HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Dương Kiên

NGHIÊN CỨU THUẬT TOÁN TIẾN HÓA ĐA NHÂN TỐ GIẢI QUYẾT BÀI TOÁN TỐI ƯU

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

Hà Nội - 2019

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: **TS. Trần Quý Nam**

Phản biện 1: ……

Phản biện 2: ……

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: ..... giờ ..... ngày ..... tháng ..... năm

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông

LỜI NÓI ĐẦU

Trong những năm vừa qua, các thuật toán tiến hóa được áp dụng để giải quyết nhiều bài toán tối ưu trong khoa học máy tính và trong thực tế. Tuy nhiên, việc thiết kế các thuật toán tiến hóa mới chỉ tập trung vào việc giải quyết có hiệu quả một bài toán tối ưu tại một thời điểm, chưa có thuật toán tiến hóa giải quyết đồng thời các bài toán tối ưu hóa chỉ sử dụng duy nhất một quần thể. Do vậy, luận văn sẽ tìm hiểu một mô hình tiến hóa mới trong tính toán tiến hóa: mô hình tiến hóa đa nhân tố (Multifactorial Optimization) cho phép giải đồng thời nhiều bài toán tối ưu mà chỉ dựa trên một quần thể tiến hóa duy nhất.

Thuật toán tiến hóa (Evolutionary Algorithms - EAs) dựa theo học thuyết Darwin nói chung được hình thành trên quan niệm cho rằng, quá trình tiến hóa là quá trình hoàn hảo nhất vì tự nó đã mang tính tối ưu [1]. Tính tối ưu được thể hiện ở chỗ, cá thể sau được sinh ra bao giờ cũng tốt hơn, hoàn hảo hơn cá thể cha-mẹ, chúng có khả năng thích nghi với sự thay đổi của môi trường cao hơn cá thể cha-mẹ. Thuật toán tiến hóa được áp dụng trong các bài toán tối ưu. Bài toán tối ưu là bài toán tìm giá trị cực đại hoặc cực tiểu của một hàm hoặc một quá trình nào đó. Cơ chế này được sử dụng trong nhiều lĩnh vực như vật lý, hóa học, kinh tế,... Trong thuật toán tiến hóa, một nhóm các cá thể (giải pháp của bài toán) sẽ được khởi tạo ngẫu nhiên. Trong mỗi thế hệ, những cá thể tốt, thích nghi với môi trường (bài toán) sẽ được giữ lại. Quá trình tiếp tục cho đến khi gặp điều kiện dừng của bài toán. Có nhiều thuật toán tiến hóa khác nhau như: thuật toán di truyền (Genetic Algorithm – GA), thuật toán tối ưu hóa bầy đàn (Particle Swarm Optimization – PSO), thuật toán đàn kiến (Ant Colony Optimization - ACO),....Trong đó, thuật toán di truyền được xây dựng dựa trên quy luật tiến hóa sinh học hay phát triển tự nhiên của một quần thể sống. Các cá thể trải qua một quá trình phát triển và sinh sản để tạo ra những cá thể mới cho thế hệ kế tiếp. Trong quá trình tăng trưởng và phát triển những cá thể xấu tức là những cá thể không thích nghi được với môi trường sẽ bị đào thải, ngược lại, những cá thể tốt sẽ được giữ lại (đây chính là quá trình chọn lọc) và được lai ghép (quá trình lai ghép) để tạo ra những cá thể mới cho thế hệ sau. Những cá thể mới được sinh ra mang những tính trạng của cá thể cha-mẹ (còn gọi là hiện tượng di truyền). Thuật toán tối ưu bầy đàn được xây dựng dựa vào quá trình mô phỏng sinh học của đàn chim. Để hiểu rõ về thuật toán, hãy xem một ví dụ về quá trình tìm kiếm thức ăn của một đàn chim. Tại thời điểm tìm kiếm cả đàn bay theo một hướng nào đó, có thể

là ngẫu nhiên. Tuy nhiên, sau một thời gian tìm kiếm một số cá thể trong đàn bắt đầu tìm ra được nơi có chứa thức ăn. Tùy vào số lượng thức ăn vừa tìm được mà các cá thể gửi tín hiệu đến các cá thể đang tìm kiếm ở vùng lân cận. Tín hiệu này được lan truyền trên toàn quần thể. Dựa vào thông tin nhận được, mỗi cá thể sẽ điều chỉnh hướng bay và vận tốc bay theo hướng về nơi có nhiều thức ăn nhất. Cơ chế truyền tin như vậy thường được xem là một kiểu hình của trí tuệ bầy đàn. Cơ chế này giúp đàn chim tìm ra nơi có nhiều thức ăn nhất trên không gian tìm kiếm [2]. Các thuật toán tiến hóa trên chỉ dừng lại ở việc giải quyết một bài toán tối ưu tại một thời điểm. Tuy nhiên, hầu hết các ứng dụng trong thực tế đều yêu cầu phải giải quyết nhiều bài toán tối ưu cùng lúc, ví dụ như ứng dụng trong tính toán đám mây. Do đó, luận văn sẽ tìm hiểu một mô hình tiến hóa mới: mô hình tiến hóa đa nhân tố (Multifactorial Optimization - MFO). Mô hình tiến hóa đa nhân tố là mô hình tiến hóa tổng hợp để giải quyết đồng thời nhiều bài toán tối ưu. Mỗi bài toán tối ưu được coi như một nhân tố ảnh hưởng đến quá trình tiến hóa. Ưu điểm của phương pháp này là chúng ta có thể chuyển vật liệu di truyền từ các bài toán tối ưu đơn giản đến các bài toán tối ưu phức tạp. Điều này có thể đẩy nhanh quá trình tối ưu hóa, giảm thời gian thực hiện.

Cấu trúc luận văn được tổ chức như sau

Chương 1: Luận văn sẽ trình bày tổng quan về bài toán tối ưu hóa và các phương pháp để giải quyết một bài toán tối ưu hóa.

Chương 2: Luận văn sẽ trình bày về mô hình tiến hóa đa nhân tố và giải thuật tiến hóa đa nhân tố để giải quyết bài toán tối ưu hóa.

Chương 3: Áp dụng thuật toán tiến háo đa nhân tố để giải các bài toán tối ưu đơn mục tiêu

Chương 1 TỔNG QUAN

1.1 Bài toán tối ưu

Giải thuật di truyền (Di truyền - Genetic Algorithm (GA)) Tối ưu hóa là cơ chế tìm giá trị cực tiểu hoặc cực đại của một hàm hoặc một quá trình nào đó. Cơ chế này được sử dụng trong nhiều lĩnh vực như vật lý, hóa học, kinh tế...để đạt được mục đích là tối đa hóa hiệu quả, sản xuất hoặc các thước đo khác. Tối ưu hóa liên quan đến hai khái niệm cực tiểu và cực đại của một hàm f nào đó. Đây là hai bài toán đối lập nhau, trong đó, tìm cực tiểu của hàm f tương đương với tìm cực đại của hàm $-f$.

Bài toán tối ưu hóa được chia làm hai loại chính đó là Tối ưu rời rạc hay còn gọi là tối ưu tổ hợp (TƯTH) và Tối ưu liên tục. Trong chương này tác giả sẽ chỉ tập trung vào tối ưu hóa tổ hợp

- Dựa vào số lượng mục tiêu: đơn mục tiêu, đa mục tiêu
- Dựa vào ràng buộc: có ràng buộc, không có ràng buộc
- Dựa vào miền giá trị của biến: tối ưu liên tục hay còn gọi là tối ưu tổ hợp (TƯTH), tối ưu rời rạc

Trong chương này tác giả sẽ chỉ tập trung vào tối ưu hóa tổ hợp

1.1.1 Tối ưu hóa tổ hợp

Một cách tổng quát, mỗi bài toán TƯTH có thể phát biểu như sau: Cho một bộ ba (S,f,Ω), trong đó S là tập hữu hạn trạng thái (lời giải tiềm năng hay phương án), f là hàm mục tiêu xác định trên S, còn Ω là tập các ràng buộc. Mỗi phương án $s \in S$ thỏa mãn các ràng buộc ω gọi là phương án (hay lời giải) chấp nhận được. Mục đích của ta là tìm phương án chấp nhận được $s*$ tối ưu hóa toàn cục hàm mục tiêu f. Chẳng hạn với bài toán cực tiểu thì $f(s*) \leq f(s)$ với mọi phương án chấp nhận được s. Hay có thể tóm gọn lại: bài toán được gọi là tối ưu tổ hợp khi các biến quyết định nhận giá trị trong một tập rời rạc, được giới hạn bởi một số ràng buộc. Chúng ta có một số bài toán tiểu biểu cho lớp bài toán này là [4]:

- Bài toán người du lịch (Traveling Salesman Problem)
- Cây khung nhỏ nhất (Miminum Spanning Tree)
- Bài toán phân công (Assignment Problem)
- Bài toán cái túi (Knapsack Problem)

Và để minh họa cho phần lý thuyết tổng quát, tác giả sẽ tập trung vào hai bài toán là bài toán phân công và bài toán cây khung nhỏ nhất.

Bài toán phân công

Bài toán cây khung ngắn nhất

Cho $G = (X,E)$ là một đồ thị liên thông và $T = (X,F)$ là một đồ thị bộ phận của G. Nếu T là cây thì T được gọi là một cây khung của G Cây khung còn có thể được gọi bằng các tên khác như cây bao trùm, cây phủ hoặc là cây tối đại. Sử dụng thuật toán Prim ta có thể giải bài toán như sau:

Đầu vào:

Đồ thị liên thông $G = (X,E)$,X gồm N đỉnh

Đầu ra:

Cây khung $T = (V,U)$ của G

Ràng buộc:

1. Chọn tùy ý 1 đỉnh $v \in X$ và khởi tạo $V := \{v\}$; $U := \varphi$;
2. Chọn cạnh e có trọng lượng nhỏ nhất trong các cạnh (w,v) mà $w \in X/V$ và $v \in V$.
3. $V := V \cup \{w\}$; $U := U \cup \{e\}$
4. Nếu U đủ $N - 1$ cạnh thì dừng, ngược lại lặp từ thao tác số 2.

Bài toán người du lịch

Bài toán cái túi

1.1.2 Giải bải toán tối ưu

Để giải bài toán tối ưu, việc cần làm chính là tìm kiếm phương án làm cho hàm mục tiêu đạt giá trị nhỏ nhất (hoặc lớn nhất). Với các bài toán khó cỡ nhỏ, người ta có thể tìm lời giải tối ưu nhờ tìm kiếm vét cạn. Tuy nhiên với các bài toán cỡ lớn thì đến nay chưa thể có thuật toán tìm lời giải đúng với thời gian đa thức nên chỉ có thể tìm lời giải gần đúng hay đủ tốt

Theo cách tiếp cận truyền thống hay là tiếp cận cứng, các thuật toán gần đúng phải được chứng minh tính hội tụ hoặc ước lượng được tỷ lệ tối ưu. Với việc đòi hỏi khắt khe về

toán học như vậy làm hạn chế số lượng các thuật toán công bố, không đáp ứng được nhu cầu ngày càng phong phú và đa dạng trong nghiên cứu và ứng dụng.

Có những phương pháp đưa ra được lời giải chính xác nhưng chi phí tính toán lớn, nhưng cũng có những phương pháp lại chỉ đưa ra được lời giải gần đúng, tương đối tuythế thời gian để tính toán bài toán lại nhỏ hơn rất nhiều.

Chúng ta có thể phân thành hai phương pháp chính đó là : phương pháp giải chính xác và phương pháp giải gần đúng

Giải thuật chính xác

- Vét cạn

Vét cạn, duyệt, quay lui... là một số tên gọi tuy không đồng nghĩa nhưng cùng chỉ một phương pháp rất đơn giản trong tin học và toán học: tìm nghiệm của một bài toán bằng cách xem xét tất cả các phương án có thể. Đối với con người thì phương pháp này thường là không khả thi vì số phương án cần kiểm tra quá lớn. Tuy nhiên đối với máy tính, nhờ tốc độ xử lý nhanh, máy tính có thể giải rất nhiều bài toán bằng phương pháp này. Nhưng về cơ bản phương pháp này tốn rất nhiều thời gian và khó thực hiện, ngay cả trên những máy tính hiện đại nhất vì sự xuất hiện của bùng nổ tổ hợp

- Thuật toán nhánh cận

Như đã đề cập ở trên, nhánh – cận là một thuật toán cải tiến dựa trên vét cạn. Mục đích là để xây dựng những phương án khả thi, trong quá trình liệt kê, những phương án này sẽ dựa vào thông tin tìm được để loại bỏ sớm những phương án chắc chắn không phải tối ưu. Nhờ vậy không gian tìm kiếm cũng được thu hẹp lại mà vẫn đảm bảo kết quả chính xác và thời gian giải cũng giảm xuống. Thuật toán này lần đầu được giới thiệu vào năm 1960 bởi Land A.H và Doig A.G trong [8] để giải bài toán quy hoạch nguyên. Cho tới nay, phương pháp này vẫn được áp dụng rộng rãi để giải các bài toán tối ưu khó giải quyết. Trong thuật toán này, chúng ta sẽ từng bước xây dựng các phương án cho bài toán với tất cả các khả năng có thể xảy ra, trong đó mỗi nhánh của phương án đang được xây dựng bởi thuật toán sẽ chấm dứt khi biết được tổng trọng số của phương án này vượt quá giá trị cận dưới (giá trị hàm mục tiêu của phương án đã được xác định trước đó tính đến thời điểm hiện tại là tốt nhất).

Tư tưởng chính của nhánh cận như sau: Giả sử ta đã xây dung được k thành phần từ x_1 đến x_k, giờ ta chuẩn bị mở rộng thành phần thứ x_k+1 từ x_k. Nhưng khi đánh giá ta lại thấy tất cả các nghiệm mở rộng từ x_k không có nghiệm nào có giá trị tốt hơn giá trị tối ưu ta đã biết tại thời điểm đó, vậy thì ta không cần mở rộng nữa, như vậy ta đã cắt bỏ đi một nhánh, giảm được số nghiệm phải tìm rất nhiều.

Điều khó nhất ở đây là phải đánh giá được các nghiệm mở rộng, nếu đánh giá được tốt, thuật toánh nhánh cận sẽ chạy nhanh hơn rất nhiều so với vét cạn.

- Quy hoạch động

Phương pháp quy hoạch động dùng để giải bài toán tối ưu có bản chất đệ quy, tức là việc tìm phương án tối ưu cho bài toán đó có thể đưa về tìm phương án tối ưu của một số hữu hạn các bài toán con.

Đối với một số bài toán đệ quy, nguyên lý chia để trị (divide and conquer) thường đóng vai trò chủ đạo trong việc thiết kế thuật toán. Để giải quyết một bài toán lớn, ta chia nó thành nhiều bài toán con cùng dạng với nó để có thể giải quyết độc lập.

Phép phân giải đệ quy bắt đầu từ bài toán lớn phân ra thành nhiều bài toán con và đi giải từng bài toán con đó. Việc giải từng bài toán con lại đưa về phép phân ra nhiều bài toán nhỏ hơn và lại đi giải các bài toán nhỏ hơn đó bất kể nó đã được giải hay chưa – hay còn gọi là phương pháp theo cấu trúc từ trên xuống (top-down)

Quy hoạch động bắt đầu từ việc giải tất cả các bài toán nhỏ nhất (bài toán cơ sở) để từ đó từng bước giải quyết những bài toán lớn hơn, cho tới khi giải được bài toán lớn nhất (bài toán ban đầu) – hay còn gọi là phương pháp đi từ dưới lên (bottom-up)

Giải thuật xấp xỉ (phương pháp giải gần đúng)

Giải thuật xấp xỉ được đưa ra nhằm khắc phục các hạn của giải thuật chính xác khi có sự bùng nổ tổ hợp, nghĩa là không gian tìm kiếm quá lớn mà nguyên nhân là do kích thước dữ liệu đầu vào tăng lên. Mục đích của loại thuật toán này không phải để tìm ra lời giải tối ưu chắc chắn mà để tìm ra lời giải gần tối ưu nhất nhưng trong thời gian chấp nhận được.

Sau đây tác giả sẽ đi sâu vào hai cách chính đó là Heuristic và Meta-heuristic

Heuristic

- Phương pháp tham lam (*Gready*)

Thuật toán tham lam là một thuật toán giải quyết một bài toán theo kiểu Heuristic để tìm kiếm lựa chọn tối ưu ở mỗi bước với hy vọng tìm đuọc tối ưu toàn bộ. Hay nói cách khác, sự lựa chọn tốt nhất ở mỗi bước sẽ dẫn tới lời giải tối ưu nhất.

Vậy thì chọn lựa tối ưu hóa bằng cách nào?

Giải sử bạn có một hàm cần để tối ưu hóa (cực đại hóa hoặc cực tiểu hóa). Một thuật toán tham lam sẽ thực hiện các lựa chọn tham lam ở mỗi bước để đảm bảo rằng hàm đã cho là tối ưu. Thuật toán tham lam chỉ có một lần tính toán lời giải tối ưu với mục đích nó không bao giờ trở lại và đảo ngược quyết định.

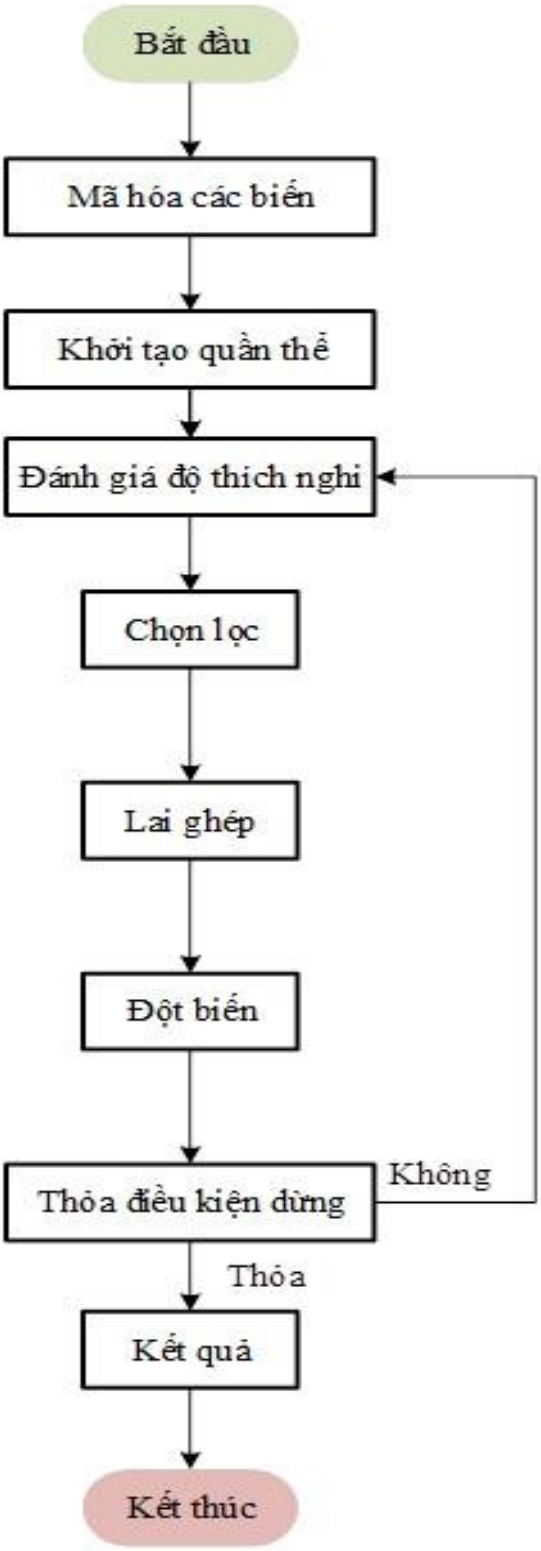
Meta-heuristic

- Thuật toán tiến hóa (Evolutionary Algorithms- EA)

Các thuật toán tiến hóa [1,2,3] là các kỹ thuật tối ưu Meta-Heuristics dựa trên nguyên lý của Darwin về sự lựa chọn tự nhiên. Thuật toán bắt đầu với nhóm các cá thể (gọi là quần thể) trả qua các thao tác (lai ghép, đột biến) tương tụ như quá trình sinh sản trong tự nhiên để tạo ra thế hệ các con cháu. Tiếp theo đó, các tương tác trên chính các cá thể đó bảo tồn tính di truyền làm cho một số cá thể phù hợp tốt hơn với "môi trường" và loại bỏ các cá thể xấu đối với "môi trường". Từ "môi trường" ở đây được sử dụng như một phép ẩn dụ cho ngữ cảnh của hàm mục tiêu đang được tối ưu hóa.

- Giải thuật di truyền (*Genetic Algorithms -GA*)

Trong số các giải thuật kể trên, thì giải thuật di truyền là một trong những mô hình tính toán phổ biến và thành công nhất trong lĩnh vực tính toán thông minh. Cùng với các kỹ thuật tính toán thông minh khác như tính toán mờ (*fuzzy computing*), mạng Nơ-ron (*neural networks*), hệ đa tác tử (*multiagent systems*), trí tuệ bầy đàn (*swarm intelligence*), giải thuật di truyền ngày càng phát triển, được áp dụng rộng rãi trong các lĩnh vực của cuộc sống. Tác giả sẽ giới thiệu sơ lược về GA và những bước cơ bản trong GA là gì. Giải thuật di truyền là một kỹ thuật dựa trên cách mô phỏng sự tiến hóa của con người hay của sinh vật nói chung (dựa trên thuyết tiến hóa muôn loài của Darwin – hay nói cách khác là cùng hệ tư tưởng với thuật toán tiến hóa) trong điều kiện quy định sẵn của môi trường. Ý tưởng của giải thuật di truyền để giải một bài toán tối ưu là tìm một tập hợp của những giải pháp, sau đó cho "tiến triển" theo hướng chọn lọc để tìm những giải pháp tốt dần hơn. Mục tiêu của giải thuật di truyền là đưa ra lời giải "tốt" có thể là tối ưu hay xấp xỉ tối ưu.



Hình 1-1: Sơ đồ khối cấu trúc thuật toán di truyền

Sơ đồ thuật toán GA tổng quát có thể được biểu diễn như sau:

Thuật toán: Giải thuật di truyền	
1	Khởi tạo biến đếm t = 0
2	Khởi tạo quần thể C(0) với n cá thể
3	**while** chưa gặp điều kiện dừng **do**
4	Với mỗi cá thể $x_i(t)$, tính độ thích nghi $f(x_i(t))$ với $i = 1..n$
5	Thực hiện lai ghép và đột biến
6	Lựa chọn ra quần thể mới $C(t+1)$
7	$t = t + 1$
8	**end while**
9	**return** Cá thể tốt nhất trong quần thể

- Kỹ Thuật mã hóa

Mã hóa trong giải thuật di truyền là biểu diễn các nhiễm sắc thể chứa thông tin cho lời giải. Một số cách mã hóa được sử dụng là: mã hóa nhị phân – Binary coding, mã hóa k mức – K-nary coding, mã hóa theo số thực – Real-number coding. Quá tình mã hóa có thể biểu diễn các đầu vào thành các dãy nhiễm ăsc thể theo mảng một chiều hoặc nhiều chiều.

Việc lựa chọn phương thức mã hóa tùy thuộc vào bài toán giải quyết. Thông thường hay dùng mã hóa nhị phân. Ví dụ dưới đây mô tả cách mã hóa các số thực thành các bit nhị phân:

- Chọn lọc

Ở mỗi thế hệ, dựa trên giá trị của hàm mục tiêu, các cá thể có độ thích nghi tốt sẽ được chọn lọc để tạo thành quần thể ở thế hệ mới và được chuẩn bị cho việc thực hiện các phép toán lai ghép và đột biến sau này. Mục đích của phép chọn lọc là tập trung sự tìm kiếm trên miền "hứa hẹn". Phép này bắt nguyền từ học thuyết của Darwin về "Sự sống sót của các cá thể thích nghi nhất". Có hai chiến lược chọn lọc quan trọng là:

+ Lựa chọn quần thể mới.
+ Lựa chọn cha mẹ.

Một số phép chọn lọc thường được sử dụng bao gồm:

+ Roulette wheel Selection - Chọn lọc ngẫu nhiên theo bánh xe Roulette;
+ Fitness Proportionate Selection- Chọn lọc theo tỷ lệ thích nghi;

+ Linear Ranking Selection- Chọn lọc theo thứ hạng tuyến tính;
+ Local Tournament Selection- Chọn lọc theo cạnh tranh cục bộ.

- Lai ghép

Trong giải thuật di truyền, số lượng các thể trong quần thể ở mỗi thế hệ là không đổi. Phép chọn lọc đã chọn ra một số cá thể có độ thích nghi cao và loại bỏ đi một số cá thể thích nghi thấp. Sự thiếu hụt của số lượng quần thể khi mất đi các cá thể thích nghi thấp sẽ được bổ xung bằng việc lấy các cá thể có độ thích nghi cao là thế hệ cha mẹ, tạo ra các thế hệ con cái bằng phép lai ghép và đột biến trên các cá thể thích nghi cao này.

Phép lai ghép là tạo ra các nhiễm sắc thể con cái (*offspring*) từ các nhiễm sắc thể cha mẹ (*parent*) được lựa chọn.

- Đột biến

Thuật toán tiến hóa được đề cập ở mục này chính là đối tượng nghiên cứu chính của tác giả trong bài viết này. Trong chương tiếp theo tác giả sẽ đi sâu hơn và phân tích chi tiết về thuật toán này cùng những yếu tố xung quanh đối tượng.

- Thuật toán bầy đàn (Partical Swarm Optimization - PSO)

PSO được khởi tạo bằng một nhóm cá thể (nghiệm) ngẫu nhiên $p_1,p_2,...,p_{Npop}$, và sau đó, tìm nghiệm tối ưu bằng cách cập nhật các thế hệ. Trong mỗi thế hệ, mỗi cá thể p_i được cập nhật theo hai giá trị tốt nhất. Giá trị thứ nhất là nghiệm tốt nhất ở thời điểm hiện tại, gọi là $pbest_i$. Một nghiệm tối ưu khác mà các cá thể này bám theo là nghiệm tối ưu toàn cục *gbest*, đây là nghiệm tốt nhất mà cá 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 theo vị trí tốt nhất của nó và vị trí tôt nhất của các thể trong quần thể tính đến thời điểm hiện tại. Đối với mỗi cá thể p_i chúng ta có:

- $X_i^t[]$ là vị trí của cá thể *i* ở thế hệ thứ *t*,
- $V_i^t[]$ là vận tốc của cá thể *i* ở thế hệ thứ *t*
- $pbest^t{}_i[]$ là vị trí tốt nhất của cá thể *i* ở thế hệ thứ *t*,

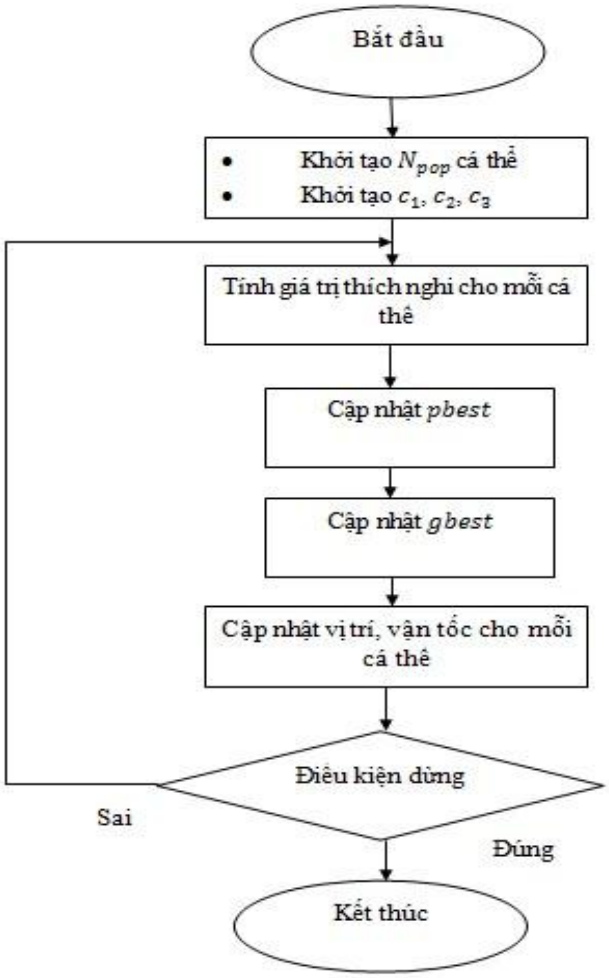
$$pbest_i^t[] = \begin{cases} pbest_i^{t-1}[], & nếu\ f(X_i^t[]) \le f(pbest_i^{t-1}[]) \\ X_i^t[], & nếu\ f(X_i^t[]) > f(pbest_i^{t-1}[]). \end{cases} \quad (1.3)$$

- *gbest*(*t*) là cá thể tốt nhất của tất cả các cá thể ở thế hệ thứ *t*

$$gbest^t[] \in \{pbest^t{}_1[],...,pbest^t{}_{Npop}[]\} | f(gbest^t[]) = max\{f(pbest^t{}_1[]),...,f(pbest^t{}_{Npop}[])\}. \quad (1.4)$$

Thuật toán PSO bao gồm ba bước chính được lặp đi lặp lại cho đến khi gặp điều kiện dừng [2].

1. Bước 1: Đánh giá độ thích nghi của các cá thể,
2. Bước 2: Cập nhật các thể tốt nhất ở thời điểm hiện tại, và các cá thể tốt nhất qua các thế hệ,
3. Bước 3: Cập nhật vận tốc và vị trí của mỗi cá thể.



Hình 1-2: Sơ đồ khối thuật toán PSO

Thuật toán PSO là một quá trình ngẫu nhiên, nên chúng ta không thể đảm bảo chắc chắn nó sẽ dừng sau hữu hạn bước. Vì vậy, để đảm bảo thuật toán PSO sẽ kết thúc, người dùng thường phải định nghĩa điều kiện dừng cho thuật toán. Một vài trường hợp dừng thông thường như sau:

1. Kết thúc theo kết quả: Một khi đạt đến mức giá trị yêu cầu thì kết thúc kết quả thực hiện.
2. Kết thúc dựa vào số lần lặp: Thuật toán dừng khi kết thúc một số hữu hạn các lần lặp.
3. Kết thúc theo thời gian: không quan tâm đến số lần lặp và kết quả như thế nào, giải thuật sẽ kết thúc sau một thời gian quy định nào đó.
4. Tổ hợp: dùng nhiều phương pháp khác nhau để kết thúc giải thuật: chạy theo số lần lặp, tiếp đến đánh giá cho kết quả....

• Thuật toán tìm kiếm cục bộ (Local search)[39, 40]

Kỹ thuật tìm kiếm cục bộ hay còn gọi là tìm kiếm địa phương, thực hiện bằng cách bắt đầu từ một phương án chấp nhận được, lặp lại bước cải tiến lời giải nhờ các thay đổi cục bộ. Để thực hiện kỹ thuật này, ta cần xác định được cấu trúc lân cận của mỗi phương án (lời giải) đang xét, tức là những phương án chấp nhận được, gần với nó nhất, nhờ thay đổi một số thành phần. Cách thưởng dùng là lân cần k-thay đổi, tức là lân cận bao gồm các phương án chấp nhận được khác với phương án đang xét nhờ thay đổi nhiều nhất k thành phần.

Bên cạnh những ưu điểm của giải thuật xấp xỉ, tìm kiếm cục bộ có nhược điểm là thường chỉ cho cực trị địa phương.

1.2 Thuật toán tiến hóa

Nói một cách đơn giản, trong một EA, các thành viên phù hợp sẽ tồn tại và sinh sôi nảy nở, trong khi các thành viên khác không còn phù hợp sẽ chết đi và không đóng góp vào nhóm gen của các thế hệ tiếp theo nữa, giống như chọn lọc tự nhiên.

Các thuật toán tiến hóa hầu hết đều được thiết kế để tập trung vào việc giải một ài toán tối ưu tại mỗi thời điểm dựa trên một quần thể mà chưa có sự quan tâm đến việc giải quyết nhiều bài toán tối ưu khác nhau đồng thời trên cùng một quần thể. Và do đó dẫn đến sự ra đời của một dạng hoàn toàn mới của các thuật toán tiến hóa, đó là đa tác vụ tiến hóa (evolutionary multitasking), cho phép giải đồng thời nhiều bài toán tối ưu khác nhau trên

một quần thể duy nhất và được gọi là tối ưu hóa đa nhân tố (multifactorial optimization-MFO) [29].

Chương 2 TIẾN HÓA ĐA NHÂN TỐ

2.1 Các khái niệm liên quan

Trong phần này, tác giả sẽ giải thích các khái niệm của multifactorial optimization (MFO) và mô tả hiệu suất tương đối của một cá nhân trong một dân số được đánh giá trong một môi trường đa tác vụ.

Định nghĩa 1 (*Factorial Cost* – giá của của một cá thể đối với từng tác vụ):

Factorial cost Ψ^i_j của cá thể pi đối với tác vụ T_j là $\Psi^i_j = \lambda.\delta_j^i + f_j^i + f_j^i$; trong đó λ là một hằng số phạt lớn, f_j^i và δ_j^i là giá trị mục tiêu và tổng số sự vi phạm ràng buộc tương ứng của pi đối với tác vụ T_j. Do vậy, nếu pi phù hợp đối với tác vụ T_j (sự vi phạm ràng buộc bằng 0), chúng ta có $\Psi^i_j = f_j^i$.

Định nghĩa 2 (*Factorial rank* – Xếp hạng của một cá thể đối với từng tác vụ):

Factorial rank r_j^i của cá thể pi trên tác vụ T_j là chỉ số (index) cuả p_i trong quần thể P được sắp sếp theo thứ tự tăng dần của Ψ^i_j.

Trong khi gán Factorial rank, mỗi khi $\Psi_j^a = \Psi_j^b$ đối với một cặp cá thể p_a và p_b, thứ tự trước sau được lựa chọn ngẫu nhiên. Tuy nhiên, hiệu năng của hai cá thể là tương đương đối với tác vụ thứ j^{th}, vì vậy gán nhãn chung như là $j-$ *counterparts* (bản sao hay giống nhau đối với tác vụ thứ j^{th}).

Định nghĩa 3 *(Scalar Fitness* – Đo độ thích nghi):

Danh sách các xếp hạng của một cá thể pi trên tất cả các tác vụ (K tác vụ) $\{r_1^i, r_2^i, \ldots, r_K^i\}$ có thể được biến đổi sang một hình thức đơn giản hơn là giá trị Scalar Fitness ϕ_i dựa trên thứ tự xếp hạng tốt nhất của p_i trên tất cả các tác vụ, nghĩa là $\varphi_i = 1/\min_{j\in\{1,2,\ldots,K\}}\{r_j^i\}$.

Định nghĩa 4 (*Skill Factor* - Tác vụ đầy đủ or tác vụ tốt nhất (nhân tố kỹ năng):

Skill factor τ_i của cá thể p_i là tác vụ trong số tất cả các tác vụ của MFO mà cá thể p_i là hiệu quả nhất, nghĩa là $\tau_i = argmin\{r_j^i\}$ trong đó $j \in \{1,2,...,K\}$.

Khi sự thích hợp của các cá thể được qui theo định nghĩa 3, so sánh hiệu năng có thể được thực hiện một cách dễ dàng. Ví dụ, p_a được xem là trội hơn p_b trong ngữ cảnh đa nhân

tố nếu $\phi_a > \phi_b$. Nếu hai cá thể có cùng skill factor $\tau_a = \tau_b = j$ (phù hợp với tác vụ thứ $j\{th\}$ nhất), gán nhãn chúng là strong counterparts (giống nhau mạnh).

Điều quan trọng cần lưu ý là các thủ tục được mô tả từ trước đến nay để so sánh các cá thể không phải là tuyệt đối. Vì factorial rank của một cá thể (và rõ ràng scalar finess và skill factor của nó) phụ thuộc vào hiệu suất của mọi cá thể khác trong quần thể, sự so sánh là phụ thuộc vào quần thể thực tế. Tuy nhiên, thủ tục đảm bảo rằng nếu một cá thể p^* ánh xạ tới sự tối ưu hóa toàn cục của một tác vụ bất kỳ thì $\phi^* \geq \phi_i$ với mọi $i \in \{1,2,...,K\}$.

Định nghĩa 5 (*Multifactorial optimality* - Sự tối ưu đa nhân tố): Một cá thể p^*, với một danh sách các giá trị mục tiêu $\{f_1^*, f_2^*, ..., f_K^*\}$, được xem là tối ưu hóa trong ngữ cảnh đa nhân tố nếu và chỉ nếu $\exists j \in \{1,2,...,K\}$ sao cho $f_{j^*} \leq f_j(x_j)$ đối với mọi x_j phù hợp trong X_j.

Tối ưu hóa đa yếu tố (MFO) vs Tối ưu hóa đa mục tiêu (MOO):

Vì tối ưu hóa đa mục tiêu và MFO đều liên quan đến việc tối ưu hóa một tập hợp các hàm mục tiêu, nên có thể thấy sự tương đồng về khái niệm tồn tại giữa chúng. Tuy nhiên, cần nhấn mạnh một sự khác biệt cơ bản giữa hai mô hình này. Trong khi Tiến hóa đa tác vụ nhằm thúc đẩy sự tương đồng ẩn tàng của quá trình tìm kiếm dựa trên dân số để khai thác sự bổ sung tìm ẩn giữa các tác vụ riêng biệt

2.2 Giải thuật tiến hóa đa nhân tố

Trong tiểu mục này, tác giả sẽ trình bày về Thuật toán tiến hóa đa nhân tố (MFEA), một bộ khung đa tác vụ có hiệu quả dựa trên các mô hình văn hóa sinh học của kế thừa đa yếu tố [34]. Vì các hoạt động của phương pháp này dựa trên việc truyền các yếu tố sinh học cũng như các khối văn hóa từ phía bố mẹ sang con cái, MFEA được coi là thuộc về lĩnh vực tính toán ghi nhớ [35, 36] – một lĩnh vực mà gần đây đang nổi lên như một mô hình tính toán thành công tổng hợp các nguyên tắc chọn lọc tự nhiên của Darwin với khái niệm Dawkins về meme như là một đơn vị cơ bản của tiến hóa văn hóa [37].

2.3 Khởi tạo quần thể

Trong phần này, tác giả sẽ đi vào phân tích bộ mã giả của MFEA. Như đã chỉ ra ở mục 2.3.1, MFEA bắt đầu bằng cách tạo ngẫu nhiên một quần thể gồm n cá thể trong một không gian tìm kiếm đơn nhất Y . Ngoài ra, mỗi cá thể trong tập hợp quần thể đều được chỉ định trước một yếu tố kỹ năng cụ thể (Xem định nghĩa 4) theo cách đảm bảo cho mọi tác vụ đều

có số lượng đại diện hợp lý. Nhấn mạnh rằng yếu tố kỹ năng của một cá thể (nghĩa là, tác vụ mà cá thể đó được liên kết) được xem như là một đại diện tính toán của đặc điểm văn hóa được chỉ định trước đó. Ý nghĩa của bước này là đảm bảo rằng một cá thể chỉ được đánh giá liên quan đến một tác vụ duy nhất (nghĩa là chỉ có nhân tố kỹ năng của nó) trong số tất cả các tác vụ khác trong môi trường đa tác vụ. Làm như vậy được coi là thiết thực vì việc đánh giá toàn bộ từng cá thể cho mọi tác vụ thường sẽ đòi hỏi tính toán, đặc biệt là khi K (số lượng tác vụ trong môi trường đa tác vụ) trên nên lớn hơn.

Thuật toán 1: Cấu trúc cơ bản của MFEA

1. Khởi tạo quần thể ban đầu và lưu trữ trong *current-pop* (*P*)
2. Đánh giá mọi cá thể đối với mọi tác vụ tối ưu trong môi trường đa tác vụ
3. Tính toán skill factor τ của mỗi cá thể
4. **While** (điều kiện dừng chưa thỏa mãn) **do**
   - i) Áp dụng các thao tác di truyền tới *current-pop* (*P*) để sinh ra *offspring-pop* (*C*) [*Xem thuật toán 2*]
   - ii) Đánh giá các cá thể trong *offspring-pop* **chỉ đối với các tác vụ được lựa chọn** [*Xem thuật toán 3*]
   - iii) Gộp *current-pop* (*P*) và*offspring-pop* (*C*) tạo thành *intermediate-pop* (*P*∪*C*)
   - iv) Cập nhật *scalar fitness* và *skill factor* cho mọi cá thể trong *intermediate-pop* (*P*∪*C*)
   - v) Lựa chọn các cá thể tốt nhất từ *intermediate-pop* (*P*∪*C*) để hình thành *current-pop* (*P*) kế tiếp.
5. **End while**

Hình 2-1: Mô tả các bước tổng thể của các EA(thuật toán 1)

2.4 Kỹ thuật di truyền

Tương tự như các thuật toán EA cổ điển, MFEA cũng sử dụng hai toán tử di truyền đó là *lai ghép (crossover)* và *đột biến (mutation)*. Điểm khác biệt chính của MFEA là hai cá thể cha mẹ được lựa chọn ngẫu nhiên cho phép toán lai ghép phải đáp ứng một số điều kiện. Nguyên tắc tiếp theo là việc lai ghép ngẫu nhiên chỉ ra rằng các cá thể thích kết hợp với cá thể thuộc cùng một kiểu "văn hoá": Trong MFEA, *skill factor* (τ) được xem như là một đại diện tính toán của sự thiên vị văn hoá *(culture bias)* của một cá thể. Do vậy, hai cá thể cha mẹ được lựa chọn ngẫu nhiên chỉ được thực hiện lai ghép nếu chúng có cùng *skill factor*. Ngược lại, nếu *skill factor* là khác nhau, sự lai ghép chỉ xảy ra theo một xác suất lai ghép

ngẫu nhiên được qui định *(rmp)* hoặc phép đột biến được sử dụng. Các bước tạo ra các cá thể con được mô tả trong thuật toán 2.

Thuật toán 2: Các thao tác di truyền

Xét hai cá thể cha mẹ p_a và p_b được lựa chọn ngẫu nhiên trong *current-pop* (P)

Sinh một số ngẫu nhiên *rand* trong khoảng [0,1]

If ($\tau_a = \tau_b$) hoặc ($rand < rmp$) **then**

i) Thực hiện lai ghép hai cá thể p_a và p_b sinh ra hai cá thể con c_a và c_b

else

i) Đột biến p_a sinh ra cá thể con c_a

ii) Đột biến p_b sinh ra cá thể con c_b

End if

Hình 2-2: Mô tả các bước tổng thể của các EA (thuật toán 2)

2.5 Đánh giá có chọn lọc

Trong khi đánh giá một cá thể đơn lẻ cho một tác vụ T_j, bước đầu tiên là giải mã các khóa ngẫu nhiên của nó thành một đầu vào có ý nghĩa cho tác vụ đó. Nói cách khác, biểu diễn khóa ngẫu nhiên đóng vai trò là không gian tìm kiếm thống nhất, từ đó có thể suy ra các biểu diễn giải pháp cụ thể cho vấn đề. Đối với trường hợp tối ưu hóa liên tục, điều này được hoàn thành một cách đơn giản. Ví dụ, hãy xem xét biến thứ i(xi) của vấn đề thực tế được giới hạn trong phạm vị $[L_i,U_i]$. Nếu khóa ngẫu nhiên tương ứng của một cá nhận có giá trị y_i thì ánh xạ của nó ở trong không gian tìm kiếm của vấn đề thực tế được đưa ra bởi$x_i = L_i + (U_i - L_i).y_i$. Ngược lại, đối với trường hợp tối ưu hóa rời rạc, sơ đồ giải mã nhiễm sắc thể thường phụ thuộc vào vấn đề.

Thuật toán 3: Di truyền văn hóa theo chiều dọc thông qua sự bắt chước có chọn lọc

Một cá thể 'c' hoặc có cá thể cha mẹ p_a và p_b hoặc một cha mẹ (p_a hay p_b)

1 **If** ('c' có hai cha mẹ) **then**

i) Sinh một số ngẫu nhiên *rand* giữa 0 và 1

ii) **If** (*rand* <0.5) **then**

'c' bắt chước p_a: cá thể con được đánh giá chỉ cho tác vụ có thứ tự là τ_a

iii) **else**

'c' bắt chước p_b: cá thể con được đánh giá chỉ cho tác vụ có thứ tự là τ_b

iv) **end if**

2 **Else**

i) 'c' bắt chước theo cá thể cha mẹ (đột biến): cá thể con được đánh giá chỉ cho tác vụ có thứ tự τ của cá thể cha mẹ đột biến tạo ra nó

3 **end if**

4 Factorial cost của cá thể c trên các tác vụ còn lại được đặt là ∞ (một số rất lớn)

Hình 2-3: Mô tả các bước tổng thể của EA (thuật toán 3)

2.6 Sự lựa chọn

Như được thể hiện trong Thuật toán 1, MFEA tuân theo chiến lược tinh hoa, đảm bảo rằng những cá thể tốt nhất sẽ tồn tại qua các thế hệ. Để xác định các cá nhân tốt nhất, MFEA cũng lựa chọn các cá thể tốt hơn cho thế hệ kế tiếp, tức là các cá thể có giá trị ϕ (*scalar fitness*) lớn hơn.

Tóm tắt các tính năng nổi bật của MFEA

Các EA tiêu chuẩn thường tạo ra lượng lớn quần thể của lời giải đề cử, tất cả gần như không đủ sức cho tác vụ trong tầm tay. Ngược lại trong một môi trường đa nhiệm, khả năng cao một cá thể được tạo ngẫu nhiên hoặc biến đổi có khả năng đủ sức hoàn thành ít nhất một tác vụ. Cơ chế của MFEA xây dựng dựa trên quan sát bằng cách phân chia quần thể thành các nhóm kỹ năng khác nhau, mỗi nhóm xuất sắc trong một tác vụ khác nhau. Thú vị hơn và quan trong hơn là, có thể có vật liệu di truyền được tạo ra trong một nhóm hóa ra cũng hữ ích cho một tác vụ khác. Do đó, trong các tình huống như vậy, phạm vi chuyển gen qua các tác vụ có thể có khả năng dẫn đến việc không khó phát hiện ra tối ưu hóa toàn cục. Trong MFEA, việc chuyển giao vật liệu di truyền được tạo điều kiện bằng cách cho phép các nhóm kỹ năng khác nhau giao tiếp với nhau một cách có kiểm soát, thông qua thỉnh thoảng trao đổi nhiễm sắc thể.

Chương 3 ÁP DỤNG THUẬT TOÁN TIẾN HÓA ĐA NHÂN TỐ ĐỂ GIẢI CÁC BÀI TOÁN TỐI ƯU ĐƠN MỤC TIÊU

Như mục tiêu ban đầu, đề tài sẽ không quá đi chi tiết vào thực nghiệm của việc áp dụng thuật toán trên một bài toán cụ thể nào, thay vào đó, sẽ cố gắng giải thích chi tiết các khái niệm, các bước thực hiện của thuật toán. Thêm vào đó, vấn đề quan trọng nhất cho người muốn thiết kế các thuật toán EA là làm sao biểu diễn mỗi giải pháp của bài toán thực tế thành một cá thể trong thuật toán EA và ngược lại. Vì vậy, phần này tập trung vào việc giới thiệu cách biểu diễn (mã hóa) các giải pháp của các bài toán cụ thể vào cùng "không gian tìm kiếm hợp nhất" và ngược lại (giải mã). Ý tưởng mã hóa và giải mã của thuật toán MFEA có thể minh họa như hình 5. Theo đó, trong với EA truyền thống, mỗi tác vụ có một sự biểu diễn cụ thể (Specific Problem Representation), từ đó tìm ra các lời giải cụ thể (Specific Problem Solvers). Khi chuyển sang MFEA, tất cả các sự biểu diễn của các tác vụ được được chuyển thành một biểu diễn hợp nhất (Unified Representation), và từ đó tìm ra lời giải tổng quát (General Solver).

3.1 Bài toán Knapsack và bài toán Quadratic Assignment Problem

3.1.1 Bài toán Knapsack

Đầu vào:

- n: số lượng đồ vật
- W: trọng lượng tối đa của ba lô
- c = {c1,c2,...,cn}: véc tơ lưu trọng lượng của các vật
- V = {v1,v2,...,vn}: véc tơ lưu giá trị của các vật

Đầu ra:

- Danh sách các vật được chọn cho vào ba lô Mục tiêu:
- Tổng trọng lượng thu được của ba lô là lớn nhất

Ràng buộc

- Tổng trọng lượng của các vật được chọn không vượt quá trọng lượng của ba lô

3.1.2 Bài toán Quadratic Assignment

Bài toán QAP [31] được phát biểu như sau: QAP có thể phát biểu như sau: Đưa ra *n* phương tiện và n vị trí với khoảng cách giữa các vị trí là *d*(*i,j*) và yêu cầu luồng (flow) từ vị trí i đến vị trí j là *f*(*i,j*).

Đầu vào:

- n: số lượng vị trí và số lượng cơ sở định đặt tại các vị trí
- d(n×n): ma trận khoảng cách dij khoảng cách giữa vị trí i và vị trí j.
- f(n×n): là luồng yêu cầu giữa cơ sở i và vị trí j

Đầu ra:

- Cách bố trí các cơ sở vào các vị trí. Hay nói cách khác đầu ra là một hoán vị của tập *n* phần tử ($\varphi_1,\varphi_2,...,\varphi_n$). Trong đó φ_i cho biết cơ sở nào được đặt tại vị trí *i*.

Mục tiêu:

- Tổng chi phí là rõ nhất, nghĩa là $\Sigma_{i=1}^{n}\Sigma_{j=1}^{n} f(i,j).d(\varphi(i),\varphi(j)) \rightarrow min$.

3.2 Áp dụng thuật toán tiến hóa đa nhân tố để giải đồng thời hai bài toán Knapsack và bài toán Quadratic Assignment Problem

Trong các thuật toán EA truyền thống cho bài toán KP, mỗi một cá thể được biểu diễn bằng một vectơ $x = \{x_1,x_2,...x_n\}$ của các biến nhị phân, trong đó $x_i = 1$ chỉ ra rằng đồ vật thứ ith được chọn để cho vào ba lô, ngược lại, không cho đồ vật thứ ith vào ba lô. Một cách đơn giản để suy luận các biến nhị phân từ các khóa ngẫu nhiên là $x_i = 1$ nếu khóa ngẫu nhiên $y_i \geq 0.5$, ngược lại $x_i = 0$.

3.3 Kết quả mô phỏng

Thuật toán được cài đặt bằng ngôn ngữ lập trình Matlab. Thông số máy tính sử dụng: hệ điều hành Ubuntu, CORE I5, RAM 4GB.

3.3.1 Dữ liệu

- Bài toán QAP: Dữ liệu được khởi tạo ngẫu nhiên 6 bộ dữ liệu với ma trận khoảng cách và ma trận giá trị. Các bộ dữ liệu có kích thước số vị trí lần lượt là 4, 5, 6, 7, 8, 9.
- Bài toán KP: Dữ liệu tạo ngẫu nhiên 6 bộ dữ liệu với véc tơ trọng số và véc tơ giá trị. Các bộ có kích thước là 4, 5, 6, 7, 8, 9

Các bài toán sẽ giải kết hợp:

Bảng 3-1: Kích thước các bài toán sẽ kết hợp giải

Test Bài toán KP	Bài toán QAP	
Test 1	9	4
Test 2	8	5
Test 3	7	6
Test 4	6	7
Test 5	5	8
Test 6	4	9

3.3.2 Tham số thực nghiệm

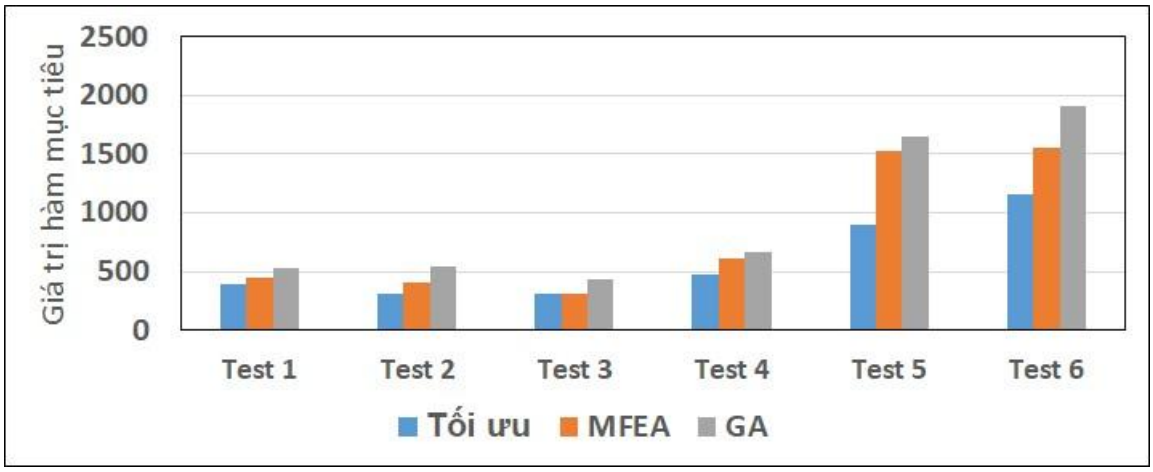
Bảng 3-2: Tham số thực nghiệm

Kích thước quần thể	100
Xác suất lai ghép	0.8
Xác suất đột biến	0.1
Số thế hệ	100

3.3.3 Kết quả thực nghiệm

Luận văn sẽ so sánh kết quả đạt được theo các tiêu chí sau:

- Đánh giá kết quả của thuật toán tối ưu đa nhân tố so với giải thuật di truyền trên bài toán QAP và KP



Hình 3-1: So sánh kết quả cảu MFGA với GA và lời giải tối ưu trên bài toán QAP

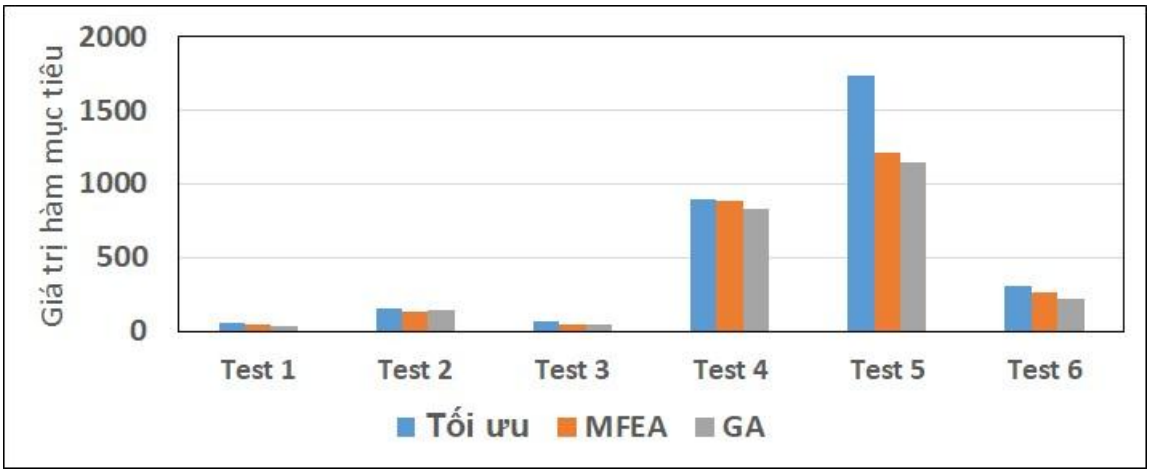
- Đánh giá kết quả của giải thuật tối ưu đa nhân tố và lời giải tối ưu (luận án tìm giải thuật tối ưu sử dụng thuật toán vét cạn) trên bài toán QAP và KP

Hiệu quả của MFEA trên bài toán QAP

Nhìn vào hình 3.2 chúng ta thấy rằng, MFEA cho kết quả tốt hơn GA trên hầu hết các bộ dữ liệu. Kết quả cho thấy rằng, MFEA tốt hơn GA trung bình khoảng 21%. Lời giải của MFGA khá sát với tối ưu. Trên test 3, MFEA cho kết quả bằng với tối ưu

Hiệu quả của MFEA trên bài toán KP

Nhìn vào hình 3.3 chúng ta thấy rằng, MFEA cho kết quả tốt hơn GA trên hầu hết các bộ dữ liệu. Kết quả cho thấy rằng, MFEA tốt hơn GA trung bình khoảng 5%. Lời giải của MFGA khá sát với tối ưu (bằng 92% tối ưu).



Hình 3-2: So sánh kết quả của MFGA với GA và lời giải tối ưu trên bài toán KP

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả đạt được

Bài toán tối ưu tổ hợp là bài toán khó, đã được nhiều các tác giả nghiên cứu trong thời gian gần đây. Luận văn tập trung vào việc tìm thiểu thuật toán tiến hóa đa nhân tố để giải quyết bài toán tối ưu đơn mục tiêu. Những kết quả đạt được của luận văn như sau:

- Thứ nhất: Luận văn trình bày được những kiến thức tổng quan về bài toán tôi ưu, các cách giải bài toán ối ưu
- Thứ hai: Luận văn đã tìm hiểu sơ đồ chung của thuật toán tiến hóa đa nhân tố để giải bài toán tối ưu hóa đơn mục tiêu.
- Thứ ba: Luận văn tìm hiểu cách áp dụng thuật toán tiến hóa đa nhân tố để giải quyết đồng thời hai bài toán là KP và QAP. Kết quả mô phỏng trên các bộ dữ liệu cho thấy thuật toán hoạt động tốt, cho kết quả sát với tối ưu

Vấn đề tồn đọng và hướng phát triển

- Thực nghiệm trên các bộ dữ liệu lớn hơn để đưa ra được những đánh giá khách quan.
- Tìm hiểu thuật toán tiến hóa đa nhân tố giải quyết các bài toán đơn mục tiêu trong thực tế.
- Tìm hiểu để thuật toán tiến hóa đa nhân tố để giải quyết các bài toán tối ưu hóa đa mục tiêu.