**Áp dụng MOEA framework giải bài toán đấu thầu nhiều vòng**

*Nguyễn Thanh Hà*  
Hanoi University of Science and Technology  
Hanoi, Vietnam  
thanhha.nguyen2211@gmail.com

*Nguyễn Thượng Khánh*  
Hanoi University of Science and Technology  
Hanoi, Vietnam  
thuongkhanh96@gmail.com

*Nguyễn Xuân Tùng*  
Hanoi University of Science and Technology  
Hanoi, Vietnam  
tungdptry@gmail.com

*Abstract: Trong bài báo này, dựa trên lý thuyết trò chơi và cân bằng Nash, chúng em sẽ sử dụng MOEA Framework để so sánh các thuật toán khác nhau trong việc giải bài toán đấu thầu nhiều vòng. Những thuật toán được sử dụng trong bài báo bao gồm: NSGA-III, ε-MOEA, GDE3, PESA2, IBEA, SMPSO. Tất cả đều có thể sử dụng để giải bài toán tuy nhiên lại cho kết quả khác nhau với thời gian khác nhau. Chúng em sẽ so sánh dựa trên thời gian và kết quả thu được để đưa ra kết luận thuật toán nào tốt hơn thuật toán nào ở trong điều kiện nào.*

Từ khóa: MOEA, đấu thầu nhiều vòng, di truyền, tiến hóa, đa mục tiêu.

# **GIỚI THIỆU**

**Đấu thầu** là một quá trình chủ đầu tư lựa chọn được một nhà thầu đáp ứng các yêu cầu của mình theo quy định của luật pháp. Trong nền kinh tế thị trường, người mua tổ chức đấu thầu để người bán (các nhà thầu) cạnh tranh nhau. Mục tiêu của người mua là có được hàng hóa và dịch vụ thỏa mãn các yêu cầu của mình về kỹ thuật, chất lượng và chi phí thấp nhất. Mục đích của nhà thầu là giành được quyền cung cấp hàng hóa dịch vụ đó với giá đủ bù đắp các chi phí đầu vào và đảm bảo mức lợi nhuận cao nhất có thể.

# **CƠ SỞ LÝ THUYẾT**

## **Lý thuyết trò chơi**

Lý thuyết trò chơi là một nhánh của toán học ứng dụng được sử dụng để phân tích các tình huống cạnh tranh mà kết quả không phụ thuộc vào sự lựa chọn của một bên hay còn là cơ hội lựa chọn của các người chơi khác. Bởi vậy, kết quả sẽ phụ thuộc vào quyết định của tất cả người chơi, trong đó mỗi người chơi sẽ cố gắng dự đoán sự lựa chọn của những người chơi còn lại để có thể đưa ra lựa chọn tốt nhất cho mình.

Trong lý thuyết trò chơi, một trò chơi được biểu diễn bởi:

*G = (N, Si, ui)*

Trong đó:

* *G* là trò chơi
* *N =* {1, 2, …, n} là tập gồm *n* người chơi
* *Si* là tập hành động của người chơi thứ *i* (*i ∈ N*)
* *S =*{*s* *|* *s = (si)i∈N,* *si∈Si*, ∀*i ∈ N* } là tổ hợp hành động của các người chơi (một cấu hình/chiến lược của trò chơi)
* *ui* là hàm payoff (thể hiện cơ chế thưởng phạt) của người chơi thứ *i*với một cấu hình S: *(s1, s2, …, sn)* 🡪 *ui(s1, s2, …, sn)*, có thể là lợi nhuận (cần tối đa hoá) hoặc chi phí (cần tối thiểu hoá).

## **Cân bằng Nash**

Cân bằng Nash xác định một chiến lược tối ưu cho các trò chơi khi chưa có điều kiện tối ưu nào được xác định trước đó. Định nghĩa cơ bản của cân bằng Nash là: Nếu tồn tại một tập hợp các chiến lược cho một trò chơi với đặc tính là không có một đối thủ nào có thể hưởng lợi bằng cách thay đổi chiến lược hiện tại của mình khi các đối thủ khác không thay đổi, tập hợp các chiến lược đó cùng kết quả tương ứng nhận được tạo nên cân bằng Nash. Nói cách khác, cân bằng Nash đạt được nếu như thay đổi một cách đơn phương của bất cứ ai trong số các đối thủ sẽ làm cho chính người đó thu lợi ích ít hơn mức có được với chiến lược hiện.

Một cấu hình *s\* = (s1, …, si, …, sn)* được gọi là cân bằng Nash khi:

*ui(si\*, s-i\*) ≥ ui(si, s-i\*),* ∀*si ∈ Si, ∀ i ∈ N*

## **Bài toán tối ưu hóa đa mục tiêu**

Bài toán tối ưu hoá đa mục tiêu là bài toán tối ưu hoá (cực đại hoặc cực tiểu) một tập các hàm mục tiêu *fi(x)* xác định trên tập Ω :

*F(x) = (f1(x), f2(x), …, fM(x)),*

đồng thời thoả mãn một số ràng buộc :

*ci(x) = 0,* ∀ *i ∈ E*

*cj(x) ≤ 0,* ∀ *j ∈ I*

Trong đó, *x* gọi là biến quyết định, là vector *L* chiều :

Thông thường, bài toán tối ưu hoá đa mục tiêu thường không có một lời giải mà tối ưu trên tất cả các mục tiêu mà sẽ tồn tại một tập các lời giải "**tốt**". Để đánh giá lời giải nào tốt hơn lời giải nào, người ta đưa ra khái niệm **dominate.** Lời giải *x1* được gọi là dominate (trội hơn) lời giải *x2* khi :

*fi(x1) ≤ fi(x2)* ∀ *i ∈ {1, 2, …, M}*

*j, fj(x1) < fj(x2)*

tức là *x1* không tồi hơn *x2* tại tất cả các mục tiêu và tốt hơn *x2* tại ít nhất một mục tiêu.

Một lời giải được gọi là Pareto optimal (tối ưu Pareto) nếu không tìm được lời giải nào dominate nó. Tập các lời giải như vậy gọi là một Pareto Front (tập tối ưu Pareto). Các giải thuật cho bài toán tối ưu hoá đa mục tiêu thường hướng đến việc tìm ra một tập tối ưu Pareto của bài toán.

## **Giải thuật di truyền**

Giải thuật di truyền là một loại thuật toán ứng dụng nguyên tắc di truyền gen trong sinh học vào để giải các bài toán. Cụ thể là trong sinh học, các cá thể trong một quần thể đều được đặc trưng bởi các gen. Và những gen này sau các quá trình chọn lọc, lai ghép, đột biến thì đều tiến hóa để thích nghi hơn với môi trường xung quanh.

Tương tự trong toán học, các lời giải của bài toán được coi như các gen (ban đầu có thể chưa phải tốt nhất), sau khi ta chọn lọc, lai ghép và đột biến các lời giải theo quy tắc toán học nhất định ta sẽ được lời giải mới “tốt hơn” so với lời giải cũ, cứ như thế cho đến khi ta tìm được lời giải đủ tốt thỏa mãn 1 tiêu chí đặt trước.

Giải thuật di truyền rất phù hợp cho những bài toán đa mục tiêu và đấu thầu nhiều vòng là một trong số đó khi mà ta phải thỏa mãn rất nhiều tiêu chí: lợi ích chủ thầu, lợi ích từng nhà thầu, tổng lợi ích các nhà thầu, tổng lợi ích nhà thầu và chủ thầu.

# **Mô hình bài toán đấu thầu nhiều vòng**

**Trạng thái cân bằng lợi ích:**

* Chủ đầu tư không phải chi trả quá cao cho dự án mà vẫn nhận được các sản phẩm giá trị, các nhà thầu với mức giá hợp lý đều thu được lợi nhuận nhất định từ dự án.
* Các nhà thầu với mức giá hợp lý đều thu được lợi nhuận nhất định từ dự án và được đánh giá cao, là đối tác làm ăn lâu dài với các chủ dự án.

**Đầu vào bài toán:**

* Các khoảng thời gian đấu thầu.
* Các nhà thầu đã qua vòng sơ loại.
* Giá cả các nhà thầu đưa ra cho dịch vụ của mình.
* Các dịch vụ cần cho dự án.

**Đầu ra bài toán:** Danh sách gói thầu với nhà thầu được lựa chọn, và thời điểm tổ chức gói thầu tương ứng

## **Mô hình toán học**

Gọi:

* N là số lượng gói thầu
* M là số nhà thầu tham gia
* T là thời điểm bắt đầu dự án
* ECi là chi phí ước tính (Estimated Cost) của gói thầu i.
* r là tỉ lệ lạm phát của đồng tiền
* mi là số lượng hàng hóa trong gói thầu i
* ci là nhà thầu đã trúng gói thầu i
* ti là thời điểm tố chức gói thầu i
* SPjci là giá bán (Sell price) sản phẩm j của nhà thầu ci ở gói thầu i
* BPjci là giá mua vào (Buy price) sản phẩm j của nhà thầu ci ở gói thầu i
* Mij là số lượng sản phẩm j cần mua ở gói thầu i
* Dijtci là chiết khấu (discount) mà nhà thầu ci đã áp dụng cho sản phẩm thứ j của gói thầu i tại thời điểm ti
* ai là tham số đánh giá quan hệ nhà thầu i với chủ đầu tư
* qi là tham số đánh giá chất lượng của nhà thầu i

**Lợi ích của chủ đầu tư:**

**Lợi ích của nhà thầu thứ k:**

Trong đó:

* i xác định đối với gói thầu i nều nhà thầu k mua gói thầu i, ci = k

**Cân bằng lợi ích giữa các nhà thầu:**

**Tổng lợi ích của tất cả các nhà thầu:**

**Hàm đảm bảo chất lượng dự án:**

Ngoài mục tiêu lợi ích giữa các nhà thầu đánh giá cực tiểu, các mục tiêu còn lại sẽ đánh giá cực đại. Dự án có M nhà thầu tham gia thì số mục tiêu của bài toán là:

## **MOEA Framework**

Để giải quyết bài toán này thì ta có thể tự xây dựng thuật toán từ đầu, tuy nhiên cũng có 1 cách tiếp cận khác nhanh hơn là sử dụng Framework (bao gồm các thuật toán đã được xây dựng sẵn) để chạy ra kết quả, vì mục đích của đề tài là so sánh các thuật toán với nhau chứ không phải là tối ưu thuật toán.

Một trong các Framework đó là **MOEA Framework**. MOEA Framework là một thư viện mã nguồn mở viết bằng Java dành cho việc phát triển và thí nghiệm những thuật toán đa mục tiêu. MOEA Framework hỗ trợ giải thuật di truyền, tiến hóa khác biệt, phương pháp tối ưu bầy đàn, lập trình di truyền, tiến hóa ngữ pháp và nhiều cái khác. Rất nhiều thuật toán được Framework cung cấp sẵn ví dụ như NSAG-II, NSGA-III, ε-MOEA, GDE3, PAES, PESA2, SPEA2, IBEA, SMS-EMOA, SMPSO, OMOPSO, CMA-ES, and MOEA/D. Người dùng chỉ cần cung cấp đầu vào bài toán, chọn thuật toán, sau đó Framework sẽ xử lý và đưa ra kết quả. Ngoài ra, MOEA Framework còn cung cấp những công cụ cần thiết cho việc thiết kế, phát triển, triển khai và kiểm thử thống kê tối ưu thuật toán.

# **CÁC THUẬT TOÁN SỬ DỤNG**

Trong đề tài này, dưới sự hướng dẫn của anh NCS, nhóm chúng em sử dụng **6 thuật toán** là: **NSGA-III, ε-MOEA, GDE3, PESA2, IBEA, SMPSO**. Lý do là vì 6 thuật toán này có cách tiếp cận khác nhau với bài toán, đảm bảo độ đa dạng để có thể so sánh được.

## **NSGA-III**

Thuật toán **NSGA-III** được phát triển trên cở sở của **NSGA** và **NSGA-II**. **NSGA** là thuật toán di truyền sử dụng khái niệm dominate trong quá trình đánh giá hàm thích nghi của các cá thể trong quần thể.

Tổng quan các bước của **NSGA:**

1. Sinh tập cá thể ngẫu nhiên.
2. Lai ghép và đột biến.
3. Xếp hạng rồi chọn lọc.
4. Nếu thỏa mãn thì dừng lại, chưa thì lại bắt đầu quay lại bước 2.

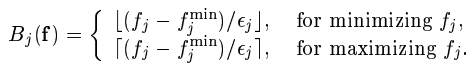
**NSGA-II** thì cải thiện hơn NSGA ở bước phân lớp Pareto Front.

**NSGA-III** thì các bước từ đầu đến lai ghép và đột biến y hệt như NSGA-II tuy nhiên có khác ở bước chọn lọc:

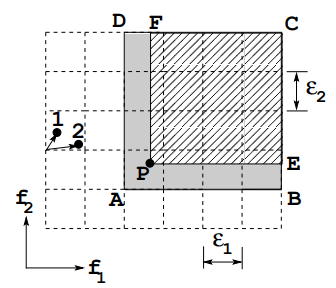
* Đầu tiên xác định số **reference point**.
* Sau khi lai ghép đột biến để sinh ra con thì tính min với từng **Objective Function (OF)**, sau đót trừ tất cả f khác cho min này.
* Mỗi cá thể sẽ cho ra 1 kết quả tương ứng với từng hàm mục tiêu -> chuyển hóa thành 1 điểm trên mỗi trục tọa độ -> gộp các điểm này được 1 điểm trên cả hệ trục đại diện cho lời giải.
* Lập ra **hyper plan** với các reference point dựa vào các điểm vừa được kéo vào trục tọa độ. Các refernce point được chia đều ra trên mặt phẳng reference.
* Với tất cả điểm còn lại, xác định xem gần trục tọa độ nào nhất, sau khi đã xác định được, vẽ 1 đường vuông góc từ điểm này đến trục tọa độ đấy.
* Xác định điểm reference point nào gần với đường vuông góc đó nhất thì sẽ gắn điểm đó với điểm đang xét.
* Tiếp tục làm như vậy, nếu có nhiều hơn 1 điểm gắn với cùng 1 **reference point** thì sẽ lấy điểm nào gần hơn.
* Cuối cùng ta sẽ lấy ra được các điểm cho quá trình di truyền tiếp theo.

## **ε-MOEA**

**ε-dominate** là một khái niệm tổng quát của **dominate**



Ở đây, có thể coi như chia khoảng giá trị của các hàm mục tiêu *fj* thành các đoạn liên tiếp có độ dài *εj*, kết quả ta thu được một hyper-boxes. Trong trường hợp *m=2*, ta thu được một lưới trên mặt phẳng toạ độ (hai chiều ứng với hai hàm mục tiêu) như hình dưới đây:

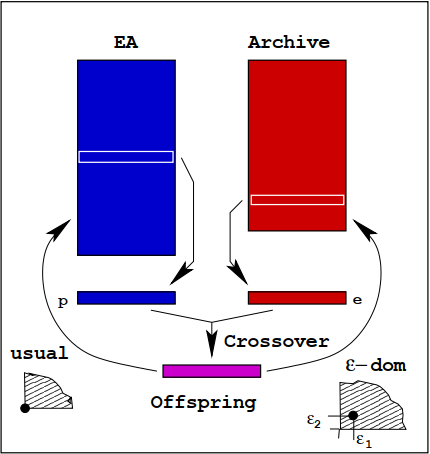


Các điểm A, B, C, D, E, F, P biểu diễn biến quyết định có giá trị hàm mục tiêu là toạ độ tương ứng. Trong trường hợp tối ưu hoá min cả *f1*và *f2*, theo định nghĩa **dominate** thông thường, dễ thấy, P dominate các điểm trong hình chữ nhật PECF. Tuy nhiên, trong khái niệm **ε-dominate**,điểm A sẽ được dùng để đánh giá ε-dominate cho điểm P, tức là với trường hợp này điểm P ε-dominate các điểm trong hình chữ nhật ABCD.

Với định nghĩa như trên, ta thấy, hai điểm có thể ε-dominate lẫn nhau khi cùng nằm trong một ô vuông của lưới (có cùng vecto đặc trưng). Đồng thời, khi các giá trị *εj* càng tiến đến 0 thì ε-dominate càng gần với dominate thông thường.

Thuật toán **ε-MOEA** là một giải thuật di truyền, tức mỗi lời giải được coi như một cá thể, tập lời giải là quần thể, hai lời giải (cha, mẹ) có thể kết hợp (lai) với nhau để tạo thành lời giải mới (cá thể con). Thuật toán ε-MOEA gồm các bước:

* Bước 1: Khởi tạo ngẫu nhiên quần thể ban đầu P(0).
* Bước 2: Chọn các các thể tốt nhất trong P(0) đưa vào quần thể E(0) (quần thể lưu trữ/bảo tồn)
* Bước 3: Chọn ngẫu nhiên một cá thể trong mỗi quần thể P(0), E(0), lai ghép với nhau tạo thành cá thể con *c* (trong trường hợp tổng quát có thể tạo ra nhiều cá thể con *ci*)
* Bước 4 (chọn lọc): với mỗi cá thể con được tạo ra, quyết định giữ lại hay không dựa vào khái niệm **ε-dominate**.
* Bước 5: Dừng lại nếu thoả mãn điều kiện dừng (số vòng lặp tối đa, quần thể đủ tốt, không thay đổi, …) hoặc quay lại bước 3 nếu không thoả mãn.



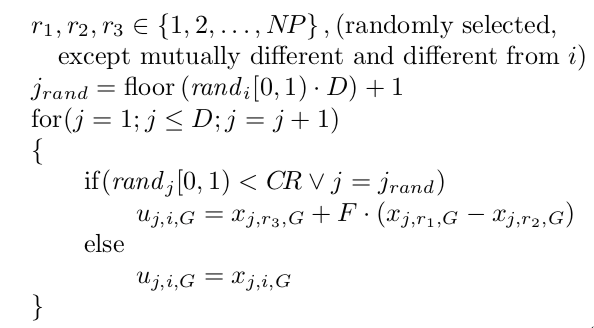
Việc quyết định giữ lại cá thể con *c* trong P (hoặc E) hay không dựa trên khái niệm **ε-dominate** đã trình bày ở trên theo quy tắc:

* Nếu tất cả cá thể trong quần thể đều ε-dominate *c* thì bỏ qua *c.*
* Nếu có ít nhất 1 cá thể bị dominate bởi *c* thì giữ lại *c* và loại bỏ 1 cá thể ngẫu nhiên trong số các cá thể đó.
* Nếu không xảy ra 1 trong 2 trường hợp trên:
  + Nếu 1 cá thể thuộc quần thể có cùng vecto đặc trưng với *c* thì chọn giữ lại cá thể nào có giá trị hàm mục tiêu gần giống vecto đặc trưng hơn (theo khoảng cách Euclide)
  + Ngược lại, giữ lại *c* và loại bỏ một cá thể ngẫu nhiên trong quần thể.

## **GDE3**

Thuật toán **GDE3** cũng dựa trên giải thuật di truyền tuy nhiên lại không có lai ghép mà chỉ có đột biến và chọn lọc. Tư tưởng của **GDE** thuật toán là đột biến cá thể x thành cá thể x’ theo 1 quy tắc nhất định. Sau đó so sánh x với x’ xem cái nào tốt hơn thì lấy. **GDE2** và **GDE3** sau đó phát triển dựa trên nền tảng của **GDE**.

Công thức đột biến của **GDE** là:



Trong đó **F** và **CR** là 2 biến điều khiển (do ta chọn trước). Đoạn **j = jrand** là để đảm bảo sau 1 thế hệ thì có ít nhất 1 cá thể được đột biến.

Nhận thấy **GDE** quá nhạy cảm trong việc chọn biến điều khiển (**F** và **CR**) nên **GDE2** cải tiến bằng việc ra quyết định dựa trên **Crowdedness** khi mà x và x’ không **dominate** được nhau, nâng cao độ đa dạng lời giải nhưng lại làm chậm độ hội tụ của toàn quần thể vì nó ưu tiên các cá thể biên.

**GDE3** mở rộng phương thức đột biến với **M Objectives** và **K Constraints**.

Quy tắc selection của **GDE3**:

* Cả 2 không khả thi: cái mới được chọn nếu violate constraints ít hơn cái cũ.
* 1 cái khả thi và 1 không khả thi thì cái khả thi được chọn.
* Cả 2 khả thi: cái mới được chọn nếu weakly dominate cái cũ ở hàm mục tiêu. Cái cũ được chọn nếu dominate cái cũ. Chọn cả hai nếu không cái nào dominate cái nào.
* Sau 1 thế hệ, số lượng cá thể có thể tăng lên. Nếu vào trường hợp này thì sẽ giảm size quần thể giống **NSGA-II** (dựa trên **non-dominance** & **crowdedness**).

## **PESA2**

Thuật toán PESA2 dựa trên nguyên lý chuẩn của một thuật toán tiến hoá, duy trì 2 quần thể: một quần thể nội bộ (internal population) kích thước cố định và một quần thể ngoài (external population) (ví dụ như một tập lưu trữ) không cố định số cá thể nhưng giới hạn kích thước quần thể.

Quần thể nội bộ lưu trữ các solution được tạo ra từ tập lưu trữ bằng các phép biến đổi khác nhau, và tập lưu trữ chỉ chứa các solution vượt trội được phát hiện trong suốt quá trình tìm kiếm.

Một lưới quyết định được tạo ra trên không gian mục tiêu để duy trì đa dạng quần thể. Lưới quyết định được chia thành nhiếu khối nhỏ (hyperbox). Số lượng giải pháp trong một hyperbox được gọi là mật độ của hyperbox và được sử dụng để phân biệt các giải pháp trong 2 quá trình quan trọng của một giải thuật tiến hoá đa mục tiêu (EMO): chọn lọc dựa vào giao phối và chọn lọc từ môi trường.

Không giống như những thuật toán EMO khác, việc giao phối trong PESA-II được thực hiện theo khu vực hơn là theo cá nhân. Một hyperbox được chọn đầu tiên và sau đó cá thể được chọn để tiến hoá sẽ được chọn ngẫu nhiên từ hyperbox đã chọn, do đó các hyperbox đông hơn sẽ không đóng góp nhiều cá thể hơn so với các hyperbox thưa hơn.

Trong quá trình chọn lọc từ môi trường, các cá thể ứng cử viên trong quần thể nội bộ lần lượt được thêm vào tập lưu trữ (archive set) nếu nó không bị trội bởi bất kỳ cá thể nào trong quần thể, và không bị trội bởi bất cứ cá thể nào trong archive set. Khi một cá thể được thêm vào archive set, sẽ có sự thay đổi của archive set và lưới môi trường. Đầu tiên, các cá thể trong tập lưu trữ bị trội bởi cá thể ứng cử viên sẽ bị loại bỏ để chắc chắn rằng chỉ có các cá thể không bị trội mới có trong archive set. Sau đó, lưới được kiểm tra xem biên của nó có bị thay đổi bởi việc thêm và bớt các cá thể. Cuối cùng, nếu việc thêm phần tử làm tập lưu trữ bị đầy, một phần từ ngẫu nhiên trong hyperbox đông nhất sẽ bị loại bỏ.

## **IBEA**

Ý tưởng chính của thuật toán IBEA (IBEAHD) là sử dụng một chỉ số hypervolume nhị phân trong quá trình chọn lọc, khi xác định solution nào sẽ được giữ lại trong thế hệ tiếp theo. Chỉ số nhị phân hypervolumn này gán một giá trị thực cho 2 tập giải pháp liên quan đến cùng một điểm trong không gian mục tiêu.

được định nghĩa là miền không gian bị trội bởi B nhưng không bị trội bởi A

Với biểu thị hypervolume được hình thành bởi bộ giải pháp A, tương tự, là hypervolume của sự kết hợp giữa 2 bộ giải pháp A và B. âm nếu tất cả các solution trong B đều bị trội bởi solution trong A.

Mã giả của thuật toán IBEA được trình bày ở dưới (IBEA algorithm).

Trước hết, IBEA tạo ngẫu nhiên quần thể khởi tạo ở bước 1, sau đó các bước sau được lặp cho đến khi điều kiện dừng được thoả mãn. Giá trị của mục tiêu được co dãn và giá trị hàm fitness được gán cho mỗi cá thể ở bước 2 và bước 3. Bước 4 thực hiện việc chọn lọc từ môi trường, duyệt qua các cá thể kém nhất trong P dựa vào chỉ số I cho đến khi còn lại cá thể. Giá trị chỉ số của các cá thể còn lại trong P sẽ được cập nhật. Bước 6 và bước 7 thực hiện việc tạo các cá thể mới và thêm chúng vào quần thể P.

**IBEA algorithm**

**Input**

: kích thước quần thể

N: tổng số lần đánh giá các giải pháp

: nhân tố tăng giảm giá trị mục tiêu

: giá trị chỉ số (sai khác hypervolume)

**Output:** A - tập xấp xỉ Pareto

**Bước 1. Khởi tạo**

Khởi tạo ngẫu nhiên quần thể P và đặt giá trị đếm m.

**Bước 2. Mở rộng giá trị mục tiêu**

1. Tìm cận dưới và cận trên của mỗi mục tiêu *i.*

và

2. Tăng cận trên:

3. Chuyển đổi mỗi mục tiêu vào miền [0, 1]

**Bước 3. Gán giá trị hàm fitness**

1. Tính chỉ số sử dụng các giá trị thay vì và xác định chỉ số có trị tuyệt đối lớn nhất

2. Tính giá trị hàm fitness cho tất cả giá trị

**Bước 4. Lựa chọn từ môi trường**

Duyệt qua 3 bước sau cho đến khi kích thước quần thể P

1. Chọn một cá thể có giá trị hàm fitness nhỏ nhất:

2. Xoá khỏi P

3. Cập nhật giá trị hàm fitness cho toàn bộ cá thể còn lại

**Bước 5. Điều kiện dừng**

Nếu trả về tập A là các giải pháp không bị trội trong P

Nếu không, tiếp tục.

**Bước 6. Chọn lọc từ giao phối**

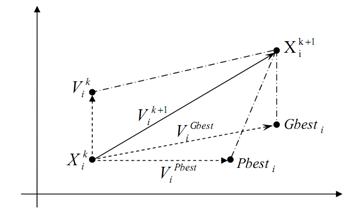
Áp dụng các toán tử chọn lọc để chọn cặp bố mẹ từ P.

**Bước 7. Biến đổi**

Thực hiện lai ghép trên cặp bố mẹ để tạo ra 2 cá thể con, và dùng các toán tử đột biến trên các cá thể con này. Sau đó thêm vào P. Tăng m lên 1 đơn vị. Quay lại bước 2.

## **SMPSO**

PSO được khởi tạo bằng một nhóm cá thể(nghiệm) ngẫu nhiên và sau đó tìm nghiệm tối ưu bằng cách cập nhật các thế hệ. Trong mỗi thế hệ, mỗi cá thể được cập nhật theo hai giá trị tốt nhất. Gía trị thứ nhất là nghiệm tốt nhất đạt được cho tới thời điểm hiện tại, gọi là Pbest. Một nghiệm tối ưu khác mà cá thể này bám theo là nghiệm tối ưu toàn cục Gbest, đó là nghiệm tốt nhất mà cá thể lân cận cá thể này đạt được cho tới thời điểm hiện tại. Nói cách khác, mỗi cá thể trong quần thể cập nhật vị trí của nó theo vị trí tốt nhất của nó và của cá thể trong quần thể tính tới thời điểm hiện tại.



Trong đó :

 : Vị trí cá thể thứ i tại thế hệ thứ k

 : Vận tốc cá thể i tại thế hệ thứ k

Pbesti : Vị trí tốt nhất của cá thể thứ i

Gbesti : Vị trí tốt nhất của cá thể trong quần thể

**Vận tốc và vị trí của mỗi cá thể được tính như sau**:

vik+1 = w.vik + c1.r1().(pbesti – xik) + c2.r2().(gbest – xik)

xik+1 = xit + vik+1

Trong đó

w : trọng số quán tính

c1, c2 : các hệ số gia tốc

r1, r2 :  số ngẫu nhiên giữa 0 và 1

# **CÀI ĐẶT VÀ ĐÁNH GIÁ THỬ NGHIỆM**

## Định nghĩa bài toán

Một bài toán được định nghĩa bằng cách *implement* lớp trừu tượng AbstractProblem. MOEA Framework chấp nhận các biến quyết định thuộc một số kiểu dữ liệu: số nguyên, số thực, chuỗi bit, … Vì vậy, để đơn giản, ta quy một lời giải của bài toán đấu thầu nhiều vòng về một vector số nguyên (có số chiều bằng số biến quyết định). Do có *N* gói thầu, mỗi gói thầu cần xác định 2 yếu tố là thời gian tổ chức và nhà thầu trúng gói thầu đó, mỗi lời giải sẽ được quy về một vector *2\*N* chiều, với thời gian tổ chức xác định bởi số ngày tính từ thời điểm bắt đầu dự án, nhà thầu được xác định bởi mã số (số thứ tự) nhà thầu.

Cần tối ưu 4 hàm mục tiêu:

* Lợi nhuận của chủ đầu tư
* Tổng lợi nhuận các nhà thầu
* Cân bằng lợi ích giữa các nhà thầu
* Chất lượng dự án

Cần thoả mãn các ràng buộc:

* Lợi nhuận chủ đầu tư không âm
* Lợi nhuận mỗi nhà thầu không âm

## Các tiêu chí đánh giá

Đánh giá lời giải qua giá trị **payoff**:

* Với mỗi mục tiêu, tìm giá trị *fmin, fmax* như đối với bài toán tối ưu hoá đơn mục tiêu.
* Với mỗi lời giải, “**độ** **tốt**” tại từng mục tiêu *f* được tính theo công thức:
* với tối ưu hoá min
* với tối ưu hoá max
* Pay-off của lời giải được tính bằng tổng “độ tốt” tại từng mục tiêu nhân trọng số tương ứng. Như vậy, giá trị payoff nằm trong khoảng [0; 1] và càng gần 0 thì lời giải càng tốt.

Lời giải tốt nhất mà thuật toán tìm được (sau một số lần chạy) sẽ dùng để so sánh với các thuật toán khác.

## Kết quả thử nghiệm

Chương trình được thử nghiệm trên 2 bộ dữ liệu đầu vào (gồm 1 bộ dữ liệu được cung cấp và 1 bộ dữ liệu tự tạo). Với một bộ dữ liệu, mỗi thuật toán được chạy 20 lần để tìm ra lời giải tốt nhất trong tất cả các lần, thời gian chạy của thuật toán được tính bằng thời gian trung bình của 20 lần chạy. Quần thể được khởi tạo ngẫu nhiên gồm 100 cá thể, số cá thể tối đa được tạo ra trong một lần chạy là 10000 (MOEA Framework yêu cầu đặt một giới hạn cho số cá thể tối đa được tạo ra).

Môi trường thử nghiệm :

* Windows 10
* Intel Core i5 6200U (Skylake), 2.7GHz, RAM 8GB.
* Java 11

Bảng so sánh kết quả:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thuật toán | NSGA-III  *O(mN2)* | **ε**-MOEA  *O(mN2)* | GDE3 | PESA2  *O(Np)* | IBEA  *O(pN2)* | SMPSO  *O(pnlog(n))* |
| Bộ dữ liệu 1 | | | | | | |
| Thời gian | 5.255 | 7.285 | 6.169 | 4.881 | 6.048 | 4.758 |
| Payoff | 0.3159 | 0.3036 | 0.3157 | 0.3040 | 0.3035 | 0.3097 |
| Bộ dữ liệu 2 | | | | | | |
| Thời gian | 6.604 | 8.951 | 7.998 | 6.119 | 7.099 | 6.424 |
| Payoff | 0.2280 | 0.2349 | 0.2144 | 0.2253 | 0.2077 | 0.2077 |

## Nhận xét và đánh giá

##### **TÀI LIỆU THAM KHẢO**

[1] ZITZLER, Eckart; KÜNZLI, Simon. Indicator-based selection in multiobjective search. In: *International Conference on Parallel Problem Solving from Nature*. Springer, Berlin, Heidelberg, 2004. p. 832-842.

[2] NEBRO, Antonio J., et al. Smpso: A new pso-based metaheuristic for multi-objective optimization. In: *Computational intelligence in multi-criteria decision-making, 2009. mcdm'09. ieee symposium on*. IEEE, 2009. p. 66-73.

[3] DEB, Kalyanmoy, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 2002, 6.2: 182-197.

[4] CORNE, David W.; KNOWLES, Joshua D.; OATES, Martin J. The Pareto envelope-based selection algorithm for multiobjective optimization. In: *International conference on parallel problem solving from nature*. Springer, Berlin, Heidelberg, 2000. p. 839-848.

[5] COELLO, CA Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine*, 2006, 1.1: 28-36.

[6] DEB, Kalyanmoy; MOHAN, Manikanth; MISHRA, Shikhar. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. *Evolutionary computation*, 2005, 13.4: 501-525.

[7] HADKA, David. Moea framework-a free and open source java framework for multiobjective optimization. version 2.11. *URL http://www. moeaframework. org* [access on November 1st], 2015.