# CHƯƠNG 1. GIỚI THIỆU

## 1.1 Bài toán lập kế hoạch gia công

### 1.1.1 Đầu vào

### 1.1.1.1 Danh sách lô hàng

bao gồm các lô hàng có cùng một loại vật liệu.

Ví dụ:

* Lô hàng 1
  + 10 thanh sắt loại A chiều dài 3000mm
  + 20 thanh sắt loại A chiều dài 2000mm
* Lô hàng 2
* 10 thanh sắt loại A chiều dài 5000mm
* 20 thanh sắt loại A chiều dài 7000mm

### 1.1.1.2 Danh sách vật liệu

bao gồm các vật liệu có loại giống với danh sách lô hàng cần gia công, mỗi thanh sắt sẽ có ngày nhập kho hoặc ngày dự kiến nhập kho cố định để phục vụ cho việc tính toán.

* Vật liệu gốchiện tại (B1): các thanh sắt gốc trong kho với chiều dài ban đầu là 11700mm.
* Vật liệu dự định nhập kho => Vật liệu gốc dự định (B2): là các thanh vật liệu gốc trong tương lai (dự định nhập kho) có chiều dài giống vật liệu gốc.
* Vật liệu còn lại (R) => Vật liệu thừa hiện tại (R1): các thanh sắt còn thừa từ các lần gia công trước đó, có chiều dài từ khoảng 500mm đến bé hơn 11700mm.
* Vật liệu dự định (Y) => Vật liệu thừa dự định (R2): các thanh sắt còn thừa đã nằm trong kế hoạch gia công, *nhưng chưa thực sự tồn tại mà sẽ được sản sinh trong các lần gia công tiếp theo*, có chiều dài từ khoảng 500mm đến 11700mm.

### 1.1.1.3 Độ dày lưỡi dao

Mỗi nhát cắt được áp dụng trên mỗi thanh sắt thì sẽ có độ hao mòn nhất định bởi độ dày của lưỡi cưa cắt sắt.

- độ dày lưỡi cưa: mỗi máy cắt sẽ có một lưỡi cưa, lưỡi cưa sẽ có độ dày nhất định.

Ví dụ, một thanh vật liệu có chiều dài 10000mm, được đưa vào máy cắt có độ dày lưỡi cưa là 5mm, cần cắt ra 1 sản phẩm có độ dài 3000mm. Ta sẽ có được độ dài phần thừa còn lại của thanh sắt: remain = 10000 – 3000 – 5 = 6995mm.

### 1.1.2. TIÊU CHÍ

Bài toán trên gồm có các tiêu chí lựa chọn sau:

* Tiết kiệm: ưu tiên chọn vật liệu thừa để cắt, nếu không đủ thì sẽ dùng thêm vật liệu gốc, mục tiêu là phần thừa còn lại phải bé nhất có thể
* Nhanh: sẽ ưu tiên chọn các thanh nguyên liệu gốc (các thanh sắt có sẵn hay dự định nhập kho), nếu các thanh vật liệu gốc không đủ thì sẽ dùng đến các thanh vật liệu thừa, mục tiêu là số lượng thanh sắt đem đi gia công ít nhất có thể

### 1.1.3. ĐẦU RA

* Chuổi số chỉ định thanh đơn hàng được cắt bởi thanh đơn hàng nào.
* Số lượng thanh nguyên liệu cần dùng
* Tỉ lệ dư thừa

## 1.2 Một số nghiên cứu liên quan

### 1.2.1 Cắt vật liệu dạng thanh đồng nhất

Bài báo [1] đã trình bày phương pháp tối ưu cắt vật liệu dạng thanh đồng nhất, các thanh đầu vào có chiều dài như nhau. Để thực hiện điều đó, bài báo đã nêu phương pháp thiết lập mối quan hệ giữa số lượng các sản phẩm cắt được với số lượng TSNL, phương pháp xây dựng các hàm số thể hiện các điều kiện ràng buộc. Phương pháp được thực hiện theo ba bước:

- Xác định số lượng các cách cắt

- Xác định phương án tối ưu trong mỗi cách cắt (khi thực hiện dùng lệnh: Constrained Max[func,ineqs,vars] của Mathematica.

- Tổng hợp kết quả các cách cắt tối ưu từ đó xác định các điều kiện ràng buộc.

Việc sử dụng Mathematica giải tối ưu bài toán trên cơ sở các điều kiện ràng buộc vừa thiết lập là một hướng tiếp cận tiên tiến, cho phép nhanh chóng xác định được phương án tối ưu cắt vật liệu mà phương pháp cắt vật liệu truyền thống phải mất nhiều thời gian và rất khó thực hiện. Phương pháp có phạm vi ứng dụng rộng trong công nghiệp, dân dụng, thuận lợi trong sử dụng.

Ngoài ra, các hàm số thể hiện các ràng buộc ở trên cũng có thể phục vụ cho việc giải quyết bài toán bằng thuật toán quy hoạch tuyến tính bằng phương pháp đơn hình.

Bên cạnh những ưu điểm trên thì hai cách giải quyết bài toán này lại có một hạn chế lớn đó là chỉ giải quyết được bài toán khi mà dữ liệu đầu vào (nguyên liệu đầu vào để phục vụ việc cắt vật liệu thanh) là đồng bộ, có chiều dài như nhau. Trong thực tế, việc cắt các vật liệu dạng thanh không phải lúc nào cũng chỉ dùng các thanh nguyên liệu mới mà còn dùng những thanh vật liệu còn thừa từ những lần cắt trước đang tồn ở trong kho. Vậy nên hai phương pháp này chưa phải là phương pháp tốt có thể bao quát hết được các trường hợp trong thực tế.

### 1.2.2 Cắt vật liệu dạng thanh không đồng nhất

Để cắt các phần tử tuyến tính như thanh thép hoặc kệ đá cẩm thạch từ chiều dài tiêu chuẩn, thường xuyên phải tối ưu hóa để tận dụng tốt nhất nguyên liệu thô để giảm thiểu chất thải và giảm chi phí sản xuất với việc đóng gói các phần tử nhỏ hơn theo chiều dài tiêu chuẩn. Vì nguyên liệu đầu vào có thể phụ thuộc vào đơn đặt hàng, việc xử lý tại cửa hàng chỉ giới hạn ở những phần còn sót lại tốt từ các cành giâm trước đó hoặc tồn kho quá nhiều. Để xử lý sản xuất loại này một cách hợp lý, giải pháp bao gồm một số loại tối ưu hóa. Việc đưa lý thuyết vào hoạt động sản xuất thực tế đòi hỏi một số khía cạnh ứng dụng được xem xét để phù hợp với nhu cầu của các nhà thiết kế và nhân viên đặt hàng.

Trong bài báo [2], tác giả mô tả mô hình tối ưu hóa hoàn chỉnh NP. Về mặt lý thuyết, việc cắt giảm các kích thước tiêu chuẩn được gọi là bài toán đóng gói thùng rác với một số cách tiếp cận khác nhau từ các giải pháp xấp xỉ di truyền đến phương pháp heuristic. Sản xuất thực tế cũng đòi hỏi phải lưu trữ những phần còn lại tốt để có thể sử dụng trong tương lai, điều này cũng làm phức tạp thêm vấn đề tính toán chuyên sâu này. Để xem xét lưu trữ và vẫn cho kết quả hữu ích, vấn đề được chia thành tối ưu hóa gói và tối ưu hóa đóng gói thùng. Với giải pháp xử lý nhiều vấn đề về bao gói, tác giả sẽ kiểm tra giải pháp từng phần từ cửa hàng sau đó giảm bớt vấn đề đối với việc đóng gói thùng. Đóng gói thùng được giữ trên một số chiều dài tiêu chuẩn để tối ưu hóa. Việc chuyển đổi độ dài đầu vào được đưa vào để cho phép bổ sung cắt. Thép kết cấu được cắt theo kích thước ưa thích từ các chiều dài tiêu chuẩn. Cách sản xuất này giảm thiểu chi phí, vận chuyển hậu cần và cho phép sản xuất linh hoạt. Do đó, việc cắt thanh được lên kế hoạch ở giai đoạn thiết kế, nơi nguyên liệu sản xuất không có trong kho và cách ưu tiên là đặt hàng các chùm có chiều dài tiêu chuẩn và sau đó cắt nó với lượng chất thải giảm thiểu. Sau khi biết kích thước, tổng quan được chuẩn bị cho sản xuất và đơn đặt hàng. Kế hoạch sản xuất chi tiết được chuẩn bị cho sản xuất và chỉ rõ nguồn cắt từ đơn đặt hàng hoặc cửa hàng. Cửa hàng thường cung cấp dầm có chiều dài không tiêu chuẩn do kết quả của việc giâm cành trước đó. Cửa hàng chỉ được sử dụng cho những thứ còn sót lại tốt, việc cắt giảm và tồn kho quá mức để dự trữ từ các đơn đặt hàng trước. Việc kết hợp thủ công các độ dài quy định thành độ dài tiêu chuẩn có thể là một công việc tẻ nhạt, dễ xảy ra sai sót và không tối ưu. Nếu phải xem xét một số độ dài tiêu chuẩn, việc tìm kiếm giải pháp tốt nhất cũng tốn thời gian. Sự lựa chọn rõ ràng về sức mạnh tính toán dẫn đến sự phát triển của một số phương pháp để tấn công các nhu cầu cụ thể và tính linh hoạt trong tối ưu hóa.

Khi cắt dầm, tàn dư có thể đáng kể ngay cả khi một số chiều dài tiêu chuẩn được sử dụng để tối ưu hóa. Do đó, việc lưu trữ chùm tia từng phần này để tái sử dụng trong tương lai là chính đáng. Do đó, tối ưu hóa cũng phải xem xét các mặt hàng trong kho. Việc tối ưu hóa việc sử dụng cửa hàng bao gồm các bước quan trọng sau:

1. Đối với một hình dạng nhất định của dầm, người ta phải xác định nhịp của các chiều dài (tối thiểu / tối đa) trên kho. Từ đơn đặt hàng, các mục được chọn phù hợp với khoảng thời gian lưu trữ để tối ưu hóa MSSP. Điều này loại bỏ các mục rõ ràng không được trang bị khỏi tối ưu hóa mà được để lại cho tối ưu hóa BPP.
2. Thuật toán cho một MSSP được lựa chọn theo loại bài toán được chỉ định bởi số hạng mục n, số gói dữ liệu m và loại gói dữ liệu (dung lượng dày đặc, dung lượng được nhóm lại, tổng quát). Nếu thương số n / m tương đối lớn (> 10) và m <12 thì áp dụng phương pháp chính xác với lập trình động. Sơ đồ xấp xỉ thời gian đa thức (PTAS) [*Caprara et al., 1999*] được áp dụng cho các kích thước khác và xem xét loại ba lô.
3. Các mục không được gán cho knapsacks và các mục không được sử dụng trong tối ưu hóa MSSP được sử dụng trong tối ưu hóa BPP. Ở đây độ dài tiêu chuẩn tốt nhất không được biết và được chọn bằng cách quét trên phạm vi độ dài. Không có gợi ý nào về phương pháp đóng gói và có thể được lựa chọn theo kinh nghiệm về hiệu suất của phương pháp trong quá trình tối ưu hóa.
4. Nếu độ dài chuẩn được chọn từ tối ưu hóa BPP và độ dài đã chọn cũng được sử dụng trong MSSP từ cửa hàng, thì vấn đề sẽ được giải quyết bằng cách loại bỏ các knapsacks có độ dài chuẩn khỏi tối ưu hóa MSSP. Việc tối ưu hóa sau đó được lặp lại thường mang lại giải pháp tốt hơn.

Khi cắt dầm người ta cũng phải tính đến chiều rộng của vết cắt. Điều này rất quan trọng nếu có một số vết cắt và tổng tổng vượt quá dung sai kích thước chùm. Tối ưu hóa với việc bổ sung cắt có thể được thực hiện bằng cách sử dụng biến đổi đơn giản như trong hình.



Hình1.1 Biến đổi bổ sung cắt thêm kích thước cắt vào cácmục(trêncùng) và vào thùng (dưới cùng)

Để đảm bảo rằng mục cuối cùng sẽ vẫn vừa với thùng ngay cả khi không có vết cắt ở cuối, kích thước thùng cũng được mở rộng để cắt thêm. Chuyển đổi này có thể áp dụng cho tối ưu hóa BPP và MSSP. Biến đổi nghịch đảo là đơn giản.

Sử dụng ứng dụng trong thực hiện công nghiệp có chương trình tối ưu hóa để xem xét hầu hết các tình huống dữ liệu đầu vào. Điều này đặc biệt đúng đối với phần tối ưu hóa MSSP trong đó loại tối ưu hóa phụ thuộc vào kích thước và loại mặt hàng trong cửa hàng.

Các vấn đề khác gây ra giao tiếp với cửa hàng vì việc lưu trữ mặt hàng phải được điều chỉnh để đặt trước, chạy thử và loại bỏ tối ưu hóa. Lựa chọn các mặt hàng từ cửa hàng cũng có thể là một điểm quan trọng thường đòi hỏi sự quyết định của người dùng. Số lượng các mục được lưu trữ có thể tăng đến kích thước mà việc tối ưu hóa MSSP có thể bị nghẹt thở. Trong số đó có thể có nhiều phần tử có độ dài tiêu chuẩn. Cũng có trường hợp MSSP có thể chiếm ưu thế trong việc lựa chọn và để lại một số lượng nhỏ các mặt hàng cho BPP, nhưng chất lượng đóng gói là không thể chấp nhận được và người dùng buộc phải bỏ một số hoặc toàn bộ MSSP truyền tối ưu hóa cho BPP. Các thuật toán di truyền cho BPP hoạt động tốt nhất trên kích thước dữ liệu điển hình. Khi thời gian phản hồi được ưu tiên để tính toán sơ bộ, các thuật toán khác nhanh hơn có thể được chọn. Điều quan trọng không cần lưu ý là đối với hàm chi phí GGA (2) có xu hướng giảm thiểu chất thải vì nó thúc đẩy các thùng được đóng gói tốt hơn. Loại tối ưu hóa này để lại những tàn dư tốt hơn cho cửa hàng.

## 1.3 Đóng góp của khóa luận

## 1.4 Nội dung của khóa luận

# CHƯƠNG 2. THUẬT TOÁN ĐƯỢC ĐỀ XUẤT

## 2.1 Cắt nhanh

Giải thuật nhận vào các dữ liệu như đã mô tả ở phần giới thiệu, và sẽ tính toán trả về chi tiết cách gia công cho từng thanh đơn hàng. Trong đó sẽ chỉ rõ thanh đơn hàng nãy sẽ được cắt bởi thanh nguyên liệu nào, cắt bởi máy cắt số bao nhiêu và cắt tại thời điểm nào.

Kết quả trả về của thuật toán là ba mảng có số lượng phần tử bằng với số lượng danh sách thanh đơn hàng:

* Mảng lưu vị trí các thanh nguyên liệu: trong đó giá trị của các phần tử trong mảng là số thứ tự hay vị trí của thanh nguyên liệu trong danh sách các thanh nguyên liệu.
* Mảng lưu vị trí các máy cắt: trong đó giá trị của các phần tử trong mảng là số thứ tự hay vị trí của máy cắt trong danh sách các máy cắt phù hợp.
* Mảng lưu thời gian cắt các thanh đơn hàng: trong đó giá trị của các phần tử trong mảng là thời gian dự kiến cắt các thanh đơn hàng trên máy cắt.

Phương pháp cắt nhanh có hai hàm mục tiêu chính được sắp xếp theo thứ tự ưu tiên sau:

* Số lượng các thanh nguyên liệu đem ra gia công: mục tiêu là phải ít nhất có thể
* Tổng chiều dài các thanh nguyên liệu còn lại sau khi cắt: Chiều dài còn lại cũng phải ít nhất trong các phương án đảm bảo hàm mục tiêu trên.

Cắt nhanh được thực hiện theo chiến thuật tham lam. Các bước của giải thuật được minh họa ở *Hình 3.1* bao gồm:

1. Sắp xếp thanh nguyên liệu: tiến hành sắp xếp các thanh nguyên liệu theo chiều dài giảm dần.
2. Sắp xếp thanh đơn hàng: tương tự như bước 2, sắp xếp các thanh đơn hàng theo chiều dài giảm dần.
3. Có thanh nguyên liệu dài hơn thanh đơn hàng: kiểm tra nếu không có thanh nguyên liệu nào lớn hơn thanh đơn hàng thì chuyển đến bước 6; ngược lại chuyển đến bước 4.
4. Tìm phương án: sau khi qua được bước 3, tiến hành tìm một phương án với danh sách các thanh nguyên liệu hiện tại (chi tiết sẽ được trình bày ở mục 3.1.2).
5. Bỏ thanh nguyên liệu dài nhất: Sau khi qua bước 4, tiến hành bỏ đi một thanh nguyên liệu dài nhất trong danh sách các nguyên liệu ban đầu để tiếp tục quá trình tìm các phương án có thể cắt.
6. Chọn phương án tốt nhất: sau khi duyệt hết mảng danh sách nguyên liệu và có được các phương án cắt, tiến hành chọn ra một phương án theo tiêu chí ưu tiên là số lượng thanh nguyên liệu cần dùng là ít nhất, nếu có nhiều phương án thỏa mãn tiêu chí đầu, chọn ra phương án mà cho ra tổng chiều dài các thanh nguyên liệu thừa ít nhất.



Hình 2.1 Sơ đồ khối giải thuật cắt nhanh

Khi nhận được các thanh nguyên liệu được trình bày ở trên, tiến hành đi tìm phương án cắt cho bài toán.



Hình 2. 2 Tìm phương án cắt – cắt nhanh

Các bước của phương thức tìm phương án cắt được minh họa ở *hình 3.2* bao gồm:

1. Bước 2 – Đưa thanh nguyên liệu dài nhất lên máy: Chọn thanh dài nhất trong danh sách nguyên liệu được đưa vào để đặt lên máy cắt.
2. Bước 3 – Có thanh đơn hàng ngắn hơn: Kiểm tra lần lượt các thanh trong danh sách đơn hàng, nếu có thanh đơn hàng ngắn hơn hoặc bằng thanh trên máy cắt thì chuyển đến bước 4; ngược lại chuyển đến bước 5.
3. Bước 4 – Cắt theo thanh đơn hàng dài nhất có thể: cắt thanh nguyên liệu trên máy để tạo ra sản phầm là thanh đơn hàng đã được chọn. Sau đó tiếp tục quay lại bước 3 để tiếp tục cắt.
4. Bước 5 – Bỏ thanh nguyên liệu ra khỏi máy: Loại bỏ phần còn lại của thanh nguyên liệu trên máy cắt ra và cho nó vào danh sách vật liệu còn lại để phục vụ việc tính toán tổng chiều dài phần thừa sau khi tìm ra phương án cắt.
5. Bước 6 – Đã hết đơn hàng hoặc không còn thanh nguyên liệu đủ dài: Kiểm tra xem đơn hàng đã cắt xong chưa hoặc có còn thanh nguyên liệu nào phù hợp với tiêu chí cắt hay không, nếu đơn hàng đã cắt xong hoặc không có thanh đơn hàng thì chuyển đến bước 7; ngược lại chuyển đến bước 2.
6. Bước 7 – Ghi nhận phương án mới: nếu đơn hàng đã cắt xong thì tiến hành tính toán sẵn các hàm mục tiêu và ghi nhận phương án mới và kết thúc; ngược lại có nghĩa là không tìm được phương án cắt thì cũng kết thúc quá trình.

## 2.2 Cắt tiết kiệm

### *2.2.1 Tổng quan*

Ngược lại với giải thuật cắt nhanh, có thể tìm được phương án gần như ngay lập tức thì phương pháp cắt tiết kiệm lại không đơn giản. để tìm được một phương án cắt tiết kiệm tối ưu ta phải tốn rất nhiều thời gian và công sức, và theo một tỉ lệ thời gian nhất định thì kết quả của giải thuật cắt này sẽ tỉ lệ thuận với thời gian thực hiện thuật toán. Bài toán mà ta đặt ra ở đây có một không gian tìm kiếm rất rộng nên không thể tìm được phương án tốt nhất mà chỉ tìm được phương án *tối ưu - chấp nhận được* trong một khoảng thời gian cho phép. Giải thuật di truyền là một ứng cử viên sáng giá để có thể giải quyết bài toán đặt ra. sau đây là sơ đồ, một quy trình chung của giải thuật di truyền để giải quyết các vấn đề tìm kiếm.



Hình 2.3 Lược đồ giải thuật di truyền

1. Thuật toán di truyền bắt đầu bằng cách khởi tạo một tập hợp các cá thể (giải pháp) và gọi chúng là một quần thể. Điều này được thực hiện ngẫu nhiên để cung cấp phạm vi phủ sóng đồng đều cho toàn bộ không gian tìm kiếm.
2. Tiếp theo, dân số được đánh giá bằng thể lực cho mỗi cá nhân trong quần thể, ở giai đoạn này thường là sẽ chú ý đến các cá thể phù hợp nhất hiện tại và thể lực trung bình của mỗi dân số.
3. Sau khi đánh giá, thuật toán sẽ quyết định xem nó nên kết thúc tìm kiếm hay không tùy thuộc vào các điều kiện kết thúc được đặt ra. Thông thường điều này sẽ là do thuật toán đã đạt đến một số thế hệ nhất định hoặc một giải pháp thích hợp đã được tìm thấy.
4. Nếu điều kiện dừng không được đáp ứng, quần thể sẽ trải qua một giai đoạn chọn lọc trong đó các cá thể trong quần thể được chọn dựa trên điểm thể lực hay còn gọi là sức khỏe của họ - thể lực càng cao, cá thể càng có cơ hội được chọn.
5. Giai đoạn tiếp theo là áp dụng trao đổi chéo và đột biến cho các cá thể đã chọn, giai đoạn này là nơi các cá thể mới được tạo ra cho thế hệ tiếp theo.
6. Tại thời điểm này, quần thể mới quay lại được tạo ra và quay lại bước đánh giá, quá trình bắt đầu lại. Và mỗi chu kì của vòng lặp này là một thế hệ.
7. Khi điều kiện kết thúc cuối cùng được đáp ứng, thuật toán sẽ thoát ra khỏi vòng lặp và trả lại kết quả tìm kiếm được.

### *2.2.2 Khởi tạo quần thể*

Đối với bài toán trên, ta sử dụng giải thuật quay lui vét cạn để tìm kiếm các cá thể và thêm chúng vào quần thể. Khi sử dụng giải thuật quay lui, thời gian chạy giải thuật khá tốn thời gian, ta không thể chờ để thuật toán vét cạn hết mọi trường hợp mà chỉ giới hạn thời gian chạy để tìm kiếm một số phương án trong một không gian tìm kiếm rộng.

Một quần thể là một phần trừu tượng của một tập hợp các cá thể. Lớp quần thể sẽ được sử dụng để thực hiện các hoạt động cấp nhóm đối với các cá thể của nó, chẳng hạn như tìm kiếm những cá thể mạnh nhất, thu thập số liệu thống kê về toàn bộ quần thể và lựa chọn các cá thể để gây đột biến hoặc lai tạo. Sức khỏe của quần thể sẽ được tính trung bình cộng của các cá thể có trong quần thể.

Một cá thể là một phần trong quần thể, đại diện cho một giải pháp, ứng viên. Phần thông tin cốt lõi của về một cá nhân là *nhiễm sắc thể* của nó, là bản mã hóa của một giải pháp khả thi cho vấn đề đang gặp phải. Nhiễm sắc thể có thể là một chuỗi, một mảng, một danh sách… Trong bài toán này, mỗi cá thể có ba nhiễm sắc thể là ba mảng số nguyên. Một vị trí riêng lẻ trong các nhiễm sắc thể được gọi là gen, và đây là những mảnh ghép có thể được điều khiển hoặc đột biến. Chiều dài của nhiễm sắc thể sẽ bằng số lượng các thanh đơn hàng. Và các gene trong đó sẽ minh họa cụ thể cho nhiễm sắc thể đó.

1. Nhiễm sắc thể - nguyên liệu: đây là nhiễm sắc thể chính, dùng để xác định sức khỏe nổi bật của một cá thể. Mỗi gene trong nhiễm sắc thể này đại diện cho vị trí thanh đơn hàng được cắt bởi thanh nguyên liệu nào. Giá trị gene ở đây chính là vị trí của thanh nguyên liệu trong danh sách nguyên liệu được đưa vào thuật toán.
2. Nhiễm sắc thể - máy cắt: đây là nhiểm sắc thể dùng để mô tả các thanh đơn hàng được cắt bởi máy cắt nào. Giá trị gene ở đây chính là vị trí của máy cắt trong danh sách máy cắt được đưa vào thuật toán.
3. Nhiễm sắc thể - thời gian cắt: đây là nhiểm sắc thể dùng để mô tả thời gian dự kiến cắt cho mỗi thanh đơn hàng. Giá trị gene ở đây chính là thời gian cắt đơn hàng tại máy cắt được định sẵn.

Mỗi cá thể sẽ có mỗi chỉ số *sức khỏe* riêng, để xác định và phân loại *thể lực* của các cá thể trong quần thể; đây là một con số thể hiện một giải pháp tốt như thế nào cho vấn đề của cá thể này. Ý nghĩa của chỉ số *thể lực* sẽ khác nhau tùy thuộc vào vấn đề đặt ra. Và trong ngữ cảnh của bài toán này, thể lực tỉ lệ nghịch với tổng chiều dài phần thừa của các nguyên liệu được đem đi gia công. Giải pháp cho mỗi cá thể cho ra *phần thừa còn lại* càng ít thì cá thể đó càng có thể lực càng cao.

Sau đây là chi tiết giải thuật quay lui:



Hình 2.4 Khởi tạo quần thể

1. Tạo quần thể: Khởi tạo hồ chứa hay còn gọi là quần thể rỗng.
2. Phương án hiện tại: Ban đầu tạo một phương án rỗng sau đó phương án này có thể sẽ được cập nhật sau mỗi lần đệ quy.
3. Điều kiện dừng: Kiểm tra thời gian tạo ra các cá thể đã quá thời gian cho phép chưa hoặc số lượng các cá thể được tạo ra đã đạt giới hạn tối đa của một quần thể hoặc đã tìm kiếm hết tất cả các phương án. Nếu thỏa mãn thì nhảy đến *bước 12*, ngược lại chuyển đến *bước 4*.
4. Điều kiện thêm vào quần thể: Kiểm tra phương án hiện tại đã cắt đủ cho danh sách đơn hàng chưa, nếu đã đủ thì tiến hành *bước 10*; sai thì chuyển đến *bước 5*.
5. Điều kiện Chọn nguyên liệu: Có thể hiểu như điều kiện duyệt danh sách nguyên liệu. Nếu đủ điều kiện tiến hành bước 6, ngược lại thực hiện *bước 11*
6. Chọn Nguyên liệu: lấy ra thanh nguyên liệu ở vị trí hiện tại của danh sách vật liệu
7. Điều kiện cắt: Kiểm tra nếu thanh nguyên liệu có thể cắt đc cho thanh đơn hàng tương ứng thì chuyển đến *bước 8*, ngược lại quay lại *bước 5*.
8. Cắt: Tiến hành cắt nguyên liệu cho đơn hàng, sau đó chuyển đến *bước 9.*
9. Cập nhật phương án hiện tại: Thêm vào phương án hiện tại dữ liệu mới và chuyển đến *bước 2*.
10. Thêm vào quần thể: Sau khi đã tạo ra một cá thể hoàn chỉnh thì thêm phương án đó vào quần thể rồi chuyển đến *bước 11*.
11. Xóa phần tử cuối của phương án hiện tại: Sau khi đã tìm được 1 cá thể mới và thêm vào quần thể thì tiến xóa phần tử cuối cùng của cá thể đó và quay lại trạng thái trước đó (trạng thái cận hoàn chỉnh) và tiến hành tìm kiếm một phương án mới (giải thuật quay lui); tiếp tục quay lại bước 5.

### *2.2.2 Đánh giá độ thích nghi của quần thể*

Sau khi tạo ra một quần thể, ta tiến hành đánh giá lại quần thể đó, dựa vào thể lực của các cá thể trong quần thể, độ thích nghi của của quần thể chính là tổng sức khỏe (fitness) của mỗi cá thể trên tổng dân số của quần thể.

### *2.2.3 Điều kiện dừng*

Trong bài toán này, điều kiện dừng bao gồm các ý sau:

1. Thời gian chạy thuật toán: nếu tồng thời gian chạy đạt đến ngưỡng quy định thì tiến hành dừng thuật toán và trả về kết quả.
2. Giới hạn số thế hệ (generation limited): nếu số *thế hệ* hay số *đời di tuyền* trong quần thể đạt đến ngưỡng quy định thì tiến hành dừng thuật toán và trả về kết quả.
3. Tốc độ phát triển của các đời con: sau một số thế hệ n đời di truyền nhất định, nếu các cá thể đời sau không xuất hiện nhân vật xuất chúng hơn cá thể mạnh nhất của các đời trước thì tiến hành dừng thuật toán và trả về kết quả.

Các hằng số trên sẽ được quy định sẵn trước khi thực hiện thuật toán.

### *2.2.4 Chọn lọc tự nhiên*

Sau khi đã đánh giá được quần thể, các cá thể có khả năng sinh tồn tốt hơn sẽ có cơ hội được chọn lọc và sinh sản nhiều hơn các cá thể còn lại. Ta tiến hành chọn lựa chúng để bước vào giai đoạn tiếp theo. Tuy nhiên đối với bàn toán cụ thể này, sẽ áp dụng thêm một chút khác biệt. Với mong muốn sẽ tìm thấy một số thành phần may mắn trúng với câu “Vịt hóa Thiên Nga”, trong quần thể sẽ chọn lọc một lượng cá thể cá biệt có sức khỏe kém để làm phong phú thêm các loại gene trong quần thể và phục vụ cho việc đột biến sau này.

### *2.2.5 Lai tạo*

Kế tiếp là giai đoạn sinh sản. Áp dụng phép lai chéo với quần thể, được coi là “Giao phối”, chọn lọc các cá thể cha mẹ và lai tạo chúng để tạo ra các cá thể con. Các cá thể con sẽ được kế thừa các đặc tính từ cả cha và mẹ. các cá thể con có thể sẽ thích nghi tốt hơn, hoặc kém hơn. Vì thế, để bảo tồn các gene tốt trong quần thể, trong quá trình lai tạo sẽ không tiến hành lai tạo một số lượng cá thể nhất định có sức khỏe tốt nhất, thường thì số lượng này rất ít chỉ chiếm khoảng 1%. Sau đây là chi tiết sơ đồ khối minh họa quá trình lai tạo giữa hai cá thể cha mẹ.



Hình 2.5 Trao đổi chéo gene giữa hai cá thể cha mẹ

1. Chọn cá thể cha: chọn ngẫu nhiên một cá thể cha trong quần thể với điều kiện được nêu ở phần mô tả trên.
2. Chọn cá thể mẹ: Chọn ngẫu nhiên một cá thể có sức khỏe lớn hơn sức khỏe của quần thể
3. Gene của con bằng gene của cha: tiến hành sao chép tất cả các gene của cha gắn vào cá thể con. sau đó tiếp tục quá trình lai ghép.
4. Tìm chuỗi gene liên tục trong con mà khác của mẹ
5. Tìm thấy: nếu tìm thấy chuỗi gene liên tục trong con mà khác của mẹ thì tiến hành bước 6; ngược lại nếu không tìm thấy thì kết thúc lai tạo.
6. Thử thay bằng đoạn gene tương ứng của mẹ: thay đoạn gen được tìm thấy ở bước 4 vào trong cá thể con, chuỗi gene này trong cá thể mẹ nằm ở vị trí nào thì khi chuyển sang cá thể con cũng phải chọn đúng vị trí đó để thay.
7. Cá thể con lai tạo tốt hơn: kiểm tra sức khỏe của cá thể con nếu tốt hơn cá thể của cha mẹ thì đi đến bước 8; ngược lại tiếp tục quay lại bước 4.
8. Ghi nhận phép lai: khi tạo ra được cá thể con có sức khỏe tốt, tiến hành thêm chúng vào quần thể.

### *2.2.6 Đột biến*

Dễ nhận thấy rằng, nếu chỉ bằng việc sinh ngẫu nhiên và lai tạo sẽ rất khó để tìm được nghiệm. Trừ khi cá thể khởi tạo phù hợp luôn với yêu cầu đề, tức là có đáp án ngay từ đầu – không gian tìm kiếm nhỏ. Đột biến chính là nguyên liệu của *chọn lọc tự nhiên*, bằng việc lựa chọn ngẫu nhiên các vị trí và thay thế bằng một kí tự ngẫu nhiên nào đó – *đột biến điểm*.

Ứng dụng đột biến vào quần thể. Đột biến ảnh hưởng đến các cá thể hơn là quần thể. Ta xem xét từng cá nhân trong quần thể, và nếu họ đủ may mắn (hoặc không may mắn) ta sẽ áp dụng một số biến đổi ngẫu nhiên vào nhiễm sắc thể của chúng. giống như ở mục *lai tạo,* loại đột biến được áp dụng phụ thuộc vào vấn để cụ thể mà ta đang giải quyết. Trong trường hợp này, chúng ta phải tìm kiếm các vị trí gene chưa tốt và thay thế chúng bởi các gene khác để cường hóa sức mạnh của cá thể đó. Sau đây là các bước của phương thức đột biến trên một cá thể được mô phỏng bằng sơ đồ khối bên dưới.



Hình 2.6 Đột biến cá thể

1. Chọn cá thể đột biến: như đã mô tả ở phần đầu, ta sẽ có hai nhóm lựa chọn chính đó là những cá thể tốt, và những cá thể kém (tồi tệ nhất) để tiến hành đột biến.
2. Tính lượng nguyên liệu còn lại: với mỗi cá thể đại diện cho một giải pháp cắt sắt. ta tiến hình tính toán phần thừa nguyên liệu còn lại khi đi theo giải pháp này. Có thể là tận dụng các thanh đã bị cắt trong phương pháp hoặc những thanh chưa được cắt nhưng có ưu thế hơn về phần thừa nếu được lựa chọn.
3. Vị trí gene = 1: Khởi tạo giá trị biến lặp nhiễm sắc thể, đánh dấu vị trí(index) trong nhiểm sắc thể
4. Vị trí thanh nguyên liệu = 1: Khởi tạo giá trị biến lặp nguyên liệu, đánh dấu vị trí(index) trong danh sách nguyên liệu còn lại.
5. Tạo phương án mới bằng cách thay thế nguyên liệu vào vị trí gene hiện tại: dùng thanh nguyên liệu hiện tại để cắt thanh đơn hàng tại vị trí đơn hàng chính là vị trí gene. Có thể ta sẽ được phương án hợp lệ mới hoặc không.
6. Đã đến vị trí thanh nguyên liệu cuối: nếu sai chuyển đến *bước 7*; ngược lại chuyển đến *bước 8*.
7. Tăng vị trí thanh nguyên liệu: bước nhảy là một đơn vị cho mỗi lần tăng, để duyệt hết tất cả các thanh nguyên liệu còn lại. sau đó lại nối tiếp đến *bước 5*.
8. Đã đến vị trí gene cuối: nếu sai chuyển đến bước 9; ngược lại chuyển đến *bước 10*.
9. Tăng vị trí gene: bước nhảy là một đơn vị cho mỗi lần tăng, để duyệt hết tất cả các gene trong nhiểm sắc thể. sau đó lại nối tiếp đến *bước 4*.
10. Chọn phương án thay thế tốt nhất: sau khi duyệt hết nhiễm sắc thể cũng như danh sách nguyên liệu còn lại ta có được một danh sách các phương án thay thế. Ta sẽ chọn một phương án tốt nhất trong đó (mỗi phương án là một cá thể mới, chọn cá thể có sức khỏe tốt nhất) để đặt làm kết quả đột biến – cá thể mới.

# CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ

## 3.1 Cài đặt các thuật toán đã đề xuất

### 3.1.1 Thuật toán cắt nhanh

Thuật toán cắt nhanh được cài đặt như sau:

* Khởi tạo mảng danh sách các phương án, mỗi phương án được lưu trong đối tượng FastCutBean bao gồm các thông tin
  + numberMaterial: số lượng các thanh sắt cần dùng
  + rateRemain: tỉ lệ dư thừa
  + arrIndexStockUsed: lưu thông tin chỉ số của các thanh nguyên liệu cần dùng, mảng có kích thược bằng số lượng thanh sắt trong đơn hàng.

List<FastCutBean> listPlan = new ArrayList<> ();

* Khởi tạo danh sách thanh đơn hàng ban đầu với chuỗi *orders* được truyền vào thuật toán.

int [] arrOrderInit = Arrays.*stream*(orders.split(*COMMA*))  
 .map(s -> Integer.*parseInt*(s.trim()))  
 .mapToInt(Integer::intValue)  
 .toArray();

* Sắp xếp mảng các thanh đơn hàng theo thứ tự chiều dài giảm dần.

int [] arrOrder = IntStream.*of*(arrOrderInit)

.boxed().sorted(Comparator.*reverseOrder*())  
 .mapToInt(Integer::intValue)  
 .toArray();

* Từ mảng đơn hàng ban đầu và mảng đơn hàng đã được sắp xếp ở trên, tiến hành tạo một mảng lưu lại chỉ số (index) ban đầu của các thanh đơn hàng trước khi được sắp xếp.

int [] sortedIndicesOrder = IntStream.*range*(0, arrOrderInit.length)

.boxed().sorted(Comparator.*comparing*(i -> -arrOrderInit[i]))  
 .mapToInt(Integer::intValue)  
 .toArray();

* Khởi tạo danh sách thanh nguyên liệu ban đầu với chuỗi *stocks* được truyền vào thuật toán.

int [] arrStockInit = Arrays.*stream*(stocks.split(*COMMA*))  
 .map(s -> Integer.*parseInt*(s.trim()))  
 .mapToInt(Integer::intValue)  
 .toArray();

* Sắp xếp mảng các thanh nguyên liệu theo thứ tự chiều dài giảm dần.

int [] arrStock = IntStream.*of*(arrStockInit)  
 .boxed().sorted(Comparator.*reverseOrder*())  
 .mapToInt(Integer::intValue)  
 .toArray();

* Từ mảng nguyên liệu ban đầu và mảng nguyên liệu đã được sắp xếp ở trên, tiến hành tạo một mảng lưu lại chỉ số (index) ban đầu của các thanh nguyên liệu trước khi được sắp xếp.

int [] sortedIndicesStock = IntStream.*range*(0, arrStockInit.length)  
 .boxed().sorted(Comparator.*comparing*(i -> -arrStockInit[i]))  
 .mapToInt(Integer::intValue)  
 .toArray();

* Khởi tạo mảng kiểm tra điều kiện được đưa vào phương pháp *tìm phương án cắt* của các thanh nguyên liệu, với kích thước bằng kích thước của mảng nguyên liệu.

int [] arrCheckMaterialCanBeCut = new int [arrStock.length];

* Khởi tạo giá trị số lượng các thanh sắt bị loại bỏ dần trong quá trình tìm kiếm phương án.

int numberMaterialRemoved = 0;

* Tìm phương án cắt bằng các tham số được chuẩn bị ở trên, các phương án tìm được sẽ được thêm vào danh sách listPlan.

*fastCutMain*(listPlan, arrCheckMaterialCanBeCut, numberMaterialRemoved, arrStock, arrOrder);

* Sau khi tìm được một danh sách các phương án, tiến hành tìm kiếm phương án tốt nhất nếu có.

if (!listPlan.isEmpty()) {  
 listPlan = listPlan.stream()  
 .sorted(Comparator.*comparing*(FastCutBean::getNumberMaterial)  
 .thenComparing(FastCutBean::getRateRemain))  
 .collect(Collectors.*toList*());  
 return listPlan.get(0);  
 } else {  
 return *BLANK*;  
}

Tiếp theo là chi tiết phương pháp tìm phương án cắt – **fastCutMain:**

* Khởi tạo mảng lưu chiều dài các thanh sắt còn thừa sau khi cắt

int [] arrRemain = new int [arrStock.length];

* Tìm kiếm giá trị nhỏ nhất trong danh sách các thanh đơn hàng

OptionalInt minOder = Arrays.*stream*(arrOrder).min();

* Khởi tạo mảng lưu cách cắt đơn hàng, mảng có kích thước bằng số lượng các thanh đơn hàng. Các phần tử có giá trị mặc định là -1.

int [] arrIndexStockUsed = new int [arrOrder.length];

* Tiến hành tính toán, lưu cách cắt nguyên liệu và các thanh nguyên liệu còn lại.

for (int i = 0; i < arrStock.length; i++) {  
 if (arrCheckMaterialCanBeCut[i] == 1) {  
 int remaining = arrStock[i];  
 for (int j = 0; j < arrOrder.length; j++) {  
 if (remaining < minOder.getAsInt()) break;  
 if (remaining >= arrOrder[j] && arrIndexStockUsed[j] == -1) {  
 arrIndexStockUsed[j] = i;  
 remaining -= (arrOrder[j] + unit);  
 if (remaining < 0) {  
 remaining = 0;  
 }  
 }  
 }  
 arrRemain[i] = remaining;  
 }  
}

* Dừng tìm kiếm các phương án nếu không thể cắt được nữa trên danh sách nguyên liệu được đưa vào.

if (Arrays.*stream*(arrIndexStockUsed).anyMatch(item -> item == -1)) {  
 return;  
}

* Tính toán số lượng thanh sắt nguyên liệu cần dùng và tỉ lệ dư thừa sau đó thêm vào danh sách các phương án.

int numberMaterial = Arrays.*stream*(arrIndexStockUsed).distinct().count();  
  
IntStream otherRemainStock = IntStream.*range*(0, arrRemain.length).filter(i -> arrRemain[i] != arrStock[i]);  
  
int remain = otherRemainStock.mapToObj(i -> arrRemain[i])

.mapToInt(Integer::intValue)

.sum();  
int stockUsed = otherRemainStock.mapToObj(i -> arrStock[i])

.mapToInt(Integer::intValue)

.sum();  
  
double rateRemain = (double)remain / stockUsed;  
  
listArn.add(new FastCutBean(numberMaterial, rateRemain, arrIndexStockUsed));

* Sau đó xóa dần các thanh sắt đầu tiên trong danh sách nguyên liệu và tiếp tục tìm kiếm các phương án khác.

arrCheckMaterialCanBeCut[numberMaterialRemoved++] = 0;

*fastCutMain*(listArn, arrCheckMaterialCanBeCut, numberMaterialRemoved, arrStock, arrOrder);

### 3.1.2 Thuật toán di truyền

Do độ phức tạp của thuật toán di truyền khá lớn, vậy nên trong báo cáo này chỉ trình bày cách cài đặt của các các hàm chính trong thuật toán, các chi tiết nhỏ sẽ bị bỏ qua.

#### 3.1.2.1 Khởi tạo quần thể

Phương thức khởi tạo quần thể được chia làm ba phần, phần đầu là kiểm tra điều kiện dừng của quá trình khởi tạo, phần tiếp theo là thêm cá thể vào quần thể và phần cuối cùng là tạo một cá thể. Các phần này cứ lặp đi lặp lại cho đến khi đáp ứng điều kiện dừng của phương thức.

Phương thức có ba tham số đầu vào:

* currentChromosome: trạng thái nhiễm sắc thể hiện tại.
* stockState: trạng thái các thanh nguyên liệu hiện tại.
* populationSize: giới hạn kích thước của quần thể.

void generatePopulation(List<Integer> currentChromosome, List<Integer> stockState, int populationSize) {  
 try {  
 // Population limit reached!  
 termGenerationPopulation();

// Get a complete individual

addIndividualToPopulation();

// Next Node  
 generateIndividual();  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
}

1. **termGenerationPopulation**: điều kiện dừng của phương thức bao gồm: thời gian thực hiện phương thức, số lượng cá thể được sinh ra. Nếu thời gian thực hiện vượt quá thời gian cho phép hoặc số cá thể đã đạt tới ngưỡng chịu đựng của quần thể thì chấm dứt quá trình khởi tạo quần thể.

if (Boolean.*TRUE*.equals(isFinishedGen)) {  
 return;  
}  
long duration = Duration.*between*(startGen, Instant.*now*()).getSeconds();  
if (duration > generationLimit || realPopulationSize >= populationSize) {  
 isFinishedGen = true;  
 return;  
}

Trong đó:

* isFinishedGen: kiểm tra điều kiện dừng của quá trình khởi tạo quần thể
* startGen: thời điểm bắt đầu quá trình khởi tạo quần thể
* generationLimit: giới hạn thời gian thực hiện quá trình khởi tạo quần thể
* realPopulationSize: kích thước quần thể trong quá trình khởi tạo thực tế

1. **addIndividualToPopulation**: nếu nhiễm sắc thể hiện tại đã đáp ứng được yêu cầu của bài toán (có kích thược bằng kích thước của mảng thanh đơn hàng) thì tạo mới một cá thể dựa trên nhiễm sắc thể đó, sau đó tính toán sức khỏe cho cá thể vừa tạo rồi tiến hành thêm vào quần thể. Đồng thời tăng kích thước thực tại của quần thể và kiểm tra để cập nhật biến ***isFinishedGen.***

if (currentChromosome.size() == orders.size()) {  
 Individual individual = new Individual(new ArrayList<>(currentChromosome));  
 double rate = getFitnessOfChromosome(currentChromosome);  
 individual.setFitness(rate);  
 this.population.add(individual);  
 this.realPopulationSize++;  
 if(this.realPopulationSize >= populationSize) {  
 isFinishedGen = true;   
 }  
 return;  
}

1. **generateIndividual:**  Sử dụng ý tưởng của thuật toán quay lui để thực hiện sinh các gene cho nhiễm sắc thể. Thủ tục này thử cho genei nhận lần lượt các giá trị mà nó có thể nhận (các giá trị ở đây là vị trí của các thanh nguyên liệu trong danh sách). Thử cho genei nhận giá trị S và nếu genei là phần tử cuối cùng trong nhiễm sắc thể thì tiến hành thực hiện bước 2 ở trên; Ngược lại ghi nhận việc cho genei nhận giá trị S (nếu cần). Gọi tiếp đệ quy để chọn genei + 1. Sau mỗi bước lại tiến hành bỏ ghi nhận việc thử genei := S, để thử giá trị khác.

for (int stockIndex = 0; stockIndex < stockState.size(); stockIndex++) {  
 // Population limit reached!

termGenerationPopulation();

int idx = currentChromosome.size();  
 if (isCanCut) {  
 // update stocks  
 stockState.set(stockIndex, stockState.get(stockIndex) - orders.get(idx));  
 currentChromosome.add(stockIndex);  
 generatePopulation(currentChromosome, stockState, populationSize);  
 currentChromosome.remove(currentChromosome.size() - 1);

stockState.set(stockIndex, stockState.get(stockIndex) + orders.get(idx));  
  
 // Population limit reached!

termGenerationPopulation();

}

}

#### 3.1.2.2 Lai tạo

Quá trình lai tạo được viết dựa trên tư tưởng trao đổi chéo các gene giữa các nhiểm sắc thể cha mẹ để tạo ra cá thể con có sức khỏe tốt hơn hai cá thể cha và mẹ. Ban đầu duyệt qua danh sách các cá thể trong quần thể ngoại trừ các cá thể có sức khỏe tốt cần được bảo lưu lại trong quần thể (số lượng cá thể tốt được bảo lưu được quy định bởi hằng số elitismCount). Cá thể cha (parent1) chính là cá thể hiện tại trong danh sách vòng lặp, mỗi cá thể cha cần chọn cho mình một cá thể mẹ (parent2) để tiến hành việc trao đổi chéo tạo cá thể con (offspring). Trong cài đặt thuật toán, tiến hành sao chép các gene của cá thể cha sang cá thể con để thuận tiện trong việc triển khai mã nguồn.

public Population crossoverPopulation(Population population) {  
   
 // Loop over current population by fitness  
 for (int index = this.elitismCount; index < size; index++) {  
 Individual parent1 = getFittest(index);

// Find second parent  
 Individual parent2 = selectParent(population);

// initialize offspring

Individual offspring = new Individual(parent1);

// Apply crossover to this individual

crossoverInPairs(parent1, parent2);

}  
 return population;  
}

Sau khi chọn được hai cá thể cha mẹ, tiến hành lai tạo theo phương pháp *trao đổi chéo*. Duyệt qua danh sách các gene của cá thể con và cá thể mẹ, nếu tìm thấy một hoặc một đoạn gene khác nhau và đồng thời đoạn gen đó của cá thể mẹ có thể thay thế cho đoạn gene của cá thể con thì tiến hành thay thế vào cá thể con.

for (int geneIndex in parent1.getChromosome){  
 // Use half of parent1's genes and half of parent2's genes  
 int gene1 = offspring.getGene(geneIndex);  
 int gene2 = parent2.getGene(geneIndex);  
  
 // The father's genes are different from the mother's, and the mother's genes are qualified for replacement  
 if(isEligible){  
 offspring.setGene(geneIndex, gene2);  
 }  
}

Tính toán sức khỏe của cá thể con được tạo ra, nếu tốt hơn cá thể cha thì thêm cá thể con vào quần thể

double currentFitness = getFitnessOfChromosome(offspring);  
  
if (currentFitness < offspring.getFitness()) {  
 offspring.setFitness(currentFitness);

// Add offspring to new population  
 population.setIndividual(populationIndex, offspring);  
}

#### 3.1.2.3 Đột biến

Ban đầu, khởi tạo biến bestFitnessOfAll dùng để lưu trữ sức khỏe của cá thể đạt được sau đột biến; finalGeneIndexFrom dùng để lưu trữ vị trí gene được đột biến trong nhiễm sắc thể; finalStockIndexMove dùng để lưu trữ gene tại vị trí finalGeneIndexFrom sau đột biến trong nhiễm sắc thể; tính toán lượng nguyên liệu còn lại (stockTemp) khi đã được chuẩn bị cho cá thể hiện tại (cá thể đem đi đột biến).

List<Integer> stockTemp = getStockRemain(chromosome);  
double bestGapOfAll = 1;  
int finalMoveTo = -1;  
int finalFromPosition = -1;

Tiếp theo, duyệt qua danh sách gene trong nhiễm sắc thể để tìm kiếm các giá trị bestFitnessOfAll, finalStockIndexMove, finalGeneIndexFrom đã khởi tạo ở trên. Ở mỗi vị trí gene cần phải khởi tạo các giá trị bestFitness, stockIndexMove, geneIndexFrom tương ứng để lưu trữ tạm thời cho các giá trị trên. Sau đó duyệt qua danh sách mảng nguyên liệu để tìm thanh nguyên liệu phù hợp và thay thế cho gene hiện tại của nhiễm sắc thể.

for (int i = 0; i < orders.size(); ++i) {  
 double bestGap = 1, moveTo = -1, fromPosition = -1;  
 for (int j = 0; j < stocks.size(); ++j) {  
 //Check whether current stock location is a potential candidate   
 if (isEligible) {  
 bestGap = rateRemain; moveTo = j; fromPosition = i;  
 }  
 }  
  
 if (bestGapOfAll > bestGap) {  
 bestGapOfAll = bestGap;  
 finalMoveTo = moveTo;  
 finalFromPosition = fromPosition;  
 }  
}

Tiếp theo, tiến hành tạo cá thể đột biến nếu quá trình đột biến thành công; ngược lại thì dừng quá trình và chuyển giai đoạn khác.

chromosome.set(finalFromPosition, finalMoveTo);  
  
// mutant individual  
Individual mutant = new Individual(chromosome);  
mutant.setFitness(getFitnessOfChromosome(chromosome));  
  
if (mutant.getFitness() < worstValue && mutant.getFitness() < currentFitness) {  
 newPopulation.getIndividuals().set(worstPosition, mutant);  
}

## 3.2 Cài đặt thuật toán quy hoạch tuyến tính

Để có thể so sánh và đánh giá kết quả so với thuật toán tham lam và thuật toán di truyền, chúng tôi sử dụng thuật toán quy hoạch tuyến tính bằng phương pháp đơn hình được nghiên cứu trong bài báo [4]. Phương pháp này đòi hỏi đầu vào đồng nhất, nghĩa là các thanh nguyên liệu đầu vào phải có cùng chiều dài.

Giả sử cần cắt thanh có chiều dài thành đoạn, mỗi đoạn có chiều dài  tương ứng. Các phương án cắt khác nhau đều nhằm xác định được số lượng các đoạn xi sao cho  lớn nhất nghĩa lànhỏ nhất. Như vậy, mối quan hệ số lượng các thanh được cắt ra từ vật liệu cho trước là quan hệ tuyến tính, khi đó sử dụng bài toán quy hoạch tuyến tính tổng quát như sau: Tìm max, min của (1) với các ràng buộc: 

. Trong đó:

* z: hàm mục tiêu
* c: véc tơ hệ số hàm mục tiêu, 
* a: ma trận hệ số các điều kiện ràng buộc



* b: véc tơ cột hệ số vế phải 

Để giải bài toán quy hoạch tuyến tính dạnh tổng quát (1) trước hết phải đưa bài toán về dạng chính tắc: với ràng buộc 

.

Mỗi ràng buộc đẳng thức “=” có thể viết thành hai ràng buộc bất đẳng thức:







Như vậy, mỗi ràng buộc ban đầu  được thay thế bởi hai ràng buộc:

 và  làm cơ sở để giải bài toán sau này.

Có nhiều phương pháp giải tối ưu bài toán, ví dụ lập đồ thị, lập bảng tính, tính toán qua phần mềm Mathematica. Tuy nhiên, khóa luận này sẽ đề cập đến thuật toán quy hoạch tuyến tính bằng phương pháp đơn hình, thuật toán này có thể cài đặt dễ dàng trong excel.

## 3.3 So sánh kết quả chạy các thuật toán

So sánh hai tiêu chí của của các thuật toán.

1. Số lượng thanh đưa vào máy gia công là ít nhất (trực tiếp liên quan đến thời gian)
2. Tỉ lệ dư thừa = Tổng chiều dài các thanh dư thừa / Tổng chiều dài các thanh nguyên liệu đưa vào máy gia công

Hai thí nghiệm đầu tiên, khóa luận thử nghiệm với bộ dữ liệu chỉ bao gồm các thanh nguyên liệu mới, đồng nhất không sử dụng thêm các thanh dư thừa. Hai thử nghiệm tiếp theo khóa luận thử nghiệm với bộ dữ liệu bao gồm các thanh nguyên liệu và các thanh dư thừa. Khi đầu vào bao gồm cả các thanh dư thừa, không đồng nhất thì thuật toán quy hoạch tuyến tính không hoạt động được vì vậy trong bảng báo cáo không có kết quả.

### 3.3.1 Trường hợp thử nghiệm thứ nhất

Dữ liệu đầu vào bao gồm:

1. Lô hàng một cần cắt 2000 thanh 2m.
2. Lô hàng hai cần cắt 1000 thanh 3m
3. Trong kho có 10000 thanh nguyên liệu có chiều dài là 11,7m
4. Có ba máy cắt, độ dày lưỡi cắt thép là 5mm.

Kết quả của các thuật toán như sau:

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Số lượng thanh sử dụng** | **Tỉ lệ dư thừa** |
| Quy hoạch tuyến tính | 700 | 15% |
| Cắt nhanh | 700 | 15% |
| Cắt tiết kiệm | 700 | 15% |

### 3.3.2 Trường hợp thử nghiệm thứ hai

Dữ liệu đầu vào bao gồm:

1. Lô hàng một cần cắt 1000 thanh 3m, 2000 thanh 2m, 3000 thanh 5m, 4000 thanh 7m
2. Trong kho có 100 thanh nguyên liệu có chiều dài là 11,7m
3. Có ba máy cắt, độ dày lưỡi cắt thép là 5mm.

Kết quả của các thuật toán như sau:

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Số lượng thanh sử dụng** | **Tỉ lệ dư thừa** |
| Quy hoạch tuyến tính | 5500 | 22% |
| Cắt nhanh | 5500 | 22% |
| Cắt tiết kiệm | 5500 | 22% |

### 3.3.3 Trường hợp thử nghiệm thứ ba (xem lại)

Dữ liệu đầu vào bao gồm:

* Lô hàng một cần cắt 300 thanh 1,250m, 200 thanh 1,2m, 400 thanh 1m.
* Trong kho có 600 thanh nguyên liệu có chiều dài là 11,7m; 100 thanh 5,623m; 100 thanh 1,009m; 200 thanh 1,64m.

Kết quả của các thuật toán như sau:

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Số lượng thanh sử dụng** | **Tỉ lệ dư thừa** |
| Quy hoạch tuyến tính | \_ | \_ |
| Cắt nhanh | 92 | 5.3% |
| Cắt tiết kiệm | 92 | 5.3% |

### 3.3.4 Trường hợp thử nghiệm thứ bốn

Dữ liệu đầu vào bao gồm:

1. Lô hàng một cần cắt 1000 thanh 3m, 2000 thanh 2m, 3000 thanh 5m.
2. Lô hàng hai cần cắt 4000 thanh 5m, 5000 thanh 7m, 2000 thanh 3m.
3. Trong kho có 20000 thanh nguyên liệu có chiều dài là 11,7m; 1500 thanh; 3000 thanh 7,995m.
4. Có ba máy cắt có các tham số sau: mất một phút để cắt một nhát cắt, thời gian rảnh từ 13h ngày 2021/02/20; độ dày lưỡi cắt thép là 0mm.

Kết quả của các thuật toán như sau:

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Số lượng thanh sử dụng** | **Tỉ lệ dư thừa** |
| Quy hoạch tuyến tính | \_ | \_ |
| Cắt nhanh | 8000 | 17% |
| Cắt tiết kiệm | 9200 | 13% |

### 3.3.5 Nhận xét

Từ các trường hợp kiểm thử ở trên, nhận thấy rằng có hai trường hợp xảy ra:

* Trường hợp chỉ sử dụng các thanh nguyên liệu mới: cả ba thuật toán tham lam, thuật toán quy hoạch tuyến tính và thuật toán di truyền cho cùng kết quả tối ưu. Tuy nhiên Thuật toán di truyền sẽ tốn kém thời gian hơn.
* Trường hợp sử dụng cả các thanh nguyên liệu mới và các thanh nguyên liệu thừa từ các lần gia công trước: trường hợp này không thể áp dụng giải thuật quy hoạch tuyến tính. giải thuật di truyền đã chiếm ưu thế hơn so với giải thuật tham lam đối với tiêu chí tiết kiệm.

Từ những nhận xét trên, chúng tôi đưa ra khuyến nghị, nếu cần tận dụng các thanh dư thừa thì nên áp dụng giải thuật cắt tiết kiệm; ngược lại chỉ sử dụng các thanh nguyên liệu mới thì áp dụng giải thuật cắt nhanh.

KẾT LUẬN(tóm tắt khóa luận)

# TÀI LIỆU THAM KHẢO

[2] Leon Kos and Joze Duhovnik, “ROD CUTTING OPTIMIZATION WITH STORE UTILIZATION”, INTERNATIONAL DESIGN CONFERENCE - DESIGN 2000 Dubrovnik, May 23-26, 2000.