SINGLE MACHINE SCHEDULING

BÀI TOÁN LẬP LỊCH CHO MÁY ĐƠN

Nguyễn Chí Bằng

Ngày 12 tháng 7 năm 2024

Mục lục

Danh mục ký hiệu và ý nghĩa

$\alpha\|\beta\|\gamma$	Ký hiệu dùng để nhận dạng loại bài toán. Trong đó, trường α chỉ số lượng máy cần lập lịch, trường hợp cho máy đơn ta ký hiệu $\alpha = 1$, tức $1\|\beta\|\gamma$. Trường β chỉ đặc tính hay kiểu ràng buộc của bài toán. Trường γ chỉ hàm mục tiêu cần tối ưu.
J_{ij}	Công việc (job) thứ j được xử lý trên máy thứ i, trong trường hợp đơn máy thì $i = 1$, kể từ đây ta chỉ ký hiệu J_j.
p_j	Khoảng thời gian xử lý (processing time) của công việc thứ j hay quá trình của công việc thứ j, tức từ thời điểm bắt đầu xử lý công việc đến thời điểm hoàn thành công việc.
d_j	Thời điểm đáo hạn (due date) của công việc thứ j.
C_j	Thời điểm hoàn thành (completion time) của công việc thứ j.
$C_{\max}$	Thời gian hoàn thành tối đa (makespan) của toàn bộ công việc, được tính bằng công thức $C_{\max} = \max\limits_{1 \le j \le n} C_j$.
S_j	Thời điểm bắt đầu (starting time) của công việc thứ j, được định nghĩa bằng công thức $S_j = \max(C_{j-1}, r_j)$.
r_j	Thời điểm sẵn sàng (release time) của công việc thứ j. Nếu $r_j \neq 0$ thì công việc thứ j sẽ không được phép bắt đầu trước thời điểm sẵn sàng r_j ($S_j \geq r_j$), ngược lại, nếu $r_j = 0$ thì các công việc sẽ được phép bắt đầu tại bất kỳ thời điểm nào.
W_j	Thời gian chờ (waiting time) của công việc thứ j, tức khoảng thời gian kể từ thời điểm công việc đã sẵn sàng cho đến thời điểm bắt đầu công việc, được định nghĩa bằng công thức $W_j = S_j - r_j = C_j - p_j - r_j$.
F_j	Chu trình (flow time) của công việt thứ j, tức khoảng thời gian kể từ thời điểm công việc đã sẵn sàng cho đến khi hoàn thành, được định nghĩa bằng công thức $F_j = C_j - r_j = W_j + p_j$.

w_j Trọng số (weight) của công việc thứ j, tức mức độ ưu tiên để được xử lý của công việc thứ j.

L_j Độ đáo hạn (lateness) của công việc thứ j, được định nghĩa là độ dài từ d_j đến C_j, xác định bằng công thức $L_j = C_j - d_j$. Từ đây có thể thấy, nếu $L_j < 0$ thì công việc đã hoàn thành sớm hơn thời điểm đáo hạn, nếu $L_j > 0$ thì công việc đã hoàn thành muộn hơn thời điểm đáo hạn.

T_j Độ trễ (tardiness) của công việc thứ j, là thang đo độ trễ của công việc thứ j được định nghĩa thông qua L_j. Nếu $L_j \leq 0$ thì $T_j = 0$, ngược lại nếu $L_j > 0$ thì $T_j = L_j$, hay $T_j = \max(L_j, 0)$.

E_j Độ sớm (earliness) của công việc thứ j, là thang đo độ sớm của công việc thứ j được định nghĩa thông qua L_j. Nếu $L_j \geq 0$ thì $E_j = 0$, ngược lại nếu $L_j < 0$ thì $E_j = L_j$, hay $E_j = \max(|L_j|, 0)$.

prec Bài toán tồn tại ràng buộc có thứ tự (precedence constraint). Nếu *prec* xuất hiện trong trường β của bài toán thì bài toán tồn tại những công việc đòi hỏi phải hoàn thành trước khi công việc khác được bắt đầu, hay còn gọi là công việc tiền nhiệm (predecessor) và công việc kế nhiệm (successor). Nếu trường hợp bài toán có mỗi công việc tồn tại tối đa một tiền nhiệm và một kế nhiệm, bài toán có ràng buộc dạng dây chuyền (chains). Trường hợp có tối đa một kế nhiệm, bài toán có ràng buộc dạng in-tree. Trường hợp có tối đa một tiền nhiệm, bài toán có ràng buộc dạng out-tree. Ngược lại, nếu *prec* không xuất hiện trong trường β của bài toán, bài toàn được phép có các thứ tự công việc được sắp tự do.

prmp Bài toán tồn tại tính ưu tiên ngắt (preemption), thường được sử dụng khi có sự xuất hiện của $r_j \neq 0$. Nếu *prmp* xuất hiện trong trường β của bài toán thì công việc được phép ngắt quãng tại bất kỳ thời điểm nào để ưu tiên cho công việc khác nhằm mục đích tối ưu hàm mục tiêu của bài toán. Ngược lại, nếu *prmp* không xuất hiện trong trường β của bài toán, công việc sẽ không được phép ngắt quãng.

Chương 1

Giới thiệu bài toán lập lịch cho máy đơn và phương pháp xử lý

1.1. Khái quát về bài toán lập lịch

Vấn đề lập lịch hay còn gọi là quản lý thời gian và tối ưu hoá quy trình là một lĩnh vực quan trọng trong Vận trù học (Operation research) giúp ra quyết định phân bổ thời gian hợp lý và được sử dụng thường xuyên trong nhiều ngành công nghiệp sản xuất, dịch vụ, khoa học máy tính và trí tuệ nhân tạo. Động cơ của lĩnh vực lập lịch liên quan đến việc phân bổ tài nguyên cho các nhiệm vụ trong các khoảng thời gian nhất định nhằm tối ưu hóa hàm mục tiêu cụ thể.

Mục tiêu của lập lịch rất đa dạng tuỳ theo nhu cầu của nhà lập lịch. Có thể mục tiêu là giảm thiểu thời gian hoàn thành của nhiệm vụ cuối cùng hoặc giảm thiểu số lượng nhiệm vụ bị trễ so với thời điểm đáo hạn tương ứng của chúng, cả hai là phương pháp cốt lõi sẽ được tập trung trong mục 1.2.

Tuỳ thuộc theo dạng bài toán yêu cầu mà ta sẽ có phương tiếp cận và xử lý khác nhau. Ở đây ta chỉ tập trung xử lý cái bài toán cho máy đơn ($\alpha = 1$), các dạng bài toán được giới thiệu được chia theo bảng bên dưới

		γ			
		$\sum C_j$	$L_{\max}$	$T_{\max}$	$\sum w_j C_j$
β	$\varnothing$	$1 \Vert \sum C_j$	$1 \Vert L_{\max}$	$1 \Vert T_{\max}$	$1 \Vert \sum w_j C_j$
	$prec$	$1 \vert prec \vert \sum C_j$	$\times$	$\times$	$1 \vert prec \vert \sum w_j C_j$
	r_j	$1 \vert r_j \vert \sum C_j$	$\times$	$\times$	$\times$
	$r_j, prmp$	$1 \vert r_j, prmp \vert \sum C_j$	$\times$	$\times$	$\times$

Hình 1.1: Bảng liệt kê các dạng bài toán.

Ở phần thứ hai của chương (1.2.) sẽ giới thiệu các phương pháp lập lịch với giả định bài toán ở trạng thái tĩnh, tức tất cả các công việc đều có thể bắt đầu cùng lúc tại thời điểm $t = 0$. Trong đó, phương pháp sắp xếp theo thứ tự công việc là phương pháp cơ bản nhất. Tiếp theo là phương pháp ưu tiên đáo hạn (EDD - Earliest Due Date) và cuối cùng là phương pháp thời gian xử lý ngắn nhất tập trung vào việc tối

thiểu hóa tổng thời gian hoàn thành (SPT - Shortest Process Time) và tổng thời gian hoàn thành có trọng số (WSPT - Weighted Shortest Process Time).

Phần tiếp theo của chương (1.3.) sẽ xem xét các bài toán phức tạp hơn, nơi tồn tại các công việc có thời điểm sẵn sàng không đồng nhất. Điều này làm cho bài toán mang tính thực tế hơn trong các quy trình sản xuất khi các công việc thường không bắt đầu một cách đồng thời. Trong đó ta sẽ tập trung vào hai phương pháp chính, bao gồm: Phương pháp không ngắt quãng (non-preemptive) và phương pháp ngắt quãng (preemptive). Trong đó phương pháp ngắt quãng cho ta sự linh hoạt cao hơn trong quá trình lập lịch.

Những phương pháp này sẽ giúp các nhà quản lý sản xuất hay chuyên gia tối ưu hóa quy trình tìm được những giải pháp hiệu quả giúp cải thiện hiệu suất và giảm thiểu chi phí trong quá trình sản xuất một các tốt nhất.

Phần lớn kiến thức của chương được tham khảo từ tài liệu [1], [2], [3], [4], [5].

1.2. Bài toán trạng thái tĩnh ($r_j = 0$)

Bài toán tĩnh trong lập lịch cho máy đơn là một trong những bài toán cơ bản và quan trọng trong lĩnh vực quản lý thời gian và tối ưu hóa quy trình. Đặc điểm của bài toán tĩnh là thời điểm sẵn sàng r_j của các công việc đều đồng nhất tại $t = 0$, hay $r_j = 0$, $\forall j = \overline{1,n}$ và ký hiệu r_j lúc này không tồn tại trong trường β của bài toán.

Trong bối cảnh bài toán ở trạng thái tĩnh, phương pháp sắp xếp theo thứ tự công việc sẽ được trình bày như một cách cơ bản cho bước đầu tiếp cận các phương pháp lập lịch tối ưu hơn. Ở các phương pháp lập lịch tối ưu hơn, các công việc sẽ được sắp xếp sao cho tối thiểu hoá một hàm mục tiêu nhất định, trong đó bao gồm các dạng bài toán: Tối thiểu độ đáo hạn cực đại ($1\|L_{\max}$) hay độ trễ cực đại ($1\|T_{\max}$), tối thiểu tổng thời gian hoàn thành ($1\|\sum C_j$) và tối thiểu tổng thời gian hoàn thành có trọng số ($1\|\sum w_j C_j$).

Bài toán tĩnh cung cấp một nền tảng lý thuyết vững chắc giúp phát triển các phương pháp xử lý những dạng bài toán lập lịch phức tạp hơn, đồng thời là bước đầu tiên và quan trọng trong việc nghiên cứu và ứng dụng các thuật toán tối ưu trong lĩnh vực quản lý thời gian và tối ưu hoá quy trình.

1.2.1 Phương pháp sắp xếp theo thứ tự công việc

Phương pháp sắp xếp theo thứ tự công việc trong lập lịch cho máy đơn là một phương pháp cơ bản và dễ hiểu. Bằng cách dựa trên số thứ tự công việc được định sẵn, nguyên lý của phương pháp là sắp xếp các công việc sao cho thứ tự của công việc được sắp theo hướng tăng dần hoặc giảm dần.

Ví dụ 1. *Minh hoạ trường hợp bài toán với $n = 4$ được sắp xếp theo thứ tự giảm dần* (1.2).

Ví dụ 2. *Minh hoạ trường hợp bài toán với $n = 4$ được sắp xếp theo thứ tự tăng dần* (1.3).

Công việc (j)	4	3	2	1
p_j	2	5	1	3
d_j	6	9	8	3

Hình 1.2: Trường hợp bài toán được sắp xếp theo thứ tự giảm dần.

Công việc (j)	1	2	3	4
p_j	3	1	5	2
d_j	3	8	9	6

Hình 1.3: Trường hợp bài toán được sắp xếp theo thứ tự tăng dần.

Xét (1.3) *và* p_j *cho sẵn, ta có thể dễ dàng tính toán được các thông số* $C_j, S_j, W_j, F_j, L_j, T_j, E_j$ *và thu được bảng sau*

Công việc (j)	r_j	p_j	d_j	C_j	S_j	W_j	F_j	L_j	T_j	E_j
1	0	3	3	3	0	0	3	0	0	0
2	0	1	8	4	3	3	4	-4	0	4
3	0	5	9	9	4	4	9	0	0	0
4	0	2	6	11	9	9	11	5	5	0

Vì bài toán ở trạng thái tĩnh nên hiển nhiên $r_j = 0$, $\forall j = \overline{1,4}$. *Từ đây ta có thể xác định được*

$$C_{\max} = \sum_{j=1}^{4} p_j = 11,$$

$$L_{\max} = \max_{1 \le j \le 4} \{L_j, 0\} = 5,$$

và

$$T_{\max} = \max_{1 \le j \le 4} \{T_j, 0\} = 5.$$

Ta tính được trung bình thời gian chờ là

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = 4,$$

trung bình chu trình là

$$\overline{F} = \frac{\sum_{j=1}^{4} F_j}{4} = 6.75,$$

trung bình độ đáo hạn là

$$\overline{L} = \frac{\sum_{j=1}^{4} L_j}{4} = 0.25,$$

trung bình độ trễ là

$$\overline{T} = \frac{\sum_{j=1}^{4} T_j}{4} = 1.25,$$

và trung bình độ sớm là

$$\overline{E} = \frac{\sum_{j=1}^{4} E_j}{4} = 1.$$

Từ lý thuyết và ví dụ minh hoạ trên có thể thấy phương pháp sắp xếp theo thứ tự công việc không yêu cầu tính toán quá phức tạp. Tuy nhiên, còn tồn tại nhiều hạn chế, phương pháp sắp xếp theo thứ tự công việc không xem xét đến các yếu tố cần được tối ưu như độ đáo hạn, độ trễ hay thời gian hoàn thành, do đó có thể không đạt được điều kiện tối ưu như mong muốn và không mang lại hiệu quả cao trong ứng dụng.

Do đó, ta sẽ tập trung vào các phương pháp có khả năng cải thiện mô hình hiệu quả hơn bằng cách tối thiểu độ đáo hạn cực đại $L_{\max}$ (hay độ trễ cực đại $T_{\max}$) hoặc phương pháp giúp tối thiểu hoá tổng thời gian hoàn thành $\sum C$.

1.2.2 Phương pháp ưu tiên đáo hạn

Phương pháp ưu tiên đáo hạn, viết tắt là EDD (Earliest Due Date) là một trong những phương pháp phổ biến và hiệu quả trong lập lịch cho máy đơn với mục đích giúp tối thiểu hoá độ đáo hạn cực đại $L_{\max}$ hay độ trễ cực đại $T_{\max}$.

Nguyên lý của phương pháp ưu tiên đáo hạn là ưu tiên xử lý các công việc có thời điểm đáo hạn nhỏ nhất, từ đó sắp xếp các công việc theo thứ tự từ thời điểm đáo hạn nhỏ nhất đến thời điểm đáo hạn lớn nhất. Mục đích là giúp đảm bảo các công việc có thời điểm đáo hạn nhỏ nhất được hoàn thành trước, hạn chế khả năng các công việc bị trễ, từ đó tối thiểu được độ đáo hạn cực đại $L_{\max}$ và độ trễ cực đại $T_{\max}$.

Tối thiểu độ đáo hạn cực đại ($1\|L_{\max}$)

Với hàm mục tiêu là $L_{\max}$, ta dễ dàng tối thiểu hàm mục tiêu $L_{\max}$ bằng cách sử dụng phương pháp ưu tiên đáo hạn.

Ví dụ 3. *Giả sử ta có dữ liệu ban đầu như sau* (1.4).

Công việc (j)	1	2	3	4	5
p_j	8	10	15	9	5
d_j	30	9	23	3	11

Hình 1.4: Dữ liệu ban đầu của bài toán minh hoạ phương pháp EDD.

Ta dễ dàng tìm được C_j và L_j (1.5).
Ta có

$$L_{\max} = \max_{1 \le j \le 5} \{L_j, 0\} = 39,$$

Sử dụng phương pháp ưu tiên đáo hạn (EDD) ta được (1.6)
Kết quả là

$$L_{\max} = \max_{1 \le j \le 5} \{L_j, 0\} = 17.$$

Công việc (j)	1	2	3	4	5
p_j	8	10	15	9	5
C_j	8	18	33	42	47
d_j	30	9	23	3	11
L_j	-22	9	10	39	36

Hình 1.5: Dữ liệu bài toán minh hoạ phương pháp EDD.

Công việc (j)	4	2	5	3	1
p_j	9	10	5	15	8
C_j	9	19	24	39	47
d_j	3	9	11	23	30
L_j	6	10	13	16	17

Hình 1.6: Dữ liệu bài toán sau khi áp dụng phương pháp EDD.

Vậy để tối thiểu hàm mục tiêu $L_{\max}$ thì ta cần sắp xếp công việc theo thứ tự $4-2-5-3-1$.

Tối thiểu độ trễ cực đại ($1\|T_{\max}$)

Tương tự với hàm mục tiêu $L_{\max}$ với $T_{\max} = \max\limits_{1\le j\le n}(L_j, 0)$, ta cũng dễ dàng tối thiểu hàm mục tiêu $T_{\max}$ bằng cách sử dụng phương pháp ưu tiên đáo hạn như sau, với dữ liệu ban đầu như ở trên, ta tính được

Công việc (j)	1	2	3	4	5
p_j	8	10	15	9	5
C_j	8	18	33	42	47
d_j	30	9	23	3	11
L_j	-22	9	10	39	36
T_j	0	9	10	39	36

Ta có

$$T_{\max} = \max_{1\le j\le 5}\{T_j, 0\} = 39,$$

Trong khi đó, ta sử dụng phương pháp ưu tiên đáo hạn thì

Công việc (j)	4	2	5	3	1
p_j	9	10	5	15	8
C_j	9	19	24	39	47
d_j	3	9	11	23	30
L_j	6	10	13	16	17
T_j	6	10	13	16	17

Ta nhận được

$$T_{\max} = \max_{1 \le j \le 5} \{T_j, 0\} = 17.$$

Vậy để tối thiểu hàm mục tiêu $T_{\max}$ thì ta cần sắp xếp công việc theo thứ tự $4 - 2 - 5 - 3 - 1$.

1.2.3 Phương pháp thời gian xử lý ngắn nhất

Một cách tiếp cận tối ưu khác của phương pháp lập lịch cho máy đơn có thể kể đến là phương pháp thời gian xử lý ngắn nhất, viết tắt là SPT (Shortest Process Time) và phương pháp thời gian xử lý ngắn nhất có trọng số, viết tắt là WSPT (Weighted Shortest Process Time).

Ý tưởng của cả hai phương pháp là tập trung vào việc tối thiểu hoá trung bình thời gian chờ $\overline{W}$, đồng thời tối thiểu hoá tổng thời gian hoàn thành $\sum C_j$ hoặc $\sum w_j C_j$ nếu bài toán có tồn tại trọng số w_j, bằng cách ưu tiên các công việc có khoảng thời gian xử lý ngắn nhất sẽ được thực hiện trước.

Tối thiểu tổng thời gian hoàn thành ($1\|\sum C_j$) và tổng thời gian hoàn thành có trọng số ($1\|\sum w_j C_j$)

Ta có thể dễ dàng nhận thấy bài toán $1\|\sum C_j$ chính là trường hợp đặc biệt của bài toán $1\|\sum w_j C_j$ với trọng số $w_j = 1$. Do đó đồng nghĩa phương pháp thời gian xử lý ngắn nhất chính là phương pháp thời gian xử lý ngắn nhất có trọng số $w_j = 1$. Kể từ đây, ta chỉ cần tập trung vào bài toán $1\|\sum w_j C_j$.

Nhận xét 1 (Quy tắc thời gian xử lý ngắn nhất có trọng số)**.** *Với $\frac{w_j}{p_j}$ (hay $\frac{p_j}{w_j}$), nếu bài toán chưa tối ưu thì việc sắp xếp các công việc theo thứ tự giảm dần (hay tăng dần) theo giá trị $\frac{w_j}{p_j}$ (hay $\frac{p_j}{w_j}$) sẽ giúp bài toán tối thiểu được hàm mục tiêu $\sum w_j C_j$.*

Định lý 1. *Phương pháp thời gian xử lý ngắn nhất và phương pháp thời gian xử lý ngắn nhất có trọng số lần lượt là phương pháp tối ưu cho dạng bài toán $1\|\sum C_j$ và $1\|\sum w_j C_j$.*

Chứng minh. Ta gọi tổng thời gian hoàn thành có trọng số của hai công việc liền kề $j - k$ là S, ta đặt

$$S = w_j C_j + w_k C_k = w_j(t + p_j) + w_k(t + p_j + p_k), \tag{1.1}$$

trong đó công việc thứ k có thời điểm bắt đầu sau công việc thứ j và thời điểm bắt đầu xử lý công việc j là t, tức phương án của bài toán lúc này là $j - k$. (1.7)

Công việc	1	2	...	j	k	...	n
p	p_1	p_2	$\cdots$	p_j	p_k	$\cdots$	p_n
C	C_1	C_2	...	C_j	C_k	...	C_n

Hình 1.7: Phương án $j - k$ của bài toán.

Giả sử phương án này thoả phương pháp WSPT, tức là

$$\frac{w_j}{p_j} \leq \frac{w_k}{p_k}. \tag{1.2}$$

Giả sử ta cần sắp xếp lại vị trí 2 công việc này bằng cách tráo đổi vị trí với nhau thì ta nhận được

$$S' = w_k(t + p_k) + w_j(t + p_k + p_j) \tag{1.3}$$

với phương án là $k - j$ và thu bảng công việc như sau

Công việc	1	2	...	k	j	...	n
p	p_1	p_2	...	p_k	p_j	...	p_n
C	C_1	C_2	...	C_k	C_j	...	C_n

Hình 1.8: Phương án $k - j$ của bài toán.

Mục đích của phương pháp là tối thiểu hoá tổng thời gian hoàn thành của bài toán, vậy nên

$$S > S' \tag{1.4}$$

Từ (1.1), (1.3) và (1.4) ta được

$$\begin{aligned}
& w_j(t + p_j) + w_k(t + p_j + p_k) && > && w_k(t + p_k) + w_j(t + p_k + p_j) \\
\Leftrightarrow\ & w_j t + w_j p_j + w_k t + w_k p_j + w_k p_k && > && w_k t + w_k p_k + w_j t + w_j p_k + w_j p_j \\
\Leftrightarrow\ & w_k p_j && > && w_j p_k \\
\Leftrightarrow\ & \frac{w_k}{p_k} && > && \frac{w_j}{p_j}
\end{aligned}$$

Vậy nếu muốn $S' < S$ thì $\frac{w_j}{p_j} < \frac{w_k}{p_k}$ hay $\frac{p_j}{w_j} > \frac{p_k}{w_k}$ (đpcm). □

Ví dụ 4 (Tối thiểu tổng thời gian hoàn thành có trọng số $(1\|\sum w_j C_j)$)**.** *Ta có bảng công việc sau* (1.9)

Công việc (j)	1	2	3	4
p_j	12	4	9	10
C_j	12	16	25	35
W_j	0	12	16	25

Hình 1.9: Dữ liệu bài toán minh hoạ phương pháp WSPT.

Từ đây ta có

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = 13.25.$$

Công việc (j)	2	3	4	1
p_j	4	9	10	12
C_j	4	13	23	35
W_j	0	4	13	23

Áp dụng quy tắc quá trình ngắn nhất có trọng số với trọng số $w_j = 1$, ta được bảng công việc sau

Ta có

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = 10.$$

Vậy để tối thiểu hàm mục tiêu $\sum w_j C_j$ thì ta cần sắp xếp công việc theo thứ tự $2-3-4-1$.

Tối thiểu tổng thời gian hoàn thành có trọng số với ràng buộc có thứ tự $(1|prec|\sum w_j C_j)$

Bài toán tối thiểu tổng thời gian hoàn thành có trọng số với ràng buộc có thứ tự, ký hiệu là $1|prec|\sum w_j C_j$, là một trong những bài toán quan trọng trong lý thuyết lập lịch cho máy đơn và là bài toán mở rộng thêm ràng buộc có thứ tự của bài toán tối thiểu tổng thời gian hoàn thành có trọng số thông thường.

Bài toán $1|prec|\sum w_j C_j$ tồn tại những công việc đòi hỏi phải hoàn thành trước khi công việc khác được bắt đầu, hay còn gọi là công việc tiền nhiệm (predecessor) và công việc kế nhiệm (successor). Ở đây, ta chỉ quan tâm dạng dây chuyền (chains), tức trường hợp bài toán có mỗi công việc tồn tại tối đa một tiền nhiệm và một kế nhiệm.

Định nghĩa 1. *Cho một tập hợp gồm n công việc ($\{J_1, J_2, \ldots, J_n\}$) cần được xử lý trên một máy đơn thì lúc này ta nhận được hệ số p (p-factor) được định nghĩa bằng công thức sau*

$$\underset{1\le j\le n}{\text{hệ số } p} = \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} p_j} \tag{1.5}$$

Giả sử trên tập hợp gồm n công việc ($\{J_1, J_2, \ldots, J_n\}$), trong đó tồn tại dây chuyền I với

$$1 - 2 - \ldots - k,$$

và dây chuyền II với

$$k+1 - k+2 - \ldots - n.$$

Thì ta nhận được hệ số p của dây chuyền I là

$$\underset{1\le j\le k}{\text{hệ số } p} = \frac{\sum_{j=1}^{k} w_j}{\sum_{j=1}^{k} p_j}, \tag{1.6}$$

và hệ số p của dây chuyền II là

$$\underset{k+1\le j\le n}{\text{hệ số } p} = \frac{\sum_{j=k+1}^{n} w_j}{\sum_{j=k+1}^{n} p_j}. \tag{1.7}$$

Từ đây, ta có định lý sau

Định lý 2. *Nếu hệ số* $p_{1\le j\le k} >$ *hệ số* $p_{k+1\le j\le n}$ *thì phương pháp sắp xếp tối ưu của bài toán sẽ bắt đầu từ dây chuyền* I *rồi đến dây chuyền* II *và ngược lại.*

Chứng minh. Đặt S_I là tổng thời gian hoàn thành có trọng số của dây chuyền thứ I với

$$S_I = w_1p_1 + w_2p_2 + \ldots + w_k\sum_{j=1}^{k} p_j, \tag{1.8}$$

và S_{II} là tổng thời gian hoàn thành có trọng số của dây chuyền thứ II với

$$S_{II} = w_{k+1}\sum_{j=1}^{k+1} p_j + \ldots + w_n\sum_{j=1}^{n} p_j. \tag{1.9}$$

Ta gọi tổng thời gian hoàn thành có trọng số ban đầu là S_A với

$$S_A = S_I+S_{II} = w_1p_1+w_2(p_1+p_2)+\ldots+w_k\sum_{j=1}^{k} p_j+w_{k+1}\sum_{j=1}^{k+1} p_j+w_{k+2}\sum_{j=1}^{k+2} p_j+\ldots+w_n\sum_{j=1}^{n} p_j \tag{1.10}$$

Hoán đổi vị trí của S_I và S_{II} trên sơ đồ Gantt ta được

$$S_I' = w_{k+1}p_{k+1} + w_{k+2}(p_{k+1} + p_{k+2}) + \ldots + w_n\sum_{j=k+1}^{n} p_j, \tag{1.11}$$

và

$$S_{II}' = w_1(\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k\sum_{j=1}^{n} p_j. \tag{1.12}$$

Ta gọi tổng thời gian hoàn thành có trọng số sau khi hoán đổi là S_B với

$$\begin{aligned} S_B &= S_I' + S_{II}' \\ &= w_{k+1}p_{k+1} + w_{k+2}(p_{k+1} + p_{k+2}) + \ldots + w_n\textstyle\sum_{j=k+1}^{n} p_j \\ &\quad +w_1(\textstyle\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k\sum_{j=1}^{n} p_j \end{aligned} \tag{1.13}$$

Giả sử ta xét trường hợp

$$S_A < S_B \tag{1.14}$$

Từ (1.10), (1.13) và (1.14) ta được

$$\begin{array}{rcl} \begin{array}{c} w_1p_1 + w_2(p_1+p_2) \\ +\ldots+ w_k\sum_{j=1}^{k} p_j + w_{k+1}\sum_{j=1}^{k+1} p_j \\ +w_{k+2}\sum_{j=1}^{k+2} p_j + \ldots + w_n\sum_{j=1}^{n} p_j \end{array} & < & \begin{array}{c} w_{k+1}p_{k+1} + w_{k+2}(p_{k+1}+p_{k+2}) \\ +\ldots+ w_n\sum_{j=k+1}^{n} p_j \\ +w_1(\sum_{j=k+1}^{n} p_j + p_1) \\ +w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) \\ +\ldots+ w_k\sum_{j=1}^{n} p_j \end{array} \\ \Leftrightarrow \quad \begin{array}{c} -(w_1\sum_{j=k+1}^{n} p_j + \ldots + w_k\sum_{j=k+1}^{n} p_j) \\ +w_{k+1}\sum_{j=1}^{k} p_j + \ldots + w_n\sum_{j=1}^{k} p_j \end{array} & < & 0 \\ \Leftrightarrow \quad -(\sum_{j=1}^{k} w_j\sum_{j=k+1}^{n} p_j) + \sum_{j=k+1}^{n} w_j\sum_{j=1}^{k} p_j & < & 0 \\ \Leftrightarrow \quad \sum_{j=k+1}^{n} w_j\sum_{j=1}^{k} p_j & < & \sum_{j=1}^{k} w_j\sum_{j=k+1}^{n} p_j \\ \Leftrightarrow \quad \frac{\sum_{j=k+1}^{n} w_j}{\sum_{j=k+1}^{n} p_j} & < & \frac{\sum_{j=1}^{k} w_j}{\sum_{j=1}^{k} p_j} \\ \Leftrightarrow \quad \underset{k+1\le j\le n}{\text{hệ số } p} & < & \underset{1\le j\le k}{\text{hệ số } p} \end{array}$$

Tương tự, xét trường hợp $S_A > S_B$ ta được $\underset{k+1\le j\le n}{\text{hệ số } p} > \underset{1\le j\le k}{\text{hệ số } p}$ (đpcm). □

Do đặc tính của bài toán $1|prec|\sum w_jC_j$ cho phép ta có thể chuyển giao qua lại giữa các dây chuyền miễn vẫn đảm bảo ràng buộc thứ tự của các công việc. Từ đặc tính này ta có thể cải thiện định lý (2) giúp cho ra giải pháp tối ưu hơn.

Ý tưởng cải thiện là tiếp tục phân nhỏ các dây chuyền sẵn có, từ đó tìm ra hệ số p tối ưu trên mỗi dây chuyền, đánh giá các giá trị của hệ số p tối ưu sẽ giúp ta chuyển giao qua lại giữa các dây chuyền một cách hiệu quả từ đó tối thiểu hoá thời gian hoàn thành.

Định nghĩa 2 (hệ số p tối ưu)**.** *Ta gọi* $\underset{1\le j\le l^*}{\text{hệ số } p}$ *là hệ số p tối ưu của dây chuyền* $1-2-\ldots-k$*, ký hiệu* $p(1,\ldots,k)$*, được định nghĩa bằng công thức sau*

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right). \tag{1.15}$$

Tương tự với các dây chuyền còn lại.

Giả sử trên tập hợp gồm n công việc ($\{J_1, J_2, \ldots, J_n\}$), trong đó tồn tại dây chuyền I với

$$1-2-\ldots-k$$

và dây chuyền II với

$$k+1-k+2-\ldots-n.$$

Ta có định lý sau

Định lý 3 (Tính bất định)**.** *Nếu ta xác định được $p(1,\ldots,k)$ từ dây chuyền I thì $1,\ldots,l^*$ chính là chuỗi công việc tối ưu của dây chuyền I và chuỗi kế nhiệm phải là chuỗi công việc $p(k+1,\ldots,n)$ của dây chuyền II.*

Chứng minh. Giả sử ta có dây chuyền S có thứ tự công việc là

$$1,\ldots,u,v,u+1,\ldots,l^*$$

trong đó v là phần tử của dây chuyền khác. Ta có dây chuyền S' với chuỗi công việc

$$v,1,\ldots,l^*$$

và S'' với chuỗi công việc

$$1,\ldots,l^*,v$$

Do ta có $1,\ldots,u,u+1,\ldots,l^*$ là chuỗi công việc tối ưu, vì thế

$$\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right)>\sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.16}$$

TH1. Nếu $S'<S$ và $S''>S$ thì lần lượt có các bất đẳng thức sau

$$\frac{w_v}{p_v}>\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.17}$$

và

$$\frac{w_v}{p_v}>\sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.18}$$

Từ (1.16), (1.17) và (1.18) ta được

$$\frac{w_v}{p_v}>\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right)>\sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right)\quad\text{(thoả)}. \tag{1.19}$$

TH2. Nếu $S'>S$ và $S''<S$ thì lần lượt có các bất đẳng thức sau

$$\frac{w_v}{p_v}<\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.20}$$

và

$$\frac{w_v}{p_v}<\sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.21}$$

Từ (1.16), (1.20) và (1.21) ta được

$$\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right)>\sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right)>\frac{w_v}{p_v}\quad\text{(thoả)}. \tag{1.22}$$

TH3. Nếu $S' > S$ và $S'' > S$ thì lần lượt có các bất đẳng thức sau

$$\frac{w_v}{p_v} < \sum_{j=1}^{u} \left(\frac{w_j}{p_j} \right) \tag{1.23}$$

và

$$\frac{w_v}{p_v} > \sum_{j=u+1}^{l^*} \left(\frac{w_j}{p_j} \right) \tag{1.24}$$

Từ (1.16), (1.23) và (1.24) $\Longrightarrow$ vô lý.
TH4. Nếu $S' < S$ và $S'' < S$ thì lần lượt có các bất đẳng thức sau

$$\frac{w_v}{p_v} > \sum_{j=1}^{u} \left(\frac{w_j}{p_j} \right) \tag{1.25}$$

và

$$\frac{w_v}{p_v} < \sum_{j=u+1}^{l^*} \left(\frac{w_j}{p_j} \right) \tag{1.26}$$

Từ (1.16), (1.25) và (1.26) $\Longrightarrow$ vô lý.
Vậy luôn tồn tại $S' < S$ hoặc $S'' < S$ (đpcm). □

Ví dụ 5 (Tối thiểu tổng thời gian hoàn thành có trọng số với ràng buộc có thứ tự $(1|prec|\sum w_j C_j)$)**.** *Ta có bảng công việc sau* (1.10)

Công việc (j)	1	2	3	4	5
p_j	5	18	19	3	14
w_j	13	14	6	7	10

Hình 1.10: Dữ liệu minh hoạ bài toán $1|prec|\sum w_j C_j$.

Trong đó dây chuyền thứ I là

$$1 - 2 - 3$$

và dây chuyền thứ II là

$$4 - 5.$$

ta xác định hệ số p của dây chuyền thứ nhất bằng công thức

$$\textit{Hệ số } p = \frac{\sum w_j}{\sum p_j}$$

Chuỗi I	1	1-2	1-2-3
Hệ số p	2.6	1.17	0.78

Chuỗi II	4	4-5
Hệ số p	2.33	1

Ta có chuỗi I có

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right) = 2.6.$$

Ta có chuỗi II có

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right) = 2.33.$$

Vì $2.6 > 2.3$ *nên ta chọn chuỗi I với công việc 1.*

Chuỗi I	2	2-3
Hệ số p	0.77	0.54

Ta có chuỗi I có

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right) = 0.77.$$

Tiếp tục so với chuỗi II ta có $2.33 > 0.77$ *nên ta chọn chuỗi II với 4. Ta được chuỗi*

$$1 - 4 - \ldots - \ldots - \ldots$$

Chuỗi II	5
Hệ số p	0.71

Ta có chuỗi II có

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right) = 0.71.$$

Ta so với chuỗi I được $0.77 > 0.71$ *nên ta chọn chuỗi I với 2. Ta được chuỗi*

$$1 - 4 - 2 - \ldots - \ldots$$

Chuỗi I	3
Hệ số p	0.31

Ta so với chuỗi II được $0.71 > 0.31$ *nên ta chọn chuỗi II với 5. Ta được chuỗi*

$$1 - 4 - 2 - 5 - \ldots$$

Công việc (j)	1	4	2	5	3
p_j	5	3	18	14	19
C_j	5	8	26	40	59
w_j	13	7	14	10	6
W_j	0	5	8	26	40

Vì không còn công việc nào nên vị trí cuối cùng là công việc 3.
Vậy phương án tối ưu của bài toán là

$$1 - 4 - 2 - 5 - 3$$

Và trung bình thời gian chờ

$$\overline{W} = \frac{\sum_{j=1}^{5} W_j}{5} = 15.8.$$

1.3. Bài toán có thời điểm sẵn sàng không đồng nhất ($r_j \neq 0$)

Bài toán lập lịch máy đơn với thời điểm sẵn sàng không đồng nhất là một dạng bài toán có ràng buộc phức tạp nhưng lại vô cùng quan trọng trong thực tế. Trong dạng này, mỗi công việc có một thời điểm sẵn sàng r_j khác nhau, tức các công việc lúc này không thể được sắp xếp một cách tự do mà phải thoả điều kiện sao cho thời điểm bắt đầu công việc thứ j không được sớm hơn thời điểm sẵn sàng cho trước. Điều này thêm vào một lớp phức tạp cho quá trình lập lịch, đòi hỏi phải xem xét không chỉ thứ tự thực hiện các công việc mà còn cả thời điểm bắt đầu sao cho khả thi.

Có hai phương pháp tiếp cận hiệu quả, đó là phương pháp không ngắt quãng (non-preemptive) và phương pháp ngắt quãng (preemptive). Trong đó, phương pháp ngắt quãng cho thấy sự hiệu quả và linh hoạt hơn khi áp dụng vào thực tiễn, cũng như đưa ra giải pháp tối ưu hơn.

1.3.1 Phương pháp không ngắt quãng ($1|r_j|\sum C_j$)

Nguyên lý của phương pháp không ngắt quãng dựa trên phương pháp ưu tiên đáo hạn và xử lý các thời điểm sẵn sàng r_j bằng các quãng nghỉ nhàn rỗi, từ đó cộng dồn và làm tăng tổng thời gian hoàn thành $C_{\max}$.

Ưu điểm của phương pháp là dễ triển khai và đơn giản nhưng nhược điểm và hạn chế của phương pháp khiến cho tổng thời gian hoàn thành $C_{\max}$ tăng, từ đó giảm hiệu suất tổng thể.

Ví dụ 6. *Minh hoạ trường hợp bài toán toán với $n = 4$ áp dụng phương pháp không ngắt quãng. Ta có bảng công việc* (1.11)

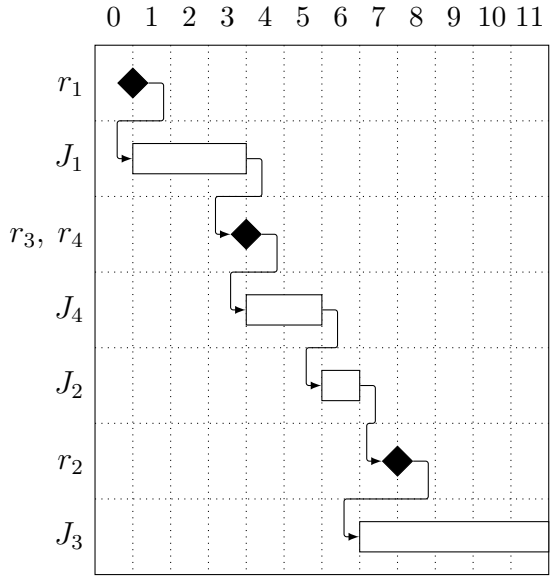
Sử dụng phương pháp ưu tiên đáo hạn ta được phương án (1.12)

Công việc (j)	1	2	3	4
p_j	3	1	5	2
d_j	3	8	9	6
r_j	0	7	3	3

Hình 1.11: Dữ liệu minh hoạ bài toán không ngắt quãng.

Công việc (j)	1	4	2	3
p_j	3	2	1	5
C_j	3	5	6	11
d_j	3	6	8	9
r_j	0	3	7	3
L_j	0	-1	-2	2

Hình 1.12: Dữ liệu minh hoạ bài toán không ngắt quãng.

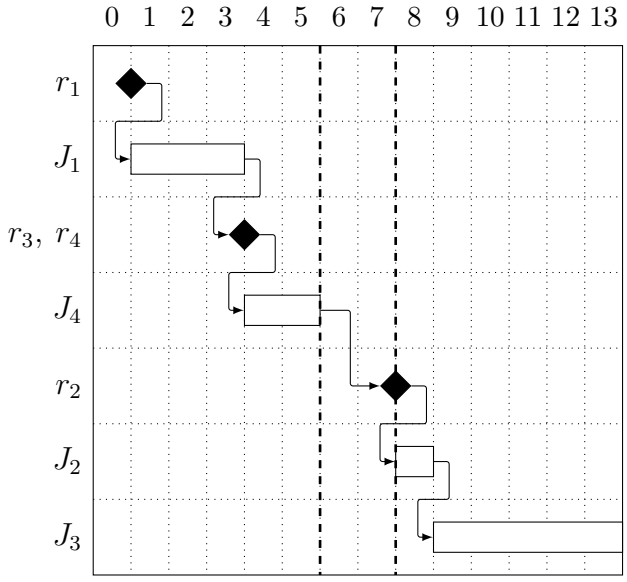


Hình 1.13: Sơ đồ Gantt của bài toán khi chưa áp dụng phương pháp không ngắt quãng

Từ ví dụ minh hoạ ta có thể thấy, công việc thứ 2 được bắt đầu ở thời điểm 7, trong khi đó khoảng thời gian từ 5 đến 7 trống và cộng dồn khiến $C_{\max}$ từ 11 tăng lên 13 và $L_{\max}$ tăng từ 2 lên 4.

1.3.2 Phương pháp ngắt quãng ($1|r_j, prmp|\sum C_j$)

Phương pháp ngắt quãng (preemptive) là một phương pháp cho phép tạm dừng công việc hiện tại để thực hiện một công việc khác có độ ưu tiên cao hơn. Sau khi công việc ưu tiên cao được hoàn thành, công việc bị tạm dừng sẽ tiếp tục được xử lý từ điểm dừng đã được ghi nhớ trước đó.



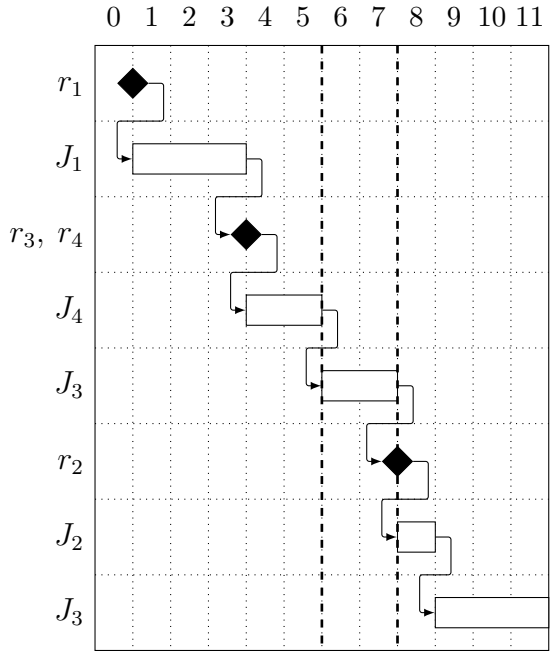
Hình 1.14: Sơ đồ Gantt của bài toán khi đã áp dụng phương pháp không ngắt quãng

Phương pháp này cải thiện điểm hạn chế của phương pháp không ngắt quãng bằng cách cho phép các công việc khác có thể lấp đầy khoảng thời gian rỗi, từ đó giúp tối thiểu hoá tổng thời gian hoàn thành hay độ đáo hạn cực đại và mang lại nhiều ứng dụng trong thực tiễn, nơi công việc có thể được sắp xếp một cách linh hoạt.

Từ ví dụ (6), ta có thể áp dụng phương pháp ngắt quãng để cải thiện giải pháp như sau

- $t = 0$: Công việc thứ 1 sẵn sàng, bắt đầu xử lý tại $t = 0$ và hoàn thành công việc tại thời điểm $t = 3$.
- $t = 3$: Công việc thứ 3 và 4 sẵn sàng, vì 4 có thứ tự ưu tiên cao hơn nên công việc thứ 4 bắt đầu xử lý tại $t = 3$ và hoàn thành công việc tại $t = 5$.
- $t = 5$: Lúc này đến công việc thứ 2 nhưng vì thời điểm sẵn sàng $r_2 = 7$ vì thế ta bỏ qua và tiếp tục công việc có mức độ ưu tiên tiếp theo. Công việc thứ 1 và thứ 4 đã hoàn thành, công việc ưu tiên tiếp theo là công việc thứ 3. Công việc thứ 3 bắt đầu xử lý tại $t = 5$ và tạm dừng tại $t = 7$ nhường chỗ cho công việc có mức độ ưu tiên cao hơn là công việc thứ 2.
- $t = 7$: Công việc thứ 2 sẵn sàng, bắt đầu xử lý tại $t = 7$ và hoàn thành tại $t = 8$.
- $t = 8$: công việc thứ 2 đã hoàn thành, ta tiếp tục xử lý công việc thứ 3, vì công việc thứ 3 đã bắt đầu tại $t = 5$ và tạm dừng tại $t = 7$, vậy còn lại 3 đơn vị thời gian cần xử lý còn lại. Ta bắt đầu công việc thứ 3 tại $t = 8$ và hoàn thành công việc thứ 3 tại $t = 11$.
- $t = 11$: Tất cả công việc đã hoàn thành và thuật toán dừng.

Lúc này giải pháp tối ưu của bài toán là $C_{\max} = 11$ và $L_{\max} = 2$, mang lại kết quả tối ưu hơn phương pháp không ngắt quãng với $C_{\max} = 13$ và $L_{\max} = 4$.



Hình 1.15: Sơ đồ Gantt của bài toán khi đã áp dụng phương pháp không ngắt quãng

a
a
a
a
a
a
a
a
a
a

Job (j)	p_j	w_j
1	p_1	w_1
2	p_2	w_2
$\vdots$	$\vdots$	$\vdots$
k	p_k	w_k
$\vdots$	$\vdots$	$\vdots$
n	p_n	w_n

Chứng minh. Giả sử $p_k = const$ với *const* là một số vô cùng lớn ($C \simeq \infty$).

$$\Longrightarrow \underset{1\le j\le n}{p\text{-factor}} = \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{k-1} p_j + p_k + \sum_{j=k+1}^{n} p_j}. \tag{1.27}$$

Từ (1.27) $\Longrightarrow \underset{1\le j\le n}{p\text{-factor}} \simeq 0 \quad (dpcm)$.

Vì ta tìm chuỗi có giá trị lớn nhất, vậy p-factor tối ưu luôn luôn là

$$\underset{1\le j\le l^*}{p\text{-factor}} = \max_{1\le l\le k-1}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right).$$

Ta chứng minh trường hợp ngược lại,

$$\underset{1\le j\le n}{p\text{-factor}} = \frac{\sum_{j=1}^{k-1} p_j + p_k + \sum_{j=k+1}^{n} p_j}{\sum_{j=1}^{n} w_j}. \tag{1.28}$$

Từ (1.28) $\Longrightarrow \underset{1\le j\le n}{p\text{-factor}} \simeq \infty \quad (dpcm)$.

Vì ta tìm chuỗi có giá trị nhỏ nhất, vậy p-factor tối ưu luôn luôn là

$$\underset{1\le j\le l^*}{p\text{-factor}} = \min_{1\le l\le k-1}\left(\frac{\sum_{j=1}^{l} p_j}{\sum_{j=1}^{l} w_j}\right).$$

□

Tài liệu tham khảo

[1] Le Minh Huy. *Single Machine Scheduling.*

[2] Michael Pinedo. *Scheduling: Theory, Algorithms, And Systems.* 01 2008.

[3] M.L. Pinedo. *Planning and Scheduling in Manufacturing and Services.* Springer series in operations research. Springer New York, 2009.

[4] PMI, editor. *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Project Management Institute, Newtown Square, PA, 5 edition, 2013.

[5] Krzysztof Postek, Alessandro Zocca, Joaquim Gromicho, and Jeffrey Kantor. *Hands-On Mathematical Optimization with AMPL in Python.* Online, 2024.