Phương pháp phân bổ nguồn lực cho nhiều dự án với thuật toán DLHS điều chỉnh

1 Giới thiệu

Trong nghiên cứu này, chúng tôi tập trung vào bài toán phân bổ nguồn lực cho nhiều dự án diễn ra đồng thời. Một trong những yêu cầu quan trọng của bài toán là **nhân viên không được làm việc trên nhiều dự án khác nhau trong cùng một ngày**, nhằm đảm bảo sự tập trung và hiệu quả. Bài toán này mở rộng từ vấn đề phân bổ nguồn lực cho một dự án duy nhất bằng cách tích hợp thêm các ràng buộc phức tạp liên quan đến quản lý đa dự án.

Mục tiêu chính là tối ưu hóa việc sử dụng tài sản, nhân lực và thời gian, đồng thời thỏa mãn các ràng buộc về **thời gian hoàn thành**, **KPI (Key Performance Indicators)** của từng dự án, và **chi phí vận hành**. Các đóng góp chính của nghiên cứu bao gồm:

- Điều chỉnh cấu trúc dữ liệu để hỗ trợ quản lý đồng thời nhiều dự án.
- Phát triển thuật toán lập lịch tổng hợp cho tất cả các công việc từ nhiều dự án.
- Mở rộng thuật toán DLHS (Dựa trên Harmony Search) để gán nhân lực với ràng buộc không làm việc đa dự án trong cùng ngày.
- Đề xuất cơ chế xử lý xung đột thời gian nhằm đảm bảo tính khả thi của lịch trình.

2 Cấu trúc dữ liệu

Để quản lý hiệu quả nhiều dự án, chúng tôi đã mở rộng cấu trúc dữ liệu như sau:

- **Công việc**: Mỗi công việc T_i được bổ sung trường `project_Id` để chỉ định dự án mà nó thuộc về. Ví dụ:

$$\begin{aligned}\{T_1 : \{&\text{project\_Id} : P1,\\ &\text{duration} : 2,\\ &\text{predecessors} : []\},\\ T_2 : \{&\text{project\_Id} : P2,\\ &\text{duration} : 3,\\ &\text{predecessors} : [T_1]\}\}\end{aligned}$$

- **Nhân viên và tài sản**: Trạng thái của nhân viên (E_i) và tài sản (A_i) được theo dõi trên tất cả các dự án, bao gồm thời gian làm việc và dự án liên quan. Ví dụ:

$$\{E_1 : \{\text{time} : [t1, t2], \text{project\_Id} : P1\}, A_1 : \{\text{time} : [t3, t4], \text{project\_Id} : P2\}\}$$

Cấu trúc này cho phép theo dõi và quản lý nguồn lực một cách linh hoạt giữa các dự án.

3 Sắp xếp và lập lịch

Quá trình sắp xếp và lập lịch là nền tảng để tổ chức các công việc trong nhiều dự án, đảm bảo chúng được thực hiện theo thứ tự hợp lý và tối ưu hóa việc sử dụng nguồn lực. Phần này bao gồm hai giai đoạn chính: **sắp xếp topo cho từng dự án** và **lập lịch tổng hợp cho tất cả các công việc**. Dưới đây là chi tiết của từng giai đoạn, kèm theo công thức và mã giả.

3.1 Sắp xếp topo cho từng dự án

Trong mỗi dự án, các công việc có thể có quan hệ phụ thuộc, nghĩa là một số công việc phải hoàn thành trước khi công việc khác có thể bắt đầu. Để giải quyết vấn đề này, chúng ta sử dụng **thuật toán sắp xếp topo (Topological Sort)** cho từng dự án dựa trên trường `project_Id`.

3.1.1 Lý do sử dụng sắp xếp topo

Sắp xếp topo được chọn vì: - **Đảm bảo thứ tự thực hiện hợp lệ**: Thuật toán tạo ra một thứ tự tuyến tính cho các công việc, trong đó mỗi công việc T_i chỉ bắt đầu sau khi tất cả các công việc tiền nhiệm T_i^p đã hoàn thành. - **Phát hiện vòng lặp**: Nếu có vòng lặp trong quan hệ phụ thuộc (ví dụ: $T_1 \to T_2 \to T_3 \to T_1$), thuật toán sẽ phát hiện và báo lỗi.

Quy trình thực hiện Quy trình sắp xếp topo cho từng dự án bao gồm các bước sau: 1. **Nhóm công việc theo dự án**: - Phân loại tất cả công việc $\{T_i\}$ theo `project_Id`. - Ví dụ:

$$\text{tasksByProject} = \{\text{P1} : [T_1, T_3, T_5], \text{P2} : [T_2, T_4, T_6]\}$$

2. **Áp dụng thuật toán sắp xếp topo**: - Với mỗi dự án P_j, thực hiện thuật toán topo: - **Bước 1**: Tính *indegree* (số công việc tiền nhiệm) cho mỗi công việc. - **Bước 2**: Khởi tạo một hàng đợi với các công việc có *indegree* = 0. - **Bước 3**: Lặp lại cho đến khi hàng đợi rỗng: - Lấy công việc từ hàng đợi, thêm vào danh sách thứ tự. - Giảm *indegree* của các công việc phụ thuộc vào nó. - Nếu *indegree* của công việc phụ thuộc giảm xuống 0, thêm vào hàng đợi. - **Bước 4**: Nếu không sắp xếp được tất cả công việc, báo lỗi do có vòng lặp. 3. **Kết quả**: - Một danh sách thứ tự công việc cho mỗi dự án, ví dụ:

$$\text{sortedTasks} = \{P1 : [T_1, T_3, T_5], P2 : [T_2, T_4, T_6]\}$$

Mã giả cho sắp xếp topo

Dưới đây là mã giả cho hàm ‘TopologicalSort‘ áp dụng cho một dự án:

```
function TopologicalSort(tasks)
    indegree ← tính indegree cho mỗi T_i trong tasks
    queue ← các T_i có indegree = 0
    result ← danh sách rỗng
    while queue không rỗng do
        T_i ← lấy công việc từ queue
        thêm T_i vào result
        for each T_j phụ thuộc vào T_i do
            indegree[T_j] ← indegree[T_j] - 1
            if indegree[T_j] = 0 then
                thêm T_j vào queue
    if số công việc trong result < số công việc trong tasks then
        return "Lỗi: Có vòng lặp trong dự án"
    return result
```

3.2 Lập lịch tổng hợp

Sau khi có thứ tự topo cho từng dự án, chúng ta gộp tất cả công việc từ các dự án và sử dụng **phương pháp đường găng (Critical Path Method - CPM)** để lập lịch tổng hợp. CPM giúp xác định thời gian thực hiện tối ưu và quản lý các ràng buộc giữa các dự án.

Lý do sử dụng CPM

CPM được chọn vì: - **Xác định thời gian khả thi**: Tính toán thời gian bắt đầu sớm nhất (ES), kết thúc sớm nhất (EF), bắt đầu muộn nhất (LS), và kết thúc muộn nhất (LF) cho mỗi công việc. - **Tối ưu hóa lịch trình**: Xác định đường găng và các công việc có thể thực hiện song song.

Quy trình thực hiện

Quy trình lập lịch tổng hợp bao gồm các bước sau: 1. **Gộp tất cả công việc**: - Hợp nhất các danh sách công việc đã sắp xếp topo từ tất cả dự án thành một danh sách lớn. - Ví dụ:

$$\text{allTasks} = [T_1, T_2, T_3, T_4, T_5, T_6]$$

2. **Tính toán ES và EF (Forward Pass)**: - Với mỗi công việc T_i: - **Thời gian bắt đầu sớm nhất (ES)**:

$$ES(T_i) = \begin{cases} \max_{T_j \in T_i^p} EF(T_j) & \text{nếu } T_i^p \neq \emptyset \\ P_j^s & \text{nếu } T_i^p = \emptyset \end{cases}$$

Trong đó P_j^s là thời gian bắt đầu của dự án P_j chứa T_i, T_i^p là tập hợp các công việc tiền nhiệm của T_i. - **Thời gian kết thúc sớm nhất (EF)**:

$$EF(T_i) = ES(T_i) + T_i^d$$

Trong đó T_i^d là thời gian thực hiện của công việc T_i. 3. **Tính toán LF và LS (Backward Pass)**: - Với mỗi công việc T_i: - **Thời gian kết thúc muộn nhất (LF)**:

$$LF(T_i) = \begin{cases} \min_{T_j \in T_i^{\text{successor}}} LS(T_j) & \text{nếu } T_i^{\text{successor}} \neq \emptyset \\ P_j^e & \text{nếu } T_i^{\text{successor}} = \emptyset \end{cases}$$

Trong đó $T_i^{\text{successor}}$ là các công việc tiếp theo T_i, P_j^e là thời gian kết thúc dự kiến của dự án P_j. - **Thời gian bắt đầu muộn nhất (LS)**:

$$LS(T_i) = LF(T_i) - T_i^d$$

4. **Xác định đường găng và độ trễ cho phép**: - **Đường găng**: Các công việc có $ES(T_i) = LS(T_i)$ và $EF(T_i) = LF(T_i)$. - **Độ trễ cho phép (slack)**:

$$\text{slack}(T_i) = LS(T_i) - ES(T_i)$$

Mã giả cho CPM

Dưới đây là mã giả cho hàm 'ApplyCPM' để lập lịch tổng hợp:

function ApplyCPM(allTasks)
    ▷ Forward Pass
    for each T_i in allTasks (theo thứ tự topo) **do**
        if T_i^p rỗng **then**
            $ES(T_i) \leftarrow P_j^s$ ▷ thời gian bắt đầu của dự án P_j chứa T_i
        else
            $ES(T_i) \leftarrow \max_{T_j \in T_i^p} EF(T_j)$
        $EF(T_i) \leftarrow ES(T_i) + T_i^d$
    ▷ Backward Pass
    for each T_i in allTasks (ngược thứ tự topo) **do**
        if $T_i^{\text{successor}}$ rỗng **then**
            $LF(T_i) \leftarrow P_j^e$ ▷ thời gian kết thúc của dự án P_j chứa T_i
        else
            $LF(T_i) \leftarrow \min_{T_j \in T_i^{\text{successor}}} LS(T_j)$
        $LS(T_i) \leftarrow LF(T_i) - T_i^d$
    ▷ Xác định đường găng và độ trễ
    for each T_i in allTasks **do**
        if $ES(T_i) = LS(T_i)$ và $EF(T_i) = LF(T_i)$ **then**
            đánh dấu T_i là công việc trên đường găng
        $\text{slack}(T_i) \leftarrow LS(T_i) - ES(T_i)$

3.3 Kết quả của quá trình sắp xếp và lập lịch

Sau khi hoàn tất, chúng ta thu được: - **Thứ tự thực hiện công việc**: Đảm bảo các công việc trong từng dự án được thực hiện theo thứ tự hợp lệ. - **Khung thời gian khả thi**: Các giá trị ES, EF, LS, LF giúp xác định khoảng thời gian linh hoạt cho mỗi công việc, đồng thời xác định các công việc quan trọng trên đường găng.

Những kết quả này là cơ sở cho việc gán tài nguyên và nhân lực trong các bước tiếp theo, đảm bảo lịch trình được xây dựng một cách logic và tối ưu.

3.4 Sắp xếp topo cho từng dự án

Đầu tiên, chúng tôi thực hiện **sắp xếp topo** (topological sort) cho từng dự án dựa trên trường `project_Id`. Điều này đảm bảo rằng các công việc tiền nhiệm (T_i^p) phải được

hoàn thành trước khi bắt đầu các công việc phụ thuộc. Kết quả là danh sách thứ tự công việc cho mỗi dự án:

$$\{P1 : [T_1, T_3, T_5], P2 : [T_2, T_4, T_6]\}$$

3.5 Lập lịch tổng hợp

Sau đó, tất cả công việc từ các dự án được gộp thành một tập hợp lớn. Chúng tôi áp dụng **phương pháp CPM (Critical Path Method)** để tính toán các thông số thời gian quan trọng:

- **ES (Earliest Start)**: Thời gian bắt đầu sớm nhất.
- **EF (Earliest Finish)**: Thời gian kết thúc sớm nhất.
- **LS (Latest Start)**: Thời gian bắt đầu muộn nhất.
- **LF (Latest Finish)**: Thời gian kết thúc muộn nhất.

Phương pháp này giúp xác định khung thời gian khả thi cho từng công việc, đồng thời xem xét các ràng buộc giữa các dự án.

4 Gán tài sản và khung thời gian

4.1 Gán tài sản

Chúng tôi duy trì một **lịch sử sử dụng tài sản** trên tất cả các dự án. Khi gán tài sản cho công việc T_i:

1. Kiểm tra thời gian khả dụng của tài sản dựa trên lịch sử.
2. Chọn tài sản phù hợp nhất (ít xung đột nhất).
3. Cập nhật trạng thái tài sản sau khi gán.

Ví dụ: Nếu A_1 trống từ t_5 đến t_7, nó sẽ được gán cho T_3 nếu khung thời gian phù hợp.

4.2 Gán khung thời gian

Thời gian bắt đầu (T_i^s) và kết thúc (T_i^e) của công việc T_i được tính toán dựa trên:

- Thời gian khả dụng của tài sản đã gán.
- Thời gian hoàn thành của các công việc tiền nhiệm trong cùng dự án.

Điều này đảm bảo không có vi phạm về thứ tự thực hiện hoặc sự chồng lấn tài nguyên.

5 Gán nhân lực

5.1 Phân chia KPI cho nhiều dự án

KPI mục tiêu được phân bổ cho từng dự án, ví dụ:

$$\{P1 : \text{KPI}_1 = 80\%, P2 : \text{KPI}_2 = 90\%\}$$

Sau đó, KPI của từng dự án được phân bổ cho nhân viên dựa trên năng lực của họ, với ràng buộc: **mỗi nhân viên chỉ làm việc cho một dự án duy nhất trong một ngày**.

5.2 Gán nhân lực bằng thuật toán DLHS

Chúng tôi sử dụng thuật toán **DLHS** (Dựa trên Harmony Search) để gán nhân lực:

- Một **vector hài hòa** đại diện cho phương án gán nhân lực cho tất cả công việc của tất cả dự án.
- Trong quá trình tạo phương án cải tiến, kiểm tra để đảm bảo không nhân viên nào làm việc trên nhiều dự án cùng ngày.

6 Xử lý xung đột thời gian

Sau khi gán nhân lực, chúng tôi kiểm tra lịch trình để phát hiện xung đột:

- Nếu nhân viên E_i được gán cho T_1 (dự án $P1$) và T_2 (dự án $P2$) trong cùng ngày, xung đột xảy ra.
- **Giải pháp**: Dời công việc của dự án ít ưu tiên hơn sang ngày tiếp theo. Cơ chế ưu tiên dựa trên:
  1. Độ trễ cho phép (slack time) của công việc.
  2. Tầm quan trọng của dự án (được định nghĩa trước).

7 Tối ưu hóa tổng thể

Chúng tôi định nghĩa một **hàm mục tiêu tổng thể** để đánh giá chất lượng phân bổ nguồn lực:

$$F = w_1 \cdot \text{Total Time} + w_2 \cdot \text{Total Cost} + w_3 \cdot \text{KPI Deviation} + w_4 \cdot \text{Conflicts}$$

Trong đó w_1, w_2, w_3, w_4 là các trọng số. Thuật toán DLHS được điều chỉnh để tối ưu hóa đồng thời cho nhiều dự án, sử dụng cơ chế trao đổi thông tin giữa các **sub-HM** (sub-Harmony Memories) nhằm cải thiện hiệu quả tìm kiếm.

8 Thuật toán điều chỉnh

Dưới đây là mã giả của thuật toán DLHS điều chỉnh:

9 Kết luận

Nghiên cứu này đã trình bày một phương pháp toàn diện để phân bổ nguồn lực cho nhiều dự án, với ràng buộc đặc biệt rằng nhân viên không làm việc trên nhiều dự án cùng ngày. Bằng cách điều chỉnh cấu trúc dữ liệu, thuật toán lập lịch, và thuật toán DLHS, chúng tôi đã phát triển một giải pháp hiệu quả, khả thi và tối ưu. Kết quả cho thấy phương pháp có thể xử lý các ràng buộc phức tạp và cung cấp lịch trình chất lượng cao. Trong tương lai, chúng tôi dự kiến khám phá việc tích hợp các yếu tố động như thay đổi tiến độ hoặc sự sẵn có của nguồn lực.

Algorithm 1 DLHSMultiProjects

Input: $T_i, E_i, A_i, KPI_{target}, DLHS_{Args}$
Output: Assignment of resources
Step 1: Sắp xếp topo cho từng dự án
for each P_j in projects **do**
    $sortedTasks[P_j] \leftarrow$ TopologicalSort($tasksByProject[P_j]$)
Step 2: Lập lịch tổng hợp
$allTasks \leftarrow$ Flatten($sortedTasks$)
Apply CPM on $allTasks$ to get ES, EF, LS, LF
Step 3 & 4: Gán tài sản và thời gian
for each T_i in $allTasks$ **do**
    Assign assets and calculate T_i^s, T_i^e
Step 5 & 6: Gán nhân lực với DLHS
Initialize harmony memory (HM)
while not converged **do**
    Generate new solutions and update HM
Step 7: Xử lý xung đột
for each E_i in E_i **do**
    Resolve conflicts across projects
Return: Final assignment