SINGLE MACHINE SCHEDULING

BÀI TOÁN LẬP LỊCH CHO MÁY ĐƠN

Nguyễn Chí Bằng

Ngày 12 tháng 11 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. Được tính bằng công thức $C_j = S_j + p_j$.
$C_{\max}$	Thời gian hoàn thành cực đại (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) hay thời điểm phát hành 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 công việc thứ j 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 thuật toán 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 một hàm mục tiêu cụ thể.

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

Tuỳ thuộc theo dạng bài toán yêu cầu mà ta sẽ có phương pháp 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 chia theo bảng bên dưới

		γ			
		$C_{\max}$	$L_{\max}$	$T_{\max}$	$\sum w_j C_j$
β	$\varnothing$	$1 \| C_{\max}$	$1 \| L_{\max}$	$1 \| T_{\max}$	$1 \| \sum w_j C_j$
	$prec$				$1 \| prec \| \sum w_j C_j$
	r_j	$1 \| r_j \| C_{\max}$	$1 \| r_j \| L_{\max}$		
	$r_j, prmp$		$1 \| r_j, prmp \| L_{\max}$		

Hình 1.1: Bảng liệt kê các dạng bài toán.

Ở phần thứ hai của mục (1.2) sẽ giới thiệu các thuật toán lập lịch với giả định bài toán ở trạng thái tĩnh ($r_j = 0$), tức tất cả các công việc đều có thể bắt đầu cùng lúc tại thời điểm $t = 0$. Trong đó, thuật toán sắp xếp theo thứ tự công việc là thuật toán cơ bản nhất. Tiếp theo là thuật toán ưu tiên đáo hạn (EDD - Earliest Due Date) và cuối cùng là thuật toán 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 ($r_j \neq 0$). Điều này làm cho bài toán mang tính thực tế hơn trong các quy trình sản xuất khi các công việc thường có khoảng thời gian bắt đầu không đồng nhất tại thời điểm $t = 0$. Trong đó, ta sẽ tập trung vào thuật toán chính là thuật toán nhánh cận (branch and bound) - một trong những phương pháp quan trọng giúp hỗ trợ xử lý hầu hết các bài toán có thời điểm sẵn sàng không đồng nhất. Cùng với đó là hai thuật toán bổ trợ khác, bao gồm: thuật toán không ngắt quãng (non-preemptive) và thuật toán ngắt quãng (preemptive). Trong đó thuật toán ngắt quãng cho ta sự linh hoạt cao hơn trong quá trình lập lịch.

Những thuật toán này sẽ giúp các nhà quản lý, các nhà phân tích nghiệp vụ 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 cách tối ưu.

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, thuật toán sắp xếp theo thứ tự công việc sẽ được trình bày như một cách cơ bản cho bước đầu tiếp cận các thuật toán lập lịch tối ưu hơn. Ở các thuật toán 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 ($1\|\sum C_j$) hay 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 thuật toán 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 Thuật toán sắp xếp theo thứ tự công việc

Thuật toán sắp xếp theo thứ tự công việc trong lập lịch cho máy đơn là một thuật toán cơ bản và dễ hiểu. Bằng cách dựa trên số thứ tự công việc được định sẵn, nguyên lý của thuật toán 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).

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.

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)	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 thuật toán sắp xếp theo thứ tự công việc không yêu cầu tính toán quá phức tạp. Tuy nhiên, còn tồn tại nhiều hạn chế, thuật toán 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 thuật toán 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 thuật toán giúp tối thiểu hoá tổng thời gian hoàn thành $\sum C$.

1.2.2 Thuật toán ưu tiên đáo hạn

Thuật toán ưu tiên đáo hạn, viết tắt là *EDD (Earliest Due Date)* là một trong những thuật toán 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 thuật toán 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 thuật toán ưu tiên đáo hạn.

Định lý 1. *Thuật toán ưu tiên đáo hạn (EDD) là thuật toán tối ưu cho dạng bài toán* $1\|L_{\max}$.

Chứng minh. Ta gọi S là chuỗi công việc trong đó tồn tại hai công việc liền kề $j-k$, 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 $j-k$. (1.4)

Do đó ta được độ đáo hạn của công việc j và k trong chuỗi S lần lượt là

$$L_j(S) = p_j - d_j \tag{1.1}$$

và

$$L_k(S) = p_j + p_k - d_k. \tag{1.2}$$

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

Hình 1.4: Phương án $j - k$ của bài toán.

Giả sử phương án từ chuỗi công việc S thoả thuật toán EDD, tức là

$$d_j < d_k. \tag{1.3}$$

Giả sử ta cần sắp xếp lại vị trí hai công việc này bằng cách tráo đổi vị trí với nhau thì ta nhận được chuỗi công việc S', từ đó phương án là $k - j$. (1.5)

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 1.5: Phương án $k - j$ của bài toán.

Lúc này độ đáo hạn của hai công việc k và j trong chuỗi công việc S' lần lượt là

$$L_k(S') = p_k - d_k \tag{1.4}$$

và

$$L_j(S') = p_k + p_j - d_j. \tag{1.5}$$

Từ (1.5), (1.4) và (1.3) ta dễ dàng nhận thấy

$$L_j(S') > L_k(S') \tag{1.6}$$

Mục đích của thuật toán là tối thiểu độ đáo hạn cực đại, vậy nên

$$L_{\max}(S) \quad < \quad L_{\max}(S') \tag{1.7}$$

hay

$$\max\left[L_j(S), L_k(S)\right] \quad < \quad \max\left[L_k(S'), L_j(S')\right]. \tag{1.8}$$

Từ (1.6), ta có thể viết lại bất đẳng thức (1.8) như sau

$$\left[\begin{array}{rcl} L_j(S) & < & L_j(S') \\ L_k(S) & < & L_j(S') \end{array}\right. \tag{1.9}$$

Từ (1.1), (1.2) và (1.5) $\Rightarrow$

$$\left[\begin{array}{rcl} p_j - d_j & < & p_k + +p_j - d_j \\ p_j + p_k - d_k & < & p_k + p_j - d_j \end{array}\right. \Leftrightarrow \left[\begin{array}{rcl} p_k & > & 0 \\ d_k & > & d_j \end{array}\right.$$

Vậy nếu muốn $L_{\max}(S) < L_{\max}(S')$ thì $d_k > d_j$ (đpcm). □

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

Hình 1.6: Dữ liệu ban đầu của bài toán minh hoạ thuật toán EDD.

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 1.7: Dữ liệu bài toán minh hoạ thuật toán 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 1.8: Dữ liệu bài toán sau khi áp dụng thuật toán EDD.

Ví dụ 3. *Giả sử ta có dữ liệu ban đầu như sau* (1.6).
Ta dễ dàng tìm được C_j và L_j (1.7).
Ta có

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

Sử dụng thuật toán ưu tiên đáo hạn (EDD) ta được (1.8)
Kết quả là

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

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

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

Trong khi đó, ta sử dụng thuật toán ưu tiên đáo hạn thì
Ta nhận được

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

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
T_j	0	9	10	39	36

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

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 Thuật toán ưu tiên phát hành

Thuật toán ưu tiên phát hành, viết tắt là *ERD (Earliest Release Date)* là một trong những thuật toán phổ biến và hiệu quả trong lập lịch cho máy đơn với mục đích giúp tối thiểu hoá thời gian hoàn thành cực đại $C_{\max}$.

Nguyên lý của thuật toán ưu tiên phát hành là ưu tiên xử lý các công việc có thời điểm phát hành nhỏ nhất, từ đó sắp xếp các công việc theo thứ tự từ thời điểm phát hành nhỏ nhất đến thời điểm phát hành lớn nhất. Mục đích là giúp đảm bảo các công việc có thời điểm phát hành nhỏ nhất được hoàn thành trước, hạn chế khả năng các công việc bị trễ, từ đó tối thiểu được thời gian hoàn thành cực đại $C_{\max}$.

Tối thiểu thời gian hoàn thành cực đại ($1||C_{\max}$)

Với hàm mục tiêu là $C_{\max}$, ta dễ dàng tối thiểu hàm mục tiêu $C_{\max}$ bằng cách sử dụng thuật toán ưu tiên phát hành.

Định lý 2. *Thuật toán ưu tiên phát hành (ERD) là thuật toán tối ưu cho dạng bài toán* $1||C_{\max}$.

Chứng minh. Ta gọi S là chuỗi công việc trong đó tồn tại hai công việc liền kề $j-k$, 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 $j-k$. (1.9)

Do đó ta được thời gian hoàn thành của công việc j và k trong chuỗi S lần lượt là

$$C_j(S) = S_j + p_j = \max(C_{j-1}, r_j) + p_j \tag{1.10}$$

và

$$C_k(S) = S_k + p_k = \max(C_j, r_k) + p_k \tag{1.11}$$

Từ (1.10) ⇒

$$C_k(S) = \max\left[\max(C_{j-1}, r_j) + p_j, r_k\right] + p_k \tag{1.12}$$

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

Hình 1.9: Phương án $j - k$ của bài toán.

Giả sử phương án từ chuỗi công việc S thoả thuật toán ERD, tức là

$$r_j < r_k \tag{1.13}$$

Giả sử ta cần sắp xếp lại vị trí hai công việc này bằng cách tráo đổi vị trí với nhau thì ta nhận được chuỗi công việc S', từ đó phương án là $k - j$. (1.10)

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 1.10: Phương án $k - j$ của bài toán.

Lúc này thời gian hoàn thành của hai công việc k và j trong chuỗi công việc S' lần lượt là

$$C_k(S') = S_k + p_k = \max(C_{k-1}, r_k) + p_k \tag{1.14}$$

và

$$C_j(S') = S_j + p_j = \max(C_k, r_j) + p_j \tag{1.15}$$

Từ (1.14) ⇒

$$C_j(S') = \max\left[\max(C_{k-1}, r_k) + p_k, r_j\right] + p_j \tag{1.16}$$

Từ (1.14), (1.16) và (1.13) ta dễ dàng nhận thấy

$$C_j(S') > C_k(S') \tag{1.17}$$

Mục đích của thuật toán là tối thiểu thời gian hoàn thành cực đại, vậy nên

$$C_{\max}(S) < C_{\max}(S') \tag{1.18}$$

hay

$$\max\left[C_j(S), C_k(S)\right] < \max\left[C_k(S'), C_j(S')\right] \tag{1.19}$$

Từ (1.17), ta có thể viết lại bất đẳng thức (1.19) như sau

$$\left[\begin{array}{rcl} C_j(S) & < & C_j(S') \\ C_k(S) & < & C_j(S') \end{array}\right. \tag{1.20}$$

Từ (1.10), (1.12) và (1.16) $\Rightarrow$

$$\left[\begin{array}{rcl} \max(C_{j-1}, r_j) + p_j & < & \max\left[\max(C_{k-1}, r_k) + p_k, r_j\right] + p_j \\ \max\left[\max(C_{j-1}, r_j) + p_j, r_k\right] + p_k & < & \max\left[\max(C_{k-1}, r_k) + p_k, r_j\right] + p_j \end{array}\right. \tag{1.21}$$

Ta đặt chuỗi các công việc phía trước hai công việc j và k là A, từ đó ta có thể viết lại hệ bất đẳng thức (1.21) thành

$$\left[\begin{array}{rcl} \max(C_A, r_j) + p_j & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] + p_j \\ \max\left[\max(C_A, r_j) + p_j, r_k\right] + p_k & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] + p_j \end{array}\right. \tag{1.22}$$

Xét bất phương trình đầu tiên trong hệ (1.22)

$$\begin{array}{rcl} \max(C_A, r_j) + p_j & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] + p_j \\ \Leftrightarrow \quad \max(C_A, r_j) & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] \end{array} \tag{1.23}$$

$$\left[\begin{array}{rcl} C_A & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] \\ r_j & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] \end{array}\right. \tag{1.24}$$

$C_A < \max\left[\max(C_A, r_k) + p_k, r_j\right]$ hiển nhiên.

$$\begin{array}{rcl} r_j & < & \max\left[\max(C_A, r_k) + p_k, r_j\right] \\ \Leftrightarrow \quad r_j & < & \max(C_A, r_k) + p_k \end{array} \tag{1.25}$$

$r_j < \max(C_A, r_k) + p_k$ hiển nhiên do $r_j < r_k$.

$$\max\left[\max(C_A, r_j) + p_j, r_k\right] + p_k < \max\left[\max(C_A, r_k) + p_k, r_j\right] + p_j \tag{1.26}$$

$$\max\left[\max(C_A, r_j) + p_j, r_k\right] + p_k < \max(C_A, r_k) + p_k + p_j \tag{1.27}$$

$$\max\left[\max(C_A, r_j) + p_j, r_k\right] < \max(C_A, r_k) + p_j \tag{1.28}$$

$$\left[\begin{array}{rcl} \max(C_A, r_j) + p_j & < & \max(C_A, r_k) + p_j \\ r_k & < & \max(C_A, r_k) + p_j \end{array}\right. \tag{1.29}$$

$r_k < \max(C_A, r_k) + p_j$ hiển nhiên.

$$\begin{array}{rcl} \max(C_A, r_j) + p_j & < & \max(C_A, r_k) + p_j \\ \Leftrightarrow \quad \max(C_A, r_j) & < & \max(C_A, r_k) \\ \Leftrightarrow \quad r_j & < & r_k \end{array} \tag{1.30}$$

□

1.2.4 Thuật toán thời gian xử lý ngắn nhất

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

Ý tưởng của cả hai thuật toán 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 thuật toán thời gian xử lý ngắn nhất chính là thuật toán 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 3. *Thuật toán thời gian xử lý ngắn nhất và thuật toán thời gian xử lý ngắn nhất có trọng số lần lượt là thuật toán tối ưu cho dạng bài toán $1\|\sum C_j$ và $1\|\sum w_j C_j$.*

Chứng minh.

- **Trường hợp công việc liền kề:**

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

  $$S = w_j C_j + w_k C_k = w_j(t + p_j) + w_k(t + p_j + p_k), \tag{1.31}$$

  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 $j - k$. (1.11)

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 1.11: Phương án $j - k$ của bài toán.

Giả sử phương án này thoả thuật toán WSPT, tức là

$$\frac{w_j}{p_j} \leq \frac{w_k}{p_k}. \tag{1.32}$$

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

$$S' = w_k(t + p_k) + w_j(t + p_k + p_j) \tag{1.33}$$

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 1.12: Phương án $k - j$ của bài toán.

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

$$S > S' \tag{1.34}$$

Từ (1.31), (1.33) và (1.34) ta được

$$\begin{aligned}
& w_j(t + p_j) + w_k(t + p_j + p_k) && > && w_k(t + p_k) + w_j(t + p_k + p_j) \\
\Leftrightarrow\ & w_jt + w_jp_j + w_kt + w_kp_j + w_kp_k && > && w_kt + w_kp_k + w_jt + w_jp_k + w_jp_j \\
\Leftrightarrow\ & w_kp_j && > && w_jp_k \\
\Leftrightarrow\ & \frac{w_k}{p_k} && > && \frac{w_j}{p_j}
\end{aligned}$$

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).

- **Trường hợp công việc không liền kề:**

  Giả sử công việc thứ k có thời điểm bắt đầu sau một dãy các công việc khác. Ta lấy công việc thứ j là công việc cần xét, tức phương án của bài toán lúc này là $j \to j + 1 \to \ldots \to k - 1 \to k$. (1.13)

Công việc	1	2	...	j	$j+1$	...	$k-1$	k	...	n
p	p_1	p_2	...	p_j	p_{j+1}	...	p_{k-1}	p_k	...	p_n
C	C_1	C_2	...	C_j	C_{j+1}	...	C_{k-1}	C_k	...	C_n

Hình 1.13: Phương án $j \to j + 1 \to \ldots \to k - 1 \to k$ của bài toán.

Từ chứng minh của công việc liền kề, ta thấy rằng, công việc j và $j + 1$ chỉ hoán đổi vị trí khi và chỉ khi

$$\frac{w_{j+1}}{p_{j+1}} < \frac{w_j}{p_j}$$

hay

$$\frac{p_j}{w_j} > \frac{p_{j+1}}{w_{j+1}}.$$

Nếu tồn tại một công việc thứ 3, giả sử $j+2$, thì công việc j và $j+2$ chỉ hoán đổi khi và chỉ khi

$$\frac{w_{j+2}}{p_{j+2}} < \frac{w_{j+1}}{p_{j+1}} < \frac{w_j}{p_j}$$

hay

$$\frac{p_j}{w_j} > \frac{p_{j+1}}{w_{j+1}} > \frac{p_{j+2}}{w_{j+2}}.$$

Vậy hiển nhiên, chuỗi công việc $j \to j+1 \to \ldots \to k-1 \to k$ sẽ được sắp xếp theo chiều tăng dần dựa trên giá trị phân thức $\frac{p}{w}$ hoặc theo chiều giảm dần dựa trên phân thức $\frac{w}{p}$. (đpcm)

□

Ví dụ 4 (Tối thiểu tổng thời gian hoàn thành có trọng số $(1\|\sum w_jC_j)$)**.** *Ta có bảng công việc sau* (1.14)

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 1.14: Dữ liệu bài toán minh hoạ thuật toán WSPT.

Từ đây ta có

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

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

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

Ta có

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

Vậy để tối thiểu hàm mục tiêu $\sum w_jC_j$ 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 $(1|prec|\sum w_jC_j)$

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 hệ số p (p-factor) được định nghĩa bằng công thức sau*

$$\underset{1\le j\le n}{\text{hệ số } p} = \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} p_j} \tag{1.35}$$

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

$$\underset{1\le j\le k}{\text{hệ số } p} = \frac{\sum_{j=1}^{k} w_j}{\sum_{j=1}^{k} p_j}, \tag{1.36}$$

và hệ số p của dây chuyền II là

$$\underset{k+1\le j\le n}{\text{hệ số } p} = \frac{\sum_{j=k+1}^{n} w_j}{\sum_{j=k+1}^{n} p_j}. \tag{1.37}$$

Từ đây, ta có định lý sau

Định lý 4. *Nếu* $\underset{1\le j\le k}{\text{hệ số } p} > \underset{k+1\le j\le n}{\text{hệ số } p}$ *thì thuật toán 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

$$S_I = w_1p_1 + w_2p_2 + \ldots + w_k\sum_{j=1}^{k} p_j, \tag{1.38}$$

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

$$S_{II} = w_{k+1}\sum_{j=1}^{k+1} p_j + \ldots + w_n \sum_{j=1}^{n} p_j. \tag{1.39}$$

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

$$S_A = S_I + S_{II} = w_1p_1 + w_2(p_1+p_2) + \ldots + w_k\sum_{j=1}^{k} p_j + w_{k+1}\sum_{j=1}^{k+1} p_j + w_{k+2}\sum_{j=1}^{k+2} p_j + \ldots + w_n\sum_{j=1}^{n} p_j \tag{1.40}$$

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

$$S_I' = w_{k+1}p_{k+1} + w_{k+2}(p_{k+1}+p_{k+2}) + \ldots + w_n \sum_{j=k+1}^{n} p_j, \tag{1.41}$$

và

$$S_{II}' = w_1(\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k\sum_{j=1}^{n} p_j. \tag{1.42}$$

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

$$\begin{array}{rcl} S_B & = & S_I' + S_{II}' \\ & = & w_{k+1}p_{k+1} + w_{k+2}(p_{k+1}+p_{k+2}) + \ldots + w_n\sum_{j=k+1}^{n} p_j \\ & & +w_1(\sum_{j=k+1}^{n} p_j + p_1) + w_2(\sum_{j=k+1}^{n} p_j + p_1 + p_2) + \ldots + w_k\sum_{j=1}^{n} p_j \end{array} \tag{1.43}$$

Giả sử ta xét trường hợp

$$S_A < S_B \tag{1.44}$$

Từ (1.40), (1.43) và (1.44) ta được

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

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

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

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

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

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

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 (4) 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*

$$\underset{1\le j\le l^*}{\text{hệ số } p} = \frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{1\le l\le k}\left(\frac{\sum_{j=1}^{l} w_j}{\sum_{j=1}^{l} p_j}\right). \tag{1.45}$$

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 5 (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

$$\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) > \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.46}$$

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

$$\frac{w_v}{p_v} > \sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.47}$$

và

$$\frac{w_v}{p_v} > \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.48}$$

Từ (1.46), (1.47) và (1.48) ta được

$$\frac{w_v}{p_v} > \sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) > \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \quad \text{(thoả)}. \tag{1.49}$$

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

$$\frac{w_v}{p_v} < \sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.50}$$

và

$$\frac{w_v}{p_v} < \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.51}$$

Từ (1.46), (1.50) và (1.51) ta được

$$\sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) > \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) > \frac{w_v}{p_v} \quad \text{(thoả)}. \tag{1.52}$$

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

$$\frac{w_v}{p_v} < \sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.53}$$

và

$$\frac{w_v}{p_v} > \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.54}$$

Từ (1.46), (1.53) và (1.54) $\Longrightarrow$ vô lý.
TH4. Nếu $S' < S$ và $S'' < S$ thì lần lượt có các bất đẳng thức sau

$$\frac{w_v}{p_v} > \sum_{j=1}^{u}\left(\frac{w_j}{p_j}\right) \tag{1.55}$$

và

$$\frac{w_v}{p_v} < \sum_{j=u+1}^{l^*}\left(\frac{w_j}{p_j}\right) \tag{1.56}$$

Từ (1.46), (1.55) và (1.56) $\Longrightarrow$ 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 $(1|prec|\sum w_jC_j)$). *Ta có bảng công việc sau* (1.15)

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

Hình 1.15: Dữ liệu minh hoạ bài toán $1|prec|\sum w_jC_j$.

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

Ta có chuỗi I có

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

Chuỗi II	4	4-5
Hệ số p	2.33	1

Ta có chuỗi II có

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

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

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

Chuỗi II	5
Hệ số p	0.71

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

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

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*

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

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

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

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

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.

Tồn tại một thuật toán giúp tiếp cận bài toán một cách hiệu quả, đó là thuật toán nhánh cận (branch and bound) cùng với hai thuật toán hậu phân nhánh, đó là thuật toán không ngắt quãng (non-preemptive) và thuật toán ngắt quãng (preemptive). Trong đó, thuật toán 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.

Thuật toán không ngắt quãng

Nguyên lý của *thuật toán không ngắt quãng (non-preemptive)* dựa trên thuật toán 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 thuật toán là dễ triển khai và đơn giản nhưng nhược điểm và hạn chế của thuật toán 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 thuật toán không ngắt quãng. Ta có bảng công việc* (1.16)

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

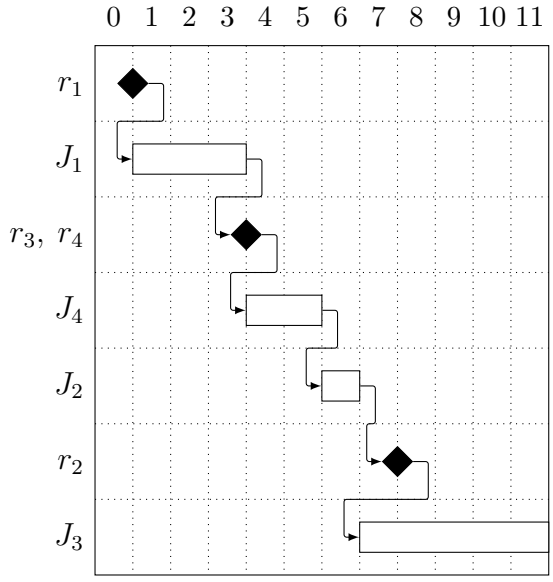
Hình 1.16: Dữ liệu minh hoạ bài toán không ngắt quãng.

Sử dụng thuật toán ưu tiên đáo hạn ta được phương án (1.17)

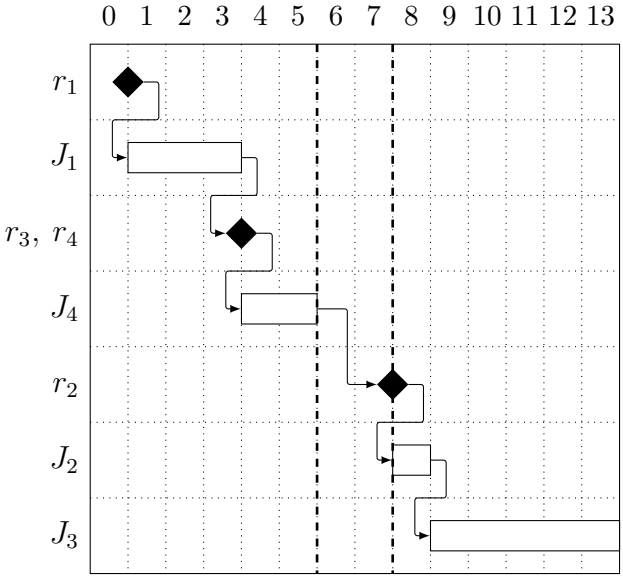
Công việc (j)	1	4	2	3
p_j	3	2	1	5
C_j	3	5	6	11
d_j	3	6	8	9
r_j	0	3	7	3
L_j	0	-1	-2	2

Hình 1.17: Dữ liệu minh hoạ bài toán 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. (1.18) (1.19)



Hình 1.18: Sơ đồ Gantt của bài toán khi chưa áp dụng thuật toán không ngắt quãng.



Hình 1.19: Sơ đồ Gantt của bài toán khi đã áp dụng thuật toán không ngắt quãng.

Thuật toán ngắt quãng ($1|prmp|\gamma$)

Thuật toán ngắt quãng (preemptive) là một thuật toán 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.

Thuật toán này cải thiện điểm hạn chế của thuật toán 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 thuật toán 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.

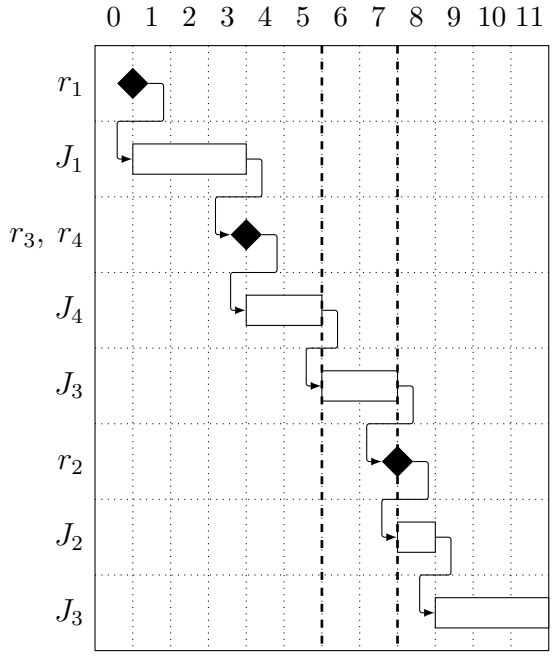
Lúc này giải pháp tối ưu của bài toán là $C_{\max} = 11$ và $L_{\max} = 2$, mang lại kết quả tối ưu hơn thuật toán không ngắt quãng với $C_{\max} = 13$ và $L_{\max} = 4$. (1.20)

Thuật toán nhánh cận

Thuật toán nhánh cận (branch and bound) là một khung thuật toán sử dụng cấu trúc cây để xử lý bài toán bằng cách phân nhánh thành các bài toán con và duyệt qua từng bài toán. Về mặt ý tưởng, xuất phát từ giải pháp ban đầu, thuật toán sẽ mở rộng các nhánh thành tập con các bài toán, từ đó liệt kê các phương án khả thi và loại bỏ các phương án không khả thi để thu được phương án tối ưu cuối cùng.

Cụ thể trong lĩnh vực lập lịch, thuật toán nhánh cận được áp dụng để xử lý lớp bài toán có ràng buộc r_j không đồng nhất. Trong những bài toán này, việc sắp xếp các công việc sao cho tối ưu một hàm cụ thể (chẳng hạn như độ đáo hạn cực đại hoặc thời gian hoàn thành cực đại) đòi hỏi phải xem xét nhiều khả năng mà công việc có thể được sắp xếp. Tuy nhiên, do số lượng công việc có thể rất lớn, cụ thể, số cách sắp xếp tăng theo giai thừa của n công việc ($n!$). Ví dụ, với 10 công việc, sẽ có đến $10! = 3,628,800$ phương án sắp xếp khác nhau.

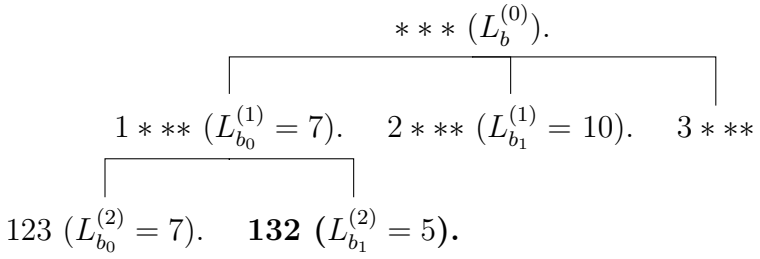
Có thể thấy, với số lượng công việc lớn, số lượng phương án khả thi trở nên khổng lồ và rất khó kiểm soát nếu không có một thuật toán loại trừ hiệu quả. Do đó, thuật toán nhánh cận giúp phân tích và loại bỏ các phương án không khả thi bằng điều kiện đã được chứng minh trước đó kết hợp với *cận dưới (lower bound)*.



Hình 1.20: Sơ đồ Gantt của bài toán khi đã áp dụng thuật toán không ngắt quãng.

Trong đó, cận dưới là một mục tiêu tối ưu của bài toán được xác định, giúp so sánh sự ưu tiên giữa các nhánh và cũng là một nhân tố quan trọng giúp cải thiện độ phức tạp tính toán một cách hiệu quả. Từ đó giúp thu hẹp không gian nghiệm và tối ưu hóa hiệu quả tính toán.

Giả sử ta có ba công việc 1, 2 và 3 cần được sắp xếp. Thuật toán nhánh cận sẽ duyệt qua các tổ hợp phương án khác nhau bằng cách phân nhánh và loại trừ những phương án không khả thi. Dưới đây là hình minh họa về cách thuật toán hoạt động (1.21).



Hình 1.21: Minh hoạ thuật toán nhánh cận.

1.3.1 Bài toán tối thiểu độ đáo hạn cực đại $L_{\max}$

Bài toán $1|r_j|L_{\max}$ về cơ bản là phiên bản mở rộng của bài toán $1||L_{\max}$ mà ta đã xử lý trước đó, với sự khác biệt là có thêm ràng buộc về thời điểm sẵn sàng không đồng nhất r_j tại thời điểm $t = 0$. Đây là một bài toán quan trọng, đóng vai trò nền

tảng cho các bài toán đa máy (multi-machine) mà ta sẽ tìm hiểu sau này. Cùng với đó, bài toán $1|r_j|L_{\max}$ cũng là một trong những bài toán mà thuật toán nhánh cận được áp dụng một cách hiệu quả.

Như đã thảo luận trước đó, cốt lõi của thuật toán nhánh cận nằm ở việc phân chia bài toán thành các bài toán con nhỏ hơn, từ đó duyệt qua các phương án khả thi, đồng thời loại bỏ những phương án không khả thi. Do đó, để giải quyết bài toán $1|r_j|L_{\max}$ một cách hiệu quả, ta cần một kỹ thuật duyệt các bài toán con sao cho tối ưu hóa quá trình tìm kiếm phương án tối ưu. Sau đây, ta sẽ đi sâu vào phương pháp duyệt và phân tích cách thức áp dụng phương pháp duyệt để thu được phương án tối ưu cho bài toán $1|r_j|L_{\max}$.

Phương pháp duyệt

Ứng với n công việc ta có $n! = n \times (n-1) \times (n-2) \times (n-3) \ldots 3 \times 2 \times 1$ phương án sắp xếp. Do đó, xuất phát từ đỉnh cây, ta gọi cấp 0, chứa 0 nút; cấp 1 chứa n nút; cấp 2 chứa $n \times (n-1)$ nút; cấp k chứa $n \times (n-1) \times (n-2) \times (n-3) \ldots (n-k+1)$ nút và cấp n chứa $n \times (n-1) \times (n-2) \times (n-3) \ldots 3 \times 2 \times 1$ nút.

Chú ý 1. *Ta ký hiệu $J_i^{(r)}$ là bài toán (hay phương án) nằm ở nút thứ i của cấp thứ r, trong đó, nút thứ i được tính bắt đầu từ trái sang phải trên cấu trúc cây, với $i, n \in \mathbb{N}$.*

Vậy đồng nghĩa ở cấp k, ta có k công việc đã được sắp xếp

$$j_1, \ldots, j_k.$$

Do đó, ta cần xác định công việc j_{k+1} sao cho tối thiểu quá trình phân nhánh. Gọi $J^{'}$ là danh sách công việc chưa được sắp xếp ngoại trừ công việc j_{k+1},

$$J^{'} = \{j_{k+2}, \ldots, j_n\}$$

và $l \in J^{'}$. Từ đây ta thu được định lý sau

Định lý 6 (Điều kiện nghiệm)**.** *Với $l \in J^{'}$, phương án $J_i^{(r)}$ là phương án chấp nhận được khi và chỉ khi $r_{j_{k+1}}$ thoả điều kiện sau,*

$$r_{j_{k+1}} < \min_{l \in J^{'}}(\max_{l \in J^{'}}(t, r_l) + p_l) \tag{1.57}$$

Chứng minh.

Để danh sách công việc được tối ưu, tức không có bất kỳ công việc nào trễ hạn, độ đáo hạn phải luôn nhỏ hơn hoặc bằng 0. Do đó,

$$L_l \leq 0,$$

mà ta có $L_l = C_l - d_l$, nên

$$C_l - d_l \leq 0,$$

hay

$$C_l \leq d_l. \tag{1.58}$$

Do thuật toán EDD sắp xếp công việc có thời điểm đáo hạn từ nhỏ đến lớn, vậy công việc tiếp theo cần thực hiện phải là

$$\min_{l\in J'}(d_l). \tag{1.59}$$

Từ (1.58) và (1.59), ta được

$$\min_{l\in J'}(C_l) \leq \min_{l\in J'}(d_l). \tag{1.60}$$

Hiển nhiên, thời điểm hoàn thành của công việc tiếp theo, $\min_{l\in J'}(C_l)$, luôn phải lớn hơn thời điểm xuất phát của công việc trước đó là $r_{j_{k+1}}$, hay

$$\min_{l\in J'}(C_l) > r_{j_{k+1}}. \tag{1.61}$$

Mặt khác, ta có thể viết C_l dưới dạng sau

$$C_l = \max_{l\in J'}(t, r_l) + p_l. \tag{1.62}$$

Từ (1.61) và (1.62), ta được

$$r_{j_{k+1}} < \min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) \tag{1.63}$$

Vậy để phương án $J_i^{(r)}$ là phương án chấp nhận được, $r_{j_{k+1}}$ phải thoả điều kiện (1.63). (đpcm) □

Định nghĩa 3 (Cận dưới)**.** *Với mỗi bài toán $J_i^{(r)}$ nếu thoả điều kiện nghiệm thì luôn tồn tại một phương án chấp nhận được có giá trị tối ưu được gọi là cận dưới của bài toán $J_i^{(r)}$, ký hiệu $L_{b_i}^{(r)}$.*

Ví dụ 7. *Ta có bảng công việc sau* (1.22)

Công việc (j)	1	2	3	4
p_j	4	2	6	5
r_j	0	1	3	5
d_j	8	12	11	10

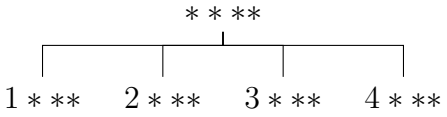
Hình 1.22: Dữ liệu minh hoạ bài toán $1|r_j|L_{\max}$.

Bằng cách áp dụng thuật toán nhánh cận, ta thu được sơ đồ cây cấp 1 như sau (1.31)

Ở phương án $J_0^{(1)}$ $(1 * **)$ có $r_{j_{k+1}} = r_1 = 0$, áp dụng phương pháp duyệt, ta xét điều kiện nghiệm

Từ bảng (1.24)*, ta được*

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 3 > r_{j_{k+1}},$$



Hình 1.23: Sơ đồ cây cấp 1.

Công việc (j)	1	2	3	4
p_j	4	2	6	5
r_j	0	1	3	5
d_j	8	12	11	10
$\max_{l\in J'}(t, r_l) + p_l$		3	9	10

Hình 1.24: Bảng dữ liệu $\max_{l\in J'}(t, r_l) + p_l$ cho phương án $J_0^{(1)}$.

vậy phương án $J_0^{(1)}$ thoả điều kiện nghiệm.

Tương tự, ở phương án $J_1^{(1)}$ $(2 * **)$ có $r_{j_{k+1}} = r_2 = 1$, xét điều kiện nghiệm ta được

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 4 > r_{j_{k+1}},$$

vậy phương án $J_1^{(1)}$ thoả điều kiện.

Ở phương án $J_2^{(1)}$ $(3 * **)$ có $r_{j_{k+1}} = r_3 = 3$, xét điều kiện nghiệm ta được

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 3 = r_{j_{k+1}},$$

vậy phương án $J_2^{(1)}$ không thoả điều kiện.

Tương tự, ở phương án $J_3^{(1)}$ $(4 * **)$ có $r_{j_{k+1}} = r_4 = 5$, xét điều kiện nghiệm ta được

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 3 < r_{j_{k+1}},$$

vậy phương án $J_3^{(1)}$ không thoả điều kiện.

Ta giữ lại các phương án thoả điều kiện nghiệm, gồm phương án $J_0^{(1)}$, $J_1^{(1)}$. Từ đây, ta tiếp tục xét cận dưới của các phương án.

Ở phương án $J_0^{(1)}$, áp dụng thuật toán EDD, ta thu được

Từ bảng (1.25), cận dưới của phương án $J_0^{(1)}$ là $L_{b_0}^{(1)} = 5$.

Tương tự, ở phương án $J_1^{(1)}$

Từ (1.26), cận dưới của phương án $J_1^{(1)}$ là $L_{b_1}^{(1)} = 6$.

Từ đây, ta dễ dàng loại trừ phương án $J_1^{(1)}$ do $L_{b_0}^{(1)} < L_{b_1}^{(1)}$ (1.27) và tiếp tục phân nhánh với gốc nằm ở phương án $J_0^{(1)}$.

Vì công việc 1 đã được sắp, do đó $t = 4$.

Phương án $J_0^{(2)}$ $(12 * *)$ có $r_{j_{k+1}} = r_2 = 1$, xét điều kiện nghiệm ta được

Công việc (j)	1	4	3	2
p_j	4	5	6	2
r_j	0	5	3	1
d_j	8	10	11	12
$L_{\max}$	-4	-1	4	5

Hình 1.25: Bảng dữ liệu thuật toán EDD cho phương án $J_0^{(1)}$.

Công việc (j)	2	1	4	3
p_j	2	4	5	6
r_j	1	0	5	3
d_j	12	8	10	11
$L_{\max}$	-10	-2	1	6

Hình 1.26: Bảng dữ liệu thuật toán EDD cho phương án $J_1^{(1)}$.

$**** (L_b^{(0)}).$

$1*** (L_{b_0}^{(1)} = 5).$ $\quad 2*** (L_{b_1}^{(1)} = 6).$ $\quad 3***$ $\quad 4***$

Hình 1.27: Sơ đồ cây cấp 1 với cận dưới.

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 10 > r_{j_{k+1}},$$

vậy phương án $J_0^{(2)}$ *thoả điều kiện, tương tự, ta thu được* $L_{b_0}^{(2)} = 6$.

Phương án $J_1^{(2)}$ $(13**)$ *có* $r_{j_{k+1}} = r_3 = 3$, *xét điều kiện nghiệm ta được*

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 6 > r_{j_{k+1}},$$

vậy phương án $J_1^{(2)}$ *thoả điều kiện, tương tự, ta thu được* $L_{b_1}^{(2)} = 5$.

Phương án $J_2^{(2)}$ $(14**)$ *có* $r_{j_{k+1}} = r_4 = 5$, *xét điều kiện nghiệm ta được*

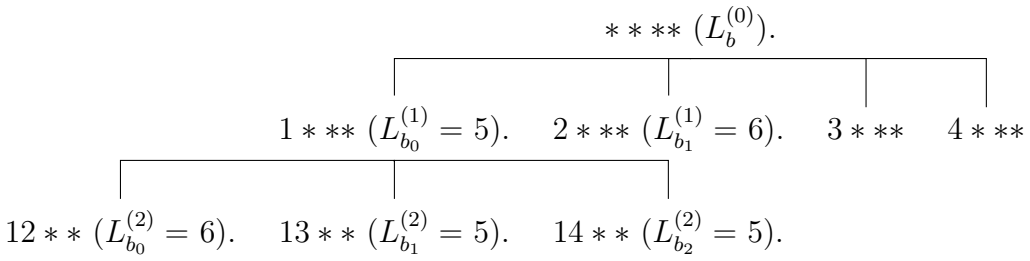
$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 6 > r_{j_{k+1}},$$

vậy phương án $J_2^{(2)}$ *thoả điều kiện, tương tự, ta thu được* $L_{b_2}^{(2)} = 5$.

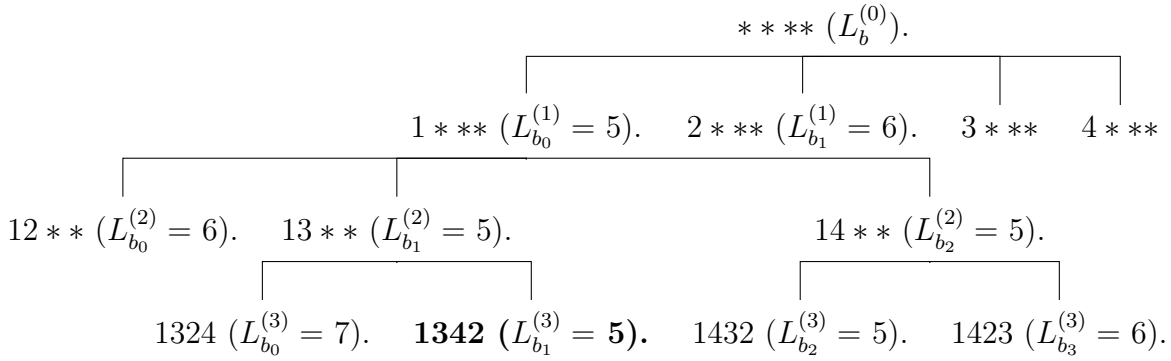
Từ đây, ta dễ dàng loại trừ phương án $J_0^{(2)}$ (1.28) *và tiếp tục phân nhánh với gốc nằm ở phương án* $J_1^{(2)}$ *và* $J_2^{(2)}$.

Thực hiện thuật toán tương tự, ta được $J_0^{(3)}$, $J_1^{(3)}$, $J_2^{(3)}$ *và* $J_3^{(3)}$ *đều thoả điều kiện nghiệm với cận dưới lần lượt là* $7, 5, 5, 6$. (1.29)

Loại trừ phương án $J_0^{(3)}$ *và phương án* $J_3^{(3)}$. *Ta áp dụng thuật toán không ngắt quãng, ở phương án* $J_1^{(3)}$ *ta có* $C_{\max_1}^{(3)} = 17$ (1.35), *phương án* $J_2^{(3)}$ *có* $C_{\max_2}^{(3)} = 18$ (1.36). *Từ đây ta thu được phương án tối ưu là phương án* $J_1^{(3)}$ *do* $C_{\max_1}^{(3)} < C_{\max_2}^{(3)}$.



Hình 1.28: Sơ đồ cây cấp 2 với cận dưới.



Hình 1.29: Sơ đồ cây cấp 3 với cận dưới.

Vậy phương án tối ưu của bài toán là $1 - 3 - 4 - 2$.

Áp dụng thuật toán ngắt quãng cho phương án $J_3^{(3)}$*, ta được sơ đồ* (1.37)*. Từ đây, nếu áp dụng thuật toán ngắt quãng, ta có thể thu một trong hai phương án* $J_0^{(3)}$ *hay* $J_3^{(3)}$ *đều tối ưu.*

Ta gọi bài toán có áp dụng thuật toán ngắt quãng là bài toán $1|prmp|L_{\max}$. Bài toán $1|prmp|L_{\max}$ đơn giản là bài toán $1|r_j|L_{\max}$, trong đó, ta mở rộng việc áp dụng thuật toán ngắt quãng để thu được phương án tối ưu.

1.3.2 Bài toán tối thiểu thời gian hoàn thành cực đại $C_{\max}$

Một bài toán tương tự sử dụng thuật toán nhánh cận để giải quyết là bài toán tối thiểu thời gian hoàn thành cực đại, ký hiệu $1|r_j|C_{\max}$. Giống như bài toán tối thiểu độ dao hạn cực đại $1|r_j|L_{\max}$, bài toán $1|r_j|C_{\max}$ cũng áp dụng cùng một điều kiện nghiệm, nhưng cận dưới ở đây là giá trị $C_{\max}$ đối với mỗi phương án chấp nhận được. Do đó, quy trình thuật toán được thực hiện tương tự, với điểm khác biệt duy nhất là cách tính cận dưới cho mỗi phương án.

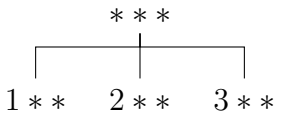
Ví dụ 8. *Ta có bảng công việc sau* (1.30)

Bằng cách áp dụng thuật toán nhánh cận, ta thu được sơ đồ cây cấp 1 như sau (1.31)

Ở phương án $J_0^{(1)}$ $(1 * *)$ *có* $r_{j_{k+1}} = r_1 = 3$*, áp dụng phương pháp duyệt, ta xét điều kiện nghiệm*

Công việc (j)	1	2	3
p_j	3	2	4
r_j	3	7	5
d_j	8	12	11

Hình 1.30: Dữ liệu minh hoạ bài toán $1|r_j|C_{\max}$.



Hình 1.31: Sơ đồ cây cấp 1.

Công việc (j)	1	3	2
p_j	3	4	2
r_j	3	5	7
d_j	8	11	12
$\max_{l\in J'}(t, r_l) + p_l$		9	9

Hình 1.32: Bảng dữ liệu $\max_{l\in J'}(t, r_l) + p_l$ cho phương án $J_0^{(1)}$.

Từ bảng (1.32)*, ta được*

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 9 > r_{j_{k+1}},$$

vậy phương án $J_0^{(1)}$ *thoả điều kiện nghiệm.*

Tương tự, ở phương án $J_1^{(1)}$ $(2 * *)$ *có* $r_{j_{k+1}} = r_2 = 7$*, xét điều kiện nghiệm ta được*

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 6 < r_{j_{k+1}},$$

vậy phương án $J_1^{(1)}$ *không thoả điều kiện.*

Ở phương án $J_2^{(1)}$ $(3 * *)$ *có* $r_{j_{k+1}} = r_3 = 5$*, xét điều kiện nghiệm ta được*

$$\min_{l\in J'}(\max_{l\in J'}(t, r_l) + p_l) = 6 > r_{j_{k+1}},$$

vậy phương án $J_2^{(1)}$ *thoả điều kiện.*

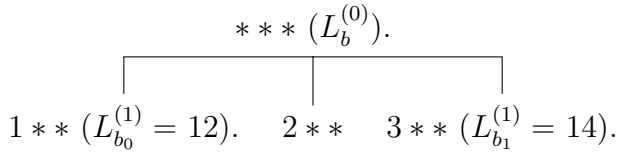
Ta giữ lại các phương án thoả điều kiện nghiệm, gồm phương án $J_0^{(1)}$*,* $J_2^{(1)}$*. Từ đây, ta tiếp tục xét cận dưới của các phương án.*

Ở phương án $J_0^{(1)}$*, áp dụng thuật toán EDD, ta thu được danh sách công việc là* $1 - 3 - 2$*. Tiếp tục áp dụng thuật toán không ngắt quãng, ta được* $C_{\max} = 12$ *(xem sơ đồ* (1.38)*), vậy ta thu được cận dưới* $L_{b_0}^{(1)} = 12$*. Tương tự với phương án* $J_1^{(1)}$*, ta thu được cận dưới* $L_{b_1}^{(1)} = 14$*.*

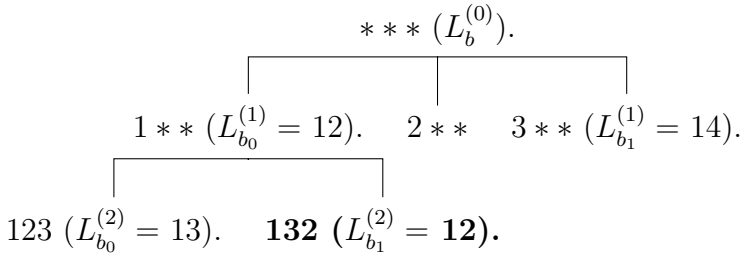
Từ đây, ta dễ dàng loại trừ phương án $J_1^{(1)}$ *do* $L_{b_0}^{(1)} < L_{b_1}^{(1)}$ *và tiếp tục phân nhánh với gốc nằm ở phương án* $J_0^{(1)}$. (1.33)

Thực hiện thuật toán tương tự, ta được $J_0^{(2)}$ *và* $J_1^{(2)}$ *đều thoả điều kiện nghiệm với cận dưới lần lượt là* $13, 12$. (1.34)

Vậy phương án tối ưu của bài toán là $1 - 3 - 2$.

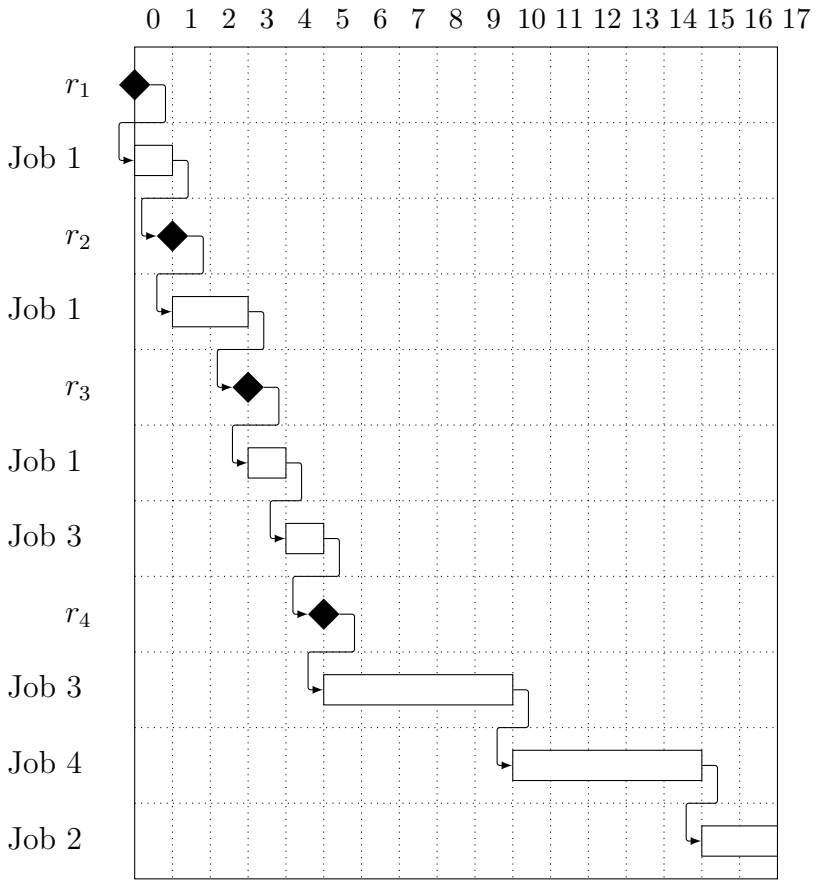


Hình 1.33: Sơ đồ cây cấp 1 với cận dưới.

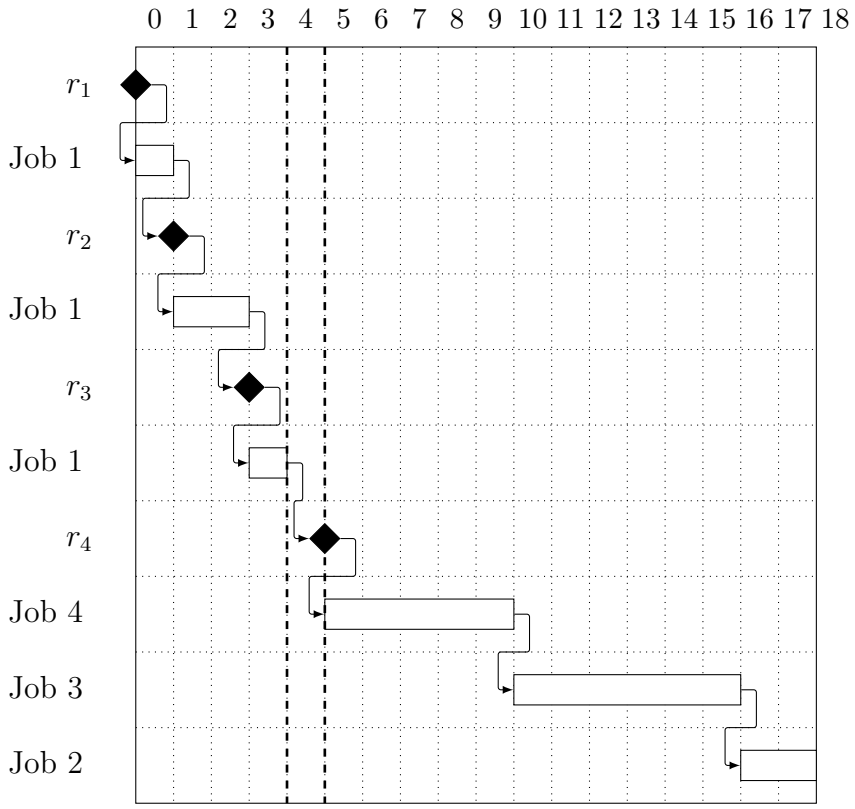


Hình 1.34: Sơ đồ cây cấp 2 với cận dưới.

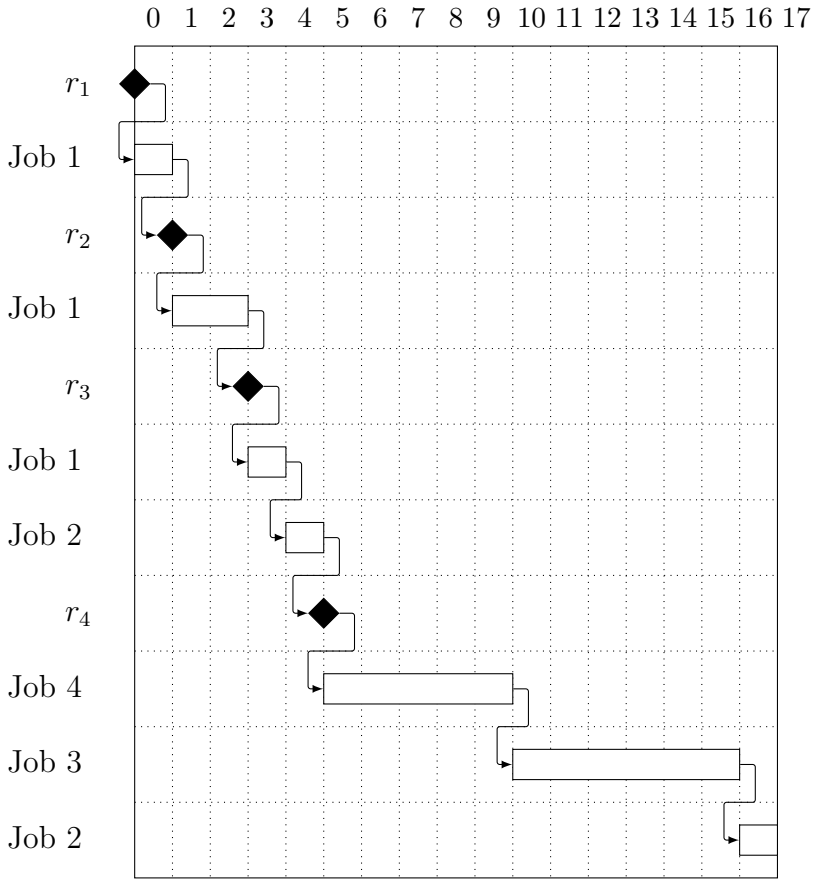
Trang chỉ mục / Phụ lục



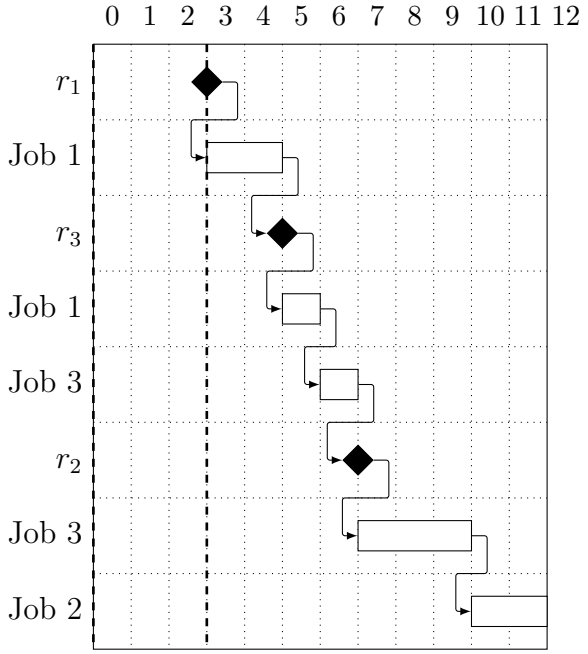
Hình 1.35: Sơ đồ Gantt không ngắt quãng của phương án $J_1^{(3)}$.



Hình 1.36: Sơ đồ Gantt không ngắt quãng của phương án $J_2^{(3)}$.



Hình 1.37: Sơ đồ Gantt ngắt quãng của phương án $J_2^{(3)}$.



Hình 1.38: Sơ đồ Gantt không ngắt quãng của phương án $J_0^{(1)}$.

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.