BÀI BÁO CÁO THU HOẠCH

BÀI TOÁN LẬP LỊCH CHO MÁY ĐƠN

Nguyễn Chí Bằng

Ngày 13 tháng 6 năm 2024

Mục lục

Mục lục **i**

Danh mục các kí hiệu **ii**

1 Giới thiệu bài toán lập lịch cho máy đơn **1**
1.1 Vấn đề 1
1.2 Ví dụ minh hoạ 1

2 Phương pháp xử lý bài toán lập lịch cho máy đơn **2**
2.1 Bài toán trạng thái tĩnh ($r_j \equiv 0$) 2
2.1.1 Phương pháp sắp xếp theo thứ tự công việc 3
2.1.2 Phương pháp ưu tiên đáo hạn 4
2.1.3 Phương pháp quá trình ngắn nhất 6
2.2 Bài toán có thời điểm sẵn sàng không đồng nhất ($r_j \neq 0$) 10
2.2.1 Phương pháp không ngắt quãng ($1|r_j|\sum C_j$) 10
2.2.2 Phương pháp ngắt quãng ($1|r_j, prmp|\sum C_j$) 10

Danh mục ký hiệu và ý nghĩa

$\alpha\|\beta\|\gamma$	Ký hiệu dùng để nhận dạng loại bài toán. Trong đó α chỉ số lượng máy cần lập lịch, trường hợp cho máy đơn ta ký hiệu $\alpha = 1$, tức $1\|\beta\|\gamma$. Ký hiệu β chỉ đặc tính hay kiểu ràng buộc của bài toán. Ký hiệu γ chỉ hàm mục tiêu cần tối ưu.
J_{ij}	Công việc (job) thứ j của 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 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.
$C_{\max}$	Tổng thời gian hoàn thành (makespan) của toàn bộ công việc, được tính bằng công thức $C_{\max} = \sum_{j=1}^{n} p_j$.
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 xuất hiện trong trường β của bài toán, đồng nghĩa 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 không xuất hiện trong trường β của bài toán, các công việc sẽ được phép bất 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 $W_j = C_j - p_j - r_j = S_j - r_j$.
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 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

1.1. Vấn đề

1.2. Ví dụ minh hoạ

Chương 2

Phương pháp xử lý bài toán lập lịch cho máy đơn

Chương 2 tập trung vào các phương pháp lập lịch cho máy đơn. Ở phần đầu của chương (2.1.) 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 quá trình 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 thứ hai của chương (2.2.) 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 XXX

2.1. Bài toán trạng thái tĩnh ($r_j \equiv 0$)

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 ($1\|\sum w_jC_j$).

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.

2.1.1 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:*

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

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

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

Từ p_j cho sẵn, ta có thể dễ dàng xác định được các phần tử $C_j, S_j, W_j, F_j, L_j, T_j, E_j$ 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 tổng thời gian hoàn thành $C_{\max}$.

2.1.2 Phương pháp ưu tiên đáo hạn

Phương pháp ưu tiên đáo hạn hay còn gọi là phương pháp 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 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 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, giả sử ta có giải pháp ban đầu như sau

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

Ta có

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

Sử dụng phương pháp ưu tiên đáo hạn ta được

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

Ta nhận được

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

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 $T_j = \max(L_j, 0)$, ta cũng dễ dàng tối thiểu hàm mục tiêu $T_{\max}$ bằng cách sử dụng phương pháp ưu tiên đáo hạn như sau, với giải pháp ban đầu

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

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

Sử dụng phương pháp ưu tiên đáo hạn ta được

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

2.1.3 Phương pháp quá trình ngắn nhất

Một cách tiếp cận tối ưu khác của phương pháp lập lịch cho máy đơn có thể kể đến là phương pháp quá trình ngắn nhất hay còn gọi là phương pháp SPT (Shortest Process Time) và phương pháp quá trình ngắn nhất có trọng số hay còn gọi là phương pháp 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 quá trình ngắn nhất chính là phương pháp quá trình ngắn nhất có trọng số $w_j = 1$. Kể từ đây, ta chỉ cần tập trung vào bài toán $1\|\sum w_j C_j$.

Định lý 1 (Quy tắc quá trình 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 2. *Phương pháp quá trình ngắn nhất và phương pháp quá trình 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 là S, ta đặt

$$S = w_j C_j + w_k C_k = w_j(t + p_j) + w_k(t + p_j + p_k), \tag{2.1}$$

trong đó công việc thứ k có thời điểm bắt đầu sau công việc thứ j, tức phương án của bài toán lúc này là $j - k$. Ta xem bảng bên dưới

Công việc	j	k
p	p_j	p_k
C	C_j	C_k

Hình 2.1: Phương án $j - k$ của bài toán.

giả sử ta cần sắp xếp vị trí công việc với nhau bằng cách tráo đổi vị trí thì ta nhận được

$$S' = w_k(t + p_k) + w_j(t + p_k + p_j) \tag{2.2}$$

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

Công việc	k	j
p	p_k	p_j
C	C_k	C_j

Hình 2.2: 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

$$S > S' \tag{2.3}$$

Từ (2.1), (2.2) và (2.3) 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).

Từ đây ta dễ dàng nhận thấy nếu trường hợp bài toán có trọng số là 1 thì $p_j > p_k$ sẽ tối thiểu tổng thời gian hoàn thành. □

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

Bài toán tối thiểu tổng thời gian hoàn thành có trọng số với ràng buộc có thứ tự, ký hiệu là $1|prec|\sum w_jC_j$, 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 $1|prec|\sum w_jC_j$ 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 p-factor (hệ số p) được định nghĩa bằng công thức sau*

$$\underset{1\le j\le n}{p\text{-}factor} = \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} p_j} \tag{2.4}$$

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 p-factor của dây chuyền I là

$$\underset{1\le j\le k}{p\text{-factor}} = \frac{\sum_{j=1}^{k} w_j}{\sum_{j=1}^{k} p_j} \tag{2.5}$$

và p-factor của dây chuyền II là

$$\underset{k+1\le j\le n}{p\text{-factor}} = \frac{\sum_{j=k+1}^{n} w_j}{\sum_{j=k+1}^{n} p_j}. \tag{2.6}$$

Từ đây, ta có định lý sau

Định lý 3. *Nếu $\underset{1\le j\le k}{p\text{-}factor} > \underset{k+1\le j\le n}{p\text{-}factor}$ 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

$$S_I = w_1p_1 + w_2p_2 + \ldots + w_k \sum_{j=1}^{k} p_j \tag{2.7}$$

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

$$S_{II} = w_{k+1} \sum_{j=1}^{k+1} p_j + \ldots + w_n \sum_{j=1}^{n} p_j \tag{2.8}$$

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

$$S_A = S_I + S_{II} = w_1p_1 + w_2(p_1+p_2) + \ldots + w_k \sum_{j=1}^{k} p_j + w_{k+1} \sum_{j=1}^{k+1} p_j + w_{k+2} \sum_{j=1}^{k+2} p_j + \ldots + w_n \sum_{j=1}^{n} p_j \tag{2.9}$$

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

$$S_I^{'} = w_{k+1}p_{k+1} + w_{k+2}(p_{k+1} + p_{k+2}) + \ldots + w_n \sum_{j=k+1}^{n} p_j \tag{2.10}$$

và

$$S_{II}^{'} = w_1(\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k \sum_{j=1}^{n} p_j \tag{2.11}$$

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

$$\begin{array}{rcl} S_B & = & S_I^{'} + S_{II}^{'} \\ & = & w_{k+1}p_{k+1} + w_{k+2}(p_{k+1} + p_{k+2}) + \ldots + w_n \sum_{j=k+1}^{n} p_j \\ & & +w_1(\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k \sum_{j=1}^{n} p_j \end{array} \tag{2.12}$$

Vì mục tiêu là tối thiểu hoá tổng thời gian hoàn thành của bài toán, vậy nên bắt buộc

$$S_A < S_B \tag{2.13}$$

Từ (2.9), (2.12) và (2.13) ta được

$$\begin{array}{rcl} w_1p_1 + w_2(p_1 + p_2) & & w_{k+1}p_{k+1} + w_{k+2}(p_{k+1} + p_{k+2}) \\ + \ldots + w_k \sum_{j=1}^{k} p_j + w_{k+1} \sum_{j=1}^{k+1} p_j & < & + \ldots + w_n \sum_{j=k+1}^{n} p_j \\ +w_{k+2} \sum_{j=1}^{k+2} p_j + \ldots + w_n \sum_{j=1}^{n} p_j & & +w_1(\sum_{j=k+1}^{n} p_j + p_1) \\ & & +w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) \\ & & + \ldots + w_k \sum_{j=1}^{n} p_j \end{array}$$

$$\begin{array}{rcl} \Leftrightarrow & -(w_1 \sum_{j=k+1}^{n} p_j + \ldots + w_k \sum_{j=k+1}^{n} p_j) & < \quad 0 \\ & +w_{k+1} \sum_{j=1}^{k} p_j + \ldots + w_n \sum_{j=1}^{k} p_j & \end{array}$$

$$\Leftrightarrow \quad -(\sum_{j=1}^{k} w_j \sum_{j=k+1}^{n} p_j) + \sum_{j=k+1}^{n} w_j \sum_{j=1}^{k} p_j \quad < \quad 0$$

$$\Leftrightarrow \quad \sum_{j=k+1}^{n} w_j \sum_{j=1}^{k} p_j \quad < \quad \sum_{j=1}^{k} w_j \sum_{j=k+1}^{n} p_j$$

$$\Leftrightarrow \quad \frac{\sum_{j=k+1}^{n} w_j}{\sum_{j=k+1}^{n} p_j} \quad < \quad \frac{\sum_{j=1}^{k} w_j}{\sum_{j=1}^{k} p_j}$$

$$\Leftrightarrow \quad \underset{k+1 \le j \le n}{p\text{-factor}} \quad < \quad \underset{1 \le j \le k}{p\text{-factor}}$$

Vậy nếu muốn $S_A < S_B$ thì $\underset{1 \le j \le k}{p\text{-factor}} > \underset{k+1 \le j \le n}{p\text{-factor}}$ (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 (3) giúp cho ra giải pháp tối ưu hơn bằng cách tiếp tục phân nhỏ các dây chuyền có sẵn giúp tìm ra p-factor tối ưu trên mỗi dây chuyền.

Giả sử ta có dây chuyền I với

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

Ta nhận được định lý sau

Định nghĩa 2 (p-factor tối ưu)**.** *Ta gọi* $\underset{1\le j\le l^*}{p\text{-}factor}$ *là p-factor tối ưu của dây chuyền* $1 - 2 - \ldots - k$*, ký hiệu* $p(1, \ldots, k)$*, nếu*

$$\underset{1\le j\le l^*}{p\text{-}factor} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right). \tag{2.14}$$

Tương tự với dây chuyền còn lại.

Định lý 4 (Tính bất định)**.** *Nếu ta xác định được* $p(1, \ldots, k)$ *từ nhánh* I *thì* $1, \ldots, l^*$ *là chuỗi công việc tối ưu trên nhánh* I *và chuỗi kế nhiệm phải là chuỗi công việc* $p(k + 1, \ldots, n)$ *của nhánh* II*.*

Chứng minh. □

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

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

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