**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**BỘ MÔN KHOA HỌC MÁY TÍNH**

---------------o0o---------------

****

**ĐỒ ÁN MÔN HỌC**

**[TÌM KIẾM CỤC BỘ DỰA TRÊN RÀNG BUỘC]**

**Nhóm 8**

**GVHD: TS. Phạm Quang Dũng**

**SVTH: Lê Hội Quang-20156304**

**Phạm Văn Duy-20150631**

**TP. HÀ NỘI, THÁNG 5 NĂM 2019**

***LỜI CẢM ƠN***

***Qua môn học Tìm kiếm cục bộ dựa trên ràng buộc và sự giảng dạy của thầy Phạm Quang Dũng cũng như sự giúp đỡ của các bạn, chúng em đã tiếp cận được với một lĩnh vực mới bổ ích để giải quyết các dạng toán tổ hợp, các bài toán đã gặp hay chưa gặp. Hiểu được sự quan trọng và thú vị của môn học này. Vận dụng điều dó, chúng em đã thực hiện miniproject với đầu bài MultiKnapsack và hoàn thành. Qua đó chúng em xin chân thành cảm ơn Thầy Phạm Quang Dũng cũng như các bạn đã hỗ trợ rất nhiều trong suốt thời gian thực hiện bài tập này !***

*Tp. Hà Nội, ngày 7 tháng 5 năm 2019 .*

**Sinh viên**

***Lê Hội Quang và Phạm Văn Duy***

**TÓM TẮT MINIPROJECT**

Đồ án này trình bày về bài toán **MultiKnapsack with MinMaxType Constraints** các hướng ý tưởng để giải quyết các ràng buộc của bài toán đặt ra để tìm ra lời giải. Từ một số nền tảng được thầy giới thiệu trong lĩnh vực “Tìm kiếm cục bộ dựa trên ràng buộc”, chúng em đưa ra 2 phương án đổ mô hình hóa bài toán và giải quyêt nó. Đưa ra đánh giá và những vấn đề còn tồn tại với kết quả đạt được khi kiểm thử với các tập dữ liệu nhỏ, vừa và lớn.

Mục lục

[1. Giới thiệu bài toán 4](#_Toc8156225)

[2. Lý thuyết 5](#_Toc8156226)

[3. Các phương án giải quyết bài toán 6](#_Toc8156227)

[3.1 Phương án 1 6](#_Toc8156228)

[3.2 Phương án 2 7](#_Toc8156229)

[3.3 Phương án 3 8](#_Toc8156230)

[3.3 Phương án 4 8](#_Toc8156231)

[4. Kết quả thử nghiệm 8](#_Toc8156232)

[5. Kết luận và hướng phát triển 8](#_Toc8156233)

[6. Tài liệu tham khảo 8](#_Toc8156234)

[7. Phụ lục 9](#_Toc8156235)

1. Giới thiệu bài toán

Có N items cần được xếp vào M bins:

**Item** (i =1,...N)

w[i]: trọng số 1

p[i]: trọng số 2

t[i]: thể loại

r[i]: lớp

D[i]: Tập các bins mà item có thể xếp vào

**Bin** (b = 1….M)

W[b]: Sức chứa 1 (tải tối đa cho trọng số 1)

LW[b]: Tải tối thiểu cho trọng số 1

P[b]: sức chứa 2 (Tải tối đa cho trọng số 2)

T[b]: Số lượng thể loại tối đa cho các items trong bin

R[b]: Số lượng lớp tối đa cho các items trong bin

**Ràng buộc:**

Mỗi bin b có các ràng buộc:

**C1:** Tổng trọng số 1 của các items được xếp vào b phải lớn hơn hoặc bằng LW[b] và nhỏ hơn hoặc bằng W[b]

**C2:** Tổng trọng số 2 của các items được xếp vào b phải nhỏ hơn hoặc bằng P[b]

**C3:** Tổng số thể loại của các items được xếp vào b phải nhỏ hơn hoặc bằng T[b]

**C4:** Tổng số lớp của các items được xếp vào b phải nhỏ hơn hoặc bằng R[b]

2. Lý thuyết

Bài toán tối ưu tổ hợp xuất hiện rất nhiều trong các lĩnh vực của đời sống xã hội đặc biệt là trong các lĩnh vực quản lý, điều hành, lập kế hoạch, ra quyết định. Bài toán tối ưu tổ hợp đưuọc mô hình háo bởi một bộ COP={X,D,C,f} trong đó:

* Tập các biến
* Tập các miền giá trị , trong đó Di là miền giá trị của biến Xi, 
* Tập các ràng buộc của bài toán, mỗi ràng buộc thể hiện mối quan hệ giữa một số biến tham gia vào ràng buộc
* F: hàm mục tiêu cần được cực tiểu hóa

Trong nhiều tường hợp, bài toán đặt ra không có hàm mục giêu f cần tối ưu mà chỉ có ràng buộc. Khi đó, bài toán còn gọi là bài toán thỏa mãn ràng buộc.

Bài toán ***MultiKnapsack with MinMaxType Constraints*** là một bài toán tìm kiếm kết quả thỏa mãn ràng buộc, không có hàm mục tiêu.

3. Các phương án giải quyết bài toán

3.1 Phương án 1

Mô hình toán học: Sử dụng với LocalSearch mô hình biến X hai chiều.

* **Biến:**
  + X[i][j] với ,  ; D(X) = {0, 1};

Nếu Item i được xếp vào Bin j thì X[i][j] = 1, ngược lại X[i][j] = 0.

* + T[i]][j], ,  ; D(T) = {0, 1};

Nếu Bin j chứa Item có type = i thì T[i][j] = 1, ngược lại T[i][j] = 0.

* + R[i][j], ,  ; D(R) = {0, 1};

Nếu Bin j chứa Item có class = i thì R[i][j] = 1, ngược lại R[i][j] = 0.

* **Ràng buộc:**
  + Mỗi Item chỉ được xếp vào 1 Bin

,

* + Item phải được xếp tập binIndices

X[i][j] = 0 ; j Item[i].getBinIndices().

* + Tổng trọng số 1 phải lớn hơn MinW và bé hơn MaxW

MinW[j] <= <= MaxW[j], 

* + Tổng trọng số 2 phải bé hơn MaxP

<= MaxP[j], 

* + Tổng số Type trong Bin phải bé hơn MaxType

X[i][j] =1 suy ra T[item[i].getT][j] = 1 , ,

<= MaxType[j], 

* + Tổng số Class trong Bin phải bé hơn MaxBin

X[i][j] =1 suy ra R[item[i].getR][j] = 1 , ,

<= MaxClass[j], 

* **Ý tưởng thuật toán:**
* Sử dụng biến mảng 2 chiều X[i][j] để biểu diễn Item i có được xếp vào Bin j không.
* Sử dụng thêm biến VarIntLS[][] T, VarIntLS[][] R biểu diễn ràng buộc C3 và C4.
* Sau đó tìm kiếm với HillClimbing hoặc TabuSearch.

3.2 Phương án 2

Mô hình toán học: Sử dụng với LocalSearch vơi mô hình biến X một chiều.

* **Biến:**
  + X[i] với ,

Biểu diễn Item i được xếp vào Bin X[i].

* + T[i]][j], ,  ; D(T) = {0, 1};

Nếu Bin j chứa Item có type = i thì T[i][j] = 1, ngược lại T[i][j] = 0.

* + R[i][j], ,  ; D(R) = {0, 1};

Nếu Bin j chứa Item có class = i thì R[i][j] = 1, ngược lại R[i][j] = 0.

* **Ràng buộc:**
  + Item phải được xếp tập binIndices

; với K = binIndices.length,

* + Tổng trọng số 1 phải lớn hơn MinW và bé hơn MaxW

MinW[j] <= <= MaxW[j], 

* + Tổng trọng số 2 phải bé hơn MaxP

<= MaxP[j],

* + Tổng số Type trong Bin phải bé hơn MaxType

X[i] =j suy ra T[item[i].getT][j] = 1 , ,

<= MaxType[j], 

* + Tổng số Class trong Bin phải bé hơn MaxBin

X[i] =j suy ra R[item[i].getR][j] = 1 , ,

<= MaxClass[j], 

* **Ý tưởng thuật toán**
  + Sử dụng mảng 1 chiều X[i] biểu diễn Item i được xếp vào Bin [i].
  + Sử dụng thêm biến VarIntLS[][] T, VarIntLS[][] R biểu diễn ràng buộc C3 và C4.
  + Sau đó tìm kiếm với HillClimbing hoặc TabuSearch.

3.3 Phương án 3

Mô hình hóa bài toán tương tự như phương án 1 nhưng sử dụng với Choco.

3.3 Phương án 4

* + Ý tưởng: Dùng thuật toán tham sắp xếp lần lượt các Item vào Bin nào đó mà không bị vi phạm ràng buộc (ở đây bỏ qua ràng buộc Tổng trọng số 1 của các items được xếp vào b phải lớn hơn hoặc bằng LW[b] ).
  + Mã giả:

For ( Item i = 1 to N ) do

For ( binIndices j = 1 to binIndices.lenght ) do

Item[i] = binIndices[j];

If ( violations > 0) then item[i] = -1

Else break;

4. Kết quả thử nghiệm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Choco | LocalSearch1 | LocalSearch2 | Greedy |
| 16 items – 3 bins | - Kết quả: tối ưu  - Thời gian:458ms | - Kết quả: tối ưu  - Thời gian: 351ms | - Kết quả: tối ưu  - Thời gian: 203ms | - Kết quả: tối ưu  - Thời gian: |
| 100 items – 1846 bins | - Kết quả: không có  - Thời gian: rất lâu | - Kết quả: tràn bộ nhớ  - Thời gian: không có | - Kết quả: không có  - Thời gian: rất lâu (> 10h) | - Kết quả:  - Thời gian: |
| 1000 items – 1846 bins | - Kết quả: không có  - Thời gian: rất lâu | - Kết quả: tràn bộ nhớ  - Thời gian: | - Kết quả: không có  - Thời gian: rất lâu | - Kết quả:  - Thời gian: |
| 3000 items – 1846 bins | - Kết quả: không có  - Thời gian: rất lâu | - Kết quả:tràn bộ nhớ  - Thời gian: | - Kết quả: tràn bộ nhớ  - Thời gian: | - Kết quả:  - Thời gian: |

5. Kết luận và hướng phát triển

Ta có thể thấy rằng mô hình hai chiều cho bài toán này là không hiệu quả với dữ liệu lớn.

* LocalSearch mô hình biến một chiều cho ta hiệu quả hơn về bộ nhớ
* Đối với thuật toán tham lam cho ta hiệu quả về mặt thời gian rất lớn, khác biệt so với các phương án còn lại. Kết quả nhận được là khả quan.

6. Tài liệu tham khảo

<http://www.choco-solver.org/>

7. Phụ lục