Trí tuệ nhân tạo, Đinh Mạnh Tường, NXB Khoa học và Kỹ thuật, Hà Nội 2002 (trang 54-58)

Tìm đối tượng tốt nhất

Trong mục này chúng ta sẽ xét vấn đề tìm kiếm sau. Trên không gian tìm kiếm U được xác định hàm giá (hàm mục tiêu) cost, ứng với mỗi đối tượng x ∈ U với một giá trị số cost(x), số này được gọi là giá trị của x. Chúng ta cần tìm một đối tượng mà tại đó hàm giá trị lớn nhất, ta gọi đối tượng đó là đối tượng tốt nhất. Giả sử không gian tìm kiếm có cấu trúc cho phép ta xác định được khái niệm lân cận của mỗi đối tượng. Chẳng hạn, U là không gian trạng thái thì lân cận của trạng thái u gồm tất cả các trạng thái v kề u; nếu U là không gian các vectơ thực n-chiều thì lân cận của vectơ $x = (x_1, x_2, ... x_n)$ gồm tất cả các vectơ ở gần x theo khoảng cách Ơcơlit thông thường.

Trong mục này, ta sẽ xét kỹ thuật tìm kiếm leo đồi để tìm đối tượng tốt nhất. Sau đó ta sẽ xét kỹ thuật tìm kiếm gradient (gradient search). Đó là kỹ thuật leo đồi áp dụng cho không gian tìm kiếm là không gian các vectơ thực n-chiều và hàm giá là là hàm khả vi liên tục. Cuối cùng ta sẽ nghiên cứu kỹ thuật tìm kiếm mô phỏng luyện kim(simulated annealing).

Tìm kiếm leo đồi

Kỹ thuật tìm kiếm leo đồi để tìm kiếm đối tượng tốt nhất hoàn toàn giống như kỹ thuật tìm kiếm leo đồi để tìm trạng thái kết thúc đã xét ở trên. Chỉ khác là trong thuật toán leo đồi ở trên, từ một trạng thái ta "leo lên" trạng thái kề tốt nhất (được xác định bởi hàm giá), tiếp tục cho tới khi đạt tới trạng thái đích; nếu chưa đạt tới trạng thái đích mà không leo lên được nữa, thì ta tiếp tục "tụt xuống" trạng thái trước nó, rồi lại leo lên trạng thái tốt nhất còn lại. Còn ở đây, từ một đỉnh u ta chỉ leo lên đỉnh tốt nhất v (được xác định bởi hàm giá cost) trong lân cận u nếu đỉnh này "cao hơn" đỉnh u, tức là cost(v) > cost(u). Quá trình tìm kiếm sẽ dừng lại ngay khi ta không leo lên đỉnh cao hơn được nữa.

Trong thủ tục leo đồi dưới đây, biến u lưu đỉnh hiện thời, biến v lưu đỉnh tốt nhất (cost(v) nhỏ nhất) trong các đỉnh ở lân cận u. Khi thuật toán dừng, biến u sẽ lưu trong đối tượng tìm được.

Procedure Hill_Climbing;

Begin

1. u ← một đối tượng ban đầu nào đó;

2. **loop do**

2.1 v ← đối tượng tốt nhất (được xác định bởi hàm giá) trong lân cận u;

2.2 **if** cost(v) > cost(u) **then** u ← v **else** stop;

end;

Tối ưu địa phương và tối ưu toàn cục

Rõ ràng là, khi thuật toán leo đồi dừng lại tại đối tương u*, thì giá của nó cost(u*) lớn hơn giá của tất cả các đối tượng nằm trong lân cận của tất cả các đối tượng trên đường đi từ đối tượng ban đầu tới trạng thái u*. Do đó nghiệm u* mà thuật toán leo đồi tìm được là ***tối ưu địa phương***. Cần nhấn mạnh rằng không có gì đảm bảo nghiệm đó là ***tối ưu toàn cục*** theo nghĩa là cost(u*) là lớn nhất trên toàn bộ không gian tìm kiếm.

Để nhận được nghiệm tốt hơn bằng thuật toán leo đồi, ta có thể áp dụng lặp lại nhiều lần thủ tục leo đồi xuất phát từ một dãy các đối tượng ban đầu được chọn ngẫu nhiên và lưu lại nghiệm tốt nhất qua mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm được nghiệm tối ưu.

Kết quả của thuật toán leo đồi phụ thuộc rất nhiều vào hình dáng của "mặt cong" của hàm giá. Nếu mặt cong chỉ có một số ít cực đại địa phương, thì kỹ thuật leo đồi sẽ tìm ra rất nhanh cực đại toàn cục. Song có những vấn đề mà mặt cong của hàm giá tựa như lông nhím vậy, khi đó sử dụng kỹ thuật leo đồi đòi hỏi rất nhiều thời gian.

Tìm kiếm gradient

Tìm kiếm gradient là kỹ thuật tìm kiếm leo đồi để tìm giá trị lớn nhất (hoặc nhỏ nhất) của hàm khả vi liên tục *f(x)* trong không gian các vectơ thực n-chiều. Như ta đã biết, trong lân cận đủ nhỏ của điểm $x = (x_1, .., x_n)$, thì hàm f tăng nhanh nhất theo hướng của vectơ gradient:

$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n})$$

Do đó tư tưởng của tìm kiếm gradient là từ một điểm ta đi tới điểm ở lân cận nó theo hướng của vectơ gradient.

Procedure Gradient_Search;

Begin

x ← điểm xuất phát nào đó;

repeat

$x \leftarrow x + \alpha \nabla f(x);$

until $|\nabla f| < \varepsilon$;

end;

Trong thủ tục trên, α là hằng số dương nhỏ nhất xác định tỉ lệ của các bước, còn ε là hằng số dương nhỏ xác định tiêu chuẩn dừng. Bằng cách lấy các bước đủ nhỏ theo hướng của vectơ gradient chúng ta sẽ tìm được điểm cực đại địa phương, đó là điểm mà tại đó $\nabla f=0$, hoặc tìm được điểm rất gần với cực đại địa phương.

Tìm kiếm mô phỏng luyện kim

Như đã nhấn mạnh ở trên, tìm kiếm leo đồi không đảm bảo cho ta tìm được nghiệm tối ưu toàn cục. Để cho nghiệm tìm được gần với tối ưu toàn cục, ta áp dụng kỹ thuật leo đồi lặp xuất phát từ các điểm được lựa chọn ngẫu nhiên. Bây giờ thay cho việc luôn luôn "leo lên đồi" xuất phát từ các điểm khác nhau, ta thực hiện một số bước "tụt xuống" nhằm thoát ra khỏi các điểm cực đại địa phương. Đó chính là tư tưởng của kỹ thuật tìm kiếm mô phỏng luyện kim.

Trong tìm kiếm leo đồi, khi ở một trạng thái u ta luôn luôn đi tới trạng thái tốt nhất trong lân cận nó. Còn bây giờ, trong tìm kiếm mô phỏng luyện kim, ta chọn ngẫu nhiên một trạng thái v trong lân cận u. Nếu trạng thái v được chọn tốt hơn u (cost(v) > cost(u)) thì ta đi tới v, còn nếu không ta chỉ đi tới v với một xác suất nào đó. Xác suất này giảm theo hàm mũ của "độ xấu" của trạng thái v. Xác suất này còn phụ thuộc vào tham số nhiệt độ T. Nhiệt độ T càng cao thì bước đi tới trạng thái xấu càng có khả năng được thực hiện. Trong quá trình tìm kiếm, tham số nhiệt độ T giảm dần tới không. Khi T gần không, thuật toán hoạt động gần giống như leo đồi, hầu như nó không thực hiện bước tụt xuống. Cụ thể ta xác định xác suất đi tới trạng thái xấu v từ u là $e^{\Delta/T}$, ở đây

$$\Delta = \text{cost}(v) - \text{cost}(u).$$

Sau đây là thủ tục mô phỏng luyện kim.

Procedure Simulated_Anneaning;

Begin

$t \leftarrow 0$;

u ← trạng thái ban đầu nào đó;

T ← nhiệt độ ban đầu;

repeat

v ← trạng thái được chọn nhẫu nhiên trong lân cận u;

if cost(v) > cost(u) **then** u ← v

else u ← v với xác suất $e^{\Delta/T}$;

T ← g(T, t);

t ← t + 1;

until T đủ nhỏ

end;

Trong thủ tục trên, hàm g(T, t) thỏa mãn điều kiện g(T, t) < T với mọi t, nó xác định tốc độ giảm của nhiệt độ T. Người ta chứng minh được rằng, nếu nhiêt độ T giảm đủ chậm, thì thuật toán sẽ tìm được nghiệm tối ưu toàn cục. Thuật toán mô phỏng luyện kim đã được áp dụng thành công cho các bài toán tối ưu cỡ lớn.