Tạo một số ngẫu nhiên F trong đoạn từ *0* đến F_m.
- Chọn cá thể thứ k đầu tiên thỏa $F \ge F_{tk}$ đưa vào quần thể của thế hệ mới.

Phép chọn là quá trình loại bỏ các cá thể xấu trong quần thể để chỉ giữ lại trong quần thể các cá thể tốt. Phép chọn có thể được mô phỏng như sau:

- Sắp xếp quần thể theo thứ tự độ thích nghi giảm dần.
- Loại bỏ các cá thể cuối dãy để chỉ giữ lại n cá thể tốt nhất. Ở đây, ta giả sử quần thể có kích thước cố định n.

Một thuật giải di truyền, giải một bài toán được cho phải có năm thành phần sau:

- Một cấu trúc dữ liệu I biểu diễn không gian lời giải của bài toán,
- Phương pháp khởi tạo quần thể ban đầu **P(0)**,
- Hàm định nghĩa độ thích nghi **eval(.)** đóng vai trò môi trường,
- Các phép toán di truyền như đã mô phỏng trên,
- Và các tham số thuật giải di truyền sử dụng (kích thước quần thể, xác suất lai, đột biến,...)

Hình 0.1. trình bày một cấu trúc thuật giải di truyền tổng quát.

Thuật giải di truyền

Bắt đầu

t = 0;

Khởi tạo P(t);

Tính độ thích nghi cho các cá thể thuộc P(t);

Khi (điều kiện dừng chưa thoả) **lặp**

t = t + 1;

Tái sinh P'(t) từ P(t)

Lai Q(t) từ P(t-1);

Đột biến R(t) từ P(t-1);

Chọn lọc P(t) từ P(t-1) $\cup$ Q(t) $\cup$ R(t) $\cup$ P(t);

Hết lặp

Kết thúc

Hình 0.1. Một thuật giải di truyền

Trong lập trình tiến hóa, khi giải một bài toán đặt ra, cần tận dụng tối đa tri thức về bài toán đó để chương trình tiến hóa đạt được hiệu quả cao nhất có thể. Việc tận dụng tri thức bài toán có thể được thể hiện (1) qua việc xây dựng một cấu trúc dữ liệu hợp lý sao cho việc xây dựng các phép toán di truyền được tự nhiên và hiệu quả nhất (2) hay qua việc sử dụng phương pháp đã và đang được sử dụng để giải bài toán này và kết hợp chúng với thuật giải di truyền (3) và cách tận dụng hay nhất là tận dụng cả 2 cách trên trong 1 chương trình tiến hóa; đây là cách được nhiều nhà nghiên cứu ứng dụng lập trình tiến hóa sử dụng nhất. Tuy nhiên, để không bị phân tán, chúng tôi chỉ tập trung vào cách thứ nhất: tận dụng tri thức để biểu diễn cấu trúc dữ liệu và xây dựng các phép di truyền. Hy vọng, khi

đã nắm vững các kỹ thuật cơ bản của lập trình tiến hóa được trình bày trong cuốn sách này, bạn có thể tận dụng tri thức bài toán theo cách (2) và (3) để giải bài toán của bạn.

Nội dung của cuốn sách được xây dựng trên cơ sở hai tài liệu chính: cuốn Evolutionary Algorithms in Theory and Practice của Thomas Back (1996) và cuốn Genetic Algorithms + Data Structures = Evolution Programs của Zbigniew Michalewicz (1999) và được chia làm ba phần chính và hai phụ lục:

- **Phần 1: Thuật giải di truyền.** Phần này trình bày chi tiết về thuật giải di truyền cũng như nguyên lý và cơ chế hoạt động của nó.
- **Phần 2: Tối ưu số.** Phần này trình bày cách áp dụng thuật giải di truyền giải các bài toán tối ưu số. Chúng tôi cũng trình bày một số phương pháp giải quyết các ràng buộc của chương trình tiến hóa trong một bài toán qui hoạch phi tuyến tổng quát.
- **Phần 3: Tối ưu tổ hợp.** Các bài toán tổ hợp thuộc lớp bài toán NP-đủ được xem xét và giải quyết bằng thuật giải di truyền sẽ được trình bày và phân tích trong phần này.

Cuối cùng, hai phụ lục sẽ trình bày các khía cạnh lý thuyết cũng như những khía cạnh ứng dụng liên quan đến thuật giải di truyền gốc và các thuật giải di truyền cải tiến. Ở đây, chúng tôi cũng trình bày tóm tắt một số thuật toán tiến hóa khác.

Phần 1

THUẬT GIẢI DI TRUYỀN

Chương 1

THUẬT GIẢI DI TRUYỀN : CÁC KHÁI NIỆM CƠ BẢN

Với khả năng hiện nay, máy tính đã giúp giải được rất nhiều bài toán khó mà trước kia thường bó tay. Mặc dù vậy, vẫn còn một số lớn các bài toán rất thú vị nhưng chưa có thuật giải hợp lý để giải chúng. Trong số đó, các bài toán tối ưu là những bài toán thường xuyên gặp phải trong các ứng dụng thực tiễn.

Trong thực tiễn, có nhiều bài toán tối ưu quan trọng đòi hỏi những thuật giải chất lượng cao. Ví dụ, ta có thể áp dụng phương pháp *mô phỏng luyện thép* để giải bài toán tìm đường đi ngắn nhất cho xe cứu hoả hay bài toán người du lịch.... Cũng có nhiều bài toán tối ưu tổ hợp (trong đó có nhiều bài đã được chứng minh là thuộc loại *NP - đủ*) có thể được giải gần đúng trên máy tính hiện đại bằng kỹ thuật Monte-Carlo.

Nói chung, bài toán tối ưu có thể được xem như bài toán tìm kiếm giải pháp (tốt nhất) trong không gian (vô cùng lớn) các giải pháp. Khi không gian tìm kiếm nhỏ, các phương pháp cổ điển như trên cũng đủ thích hợp; nhưng khi không gian lớn cần phải dùng đến những kỹ thuật *Trí Tuệ Nhân Tạo* đặc biệt. *Thuật giải Di Truyền (GA)* là một trong những kỹ thuật đó. GA là một loại thuật giải mô phỏng các hiện tượng tự nhiên: *kế thừa* và *đấu tranh sinh tồn* để cải tiến lời giải và khảo sát không gian lời giải. Khái niệm kế thừa và đấu tranh sinh tồn được giải thích qua thí dụ về sự tiến hóa của một quần thể thỏ như sau:

Có một quần thể thỏ. Trong số đó có một số con nhanh nhẹn và thông minh hơn những con khác. Những chú thỏ nhanh nhẹn và

thông minh có xác suất bị chồn cáo ăn thịt nhỏ hơn, do đó chúng tồn tại để làm những gì tốt nhất có thể: Tạo thêm nhiều thỏ tốt. Dĩ nhiên, một số thỏ chậm chạp dần độn cũng sống chỉ vì may mắn. Quần thể những chú thỏ còn sống sót sẽ bắt đầu sinh sản. Việc sinh sản này sẽ tạo ra một hỗn hợp tốt về "nguyên liệu di truyền thỏ": Một số thỏ chậm chạp có con với những con thỏ nhanh, một số thỏ nhanh với thỏ nhanh, một số thỏ thông minh với thỏ dần độn, vv... Và trên tất cả, thiên nhiên thỉnh thoảng lại ném vào một con thỏ 'hoang dã' bằng cách làm đột biến nguyên liệu di truyền thỏ. Những chú thỏ con, do kết quả này, sẽ nhanh hơn và thông minh hơn những con trong quần thể gốc vì có nhiều bố mẹ nhanh nhẹn và thông minh hơn đã thoát chết khỏi chồn cáo. (Thật hay là những con chồn cũng trải qua những tiến trình tương tự - nếu không những con thỏ sẽ trở nên nhanh và thông minh đến nỗi những con chồn không thể bắt chúng được).

Khi tìm kiếm lời giải tối ưu, thuật giải di truyền cũng thực hiện các bước tương ứng với câu chuyện đấu tranh sinh tồn của loài thỏ.

Thuật giải di truyền sử dụng các thuật ngữ vay mượn của di truyền học. Ta có thể nói về các cá thể (hay *kiểu gen*, *cấu trúc*), trong một quần thể; những cá thể này cũng còn được gọi là các *chuỗi* hay các *nhiễm sắc thể*. Điều này có thể gây chút lẫn lộn: mỗi tế bào của một cơ thể của một chủng loại đã cho, mang một số những nhiễm sắc thể nào đó (thí dụ, người có 46 nhiễm sắc thể) nhưng trong thuật giải di truyền, ta chỉ nói về những cá thể có một nhiễm sắc thể. Các nhiễm sắc thể được tạo thành từ các đơn vị - *các gen* – biểu diễn trong một chuỗi tuyến tính; mỗi gen kiểm soát một (số) đặc trưng. Gen với những đặc trưng nhất định có vị trí nhất định trong nhiễm sắc thể. Bất cứ đặc trưng nào của mỗi cá thể có thể tự biểu hiện một cách phân biệt; và gen có thể nhận một số giá trị khác nhau (các giá trị về tính năng).

Mỗi kiểu (nhóm) gen (ta gọi là một nhiễm sắc thể) sẽ biểu diễn một lời giải của bài toán đang giải (ý nghĩa của một nhiễm sắc thể cụ thể được người sử dụng xác định trước); một tiến trình tiến hóa được thực hiện trên một quần thể các nhiễm sắc thể tương ứng với một quá trình tìm kiếm lời giải trong không gian lời giải. Tìm kiếm đó cần cân đối hai mục tiêu (có vẻ mâu thuẫn nhau): Khai thác những lời giải tốt nhất và khảo sát không gian tìm kiếm. *Leo đồi* là một thí dụ về chiến lược cho phép khai thác và cải thiện lời giải tốt nhất hiện hành; nhưng, leo đồi lại bỏ qua việc khảo sát không gian tìm kiếm. Ngược lại, *tìm kiếm ngẫu nhiên* là một thí dụ điển hình của chiến lược khảo sát không gian tìm kiếm mà không chú ý đến việc khai thác những vùng đầy hứa hẹn của không gian. *Thuật giải di truyền (GA)* là phương pháp tìm kiếm (độc lập miền) tạo được sự cân đối đáng kể giữa việc khai thác và khảo sát không gian tìm kiếm.

Thực ra, GA thuộc lớp các thuật giải xác suất, nhưng lại rất khác những thuật giải ngẫu nhiên vì chúng kết hợp các phần tử tìm kiếm trực tiếp và ngẫu nhiên. Khác biệt quan trọng giữa tìm kiếm của GA và các phương pháp tìm kiếm khác là GA duy trì và xử lý một tập các lời giải (ta gọi là một quần thể) - tất cả những phương pháp khác chỉ xử lý một điểm trong không gian tìm kiếm. Chính vì thế, GA mạnh hơn các phương pháp tìm kiếm hiện có rất nhiều.

Đơn cử, ta so sánh GA với hai phương pháp tìm kiếm hiện được sử dụng rộng rãi: Leo đồi và Mô phỏng luyện thép.

Phương pháp leo đồi dùng kỹ thuật lặp và áp dụng cho một điểm duy nhất (điểm hiện hành trong không gian tìm kiếm). Trong mỗi bước lặp, một điểm mới được chọn từ lân cận của điểm hiện hành (vì thế leo đồi còn được gọi là phương pháp tìm kiếm lân cận hay tìm kiếm cục bộ). Nếu điểm mới cho giá trị (của hàm mục tiêu) tốt hơn, điểm mới sẽ trở thành điểm hiện hành. Nếu không, một lân

cận khác sẽ được chọn và thử. Quá trình trên sẽ dừng nếu không cải thiện thêm được cho lời giải hiện hành.

Rõ ràng là phương pháp leo đồi chỉ cung cấp các giá trị tối ưu cục bộ và những giá trị này phụ thuộc rất nhiều vào điểm khởi đầu. Hơn nữa, không có thông tin sẵn có về sai số tương đối (thỏa tối ưu toàn cục) của lời giải tìm được.

Để tăng cơ hội thành công, phương pháp leo đồi thường được thực hiện nhiều lần; mỗi lần với một điểm khởi đầu khác nhau (những điểm này không cần chọn ngẫu nhiên - một tập hợp các điểm khởi đầu của một lần thực thi phụ thuộc vào kết quả của những lần chạy trước đó).

Kỹ thuật mô phỏng luyện thép là một kỹ thuật khắc phục những bất lợi của phương pháp leo đồi: Lời giải không còn tùy thuộc nhiều vào điểm khởi đầu nữa và (thường là) gần với điểm tối ưu. Đạt được điều này là nhờ đưa vào xác suất nhận p. Xác suất p là hàm theo giá trị của hàm mục tiêu đối với điểm hiện hành và điểm mới, và một tham số điều khiển bổ sung, tham số "nhiệt độ" T. Nói chung, nhiệt độ T càng thấp thì cơ hội nhận điểm mới càng nhỏ. Khi thực hiện thuật giải, nhiệt độ T của hệ thống sẽ được hạ thấp dần theo từng bước. Thuật giải dừng khi T nhỏ hơn một ngưỡng cho trước; với ngưỡng này thì gần như không còn thay đổi nào được chấp nhận nữa.

Như đã đề cập, GA thực hiện tiến trình tìm kiếm lời giải tối ưu theo nhiều hướng, bằng cách duy trì một *quần thể* các lời giải, và thúc đẩy sự thành hình và trao đổi thông tin giữa các hướng này. Quần thể trải qua tiến trình trình tiến hóa: ở mỗi *thế hệ* lại *tái sinh* các lời giải tương đối "tốt", trong khi các lời giải tương đối "xấu" thì chết đi. Để phân biệt các lời giải khác nhau, hàm mục tiêu được dùng để đóng vai trò môi trường.

Cấu trúc của một thuật giải di truyền đơn giản tương tự với cấu trúc của bất kỳ chương trình tiến hóa nào (xem hình 0.1, phần mở đầu). Ở bước lặp t, thuật giải di truyền duy trì một quần thể các lời giải (các nhiễm sắc thể, các vectơ), $P(t) = \{x_1^t, ..., x_n^t\}$. Mỗi lời giải x_i^t được lượng giá để biết được độ "thích nghi" của nó. Rồi một quần thể mới (lần lặp thứ $t+1$) được hình thành bằng cách chọn giữ lại những cá thể thích nghi nhất. Một số cá thể của quần thể này trải qua những biến đổi nhờ *lai tạo (phép lai)* và *đột biến (phép đột biến)*, hình thành nên những lời giải mới. *Phép Lai* kết hợp các tính chất của hai nhiễm sắc thể 'cha' và 'mẹ' để tạo ra các nhiễm sắc thể 'con' bằng cách hoán vị các đoạn gen tương ứng của cha và mẹ. Thí dụ, nếu cha mẹ được biểu diễn bằng vectơ 5 chiều (a_1,b_1,c_1,d_1,e_1) và (a_1,b_2,c_2,d_2,e_2), thì lai tạo, hoán vị tại vị trí thứ 2, sẽ sinh ra các nhiễm sắc thể con (a_1,b_1,c_2,d_2,e_2) và (a_2,b_2,c_1,d_1,e_1). Phép lai cho phép trao đổi thông tin giữa các lời giải.

Khác với phép lai, phép đột biến thay đổi một cách ngẫu nhiên một hay nhiều gen của nhiễm sắc thể được chọn, thay đổi này được thực hiện với một xác suất thể hiện tốc độ đột biến. Phép đột biến cho phép đưa thêm các thông tin mới vào quần thể làm cho chất liệu di truyền phong phú thêm.

Một thuật giải di truyền (hay một chương trình tiến hóa bất kỳ) giải một bài toán cụ thể phải gồm năm thành phần sau đây:

- Cách biểu diễn di truyền cho lời giải của bài toán;
- Cách khởi tạo quần thể ban đầu;
- Một hàm lượng giá đóng vai trò môi trường, đánh giá các lời giải theo mức độ "thích nghi" của chúng;
- Các phép toán di truyền;

- Các tham số khác (kích thước quần thể, xác suất áp dụng các phép toán di truyền vv...)

Để dễ hình dung, chúng tôi sẽ thảo luận các tính năng chính của thuật giải di truyền qua ba thí dụ cụ thể. Trong thí dụ thứ nhất, ta áp dụng thuật giải di truyền tìm giá trị lớn nhất của một hàm thực một biến. Thí dụ thứ hai minh họa cách dùng một thuật giải di truyền để học một chiến lược của một trò chơi đơn giản. Thí dụ 3 bàn về một ứng dụng của thuật giải di truyền để tiếp cận một bài toán tổ hợp NP-đầy đủ, bài toán người du lịch.

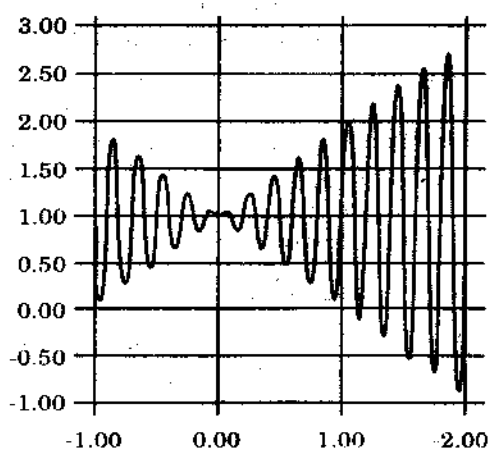
1.1. Tối ưu hàm một biến

Xét bài toán tối ưu không ràng buộc sau:

*Max f(x) = x * sin (10π * x) +1.0; x ∈ [-1,2].*

Hình 1.1 là đồ thị của *f*. Bài toán có nghĩa là tìm *x* trong khoảng [-1,2] để *f* có giá trị lớn nhất, nghĩa là tìm x_o sao cho:

$f(x_o) \geq f(x), \quad \forall x \in [-1,2]$



Hình 1.1. Đồ thị hàm $f(x) = x \times sin\ (10\pi \times x) + 1.0$

Khi đạo hàm bậc nhất bằng 0, nghĩa là,

$f'(x) = sin\ (10\pi * x) + 10\pi x * \cos(10\pi * x) = 0$

$\Leftrightarrow tang\ (10\pi * x) = -10\pi x.$

Rõ ràng là phương trình trên có vô số lời giải,

$$x_i = \frac{2i-1}{20} + \varepsilon_i, \quad I = 1, 2, ..$$

$x_0 = 0$

$$x_i = \frac{2i+1}{20} - \varepsilon_i \quad i = -1, -2, ...$$

trong đó các số hạng ε_i là các dãy số thực giảm (i = 1, 2, ..., và *i*= −1, −2, ...) dần về 0.

Cũng chú ý rằng hàm *f* đạt đến cực đại (cục bộ) tại điểm x_i, khi *i* là số nguyên lẻ, và đạt đến cực tiểu của nó tại x_i, khi *i* chẵn (xem hình 1.1).

Vì miền giá trị của bài toán là [-1,2], hàm đạt cực đại tại $x_{19} = \frac{37}{20} + \varepsilon_{19} = 1.85 + \varepsilon_{19}$, ở đây $f(x_{19})$ hơi lớn hơn *f(1.85)* = 1.85 * sin (18π + π/2) + 1.0 = 2.85.

Bây giờ ta dùng thuật giải di truyền để giải bài toán trên, nghĩa là, tìm một điểm trong đoạn [-1, 2] sao cho tại đó *f* có giá trị lớn nhất.

Ta sẽ lần lượt bàn về 5 thành phần chính của thuật giải di truyền giải bài toán này.

1.1.1. Biểu diễn

Ta sử dụng một vectơ nhị phân làm nhiễm sắc thể để biểu diễn các giá trị thực của biến x. Chiều dài vectơ phụ thuộc vào độ chính xác cần có, trong thí dụ này, ta tính chính xác đến 6 số lẻ.

Miền giá trị của x có chiều dài $2 - (-1) = 3$; với yêu cầu về độ chính xác 6 số lẻ như thế phải chia khoảng [-1, 2] thành ít nhất 3×10^6 khoảng có kích thước bằng nhau. Điều này có nghĩa là cần có 22 bit cho vectơ nhị phân (nhiễm sắc thể):

$$2097152 = 2^{21} < 3000000 \le 2^{22} = 4194304$$

Ánh xạ biến chuỗi nhị phân $(b_{21}b_{20}...b_0)$ thành số thực x trong khoảng [-1, 2] được thực hiện qua hai bước như sau:

- đổi chuỗi nhị phân $(b_{21}b_{20}...b_0)$ từ cơ số 2 sang cơ số 10:

$$(< b_{21}b_{20}...b_0 >)_2 = \left(\sum_{i=0}^{21} b_i 2^i\right)_{10} = x'$$

- tìm số thực x tương ứng

$$x = -1 + x'.\frac{3}{2^{22}-1}$$

với -1 là cận dưới của miền giá trị và 3 là chiều dài của miền.

Thí dụ, nhiễm sắc thể (1000101110110101000111) biểu diễn số 0.637197 vì

$x' = (1000101110110101000111)_2 = 2288967_{10}$

và $x = -1.0 + 2288967 \times 3/4194303 = 0.637197$

Đương nhiên nhiễm sắc thể

(0000000000000000000000) và (1111111111111111111111) biểu diễn các cận của miền, -1.0 và 2.0 cho mỗi cận.

1.1.2. Khởi tạo quần thể

Tiến trình khởi tạo rất đơn giản: Ta tạo một quần thể các nhiễm sắc thể, trong đó mỗi nhiễm sắc thể là một vectơ nhị phân 22 bit, tất cả 22 bit của mỗi nhiễm sắc thể đều được khởi tạo ngẫu nhiên.

1.1.3. Hàm lượng giá

Hàm lượng giá *eval* của các vectơ nhị phân v chính là hàm f:

$$eval(v) = f(x)$$

trong đó, nhiễm sắc thể v biểu diễn giá trị thực x như đã nói ở trên, hàm lượng giá đóng vai trò môi trường, đánh giá từng lời giải theo độ thích nghi của chúng. Thí dụ, 3 nhiễm sắc thể:

v_1= (1000101110110101000111),

v_2 =(0000001110000000010000),

v_3= (1110000000111111000101),

tương ứng với các giá trị $x_1 = 0.637197$, $x_2 = -0.958973$, và $x_3 = 1.627888$. Và có độ thích nghi tương ứng:

$eval(v_1) = f(x_1) = 1.586345$,

$eval(v_2) = f(x_2) = 0.078878$,

$eval(v_3) = f(x_3) = 2.250650$.

Rõ ràng, nhiễm sắc thể v_3 là tốt nhất trong 3 nhiễm sắc thể này, vì hàm lượng giá nó trả về giá trị cao nhất.

1.1.4. Các phép toán di truyền

Trong giai đoạn tiến hoá quần thể, ta có thể dùng 2 phép toán di truyền cổ điển: *đột biến* và *lai*.

Như đã trình bày ở trên, đột biến làm thay đổi một (số) gen (các vị trí trong một nhiễm sắc thể) với xác suất bằng tốc độ đột biến. Giả định rằng gen thứ 5 trong nhiễm sắc thể v_3 được chọn để đột biến. Và đột biến chính là thay đổi giá trị gen này: 0 thành 1 và 1 thành 0. Như vậy, sau đột biến này, v_3 sẽ là:

v'_3= (11101000001111111000101)

Nhiễm sắc thể này biểu diễn giá trị x'_3= 1.721638 và $f(x'_3)$= -0.082257. Điều này có nghĩa là đột biến cụ thể này làm giảm khá nhiều giá trị của nhiễm sắc thể v_3 . Bây giờ, nếu gen thứ 10 được chọn để đột biến trong nhiễm sắc thể v_3 thì

v''_3 = (11100000011111111000101) .

Giá trị tương ứng x''_3=1.630818 và $f(x''_3)$= 2.343555, khá hơn giá trị $f(x_3)$ = 2.250650.

Ta sẽ minh họa phép lai trên các nhiễm sắc thể v_2 và v_3. Giả định rằng điểm lai được chọn (ngẫu nhiên) ở vị trí thứ 6:

v_2 = (00000|1110000000010000),

v_3 = (11100|00000111111000101).

Hai con của kết quả lai là

v'_2= (00000|00000111111000101)

v'_3= (11100|01110000000010000).

Các con này có độ thích nghi:

$f(v'_2) = f(-0.998113) = 0.940865$,

$f(v'_3) = f(1.666028) = 2.459245$

Chú ý rằng con thứ 2 thích nghi hơn cả cha lẫn mẹ của nó.

1.1.5. Các tham số

Đối với bài toán đặc biệt này, ta đã dùng các tham số sau đây: kích thước quần thể *pop-size* = 50, xác suất lai tạo p_c =0.25, xác suất đột biến p_m= 0.01. Xác suất lại p_c = 0.25 nghĩa là cá thể v trong quần thể có 25% cơ hội được chọn để thực hiện phép lai; còn xác suất đột biến p_m = 0.01 lại là 1% 1 bít tất bất kỳ của 1 cá thể bất kỳ trong quần thể bị đột biến.

1.1.6. Các kết quả thử nghiệm

Bảng 1.1 trình bày một số kết quả hàm mục tiêu f ở 1 số thế hệ. Cột bên trái cho biết thế hệ được xem xét, và cột bên phải cho biết giá trị của hàm f. Nhiễm sắc thể tốt nhất sau 150 thế hệ là

v_{max} = (11100110100010000010 1),

tương ứng với giá trị x_{max} = 1.850773.

Đúng như ta mong đợi, $x_{max} = 1.85 + \varepsilon$, và $f(x_{max})$ lớn hơn 2.85 một chút.

Bảng 1.1. Kết quả của 150 thế hệ

Thế hệ thứ	Hàm lượng giá
1	1.441942
6	2.250003
8	2.250283
9	2.250284
10	2.250363
12	2.328077
39	2.344251
40	2.345087
51	2.738930
99	2.849246
137	2.850217
145	2.850227

1.2. Thế tiến thoái lưỡng nan của tù nhân

Ví dụ này minh họa cách áp dụng thuật giải di truyền để học chiến lược của một trò chơi đơn giản: trò tiến thoái lưỡng nan của tù nhân. Bài toán được mô tả như sau,

Hai tù nhân bị giam trong hai xà lim riêng biệt, không thể thông tin cho nhau. Mỗi tù nhân được yêu cầu là hãy ly khai và phản bội người tù kia, một cách độc lập. Nếu chỉ có một người ly khai, anh ta được thưởng, còn người kia bị phạt. Nếu cả hai đều ly khai, cả hai đều bị giam lại và tra tấn. Nếu không ai ly khai, cả hai đều được phần thưởng khá. Như vậy, cách chọn ích kỷ là ly khai sẽ luôn đem lại phần thưởng hoặc hình phạt cao hơn, bất chấp người kia chọn ra sao - nhưng nếu cả hai đều ly khai thì họ đều làm những điều tệ hại hơn là hợp tác với nhau. Tình thế khó khăn của người tù là phải quyết định nên ly khai hay hợp tác với người kia.

Trò chơi này gồm hai người chơi, trong trò chơi này, tới phiên mình, mỗi người sẽ chọn ly khai hoặc hợp tác với người kia. Các đấu thủ sẽ được ghi điểm theo cách thưởng phạt được liệt kê trong bảng 1.2.

Bảng 1.2. Bảng thưởng phạt của trò chơi "Tiến thoái lưỡng nan của tù nhân" : P_i là sự thưởng phạt dành cho người chơi *I*.

Đấu thủ 1	Đấu thủ 2	P_1	P_2	Lời bình
Ly khai	Ly khai	1	1	Phạt vì ly khai lẫn nhau
Ly khai	Hợp tác	5	0	Cám dỗ ly khai và sự thưởng phạt
Hợp tác	Ly khai	0	5	Sự thưởng phạt và cám dỗ ly khai
Hợp tác	Hợp tác	3	3	Phần thưởng cho sự hợp tác

Ta sẽ xem cách thuật giải di truyền được sử dụng để học chiến lược của trò chơi tiến thoái lưỡng nan này. Tiếp cận bằng GA nhằm duy trì quần thể các "đấu thủ", mỗi cá thể có chiến lược của riêng nó. Ban đầu, chiến lược của mỗi đấu thủ được chọn ngẫu nhiên. Sau đó, ở mỗi bước, mỗi đấu thủ chơi và được ghi điểm. Rồi có một số đấu thủ được chọn cho thế hệ mới, và một số trong đó được chọn để ghép đôi. Khi hai đấu thủ được ghép đôi, đấu thủ mới được tạo ra có chiến lược

thừa hưởng từ những chiến lược của cha mẹ (lai tạo). Một đột biến, như thường lệ, dẫn tới có biến đổi trong chiến lược của các đối thủ do các thay đổi ngẫu nhiên trên biểu diễn của các chiến lược này.

1.2.1. Biểu diễn chiến lược

Trước hết, ta cần một số cách biểu diễn về chiến lược (nghĩa là, một lời giải khả thi). Để đơn giản, ta sẽ quan tâm đến các chiến lược tất định và sử dụng kết quả của chuyển dịch của 3 bước trước để quyết định bước hiện tại. Vì có 4 khả năng có thể có cho mỗi bước, nên tất cả có $4 \times 4 \times 4 = 64$ khả năng chuyển dịch của 3 bước trước đó.

Có thể đặc tả chiến lược theo kiểu này bằng cách chỉ định chuyển dịch nào cần được thực hiện cho các khả năng có thể xảy ra. Như thế, một chiến lược có thể được biểu diễn bằng chuỗi 64 bit (mỗi bít nhận giá trị Ds -ly khai- hoặc Cs -hợp tác-), chỉ định cho chuyển dịch nào phải làm đối với 64 khả năng. Để có chiến lược khởi điểm ở đầu một chuyển dịch, chúng ta cũng cần đặc tả tiền đề về 3 chuyển dịch giả định trước khi bắt đầu trò chơi. Điều này đòi hỏi phải có hơn 6 gen, như vậy toàn bộ nhiễm sắc thể ít nhất cần 70 bít.

Chuỗi 70 bit đặc tả những gì một đấu thủ trong từng trường hợp có thể xảy ra và như thế xác định hoàn toàn một chiến lược riêng. Chuỗi 70 gen chính là một nhiễm sắc thể của đấu thủ để dùng trong tiến trình tiến hóa.

1.2.2. Thuật giải di truyền

Thuật giải di truyền dùng để học chiến lược tiến thoái lưỡng nan của tù nhân gồm 4 giai đoạn, như sau:

1. *Chọn quần thể khởi đầu*. Mỗi đấu thủ được gán ngẫu nhiên một chuỗi 70 bit, biểu diễn cho một chiến lược như đã trình bày ở trên.

2. *Kiểm tra mỗi đấu thủ để quyết định hiệu quả đạt được*. Mỗi đấu thủ dùng một chiến lược được xác định trong nhiễm sắc thể của nó để đấu với những đấu thủ khác. Điểm của đấu thủ là điểm trung bình của tất cả các bước.

3. *Chọn các đấu thủ để phát sinh*. Nếu đấu thủ có điểm trung bình sẽ được cho một điểm cộng. Khi điểm có độ lệch chuẩn trên trung bình sẽ được cho hai điểm cộng; còn khi điểm có độ lệch chuẩn dưới trung bình sẽ không được điểm cộng nào.

4. *Những đấu thủ chơi thành công sẽ được kết đôi một cách ngẫu nhiên để sinh con qua mỗi lần ghép đôi*. Chiến lược của mỗi con sẽ được quyết định từ những chiến lược của cha mẹ. Điều này được thực hiện bằng cách sử dụng hai phép toán di truyền: *lai* và *đột biến* như trong ví dụ trước.

Sau 4 giai đoạn này ta có một quần thể mới. Quần thể này sẽ biểu diễn các mẫu ứng xử giống cách cư xử của những cá thể thành công trong thế hệ trước nhiều hơn, ít giống cách cư xử của những cá thể không thành công hơn. Với mỗi thế hệ mới, những cá thể có điểm tương đối cao sẽ có nhiều khả năng truyền các phần chiến lược của chúng, trong khi những cá thể không thành công sẽ không có phần chiến lược nào được truyền lại cho thế hệ sau.

1.2.3. Kết quả thực nghiệm

Khi chạy chương trình này, ta nhận được những kết quả rất đáng lưu ý. Từ một khởi đầu rất ngẫu nhiên, thuật giải di truyền làm tiến hóa các quần thể mà thành viên trung gian cũng thành công như thuật giải dùng các heuristic sau:

1. *Đừng lắc lư thuyền*: tiếp tục hợp tác sau 3 lần tương tác (nghĩa là, C sau (CC)(CC)(CC)).

2. *Khiêu khích*: ly khai khi đấu thủ khác ly khai (nghĩa là, D sau khi nhận (CC)(CC)(CD)).

3. *Nhận lời xin lỗi*: tiếp tục hợp tác sau khi việc hợp tác được phục hồi (nghĩa là, C sau (CD)(DC)(CC)).
4. *Quên*: hợp tác khi sự tương tác đã được phục hồi sau khi bị lợi dụng (nghĩa là, C sau (DC)(CC)(CC)).
5. *Nhận ghép đôi*: ly khai sau 3 lần ly khai lẫn nhau. (nghĩa là, D sau (DD)(DD)(DD)).

1.3. Bài toán Người Du Lịch

Ví dụ cuối cùng chúng tôi muốn trình bày là tiếp cận thuật giải di truyền giải bài toán *Người Du Lịch (TSP)*. Bài toán TSP được mô tả như sau.

Một du khách muốn thăm mọi thành phố anh quan tâm; mỗi thành phố thăm qua đúng một lần; rồi trở về điểm khởi hành. Biết trước chi phí di chuyển giữa hai thành phố bất kỳ. Hãy xây dựng một lộ trình thoả các điều kiện trên với tổng chi phí nhỏ nhất.

TSP là bài toán tối ưu tổ hợp và có rất nhiều ứng dụng. Có thể giải bài toán này bằng nhiều phương pháp: phương pháp nhánh cận, phương pháp gần đúng hay những phương pháp tìm kiếm heuristic. Ta sẽ giải bài toán này bằng thuật giải di truyền.

Như ta đã biết, khó khăn đầu tiên đặt ra cho người thiết kế thuật giải di truyền là *biểu diễn nhiễm sắc thể*. Trong ví dụ thứ nhất, tối ưu hàm một biến, bằng cách chấp nhận một ít sai số, ta đã sử dụng biểu diễn nhị phân để biểu diễn nhiễm sắc thể. Còn trong ví dụ thứ hai, với một chút phân tích, ta cũng đưa được về biểu diễn nhị phân các chiến lược cần thiết. Với cách biểu diễn như thế, các phép di truyền truyền thống, lai và đột biến, được áp dụng trực tiếp không cần sửa đổi gì cả. Khi thực hiện lai hay đột biến, các nhiễm sắc thể kết quả vẫn hợp lệ, nghĩa là vẫn là một lời giải thuộc không gian tìm kiếm. Điều này không còn đúng trong bài toán TSP nữa.

Nếu biểu diễn nhị phân cho bài toán TSP có n thành phố, mỗi thành phố phải được đánh mã bằng một chuỗi $[\log_2(n)]$ bit; và nhiễm sắc thể là một chuỗi gồm $n*[\log_2(n)]$ bit. Đột biến có thể tạo ra một lộ trình không thoả điều kiện bài toán nữa: Ta có thể thăm một thành phố 2 lần. Hơn nữa, đối với một bài toán TSP có 20 thành phố (ta cần 5 bit để biểu diễn một thành phố), có một số chuỗi 5 bit nào đó (như 10101) sẽ không tương ứng với thành phố nào cả vì 5 bit có thể biểu diễn tối đa 32 trường hợp. Phép lai cũng gây ra những vấn đề tương tự. Rõ ràng nếu ta dùng các phép toán đột biến và lai truyền thống như đã định nghĩa trước đây, ta sẽ phải dùng đến một loại "thuật giải sửa chữa", thuật giải "sửa chữa" một nhiễm sắc thể không hợp lệ để đưa nó về không gian tìm kiếm.

Cách tự nhiên là ta sẽ đánh số các thành phố và dùng một vectơ nguyên để biểu diễn một nhiễm sắc thể lộ trình. Cách biểu diễn này giúp ta tránh phải dùng thuật giải sửa chữa bằng cách kết hợp những hiểu biết về bài toán vào các phép toán di truyền. Với cách biểu diễn này, một vectơ các thành phần nguyên $v = <i_1, i_2, ..., i_n>$ biểu diễn một lộ trình: từ i_1 đến i_2 ..., từ i_{n-1} đến i_n, và trở về i_1 (v là một hoán vị của vectơ <1, 2, ..., n>).

Vấn đề thứ hai là *khởi tạo quần thể đầu*. Đối với tiến trình khởi tạo, ta có thể sử một số thuật giải heuristic (chẳng hạn như dùng thuật giải Greedy áp dụng nguyên lý "tham lam" vào bài toán TSP. Ta sẽ áp dụng thuật giải Greedy nhiều lần, mỗi lần từ một thành phố đầu khác nhau); hoặc đơn giản hơn là khởi tạo quần thể bằng cách tạo ngẫu nhiên các mẫu từ các hoán vị của <1, 2, ..., n>.

Việc lượng giá một nhiễm sắc thể rất dễ dàng: cho trước chi phí của chuyến đi giữa các thành phố, ta có thể dễ dàng tính tổng chi phí của trọn lộ trình.

Về các *phép toán di truyền*, trước hết, ta xây dựng phép *lai OX* như sau: cho trước hai cá thể cha-mẹ, cá thể con có được bằng cách

chọn thứ tự lộ trình từ một cá thể và bảo toàn thứ tự tương đối giữa các thành phố trong cá thể kia. Thí dụ, nếu cha-mẹ là:

<1 2 3 4 5 6 7 8 9 10 11 12> và

<7 3 1 11 4 12 5 2 10 9 6 8>

và khúc được chọn là (4 5 6 7), cá thể con của phép lai sẽ là:

<1 11 12 4 5 6 7 2 10 9 8 3>

Như yêu cầu, cá thể con có quan hệ cấu trúc đối với cả hai cá thể cha-mẹ. Vai trò của cha và mẹ có thể hoán đổi khi xây dựng cá thể con thứ hai.

Phép đột biến thực hiện đơn giản hơn: ta chỉ cần hoán vị hai vị trí bất kỳ trong cá thể được chọn.

(Dĩ nhiên đây chỉ là một cách định nghĩa các phép toán di truyền. Bạn đọc có thể nghĩ ra những cách định nghĩa khác hay hơn).

Như thế, nếu cho trước các tham số liên quan, thuật giải di truyền có thể được thi hành.

Những vấn đề về biểu diễn và các phép toán di truyền giải bài toán TSP sẽ được bàn chi tiết hơn trong phần 3 – Tối ưu tổ hợp.

1.4. Thuật giải leo đồi, mô phỏng việc luyện thép và di truyền

Trong phần này ta bàn về ba thuật giải, đó là leo đồi, mô phỏng việc luyện thép và thuật giải di truyền; cùng áp dụng giải bài toán

tối ưu đơn giản. Thí dụ sau đây chứng tỏ khả năng ưu việt của cách tiếp cận GA.

Không gian tìm kiếm là một tập các chuỗi nhị phân v có chiều dài 30. Hàm mục tiêu f là cần cực đại hóa là hàm:

$$f(v) = |11 * one(v) - 150|,$$

trong đó, hàm one(v) cho biết tổng số số 1 có trong chuỗi v. Thí dụ, với ba chuỗi sau đây:

v_1= (11011010111010111111011011011),

v_2= (11100010010011011100101010011),

v_3= (00001000001100100000010001000),

thì

$f(v_1) = |11*22 - 150| = 92,$

$f(v_2) = |11*15 - 150| = 15,$

$f(v_3) = |11*6 - 150| = 84,$

($one(v_1) = 22$; $one(v_2) = 15$ và $one(v_3) = 6$).

f là hàm tuyến tính. Chúng tôi chỉ dùng nó để minh họa ý tưởng chính của ba thuật giải: leo đồi, mô phỏng luyện thép và thuật giải di truyền. Tuy nhiên, cần lưu ý là f có một cực đại toàn cục với

v_g= (11111111111111111111111111111)

khi đó $f(v_g) = |11*30 - 150|$ và một cực đại cục bộ khi

v_l =(00000000000000000000000000000)

$f(v_l) = |11* 0 - 150| = 150$

Có nhiều phiên bản khác nhau của thuật giải leo đồi. Chúng chỉ khác nhau theo cách chuỗi mới được chọn để so sánh với chuỗi hiện tại. Một phiên bản đơn giản (lặp) của leo đồi (*MAX* lần lặp lại) được cho trong hình 1.2. Thoạt đầu, 30 lân cận được quan tâm, và lân cận v_n, trả về giá trị lớn nhất $f(v_n)$, được chọn để cạnh tranh với chuỗi v_c hiện hành. Nếu $f(v_c) < f(v_n)$, thì chuỗi mới trở thành chuỗi hiện hành. Ngược lại thì không có cải thiện cục bộ nào xảy ra: thuật giải đạt đến (cục bộ hoặc toàn cục) tối ưu (cục bộ = TRUE). Trong trường hợp như vậy, lần lặp kế (t ←t+1) được thực hiện với một chuỗi mới, chọn theo ngẫu nhiên.

Đáng chú ý rằng thành công hay thất bại của mỗi lần lặp trong thuật giải leo đồi (nghĩa là trả về tối ưu cục bộ hay toàn cục) tuỳ thuộc chuỗi khởi đầu (được chọn lựa ngẫu nhiên). Rõ ràng là nếu chuỗi khởi đầu có 13 số 1 hoặc ít hơn, thuật giải sẽ luôn luôn chấm dứt tại tối ưu cục bộ (thất bại). Lý do là chuỗi 13 số 1 trả về giá trị 7 của hàm mục tiêu, và bất cứ cải thiện của từng bước lặp đều hướng về tối ưu cục bộ, nghĩa là, tăng số số 1 đến 40, giảm trị số của hàm mục tiêu còn 4. Mặt khác, bất cứ việc giảm lượng số 1 nào cũng sẽ làm tăng trị số của hàm: một chuỗi có 12 số 1 sẽ mang lại giá trị 18 cho hàm, chuỗi 11 số 1 cho trị 29, vv... Điều này có thể đẩy việc tìm kiếm theo hướng "sai", về phía cực đại cục bộ.

Thủ tục leo đồi

bắt đầu

$t \leftarrow 0$

lặp lại

cục bộ ← FALSE

chọn ngẫu nhiên một chuỗi v_c

tiến hóa v_c

lặp lại

+ chọn 30 chuỗi mới trong lân cận v_c bằng cách thay một các bit của v_c
+ chọn chuỗi v_n trong tập các chuỗi có giá trị hàm mục tiêu *f* lớn nhất
+ nếu $f(v_c) < f(v_n)$ thì $v_c \leftarrow v_n$ nếu không cục bộ ← TRUE

cho đến khi *cục bộ*

$t \leftarrow t+1$

cho đến khi $t = MAX$

chấm dứt

Hình 1.2. Thuật giải leo đồi đơn giản (lặp)

Đối với những bài toán có nhiều tối ưu cục bộ, cơ hội chạm được tối ưu toàn cục của thuật giải leo đồi là rất mong manh.

Còn hình 1.3. trang sau trình bày thủ tục mô phỏng luyện thép.

Hàm *random* [0,1) trả về một số ngẫu nhiên trong khoảng [0,1). Các (điều kiện - dừng) kiểm tra xem đã đạt "cân bằng nhiệt độ"

chưa, nghĩa là, phân bố xác suất của các chuỗi mới đã chọn có đạt đến phân bố Boltzmann không. Tuy nhiên, trong một số cài đặt, vòng lặp chỉ được thực thi k lần, k là tham số bổ sung của phương pháp.

Nhiệt độ T được hạ thấp theo từng bước ($g(T,t) < T$ với mọi t). Thuật giải dừng khi T đạt một giá trị đủ nhỏ: (tiêu chuẩn - dừng) kiểm tra xem hệ thống đã "đông" lại chưa, nghĩa là, không thay đổi nào được chấp nhận nữa.

Như đã nói trước đây, thuật giải mô phỏng luyện thép có thể thoát khỏi tối ưu cục bộ. Ta hãy xem một chuỗi,

v_4 = (11100000010011011100101010000),

với 12 số 1-cho giá trị của $f(v_4) = |11*12 - 150| = 18$ với v_4 là chuỗi khởi tạo, thuật giải leo đồi có thể tiến đến cực đại cục bộ

v_1 =(00000000000000000000000000000).

Do một chuỗi 30 số 1 bất kỳ (nghĩa là, một bước 'tiến đến' tối ưu toàn cục) cho giá trị 7 (nhỏ hơn 18). Mặt khác, thuật giải mô phỏng luyện thép có thể nhận một chuỗi có 30 số 1 làm chuỗi hiện hành mới với xác suất

$p = \exp\{(f(v_n) - f(v_c))/T\} = \exp\{(7-18)/T\}$,

đối với một nhiệt độ nào đó, chẳng hạn $T = 20$, sẽ cho $p = e^{-11/20} = 0.57695$,

nghĩa là các cơ hội chấp nhận tốt hơn 50%.

Thủ tục mô phỏng luyện thép

bắt đầu

t ← 0

khởi tạo nhiệt độ T

chọn ngẫu nhiên chuỗi v_c làm chuỗi hiện hành

tiến hóa v_c

lặp lại

lặp lại

+ chọn chuỗi v_n mới trong lân cận v_c **bằng cách thay một bit của** v_c
+ **nếu** $f(v_c) < f(v_n)$ **thì** $v_c \leftarrow v_n$
+ **nếu không thì nếu** random[0,1) < $\exp\{(f(v_n) - f(v_c))/T\}$ **thì** $v_c \leftarrow v_n$

cho đến khi (điều kiện dừng thỏa)

T ← $g(T,t)$

$t \leftarrow t+1$

cho đến khi (điều kiện dừng 2 thỏa)

chấm dứt

Hình 1.3. Thuật giải mô phỏng luyện thép.

Thuật giải di truyền, như đã trình bày trong 1.1, duy trì một quần thể các chuỗi. Hai chuỗi tương đối xấu,

v_5 = (11111000000011011100111010000 0) và

v_6 = (00000000000110111001010111111 1)

mỗi chuỗi được lượng giá là 16, có thể sinh ra con tốt hơn nhiều (nếu điểm lai rơi vào bất cứ vị trí nào giữa vị trí thứ 5 và thứ 12)

v_7 = (11111000000110111001010111111 1).

Con mới được lượng giá

$f(v_7)$ = {11*19 – 150}= 59

Ta kết thúc chương này bằng cách dẫn ra một thông điệp khôi hài vừa được giới thiệu trên Internet (comp.ai.neural-nets).

" Chú ý rằng, trong mọi kỹ thuật [leo đồi] đã thảo luận từ lâu, con thỏ có thể hy vọng tìm được đỉnh núi gần chỗ nó khởi hành. Không có gì bảo đảm đấy là đỉnh Everest, hoặc một ngọn núi rất cao. Nhiều phương pháp khác nhau đã được dùng để thử tìm ra tối ưu toàn cục thực sự.

Trong mô phỏng luyện thép, con thỏ đã được ăn uống và hy vọng một cách ngẫu nhiên suốt một thời gian dài, nó sáng suốt lên và có hy vọng nhảy lên đến đỉnh đồi cao nhất.

Trong thuật giải di truyền, có nhiều thỏ được thả ngẫu nhiên vào một số nơi của dãy Hy mã lạp sơn. Những con thỏ này không biết rằng chúng phải tìm đỉnh Everest. Nhưng, cứ vài năm một lần

bạn phải bắn chúng ở độ cao thấp, với hy vọng những con còn lại sẽ thành công và sinh sản thêm".

1.5. Kết luận

Ba thí dụ về thuật giải di truyền giải bài toán tối ưu hàm, thế tiến thoái lưỡng nan của tù nhân và người du lịch, cho thấy ứng dụng rộng rãi của thuật giải di truyền. Nhưng đồng thời cũng bộc lộ những khó khăn đầu tiên khi sử dụng thuật giải di truyền. Vấn đề biểu diễn cho bài toán người du lịch không rõ ràng lắm. Phép toán mới được sử dụng (lai tạo OX) không hề tầm thường. Còn những khó khăn nào nữa mà ta gặp phải ở những bài toán nào khác (khó hơn)? Trong hai thí dụ, tối ưu hàm và TSP, hàm lượng giá được định nghĩa rõ ràng; trong thí dụ thứ hai (tiến thoái lưỡng nan của tù nhân) một tiến trình mô phỏng đơn giản cho ta một lượng giá của nhiễm sắc thể (ta thử từng đấu thủ để quyết định sự thành công của nó: mỗi đấu thủ dùng một chiến lược do nhiễm sắc thể của nó định nghĩa để chơi với những đấu thủ khác, còn điểm của mỗi đấu thủ là điểm trung bình trên tất cả những trò chơi mà nó chơi). Ta phải tiến hành cách nào trong trường hợp hàm lượng gía không được định nghĩa rõ ràng? Thí dụ, Bài toán Thoả mãn biến Boolean (Boolean Satisfiability Problem - BSP) dường như cần có một biểu diễn chuỗi nhị phân (bit thứ i biểu diễn thứ tự đúng của biến Boolean thứ i), tuy nhiên, tiến trình chọn một hàm lượng giá không phải là rõ ràng (xem các chương sau).

Thí dụ thứ nhất về tối ưu hàm không ràng buộc cho phép ta dùng một biểu diễn tiện lợi, trong đó bất cứ chuỗi nhị phân nào cũng tương ứng với một giá trị trong miền giá trị của bài toán (nghĩa

là, [-1,2]). Điều này có nghĩa là bất cứ một đột biến hay một lai tạo nào cũng sản sinh một con hợp lệ. Điều này cũng đúng trong thí dụ thứ hai: Một kết hợp các bit bất kỳ biểu diễn một chiến lược hợp lệ. Bài toán thứ ba có một ràng buộc duy nhất: mỗi thành phố phải xuất hiện chính xác một lần trong lộ trình hợp lệ. Điều này gây ra vài vấn đề: ta đã dùng các vectơ số nguyên (thay vì biểu diễn nhị phân) và sửa lại phép toán lai. Thế bằng cách nào chúng ta tiếp cận một bài toán có ràng buộc tổng quát ? Chúng ta phải có những khả năng gì ?

Câu trả lời thật không dễ dàng; chúng ta sẽ xem xét các giải pháp ở trong những phần sau.

Chương 2

THUẬT GIẢI DI TRUYỀN :

CƠ CHẾ THỰC HIỆN

Trong chương này, chúng tôi sẽ trình bày về cơ chế thực hiện của thuật giải di truyền thông qua một bài toán tối ưu số đơn giản. Ta sẽ bắt đầu thảo luận một vài ý kiến chung trước khi đi vào chi tiết ví dụ.

Không mất tính tổng quát, ta giả định những bài toán tối ưu là bài toán tìm giá trị cực đại. Bài toán tìm cực tiểu hàm f chính là tìm cực đại hàm $g = -f$:

$$min\ f(x) = max\ g(x) = max\ (-f(x))$$

Hơn nữa, ta có thể giả định rằng hàm mục tiêu f có giá trị dương trên miền xác định của nó, nếu không, ta có thể cộng thêm một hằng số C dương, nghĩa là:

$$max\ g(x) = max\ (g(x) + C)$$

(vế trái và vế phải cùng đạt max tại 1 điểm x_0)

Bây giờ, giả sử ta muốn tìm cực đại một hàm k biến $f(x_1,...,x_k)$: $R^k \rightarrow R$. Giả sử thêm là mỗi biến x_i có thể nhận giá trị trong miền $D_i = [a_i, b_i] \subseteq R$ và $f(x_1,...,x_k) > 0$ với mọi x_i thuộc D_i. Ta muốn tối ưu hóa hàm f với một độ chính xác cho trước: giả sử cần 6 số lẻ đối với giá trị của các biến.

Rõ ràng là để đạt được độ chính xác như vậy mỗi miền D_i được phân cắt thành $(b_i-a_i)\times 10^6$ miền con bằng nhau. Gọi m_i là số nguyên nhỏ nhất sao cho

$$(b_i - a_i) \times 10^6 \le 2^{m_i} - 1$$

Như vậy, mỗi biến x_i được biểu diễn bằng một chuỗi nhị phân có chiều dài m_i. Biểu diễn như trên, rõ ràng thoả mãn điều kiện về độ chính xác yêu cầu. Công thức sau tính giá trị thập phân của mỗi chuỗi nhị phân biểu diễn biến x_i,

$$x_i = a_i + decimal(111001...001_2).\ \frac{b_i - a_i}{2^{m_i} - 1}$$

trong đó $decimal(chuỗi_2)$ cho biết giá trị thập phân của chuỗi nhị phân đó.

Bây giờ, mỗi nhiễm sắc thể (là một lời giải) được biểu diễn bằng chuỗi nhị phân có chiều dài $m=\sum_{i=1}^{k} m_i$; m_1 bit đầu tiên biểu diễn các giá trị trong khoảng $[a_1,b_1]$; m_2 bit kế tiếp biểu diễn giá trị trong khoảng $[a_2, b_2]$; ... ; nhóm m_k bít cuối cùng biểu diễn giá trị trong khoảng $[a_k, b_k]$.

Để khởi tạo quần thể, chỉ cần đơn giản tạo *pop-size* nhiễm sắc thể ngẫu nhiên theo từng bit. Nhưng nếu bạn biết một chút về xác suất, thì nên dùng những hiểu biết về sự phân phối để khởi tạo quần thể ban đầu sẽ tốt hơn.

Phần còn lại của thuật giải di truyền rất đơn giản: trong mỗi thế hệ, ta lượng giá từng nhiễm sắc thể (tính giá trị hàm f trên các chuỗi biến nhị phân đã được giải mã), chọn quần thể mới thỏa phân bố xác suất dựa trên độ thích nghi và thực hiện các phép đột biến và lai để tạo các cá thể thế hệ mới. Sau một số thế hệ, khi không còn cải thiện thêm được gì nữa, nhiễm sắc thể tốt nhất sẽ được xem

như lời giải của bài toán tối ưu (thường là toàn cục). Thông thường, ta cho dừng thuật giải di truyền sau một số bước lặp cố định tùy thuộc điều kiện về tốc độ và tài nguyên máy tính.

Đối với tiến trình chọn lọc (chọn quần thể mới thỏa phân bố xác suất dựa trên các độ thích nghi), ta dùng bánh xe quay Ru lét với các rãnh được định kích thước theo độ thích nghi. Ta xây dựng bánh xe Rulét như sau (giả định rằng, các độ thích nghi đều dương, trong trường hợp ngược lại thì ta có thể dùng một vài phép biến đổi tương ứng để định lại tỷ lệ sao cho các độ thích nghi đều dương)

- Tính độ thích nghi $eval(v_i)$ của mỗi nhiễm sắc thể v_i $(i= 1...pop\text{-}size)$.
- Tìm tổng giá trị thích nghi toàn quần thể: $F = \sum_{i=1}^{pop\_size} eval(v_i)$
- Tính xác suất chọn p_i cho mỗi nhiễm sắc thể v_i , $(i= 1...pop\text{-}size)$: $p_i = eval\ (v_i)/F$
- Tính vị trí xác suất q_i của mỗi nhiễm sắc thể v_i, $(i= 1...pop\text{-}size)$. $q_i = \sum_{j=1}^{i} p_j$

Tiến trình chọn lọc được thực hiện bằng cách quay bánh xe ru lét *pop-size* lần; mỗi lần chọn một nhiễm sắc thể từ quần thể hiện hành vào quần thể mới theo cách sau:

- Phát sinh ngẫu nhiên một số r trong khoảng [0..1]
- Nếu $r < q_1$ thì chọn nhiễm sắc thể đầu tiên (v_1); ngược lại thì chọn nhiễm sắc thể thứ i, v_i $(2 \le i \le pop\text{-}size)$ sao cho $q_{i-1} < r \le q_i$.

Hiển nhiên, có thể sẽ có một số nhiễm sắc thể được chọn nhiều lần. Điều này phù hợp với *lý thuyết sơ đồ* (xem chương 3): các nhiễm

sắc thể tốt nhất có nhiều bản sao hơn, các nhiễm sắc thể trung bình không thay đổi, các nhiễm sắc thể kém nhất thì chết đi.

Bây giờ ta có thể áp dụng phép toán di truyền: kết hợp, lai vào các cá thể trong quần thể mới, vừa được chọn từ quần thể cũ như trên. Một trong những tham số của hệ di truyền là xác suất lai p_c. Xác suất này cho ta số nhiễm sắc thể *pop-size*$\times p_c$ mong đợi, các nhiễm sắc thể này được dùng trong tác vụ lai tạo. Ta tiến hành theo cách sau đây:

Đối với mỗi nhiễm sắc thể trong quần thể (mới) :

- Phát sinh ngẫu nhiên một một số r trong khoảng [0..1];
- Nếu $r < p_c$, hãy chọn nhiễm sắc thể đó để lai tạo.

Bây giờ, ta ghép đôi các nhiễm sắc thể đã chọn được một cách ngẫu nhiên: đối với mỗi cặp nhiễm sắc thể được ghép đôi, ta phát sinh ngẫu nhiên một số nguyên *pos* trong khoảng [1..m-1] (m là tổng chiều dài - số bit - của một nhiễm sắc thể). Số *pos* cho biết vị trí của điểm lai. Hai nhiễm sắc thể:

$(b_1b_2...b_{pos}b_{pos+1}... b_m)$ và

$(c_1c_2...c_{pos}c_{pos+1}...c_m)$

được thay bằng một cặp con của chúng:

$(b_1b_2...b_{pos}\ c_{pos+1}... c_m)$

$(c_1c_2...c_{pos}\ b_{pos+11}... b_m)$

Phép toán kế tiếp, đột biến, được thực hiện trên cơ sở từng bit. Một tham số khác của hệ thống di truyền p_m, cho ta số bit đột biến $p_m \times m \times$ *pop-size* mong đợi. Mỗi bit (trong tất cả các nhiễm sắc thể

trong quần thể) có cơ hội bị đột biến như nhau, nghĩa là, đổi từ 0 thành 1 hoặc ngược lại. Vì thế ta tiến hành theo cách sau đây.

Đối với mỗi nhiễm sắc thể trong quần thể hiện hành (nghĩa là sau khi lai) và đối với mỗi bit trong nhiễm sắc thể:

- Phát sinh ngẫu nhiên một số r trong khoảng [0..1];
- Nếu $r < p_m$, hãy đột biến bit đó.

Sau quá trình chọn lọc, lai và đột biến, quần thể mới đến lượt lượng giá kế tiếp của nó. Lượng giá này được dùng để xây dựng phân bố xác suất (cho tiến trình chọn lựa kế tiếp), nghĩa là, để xây dựng lại bánh xe rulét với các rãnh được định kích thước theo các giá trị thích nghi hiện hành. Phần còn lại của tiến hóa chỉ là lặp lại chu trình của những bước trên (xem hình 0.1 trong phần dẫn nhập).

Toàn bộ tiến trình sẽ được minh họa trong một thí dụ. Trong ví dụ này, ta sẽ mô phỏng thuật giải di truyền giải bài toán tối ưu số.

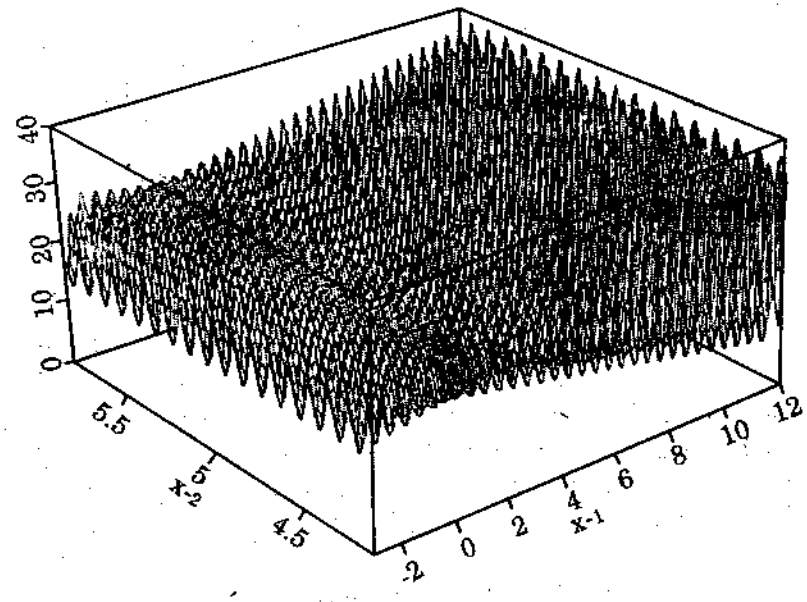
Giả sử kích thước quần thể *pop-size* = 20, và các xác suất di truyền tương ứng là p_c =0.25 và p_m =0.01.

Ta cần cực đại hóa hàm sau đây:

$$f(x_1,x_2) = 21.5 + x_1 \times sin\ (4\pi x_1) + x_2 \times sin(20\pi\ x_2)$$

với $-3.0 \le x_1 \le 12.1$ và $4.1 \le x_2 \le 5.8$.

Hình 2.1. là đồ thị của hàm f.



Hình 2.1. Đồ thị hàm $f(x_1,x_2) = 21.5 + x_1 \times \sin(4\pi x_1) + x_2 \times \sin(20\pi x_2)$

Giả sử ta cần tính chính xác đến 4 số lẻ đối với mỗi biến. Miền của biến x_1 có chiều dài 15.1; điều kiện chính xác đòi hỏi đoạn [-3.0,12.1] cần được được chia thành các khoảng có kích thước bằng nhau, ít nhất là 15.1×10000 khoảng. Điều này nghĩa là cần 18 bit làm phần đầu tiên của nhiễm sắc thể:

$$2^{17} \le 151000 \le 2^{18}$$

Miền của biến x_2 có chiều dài 1.7; điều kiện chính xác đòi hỏi đoạn [4.1,5.8] cần được được chia thành các khoảng có kích thước bằng nhau, ít nhất là 1.7×10000 khoảng. Điều này nghĩa là cần 15 bit làm phần đầu tiên của nhiễm sắc thể:

$$2^{14} \le 17000 \le 2^{15}$$

Chiều dài toàn bộ nhiễm sắc thể (vectơ lời giải) lúc này là m=18+15= 33 bit; 18 bit đầu tiên mã hóa x_1, và 15 bit còn lại (từ 19 đến 33) mã hóa x_2.

Ta hãy xét một nhiễm sắc thể làm thí dụ:

(010001001011010000111110010100010)

18 bit đầu tiên, 010001001011010000, biểu diễn

$$x_1 = -3.0 + decimal(010001001011010000_2) \times \frac{12.1-(-3.0)}{2^{18}-1}$$

$$= -3{,}0+70352\frac{15.1}{2262143} = -3.0 + 4.052426$$

15 bit kế tiếp 111110010100010, biểu diễn

$$x_2 = 4.1 + decimal(111110010100010_2) \times \frac{5.8-4.1}{2^{15}-1} = 4.1+31906.\frac{1.7}{32767}$$

$$= .1+1.655330$$

$$= 5.755330$$

Như vậy, nhiễm sắc thể

(010001001011010000111110010100010)

tương ứng với $<x_1,x_2>$ = <1.052426, 5.755330>.

Độ thích nghi của nhiễm sắc thể này là

f(1.052426, 5.755330)= 20.252640.

Để cực đại hóa hàm f bằng thuật giải di truyền, ta tạo một quần thể có pop_size = 20 nhiễm sắc thể. Cả 33 bit trong tất cả các nhiễm sắc thể đều được khởi tạo ngẫu nhiên.

Giả sử rằng sau tiến trình khởi tạo , ta có quần thể sau đây:

v_1= (10011010000000111111110100110111111)

v_2= (11100010010011011100101010011010)

v_3= (00001000001100100000101011011101)

v_4= (10001100010110100111100000110010)

v_5= (00011101100101001101011111000101)

v_6= (00010100001001010100101011111011)

v_7= (00100010000011010111101101111011)

v_8= (10000110000111010001011010110011 1)

v_9= (01000000010110001011000000111 1100)

v_{10}=(00000111100011000001101000011 1011)

v_{11}=(01100111111011010110000110111 1000)

v_{12}=(11010001011110110100010101000 0000)

v_{13}=(11101111101000100011000000100 0110)

v_{14}=(01001001100000101010011110010 1001)

v_{15}=(11101110110111000010001111101 1110)

v_{16}=(11001111000001111111000011010 01011)

v_{17}=(01101011111100111101000110111 1101)

v_{18}=(01110100000000111010011111010 1101)

v_{19}=(00010101001111111111000011000 1100)

v_{20}=(10111001011001111001100010111 1110)

Trong giai đoạn lượng giá ta giải mã từng nhiễm sắc thể và tính các giá trị hàm thích nghi từ các giá trị (x_1, x_2) mới giải mã. Ta có:

$eval(v_1) = f(6.084492, 5.652242) = 26.019600$

$eval(v_2) = f(10.348434, 4.380264) = 7.580015$

$eval(v_3) = f(-2.516603, 4.390381) = 19.626329$

$eval(v_4) = f(5.278638, 5.593460) = 17.406725$

$eval(v_5) = f(-1.255173, 4.734458) = 25.341160$

$eval(v_6) = f(-1.811725, 4.391937) = 18.100417$

$eval(v_7) = f(-0.991471, 5.680258) = 16.020812$

$eval(v_8) = f(4.910618, 4.703018) = 17.959701$

$eval(v_9) = f(0.795406, 5.381472) = 16.127799$

$eval(v_{10}) = f(-2.554851, 4.793707) = 21.278435$

$eval(v_{11}) = f(3.130078, 4.996097) = 23.410669$

$eval(v_{12})$ = f(9.356179, 4.239457) = 15.011619

$eval(v_{13})$ = f(11.134646, 5.378671) = 27.316702

$eval(v_{14})$ = f(1.335944, 5. 151378) = 19.876294

$eval(v_{15})$ = f(11.089025, 5.054515) = 30.060205

$eval(v_{16})$ = f(9.211598, 4.993762) = 23.967227

$eval(v_{17})$ = f(3.367514, 4.571343) = 13.696165

$eval(v_{18})$ = f(3.843020, 5.158226) = 15.414128

$eval(v_{19})$ = f(-1.746635, 5.395584) = 20.095903

$eval(v_{20})$ = f(7.935998, 4.757338) = 13.666916

Rõ ràng nhiễm sắc thể v_{15} mạnh nhất và nhiễm sắc thể v_2 yếu nhất.

Bây giờ ta xây dựng hệ thống kiến trúc bánh xe rulét cho tiến trình chọn lọc. Tổng độ thích nghi của quần thể là:

$$F = \sum_{i=1}^{20} eval(v_i) = 387.776822$$

Xác suất chọn lọc p_i của mỗi nhiễm sắc thể v_i (i=1,..,20) là:

p_1= $eval(v1)$ / F = 0.067099 $p_2 = eval(v)$ / F = 0.019547

$p_3 = eval(v)$ / F = 0.050355 $p_4 = eval(v)$ / F = 0.044889

$p_5 = eval(v)$ / F = 0.065350 $p_6 = eval(v)$ / F = 0.046677

$p_7 = eval(v)$ / F = 0.041315 $p_8 = eval(v)$ / F =0.046315

$p_9 = eval(v)$ / F = 0.041590 $p_{10} = eval(v)$ / F =0.054873

$p_{11} = eval(v)$ / F =0.060372 $p_{12} = eval(v)$ / F =0.038712

$p_{13} = eval(v)$ / F =0.070444 $p_{14} = eval(v)$ / F =0.051257

$p_{15} = eval(v)$ / F =0.077519 $p_{16} = eval(v)$ / F =0.061549

$p_{17} = eval(v)$ / F =0.035320 $p_{18} = eval(v)$ / F =0.039750

$p_{19} = eval(v)$ / F =0.051823 $p_{20} = eval(v)$ / F =0.035244

Các vị trí xác suất q_i của mỗi nhiễm sắc thể v_i (I=1,..,20) là:

q_1= 0.067099 q_2 = 0.086647 q_3 = 0.137001

q_4= 0.181890 q_5 = 0.247240 q_6 = 0.293917

q_7= 0.335232 q_8 = 0.381546 q_9 = 0.423137

q_{10}= 0.478009 q_{11} = 0.538381 q_{12} = 0.577093

q_{13}= 0.647537 q_{14} = 0.698794 q_{15} = 0.776314

q_{16}=0.837863 q_{17} =0.873182 q_{18} =0.812932

q_{19}=0.964756 q_{20}=1.000000

Bây giờ ta quay bánh xe rulét 20 lần; mỗi lần chọn một nhiễm sắc thể cho quần thể mới. Giả sử thứ tự (ngẫu nhiên) của 20 số trong khoảng [0,1] được phát sinh là

0.513870 0.175741 0.308652 0.534534 0.947628

0.171736 0.702231 0.226431 0.494773 0.424720

0.703899	0.389647	0.277226	0.368071	0.983437
0.005398	0.765682	0.646473	0.767139	0.780237

Số đầu tiên r =0.513870 lớn hơn q_{10} và nhỏ hơn q_{11}, nghĩa là nhiễm sắc thể v_{11} được chọn vào quần thể mới; số thứ hai, 0.175741 lớn hơn q_3 nhỏ hơn q_4, nghĩa là v_4 được chọn cho quần thể mới,vv...

Như vậy, quần thể mới gồm có các nhiễm sắc thể sau:

v'_1 = (01100111111011010110000110111111000) (v_{11})

v'_2 = (10001100010110100111110000011110010) (v_4)

v'_3 = (00100010000011010111101101111111011) (v_7)

v'_4 = (01100111111011010110000110111111000) (v_{11})

v'_5 = (00010101001111111111100000110001100) (v_{19})

v'_6 = (10001100010110100111110000011110010) (v_4)

v'_7 = (11101110110111000010001111011110) (v_{15})

v'_8 = (00011101100101001101011111100010 1) (v_5)

v'_9 = (01100111111011010110000110111111000) (v_{11})

v'_{10} = (00001000001100100000101011011101) (v_3)

v'_{11} = (11101110110111000010001111011110) (v_{15})

v'_{12} = (01000000010110001011000000111110 0) (v_9)

v'_{13} = (00010100001001010100101011111101 1) (v_6)

v'_{14} = (10000110000111010001011010110011 1) (v_8)

v'_{15} = (10111001011001111100110001011111 10) (v_{20})

v'_{16} = (10011010000000111111101001101111 1) (v_1)

v'_{17} = (00000111100011000001101000011101 1) (v_{10})

v'_{18} = (11101111101000100011000000100011 0) (v_{13})

v'_{19} = (11101110110111000010001111101111 0) (v_{15})

v'_{20} = (11001111000001111110000110100101 1) (v_{16})

Bây giờ ta sẽ áp dụng phép toán kết hợp, lai cho những cá thể trong quần thể mới (các vectơ v'_i). Xác suất lai p_c = 0.25 vì thế ta hy vọng (trung bình) 25% nhiễm sắc thể (nghĩa là, 5/20) sẽ tham gia lai tạo. Ta tiến hành theo cách sau: đối với mỗi nhiễm sắc thể trong quần thể (mới) ta phát sinh ngẫu nhiên một số r trong khoảng [0,1]; nếu r < 0.25, ta chọn một nhiễm sắc thể cho trước để lai tạo.

Giả sử thứ tự các số ngẫu nhiên là:

0.822951	0.151932	0.625477	0.314685	0.346901
0.917204	0.519760	0.401154	0.606758	0.785402
0.031523	0.869921	0.166525	0.574520	0.758400
0.581893	0.389248	0.200232	0.355635	0.826927

Điều này có nghĩa là các nhiễm sắc thể v'_2, v'_{11}, v'_{13} và v'_{18} đã được chọn để lai tạo. (Ta đã gặp may: số nhiễm sắc thể được chọn là chẵn, vậy có thể ghép thành cặp một cách dễ dàng. Trường hợp là số lẻ, chúng ta sẽ cộng thêm một nhiễm sắc thể ngoại hoặc lấy bớt một

nhiễm sắc thể - thực hiện điều này một cách ngẫu nhiên thì tốt hơn). Bây giờ ta cho phối ngẫu một cách ngẫu nhiên: tức là, hai nhiễm sắc thể đầu tiên (ví dụ v'_2 và v'_{11}) và cặp kế tiếp (v'_{13} và v'_{18}) được kết cặp. Đối với mỗi cặp trong hai cặp này, ta phát sinh một số nguyên ngẫu nhiên *pos* thuộc khoảng {1, .., 32} (32+1 là tổng chiều dài – số bit – trong một nhiễm sắc thể). Số *pos* cho biết vị trí của điểm lai tạo. Cặp nhiễm sắc thể đầu tiên là:

v'_2 = (100011000| 10110100111100000l110010)

v'_{11}= (111011101| 10111000010001111111011110)

và giả sử số phát sinh *pos* = 9. Các nhiễm sắc thể này bị cắt sau bit thứ 9 và được thay bằng cặp con của chúng:

v''_2 = (100011000| 10111000010001111111011110)

v''_{11}= (111011101| 10110100111100000l110010)

Cặp nhiễm sắc thể thứ hai là:

v'_{13}= (000101000010010101001 1010111111011)

v'_{18}= (11101111101000100011| 0000001000110)

và số phát sinh *pos* = 20. Các nhiễm sắc thể này được thay bởi một cặp con của chúng:

v''_{13} = (000101000010010101001 0000001000110)

v''_{18} = (11101111101000100011| 1010111111011)

Cuối cùng, quần thể hiện hành là:

v'_1 = (01100111111011010110000110111l000)

v'_2 = (10001100010111000010001111111011110)

v'_3 = (00100010000011010111101101111111011)

v'_4 = (01100111111011010110000110111l1000)

v'_5 = (00010101001111111111100001100011100)

v'_6 = (10001100010110100111100000111001O)

v'_7 = (11101110110111000010001111111011110)

v'_8 = (00011101100101001101011111100010l)

v'_9 = (01100111111011010110000110111l1000)

v'_{10} = (00001000001100100000101011101110l)

v'_{11} = (11101110110110100111100000111001O)

v'_{12} = (01000000010110001011000000011111OO)

v'_{13} = (00010100001001010100000000100011O)

v'_{14} = (10000110000111010001011010110011l)

v'_{15} = (10111001011001111001100010111111O)

v'_{16} = (10011010000000111111101001101111l)

v'_{17} = (00000111100011000001101000011101l)

v'_{18} = (11101111101000100011101011111101l)

v'_{19} = (11101110110111000010001111111011lO)

v'_{20} = (11001111000001111110000110100101l)

Phép toán kế tiếp, đột biến, được thực hiện trên cơ sở từng bit một. Xác suất đột biến p_m= 0.01, vì thế ta hy vọng (trung bình) 1/100 số bit sẽ qua đột biến. Có 660 bit (*m×pop-size=33×20)* trong

toàn quần thể; ta hy vọng (trung bình) 6.6 đột biến mỗi thế hệ. Mỗi bit có cơ hội đột biến ngang nhau, vì thế, đối với mỗi bit trong quần thể, ta phát sinh ngẫu nhiên một số r trong khoảng [0,1]; nếu $r <$ 0.01, ta đột biến bit này.

Điều này có nghĩa là ta phải phát sinh 660 số ngẫu nhiên. Giả sử, có 5 trong số 660 số này nhỏ hơn 0.01; vị trí bit và số ngẫu nhiên được trình bày dưới đây:

Vị trí bit	Số ngẫu nhiên
112	0.000213
349	0.009945
418	0.008809
429	0.005425
602	0.002835

Bảng sau cho biết nhiễm sắc thể vị trí của bit bị đột biến tương ứng với 5 vị trí bit trên.

Vị trí bit	Số nhiễm sắc thể	Số bit trong nhiễm sắc thể
112	4	13
349	11	19
418	13	22
429	13	33
602	19	8

Điều này có nghĩa là 4 nhiễm sắc thể chịu ảnh hưởng của phép toán đột biến: một trong số này là nhiễm sắc thể thứ 13, có hai bit bị thay đổi.

Quần thể cuối cùng được liệt kê ở dưới; các bit đột biến được tô đậm và gạch dưới.

v_1 =(01100111111011010110000110111**1**000)

v_2 =(10001100010111000010001111101111**0**)

v_3 =(00100010000011010111101101111101**1**)

v_4 =(01100111111**0**0101011000011011110**0**0)

v_5 =(00010101001111111111000011000110**0**)

v_6 =(10001100010110100111100000111001**0**)

v_7 =(11101110110111000010001111101111**0**)

v_8 =(00011101100101001101011111100010**1**)

v_9 =(01100111111011010110000110111**0**00)

v_{10} =(00001000001100100000101011101**1**01)

v_{11} =(11101110110110100**1**011000001110010)

v_{12} =(01000000010110001011000000111**1**00)

v_{13} =(00010100001001010100**1**00001000**1**1)

v_{14} =(10000110000111010001011010110**0**111)

v_{15} =(10111001011001111001100010111111**0**)

v_{16} =(10011010000000111111101001101**1**111)

v_{17} =(00000111100011000001101000011101 1)

v_{18} =(11101111101000100011101011111101 1)

v_{19} =(11101110010111000010001111101111 0)

v_{20} =(11001111000001111110000110100101 1)

Ta vừa hoàn thành một bước lặp (nghĩa là một thế hệ) của thủ tục di truyền (Hình 0.1 trong phần dẫn nhập). Ta xem xét một chút các kết quả của tiến trình tiến hóa quần thể mới. Trong thời kỳ tiến hóa, ta giải mã từng nhiễm sắc thể và tính các giá trị của hàm thích nghi từ giá trị (x_1, x_2) vừa được giải mã. Ta được:

$eval(v_1)$ = f(3.130078, 4.996097) = 23.410669

$eval(v_2)$ = f(5.279042, 5.054515) = 18.201083

$eval(v_3)$ = f(-0.991471, 5.680258) = 16.020812

$eval(v_4)$ = f(3.128235, 4.996097) = 23.412613

$eval(v_5)$ = f(-1.746635, 5.395584) =20.095903

$eval(v_6)$ = f(5.278638, 5.593460) =17.406725

$eval(v_7)$ = f(11.089025, 5.054515) = 30.060205

$eval(v_8)$ = f(-1.255173, 4.734458) = 25.341160

$eval(v_9)$ = f(3.130078, 4.996097) = 23.410669

$eval(v_{10})$ = f(-2.516603, 4.390381) =19.526329

$eval(v_{11})$ = f(11.088621, 4.743434) =33.351874

$eval(v_{12})$ = f(0.795406, 5.381472) = 16.127799

$eval(v_{13})$ = f(-1.811725, 4.209937) = 22.692462

$eval(v_{14})$ = f(4.910618, 6.703018) = 17.959701

$eval(v_{15})$ = f(7.935998, 4.757388) = 13.666916

$eval(v_{16})$ = f(6.084492, 5.652242) = 26.019600

$eval(v_{17})$ = f(-2.554851, 4.793707) = 21.278435

$eval(v_{18})$ = f(11.134646, 5.666976) = 27.591064

$eval(v_{19})$ = f(11.059532, 5.054515) = 27.608441

$eval(v_{20})$ = f(9.211598, 4.993762) = 23.867227

Chú ý rằng tổng độ thích nghi F của quần thể mới là 447.049688 cao hơn tổng độ thích nghi của quần thể trước nhiều (387.776822). Cũng thế, nhiễm sắc thể tốt nhất hiện nay v_{11} có độ thích nghi (33.351874) tốt hơn nhiễm sắc thể tốt nhất v_{15} của quần thể trước (30.060205).

Tiến trình lặp lại lần nữa và các phép toán di truyền được áp dụng, lượng giá thế hệ kế tiếp .v.v... Giả sử sau 1000 thế hệ ta được quần thể:

v_1= (11101111011001101110010101011101 1)

v_2= (11100110011000010001010101011100 0)

v_3= (11101111011101101110010101011101 1)

v_4= (11100110001000011000010101011100 1)

v_5= (11101111011101101110010101011011)

v_6= (11100110011100010000010001010000 1)

v_7= (11010110001001001000110001011000 0)

v_8= (11110110001000101000110101001000 1)

v_9= (11100110001001001000110001011000 1)

v_{10}= (11101111011101101110010101011101 1)

v_{11}= (11010110000001001000110001011000 0)

v_{12}= (11010110001001001000110001011000 1)

v_{13}= (11101111011101101110010101011101 1)

v_{14}= (11100110011000010000101010111011)

v_{15}= (11100110101011100101010011011000 1)

v_{16}= (11100110011000010100010001010000 1)

v_{17}= (11100110011000010000010101011101 1)

v_{18}= (11100110011000010000010101011100 1)

v_{19}= (11110110001000101000111000001000 1)

v_{20}= (11100110011000010000010101011100 1)

Với độ thích nghi tương ứng là:

$eval(v_1) = f$ (11.120940, 5.092514) = 30.298543

$eval(v_2) = f$ (10.588756, 4.667358) = 26.869724

$eval(v_3) = f$ (11.124647, 5.092514) = 30.316575

$eval(v_4) = f$ (10.574125, 4,242410) = 31.933120

$eval(v_5) = f$ (11.124627, 5.092514) = 30.316575

$eval(v_6) = f$ (10.588756, 4.214603) = 34.356125

$eval(v_7) = f$ (9.631066, 4.427881) = 35.458636

$eval(v_8) = f$ (11.518106, 4.452835) = 23.309078

$eval(v_9) = f$ (10.574816, 4.427933) = 34.393820

$eval(v_{10}) = f$ (11.124627, 5.092514) = 30.316575

$eval(v_{11}) = f$ (9.623693, 4.427881) = 35.477938

$eval(v_{12}) = f$ (9.631066, 4.427933) = 35.456066

$eval(v_{13}) = f$ (11.124627, 5.092514) =30.316575

$eval(v_{14}) = f$ (10.588756, 4.242514) = 32.932098

$eval(v_{15}) = f$ (10.606555, 4.653714) = 30.746768

$eval(v_{16}) = f$ (10.588814, 4.214603) = 34.359545

$eval(v_{17}) = f$ (10.588756, 4.242514) = 32.932098

$eval(v_{18}) = f$ (10.588756, 4.242410) = 32.956664

$eval(v_{19}) = f$ (11.518106, 4.472757) = 19.669670

$eval(v_{20}) = f(10.588756, 4.242410) = 32.956664$

Tuy nhiên, nếu ta khảo sát tiến trình trong khi chạy, ta có thể khám phá rằng trong những thế hệ đầu, các giá trị thích nghi của một số nhiễm sắc thể tốt hơn giá trị 35.477938 của nhiễm sắc thể tốt nhất sau 1000 thế hệ. Điều này là do các lỗi tạo mẫu hỗn loạn - trong các chương sau ta sẽ bàn về vấn đề này.

Không khó để theo dõi cá thể tốt nhất trong tiến trình tiến hóa. Thông thường (trong các cài đặt thuật giải di truyền) cá thể tốt "trội hơn cả" được lưu trữ tại một vị trí riêng biệt; bằng cách đó, thuật giải có thể duy trì cá thể tốt nhất tìm được trong suốt quá trình (không chắc là cá thể tốt nhất trong quần thể cuối cùng).

Chương 3

THUẬT GIẢI DI TRUYỀN : NGUYÊN LÝ HOẠT ĐỘNG

Nền tảng lý thuyết của thuật giải di truyền dựa trên biểu diễn chuỗi nhị phân và lý thuyết sơ đồ. Một sơ đồ là một chuỗi, dài bằng chuỗi nhiễm sắc thể, các thành phần của nó có thể nhận một trong các giá trị trong tập ký tự biểu diễn gen hoặc một ký tự *đại diện* '*'. Sơ đồ biểu diễn một không gian con của không gian tìm kiếm. Không gian con này là tập tất cả các chuỗi trong không gian lời giải mà với mọi vị trí trong chuỗi, giá trị của gen trùng với giá trị của sơ đồ; Ký tự đại diện '*' có thể trùng khớp với bất kỳ ký tự biểu diễn gen nào.

Thí dụ, các chuỗi và sơ đồ có chiều dài 10. Sơ đồ (*111100100) sẽ khớp với hai chuỗi:

{(0111100100), (1111100100)}

và sơ đồ (* 1 * 1 1 0 0 1 0 0) sẽ khớp với 4 chuỗi

{(0101100100), (0111100100), (1101100100), (1111100100)}

Đương nhiên, sơ đồ (1001110001) chỉ khớp với chính nó, và sơ đồ (**********) khớp với tất cả các chuỗi có chiều dài 10. Rõ ràng là mỗi sơ đồ cụ thể có tương ứng 2^r chuỗi, với r là số ký tự đại diện '*' có trong sơ đồ. Mặt khác, mỗi chuỗi chiều dài m sẽ khớp với 2^m sơ đồ. Thí dụ, xét chuỗi (1001110001). Chuỗi này phù hợp với 2^{10} sơ đồ sau:

(1 0 0 1 1 1 0 0 0 1)

(* 0 0 1 1 1 0 0 0 1)

(1 * 0 1 1 1 0 0 0 1)

(1 0 * 1 1 1 0 0 0 1)

:

:

(1 0 0 1 1 1 0 0 0 *)

(* * 0 1 1 1 0 0 0 1)

(* 0 * 1 1 1 0 0 0 1)

:

:

(1 0 0 1 1 1 0 0 * *)

(* * * 1 1 1 0 0 0 1)

:

:

(* * * * * * * * * *),

Một chuỗi chiều dài m, sẽ có tối đa 3^m sơ đồ. Trong một quần thể kích thước n, có thể có tương ứng từ 2^m đến $n \times 2^m$ sơ đồ khác nhau.

Các sơ đồ khác nhau có những đặc trưng khác nhau. Các đặc trưng này thể hiện qua hai thuộc tính quan trọng: *bậc* và *chiều dài xác định*.

(1) *Bậc* của sơ đồ S (ký hiệu là $o(S)$) là số các vị trí 0 và 1 có trong sơ đồ. Đây chính là các vị trí cố định (không phải là những vị trí của ký tự đại diện), trong sơ đồ. Nói cách khác,

bậc là chiều dài của chuỗi trừ đi số ký tự đại diện. *Bậc* xác định đặc trưng của sơ đồ. Thí dụ, ba sơ đồ chiều dài 10 sau:

S_1 = (* * * 0 0 1 * 1 1 0)

S_2 = (* * * * 0 0 * * 0 *)

S_3 = (1 1 1 0 1 * * 0 0 1)

có bậc tương ứng:

$o(S_1)$= 6 ; $o(S_2)$ = 3 ; $o(S_3)$= 8;

và S_3 là sơ đồ 'đặc hiệu' nhất.

Khái niệm bậc của sơ đồ giúp cho việc tính xác suất sống còn của sơ đồ do ảnh hưởng của đột biến; Ta sẽ thảo luận chi tiết sau.

(2) *Chiều dài xác định* của sơ đồ S (ký hiệu là $\delta(S)$) là khoảng cách giữa hai vị trí cố định ở đầu và cuối. Nó định nghĩa 'độ nén' của thông tin chứa trong một sơ đồ. Thí dụ:

$\delta(S_1)$= 10 – 4 = 6; $\delta(S_2)$= 9 – 5= 4; $\delta(S_3)$= 10 – 1 = 9.

Như vậy, một sơ đồ chỉ có một vị trí cố định duy nhất thì sẽ có chiều dài xác định là 0.

Khái niệm *chiều dài xác định* của sơ đồ giúp tính xác suất sống còn của sơ đồ do ảnh hưởng của phép lai; Ta cũng sẽ thảo luận chi tiết sau.

Như đã thảo luận ở các chương trước, tiến trình mô phỏng tiến hóa của thuật giải di truyền là quá trình lặp gồm có 4 bước:

$t \leftarrow t+1$

chọn P(t) từ P(t-1)

tái kết hợp P(t)

lượng giá P(t)

Bước 1, ($t \leftarrow t+1$), chỉ đơn giản đến số thế hệ tiến hóa; và bước cuối (*lượng giá P(t)*) là lượng giá để tính độ thích nghi của các cá thể trong quần thể hiện hành. Hiện tượng chủ yếu của chu trình tiến hóa xảy ra trong hai bước còn lại của: *chọn lọc* và *tái kết hợp*. Ta sẽ bàn về hiệu quả của hai bước này trên một số sơ đồ cần thiết, biểu diễn trong một quần thể.

Ta bắt đầu bằng bước *chọn lọc*.

Giả sử, quần thể có kích thước pop-size = 20, chiều dài của chuỗi (và cũng là chiều dài của các sơ đồ) là $m = 33$ (như trong thí dụ đã trình bày trong chương trước). Giả sử thêm rằng (ở thế hệ thứ t) quần thể gồm có các chuỗi sau đây:

v_1 = (10011010000000111111110100110111111)

v_2 = (11100010010011011100101010011010)

v_3 = (00001000001100100000101011011101)

v_4 = (10001100010110100111100000011100010)

v_5 = (00011101100101001101011111100010 1)

v_6 = (00010100001001010100101011111011)

v_7 = (00100010000011010111101101111101 1)

v_8 = (10000110000111010001011010110011 1)

v_9 = (01100000010110001011000000111110 0)

v_{10} = (00000111100011000001101000011101 1)

v_{11} = (01100111111011010110000110111100 0)

v_{12} = (11010001011110110100010101000000 0)

v_{13} = (11101111101000100011000000100011 0)

v_{14} = (01001001100000101010011110010100 1)

v_{15} = (11101110110111000010001111101111 0)

v_{16} = (11001111000001111110000110100101 1)

v_{17} = (01101011111100111101000110111110 1)

v_{18} = (01110100000000111010011111010110 1)

v_{19} = (00010101001111111111000011000110 0)

v_{20} = (10111001011001111001100010111111 0)

(1) Đặt $\xi(S, t)$ là số chuỗi trong quần thể, ở thế hệ thứ t, phù hợp với sơ đồ S. Thí dụ, đối với sơ đồ,

S_0 =(* * * * 1 1 1 *),

thì $\xi(S_0, t) = 3$; vì có 3 chuỗi, v_{13}, v_{15} và v_{16}, phù hợp với sơ đồ S_0. Chú ý rằng bậc của sơ đồ S_0, $o(S_0) = 3$, và chiều dài xác định của nó $\delta(S_0) = 7 - 5 = 2$.

(2) Gọi *eval(S,t)* là độ thích nghi của sơ đồ S ở thế hệ *t*. Giả sử có *p* chuỗi $\{v_{i1},...,v_{ip}\}$ trong quần thể phù hợp với sơ đồ *S* vào thời điểm *t*. Thì:

$$eval(S,t) = \frac{\sum_{j=1}^{p} eval(v_{ij})}{p}.$$

Trong bước *chọn lọc*, một quần thể trung gian được tạo ra gồm *pop-size* = 20 các chuỗi được chọn ra từ quần thể hiện hành. Các chuỗi được chọn dựa và độ thích nghi của nó và được chép vào quần thể thế hệ mới. Như ta đã biết trong chương trước, chuỗi v_i có xác suất được chọn là $p_i = eval(v_i)/F(t)$ ($F(t)$ là tổng thích nghi của toàn quần thể vào thời điểm *t*, $F(t) = \sum_{i=1}^{20} eval(v_i)$).

Sau bước *chọn lọc*, ta có $\xi(S, t+1)$ chuỗi phù hợp với sơ đồ *S*. Do:

(1) Với một chuỗi phù hợp với sơ đồ *S*, trung bình xác suất được chọn của nó là *eval(S,t) / F(t)*,

(2) Ở thế hệ *t*, số chuỗi phù hợp với sơ đồ *S* là $\xi(S, t)$ và,

(3) Chọn trong *pop-size* chuỗi,

vậy:

$$\xi(S, t+1) = \xi(S, t) * pop\text{-}size * eval(S,t) / F(t).$$

Với $\overline{F(t)} = F(t)/pop\_size$ là độ thích nghi trung bình của quần thể, ta có thể viết lại công thức trên thành:

$$\xi(S, t+1) = \xi(S, t) * eval(S,t) / \overline{F(t)} \quad (3.1)$$

Nói cách khác, số chuỗi trong quần thể tăng bằng với tỉ lệ độ thích nghi của sơ đồ với độ thích nghi trung bình của quần thể. Điều

này có nghĩa là sơ đồ "trên trung bình" sẽ nhận được thêm số chuỗi trong quần thể thế hệ kế tiếp, sơ đồ "dưới trung bình" nhận số chuỗi giảm đi, còn sơ đồ trung bình vẫn giữ nguyên mức.

Hệ quả lâu dài của luật trên cũng rõ ràng. Nếu ta cho rằng sơ đồ *S* vẫn giữ trên trung bình ε % (nghĩa là, $eval(S,t) = \overline{F(t)} + \varepsilon * \overline{F(t)}$, thì:

$$\xi(S, t) = \xi(S, 0)\ (1+\varepsilon)^t,$$

và $\varepsilon = (eval(S,t) - \overline{F(t)}) / \overline{F(t)}$ ($\varepsilon > 0$, đối với các sơ đồ trên trung bình và $\varepsilon < 0$ đối với các sơ đồ dưới trung bình).

Như vậy, ta có thể nói rằng, chẳng những một sơ đồ "trên trung bình" nhận số chuỗi tăng lên trong thế hệ kế tiếp, mà nó cũng tiếp tục nhận số chuỗi tăng theo lũy thừa trong các thế hệ kế tiếp.

Ta gọi phương trình (3.1) là phương trình tăng trưởng sinh sản của sơ đồ.

Trở về với sơ đồ thí dụ, S_0. Vì có ba chuỗi là v_{13}, v_{15} và v_{16} (vào thời điểm *t*) phù hợp với sơ đồ S_0, độ thích nghi $eval(S_0)$ của sơ đồ là:

$$eval(S_0,t) = (27.316702 + 30.060205 + 23.867227) / 3 = 27.081378$$

đồng thời, độ thích nghi trung bình của toàn quần thể là:

$$\overline{F(t)} = \sum_{i=1}^{20} eval(v_i) / pop\_size = 387.776822 / 20 = 19.388841$$

và tỉ lệ thích nghi của sơ đồ S_0 đối với độ thích nghi trung bình của quần thể là:

$$eval(S_0,t) / \overline{F(t)} = 1.396751$$

Điều này có nghĩa là, nếu sơ đồ S_0 trên trung bình, thì nó nhận số chuỗi tăng theo lũy thừa trong các thế hệ kế tiếp. Đặc biệt, nếu sơ đồ S_0 vẫn giữ trên trung bình do nhân với hằng số 1.396751, thì, vào thời điểm $t+1$, ta sẽ có 3*1.396751 = 4.19 chuỗi phù hợp với S_0 (nghĩa là 4 hoặc 5), vào thời điểm $t+2$: $3*1.396751^2$ = 5.85 chuỗi như vậy (nghĩa là gần 6 chuỗi), vv... .

Dễ thấy rằng, sơ đồ S_0 như vậy xác định một phần hứa hẹn của không gian tìm kiếm và số mẫu đại diện của nó trong quần thể sẽ tăng theo lũy thừa.

Ta sẽ kiểm tra dự đoán này vào thí dụ của ta, với sơ đồ S_0. Trong quần thể vào thời điểm t, sơ đồ S_0 phù hợp với 3 chuỗi v_{13}, v_{15}, và v_{16}. Trong chương trước, ta đã mô phỏng tiến trình *chọn lọc* sử dụng cùng một quần thể để tạo ra quần thể mới. Quần thể mới gồm các nhiễm sắc thể sau:

v'_1 = (01100111111011010110000110111100) (v_{11})

v'_2 = (10000100010110100111100000111001 0) (v_4)

v'_3 = (00100010000011010111101101111101 1) (v_7)

v'_4 = (01100111111011010110000110111100 0) (v_{11})

v'_5 = (00010101001111111111000001100011 00) (v_{19})

v'_6 = (10001100010110100111100000111001 0) (v_4)

v'_7 = (11101110110111000010001111101111 0) (v_{15})

v'_8 = (00011101100101001101011111100010 1) (v_5)

v'_9 = (01100111111011010110000110111100 0) (v_{11})

v'_{10} = (00001000001100100000101011101110 1) (v_3)

v'_{11} = (11101110110111000010001111101111 0) (v_{15})

v'_{12} = (01000000010110001011000000111110 0) (v_9)

v'_{13} = (00010100001001010100101011111101 1) (v_6)

v'_{14} = (10000110000111010001011010110011 1) (v_8)

v'_{15} = (10111001011001111100110001011111 10) (v_{20})

v'_{16} = (11100110011000010100010001010000 1) (v_1)

v'_{17} = (11100110011000010000010101011101 1) (v_{10})

v'_{18} = (11101111101000100011000000100011 0) (v_{13})

v'_{19} = (11101110110111000010001111101111 0) (v_{15})

v'_{20} = (11001111000001111110000110100101 1) (v_{16})

Sơ đồ S_0 bây giờ phù hợp với 5 chuỗi v'_7, v'_{11}, v'_{18}, v'_{19} và v'_{20}.

Tuy nhiên, chỉ riêng phép chọn thì không giới thiệu một điểm mới nào đáng lưu tâm từ không gian tìm kiếm; chọn lọc chỉ sao chép một số chuỗi để hình thành quần thể trung gian. Vì thế, bước thứ hai của chu kỳ tiến hóa *tái kết hợp*, có nhiệm vụ giới thiệu những cá thể mới trong quần thể. Điều này được thực hiện bởi các phép toán di truyền: *lai* và *đột biến*. Ta lần lượt xem xét tác động của hai phép toán này trên một số sơ đồ trong quần thể.

PHÉP LAI

Bắt đầu với phép lai và xét thí dụ sau. Như ta đã biết, một chuỗi trong quần thể, chẳng hạn

v'_{18} = (11101111101000100011000000100011 0),

phù hợp với tối đa 2^{33} sơ đồ; cụ thể là chuỗi trên phù hợp với hai sơ đồ sau:

S_0= (* * * * 1 1 1 *) và

S_1= (1 1 1 * 1 0).

Giả sử thêm rằng chuỗi này được chọn để thi hành phép lai (như ở chương 2) và (theo như ví dụ ở chương 2, v'_{18} được lai với v'_{13}) vị trí lai phát sinh tại *pos* = 20. Rõ ràng là sơ đồ S_0 vẫn tồn tại, nghĩa là vẫn còn một con phù hợp với S_0. Lý do là vị trí lai này bảo tồn chuỗi '111' trên các vị trí thứ 5, thứ 6, và thứ 7 của chuỗi trong một đứa con: các chuỗi:

v'_{18} = (11101111101000100011| 0000001000110),

v'_{13} = (00010100001001010100| 1010111111011),

sẽ sinh ra:

v''_{18} = (11101111101000100011| 1010111111011)

v''_{13} = (00010100001001010100| 0000001000110)

Tuy nhiên, sơ đồ S_1 có thể bị phá vỡ: không con nào phù hợp với nó. Lý do là các vị trí cố định '111' ở đầu mẫu và các vị trí cố định '10' ở cuối được đặt vào con khác.

Rõ ràng là *chiều dài xác định* của sơ đồ đóng vai trò quan trọng trong xác suất bị loại bỏ hay tồn tại của sơ đồ. Chú ý rằng, *chiều dài xác định* của sơ đồ S_0 là $\delta(S_0) = 2$, còn *chiều dài xác định* của sơ đồ S_1 là $\delta(S_1) = 32$.

Và, các vị trí lai, trong số *m-1* vị trí, có cơ hội được chọn ngang nhau. Điều này có nghĩa là xác suất bị loại của sơ đồ S là:

$$p_d(S) = \frac{\delta(s)}{m-1},$$

và do đó, xác suất tồn tại là:

$$p_s(S) = 1 - \frac{\delta(s)}{m-1}.$$

Cụ thể, các xác suất tồn tại và bị loại của các sơ đồ S_0 và S_1 là:

$p_d(S_0) = 2/32$; $p_s(S_0) = 30/32$; $p_d(S_1) = 32/32 = 1$; $p_s(S_1) = 0$.

Có một điều quan trọng cần lưu ý là chỉ có một số nhiễm sắc thể trải qua lai và xác suất lai là p_c. Điều này có nghĩa xác suất tồn tại của sơ đồ thực sự sẽ là:

$$p_s(S) = 1 - p_c \cdot \frac{\delta(s)}{m-1}.$$

Ta lại xem xét sơ đồ S_0 của ta, vẫn với thí dụ đang xét (p_c= 0.25):

$p_s(S_0)$ = 1-0.25 * 2/32 = 63/64= 0.984375.

Cũng chú ý rằng, ngay cả khi đã chọn một vị trí lai trong số các vị trí cố định trong một sơ đồ, sơ đồ vẫn có cơ may tồn tại. Thí dụ, nếu cả hai chuỗi v'_{18} và v'_{13} đều bắt đầu với '111' và tận cùng bằng '10', thì sơ đồ S_1 vẫn tồn tại (nhưng, xác suất của hiện tượng này rất nhỏ). Do đó, ta nên hiệu chỉnh công thức xác suất tồn tại của sơ đồ:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(s)}{m-1}$$

Như vậy, tác động kết hợp của chọn lọc và lai cho ta một dạng mới của phương trình tăng trưởng của sơ đồ sinh sản:

$$\xi(S, t+1) = \xi(S,t).eval\ (S,t)\ /\ \overline{F(t)}\left[1 - p_c.\frac{\delta(S)}{m-1}\right] \quad (3.2)$$

Phương trình (3.2) cho biết kỳ vọng số chuỗi phù hợp với sơ đồ S trong thế hệ kế tiếp là hàm của số chuỗi đúng của sơ đồ, về độ thích nghi tương đối của sơ đồ và *chiều dài xác định* của nó. Rõ ràng là sơ đồ trên trung bình có *chiều dài xác định* ngắn vẫn có thể có số chuỗi cá thể khớp với nó và tốc độ tăng theo lũy thừa. Đối với sơ đồ S_0:

$$eval\ (S,t)\ /\ \overline{F(t)}\left[1 - p_c.\frac{\delta(S)}{m-1}\right] = 1.396751 * 0.984375 = 1.374927$$

Điều này có nghĩa là sơ đồ ngắn, trên trung bình S_0 vẫn nhận được số chuỗi tăng theo lũy thừa trong các thế hệ tiếp theo: vào lúc $(t+1)$ ta có 3*1.374927 = 4.12 chuỗi phù hợp với S_0 (chỉ hơi kém hơn 4.19 - là giá trị mà ta chỉ tính đến chọn lọc). Vào lúc $(t+2)$ có : $3*1.374927^2 = 5.67$ chuỗi như vậy (lại chỉ hơi kém 5.85).

ĐỘT BIẾN

Phép toán kế tiếp được bàn đến là đột biến. Phép đột biến thay đổi một vị trí trong nhiễm sắc thể một cách ngẫu nhiên với xác suất p_m. Thay đổi từ 0 thành 1 hay ngược lại. Rõ ràng là tất cả các vị trí cố định của sơ đồ phải giữ không đổi nếu sơ đồ muốn tồn tại qua đột biến. Thí dụ, xét một chuỗi sau trong quần thể, chẳng hạn như chuỗi:

v'_{19} = (11101110110111000010001111110111 10)

và sơ đồ S_0:

S_0= (* * * * 1 1 1 *)

Giả sử thêm rằng, chuỗi v'_{19} tham gia đột biến, nghĩa là có ít nhất một bit bị đảo, như đã xảy ra trong chương trước. (Nhớ lại

trong chương trước là 4 chuỗi đã trải qua đột biến: một trong những chuỗi này, v'_{13} bị đột biến ở hai vị trí, ba chuỗi kia - kể cả v'_{19} – chỉ đột biến ở một vị trí). Do v'_{19} đột biến tại vị trí thứ 8, nên kết quả của nó:

v''_{19} = (11101110010111000010001111110111 10),

vẫn phù hợp với sơ đồ S_0. Nếu các vị trí đột biến được chọn là từ 1 đến 4, hay từ 8 đến 33, thì chuỗi kết quả vẫn phù hợp với S_0. Chỉ 3 bit (thứ 5, 6 và 7 - các vị trí bit cố định trong sơ đồ S_0) là "quan trọng": đột biến ít nhất một trong các bit này sẽ loại bỏ sơ đồ S_0. Rõ ràng, số những bit "quan trọng" bằng với *bậc* của sơ đồ, nghĩa là bằng số các vị trí cố định.

Vì xác suất thay đổi của một bit là p_m nên xác suất tồn tại của một bit là 1 - p_m. Một lần đột biến độc lập với các đột biến khác, vì thế xác suất tồn tại của sơ đồ S qua đột biến (nghĩa là chuỗi các đột biến một-bit) là:

$$p_s(S) = (1 - p_m)^{o(S)}$$

Do $p_m << 1$, xác suất này có thể được tính gần đúng là:

$$p_s(S) \approx 1 - o\ (S) * p_m$$

Trở lại với sơ đồ thí dụ S_0 và thí dụ đang chạy (p_m= 0.01) :

$$p_s(S_0) \approx 1 - 3 * 0.01 = 0.97$$

Tác động kết hợp của chọn lọc, lai tạo và đột biến cho ta dạng mới của phương trình tăng trưởng sơ đồ sinh sản:

$$\xi(S, t+1) \geq \xi(S,t).eval\ (S,t)\ /\ \overline{F(t)}\left[1 - p_c.\frac{\delta(S)}{m-1} - o(S).p_m\right] \quad (3.3)$$

Cũng như trong dạng đơn giản (phương trình (3.1) và (3.2)), phương trình (3.3) cũng cho ta biết về kỳ vọng số chuỗi phù hợp với sơ đồ S trong thế hệ tiếp theo là hàm theo: số chuỗi phù hợp với sơ đồ, thích nghi tương đối của sơ đồ, *chiều dài xác định* và *bậc* của sơ đồ. Ta lại thấy rằng, các sơ đồ trên trung bình có *chiều dài xác định* ngắn và *bậc*-thấp vẫn có số chuỗi phù hợp với tốc độ tăng theo lũy thừa.

Đối với sơ đồ S_0:

$$eval(S_0,t)/\overline{F(t)}\left[1-\rho_c.\frac{\delta(S)}{m-1}-o(S).\rho_m\right] = 1.396751 * 0.954375 = 1.333024$$

Điều này có nghĩa là sơ đồ ngắn, bậc thấp, trên trung bình S_0 vẫn nhận một số chuỗi tăng theo lũy thừa trong các thế hệ kế tiếp. Vào lúc $t+1$, ta có $3 * 1.333024 = 4.00$ chuỗi phù hợp với S_0 (không kém 4.19 nhiều - là giá trị mà chỉ tính đến việc chọn lựa, hay 4.12 - là giá trị mà ta tính đến việc chọn lọc và lai tạo **), vào lúc $t+2$: $3*1.333024^2 = 5.33$ chuỗi như vậy (lại không kém 5.85 hoặc 5.67 nhiều).

Chú ý rằng phương trình (3.3) dựa trên giả định là hàm thích nghi f chỉ trả về các giá trị dương; khi áp dụng GA vào những bài toán tối ưu hóa mà ở đó hàm tối ưu có thể trả về các giá trị âm, ta cần có một số ánh xạ bổ sung giữa các hàm tối ưu và độ thích nghi.

Tóm lại, phương trình tăng trưởng (3.1) cho biết chọn lọc làm tăng tốc độ tạo mẫu của các sơ đồ trên trung bình, và cho biết thay đổi này là theo lũy thừa. Việc tăng trưởng tự nó không đưa ra một sơ đồ mới nào (không được biểu diễn trong tạo mẫu ban đầu, lúc $t=0$). Điều này chính là lý do mà phép lai được đưa vào để giúp việc trao đổi thông tin cấu trúc nhưng ngẫu nhiên. Ngoài ra, phép lai đưa tính biến thiên cao hơn vào quần thể. Tác động kết hợp (gây rối) của những phép này đối với một sơ đồ không hề quan trọng nếu sơ đồ

ngắn và có *bậc* thấp. Kết quả cuối cùng của phương trình tăng trưởng (3.3) có thể được phát biểu như sau:

Định lý1 (Định lý sơ đồ) *Các sơ đồ ngắn, bậc thấp, trên trung bình nhận số chuỗi tăng theo lũy thừa trong các thế hệ tiếp theo của thuật giải di truyền.*

Kết quả tức thời của định lý này là GA khảo sát không gian tìm kiếm bằng những sơ đồ ngắn, *bậc* thấp, do đó những sơ đồ này được dùng để trao đổi thông tin trong khi lai:

Giả thuyết 1 (Giả thuyết khối kiến trúc) *Thuật giải di truyền tìm việc thực hiện gần tối ưu qua việc đặt gần nhau các sơ đồ ngắn, bậc thấp, hiệu quả cao, được gọi là các khối kiến trúc.*

Chúng ta đã thấy thí dụ hoàn hảo về khối kiến trúc trong chương này:

S_0 = (* * * * 1 1 1 *).

S_0 là sơ đồ ngắn, có bậc thấp và cũng trên trung bình (ít nhất là đối với các quần thể ban đầu). Sơ đồ này góp phần lớn trong việc tìm tối ưu.

Dù đã có một số nghiên cứu được thực hiện nhằm chứng minh giả thuyết này, nhưng đối với hầu hết các ứng dụng khó, ta hầu như chỉ tin tưởng vào kết quả thực nghiệm. Suốt 15 năm sau này, nhiều ứng dụng của GA đã được phát triển để hỗ trợ giả thuyết khối kiến trúc trong nhiều lãnh vực khác nhau. Dù vậy, giả thiết này cho thấy là việc biểu diễn nhiễm sắc thể cho thuật giải di truyền có tính quyết định đối với việc thực hiện nó, và việc biểu diễn này cần thỏa mãn ý tưởng về các khối kiến trúc.

Ở phần đầu của chương, chúng ta đã phát biểu là quần thể có *pop-size* cá thể chiều dài m tạo ra ít nhất 2^m và nhiều nhất là *pop-*

size 2^m* sơ đồ. Một số sơ đồ được xử lý theo cách có lợi: chúng sinh sản theo tốc độ lũy thừa (đáng mong đợi), và không bị loại bỏ bởi lai và đột biến (điều này có thể xảy ra cho các sơ đồ có chiều dài xác định lớn và bậc cao).

Holland đã chứng tỏ cho thấy rằng ít nhất *pop-size* của chúng được xử lý một cách có ích - ông gọi thuộc tính này là *cơ chế song song ẩn*, vì nó đã đạt được mà không cần bất cứ bộ nhớ ngoài hoặc nhu cầu xử lý nào.

Chương này đã trình bày một số giải thích lý thuyết về lý do thành công của thuật giải di truyền. Nhưng hãy chú ý rằng giả thuyết về khối kiến trúc chỉ là vấn đề tin tưởng và dễ bị xâm phạm trong một số bài toán. Thí dụ, giả sử rằng hai sơ đồ ngắn, bậc thấp, (lần này, ta xét các sơ đồ có chiều dài toàn bộ là 11 vị trí),

S_1 = (1 1 1 * * * * * * * *)

S_2= (* * * * * * * * * 1 1),

là trên trung bình, nhưng kết hợp của chúng,

S_3 = (1 1 1 * * * * * * 1 1), thì ít phù hợp hơn

S_4 = (0 0 0 * * * * * * 0 0).

Giả sử thêm rằng chuỗi tối ưu là S_0 = (1111111111111) (S_3 phù hợp). Một thuật giải di truyền có thể gặp nhiều khó khăn trong việc hội tụ về S_0, vì nó có thể hội tụ về các điểm như (00011111100). Hiện tượng này được gọi là bị *lừa*: một số khối kiến trúc (các sơ đồ ngắn, bậc thấp) có thể làm lạc hướng thuật giải di truyền và là cho nó hội tụ vào các điểm dưới tối ưu.

Phần 2

TỐI ƯU SỐ

Chương 4

BIỂU DIỄN NHIỄM SẮC THỂ CHO BÀI TOÁN TỐI ƯU SỐ

Khi ứng dụng thuật giải di truyền vào thực tế, đôi khi gặp những bài toán đòi hỏi một cách biểu diễn lời giải thích hợp, nếu không, thuật giải di truyền khó có thể cho lời giải tốt được. Có thể là do GA thường hội tụ sớm về một lời giải tối ưu không - toàn cục; cũng có thể là do đó là bài toán tối ưu với các ràng buộc không tầm thường.

Biểu diễn nhị phân truyền thống có một số bất lợi khi áp dụng GA giải các bài toán số cần độ chính xác cao, trong một không gian số chiều lớn. Thí dụ, tối ưu hàm 100 biến; mỗi biến nhận giá trị trong khoảng [-500, 500], chính xác đến 6 số lẻ, thì chiều dài của vectơ lời giải nhị phân phải là 3000; phát sinh một không gian tìm kiếm khoảng 10^{1000} phần tử. Tìm kiếm trong một không gian như thế, thuật giải di truyền thực hiện rất kém hiệu quả.

Cách biểu diễn nhị phân khiến cho bất kỳ cách mã hoá nào cũng có cùng số sơ đồ, chính vì vậy, cách dùng chuỗi bít mã hóa lời giải là một trong những công việc chính của người nghiên cứu GA. Cách mã hóa chuỗi nhị phân thường giúp dễ dàng phân tích lý thuyết và cho phép xây dựng các toán tử di truyền "đẹp". Nhưng kết quả của 'cơ chế song song ẩn' thực sự không phụ thuộc vào việc sử dụng các chuỗi bit và có lẽ cũng nên thử nghiệm với tập mẫu tự lớn (thay vì nhị phân chỉ 0 và 1) và những toán tử di truyền (có thể là) mới. Đặc biệt, đối với các bài toán tối ưu số với các biến liên tục, ta có thể thử nghiệm với các gen mã hoá là các số thực cùng với các toán tử "di truyền" đặc biệt cho cách mã hoá số thực này.

Trong chương này, chúng tôi trình bày cách biểu diễn không nhị phân để giải bài toán tối ưu số với số biến đủ lớn. Các gen sẽ nhận giá trị thực, thay vì nhị phân như trong thuật giải di truyền gốc. Mục đích chính là để mở rộng không gian tìm kiếm của GA đến gần không gian thực của bài toán hơn: mở rộng đó buộc các phép toán cũng phải được sửa đổi bằng cách sử dụng một số đặc trưng cụ thể của không gian thực. Thí dụ, biểu diễn này có thuộc tính là hai điểm gần nhau trong không gian biểu diễn cũng phải gần nhau trong không gian bài toán, và ngược lại. Điều này không phải luôn luôn đúng trong cách tiếp cận nhị phân. Tuy nhiên, nếu dùng cách biểu diễn nhị phân nhưng theo nguyên tắc *Gray* ta cũng có thể mở rộng không gian tìm kiếm của GA gần với không gian thực của bài toán.

Thủ tục để chuyển một số nhị phân $\boldsymbol{b} = (b_1,..., b_m)$ thành số mã Gray $\boldsymbol{g}= (g_1,...,g_m)$ và ngược lại được trình bày trong hình 4.1; tham số m cho biết số bit trong các biểu diễn này.

Thủ tục chuyển từ nhị phân sang mã Gray

Bắt đầu

$g_1 = p_1$

Cho $k = 2$ **đến** m **Thực hiện**

$g_k = b_{k-1}$ *XOR* b_k

Kết thúc

Thủ tục chuyển từ mã Gray về biểu diễn nhị phân

Bắt đầu

giá_trị = g_1

b_1= *giá_trị*

Cho $k = 2$ **đến** m **Thực hiện**

Nếu $g_k = 1$ **Thì** *giá_trị* = *NOT giá_trị*

b_k = giá_trị

Hết lặp

Kết thúc

Hình 4.1. Các thủ tục chuyển từ nhị phân sang mã Gray và từ mã Gray về dạng nhị phân.

Bảng 4.1. Liệt kê 16 số nhị phân cùng với các mã Gray tương ứng.

Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Chú ý rằng biểu diễn Gray có đặc điểm là hai điểm gần nhau bất kỳ trong không gian bài toán chỉ sai khác nhau một bit. Nói cách khác, việc tăng giá trị tham số lên một tương ứng với việc thay đổi chỉ một bit trong mã. Cũng cần chú ý là có nhiều cách để chuyển đổi giữa nhị phân và Gray. Thí dụ (trường hợp $m = 4$), cặp ma trận:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

cho ra hai biến đổi sau:

$$\boldsymbol{g} = A\boldsymbol{b} \text{ và } \boldsymbol{b} = A^{-1}\boldsymbol{g}$$

trong đó phép nhân là phép nhân modulo 2.

Tuy nhiên, ta dùng biểu diễn chấm động, thay vì nhị phân dạng Gray, vì nó gần với không gian bài toán hơn và cũng cho phép dễ cài đặt hiệu quả các phép toán di truyền. Đã có nhiều nghiên cứu so sánh thực nghiệm các cài đặt (nhị phân và chấm động) trên một số bài toán khác nhau. Trong chương này, chúng tôi minh hoạ những khác biệt giữa biểu diễn nhị phân và chấm động cho một bài toán đặc biệt: bài toán tuyến tính - bậc 2. Bài toán tuyến tính-bậc 2, là trường hợp đặc biệt của bài toán ta sẽ sử dụng trong chương sau, được dùng để minh họa chương trình tiến hoá về tính hội tụ sớm và khả năng tìm kiếm lời giải chính xác.

4.1. Mô tả bài toán

Ta đã chọn bài toán điều khiển động sau:

$$min \quad x_N^2 + \sum_{k=0}^{N-1} (x_k^2 + u_k^2)$$

với $x_{k+1} = x_k + u_k$, k= 0,1,. . . , $N - 1$;

x_0 là trạng thái đầu được cho;

$x_k \in R$ là biến trạng thái;

và $u \in R^N$ là vectơ điều khiển.

Nghiệm giải tích của giá trị tối ưu là:

$J^* = K_0 x^2_0$,

trong đó K_k là nghiệm của phương trình Riccati:

$K_k = 1 + K_{k+1} / (1 + K_{k+1})$ và $K_N = 1$.

Trong các thực nghiệm sau, nhiễm sắc thể sẽ biểu diễn vectơ trạng thái điều khiển $\boldsymbol{u}$. Ta cũng đã thừa nhận miền cố định là (-200, 200) cho mỗi u_i. Đối với các thử nghiệm, ta chọn $x_0 = 100$ và $N = 45$, nghĩa là, nhiễm sắc thể $\boldsymbol{u} = \langle u_0, \ldots, u_{44} \rangle$, có giá trị tối ưu $J^* = 16180.4$.

4.2. Hai cài đặt thử nghiệm

Để so sánh, chúng tôi trình bày cả 2 phiên bản: một phiên bản với biểu diễn nhị phân và phiên bản kia dùng biểu diễn thực.

4.2.1. Phiên bản nhị phân

Trong phiên bản nhị phân, mỗi phần tử của vectơ nhiễm sắc thể được mã hóa bằng cùng số bit. Để giải mã nhanh khi chạy chương trình, mỗi phần tử tương ứng một từ nhớ (nói chung, ta có thể sử dụng nhiều bit hơn 1 từ nhớ, nhưng trường hợp này là trường hợp mở rộng). Theo cách này, các phần tử có thể khai báo kiểu integer, với một số thao tác nhị phân hoá cũng như thập phân hoá

một phần tử. Rồi, mỗi nhiễm sắc thể là một vectơ gồm N từ nhớ, bằng với số phần tử của mỗi nhiễm sắc thể (hoặc một bội số của từ nhớ, nếu biểu diễn cần nhiều lần số từ nhớ).

Độ chính xác của cách tiếp cận này phụ thuộc (với miền xác định có kích thước cố định) vào số bit thực sự sử dụng và bằng với $(UB\text{-}LB) / (2^n\text{-}1)$, trong đó UB và LB là các cận của miền xác định, còn n là số bit trong mỗi phần tử của một nhiễm sắc thể.

4.2.2. Phiên bản thực

Trong phiên bản thực, mỗi vectơ nhiễm sắc thể được mã hóa bằng một vectơ các số thực, có cùng độ dài như vectơ lời giải. Mỗi phần tử buộc phải nằm trong khoảng mong muốn, và các phép toán được thiết kế một cách cẩn thận để bảo toàn yêu cầu này.

Độ chính xác của cách tiếp cận này chỉ phụ thuộc vào máy. Đương nhiên, ta luôn có thể mở rộng độ chính xác của biểu diễn nhị phân bằng cách dùng nhiều bit hơn, nhưng như vậy sẽ làm giảm đáng kể tốc độ của thuật giải (xem phần 4.4).

Ngoài ra, biểu diễn thực còn có khả năng biểu diễn những miền thật lớn (hay trường hợp các miền xác định không biết trước). Mặt khác, biểu diễn nhị phân phải hy sinh độ chính xác khi tăng kích thước miền, với chiều dài nhị phân cố định cho trước. Cũng vậy, trong biểu diễn thực, việc thiết kế các công cụ đặc biệt để xử lý các ràng buộc không tầm thường sẽ dễ dàng hơn nhiều: điều này sẽ được bàn đầy đủ trong chương 6.

4.3. Thử nghiệm

Các thử nghiệm được thực hiện trên cùng một cấu hình máy tính. Tất cả các kết quả trình bày ở đây biểu diễn kết quả trung bình đạt được sau 10 lần chạy độc lập của từng phiên bản trên.

Trong tất cả các thử nghiệm, kích thước quần thể được giữ cố định là 60, và số lần lặp là 20000. Nếu không thay đổi trạng thái, biểu diễn nhị phân được dùng n = 30 bit để mã hóa một biến (một phần tử của vectơ lời giải), vậy cần 30 x 45 = 1350 bit cho toàn bộ vectơ điều khiển u.

4.3.1. Đột biến và lai tạo ngẫu nhiên

Trong thực nghiệm này, ta chạy cả hai phiên bản với những phép toán tương đương với các phép toán truyền thống.

PHIÊN BẢN NHỊ PHÂN

Phiên bản nhị phân sử dụng những phép toán đột biến và lai tạo truyền thống, nhưng, để cho giống với những phép toán của phiên bản thực hơn, ta chỉ cho phép lai giữa những phần tử. Xác suất lai được giữ cố định là 0.25, trong khi xác suất đột biến được thay đổi để đạt được tốc độ cập nhật nhiễm sắc thể mong muốn (bảng 4.2).

Bảng 4.2. xác suất cập nhật nhiễm sắc thể đối với tốc độ đột biến.

	Xác suất cập nhật nhiễm sắc thể				
Phiên bản	0.6	0.7	0.8	0.9	0.95
Nhị phân, p_m	0.00047	0.00068	0.00098	0.0015	0.0021
Thực, p_m	0.014	0.02	0.03	0.045	0.061

PHIÊN BẢN THỰC

Phép lai của phiên bản thực hoàn toàn giống với phiên bản nhị phân (các điểm phân cách giữa số chấm động) và được áp dụng với cùng xác suất (0.25). Đột biến, áp dụng cho số chấm động thay vì

cho bit; kết quả của đột biến đó là một trị ngẫu nhiên trong miền [LB, UB].

KẾT QUẢ THỰC NGHIỆM

Bảng 4.3. Kết quả trung bình là hàm của xác suất cập nhật nhiễm sắc thể

Phiên bản	Xác suất cập nhật nhiễm sắc thể					Độ lệch chuẩn
	0.6	0.7	0.8	0.9	0.95	
Nhị phân	42179	46102	29290	52769	30573	31212
Thực	46594	41806	47454	69624	82371	11275

Kết quả (bảng 4.3) cho thấy phiên bản nhị phân có vẻ hơi tốt hơn; nhưng cũng khó mà nói rằng chúng tốt hơn thực sự do tất cả đều còn cách xa lời giải tối ưu thực (16180.4). Hơn nữa, một hiện tượng cũng đáng lưu tâm: phiên bản thực ổn định hơn với độ lệch chuẩn thấp hơn nhiều.

Ngoài ra, nếu để ý rằng thực nghiệm trên không hoàn toàn thích hợp đối với biểu diễn thực; đột biến của nó không 'tự nhiên' bằng đột biến của phiên bản nhị phân.

Để minh họa, ta hãy khảo sát: xác suất để sau khi đột biến một phần tử sẽ rơi trong δ % của một khoảng (400, do khoảng giá trị là [-200, 200]) từ giá trị cũ của nó? Câu trả lời là:

Phiên bản thực: xác suất đó rõ ràng rơi trong khoảng $[\delta, 2\delta]$. Thí dụ, cho $\delta = 0.05$, nó sẽ trong khoảng [0.05, 0.1].

Phiên bản nhị phân: ở đây ta cần xét số bit bậc thấp có thể thay đổi một cách an toàn. Giả sử chiều dài phần tử $n = 30$ và m là chiều dài thay đổi cho phép, m phải thỏa $m \le n + log_2\delta$. Do m là số nguyên, nên $m = \lfloor n + \log_2\delta \rfloor = 25$ và xác suất là $m/n = 25/30 = 0.833$, một số hoàn toàn khác.

Do đó, ta sẽ cố thiết kế một phương pháp để bù lấp những khuyết điểm này.

4.3.2. Đột biến không đồng nhất

Trong thực nghiệm này, ta chạy, ngoài các phép toán được nói đến trong phần 4.3.1, còn có một phép toán đột biến đặc biệt có mục đích vừa cải thiện việc tìm chính xác một phần tử vừa làm giảm bất lợi của đột biến ngẫu nhiên trong phiên bản thực. Ta gọi nó là *đột biến không đồng nhất*; trong hai chương sau ta sẽ bàn đầy đủ về phép toán này.

PHIÊN BẢN THỰC

Phép *đột biến không đồng nhất* được định nghĩa như sau: nếu $s_v^t = \langle v_1,\ldots,v_m\rangle$ là một nhiễm sắc thể (t là số thế hệ) và phần tử v_k được chọn để đột biến, kết quả là một vectơ $s_v^{t+1} = \langle v'_1, \ldots, v'_m\rangle$, trong đó,

$$v_k = \begin{cases} v_k + \Delta(t, UB - v_k), & \text{nếu chữ số ngẫu nhiên là } 0 \\ v_k - \Delta(t, v_k - LB), & \text{nếu chữ số ngẫu nhiên là } 1 \end{cases}$$

LB và UB là cận trên và cận dưới miền xác định của biến v_k. Hàm $\Delta(t,y)$ trả về một giá trị trong khoảng $[0, y]$ sao cho xác suất của $\Delta(t,y)$, gần với 0, tăng khi t tăng. Tính chất này khiến phép toán này lúc đầu tìm kiếm không gian đồng bộ (khi t nhỏ), và rất cục bộ ở những giai đoạn sau; như vậy sẽ tăng xác suất phát sinh một số mới

gần với hậu duệ hơn là lựa chọn ngẫu nhiên. Ta đã dùng hàm sau đây:

$$\Delta(t, y) = y.(1 - r^{(1-\frac{t}{T})^b})$$

trong đó r là một số ngẫu nhiên trong khoảng [0..1]. T là số thế hệ cực đại và b là tham số hệ thống xác định mức độ phụ thuộc vào số lần lặp (ta dùng b = 5).

PHIÊN BẢN NHỊ PHÂN

Để thích hợp hơn đối với biểu diễn nhị phân, ta mô hình hóa toán tử đột biến không đồng nhất vào không gian của nó, mặc dù, thực sự, nó được đưa vào chủ yếu để cải thiện đột biến của phiên bản thực. Ở đây, tương tự với phép toán của phiên bản thực, nhưng với v'_k được xác định khác:

$$v'_k = \text{đột biến}(v_k, \nabla(t,n)),$$

trong đó, n = 30 là số bit/phần tử của một nhiễm sắc thể; $\text{độtbiến}(v_k, pos)$ có nghĩa là: giá trị đột biến của phần tử thứ k trên bit pos (0 bit là kém ý nghĩa nhất) và

$$\nabla(t,n) = \begin{cases} \lfloor \Delta(t,n) \rfloor, & \text{nếu chữ số ngẫu nhiên là } 0 \\ \lceil \Delta(t,n) \rceil, & \text{nếu chữ số ngẫu nhiên là } 1 \end{cases}$$

với tham số b của Δ được điều chỉnh thích hợp (ta dùng b= 1.5).

KẾT QUẢ THỰC NGHIỆM

Ta lặp lại các thực nghiệm tương tự các như ở phần 4.3.1, dùng đột biến không đồng nhất được áp dụng cùng tỉ lệ với các đột biến được định nghĩa trước đây.

Bảng 4.4. Kết quả trung bình là hàm xác suất của cập nhật nhiễm sắc thể

Phiên Bản	Xác suất cài đặt nhiễm sắc thể		Độ lệch chuẩn
	0.8	0.9	
Nhị phân	35265	30373	40256
Thực	20561	26164	2133

Bây giờ, phiên bản thực cho thấy hiệu quả trung bình tốt hơn (bảng 4.4). Ngoài ra, các kết quả của phiên bản nhị phân lần này cũng không ổn định hơn. Nhưng, để ý rằng ở đây mặc dù có mức trung bình cao, cài đặt nhị phân đã sinh ra hai kết quả tốt nhất cho lần này (16205 và 16189).

4.3.3. Các phép toán khác

Trong phần này ta quyết định cài đặt và dùng các phép toán bổ sung nếu có thể định nghĩa được dễ dàng trong cả hai không gian biểu diễn.

PHIÊN BẢN NHỊ PHÂN

Ngoài những phép toán đã được mô tả như trên, ta cài đặt phép lai nhiều điểm, và cũng cho phép lai trong số các bit của một phần tử. Phép lai nhiều điểm có xác suất áp dụng vào một phần tử được điều khiển bởi tham số hệ thống (là 0.3).

PHIÊN BẢN THỰC

Ở đây ta cũng cài đặt phép lai nhiều điểm tương tự. Ngoài ra, ta cũng cài đặt phép lai số học một và nhiều điểm; chúng tính trung bình các giá trị của hai phần tử chứ không trao đổi chúng, tại các điểm chọn. Những phép toán đó có đặc điểm là các phần tử của các

nhiễm sắc thể mới vẫn thuộc miền xác định của chúng. Hai chương sau sẽ cung cấp thêm chi tiết về các phép toán này.

KẾT QUẢ THỰC NGHIỆM

Ở đây, phiên bản thực cho thấy một ưu thế nổi bật (bảng 4.5); mặc dù các kết quả tốt nhất không khác nhau bao nhiêu, chỉ phiên bản thực vẫn tiếp tục đạt được chúng.

Bảng 4.5. Kết quả trung bình là hàm xác suất của cập nhật nhiễm sắc thể

Phiên	Xác suất cập nhật nhiễm sắc thể			Độ lệch	Tốt nhất
Bản	0.7	0.8	0.9	chuẩn	
Nhị phân	23814	19234	27456	6078	16188.2
Thực	16248	16798	16198	54	16182.1

4.4. Hiệu quả về thời gian

Có nhiều nhận xét cho rằng thời gian chạy GA giải các bài toán không tầm thường quá cao. Trong phần này ta sẽ so sánh hiệu quả thời gian của hai phiên bản. Kết quả được trình bày trong bảng 4.6 là kết quả của các lần chạy ở phần 4.3.3.

Bảng 4.6. Thời gian CPU (*giây*) là hàm của số phần tử.

Phiên	Số các phần tử (N)				
bản	5	15	25	35	45
Nhị phân	1080	3123	5137	7177	9221
Thực	184	398	611	823	1072

Bảng 4.6 so sánh thời gian CPU của cả hai phiên bản trên số phần tử trên một nhiễm sắc thể. Phiên bản thực nhanh hơn nhiều ngay cả đối với số bit vừa phải là 30 cho mỗi biến trong cài đặt nhị phân. Đối với các miền lớn và độ chính xác cao, chiều dài toàn bộ của nhiễm sắc thể tăng lên, và khác biệt tương đối cũng tăng lên, như sẽ được trình bày trong bảng 4.7.

4.5. Kết luận

Bảng 4.7. Thời gian CPU (*giây*) là hàm theo số bit của mỗi phần tử; $N = 45$

Phiên	Số bit trên mỗi phần tử nhị phân					
Bản	5	10	20	30	40	50
Nhị phân	4426	5355	7438	9219	10981	12734
Thực	1072 (hằng số)					

Các thực nghiệm trên cho thấy biểu diễn chấm động nhanh hơn, nhất quán hơn. Và cho độ chính xác cao hơn (nhất là với các miền xác định rộng mà mã hóa nhị phân phải cần một chuỗi dài để biểu diễn). Đồng thời kết quả của nó được nâng cao bởi các phép toán đặc biệt để đạt được độ chính xác cao (thậm chí cao hơn trong biểu diễn nhị phân). Ngoài ra, biểu diễn chấm động do bản chất đã gần hơn với không gian bài toán nên dễ thiết kế các phép toán khác kết hợp với tri thức bài toán - đặc tả hơn. Điều này cần thiết cho việc xử lý các ràng buộc bài toán đặc tả, không tầm thường (chương 6).

Những kết luận này giải thích tại sao người sử dụng các kỹ thuật tiến hóa di truyền thích biểu diễn chấm động hơn, như trong: (1) thoải mái với tương quan một biến - một gen, (2) tránh được các dốc Hamming và các thao tác khác trong việc thực hiện đột biến trên các chuỗi bit được xử lý như các số nguyên nhị phân không dấu, (3) có ít thế hệ hơn.

Tuy nhiên, nếu số biến không quá lớn, nhỏ hơn 10, và chấp nhận trước một ít sai số, phiên bản nhị phân thực sự vẫn hiệu quả hơn. Vì thế, bạn nên thực hiện một vài thử nghiệm. Chọn một số hàm thử nghiệm và thử nghiệm với ba hệ thống dựa trên GA với các biểu diễn nhị phân, Gray, và chấm động. Hơn nữa, bạn cũng có thể sử dụng GA kết hợp với leo đồi hay 1 số kỹ thuật khác có thể tăng độ chính xác gần với kết quả thực.

CHƯƠNG 5

BÀN THÊM VỀ PHÉP ĐỘT BIẾN KHÔNG ĐỒNG BỘ

Thuật giải di truyền thể hiện những khó khăn vốn có trong việc thực hiện tìm kiếm cục bộ cho các ứng dụng số. Holland đề nghị rằng thuật giải di truyền nên được dùng như một bộ tiền xử lý, trước khi giao việc xử lý tìm kiếm cho một hệ thống có khả năng sử dụng tri thức về miền, để hướng dẫn việc tìm kiếm cục bộ.

Tìm kiếm cục bộ cần sử dụng các lược đồ bậc cao và chiều dài xác định lớn hơn những gì được đề nghị trong Lý thuyết sơ đồ. Ngoài ra, có những bài toán có miền tham số không bị giới hạn, số biến lớn, và cần độ chính xác cao, những yêu cầu này có nghĩa là chiều dài của vectơ lời giải nhị phân rất lớn (đối với 100 biến, với các miền trong khoảng [-500, 500], cần độ chính xác 6 số lẻ, thì chiều dài của vectơ lời giải nhị phân là 3000). Như đã trình bày trong chương trước, thuật giải di truyền thực hiện những bài toán như thế rất kém hiệu quả.

Để cải thiện khả năng tìm chính xác của thuật giải di truyền - là điều bắt buộc của các bài toán cần độ chính xác cao - ta thiết kế một toán tử đột biến đặc biệt mà hiệu quả của nó khác hẳn đột biến truyền thống. Nhớ lại rằng một đột biến truyền thống thay đổi mỗi lần một bit của nhiễm sắc thể; vì vậy, thay đổi đó chỉ dùng tri thức cục bộ - chỉ biết đến bit đang bị đột biến. Nếu bit đó nằm ở phần bên trái của chuỗi mã hóa một biến, thì tác động đột biến trên biến đó rất có ý nghĩa. Ngược lại, những bit ở tít đầu bên phải của chuỗi lại có tác động hoàn toàn nhỏ khi đột biến. Ta quyết định dùng tri thức toàn cục về vị trí theo cách sau đây: khi quần thể già đi, những bit nằm ở tít bên phải của mỗi chuỗi mã hóa biến có xác suất được

đột biến cao hơn, trong khi những bit nằm trái nhất lại có xác suất giảm. Nói cách khác, một đột biến như thế tạo ra tìm kiếm toàn cục của không gian tìm kiếm lúc bắt đầu tiến trình lặp, nhưng càng về sau thì khai thác cục bộ lại tăng lên. Ta gọi đây là đột biến không đồng nhất và sẽ bàn về nó trong chương này.

Trước tiên, ta nói về các bài toán được dùng thử nghiệm cho toán tử mới này.

5.1. Các trường hợp thử nghiệm

Nói chung, thiết kế và cài đặt thuật giải di truyền giải những bài toán điều khiển tối ưu là rất khó. Trong chương này, chúng tôi thực nghiệm thuật giải di truyền với đột biến không đồng nhất trên ba bài toán điều khiển tối ưu thời gian – rời rạc: bài toán tuyến tính bình phương, bài toán thu hoạch và bài toán xe kéo đã được rời rạc hóa.

5.1.1. Bài toán tuyến tính bình phương

Bài toán thử nghiệm đầu tiên là mô hình tuyến tính bình phương một chiều:

$$\min q.x_N^2 + \sum_{k=0}^{N-1} (s.x_k^2 + r.u_k^2) \quad (5.1)$$

với:

$$x_{k+1} = ax_k + b^*u_k \ , \ k = 0,1,..., N\text{-}1 \quad (5.2)$$

Trong đó x_0 cho trước, a, b, q, s, r là các hằng cho trước, $x_k \in R$ là trạng thái còn $u_k \in R$ là điều khiển của hệ thống.

Lời giải chính xác của 5.1 thỏa 5.2 là:

$$J^* = K_0 x_0^2 \quad (5.3)$$

ở đây K_k là nghiệm của phương trình Ricati:

$$K_k = s + ra^2K_k + 1/(r + b^2K_{k+1}),\ K_N = q \quad (5.4)$$

Hệ quả là, bài toán (5.1) dẫn tới (5.2) sẽ được giải với các tập tham số trong bảng 5.1.

Trường hợp	N	x_0	s	r	q	a	b
I	45	100	1	1	1	1	1
II	45	100	10	1	1	1	1
III	45	100	100	1	1	1	1
IV	45	100	1	10	1	1	1
V	45	100	1	1000	1	1	1
VI	45	100	1	1	0	1	1
VII	45	100	1	1	1000	1	1
VIII	45	100	1	1	1	0.01	1
IX	45	100	1	1	1	1	0.01
X	45	100	1	1	1	1	100

Trong thử nghiệm giá trị của N là 45

5.1.2. Bài toán thu hoạch

Bài toán thu hoạch được định nghĩa là:

$$\max \sum_{k=0}^{N-1} \sqrt{u_k} \quad (5.5)$$

với phương trình tăng trưởng:

$$x_{k+1} = a^* x_k - u_k \quad (5.6)$$

và một ràng buộc là:

$$x_0 = x_N \quad (5.7)$$

với trạng thái khởi đầu cho trước là x_0 , a là hằng số, và $x_k \in R$, $u_k \in R^+$ theo thứ tự là trạng thái và biến điều khiển (không âm).

Lời giải tối ưu chính xác J^* của bài toán (5.5) thỏa (5.6) và (5.7) là:

$$J^* = \sqrt{\frac{x_0.(a^N - 1)^2}{a^{N-1}.(a-1)}} \quad (5.8)$$

Bài toán (5.5) thỏa (5.6) và (5.7) sẽ được giải với a = 1.1, x_0 = 100, và các giá trị N = 2, 4, 10, 20 và 45.

5.1.3. Bài toán xe kéo

Vấn đề của bài toán xe kéo là cực đại hóa khoảng cách du hành tổng cộng $x_1(N)$ trong một thời gian cho trước (tức là một đơn vị) trừ tổng các cố gắng.

$$x_1(k+1) = x_2(k) \quad (5.9)$$

$$x_2(k+1) = 2x_2(k) - x_1(k) + 1/N^2 * u(k) \quad (5.10)$$

và chỉ số hiệu quả cần cực đại hóa là:

$$x_1(N) - \frac{1}{2N} \sum_{k=o}^{N-1} u^2(k) \quad (5.11)$$

Với bài toán này, giá trị tối ưu của chỉ số trong (5.11) là:

$$J^* = \frac{1}{3} - \frac{3N-1}{6N^2} - \frac{1}{2N^3} \sum_{k=o}^{N-1} k^2$$

Bài toán xe kéo được giải với các giá trị N = 5, 10, 15, 20, 25, 30, 35, 40 và 45.

5.2. Chương trình tiến hóa giải bài toán tối ưu hóa số

Chương trình tiến hóa ta đã xây dựng cho các bài toán tối ưu số được dựa trên biểu diễn thực, và một số toán tử di truyền mới (chuyên biệt hóa); ta sẽ lần lượt bàn về chúng.

5.2.1. Biểu diễn

Trong biểu diễn thực, mỗi vectơ nhiễm sắc thể được mã hóa thành vectơ thực có cùng chiều dài với vectơ lời giải. Mỗi phần tử được chọn lúc khởi tạo sao cho thuộc miền xác định của nó, và các toán tử được thiết kế cẩn thận để bảo toàn các ràng buộc này (không có vấn đề như vậy trong biểu diễn nhị phân, nhưng thiết kế của các toán tử này khá đơn giản; ta không thấy điều đó là bất lợi; mặt khác, nó lại cung cấp các lợi ích khác được trình bày dưới đây).

Sự chính xác của cách tiếp cận như thế chỉ tùy thuộc máy tính, nhưng nói chung là tốt hơn nhiều so với biểu diễn nhị phân. Đương nhiên, ta luôn có thể tăng độ chính xác của biểu diễn nhị phân khi thêm các bit, nhưng điều này làm thuật giải chậm một cách đáng kể, như đã thảo luận trong chương trước.

Thêm nữa, biểu diễn thực có khả năng biểu diễn một miền rất rộng (hoặc các trường hợp miền xác định không biết trước cụ thể). Mặt khác, trong biểu diễn nhị phân, độ chính xác sẽ giảm khi tăng kích thước miền, do chiều dài nhị phân cố định cho trước. Hơn nữa, với biểu diễn thực, việc thiết kế các công cụ đặc biệt để xử lý các ràng buộc không tầm thường sẽ dễ hơn.

5.2.2. Các toán tử chuyên biệt hóa

Các toán tử ta sẽ sử dụng rất khác các toán tử cổ điển, vì chúng làm việc trong một không gian khác (có giá trị thực). Hơn nữa, một vài toán tử không đồng bộ, nghĩa là hành động của chúng phụ thuộc vào tuổi của quần thể.

NHÓM TOÁN TỬ ĐỘT BIẾN:

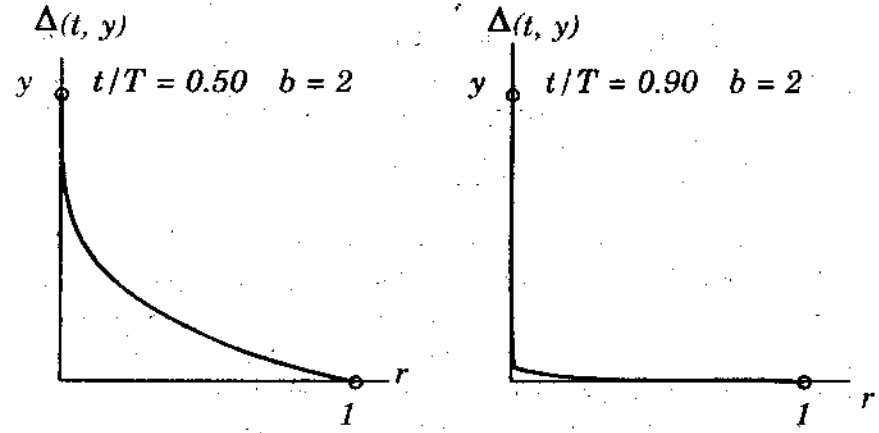
- **Đột biến đồng bộ**, được định nghĩa tương tự với định nghĩa của phiên bản cổ điển: nếu $x^t_i = \langle v_1, \ldots, v_n \rangle$ là nhiễm sắc thể, thì mỗi phần tử v_k có cơ hội trải qua tiến trình đột biến ngang nhau. Kết quả của một lần ứng dụng của toán tử này là vectơ $s^t_i = \langle v_1, \ldots, v'_k, \ldots, v_m \rangle$ và v'_k là giá trị ngẫu nhiên trong miền của tham số tương ứng.
- **Đột biến không đồng bộ** là một trong những toán tử có nhiệm vụ về tìm độ chính xác của hệ thống. Nó được định nghĩa như sau: nếu $s^t_i = \langle v_1, \ldots, v_m \rangle$ là nhiễm sắc thể và phần tử v_k được chọn đột biến này (miền của v_k là $[l_k, u_k]$), kết quả là một vectơ $s^{t+1}_v = \langle v_1, \ldots, v'_k, \ldots, v_m \rangle$ với $k \in [1, \ldots, n]$ và

$$v'_k = \begin{cases} v_k + \Delta(t,\ u_k - v_k) & \text{, nếu chữ số ngẫu nhiên là 0} \\ v_k - \Delta(t,\ v_k - l_k) & \text{, nếu chữ số ngẫu nhiên là 1} \end{cases}$$

trong đó, hàm $\Delta(t, y)$ trả về giá trị trong khoảng $[0, y]$ sao cho xác suất của $\Delta(t, y)$ gần bằng 0 sẽ tăng khi t tăng. Xác suất này buộc toán tử tìm kiếm không gian thoạt đầu là đồng bộ (khi t nhỏ), và rất cục bộ ở những giai đoạn sau. Ta sử dụng hàm sau:

$$\Delta(t, y) = y * (1 - r^{(1 - \frac{t}{T})^b}),$$

với r là số ngẫu nhiên trong khoảng [0..1], T là số thế hệ tối đa, và b là tham số hệ thống xác định mức độ không đồng bộ. Hình 5.1 biểu diễn giá trị của Δ đối với hai lần được chọn; hình này hiển thị rõ ràng cách ứng xử của toán tử.



Hình 5.1. $\Delta(t,y)$ đối với hai lần chọn.