ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA KHOA HỌC MÁY TÍNH
----------o0o----------

TỐI ƯU LẬP KẾ HOẠCH

Topic 9

Container 2D Loading Minimize Cost

GVHD: TS. Bùi Quốc Trung

Họ tên	**MSSV**	**Email**
Đàm Việt Anh	20204627	anh.dv204627@sis.hust.edu.vn
Bùi Lâm Thanh	20204606	thanh.bl204606@sis.hust.edu.vn
Nguyễn Hoàng Ban	20204514	ban.nh204514@sis.hust.edu.vn

TP. HÀ NỘI, THÁNG 7 NĂM 2023

LỜI CẢM ƠN

Trong thời gian tham gia học phần môn học Tối ưu lập kế hoạch và hoàn thành đồ án môn học, chúng em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – Thầy Bùi Quốc Trung đã nhiệt tình giảng dạy và truyền đạt những kiến thức quý báu trong cả những tiết học ở trường và ngoài giờ, giúp đỡ em trong suốt thời gian học tập vừa qua để nhóm em có thể hoàn thành bài tập lớn môn học này một cách tốt nhất có thể. Bài tập lớn môn Tối ưu lập kế hoạch với đề tài số 9 "Container 2D Loading Minimize Cost" là kết quả của quá trình tìm tòi, trau dồi của nhóm em. Trong quá trình hoàn thành báo cáo, nhóm em chắc chắn không tránh khỏi những thiếu sót và nhóm rất mong nhận được những lời nhận xét từ thầy để bài báo cáo được hoàn thiện hơn. Cuối cùng, nhóm em xin kính chúc thầy có thật nhiều sức khỏe, hạnh phúc và luôn luôn thành công trong sự nghiệp giảng dạy của mình.

PHÂN CÔNG CÔNG VIỆC

Dưới đây là các công việc chính các thành viên nhóm được phân chia, ngoài ra trong quá trình hoàn thiện đồ án môn học còn tồn tại những việc phát sinh cần các thành viên nhóm trao đổi lại với nhau nên không đề cập cụ thể.

Họ tên	**MSSV**	**Công việc chính**	**Đánh giá**
Đàm Việt Anh	20204627	– Mô hình hóa bài toán và viết mã Mixed Integer Programming. – Tìm hiểu và viết mã Guillotine Algorithm. – Đóng góp báo cáo. – Đóng góp slide.	10/10
Bùi Lâm Thanh	20204606	– Mô hình hóa bài toán và viết mã Constraint Programming. – Tìm hiểu và viết mã Maximal Rectangles Algorithm. – Đóng góp báo cáo. – Đóng góp slide.	10/10
Nguyễn Hoàng Ban	20204514	– Viết mã Brute force.	3/10

MỤC LỤC

1. GIỚI THIỆU

1.1 Mô tả bài toán

Container 2D Loading Minimize Cost là một bài toán tối ưu quen thuộc trong khoa học máy tính. Bài toán liên quan đến việc xếp những mặt hàng lên các thùng chứa sao cho tối ưu được không gian sử dụng và tối thiểu số thùng chứa cần dùng nhất có thể. Bài toán được biến đến là NP khó, tức là không có thuật toán hiệu quả để tìm ra lời giải tối ưu, do đó rất nhiều các giải thuật tham lam hay heuristic được phát triển để tìm nghiệm xấp xỉ nghiệm tối ưu. Cụ thể, bài toán được mô tả như sau:

> Có K xe tải dùng để vận chuyển N mặt hàng, trong đó xe tải và mặt hàng là các hình chữ nhật 2 chiều. Xe tải thứ k có kích thước là $W[k] \times L[k]$, với chi phí sử dụng xe tải này là $C[k]$. Mặt hàng thứ i có kích thước là $w[i] \times l[i]$. Chúng ta cần xếp N mặt hàng lên trên K xe tải với điều kiện là các mặt hàng không được chồng lấn lên nhau. Mục tiêu của bài toán là tìm cách xếp sao cho chi phí sử dụng là ít nhất có thể.

1.2 Định dạng đầu vào và đầu ra

Đầu vào của bài toán:

- Dòng 1: gồm 2 số nguyên N và K, lần lượt thể hiện số lượng mặt hàng và số lượng xe tải. $(1 \leq N,\ K \leq 1000)$.
- Dòng thứ $i + 1$ $(i = \overline{1..N})$: mỗi dòng chứa 2 số nguyên w[i] và l[i], thể hiện chiều rộng và chiều cao của mặt hàng thứ i. $(1 \leq w[i], l[i] \leq 1000)$.
- Dòng thứ $N + 1 + k$ $(k = \overline{1..K})$: mỗi dòng chứa 3 số nguyên W[k], L[k] và C[k], thể hiện chiều rộng, chiều cao và chi phí sử dụng của xe tải thứ k. $(1 \leq W[k], L[k] \leq 1000, 1 \leq C[k] \leq 1000)$.

Đầu ra của bài toán:

- Dòng thứ i $(i = \overline{1..N})$: mỗi dòng chứa 5 số nguyên i, t[i], x[i], y[i] và o[i], lần lượt thể hiện:
  + i: mặt hàng thứ i
  + t[i]: mặt hàng này được xếp vào xe tải thứ t[i].
  + x[i], y[i]: mặt hàng này có tọa độ góc trái dưới là (x[i], y[i]) khi đặt lên xe tải thứ t[i].

+ o[i]: mặt hàng này có phải xoay 90^o hay không. $o[i] = 1$ nghĩa là mặt hàng này có xoay 90^o, ngược lại $o[i] = 0$ là mặt hàng này không xoay.

2. PHƯƠNG PHÁP

Từ mô tả bài toán, chúng ta có thể thấy 3 ràng buộc chính như sau:

- Một mặt hàng chỉ có thể được xếp lên một xe tải.
- Các mặt hàng trên cùng một xe tải không được chồng lấn lên nhau.
- Các mặt hàng phải được xếp không được vượt ra ngoài xe tải (mặt hàng nằm hoàn toàn bên trong xe tải).

Các ràng buộc này sẽ được cụ thể hóa trong mô hình hóa của các phương pháp giải dưới đây. Trước hết là các hằng số chung có thể thấy:

- n_items: số lượng mặt hàng.
- n_trucks: số lượng xe tải.
- items[n_items][2]: thông tin của từng mặt hàng, trong đó items[i] lưu trữ thông tin của mặt hàng thứ i, bao gồm items[i][0] là w[i] và items[i][1] là l[i] (trong quá trình viết cũng sẽ viết là w[i] và l[i]).
- trucks[n_trucks][3]: thông tin của từng xe tải, trong đó trucks[k] lưu trữ thông tin của xe tải thứ k, bao gồm trucks[k][0] là W[k], trucks[k][1] là L[k] và trucks[k][2] là C[k] (trong quá trình viết cũng sẽ viết là W[k], L[k], C[k]).

2.1 Brute force

Các biến

- t_i: i-th item is placed in t_i-th truck, $i = \overline{0..(N-1)}$, $t_i = \overline{0..(K-1)}$
- (x_i, y_i): i-th item is placed in t_i-th truck at the position (x_i, y_i): , $i = \overline{0..(N-1)}$
- o_i: i-th item is rotated, $i = \overline{0..(N-1)}$, $o_i \in \{0; 1\}$
  - $o_i = 0$, item is not rotated
  - $o_i = 1$, item is rotated
- f: store the current cost of solution
- f_best: store the best solution
- $t\_solution_i$: store the best solution
- $(x\_solution_i, y\_solution_i)$: store the best solution
- $o\_solution_i$: store the best solution

Các ràng buộc:

- A item is only placed in one truck
- Item cannot be placed out of the truck
  - + $if\ t_i = k\ and\ o_i = 0\ (\forall i = \overline{0..(N-1)}, k = \overline{0..(K-1)})$:
    - $x_i + w_i \leq W_k$
    - $y_i + h_i \leq H_k$
    - $x_i \geq 0$
    - $y \geq 0$
  - + $if\ t_i = k\ and\ o_i = 1\ (\forall i = \overline{0..(N-1)}, k = \overline{0..(K-1)})$:
    - $x_i + h_i \leq W_k$
    - $y_i + w_i \leq H_k$
    - $x_i \geq 0$
    - $y \geq 0$
- Items placed in truck cannot be overlapped

$$x[i] + w[i] * (1 - R[i]) + l[i] * R[i] \leq x[j]$$

$$y[i] + w[i] * R[i] + l[i] * (1 - R[i]) \leq y[j]$$

$$x[j] + w[j] * (1 - R[j]) + l[j] * R[j] \leq x[i]$$

$$y[j] + w[j] * R[j] + l[j] * (1 - R[j]) \leq y[i]$$

2.2 Constraint Programming

Các biến:

- T[i][j]: thể hiện mặt hàng thứ i có được đặt trên xe tải thứ j hay không (1 nếu có và 0 nếu không)
- R[i]: thể hiện mặt hàng thứ i có xoay hay không (1 nếu có và 0 nếu không)
- x[i]: tọa độ theo trục hoành của mặt hàng thứ i
- y[i]: tọa độ theo trục tung của mặt hàng thứ i
- Z[j]: thể hiện xe tải thứ j có được sử dụng hay không (1 nếu có và 0 nếu không)

Các ràng buộc:

Mỗi mặt hàng chỉ được đặt trong một xe tải

$$\sum_{j=0}^{n\_trucks-1} T[i][j] = 1, \forall i \in [0, n\_items - 1]$$

Các mặt hàng không thể đặt ra ngoài xe tải

$$x[i] \leq W[j] - w[i] * (1 - R[i]) - l[i] * R[i]$$

$$y[i] \leq L[j] - w[i] * R[i] - l[i] * (1 - R[i])$$

Các mặt hàng trên cùng một xe tải không được chồng lấn lên nhau (chỉ cần 1 trong 4 điều kiện được thỏa mãn):

$$x[i] + w[i] * (1 - R[i]) + l[i] * R[i] \leq x[j]$$

$$y[i] + w[i] * R[i] + l[i] * (1 - R[i]) \leq y[j]$$

$$x[j] + w[j] * (1 - R[j]) + l[j] * R[j] \leq x[i]$$

$$y[j] + w[j] * R[j] + l[j] * (1 - R[j]) \leq y[i]$$

Điều kiện xe tải được sử dụng

$$Z[k] \leq \sum_{i=0}^{n\_items-1} T[i][k]$$

$$\sum_{i=0}^{n\_items-1} T[i][k] \leq Z[k] * n\_items$$

Tối thiểu hóa hàm mục tiêu:

$$\sum_{k=0}^{n\_trucks-1} Z[k] * C[k]$$

2.3 Mixed Integer Programming

Các biến:

- T[i][j]: thể hiện mặt hàng thứ i có được đặt trên xe tải thứ j hay không (1 nếu có và 0 nếu không)
- R[i]: thể hiện mặt hàng thứ i có xoay hay không (1 nếu có và 0 nếu không)
- left[i]: tọa độ theo trục hoành cạnh trái của mặt hàng thứ i
- right[i]: tọa độ theo trục hoành cạnh phải mặt hàng thứ i
- top[i]: tọa độ theo trục tung cạnh trên của mặt hàng thứ i
- bottom[i]: tọa độ theo trục tung cạnh dưới của mặt hàng thứ i

- Z[j]: thể hiện xe tải thứ j có được sử dụng hay không (1 nếu có và 0 nếu không)

Các ràng buộc:

Mỗi mặt hàng chỉ được đặt trong một xe tải

$$\sum_{j=0}^{n\_trucks-1} T[i][j] = 1, \forall i \in [0, n\_items - 1]$$

Ràng buộc về các cạnh của mặt hàng

$$right[i] \leq left[i] + w[i] * (1 - R[i]) + l[i] * R[i]$$

$$top[i] \leq bottom[i] + w[i] * R[i] + l[i] * (1 - R[i])$$

Các mặt hàng không thể đặt ra ngoài xe tải

$$right[i] \leq W[j] + max\_truck\_width * (1 - T[i][j])$$

$$left[i] \leq W[j] + max\_truck\_width * (1 - T[i][j])$$

$$top[i] \leq L[j] + max\_truck\_height * (1 - T[i][j])$$

$$bottom[i] \leq L[j] + max\_truck\_height * (1 - T[i][j])$$

Các mặt hàng trên cùng một xe tải không được chồng lấn lên nhau (chỉ cần 1 trong 4 điều kiện được thỏa mãn):

$$right[i] \leq left[j]$$

$$top[i] \leq bottom[j]$$

$$right[j] \leq left[i]$$

$$top[j] \leq top[i]$$

Điều kiện xe tải được sử dụng

$$Z[k] \leq \sum_{i=0}^{n\_items-1} T[i][k]$$

$$\sum_{i=0}^{n\_items-1} T[i][k] \leq Z[k] * n\_items$$

Tối thiểu hóa hàm mục tiêu:

$$\sum_{k=0}^{n\_trucks-1} Z[k] * C[k]$$

2.4 Guillotine Algorithm

Ý tưởng thuật toán:

- Mỗi một xe tải sẽ lưu thêm danh sách các hình chữ nhật còn trống trên xe tải đó.
- Mỗi khi thêm một mặt hàng mới vào xe tải, ta lựa chọn các hình chữ nhật còn trống trên xe để, sau đó đặt mặt hàng mới này vào góc trái dưới lên hình chữ nhật phù hợp nhất.
- Sau khi đặt xong mặt hàng, ta thực hiện chia hình chữ nhật vừa đặt ra làm các hình chữ nhật còn trống mới, sau đó lưu lại vào danh sách các hình chữ nhật còn trống.
- Thực hiện gộp các hình chữ nhật còn trống để tạo ra các hình chữ nhật còn trống lớn hơn nếu có thể.
- Lặp lại cho đến khi xếp được hết các mặt hàng lên các xe tải.

Algorithm 1: The Guillotine algorithm

Set $F = \{(W, H)\}$;
foreach *item* $i = (w, h)$ *in the list of inserted items of the bin* **do**
  Decide the free rectangle $F_j \in F$ to pack the item into;
  Decide the orientation for the item and place it at the bottom-left of F_j;
  Use the guillotine split scheme to subdivide F_j into two new free rectangles F_{j1} and F_{j2};
  Set $F \leftarrow (F \cup \{F_{j1}, F_{j2}\}) \setminus \{F_j\}$;
  foreach *ordered pair of free rectangles* F_{j1} *and* F_{j2} *in* F **do**
    if F_{j1} *and* F_{j2} *can be merged together* **then**
      $F_{merge} \leftarrow$ Merge F_{j1} and F_{j2};
      Set $F \leftarrow (F \cup \{F_{merge}\}) \setminus \{F_{j1}, F_{j2}\}$;
    end
  end
end

Ngoài ra, nhóm thực hiện việc lựa chọn thứ tự ưu tiên xe tải, ưu tiên các hình chữ nhật còn trống và ưu tiên mặt hàng như sau:

- Các xe tải ưu tiên theo tỷ lệ diện tích / chi phí, nếu tỷ lệ này bằng nhau thì ưu tiên xe tải có kích thước chiều cao, chiều rộng đều lớn và dương đối với nhau (trung bình điều hòa của chiều rộng và chiều cao lớn hơn).
- Các mặt hàng ưu tiên theo chiều cao, nếu chiều cao bằng nhau thì ưu tiên chiều rộng lớn hơn
- Các hình chữ nhật còn trống ưu tiên vừa với mặt hàng nhất, tức là phần diện tích còn thừa là ít nhất.

2.5 Maximal Rectangles Algorithm (MRA)

Ý tưởng thuật toán:

- MRA tương tự ý tưởng với thuật toán Guillotine, chỉ khác ở ý tưởng chia lại hình chữ nhật còn trống
- Thay vì chọn một trong hai trục để chia như trong thuật toán Guillotine, Maximal Rectangles chọn cả hai trục phân chia cùng lúc để đảm bảo rằng diện tích các hình chữ nhật tự do là lớn nhất có thể.
- Bởi các hình chữ nhật tự do không còn rời rạc như trong thuật toán Guillotine, nên bất kì hình chữ nhật tự do nào giao với khu vực chèn item sẽ bị chia ra để loại bỏ phần giao nhau.
- Xóa bỏ các hình chữ nhật bị trùng lặp hoàn toàn với các hình chữ nhật khác trong danh sách.

Algorithm 2: The Maximal Rectangles algorithm

Set $F = \{(W, H)\}$;
foreach *item* $i = (w, h)$ *in the list of inserted items of the bin* **do**
  Decide the free rectangle $F_j \in F$ to pack the item into;
  Decide the orientation for the item and place it at the bottom-left of F_j;
  Use the max_rec split scheme to subdivide F_j into two new free rectangles F_{j1} and F_{j2};
  Set $F \leftarrow (F \cup \{F_{j1}, F_{j2}\}) \setminus \{F_j\}$;
  foreach *free rectangle* F_j *in* F **do**
    Compute $F_j \setminus i$ and subdivide the result into at most four new free rectangles $F_{j1}, \ldots, F_{j4}$;
    Set $F \leftarrow (F \cup \{F_{j1}, \ldots, F_{j4}\}) \setminus \{F_j\}$;
  end
  foreach *ordered pair of free rectangles* F_{j1} *and* F_{j2} *in* F **do**
    if F_{j1} *contains* F_{j2} **then**
      Set $F \leftarrow F \setminus \{F_{j2}\}$;
    end
  end
end

3. KẾT QUẢ THỰC NGHIỆM

Tên phương pháp	Viết tắt
Brute force	BF
Constraint Programming	CP
Mixed Integer Programming	MIP
Guillotine Algorithm	GA
Maximal Rectangles Algorithm	MRA

3.1 Độ chính xác trên Open ERP

Tên phương pháp	Kết quả trên Open ERP
BF	100/1000
CP	100/1000
MIP	100/1000
GA	1000/1000
MRA	1000/1000

3.2 Thời gian chạy

Kích thước test case (n_items / n_trucks). Dấu X là thuật toán không cho ra kết quả trong vòng 10 phút. Kết quả của thuật toán là lấy trung bình trong 5 lần chạy.

Kích thước	BF		CP		MIP		GA		MRA	
5 5	2.937	8	3.179	8	0.397	8	1.23	8	0.929	8
10 10	X	X	X	X	X	X	0.957	14	1.360	14
100 100	X	X	X	X	X	X	1.464	105	1.391	105
200 200	X	X	X	X	X	X	1.006	201	1.187	201
500 500	X	X	X	X	X	X	1.346	532	1.148	548
600 600	X	X	X	X	X	X	1.517	565	1.455	565
700 700	X	X	X	X	X	X	1.414	554	1.353	554
800 800	X	X	X	X	X	X	1.417	782	1.213	782
900 900	X	X	X	X	X	X	1.404	863	1.282	863
1000 1000	X	X	X	X	X	X	1.307	773	1.455	773

4. TỔNG KẾT

Các thuật toán dựa trên vét cạn trường hợp (BruteForce, CP hay MIP) không thể chạy được với các case lớn (số lượng xe tải và mặt hàng nhiều) do phải thử cả tọa độ của mặt hàng bên trong xe (trong khi kích thước xe tải đều lên đến hàng trăm).

Các thuật toán tham lam (Guillotine và Maximal Rectangle Algorithm) chạy được với các case lớn trên OpenERP và cho kết quả tối ưu hơn rất nhiều (so với kết quả hiện có trên OpenERP). Ngoài ra, các case của cả 2 thuật toán đều cho ra kết quả tương đương nhau ở các case, thời gian chạy của Maximal Rectangle Algorithm có vẻ nhỉnh hơn đa số). Có 1 case với kich thước 500 500 là Maximal Rectangle Algorithm trả ra kết quả tệ hơn so với Guillotine. Mặc dù là 2 thuật toán khác nhau nhưng đều dựa trên việc lưu trữ không gian còn lại (các hình chữ nhật còn lại) để nhanh chóng quyết định tọa độ đặt hàng, đồng thời một khi đã đặt hàng vào xe tải nào thì sẽ không thử lại với mặt hàng ý nữa, do đó cả 2 thuật toán cho kết quả gần như tương đương về giá trị tối ưu.

5. TÀI LIỆU THAM KHẢO

[1] https://docplayer.net/21175043-A-thousand-ways-to-pack-the-bin-a-practical-approach-to-two-dimensional-rectangle-bin-packing.html

[2] Slide môn học Tối ưu lập kế hoạch – TS. Bùi Quốc Trung