TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

——————— * ———————

BÀI TẬP LỚN

MÔN: PROJECT I - IT3910

TỐI ƯU HÓA THAM SỐ HÀM

Sinh viên thực hiện : MSSV

Phan Văn Duy	**20121403**
Vũ Trường Giang	**20141265**
Trần Thị Hải Hà	**20141322**
Đỗ Minh Hải	**20141349**
Nguyễn Bá Hiến	**20141489**

Giáo viên hướng dẫn : **PGS.TS. Trần Đình Khang**

Hà Nội, tháng 11 năm 2016

MỤC LỤC

LỜI NÓI ĐẦU

Máy tính là một sáng chế rất tuyệt vời của con người vì nó là thiết bị có khả năng xử lý dữ liệu rất nhanh và hiệu quả đồng thời làm việc không biết mệt mỏi. Chính vì vậy việc khai thác khả năng làm việc của máy tính để giải quyết các bài toán của con người đang ngày càng trở nên vô cùng phổ biến.

Với sự gợi ý của giáo viên hướng dẫn, niềm yêu thích toán học và sự hiếu kỳ về khả năng xử lý của máy tính, các thành viên trong nhóm chúng em đã lựa chọn đề tài Tối ưu tham số hàm này để làm bài tập lớn cho Project I.

Để hoàn thành được bài tập lớn này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến Thầy giáo hướng dẫn đề tài - **Phó giáo sư Tiến Sĩ Trần Đình Khang** , Giảng viên Khoa Công nghệ Thông tin Trường Đại học Bách Khoa Hà Nội - đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này.

Hà Nội, tháng 11 năm 2016

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

- ✓ Phan Văn Duy :Xây dựng cơ sở dữ liệu
- ✓ Trần Thị Hải Hà : Xây dựng hàm tính toán giá trị biểu thức và viết báo cáo.
- ✓ Vũ Trường Giang và Đỗ Minh Hải : Xây dựng hàm tối ưu tham số k và thiết kế giao diện.
- ✓ Nguyễn Bá Hiến : Kiểm thử và soát lỗi.

CHƯƠNG 1. KHẢO SÁT, MÔ TẢ YÊU CẦU BÀI TOÁN

1.1. Mô tả yêu cầu bài toán

Xây dựng chương trình tính toán và tối ưu tham số của các hàm toán học (reference relation) có chứa tham số dạng y = f(x).

- Hoàn thành các yêu cầu sau:
  - Thiết kế cơ sở dữ liệu và nhập các mẫu dữ liệu vào cơ sở dữ liệu.
  - Tính toán các tham số k trong reference ralation sao cho sai số nhỏ nhất.

- Chương trình thực hiện các thao tác:
  - Hiển thị các mẫu dữ liệu và các bộ (x,y) tương ứng.
  - Cho phép người dùng lựa chọn mẫu đầu vào.
  - Tính toán các tham số k của mẫu sao cho sai số nhỏ nhất.
  - Hiển thị kết quả tính toán được kèm theo sai số tương ứng.

1.2. Phương hướng giải quyết bài toán

- Thiết kế cơ sở dữ liệu gồm các mẫu và các bộ (x,y) tương ứng thành các bảng - sử dụng mô hình cơ sở dữ liệu quan hệ, quản trị bằng ngôn ngữ sql.
- Xây dựng chương trình bằng ngôn ngữ lập trình java, yêu cầu:

  - Chương trình kết nối được với cơ sở dữ liệu đã xây dựng.

  - Chuyển đổi các mẫu reference relation ở dạng trung tố về dạng hậu tố bằng Ký pháp Balan ngược.

  - Tính toán tham số k sao cho sai số nhỏ nhất bằng cách sử dụng thuật toán di chuyền.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT ÁP DỤNG

2.1. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu (CSDL) là tập hợp dữ liệu biểu diễn trừu tượng các đối tượng trong thế giới thực, có liên hệ logic thống nhất, được thiết kế có cấu trúc và bao gồm các dữ liệu phục vụ một mục đích nào đó.

Quản lý dữ liệu bằng CSDL giúp dữ liệu được tổ chức moojt cách hiệu quả và có tổ chức, cho phép lưu trữ dữ liệu chính xác và truy xuất dữ liệu hiệu quả.
Có rất nhiều loại CSDL :

- Cơ sở dữ liệu dạng file
- Cơ sở dữ liệu quan hệ
- Cơ sở dữ liệu hướng đối tượng
- Cơ sở dữ liệu bán cấu trúc
- …

Để thiết kê cơ sở dữ liệu cho đề tài lần này, nhóm chúng em lựa chọn xây dựng cơ sở dữ liệu dạng quan hệ để lưu trữ và truy xuất dữ liệu các mẫu reference relation

Trong cơ sở dữ liệu quan hệ dữ liệu được lưu trữ trong các bảng dữ liệu gọi là thực thể , giữa các thực thể này có mối liên hệ với nhau được gọi là các quan hệ, mỗi quan hệ có các thuộc tính, trong đó có một thuộc tính là khóa chính.
Để xây dựng và thao tác với cơ sở dữ liệu dạng quan hệ ta sử dụng ngôn ngữ dữ liệu SQL.

SQL (Structured Query Language - ngôn ngữ truy vấn mang tính cấu trúc) là một loại ngôn ngữ máy tính phổ biến để tạo, sửa và lấy dữ liệu từ một cơ sở dữ liệu dạng quan hệ.

- Cú pháp câu lệnh tạo bảng SQL:

```
CREAT TABLE table_name(
        column_1 type1(size1) [NOT NULL],
        column_2 type2(size2) [NOT NULL],

        …
        column_n typen(sizen) [NOT NULL]
        [CONSTRAINT <constraint_name><constraint_type> clause]
);
```

trong đó:

- table_name : tên bảng(quan hệ)
- column_i : tên cột (trường)
- type : kiểu dữ liệu (int, float, varchar,...)
- size : kích thước trường dữ liệu

- []: thành phần không bắt buộc phải có trong câu lệnh.
- CONTRAINT : quy ước kiểu rằng buộc.

➢ Cú pháp câu lệnh cập nhật dữ liệu SQL:

INSERT INTO table_name[(col_1, col_2,..)]
VALUES ('value_1',value_2',...) ;

➢ Cú pháp câu lệnh truy vấn SQL:

SELECT [DISTINCT] <field_1>, <field_2>,...
FROM <table_1>, <table_2>,...
[WHERE <conditions to choose>]
[GROUP BY <field_1>, <filed_2>,...]
[ORDER BY <field> [ASC | DESC]]
[HAVING <conditions to print >]

2.2. Ký pháp BaLan ngược

2.2.1. Ký pháp BaLan ngược

Một biểu thức toán học thông thường bao gồm các toán tử (+, - , *, /,...), các toán hạng và các dấu ngoặc cho biết thứ tự tính toán. Ví dụ như:

3*((5-2) * (7+2) – 9))

Trong biểu thức trên, một toán tử luôn nằm giữa hai toán hạng nên cách viết này gọi là ký pháp trung tố. Để tính giá trị biểu thức trên, ta phải ưu tiên biểu thức trong ngoặc trước rồi mới xét đến ưu tiên toán tử. Đôi khi ta cần lưu rất nhiều giá trị trung gian cho các ngoặc này rồi sử dụng chúng như các toán hạng còn lại.
Ví dụ với biểu thức trên, ta cần tính:

- 5 – 2 = 3
- 7 + 2 = 9
- 3 * 9 = 27
- 27 – 9 = 18
- 3 * 18 = 54

Dễ có thể nhận thấy, trong ký pháp trung tố, vị trí dấu ngoặc rất quan trọng, nếu vị trí dấu ngoặc thay đổi thì giá trị của cả biểu thức sẽ thay đổi theo.
Mặc dù với con người, hình thức này có vẻ rất hợp lý, song với máy tính, việc tính toán những biểu thức như vậy phức tạp. Để dễ dàng hơn cho máy tính trong việc tính toán biểu thức, người ta đã đưa ra một hình thức trình bày khác cho biểu thức toán học, gọi là ký pháp BaLan ngược.

Ký pháp BaLan (PN - Polish notation) do nhà logic toán Jan Łukasiewicz đề xuất vào khoảng năm 1920, còn gọi là ký pháp tiền tố (prefic notation), là một hình thức ký hiệu cho logic, số học và đại số, và được phân biệt bằng cách đặt tất cả các toán tử ở bên trái các toán hạng của chúng. Đặc điểm cơ bản của hình thức này là không cần dùng đến dấu ngoặc và luôn thực hiện từ trái sang phải. Nhờ vậy nên nếu các đối số là cố định thì cú pháp thiếu dấu ngoặc vẫn có thể phân tích không mơ hồ.
Như biểu thức trung tố trong ví dụ trên có thể viết lại dưới dạng tiền tố như sau:
3 – 9 * + 2 7 – 2 5 3

Để giảm truy nhập vào bộ nhớ máy tính và sử dụng ngăn xếp để tính toán giá trị biểu thức hiệu quả hơn, khoảng những năm 1950, Ký pháp BaLan ngịch đảo đã phát minh bởi Charles Hamblin - một triết gia và khoa học gia người Úc, dựa trên công trình nghiên cứu về Ký pháp BaLan của Jan Łukasiewicz. Ký pháp BaLan nghịch đảo (RPN - Reverse Polish notation) hay còn gọi là ký pháp hậu tố (postfic notation), là một hình thức ký hiệu đại số, trong đó các toán tử đều đứng sau các toán hạng của chúng, đảo ngược lại với ký pháp BaLan.

Ký pháp BaLan ngược cho ví dụ trên là:

3 5 2 – 7 2 + * 9 – 3 *

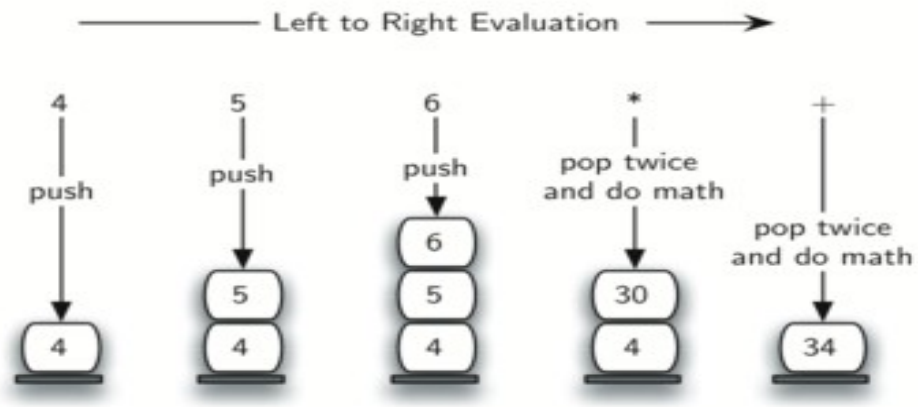
Khi đó ta có thể dễ dàng tính giá trị biểu thức bằng cách sử dụng ngăn xếp như sau: Duyệt biểu thức từ trái qua phải:

- Nếu gặp toán hạng thì đưa vào ngăn xếp.
- Nếu gặp toán tử, lấy ra 2 toán hạng từ ngăn xếp để áp dụng với toán tử, kết quả thu được lại được đưa vào ngăn xếp.

Ví dụ cho ký pháp trung tố : (4 + (5 * 6)) (có kết quả bằng 34)

Ký pháp hậu tố tương ưng là:

4 5 6 * +



Kết quả thu được cuối cùng bằng với kết quả của ký pháp trung tố đã cho.

2.2.2. Chuyển đổi từ ký pháp trung tố sang ký pháp hậu tố

Như vậy, ta có thể thấy rằng máy tính có thể tính toán dễ dàng biểu thức dạng hậu tố thông qua ngăn xếp. Tuy nhiên, với con người thì biểu thức dạng trung tố lại gần gũi và được sử dụng phổ biến hơn. Vì vậy, để con người dễ dàng trao đổi với máy tính hơn, ta vẫn nhận vào biểu thức trung tố rồi chuyển biểu thức dạng trung tố sang hậu tố cho quá trình tính toán.

Quy tắc chuyển đổi từ ký pháp trung tố sang hậu tố:

Duyệt biểu thức từ trái qua phải:

- Nếu gặp toán hạng : nạp toán hạng vào xâu
- Nếu gặp mở ngoặc : nạp mở ngoặc vào ngăn xếp
- Nếu gặp đóng ngoăc: đẩy các toán tử ra khỏi ngăn xếp cho đến khi gặp mở ngoặc đầu tiên.

- Nếu gặp toán tử: đưa ra khỏi ngăn xếp tất cả các toán tử cho đến khi gặp toán tử có thứ tự ưu tiên thấp hơn hoặc toán tử có tính kết hợp phải có cùng thứ tự ưu tiên, sau đó nạp toán tử đang xét vào ngăn xếp.
- Nếu đã duyệt hết xâu biểu thức thì đưa ra hết toàn bộ toán tử còn lại trong ngăn xếp.

Ví dụ: Cho ký pháp trung tố: 3 + 4 * 2 / (1 - 5) ^ 2^3

Ký tự	Thao tác	Ngăn xếp	Ký pháp hậu tố
3	Nạp 3 vào xâu		"3"
+	Nạp + vào ngăn xếp	+	"3"
4	Nạp 4 vào xâu	+	"3 4"
*	Nạp * vào ngăn xếp (do * có độ ưu tiên cao hơn +)	+ *	"3 4"
2	Nạp 2 vào xâu	+ *	"3 4 2"
/	Lấy * ra khỏi ngăn xếp và thêm vào xâu (vì * và / có tính kết hợp phải và có cùng thứ tự ưu tiên), sau đó nạp / vào ngăn xếp	+ /	"3 4 2 *"
(	Nạp (vào ngăn xếp	+ / (	"3 4 2 *"
1	Nạp 1 vào xâu	+ / (	"3 4 2 * 1"
-	Nạp - vào ngăn xếp	+ / (-	"3 4 2 * 1"
5	Nạp 5 vào xâu	+ / (-	"3 4 2 * 1 5"
)	Lấy - và (ra khỏi ngăn xếp, ghi - vào kết quả	+ /	"3 4 2 * 1 5 -"
^	Nạp ^ vào ngăn xếp (do ^ có thứ tuwh ưu tiên cao hơn)	+/^	"3 4 2 * 1 5 - "
2	Nạp 2 vào xâu	+/^	"3 4 2 * 1 5 - 2"
^	Nạp ^ vào ngăn xếp (do ^ và ^ có thứ tự kết hợp trái)	+/^^	"3 4 2 * 1 5 - 2"
3	Nạp 3 vào xâu	+/^^	"3 4 2 * 1 5 - 2 3"
	Duyệt hết, lấy ra toàn bộ toán tử còn lại trong ngăn xếp và nạp vào xâu		"3 4 2 * 1 5 - 2 3 ^ ^ / +"

Vậy ký pháp hậu tố tương ứng là : 3 4 2 * 1 5 - 2 3 ^ ^ / +

2.3. Thuật toán di chuyền (GAs)

Giải thuật di truyền (GTDT) là một kỹ thuật của khoa học máy tính nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu tổ hợp, là một phần của ngành giải

thuật tiến hóa, vận dụng các nguyên lý của tiến hóa như: di truyền, đột biến, chọn lọc tự nhiên và trao đổi chéo.

Giống như giải thuật tiến hóa, GTDT được hình thành trên quan niệm: *" Quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu"*. Quá trình tiến hoá thể hiện tính tối ưu ở chỗ thế hệ sau bao giờ cũng tốt hơn thế hệ trước. Vì vậy GTDT là phương pháp tìm kiếm ngẫu nhiên để tìm lời giải tối ưu, đặc biệt giải thuật này cho phép tìm kiếm được lời giải ngay cả với các không gian tìm kiếm (lời giải) không liên tục và phức tạp. Đồng nghĩa mục tiêu của GTDT không nhằm đưa ra lời giải chính xác tối ưu ,à là đưa ra lời giải tương đối tối ưu. Vậy nên thuật toán này hoàn toàn phù hợp với đề tài tối ưu hóa tham số hàm.

- Các thành phần cấu thành giải thuật di truyền:
  - Mỗi một giải pháp của bài toán được gọi là một cá thể hay một nhiễm sắc thể(NST), do trong GTDT giới hạn một cá thể có một NST nên khái niệm cá thể và NST trong GTDT có thể coi như tương đương.
  - Một NST được cấu thành từ nhiều gen, mỗi gen có giá trị khác nhau để quy định một tính trạng nào đó. Trong GTDT, một gen được coi như một phần tử trong chuỗi NST.
  - Một tập hợp các cá thể có thể có cùng một số đặc điểm được gọi là quần thể. Trong GTDT, ta quan niệm quần thể là một tập các lời giải của một bài toán.

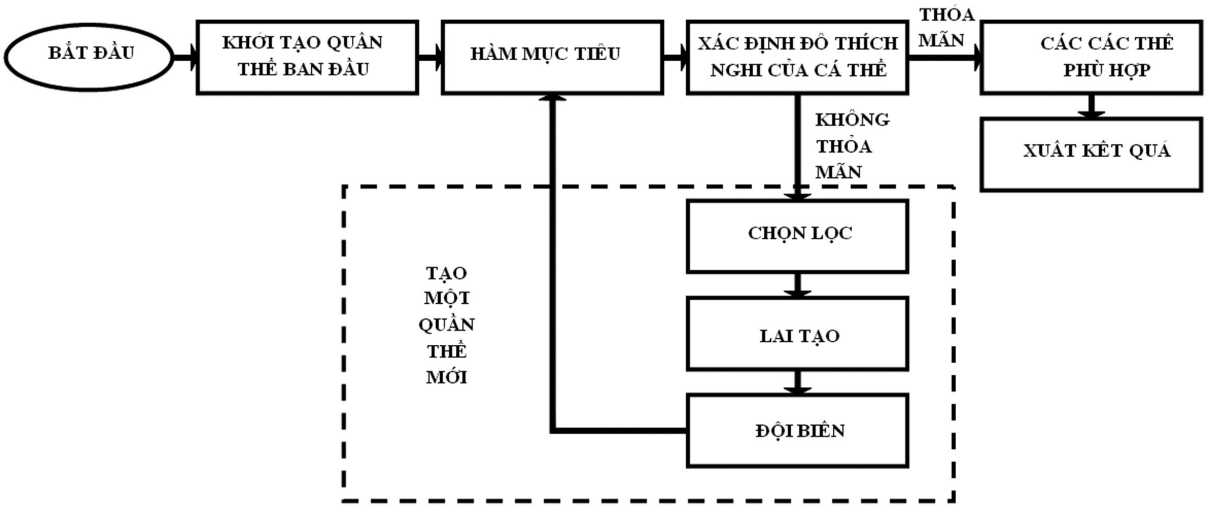
- Các toán tử di truyền:
  - Toán tử tái sản xuất (Reproduction) hay còn gọi là Toán tử chọn loc:để giữ lại cá thể tốt

  Các cá thể tốt được giữ nguyên (không thay đổi) để đưa vào thế hệ sau.
  - Lai ghép (Crossover) để sinh ra 2 cá thể mới

  Trao đổi gen của 2 cá thể cha mẹ để tạo ra 2 cá thể con, điểm lai ghép được lựa chọn một cách ngẫu nhiên.
  - Đột biến (Mutation) để sinh ra một cá thể mới

  Thay đổi một số gen để tạo thành cá thể mới.

- Mô tả giải thuật di truyền:



Như hình vẽ trên, thuật toán di truyền cơ bản gồm các bước:

- Bắt đầu: Nhận các tham số đầu vào
- Khởi tạo quần thể ban đầu: Sinh ngẫu nhiên một quần thể n cá thể

- Mỗi cá thể là một số giả thiết (giải pháp lời giải) ban đầu.
- Các cá thể là khác nhau.

- Tạo một quần thể mới:

- Đánh giá (cho điểm) mỗi cá thể của quần thể dựa trên hàm tính toán độ thích nghi(fitness).
- Kiểm tra điều kiện kết thúc
- Chọn lọc: sắp xếp các cá thể theo đánh giá và chỉ giữ lại các cá thể tốt nhất
- Sinh ra thế hệ tiếp theo nhờ lai ghép và đột biến

Lặp lại bước này cho đến khi ở một thế hệ nào đó tìm được cá thể có độ phù hợp cao hơn giá trị mong muôn.

- Chọn kết quả: Nếu điều kiện dừng được thỏa mãn thì thuật toán kết thúc và trả về lời giải tốt nhất trong quần thể hiện tại.

➢ Mô tả chi tiết từng toán tử di truyền trong GTDT cổ điển:

1. Chọn lọc

Quá trình chọn lọc được thực hiện bằng kĩ thuật qua bánh xe:

Giả sử thế hệ hiện thời P(t) gồm có n cá thể $\{x_1,\ldots,x_I\}$. Với mỗi cá thể x_i, ta tính độ thích nghi của nó là $f(x_i)$. Rồi sau đó tính tổng độ thích nghi của tất cả các cá thể trong quần thể:

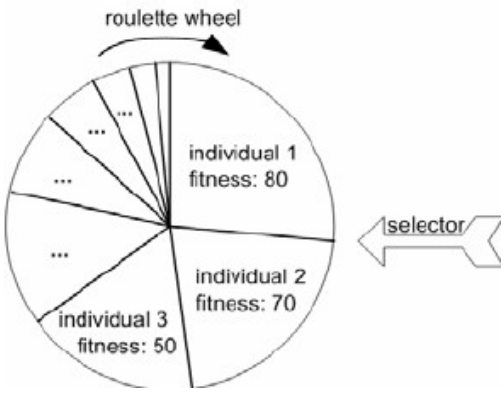
$$F = \sum_{i=1}^{n} f(x_i)$$

Để chọnlọc, ta thực hiện 2 bước sau:

- Bước 1: Sinh ra một số thực ngẫu nhiên q trong khoảng (0, F);
- Bước 2: x_k là cá thể đuowjc chọn nếu k là số nhỏ nhất sao cho:

$$F = \sum_{i=1}^{k} f(x_i) >= 4$$

Việc chọn lọc có thể minh họa như sau: Ta có một bánh xe chia làm n phần, mỗi phần ứng với độ thích nghi của một cá thể. Có một mũi tên chỉ vào bánh xe, sau khi quay bánh xe, mũi tên chỉ vào phần nào, các cá thể ứng với phần đó được chọn.



2. Lai ghép

Trên các cá thể được chọn lọc, ta tiến hành toán tử lai ghép. Đầu tiên ta đưa ra xác suất lai ghép p_c , xác suất này cho ta hi vọng có n* p_c cá thể được lai ghép.
Với mỗi cá thể ta thực hiện 2 bước sau:

- Bước 1: Sinh ra số thực ngẫu nhiên r trong đoạn [0, 1];
- Bước 2; Nếu r < p_c thì cá thể đó được chọn để lai ghép

Từ các cá thể được lai ghép, ta ghép cặp chúng một cách ngẫu nhiên.
Ví dụ trường hợp lai ghép 2 NST nhị phân dài m bits, ta chọn ngẫu nhiên một vị trí điểm ghép p trong [0, m-1] làm vị trí điểm lai ghép. Cụ thể:

$$a=(a_1,\ldots,a_p,a_{p+1},\ldots,a_m)$$
$$b=(b_1,\ldots,b_p,b_{p+1},\ldots,b_m)$$

sẽ được thay thành:

$$a=(a_1,\ldots,a_p,b_{p+1},\ldots,a_m)$$
$$b=(b_1,\ldots,b_p,a_{p+1},\ldots,b_m)$$

3. Đột biến

Ta thực hiện đột biến trên các cá thể có được sau quá trình lai ghép. Đột biến là thay đổi trạng thái một số gen nào đó trong NST. Mỗi gen chịu đột biến với âp xác suất p_m . Xác suất p_m do chúng ta quyết định và là xác suất thấp.
Ví dụ, với mỗi vị trí i trong NST a,

$$a=(a_1,\ldots,a_p,a_{p+1},\ldots,a_m)$$

ta sinh ra một số thực ngẫu nhiên p trong [0,p]. Qua đột biến thành a' như sau:

$$a'=(a'_1,\ldots,a'_p,a'_{p+1},\ldots,a'_m)$$

trong đó:

$$a'\_i = \begin{cases} a\_i & \text{nếu } p\_i >= p\_m \\ 1 - a\_i & \text{nếu } p\_i < p\_m \end{cases}$$

Sau quá trình chọn lọc, lai ghép, đột biến, một thế hệ mới được sinh ra. Công việc còn lại của thuật toán di truyền bây giờ là lặp lại các bước trên đến khi tìm được lời giải tối ưu hoặc hết số vòng lặp tối đa.

CHƯƠNG 3. GIẢI QUYẾT BÀI TOÁN

3.1. Xây dựng cơ sở dữ liệu

Trong đề tài này, cần xây dựng cơ sở dữ liệu gồm hai thực thể sau:

- Bảng Reference_Relation: là tập các mẫu bao gồm 2 thuộc tính là Mã mẫu(Refer_id) và Biểu thức của mẫu (Reference Relation)

refer_id	reference_relation
1	k*x^1.5
2	k_{1}/x+k_{2}
3	k_{1}*(x+273.1)^(0.5)+k_{2}
...	...

- Bảng :là tập các bộ (x, y) của các mẫu bao gồm 4 thuộc tính là Mã bộ (id), Mã mẫu (Refer_id) , giá trị của x (x) và giá trị của y.

id	refer_id	x	y
1	1	36	88
2	1	37.25	224.7
3	1	483.8	365.3
4	1	141.75	687
5	1	483.8	4332.1
6	2	326.75	2
7	2	300.75	4
...	...	...	...

3.2. Thiết kế chi tiết các lớp

3.2.1. Lớp kết nối cơ sở dữ liệu (Connect)

- Là lớp kết nối chương trình với cơ sở dữ liệu đã được xây dựng.
- Lớp gồm 2 phương thức:
  - **public void connec();** // hàm kết nối JDBC
  - **public void xayDungCacTapGiaTri();** // Lấy dữ liệu từ các bảng
  - **public List<String> getDsBieuThuc();**// hàm trả về danh sách mẫu
  - **public ArrayList<ArrayList<List<Double>>> getDsBoXY();**

// hàm trả về các bộ (x,y) tương ứng với các mẫu.

- Các câu lệnh kết nối cơ sở dữ liệu qua JDBC:
  - Đăng ký trình điều khiển:

    Class.forName("com.mysql.jdbc.Driver");
  - Thực hiện kết nối:

  Sử dụng một trong các phương thức sau của lớp DriverManager:

1 - **public static Connection getConnection(String url) throws SQLException;**

2 - **public static Connection getConnection(String url, String user, String pass)throws SQLException;**

3 - **public static Connection get Connection(String url, Properties info) throws SQLException;**

trong đó: url là đường dẫn tới cơ sở dữ liệu
user là tài khoản người dùng
pass là mật khẩu

Ví dụ : Connection connection = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/tri_tue_nhan_tao_4", "root", "12345678");

- Truy vấn dữ liệu:

- Tạo 1 đối tượng Statement:
Statement statement = new Statement();
- Sử dụng hàm executeQuery() của lớp Statement để thực hiện câu lệnh truy vấn SQL
ResualtSet resualtSet = statement.executeQuery("SELECT * FROM TABLE");
- Sử dụng hàm **next()** để xác định yêu cầu.

3.2.2. Lớp xử lý và tính toán xâu biểu thức (InficToPostfic)

- Là lớp xử lý xâu mẫu được chọn từ dạng ký pháp trung tố sang ký pháp dạng hậu tố và tính toán giá trị hàm.
- Lớp có 4 hàm:
  - Hàm thiết lập độ ưu tiên của toán tử:

public int priority(char c);

+ - : độ ưu tiên bằng 1
* / : độ ưu tiên bằng 2
^ , sqrt() : độ ưu tiên bằng 3
log, ln, arctan : độ ưu tiên bằng 4

  - Hàm kiểm tra một ký tự có phải là toán tử hay không:

public static boolean isOperator(char c);

  - Hàm xử lý biểu thức nhập vào thành các phần tử:

public String processString(String sMath);

input: 1 xâu dạng String là một mẫu trong CSDL
return: 1 chuỗi các xâu bao gồm các toán tử và toán hạng lấy từ xâu mẫu.

  - Hàm chuyển xâu ký pháp trung tố về dạng hậu tố:

public String postfix(String[] elementMath);

Chi tiết:
Product postfic
Begin
- Khởi tạo:
String s1 = "", E[];
Stack<String> S = new Stack<String>();

- Duyệt toàn bộ xâu từ trái sang phải:

```
for (int i=0; i<elementMath.length; i++){
char c = elementMath[i].charAt(0);
```

- Nếu gặp toán hạng: Nạp toán hạng vào xâu

```
s1= s1+elementMath[i];
```

- Nếu gặp mở ngoặc đưa mở ngoặc vào stack

```
S.push(elementMath[i]);
```

- Nếu là đóng ngoặc: đẩy các toán tử trong ngăn xếp ra ngoái cho đến khi gặp được dâu mở ngoặc đầu tiên:

```
char c1;
do{
    c1 = S.peek().charAt(0);
    if (c1 != '(') s1 = s1 + " " + S.peek();
    S.pop();
}while (c1 != '(');
```

- Nếu gặp toán tử: đưa ra khỏi ngăn xếp tất cả các toán tử cho đến khi gặp toán tử có thứ tự ưu tiên thấp hơn hoặc toán tử có tính kết hợp phải có cùng thứ tự ưu tiên, sau đó nạp toán tử đang xét vào ngăn xếp.

```
while (!S.isEmpty() &&
                    ((IFP.priority(S.peek().charAt(0))IFP.priority(c)) ||
                        ((IFP.priority(S.peek().charAt(0)) ==
                            IFP.priority(c)) && (c != '^'))){
        s1 = s1 + " " + S.peek();
        S.pop();
}
S.push(elementMath[i]);
```

- Nếu duyệt hết biểu thức: đưa toàn bộ toán tử ra khỏi Stack.

```
while (!S.isEmpty()){
    s1 = s1 + " " + S.peek();
    S.pop();
```

- Trả về kết quả.

```
    E = s1.split(" ");
    return E;
End;
```

- Hàm tính toán giá trị biểu thức:

public static Double EvaluatePostfix(String[] elementMath, double a, double[] k)

input: chuỗi các xâu ký tự của mẫu ở dạng ký pháp hậu tố, giá trị của x(a) và bộ tham số k đang xét.

return : giá trị biểu thức y' = f(x=a) trong đó các tham số k được thay bằng giá trị của bộ tham sô k đang xét.

3.2.3. Lớp tính độ thích nghi (Detaly)

- Là lớp đánh giá độ ưu tiên của mỗi bộ tham số k
- Lớp có duy nhất một phương thức là phương thức đánh giá độ thích nghi:

public double detaly(double[] x, double[] y, String sMath,double k);

input: mẫu y = f(x) (sMath), bộ (x,y) tương ứng với mẫu, bộ tham số k cần đánh giá.

return: độ thích nghi của bộ tham số k đang xét.

Chi tiết:

Production detaly
Begin
- chuyển mẫu về dạng hậu tố
IFP.EvaluatePostfix(elementMath, x[i], k);

- tính tổng sai số của bộ tham số k so với bộ (x,y) của mẫu

```
for (int i=0;i<y.length;i++){
elementMath = sMath.split(" ");
double a=IFP.EvaluatePostfix(elementMath, x[i], k);
tongDetaly = tongDetaly + Math.abs(y[i]-a)/y[i];
}
```

- trả về sai số trung bình
return detaly = detaly/y.length;

end;

3.2.4. Lớp di truyền tối ưu tham số hàm(HamGA)

- Là lớp quan trọng nhất của chương trình để tìm ra lời giải tối ưu cho đề tài, trong chương trình này là bộ giá trị tham số k.
- Thuộc tính của lớp gồm: bộ (x,y) của mẫu ta xét, xác suất lai ghép (r), xác suất đột biến (w), độ ưu tiên kỳ vọng (l), số lượng các thể của quần thể (n), số lần lặp trong mỗi trường hợp thỏa mãn độ ưu tiên mong muốn (m).
- Các phương thức của hàm:
  - Các phương thức khởi tạo
  - Các phương thức getter và setter
  - Phương thức đếm số tham số k cần tính

private int TinhSoThamSo(String sMath);

  - Phương thức tính độ thích nghi

private double uuChonLoc(double[] x2, double[] y2,
String sMath, ArrayList<double[] dsThamSo, int I);

Chi tiết:

Phương thức gọi tới hàm tính độ ưu tiên của class Detaly.

- Phương thức sắp xếp các bộ tham số k theo độ ưu tiên

public void sapXepDanhSachBoThamSoTheoDoUuTien
(double[]uuTien, ArrayList<double[]> dsBoThamSo);

- Phương thức chọn lọc

public void chonLoc(ArrayList<double[]> dsThamSo,
String sMath, double[] uuTien)

Chi tiết:
Chọn lọc n bộ tham số có độ ưu tiên tốt nhất trong danh sách các bộ tham số.

- Phương thức lai ghép

public void laiGhep(ArrayList<double[]> dsBoThamSo,
int soThamSo);

Chi tiết:
Lựa chọn (r * n/2) cặp bộ tham số từ danh sách các bộ tham số.
Từ 1 cặp bộ tham số k là (1) và (2), ta tiến hành lai ghép tạo ra 2 con là bộ (1') và (2'), trong đó:

// (1') là bộ trung bình cộng của 2 bộ (1) và (2)
k_{i}(1') = (k_{i}(1)+k_{i}(2))/2

// (2') là bộ tích chia trung bình cộng của 2 bộ (1) và (2)
k_{i}(2') = (k_{i}(1)*k_{i}(2))/k_{i}(1')

- Phương thức đột biến

public void dotBien(ArrayList<double> dsBoThamSo, int soThamSo);

Chi tiết:
- Lựa chọn (w* n) cặp bộ tham số từ danh sách các bộ tham số.
- Sau đó chọn ngẫu nhiên 1 tham số k trong bộ tham số { k1, k2, …}
rồi ngẫu nhiên lựa chọn đột biến theo một trong hai cách sau:
* Cách 1: k = k + khoảng_đột_biến*k
* Cách 2: k = k + khoảng_đột_biến*k
trong đó: khoảng_đột_biến thuộc khoảng (0.8, 0.9)

- Phương thức lặp lại quá trình di truyền GAs

public ArrayList<double[]> GAs(String sMath);

Chi tiết:
Production GAs
Begin
- Khởi tạo quần thể
- Sắp xếp danh sách bo tham số theo độ ưu tiên cho quần thể vừa được khởi tạo.
- Thực hiên vòng lặp
while (uuTien[0] >1 && lap < m){

```
int lap = 0;
laiGhep(dsBoThamSo, soThamSo);
uuTien = new double[dsBoThamSo.size()];
chonLoc(dsBoThamSo,sMath,uuTien);
dotBien(dsBoThamSo, soThamSo);
uuTien = new double[dsBoThamSo.size();
chonLoc(dsBoThamSo,sMath,uuTien);
lap++;//Tăng số lần lặp
                }
```

- Trả về kết quả:

```
return dsThamSo;
End;
```

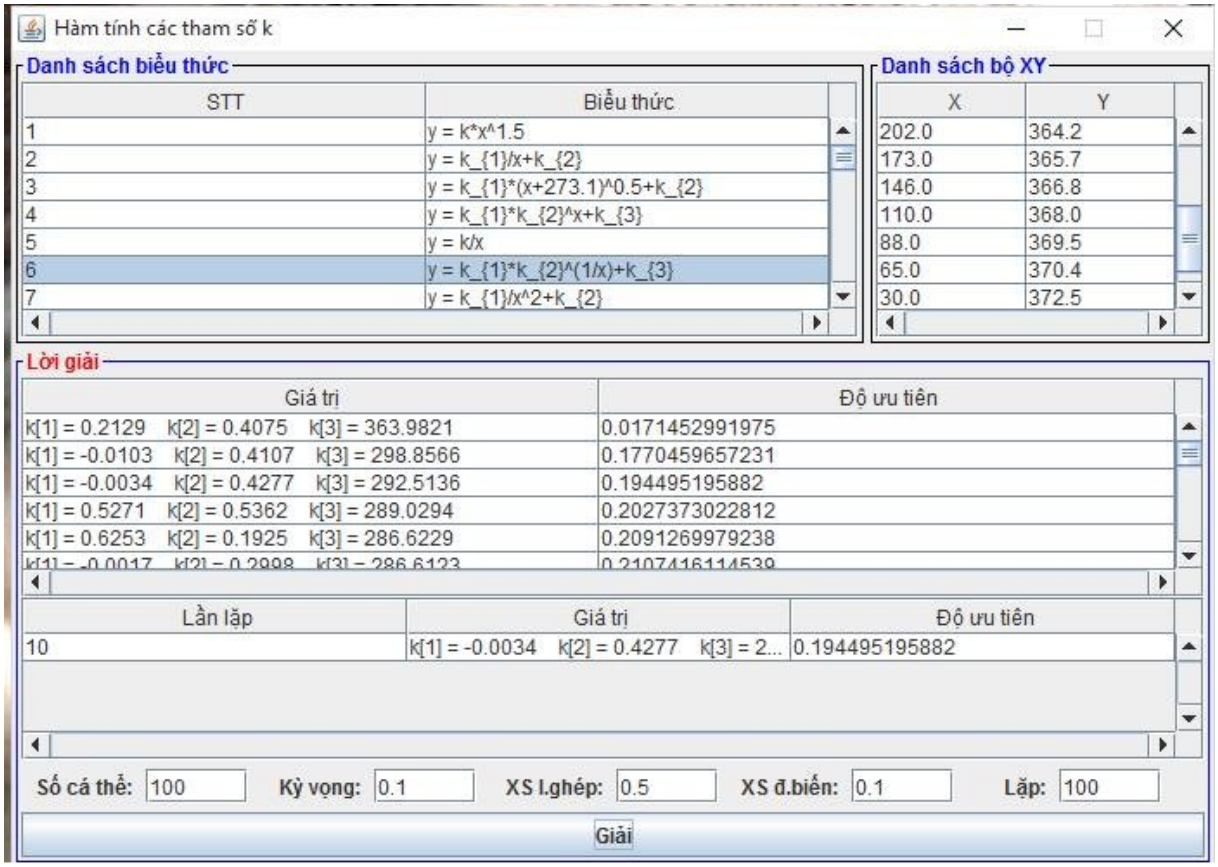
3.2.5. Một số lớp khác

- Lớp main để chạy chương trình (Test).
- Lớp tạo giao diện (UI).

CHƯƠNG 4. XÂY DỰNG CHƯƠNG TRÌNH MINH HỌA

4.1. Kết quả chương trình minh họa

Giao diện chính:



Ví dụ chọn mẫu số 6: y = k{1}*k{2}^(1/x)+k{3}
Lựa chọn thông số di truyền:

- Số cá thể : 100
- Kỳ vọng: 0.1
- Xác suất lai ghép: 0.5
- Xác suất đột biến: 0.1
- Số lần lặp tối đa: 100

Nhấn nút Giải, ta thu được kết quả như trên hình.

Bộ tham số k tối ưu tốt nhất thu được là:

k[1] = 0.2129
k[2] = 0.4075
k[3] = 363.8921

kèm theo độ ưu tiên là 0.0171452991475

Kiểm tra kết quả thu được:

Hàm có dạng: y = 0.2129 * 0.4075 ^ x+ 363.8921
Bộ (x, y) tương ứng của mẫu đã cho là:

x	y	y'
468	347	363.9133492
413	351.3	363.9133438
330	356	363.9133322
290	359	363.9133242
267	360.5	363.9133185
240	362	363.9133105
202	364.2	363.91322956
173	365.7	363.9132789
146	366.8	363.9132595
110	368	363.913217
88	369.5	363.9131739
65	370.4	363.913089
30	372.5	363.9127624
0.0001	373.3	363.8921

TÀI LIỆU THAM KHẢO

[1] Bài giảng Trí tuệ nhân tạo - Nguyễn Nhật Quang - 2015.
[2] Giáo trình Trí tuệ nhân tạo - Đinh Mạng Tường.
[3] Giải thuật di truyền (Gas) và các ứng dụng – ThS. Trần Kim Hương - Hội nghị nghiên cứ khoa học khoa sư phạm toán tin – T5/2015
[4] Wikipedia
[5] Wikipedia Tiếng Việt

PHỤ LỤC

<Phần này đưa ra hướng dẫn cài đặt, hướng dẫn sử dụng của chương trình, một số các vấn đề khác muốn trình bày…>