NGHIÊN CỨU BÀI TOÁN PHÂN CÔNG LỊCH TRÌNH TÁC VỤ, CÔNG NHÂN VÀ TRẠM TRÊN DÂY CHUYỀN LẮP RÁP ĐA CÔNG NHÂN

TASK-WORKER-STATION ASSIGNMENT AND SCHEDULING ON MULTI-MANNED ASSEMBLY LINES

Trần Phương Nhã Uyên, PGS,TS. **Phan Nguyễn Kỳ Phúc**
Khoa Kỹ thuật và Quản lý Công nghiệp, Trường Đại học Quốc tế,
Đại học Quốc gia Thành phố Hồ Chí Minh

TÓM TẮT

Bài báo tập trung nghiên cứu giải quyết bài toán lập lịch dây chuyền lắp ráp đa công nhân với mục tiêu cải thiện hiệu suất của dây chuyền. Bài toán được tiếp cận bằng mô hình quy hoạch tuyến tính nguyên hỗn hợp (MILP) nhằm thể hiện các quan hệ thứ tự ưu tiên, khả năng của các trạm, và tính khả thi giữa công nhân và tác vụ. Do MILP trở nên tốn kém về mặt tính toán đối với các trường hợp có quy mô vừa và lớn nên thuật toán di truyền (GA) được đề xuất để có được các giải pháp gần tối ưu với thời gian chạy ngắn hơn đáng kể. Xác thực trên các tập dữ liệu chuẩn nhỏ cho thấy cả MILP và GA đều tạo ra các lịch trình khả thi đáp ứng tất cả các ràng buộc, chứng minh tính chính xác của các công thức và toán tử. Các thí nghiệm tính toán tiếp theo chỉ ra rằng trong khi MILP đạt được tính tối ưu cho các trường hợp nhỏ, GA cung cấp các giải pháp chất lượng cao và hiệu suất có thể mở rộng cho các trường hợp thử nghiệm lớn hơn. Kết quả nhấn mạnh tính hiệu quả của việc kết hợp tối ưu hóa chính xác và heuristic để giải quyết các vấn đề lập lịch trình phức tạp cho dây chuyền lắp ráp nhiều người nhằm giảm thiểu thời gian hoàn thành.

Từ khóa: *Lập lịch dây chuyền lắp ráp nhiều người; Phân công tác vụ-công nhân-trạm; Quy hoạch tuyến tính nguyên hỗn hợp (MILP); Thuật toán di truyền (GA).*

ABSTRACT

The paper investigates the Multi-Manned Assembly Line Scheduling Problem (MALSP), where multiple workers operate concurrently at each station with the goal of improving the productivity of the line. The problem is approached by a mixed integer linear programming (MILP) model to capture the relations of precedence, station capabilities, and feasibility between workers and tasks. Since MILP becomes computationally expensive for medium and large-scale instances, Genetic Algorithm (GA) is proposed as a heuristic approach to obtain near optimal solutions with significantly shorter running time. Validation on small benchmark datasets shows that both the MILP and GA produce feasible schedules that satisfy all constraints, demonstrating the correctness of the formulations and operators. Further computational experiments indicate that while the MILP achieves optimality for small instances, the GA provides high-quality solutions and scalable performance for larger tests. The results underscore the effectiveness of combining exact optimization and metaheuristics to solve complex multi-manned assembly line scheduling problems aimed at minimizing makespan.

Keywords: *Multi-person assembly line scheduling; Task-worker-station assignment; Mixed Integer Linear Programming (MILP); Genetic Algorithm (GA).*

☞

1. TỔNG QUAN

Các dây chuyền lắp ráp là một quy trình sản xuất quan trọng, trong đó máy móc và/hoặc công nhân lắp ráp các thành phần của sản phẩm thông qua một chuỗi các trạm làm việc. Trong số vô vàn cấu hình của dây chuyền lắp ráp, dây chuyền lắp ráp đa công nhân – Multi-manned Assembly Lines (MAL) là một cấu trúc đáng chú ý, trong đó một trạm làm việc có thể cho phép nhiều hơn một công nhân thực hiện các tác vụ trên cùng một sản phẩm. Ứng dụng của mô hình dây chuyền lắp ráp đa công nhân đóng vai trò quan trọng trong việc sản xuất hàng loạt với quy mô lớn ví dụ như ô tô, điện tử, đồ nội thất hoặc sản xuất thép. Với loại dây chuyền lắp ráp này, các trạm có thể được gán nhiều hơn một công nhân để xử lý, tối ưu hóa nguồn tài nguyên nhân lực và năng lực sản xuất. Một MAL hiệu quả, thông qua việc cân bằng tác vụ giữa nhiều trạm làm việc và tối ưu hóa độ dài dây chuyền, có thể cải thiện năng suất, hiệu quả tài nguyên và giảm tắc nghẽn (bottleneck) [1].

Mặc dù có nhiều lợi thế, MAL cũng phải đối mặt với một số thách thức như phối hợp tác vụ và khả năng can thiệp lẫn nhau giữa các công nhân. Điều này đòi hỏi phải có sự lập lịch và cân bằng cẩn thận để đảm bảo công nhân thực hiện tác vụ hiệu quả mà không gây ra xung đột hoặc chậm trễ. Trong các tình huống thực tế, việc gán tác vụ-công nhân-trạm làm việc (task-worker-station assignment) là một thách thức đáng kể. Các cách tiếp cận phổ biến thường xử lý vấn đề này một cách riêng biệt hoặc bỏ qua một số ràng buộc, ví dụ giả định chỉ có một công nhân tại một trạm hoặc chỉ có công nhân có cùng kỹ năng tại các trạm đa người vận hành. Do đó, động lực thực tiễn của nghiên cứu này nằm ở việc tập trung vào tối ưu hóa việc lập lịch trình cho công nhân thực hiện các tác vụ tại các trạm để tăng cường sử dụng tài nguyên và nâng cao hiệu quả tổng thể của dây chuyền lắp ráp.

Từ góc độ học thuật, việc nghiên cứu các phát triển này mở ra các cơ hội nghiên cứu mới gồm việc khám phá các mô hình toán học nâng cao, các thuật toán heuristic và meta-heuristic, hoặc hyper-heuristics để quản lý và lựa chọn các thuật toán phù hợp nhằm giải quyết các vấn đề dây chuyền lắp ráp. Hoặc phát triển các giải pháp lai bằng cách kết hợp các kỹ thuật từ lý thuyết tối ưu hóa và lập lịch trình nhằm xử lý các trường hợp phức tạp và đa mục tiêu. Điều này giúp các ứng dụng có thể điều chỉnh nhanh chóng để giải quyết các thay đổi trong thực tế (ví dụ: thay đổi nhu cầu, điều kiện sản xuất) hoặc đồng thời giải quyết việc phân phối khối lượng công việc và lập lịch trình.

2. TỔNG QUAN CÁC MÔ HÌNH TỐI ƯU TRONG VIỆC LẬP LỊCH TRÌNH

MALBP là một sự chuyển đổi mô hình từ bài toán cân bằng dây chuyền lắp ráp truyền thống (Assembly Line Balancing Problem - ALBP), thuộc danh mục cân bằng dây chuyền lắp ráp tổng quát (General ALBP) liên quan đến lực lượng lao động. Việc cho phép nhiều công nhân thực hiện tác vụ tại một trạm duy nhất có thể giúp cân bằng dây chuyền hiệu quả hơn và tăng cường sử dụng tài nguyên so với ràng buộc "một công nhân mỗi trạm" của ALBP truyền thống. Trong ALBP truyền thống, thứ tự các tác vụ do một công nhân thực hiện thường không quan trọng miễn là ràng buộc trình tự được tuân theo. Tuy nhiên, trong MALBP, việc phối hợp hành động của các công nhân khác nhau tại cùng một trạm yêu cầu thứ tự các tác vụ phải được xem xét cẩn thận để tránh xung đột tác vụ [2]. MALBP còn xem xét các yếu tố như khả năng tương thích với vị trí lắp đặt (đảm bảo đủ không gian làm việc cho công nhân) và việc chia sẻ thiết bị/công cụ giữa các công nhân [3].

Qua thời gian, MALBP đã được mở rộng để tích hợp nhiều yếu tố thực tế hơn. MALBP

với công nhân lành nghề (MALBP-SW) nhằm giải quyết việc tối ưu hóa chi phí hoạt động (chi phí trạm làm việc cộng với lương công nhân) thông qua mô hình lập trình tuyến tính số nguyên hỗn hợp (MILP) [3]. MALBP với công nhân di chuyển qua các trạm (MMALBP-WW) cho phép công nhân "kích hoạt" các trạm làm việc lân cận. Các phương pháp giải quyết biến thể này bao gồm mô hình MILP dựa trên trình tự (SBM) và sử dụng quy trình quyết định Markov (MDP) để xử lý tính chất động và ngẫu nhiên [5]. Mặt khác, dựa trên đặc điểm của người lao động, bài toán phân công và cân bằng chuyền (ALWABP) đã mở rộng bằng cách chia hai nhóm công nhân khác nhau gồm đồng nhất và dị biệt (tức là công nhân có kỹ năng và khả năng tương tự, và công nhân có kỹ năng, khả năng và có thể là chi phí khác nhau), sau đó sử dụng MILP và Heuristic Chèn dựng (CIH) để tìm giải pháp [6]. Và sự tổng quát hóa của bài toán này là Bài toán tích hợp và cân bằng công nhân trên dây chuyền lắp ráp với nhiều người lao động (MALWIBP), được khơi nguồn từ các chiến lược luân chuyển công việc thân thiện với người khuyết tật và mục tiêu là giảm thiểu số lượng công nhân (chủ yếu là công nhân đồng nhất) và thứ hai là giảm thiểu số lượng trạm làm việc, được giải quyết bằng phương pháp phân rã theo cấp bậc (HDH) [1].

Có nhiều phương pháp tiếp cận bài toán MALBP, nhìn chung có thể được phân loại thành các phương pháp chính xác, thuật toán heuristic, thuật toán metaheuristic và phương pháp lai. Các phương pháp chính xác được thiết kế để tìm ra giải pháp tối ưu cho một bài toán bằng cách khám phá một cách có hệ thống toàn bộ không gian nghiệm. Ba phương pháp chính xác có thể được liệt kê là lập trình toán học, nhánh và cận (B&B) và lập trình ràng buộc (CP). Các mô hình lập trình toán học, bao gồm lập trình tuyến tính số nguyên (ILP) và lập trình tuyến tính số nguyên hỗn hợp (MILP) đã được xây dựng để giải quyết mục tiêu MALBP là giảm thiểu tổng số công nhân và số trạm [7]. Cách tiếp cận này có thể đảm bảo các giải pháp tối ưu cho các bài toán được xác định rõ; tuy nhiên, khả năng mở rộng bị hạn chế do độ phức tạp tính toán tăng theo cấp số nhân theo quy mô bài toán. Các phương pháp Branch and Bound (B&B) đã được sử dụng kết hợp với các mô hình ILP/MILP để tìm ra các giải pháp tối ưu cho việc lập kế hoạch cho hệ thống sản xuất linh hoạt [8] và cũng đã được điều chỉnh cho MALBP để giảm khối lượng tính toán [9]. Lập trình ràng buộc (Constraints Programming - CP) cũng đã được sử dụng để giải quyết các bài toán MALBP phức tạp, chứng minh khả năng giải quyết tối ưu một số trường hợp quy mô trung bình và cung cấp các giải pháp khả thi cho các trường hợp thực tế lớn, vốn là thách thức đối với các phương pháp chính xác khác như MILP [10].

Đối với một số trường hợp bài toán nhất định, để giải quyết các hạn chế tính toán của các phương pháp chính xác cho các bài toán MALBP lớn hơn, các thuật toán heuristic và meta-heuristic đã được phát triển và sử dụng rộng rãi. Các thuật toán metaheuristic được sử dụng rộng rãi để thu được các giải pháp gần tối ưu cho các trường hợp MALBP quy mô lớn và phức tạp trong thời gian tính toán hợp lý. Điều này tương đương với việc ưu tiên tốc độ hơn là các giải pháp chính xác, đặc biệt là đối với các bài toán phức tạp. Đồng thời, cần lưu ý rằng mặc dù nhanh hơn các phương pháp chính xác, các phương pháp tiếp cận heuristic không đảm bảo các giải pháp tối ưu, và chất lượng của giải pháp có thể thay đổi tùy thuộc vào thiết kế của heuristic và trường hợp bài toán cụ thể. Các giải pháp này cần được đánh giá và so sánh trước khi đưa ra quyết định. Các phương pháp tiếp cận heuristic và metaheuristic có thể được chia thành các phương pháp tìm kiếm dựa trên quần thể và dựa trên đơn lẻ hoặc tìm kiếm toàn ☞

cục. Nhiều bài toán MALBP đã áp dụng thuật toán cơ chế ủ mô phỏng (Simulated Anneealing - SA), chẳng hạn như giảm thiểu số lượng công nhân và trạm, giảm thiểu thời gian chu kỳ [11], và tối ưu hóa đa mục tiêu (ví dụ: tối đa hóa hiệu quả, giảm thiểu độ dài đường dây và chỉ số độ mượt) [12]. Thuật toán di truyền (Genetic Algorithm - GA) cũng được ứng dụng rộng rãi trong các biến thể MALBP đơn mục tiêu và đa mục tiêu, bao gồm các dòng mô hình hỗn hợp [7], [13], [14]. Tìm kiếm Tabu (TS) là một siêu thuật toán cải tiến, khám phá không gian giải pháp bằng cách di chuyển lặp đi lặp lại từ một giải pháp sang một giải pháp lân cận. Để tránh việc lặp lại và khuyến khích khám phá các vùng mới, TS duy trì một danh sách các giải pháp đã được truy cập gần đây hoặc các bước di chuyển tạm thời bị cấm. TS đã được sử dụng trong nhiều loại cân bằng dòng liên quan đến MALBP và đã được điều chỉnh cho các biến thể và ràng buộc cụ thể của vấn đề. Hiệu suất của thuật toán đã được so sánh với các heuristic và các phương pháp chính xác khác như lập trình ràng buộc CP [15]. Với sự phức tạp ngày càng tăng của MALBP, các phương pháp tiếp cận heuristic lai đã được phát triển, kết hợp các điểm mạnh của các metaheuristic khác nhau hoặc tích hợp các metaheuristic với các kỹ thuật tối ưu hóa khác. Các thuật toán lai này thường nhằm mục đích cải thiện chất lượng giải pháp, tốc độ hội tụ hoặc khả năng xử lý các ràng buộc cụ thể. Ví dụ, heuristic phân tích phân cấp (Hierarchical Decomposition Heuristic - HDH) do Michels và Costa (2024b) [1] đề xuất để xây dựng các giải pháp khả thi. Sim-heuristics tích hợp mô phỏng với metaheuristics để xử lý sự không chắc chắn về thời gian xử lý hoặc các tham số khác [16]. Tóm lại, mỗi phương pháp giải pháp đều có những ưu điểm và đánh đổi riêng biệt. Việc lựa chọn phương pháp sẽ phụ thuộc vào quy mô và độ phức tạp của bài toán dây chuyền lắp ráp, cũng như chất lượng giải pháp cần thiết và tài nguyên tính toán hiện có.

3. PHÁT TRIỂN MÔ HÌNH TOÁN HỌC VÀ THUẬT TOÁN

3.1. Mô hình MILP cho bài toán lập lịch trình dây chuyền lắp ráp đa công nhân

• Các giả định của bài toán:

– Trạm đa nhân công và bố cục dây chuyền: Mô hình cho phép nhiều hơn một công nhân được phân công vào một trạm làm việc duy nhất. Dây chuyền lắp ráp được giả định là tuần tự, thẳng và nối tiếp. Chỉ có một mô hình sản phẩm được thực hiện trên một loại sản phẩm duy nhất, không có sự đa dạng sản phẩm.

– Các trạm được trang bị các máy móc khác nhau và phục vụ một tập hợp tác vụ nhất định. Không có sự gián đoạn do thiếu máy móc. Không được phép gián đoạn quá trình xử lý hoặc thực hiện qua nhiều trạm. Các tác vụ không thể chia nhỏ.

– Thứ tự thực hiện tác vụ phải tuân thủ nghiêm ngặt các ràng buộc trình tự đã cho. Nếu nhiều hơn một tác vụ được giao cho một công nhân, thì việc tuân theo tập hợp các quan hệ trình tự đã xác định là điều bắt buộc. Thời gian thiết lập và thời gian di chuyển giữa các trạm là nhỏ và có thể bỏ qua. Việc lập lịch dựa trên thời gian xử lý tác vụ.

– Mỗi công nhân được phân công vào một và chỉ một trạm. Mỗi công nhân không thể di chuyển đến các trạm khác sau khi đã được phân công. Bài toán bỏ qua các yếu tố khác có thể ảnh hưởng đến thời gian xử lý và quyết định phân công, chẳng hạn như giới tính, tuổi tác, tôn giáo, công thái học, hoặc sự mệt mỏi của công nhân.

• Các ký hiệu và tập hợp:

Tập hợp	
$T = \{1, 2,..., numT\}$	Tập hợp tác vụ t
$W = \{1, 2,..., numW\}$	Tập hợp công nhân w
$S = \{1, 2,..., numS\}$	Tập hợp trạm s
Tuples	
$TTSet = \{(t_1, t_2) \mid t_1, t_2 \in T, t_1 \neq t_2\}$	Tập hợp cặp tác vụ $<t_1, t_2>$ với ràng buộc thứ tự
$TWSet = \{(t, w) \mid t \in T, w \in W\}$	Tập hợp cặp tác vụ – công nhân
$WSSet = \{(w, s) \mid w \in W, s \in S\}$	Tập hợp cặp công nhân – trạm
$TTWSet = \{(t_1, t_2, w) \mid (t_1, w) \in TWSet, (t_2, w) \in TWSet, t_1 \neq t_2\}$	Tập hợp $<t_1, t_2, w>$ cả hai tác vụ t_1 và t_2 được gán trên cùng một công nhân w, và $t_1 \neq t_2$
$BannedList = \{(t, s) \mid t \in T, s \in S\}$	Tập hợp các cặp tác vụ bị cấm trên trạm <t, w>, (ví dụ <1, 6> nghĩa là tác vụ 1 không thể được thực hiện ở trạm 6)
Prot	Thời gian thực hiện tác vụ $t \in T$

• Các biến quyết định

Biến nhị phân	
$X_{t,w} \in \{0, 1\}$	1 nếu tác vụ $t \in T$ được gán cho công nhân $w \in W$, 0 nếu không được gán
$Y_{w,s} \in \{0, 1\}$	1 nếu công nhân $w \in W$ được gán cho trạm $s \in S$, 0 nếu không được gán
$Z_{t1,t2,w} \in \{0, 1\}$	1 nếu tác vụ t_1 trước tác vụ t_2, được gán trên cùng công nhân w, 0 nếu không được gán
Biến số nguyên	
$Start_{t,w}$	Thời gian bắt đầu của tác vụ $t \in T$ nếu được gán cho công nhân $w \in W$
$Finish_{t,w}$	Thời gian kết thúc của tác vụ $t \in T$ nếu được gán cho công nhân $w \in W$
$Begin_t$	Thời gian bắt đầu của tác vụ $t \in T$
End_t	Thời gian kết thúc của tác vụ $t \in T$
C_{max}	Makespan (Thời gian tối đa hoàn thành qua tất cả các tác vụ)

• Hàm mục tiêu:

Hàm tối thiểu:

$$\text{Minimize } C_{max} \quad (1)$$

• Các ràng buộc:

1) Makespan được định nghĩa là thời gian hoàn thành công việc cuối cùng trong chuỗi các tác vụ, tức là tổng thời gian từ khi bắt đầu công việc đầu tiên đến khi kết thúc công việc cuối cùng (phép toán trong phương trình (2)).

$$C_{max} \geq E_{ndt} \quad \forall t \in TSet \quad (2)$$

2) Mỗi tác vụ được giao cho một người công nhân duy nhất.

$$\sum_{(t,w)\in TWSet} X_{t,w} = 1 \quad \forall t \in TSet \tag{3}$$

3) Mỗi công nhân được chỉ định một và chỉ một trạm làm việc duy nhất.

$$\sum_{(w,s)\in WSSet} Y_{w,s} = 1 \quad \forall w \in WSet \tag{4}$$

4) Ràng buộc tác vụ-trạm cấm (banned task-station constraints) theo phương trình (5) quy định nếu một công nhân được gán cho một trạm thì các tác vụ bị cấm tại trạm đó không thể được giao cho công nhân đó.

$$Y_{w,s} \le 1 - X_{t,w} \quad \begin{matrix}\forall(w, s) \in WSSet \\ (t, w) \in TWSet \\ (t, s) \in BannedList\end{matrix} \tag{5}$$

5) Tính nhất quán giữa tác vụ, công nhân và trạm được thể hiện qua phương trình (6), nghĩa là nếu tác vụ được giao cho công nhân ($X_{t,w} = 1$) thì công nhân đó phải làm việc tại trạm cho phép thực thi tác vụ đó.

$$X_{t,w} \le \sum_{(w,s)\in WSSet:(t,s)\notin BannedList} Y_{w,s} \quad \forall(t, w) \in TWSet \tag{6}$$

6) Các ràng buộc về thứ tự ưu tiên tác vụ được diễn đạt qua các phương trình (7) đến (9), đảm bảo các tác vụ có thứ tự hoàn thành rõ ràng và không bị trùng thời gian.

$$Z_{t_1,t_2,w} \le X_{t_1,w} \quad \forall(t_1, t_2, w) \in TTWSet \tag{7}$$

$$Z_{t_2,t_1,w} \le X_{t_2,w} \quad \forall(t_1, t_2, w) \in TTWSet \tag{8}$$

$$X_{t_1,w} + X_{t_2,w} - 1 \le Z_{t_1,t_2,w} + Z_{t_2,t_1,w} \quad \forall(t_1, t_2, w) \in TTWSet \tag{9}$$

7) Thời gian hoàn thành tác vụ của mỗi công nhân được mô tả trong các phương trình (10) đến (13), trong đó tác vụ không được giao có thời gian bằng 0.

$$Finish_{t_1,w} = Start_{t_2,w} + BigM(1 - Z_{t_1,t_2,w}) \quad \forall(t_1, t_2, w) \in TTWSet \tag{10}$$

$$Finish_{t,w} = Start_{t,w} + Pro_t \cdot X_{t,w} \quad \forall(t, w) \in TWSet \tag{11}$$

$$Start_{t,w} \le BigM \cdot X_{t,w} \quad \forall(t, w) \in TWSet \tag{12}$$

$$Finish_{t,w} \le BigM \cdot X_{t,w} \quad \forall(t, w) \in TWSet \tag{13}$$

8) Phương trình (14) và (15) tổng hợp thời gian bắt đầu và kết thúc của từng tác vụ qua các công nhân.

$$Begin_t = \sum_{(t,w)\in TWSet} Start_{t,w} \quad \forall t \in TSet \tag{14}$$

$$End_t = \sum_{(t,w)\in TWSet} Finish_{t,w} \quad \forall t \in TSet \tag{15}$$

9) Điều kiện về thứ tự thời gian của tác vụ được đảm bảo trong phương trình (16), với thời gian kết thúc của tác vụ trước phải nhỏ hơn hoặc bằng thời gian bắt đầu của tác vụ sau.

$$End_{t_1} \le Begin_{t_2} \quad \forall(t_1, t_2) \in TTSet \tag{16}$$

3.2. Thuật toán di truyền (Genetic Algorithm – GA)

Thuật toán di truyền (GA) được triển khai như một phương pháp meta-heuristic để tìm kiếm các giải pháp gần tối ưu cho bài toán. GA được mô phỏng theo quá trình chọn lọc tự nhiên, phát triển một quần thể các lịch trình ứng cử viên qua các thế hệ liên tiếp thông qua các toán tử di truyền như chọn lọc, lai tạo và đột biến. Mục tiêu của việc sử dụng GA là tìm kiếm các giải pháp gần tối ưu cho các bài toán MALBP quy mô lớn và phức tạp, trong

khi vẫn duy trì hiệu quả tính toán, đặc biệt khi các phương pháp chính xác (như MILP) trở nên kém hiệu quả.

3.2.1. Cấu trúc nhiễm sắc thể

Bước quan trọng nhất là mã hóa giải pháp thành "nhiễm sắc thể" để GA có thể xử lý. Nhiễm sắc thể được thiết kế để nắm bắt ba yếu tố cốt lõi của bài toán: phân công tác vụ-công nhân, phân công công nhân-trạm, và thứ tự tác vụ cho mỗi công nhân. Cấu trúc nhiễm sắc thể là một bộ hai mảng sử dụng biểu diễn hoán vị.

– Mảng phân công tác vụ là một vector có độ dài bằng số lượng tác vụ trong tập TSet. Mỗi vị trí trong mảng tương ứng với một tác vụ t, và giá trị tại vị trí đó là công nhân w được giao cho tác vụ đó.

– Mảng phân công công nhân là một vector có độ dài bằng số lượng công nhân trong tập WSet. Mỗi vị trí tương ứng với một công nhân w và giá trị là trạm làm việc s được giao cho công nhân đó.

– Thứ tự các tác vụ cho mỗi công nhân không được mã hóa rõ ràng mà được xác định trong quá trình giải mã để tuân thủ các ràng buộc trình tự.

3.2.2. Quy trình giải mã

Giải mã là quá trình chuyển đổi nhiễm sắc thể thành một lịch trình khả thi. Quy trình giải mã bao gồm:

– Trích xuất phân công nhằm xác định các tác vụ được giao cho mỗi công nhân và trạm được giao cho mỗi công nhân.

– Kiểm tra ràng buộc cấm (BannedList): Kiểm tra xem các cặp (tác vụ t, trạm s) có nằm trong danh sách cấm (BannedList) hay không. Nếu có vi phạm, một hình phạt lớn được áp dụng cho giá trị độ thích nghi (fitness) hoặc tiến hành sửa chữa nhiễm sắc thể.

– Lập lịch tác vụ cho công nhân: Đối với mỗi công nhân, các tác vụ đã được giao sẽ được sắp xếp theo một thứ tự khả thi bằng cách xây dựng đồ thị có hướng và thực hiện sắp xếp tô-pô để tôn trọng ràng buộc trình tự.

– Tính toán thời gian kết thúc C_{max}.

3.2.3. Quy trình thuật toán di truyền

Quá trình GA bao gồm các bước lặp lại như sau:

– Khởi tạo: GA bắt đầu bằng việc tạo ra một quần thể ban đầu gồm N nhiễm sắc thể. Để đảm bảo tính đa dạng và tăng tốc độ tìm kiếm giải pháp khả thi, quần thể được khởi tạo bằng cách kết hợp hai phương pháp: 80% giải pháp ban đầu được tạo ra bằng Heuristic tham lam (Greedy Heuristic) và 20% còn lại được tạo ra ngẫu nhiên. Heuristic tham lam phân công tác vụ theo thứ tự trình tự đã được sắp xếp tô-pô và chọn công nhân có tải trọng (workload) thấp nhất trong số các công nhân khả thi.

– Hàm thích nghi Fitness: Đánh giá chất lượng của mỗi nhiễm sắc thể (lịch trình). Mục tiêu là tối thiểu hóa thời gian hoàn thành C_{max}. Các giải pháp không khả thi (vi phạm ràng buộc trình tự hoặc (BannedList) sẽ bị gán một hình phạt cao.

– Chọn lọc: Chọn lọc giải đấu (Tournament Selection) được sử dụng để chọn các nhiễm sắc thể "cha mẹ" cho thế hệ tiếp theo. Một nhóm nhỏ k cá thể được chọn ngẫu nhiên, và cá thể có độ thích nghi tốt nhất (giá trị C_{max} thấp nhất) trong nhóm đó được chọn làm cha mẹ. ☞

– Lai ghép: Quá trình này được thực hiện dựa trên một xác suất lai ghép nhằm kết hợp hai nhiễm sắc thể cha mẹ để tạo ra con cái. Nếu lai ghép được chọn, hai vị trí ngẫu nhiên được chọn trong mảng phân công tác vụ để chia thành ba phần (Head, Body, Tail). Con cái được tạo ra bằng cách lấy Head từ cha mẹ 1, Body từ cha mẹ 2, và Tail từ cha mẹ 1. Nếu tác vụ-công nhân mới được tạo ra là không hợp lệ (không có trong TWSet), hệ thống sẽ chọn ngẫu nhiên một công nhân hợp lệ cho tác vụ đó.

– Đột biến: Đột biến được áp dụng với một xác suất nhỏ, giúp đưa vật chất di truyền mới vào quần thể và tránh bị mắc kẹt tại cực tiểu cục bộ. Chiến lược đột biến là đặt lại ngẫu nhiên (random resetting), trong đó giá trị của một gen (tức là phân công tác vụ hoặc công nhân) được thay thế bằng một giá trị ngẫu nhiên khác trong miền xác định của nó.

– Cơ chế sửa chữa: Sau khi lai ghép hoặc đột biến, cơ chế sửa chữa được áp dụng để khắc phục mọi vi phạm ràng buộc. Nếu xảy ra vi phạm BannedList, thuật toán sẽ cố gắng chuyển công nhân đến một trạm mới tương thích. Nếu không thể tìm thấy trạm hợp lệ, tác vụ sẽ được chuyển sang một công nhân khác có khả năng thực hiện và ở trạm hợp lệ.

– Điều kiện kết thúc: Vòng lặp tiến hóa này tiếp tục cho đến khi đạt đến một số lượng thế hệ cố định hoặc khi độ thích nghi hội tụ, hoặc khi đạt được giá trị C_{max} thỏa đáng.

4. PHÂN TÍCH KẾT QUẢ

Thí nghiệm được thiết kế để kiểm tra giả thuyết rằng GA có thể cung cấp giải pháp gần tối ưu chất lượng cao cho các bài toán quy mô nhỏ và vượt trội hơn CPLEX (MILP) về thời gian tính toán đối với các trường hợp quy mô lớn và phức tạp hơn. Các tiêu chí đánh giá chính bao gồm thời gian chạy (Runtime) để giải quyết các kích thước bài toán khác nhau và chất lượng giải pháp (được đo bằng C_{max} và phần trăm khoảng cách (% Gap) so với giải pháp tối ưu hoặc tốt nhất đã biết). Đối với MILP, sử dụng trình giải quyết CPLEX ILOG IBM Studio IDE Version 12.10 với giới hạn thời gian chạy là 120 giây (được tăng lên 180 giây cho các trường hợp lớn) và GA được triển khai bằng Python với các tham số chính của GA bao gồm: kích thước quần thể = 100, số thế hệ = 500, xác suất lai (crossover) = 0.8, xác suất đột biến (mutation) = 0.2. Các bộ dữ liệu được sử dụng có ba cấp độ quy mô như sau:

Bảng 1. Mục đích và quy mô dữ liệu

Quy mô	Số lượng tác vụ	Mục đích	Nguồn dữ liệu
Nhỏ	≤ 10	Xác nhận mô hình ban đầu, kiểm tra thủ công tính khả thi.	Bowman (1960), Scholl (1993) [17]
Trung bình	20	Kiểm tra tính mạnh mẽ của mô hình với độ phức tạp tăng lên.	Otto et al. (2013) [18]
Lớn	50	Đánh giá khả năng mở rộng (scalability) và hiệu quả tính toán.	Otto et al. (2013) [18]

Kết quả thí nghiệm cho thấy sự khác biệt rõ rệt về hiệu suất và khả năng mở rộng giữa phương pháp chính xác (MILP) và thuật toán siêu kinh nghiệm (GA) tùy thuộc vào quy mô bài toán. Kết quả trên các trường hợp quy mô nhỏ. Cả MILP và GA đều tìm thấy makespan tối ưu cho tất cả các trường hợp. Các giải pháp của cả hai phương pháp đều khớp nhau và không vi phạm bất kỳ ràng buộc nào, điều này xác nhận việc giải mã và xử lý ràng buộc của GA đã được triển khai chính xác. Không có sự khác biệt lớn về thời gian tính toán vì các bài toán nhỏ có độ phức tạp thấp.

Bảng 2. So sánh kết quả thí nghiệm của MILP và GA với dữ liệu nhỏ

Số lượng tác vụ	MILP (CPLEX)		GA (Python)	
	C_{max}	Runtime (secs)	C_{max}	Runtime (secs)
7	17	2.60	17	1.8
8	55	3.94	55	3.04
9	28	4.43	28	3.29
10	18	5.30	18	4.63
11	25	4.67	25	4.7

Khi độ phức tạp của bài toán tăng lên, hiệu suất bắt đầu phân hóa. MILP tìm thấy giải pháp cho 13/20 trường hợp, trong đó 8 trường hợp đạt được giải pháp tối ưu. GA tìm thấy giải pháp cho 100% (20/20) các trường hợp, tuy nhiên GA chỉ tìm thấy giải pháp tối ưu cho 2/20 trường hợp này. Điều đó chứng minh MILP có khả năng tìm giải pháp tối ưu vượt trội hơn với khoảng cách trung bình (% Gap) của kết quả GA so với giải pháp tốt nhất của MILP là khoảng 11.19%, GA nhanh hơn MILP đối với các trường hợp có thể giải quyết được (trung bình 46.86 giây so với 70.1 giây), nhưng GA lại có xu hướng vi phạm ràng buộc trong 7/20 trường hợp. Các trường hợp có phân bố thời gian xử lý đồng đều khó giải quyết hơn các trường hợp còn lại; cả MILP và GA đều không thể tìm thấy giải pháp tối ưu trong các trường hợp này. Với kịch bản quy mô lớn, khả năng mở rộng (scalability) của các phương pháp được thể hiện rõ ràng. MILP (CPLEX) thất bại (0/5) trong việc tìm ra giải pháp khả thi trong giới hạn thời gian (180 giây), do độ phức tạp tăng theo cấp số nhân (exponential complexity) của bài toán NP-hard. GA (Python) thành công trong việc tìm ra giải pháp khả thi cho tất cả 5/5 trường hợp trong thời gian tính toán hợp lý với thời gian trung bình là 326.67 giây.

Phân tích độ nhạy cũng được thực hiện nhằm đánh giá tính mạnh mẽ của các giải pháp khi các tham số đầu vào thay đổi. Thời gian xử lý là tham số có ảnh hưởng lớn nhất đến C_{max}. Khi thời gian xử lý tăng (15% và 30%) C_{max} tăng mạnh. MILP cho thấy tính đàn hồi cao hơn (C_{max} tăng ít hơn) so với GA. Ví dụ, khi thời gian xử lý tăng 30%, C_{max} được xử lý bằng MILP tăng 30.1% so với C_{max} cơ sở, trong khi đó C_{max} của GA tăng 46.6%. Điều này chỉ ra rằng MILP mạnh mẽ hơn trong trường hợp thời gian xử lý bị thay đổi. Mật độ ràng buộc thứ tự ưu tiên cũng được điều chỉnh để phân tích độ nhạy. Tăng mật độ ràng buộc ưu tiên (15% và 30%) làm tăng C_{max} của cả hai phương pháp, nhưng GA bị ảnh hưởng mạnh hơn cho thấy GA kém hiệu quả hơn khi phải xử lý các ràng buộc phụ thuộc cao. Kết luận xác nhận rằng MILP là lựa chọn tốt nhất để đảm bảo tính tối ưu cho các bài toán quy mô nhỏ và trung bình, nhưng GA là bắt buộc để tìm ra các giải pháp ☞

khả thi trong giới hạn thời gian cho các bài toán quy mô lớn mặc dù nó không đảm bảo tính tối ưu. Sự không chắc chắn về thời gian xử lý được xác định là tham số quan trọng nhất ảnh hưởng đến Cmax, với MILP thể hiện độ bền cao hơn.

5. KẾT LUẬN

Các kết quả thử nghiệm đã chỉ ra rằng trên bộ dữ liệu quy mô nhỏ cả mô hình MILP và giải thuật GA đều đạt 100% tính khả thi mà không vi phạm các ràng buộc và thời gian tính toán trung bình dưới 5 giây. Với quy mô trung bình, GA luôn tìm được giải pháp khả thi (100%) nhanh hơn MILP, dù khoảng cách chất lượng trung bình khoảng 11,19%. Ở quy mô lớn, MILP hoàn toàn thất bại trong khi GA vẫn giải quyết được tất cả các trường hợp chỉ trong dưới 350 giây, chứng tỏ khả năng mở rộng vượt trội. Kết luận chung về phương pháp GA được xem là hữu ích khi cần tìm kiếm giải pháp cho tất cả các bài toán, đặc biệt là các bài toán lớn, mặc dù không thể đảm bảo chất lượng tối ưu tuyệt đối, trong khi MILP phù hợp hơn để tìm kiếm giải pháp tối ưu trong thời gian chạy hợp lý đối với các bài toán quy mô nhỏ.

Nghiên cứu này mở ra nhiều hướng tiềm năng cho các phát triển trong tương lai: tối ưu đa mục tiêu bằng Hybrid-GA hoặc metaheuristic, xử lý thời gian thực (công nhân vắng, máy hỏng), áp dụng hyper-metaheuristics để thu hẹp khoảng cách chất lượng ở bài toán lớn, kết hợp mô phỏng để xác nhận giải pháp dưới điều kiện ngẫu nhiên, và cuối cùng là chuyển giao từ nền tảng lý thuyết vững chắc sang các hệ thống dây chuyền lắp ráp công nghiệp linh hoạt, thích ứng cao.

Lời cảm ơn:

Nghiên cứu này được trích từ bài luận văn tốt nghiệp thạc sĩ Khoa Kỹ thuật và Quản lý Công nghiệp, Trường Đại học Quốc tế, Đại học Quốc gia Thành phố Hồ Chí Minh năm 2025 dưới sự hướng dẫn của PGS,TS. thầy Phan Nguyễn Kỳ Phúc.❖

Ngày nhận bài: **03/12/2025**
Ngày phản biện: **15/12/2025**

Tài liệu tham khảo:

[1]. A. S. Michels và A. M. Costa, "*Model and heuristics for the multi-manned assembly line worker integration and balancing problem*". Int. J. Prod. Res., vol 62, số p.h 24, tr 8719-8744, tháng 12/2024, doi: 10.1080/00207543.2024.2347572.

[2]. F. Pilati, E. Ferrari, M. Gamberi, và S. Margelli, "*Multi-Manned Assembly Line Balancing: Workforce Synchronization for Big Data Sets through Simulated Annealing*". Appl. Sci., vol 11, số p.h 6, tr 2523, tháng 3/2021, doi: 10.3390/app11062523.

[3]. F. Güner, A. K. Görür, B. Satır, L. Kandiller, và John. H. Drake, "*A constraint programming approach to a real-world workforce scheduling problem for multi-manned assembly lines with sequence-dependent setup times*". Int. J. Prod. Res., vol 62, số p.h 9, tr 3212-3229, tháng 5/2024, doi: 10.1080/00207543.2023.2226772.

[4]. D. Giglio, M. Paolucci, A. Roshani, và F. Tonelli, "*Multi-manned Assembly Line Balancing Problem with Skilled Workers: A New Mathematical Formulation*". IFAC-Pap., vol 50, số p.h 1, tr 1211-1216, tháng 7/2017, doi: 10.1016/j.ifacol.2017.08.344.

[5]. S. E. Hashemi-Petroodi, S. Thevenin, S. Kovalev, và A. Dolgui, "*Markov decision process for multi-manned mixed-model assembly lines with walking workers*". Int. J. Prod. Econ., vol 255, tr 108661, tháng 1/2023, doi: 10.1016/j.ijpe.2022.108661.

[6]. M. C. O. Moreira, J.-F. Cordeau, A. M. Costa, và G. Laporte, "*Robust assembly line balancing with heterogeneous workers*". Comput. Ind. Eng., vol 88, tr 254-263, tháng

10/2015, doi: 10.1016/j.cie.2015.07.004.

[7]. Y. Jiao, H. Jin, X. Xing, M. Li, và X. Liu, "*Assembly line balance research methods, literature and development review*". Concurr. Eng., vol 29, số p.h 2, tr 183-194, tháng 6/2021, doi: 10.1177/1063293X20987910.

[8]. J. Ahn và H.-J. Kim, "*A Branch and Bound Algorithm for Scheduling of Flexible Manufacturing Systems*". IEEE Trans. Autom. Sci. Eng., vol 21, số p.h 3, tr 4382-4396, tháng 7/2024, doi: 10.1109/TASE.2023.3296087.

[9]. C. Miralles, J. P. García-Sabater, C. Andrés, và M. Cardós, "*Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: Application to Sheltered Work centres for Disabled*". Discrete Appl. Math., vol 156, số p.h 3, tr 352-367, tháng 2/2008, doi: 10.1016/j.dam.2005.12.012.

[10]. A. Ahmadi-Javid, M. Haghi, và P. Hooshangi-Tabrizi, "*Integrated job-shop scheduling in an FMS with heterogeneous transporters: MILP formulation, constraint programming, and branch-and-bound*". Int. J. Prod. Res., vol 62, số p.h 9, tr 3288-3304, tháng 5/2024, doi: 10.1080/00207543.2023.2230489.

[11]. A. Roshani và D. Giglio, "*Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time*", Int. J. Prod. Res., vol 55, số p.h 10, tr 2731-2751, tháng 5/2017, doi: 10.1080/00207543.2016.1181286.

[12]. A. Roshani và F. Ghazi Nezami, "*Mixed-model multi-manned assembly line balancing problem: a mathematical model and a simulated annealing approach*". Assem. Autom., vol 37, số p.h 1, tr 34-50, tháng 2/2017, doi: 10.1108/AA-02-2016-016.

[13]. P. Sivasankaran và P. Shahabudeen, "*Literature review of assembly line balancing problems"*. Int. J. Adv. Manuf. Technol., vol 73, số p.h 9-12, tr 1665-1694, tháng 8/2014, doi: 10.1007/s00170-014-5944-y.

[14]. N. Zamzam và A. K. El-Kharbotly, "*Balancing two-sided multi-manned assembly line under time and space constraint*". Ain Shams Eng. J., vol 15, số p.h 3, tr 102464, tháng 3/2024, doi: 10.1016/j.asej.2023.102464.

[15]. M. C. Possan Junior, A. S. Michels, và L. Magatão, "*An exact constraint programming method for the multi-manned assembly line balancing problem with assignment restrictions*". Expert Syst. Appl., vol 259, tr 125294, tháng 1/2025, doi: 10.1016/j.eswa.2024.125294.

[16]. V. Abu-Marrul, R. Martinelli, S. Hamacher, và I. Gribkovskaia, "*Simheuristic algorithm for a stochastic parallel machine scheduling problem with periodic re-planning assessment*". Ann. Oper. Res., vol 320, số p.h 2, tr 547-572, tháng 1/2023, doi: 10.1007/s10479-022-04534-5.

[17]. A. Scholl và R. Klein, "*Balancing assembly lines effectively – A computational comparison*". Eur. J. Oper. Res., vol 114, số p.h 1, tr 50-58, tháng 4/1999, doi: 10.1016/S0377-2217(98)00173-8.

[18]. A. Otto, C. Otto, và A. Scholl, "*Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing*". Eur. J. Oper. Res., vol 228, số p.h 1, tr 33-45, tháng 7/2013, doi: 10.1016/j.ejor.2012.12.029.