Tối ưu Lập kế hoạch

Graph partitioning Problem

Giảng viên hướng dẫn: TS.Bùi Quốc Trung
Nguyễn Tiến Long - 20180129
Nguyễn Đức Long - 20183583
Nguyễn Thành Long - 20183585

Trường ĐH CNTT&TT-ĐHBKHN

Ngày 4 Tháng 1 Năm 2022

Mục lục

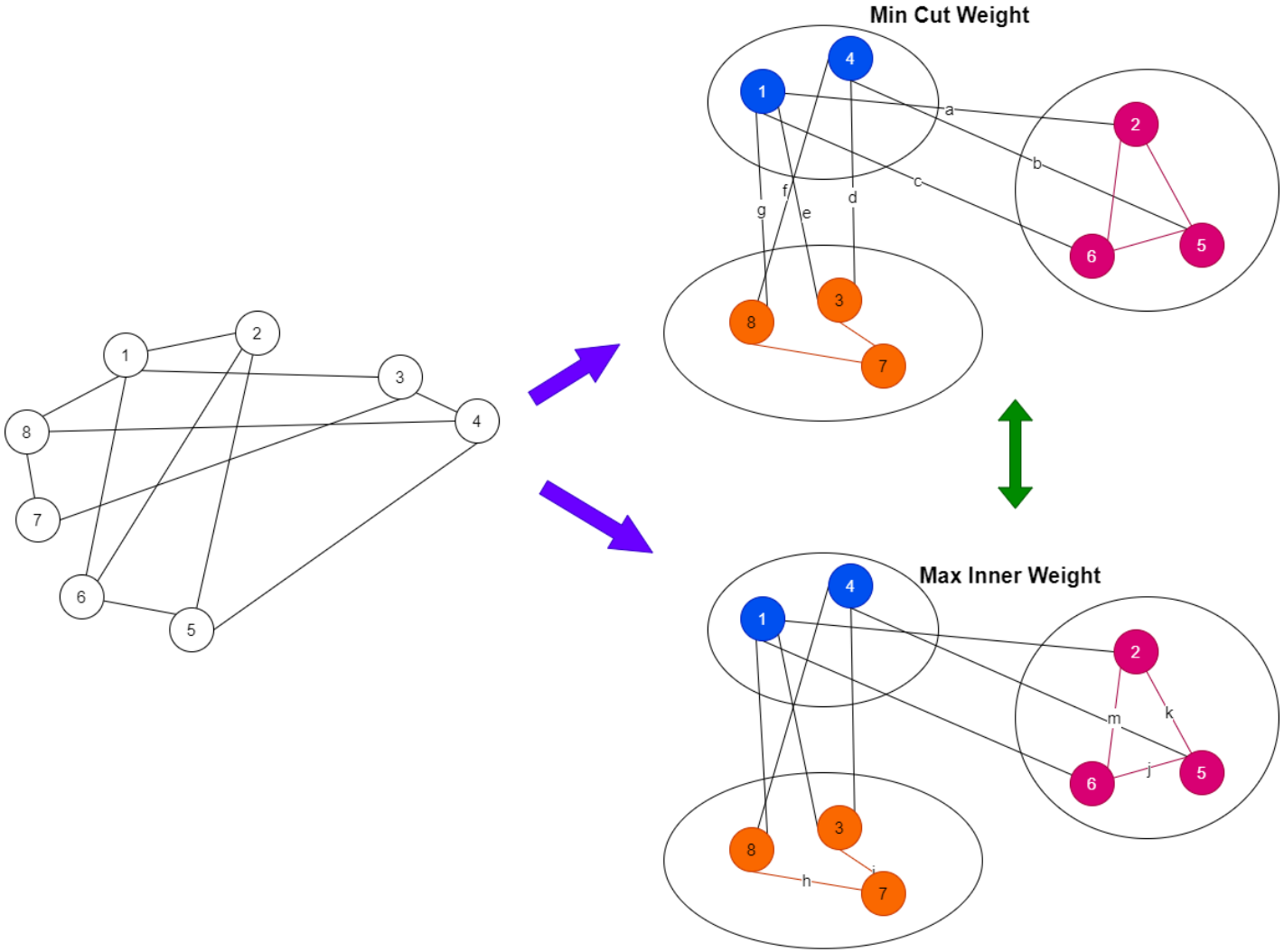
Phát biểu bài toán

Cho đồ thị vô hướng $G = (V, E)$. Với K, α là 2 hằng số cho trước:

- $|V| = N$
- $c(u, v)$ là trọng số của mỗi cạnh $(u, v) \in E$

Hãy phân hoạch tập đỉnh V thành K tập con $V_1, V_2, \cdots, V_K$.
Ràng buộc: chênh lệch số đỉnh giữa 2 tập con không vượt quá α
Mục tiêu : Tổng trọng số các cạnh có 2 đầu mút thuộc 2 tập con khác nhau (còn gọi là cut-weigth) là nhỏ nhất.
Mục tiêu tương đương: Tổng trọng số các cạnh có 2 đầu mút thuộc cùng 1 tập (còn gọi là inner-weight) là lớn nhất

Phát biểu bài toán



Hình 1: Ví dụ

Integer Programming

- Biến và miền giá trị

Integer Programming

- Biến và miền giá trị
- Ràng buộc

Integer Programming

- Biến và miền giá trị
- Ràng buộc
- Hàm mục tiêu

Biến và miền giá trị

- Biến

$$X(i,j) = \begin{cases} 1, & \text{nếu } i\ j \text{ cùng tập, } \forall i,j = \overline{0, N-1} \\ 0, & \text{nếu } i\ j \text{ khác tập, } \forall i,j = \overline{0, N-1} \\ 1, & \text{nếu } i \in V_j, \ \forall i = \overline{0, N-1}, \ j = \overline{N, N+K-1} \\ 0, & \text{nếu } i \notin V_j, \ \forall i = \overline{0, N-1}, \ j = \overline{N, N+K-1} \end{cases}$$

Biến và miền giá trị

- Biến

$$X(i,j) = \begin{cases} 1, & \text{nếu } i\ j \text{ cùng tập, } \forall i,j = \overline{0, N-1} \\ 0, & \text{nếu } i\ j \text{ khác tập, } \forall i,j = \overline{0, N-1} \\ 1, & \text{nếu } i \in V_j, \ \forall i = \overline{0, N-1}, \ j = \overline{N, N+K-1} \\ 0, & \text{nếu } i \notin V_j, \ \forall i = \overline{0, N-1}, \ j = \overline{N, N+K-1} \end{cases}$$

- Miền giá trị :

$$D(X(i,j)) = \{0,1\}$$

Ràng buộc và Hàm mục tiêu

- Ràng buộc

$$\sum_{j=N}^{N+K-1} X(i,j) = 1,\ \forall i = \overline{0, N-1} \tag{1}$$

$$-\alpha \leq \sum_{i=0}^{N-1} X(i,j) - \sum_{i=0}^{N-1} X(i,t) \leq \alpha,\ N \leq j < t \leq N+K-1 \tag{2}$$

$$X(i,j) \leq 1 + X(i,k) - X(j,k),\ \forall i,j = \overline{0, N-1},\ k = \overline{N, N+K-1} \tag{3}$$

$$X(i,j) \leq 1 + X(j,k) - X(i,k),\ \forall i,j = \overline{0, N-1},\ k = \overline{N, N+K-1} \tag{4}$$

- Hàm mục tiêu

$$\sum_{0 \leq i < j \leq N-1} X(i,j)c(i,j) \rightarrow \max \tag{5}$$

Constraint Programming

Tương tự như IP nhưng có một chút điều chỉnh cho dễ cài đặt.

- Biến

$$X(i,j) = \begin{cases} 1, & \text{nếu } i\ j \text{ cùng tập, } \forall i,j = \overline{0, N-1} \\ 0, & \text{nếu } i\ j \text{ khác tập, } \forall i,j = \overline{0, N-1} \end{cases}$$

$$Y(i,j) = \begin{cases} 0, & \text{nếu } i \in V_j\ \forall i = \overline{0, N-1}, j = \overline{0, K-1} \\ 1, & \text{nếu } i \notin V_j\ \forall i = \overline{0, N-1}, j = \overline{0, K-1} \end{cases}$$

- Miền giá trị

$$D(X(i,j)) = \{0,1\}$$

$$D(Y(i,j)) = \{0,1\}$$

Ràng buộc và Hàm mục tiêu

- Ràng buộc

$$\sum_{j=0}^{K-1} Y(i,j) = 1,\ \forall i = \overline{0, N-1} \tag{6}$$

$$-\alpha \leq \sum_{i=0}^{N-1} Y(i,j) - \sum_{i=0}^{N-1} Y(i,t) \leq \alpha,\ \forall 0 \leq t < j \leq K-1 \tag{7}$$

$$X(i,j) \leq 1 + Y(i,k) - Y(j,k),\ \forall i,j = \overline{0, N-1}, k = \overline{0, K-1} \tag{8}$$

$$X(i,j) \leq 1 + Y(j,k) - Y(i,k),\ \forall i,j = \overline{0, N-1}, k = \overline{0, K-1} \tag{9}$$

- Hàm mục tiêu

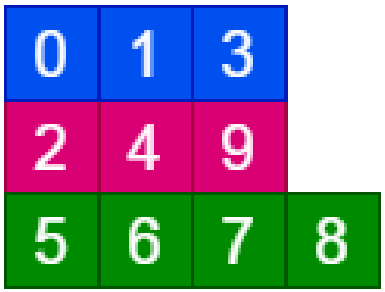
$$\sum_{0 \leq i < j \leq N-1} X(i,j)c(i,j) \rightarrow \max \tag{10}$$

Biểu diễn lời giải

- Biểu diễn lời giải
  - là 1 *List* chứa K *Array*
  - mỗi *Array* tương ứng với 1 phân hoạch.

Ví dụ:
$V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $K = 3$.
Một lời giải ngẫu nhiên *Sol*

$$Sol = \{\{0, 1, 3\}, \{2, 4, 9\}, \{5, 6, 7, 8\}\}$$



Khởi tạo lời giải

Algorithm 1: *InitTabuSearchSolution*(N, K)

Input : N, K

Output: A random solution

Init $Sol = List < Integer > [K]$ **for** $i = 0 \rightarrow N - 1$ **do**

    partition = random(K)

    Sol[partition].add(i)

end for

return *Sol*

Hàm mục tiêu

$$Objective = CutWeight + W * Violation$$

trong đó

- W là một hằng số chọn trước
- Violation là số vi phạm của phân hoạch (được tính bằng tổng vi phạm giữa mỗi cặp tập đỉnh con)

Cụ thể gọi $d(i,j)$ là số đỉnh chênh lệch giữa 2 tập đỉnh i và j thì vi phạm giữa 2 tập đỉnh là $\max(d(i,j) - \alpha, 0)$

Parameter Setup

- Các tham số vòng lặp
  - *NUM_GENERATION*: số vòng lặp
  - *stable* : tham số quyết định chiến lược tìm kiếm (tiếp tục tìm lời giải hàng xóm, quay về lời giải cải thiện gần nhất hay sinh một lời giải mới)
  - *isTimeUp*: tham số về thời gian
- Các tham số của Tabu List
  - TB_MIN, TB_MAX, số vòng lặp tối thiểu và tối đa mà 1 đỉnh nằm trong Tabu List
  - *tabuLen* số vòng lặp 1 đỉnh nằm trong Tabu List ($TB\_MIN \leq tabuLen \leq TB\_MAX$)

Tabu List Setup

Trong mỗi vòng lặp, ta cập nhật các tham số tabu như trong giải thuật sau:

Algorithm 2: *UpdateTabu*

for $i = 0 \rightarrow N - 1$ **do**
  if $tabu[i] > 0$ **then**
    $tabu[i] - -$
end for
if *newObjective* $<$ *currentObjective* **then**
  if $tabuLen > TB\_MIN$ **then**
    $tabuLen - -$
  /* Cập nhật lời giải cải thiện gần nhất */
else
  if $tabuLen < TB\_MAX$ **then**
    $tabuLen + +$

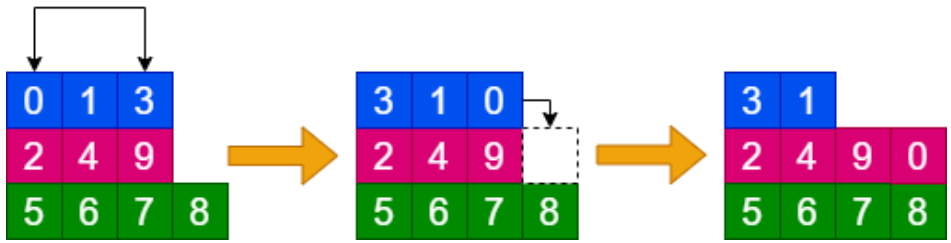
Chiến lược tìm kiếm lời giải tiếp theo

- Chiến lược
  - Duy trì 1 tabu list bằng cách gán mỗi đỉnh $v \in V$ được gán 1 giá trị $tabu[v]$ (nếu $tabu[v] = 0$ thì v không nằm trong danh sách)
  - Duyệt qua các đỉnh v không nằm trong tabu list, chuyển v sang lần lượt các tập con khác và tính toán hàm mục tiêu
  - Lưu lại phương án có hàm mục tiêu tốt nhất trong các lần thử

Cài đặt

- Tại thời điểm khởi tạo
  - Tổng trọng số của cả đồ thị được tính trước.
  - Tính Inner Weight và Cut Weight của lời giải và lưu giá trị này qua mỗi vòng lặp
- Khi cập nhật lời giải: giả sử đỉnh i là đỉnh thức j của tập con thứ k được chuyển sang tập con thứ l.
  - Đổi chỗ đỉnh thứ j và đỉnh cuối của mảng $k \rightarrow$ xoá đỉnh cuối của mảng k và thêm đỉnh bị xoá vào cuối mảng thứ l
  - Cập nhật inner weight: inner weight cũ $-$ (tổng trọng số các đỉnh trong k nối với i) $+$ (tổng trọng số của các đỉnh trong l nối với i).
  - Cập nhất cut weight : (tổng trọng số) $-$ inner weight mới

Ví dụ: 1 lời giải trong đồ thị 10 đỉnh, đỉnh đầu của tập con thứ nhất được chuyển sang tập con thứ 2



Hình 2: Cập nhật lời giải

Sơ đồ giải thuật

Algorithm 3: TabuSearch

```
currentSol ← InitTabuSearchSolution(N, K)
i ← 0
while i < NUM_GENERATION&&!isTimeUp do
    i++
    if currentSolution tốt hơn bestSolution then
        bestSoltution ← currentSolution
        stable ← 0
    else if stable = stableLimit then
        currentSolution ← lastImprovedSolution
        stable ← 0
    else
        stable++
        if i%restartFrequency = 0 then
            currentSol ← InitTabuSearchSolution(N, K)
            /* Xoá toàn bộ đỉnh ra khỏi tabu list                    */
    currentSolution ← newLocalSearchNeighbor
    UpdateTabu()
end while
```

Genetic Algorithm (GA)

Genetic Algorithm (GA)

- Mã hoá cá thể

Genetic Algorithm (GA)

- Mã hoá cá thể
- Toán tử lai ghép và đột biến

Genetic Algorithm (GA)

- Mã hoá cá thể
- Toán tử lai ghép và đột biến
- Fitness

Genetic Algorithm (GA)

- Mã hoá cá thể
- Toán tử lai ghép và đột biến
- Fitness
- Lược đồ giải thuật

Cá thể và Quần thể

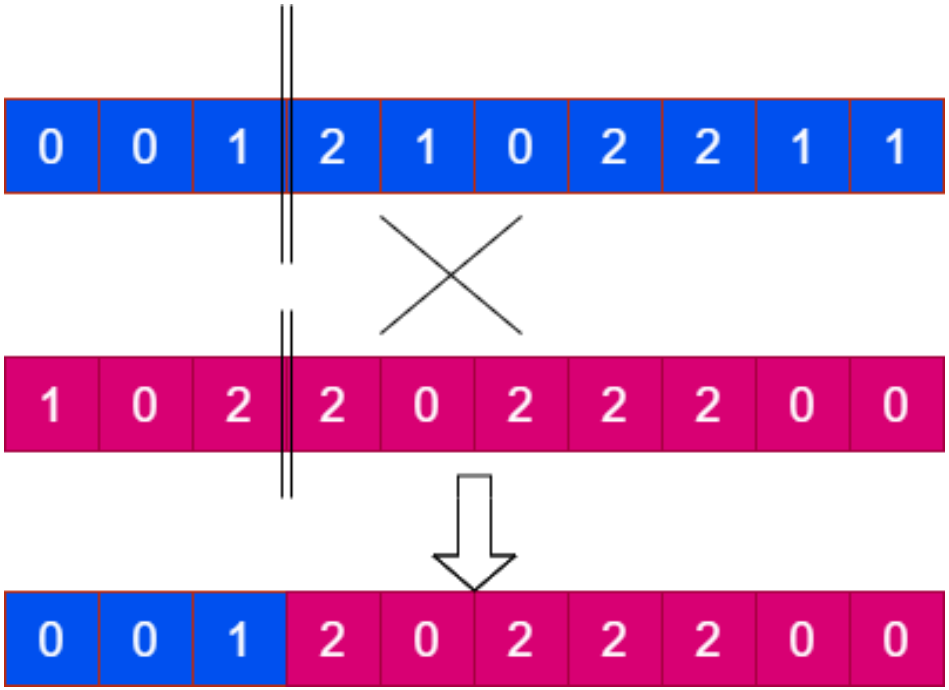
• Cá thể: mỗi cá thể được biểu diễn bởi 1 mảng 1 chiều. Giá trị j của mỗi phần i thể hiện đỉnh i thuộc phân hoạch j
Ví dụ :
$V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $K = 3$.
Một lời giải ngẫu nhiên

Hình 3: Cá thể GA

• Quần thể :
Khởi tạo ngẫu nhiên các cá thể với kích thước quần thể được xác định tỉ lệ với số đỉnh của đồ thị.

Toán tử lai ghép

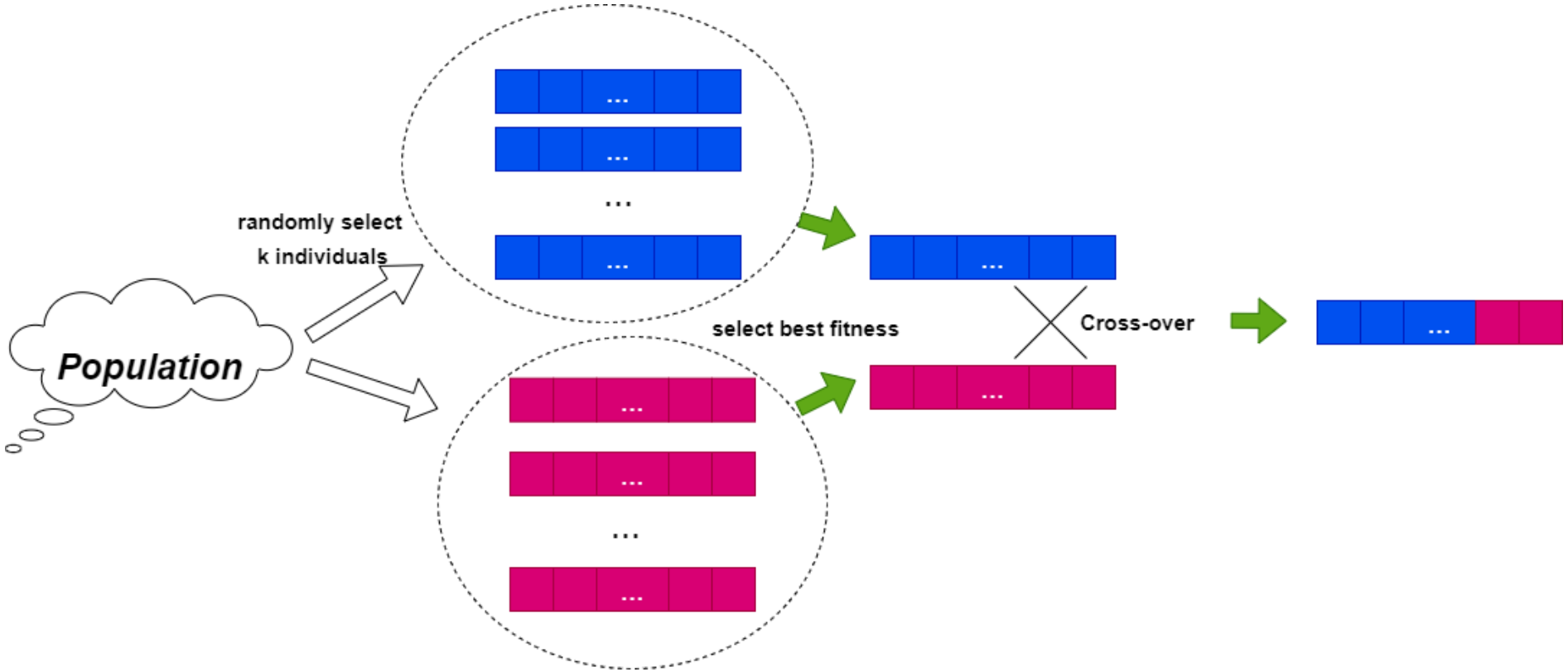
Ône point cross-over:



Hình 4: 1-point Cross-over

Chiến lược lai ghép

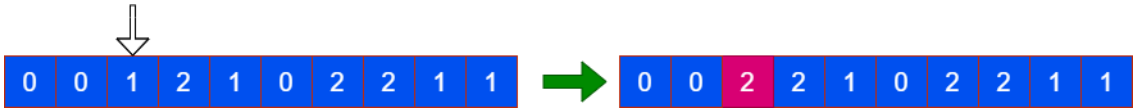
Sử dụng k-Tournament Selection.



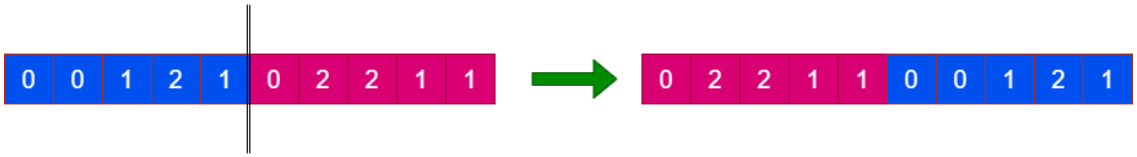
Hình 5: k Tournament Selection

Toán tử đột biến

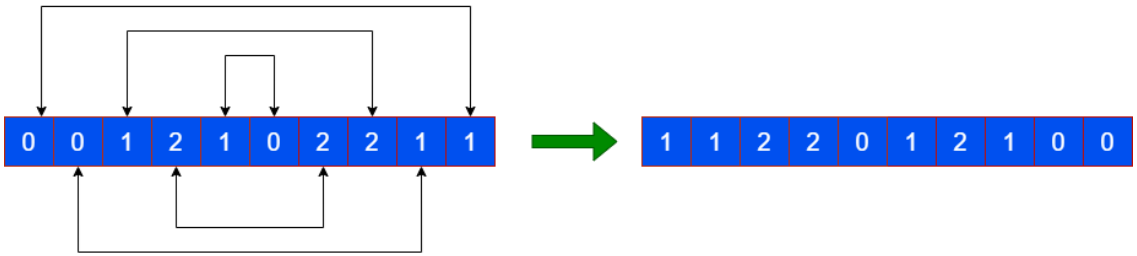
One point Mutation



SwapHalf Mutation



Reverse Mutation



Chiến lược đột biến

OnePointMutation ($R1 = 0.06$)
ReversedMutation($R2 = 0.08$)
SwapHalfMutation($R3 = 0.1$)

Algorithm 4: *mutate*

Input : r

if $r < R1$ **then**
    this.onePointMutate()
else if $r < R2$ **then**
    this.reverseMutate()
else if $r < R3$ **then**
    this.swapHalfMutate()

Hàm fitness

Với mỗi cá thể, định nghĩa Objective function như trong Tabu Search

$$Objective = CutWeight + W * Violation$$

Giá trị hàm Fitness :

$$Fitness = -Objective$$

Sơ đồ giải thuật (Vòng lặp chính)

```
/* Khởi tạo quần thể                                                    */
pop ← initPopulation(POP_SIZE)
i ← 0
while i < NUM_GENERATION&&!isTimeUp do
    i++
    for j = 0 → POP_SIZE/2 / do
        parent1 ← tournamentSelection()
        parent2 ← tournamentSelection()
        child ← crossover(parent1, parent2)
        child.mutate(r)
        /* r ∈ (0, 1) là một số được sinh ngẫu nhiên                    */
        pop.add(child)
    end for
    /* sắp xếp quần thể theo fitness và xoá đi POP_SIZE/2 cá thể tồi
       nhất                                                             */
    pop.selection()
end while
```

Kịch bản thực nghiệm

- Môi trường
  - Server : SageMaker Studio Lab
  - CPU : 4 core, Intel(R) Xeon(R) Platinum 8175M CPU @ 2.50GHz
  - RAM : 16 GB
- Các tiêu chí so sánh :
  - Kết quả (min cut weight)
  - Thời gian chạy (runtime)
  - Tính ổn định (giữa Tabu Search và GA)
- Số lần chạy :
  - IP,CP chạy 1 lần trên mỗi bộ
  - TabuSearch, GA chạy 5 lần trên mỗi bộ

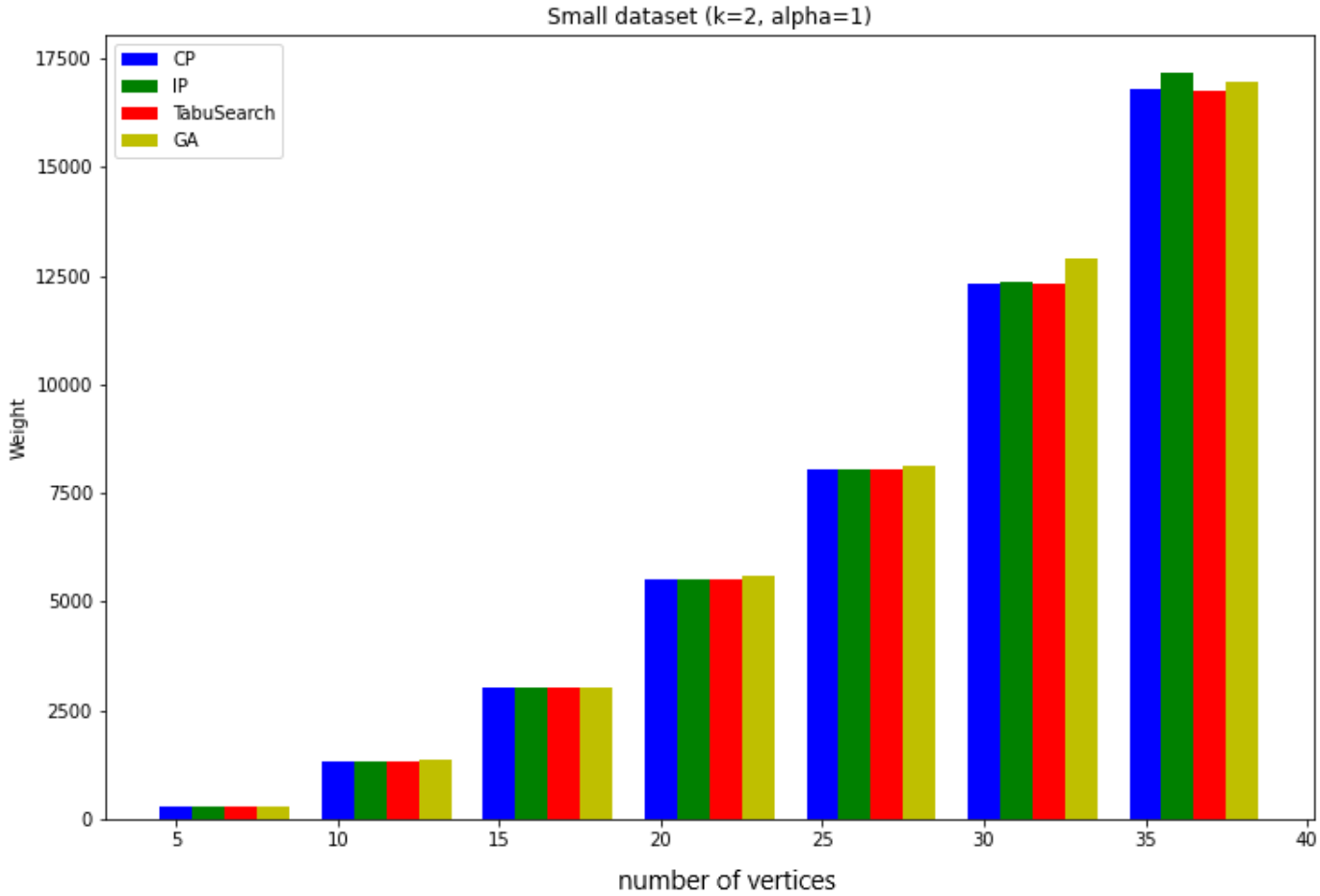
Dữ liệu thực nghiệm

- Dữ liệu tự sinh:
  - Small (số đỉnh (5,10,15,20,25,30,35))
  - Medium (số đỉnh (40,45,50,55,60,65,70))
  - Large (số đỉnh (75,80,85,90,95,100))
  - Huge (số đỉnh (100-900))
- Dữ liệu thực tế
  - Financial stocks network (pearson)
  - Financial stocks network (distance)
  - Financial stocks network (weight)
  - USAir97

Tham số thực nghiệm

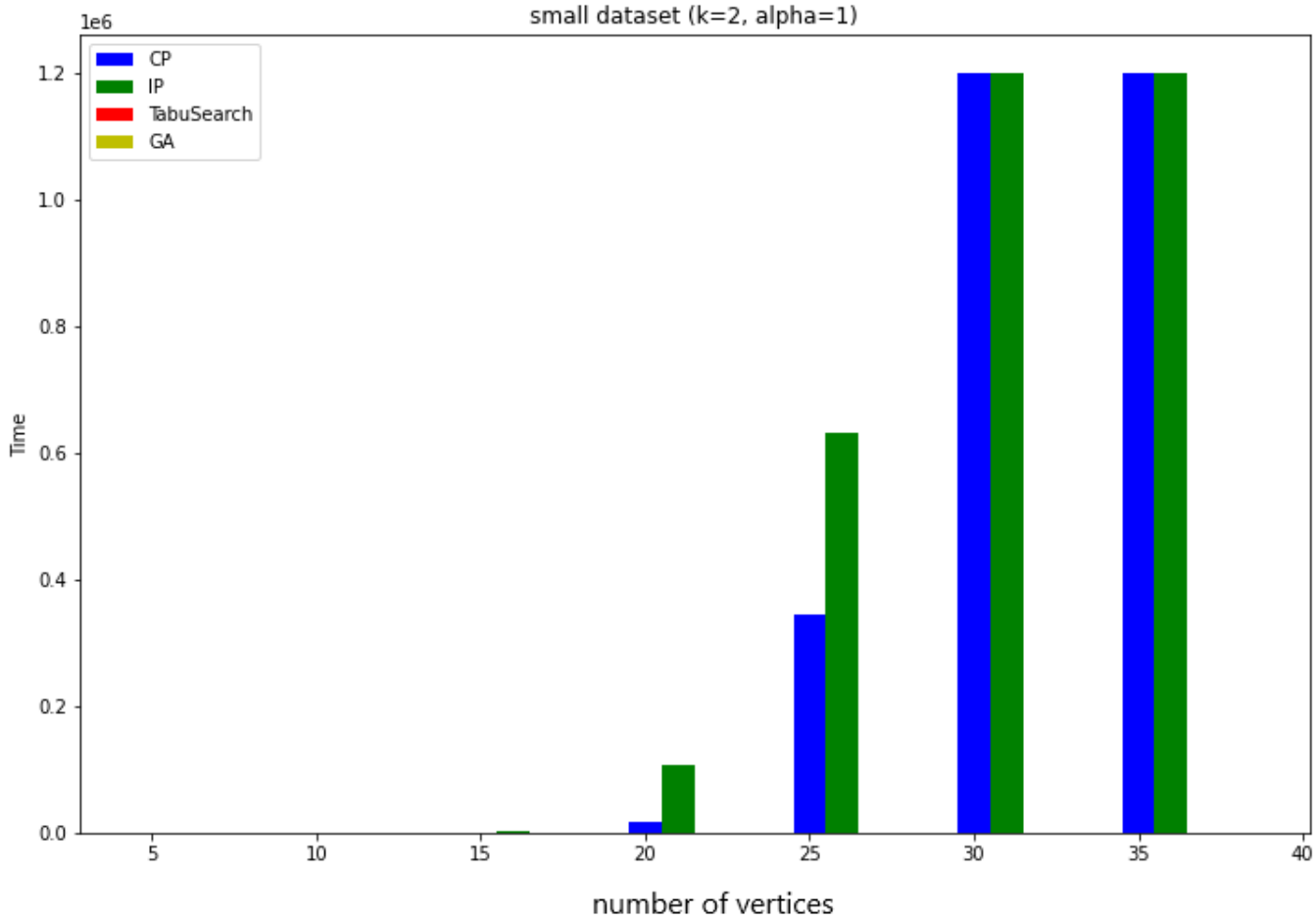
- Hyper parameter của Tabu Search
  - $NUM\_GENERATION = (\frac{\#vertices}{36} + 1) * 100$
  - $TA\_MIN = 2,\ TB\_MAX = 5$
  - $stableLimit = 30,\ restartFrequency = 100$
  - $W = 1000$ (trọng số hàm mục tiêu)
- Hyper Parameter của GA
  - $NUM\_GENERATION = (\frac{\#vertices}{36} + 1) * 300$
  - $NUM\_INDIVIDUAL = (\frac{\#vertices}{36} + 1) * 100$
  - $W = 800$ (trọng số hàm mục tiêu)
- Ngưỡng thời gian chạy
  - 20 phút cho các bộ small, medium, large
  - 30 phút cho các bộ huge, realworld

Kết quả trên Small Data



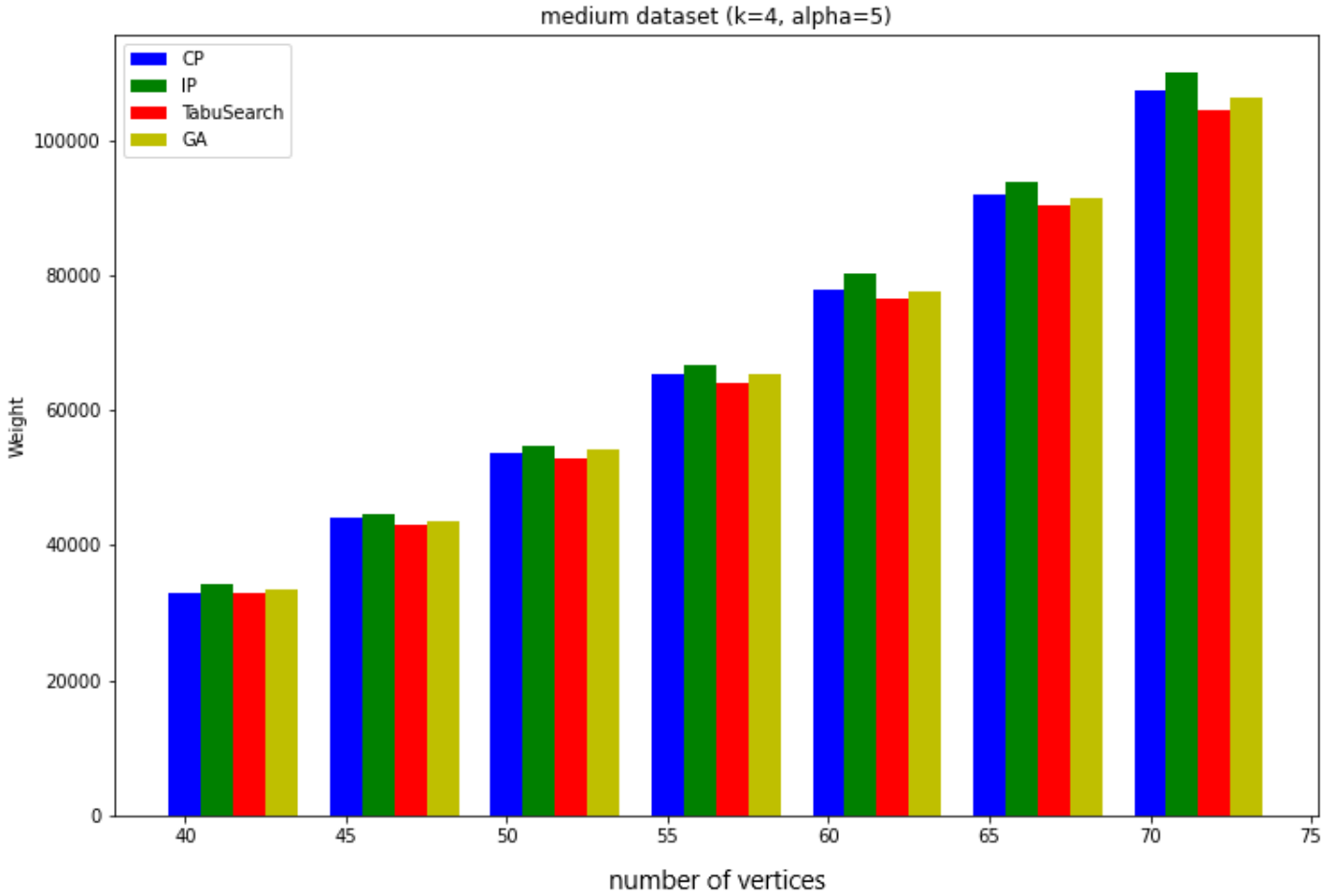
Hình 6: Results on small data $k = 2, \alpha = 1$

Runtime trên Small Data



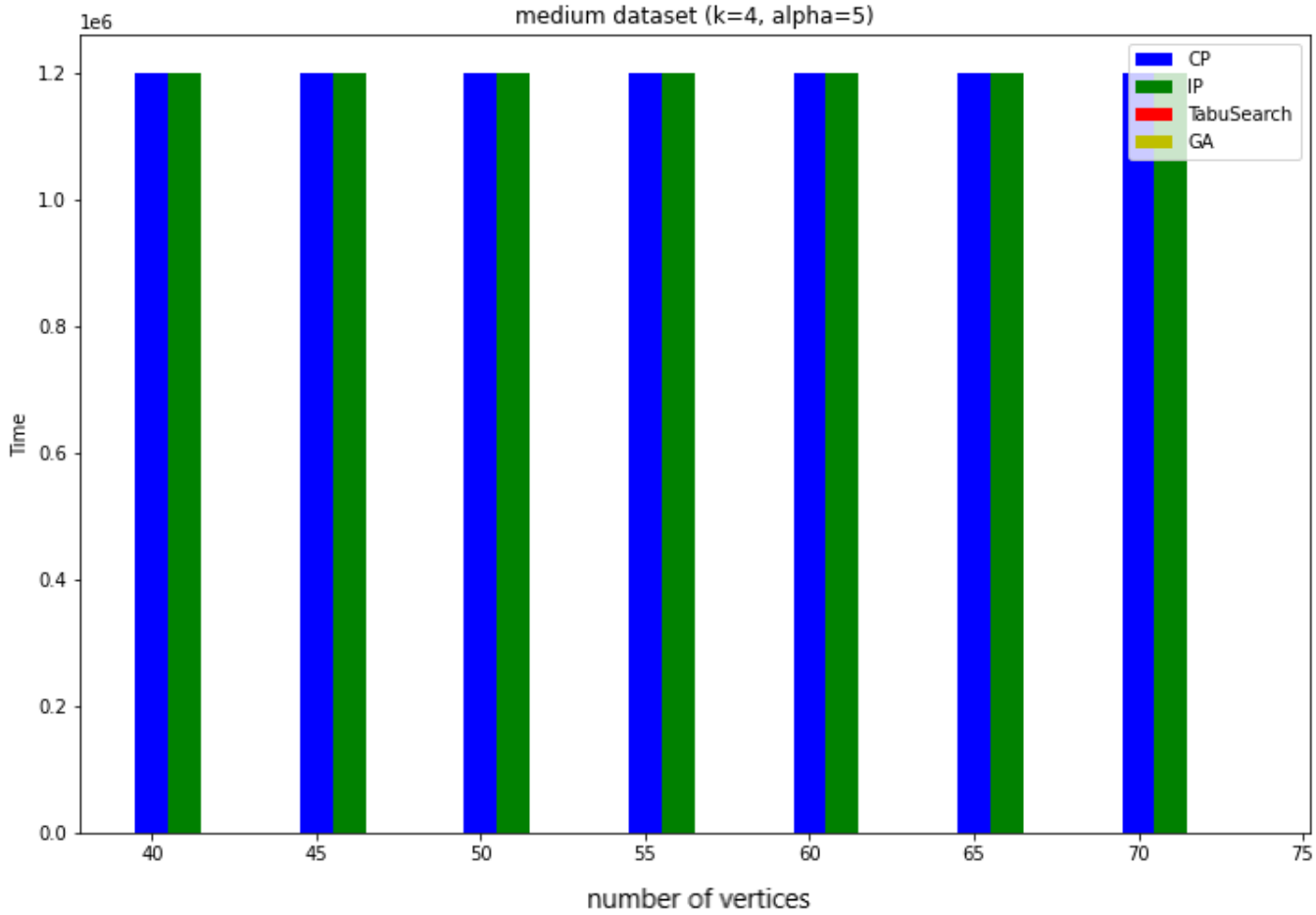
Hình 7: Runtime on small data $k = 2$, $\alpha = 1$

Kết quả trên Medium Data



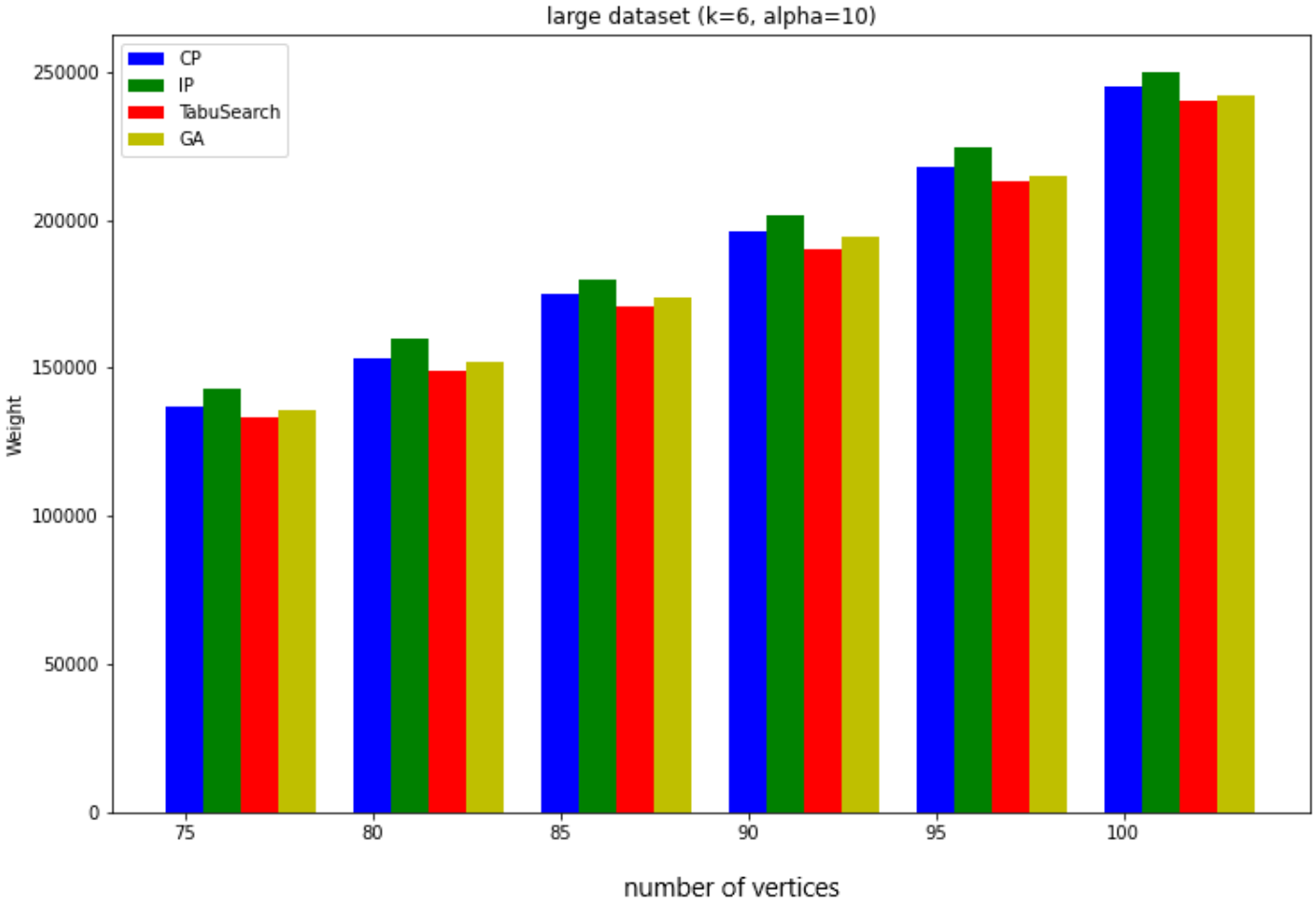
Hình 8: Results on medium data $k = 2$, $\alpha = 1$

Runtime trên Medium Data



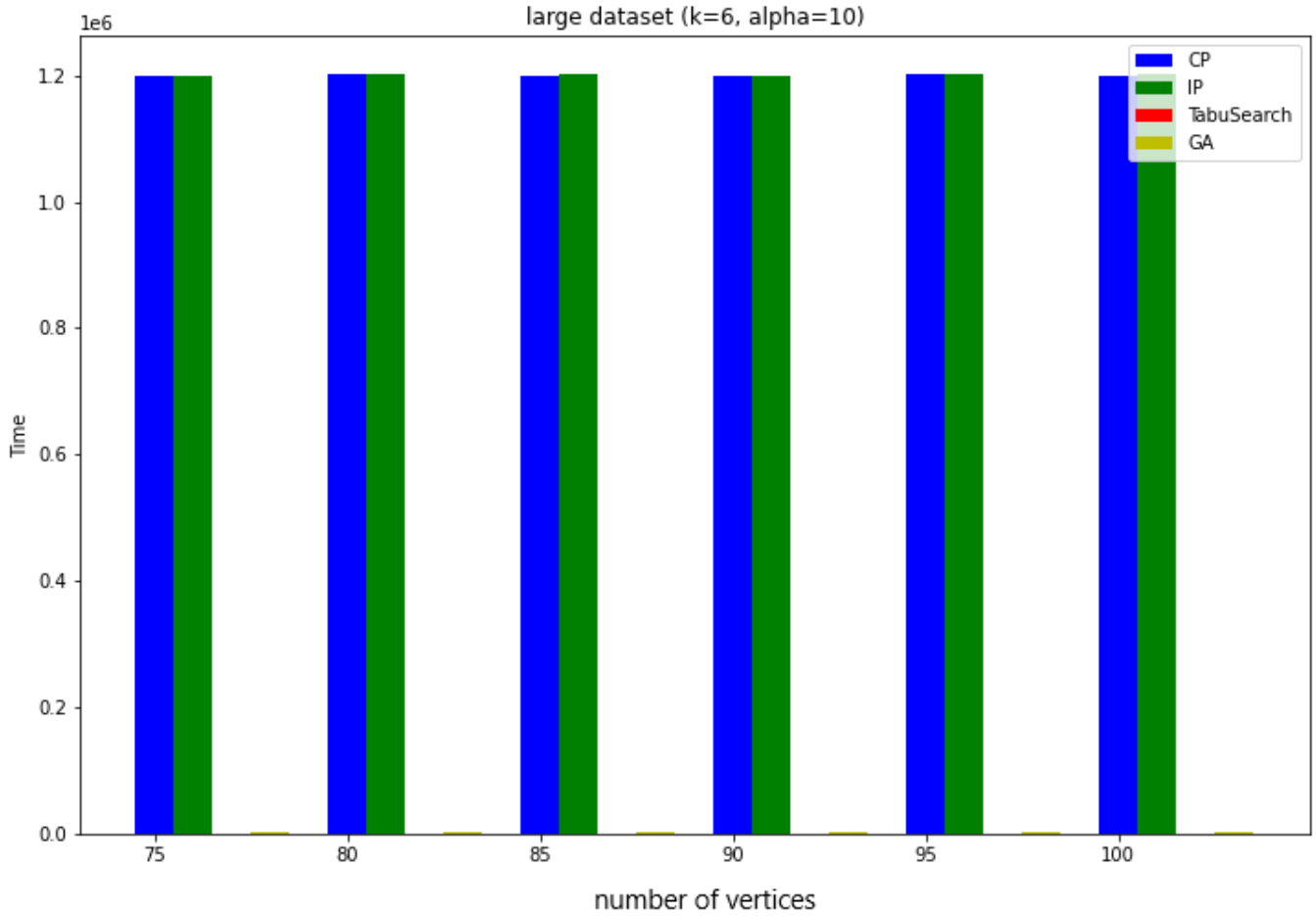
Hình 9: Runtime on medium data $k = 2$, $\alpha = 1$

Kết quả trên Large Data



Hình 10: Results on large data $k = 2$, $\alpha = 1$

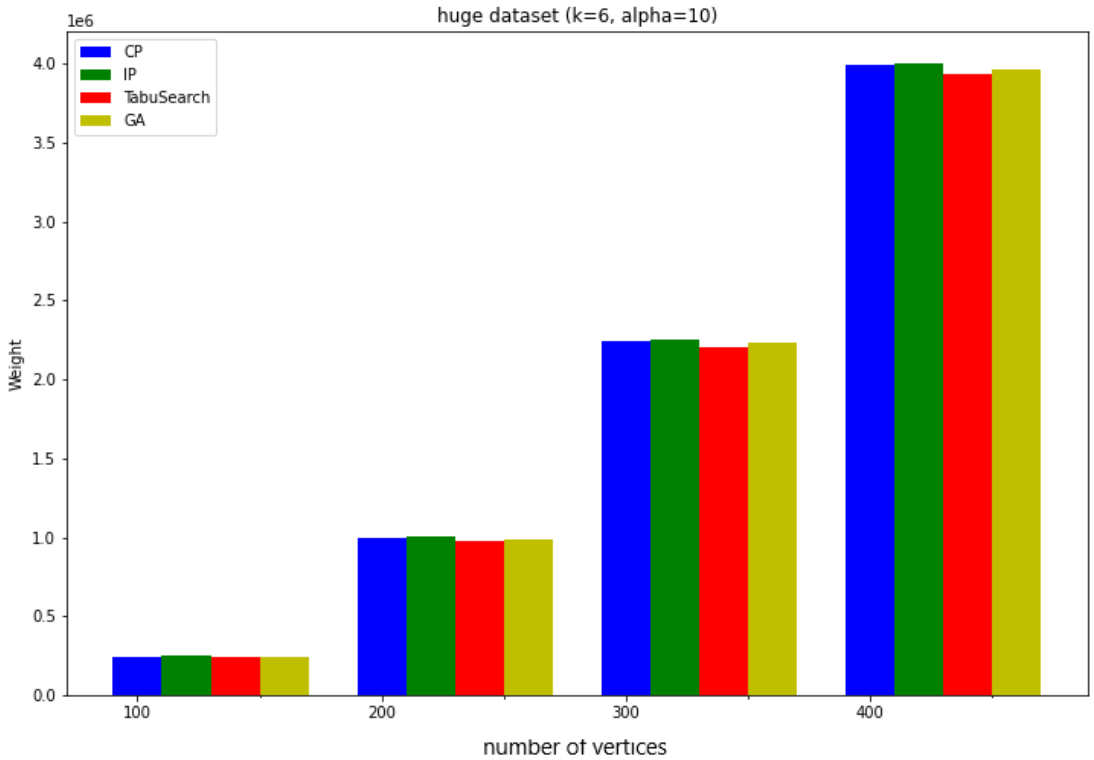
Runtime trên Large Data



Hình 11: Runtime on large data $k = 2, \alpha = 1$

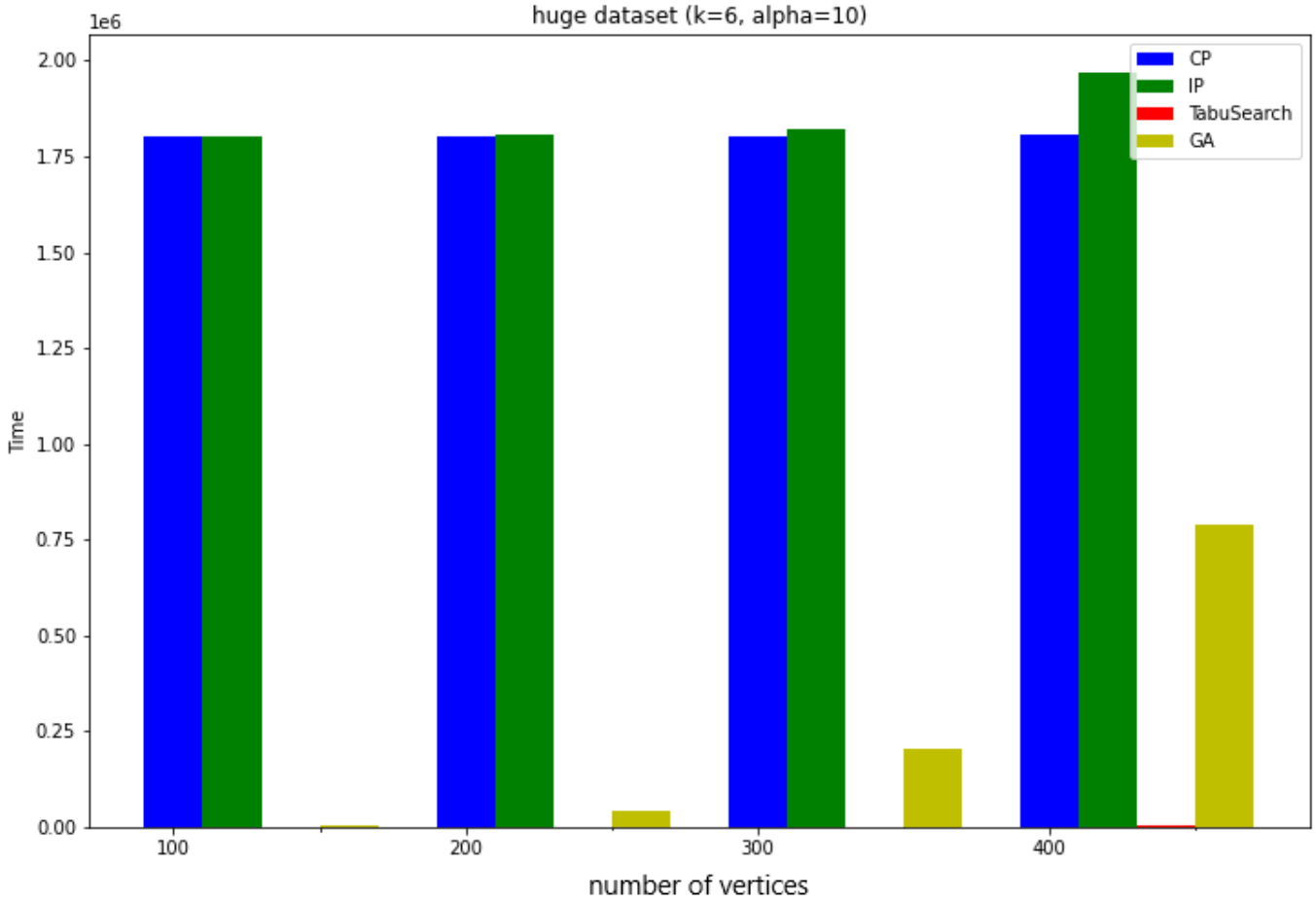
Kết quả trên Huge Data

CP và IP không thể chạy được trên bộ dữ liệu từ 500 đỉnh trở lên với $k > 3$



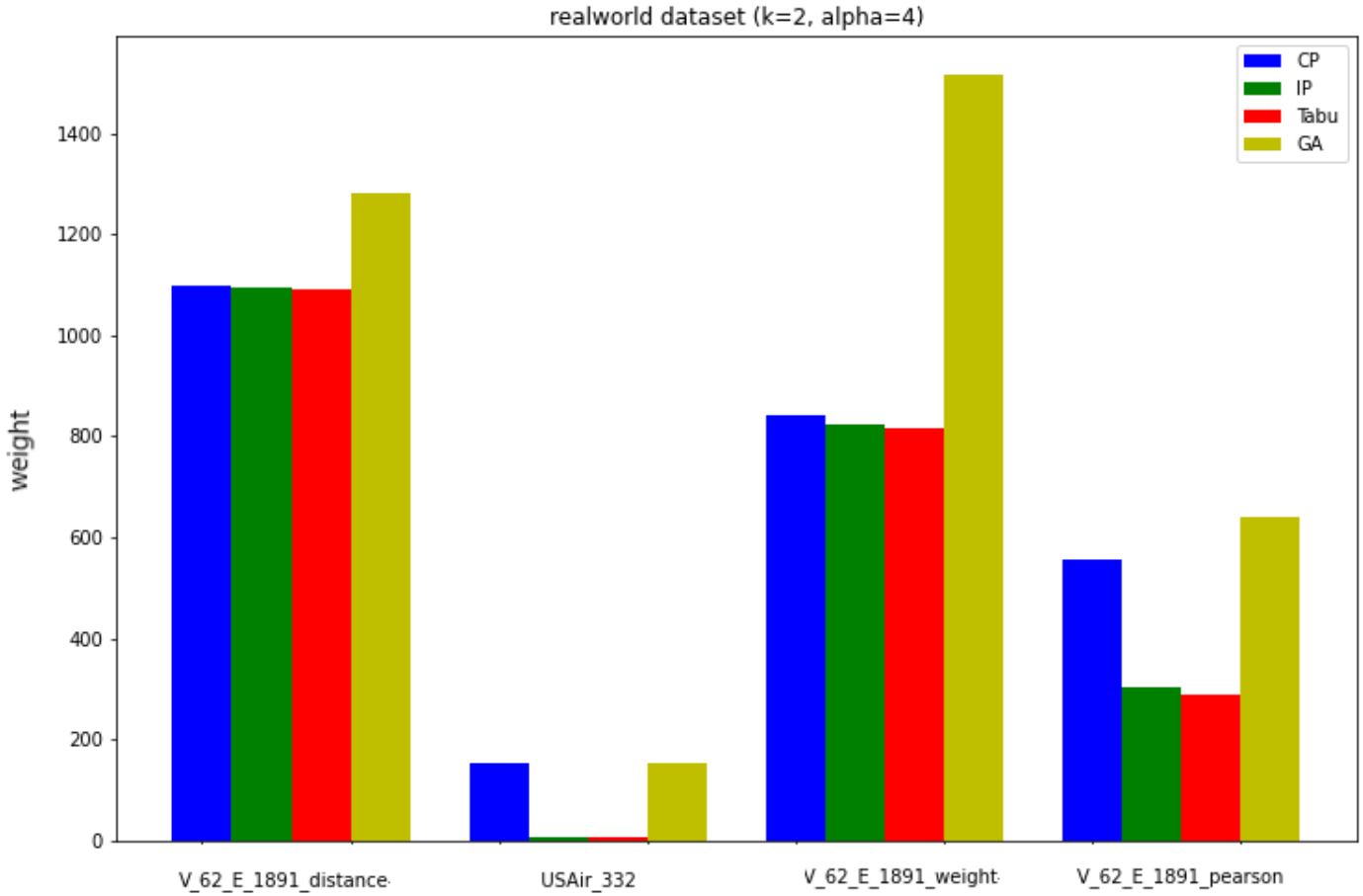
Hình 12: Results on huge data $k = 2,\ \alpha = 1$

Runtime trên Huge Data



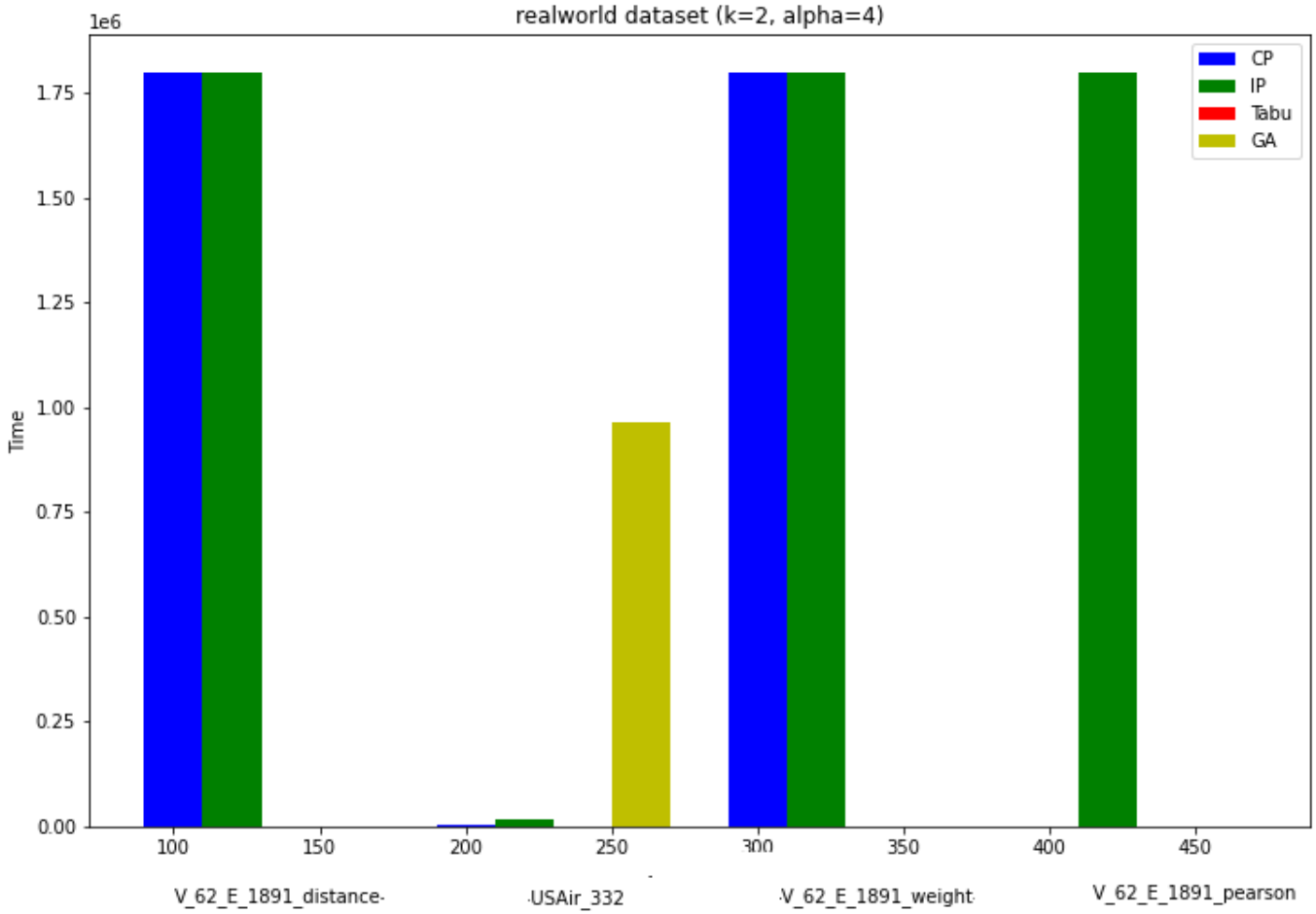
Hình 13: Runtime on huge data $k = 2, \alpha = 1$

Kết quả trên dữ liệu thực



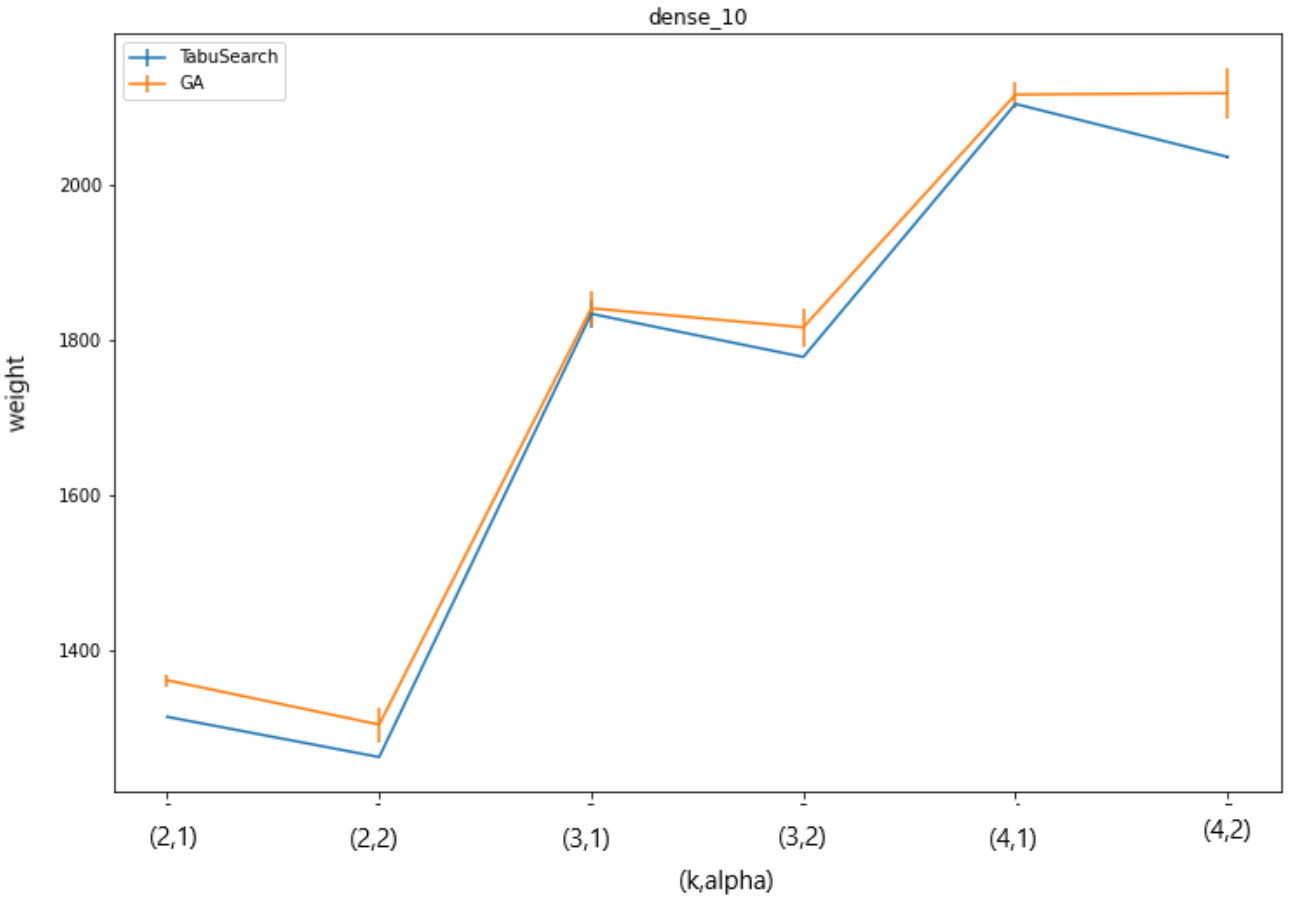
Hình 14: Results on realworld data $k = 2$, $\alpha = 4$

Runtime trên dữ liệu thực



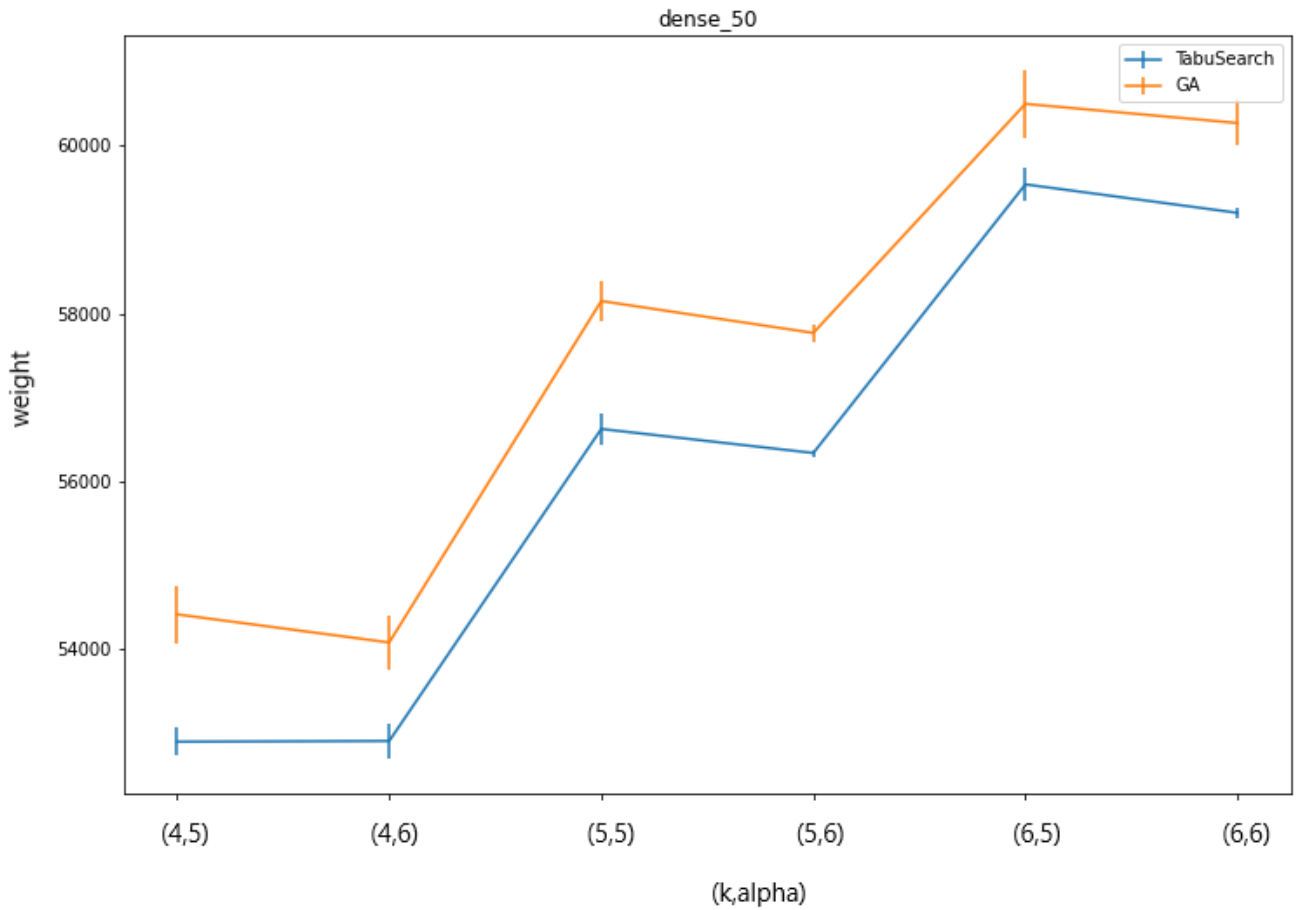
Hình 15: Runtime on realworld data $k = 2, \alpha = 4$

Small Data



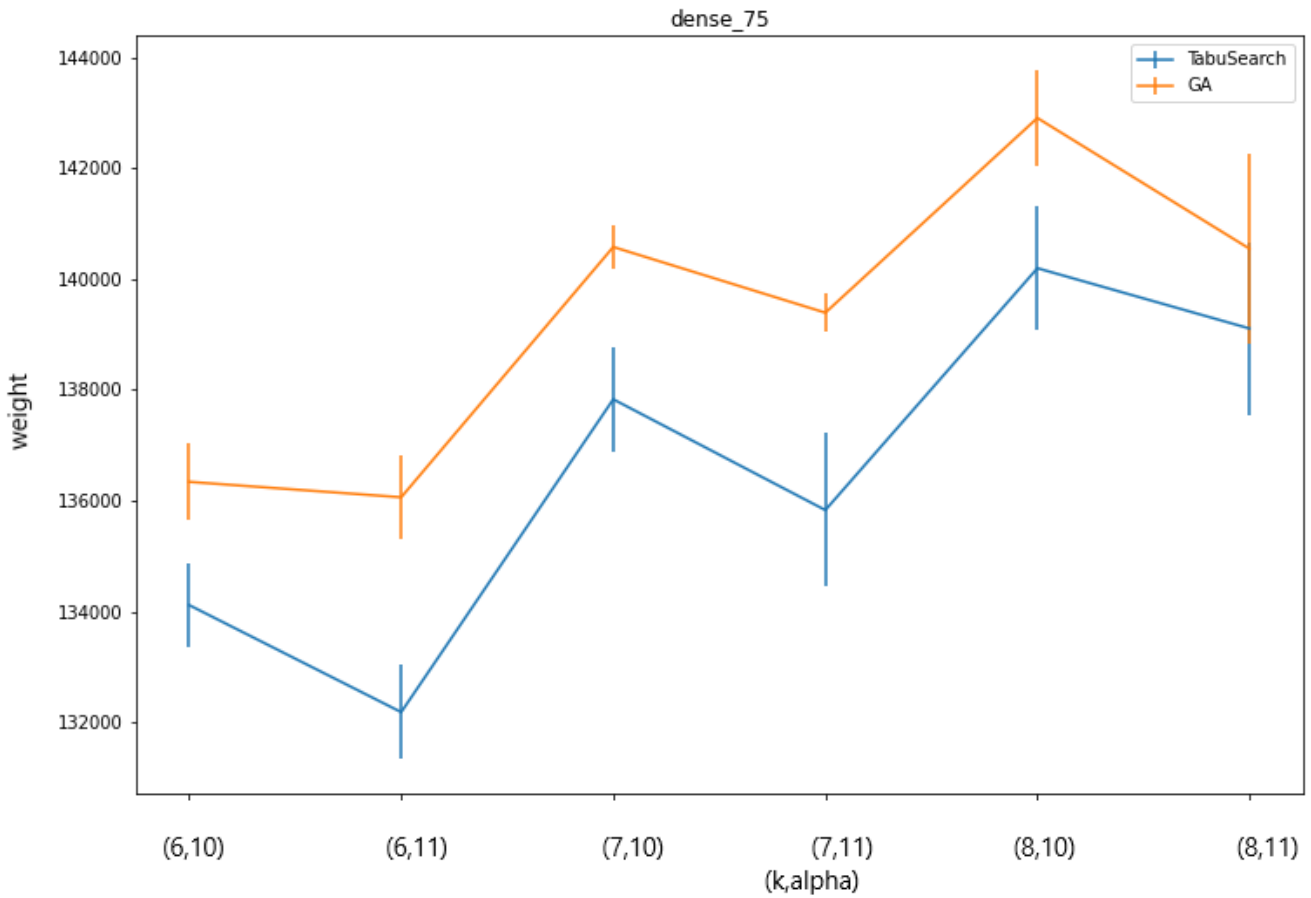
Hình 16: So sánh tính ổn định của TabuSearch và GA trên small data

Medium Data



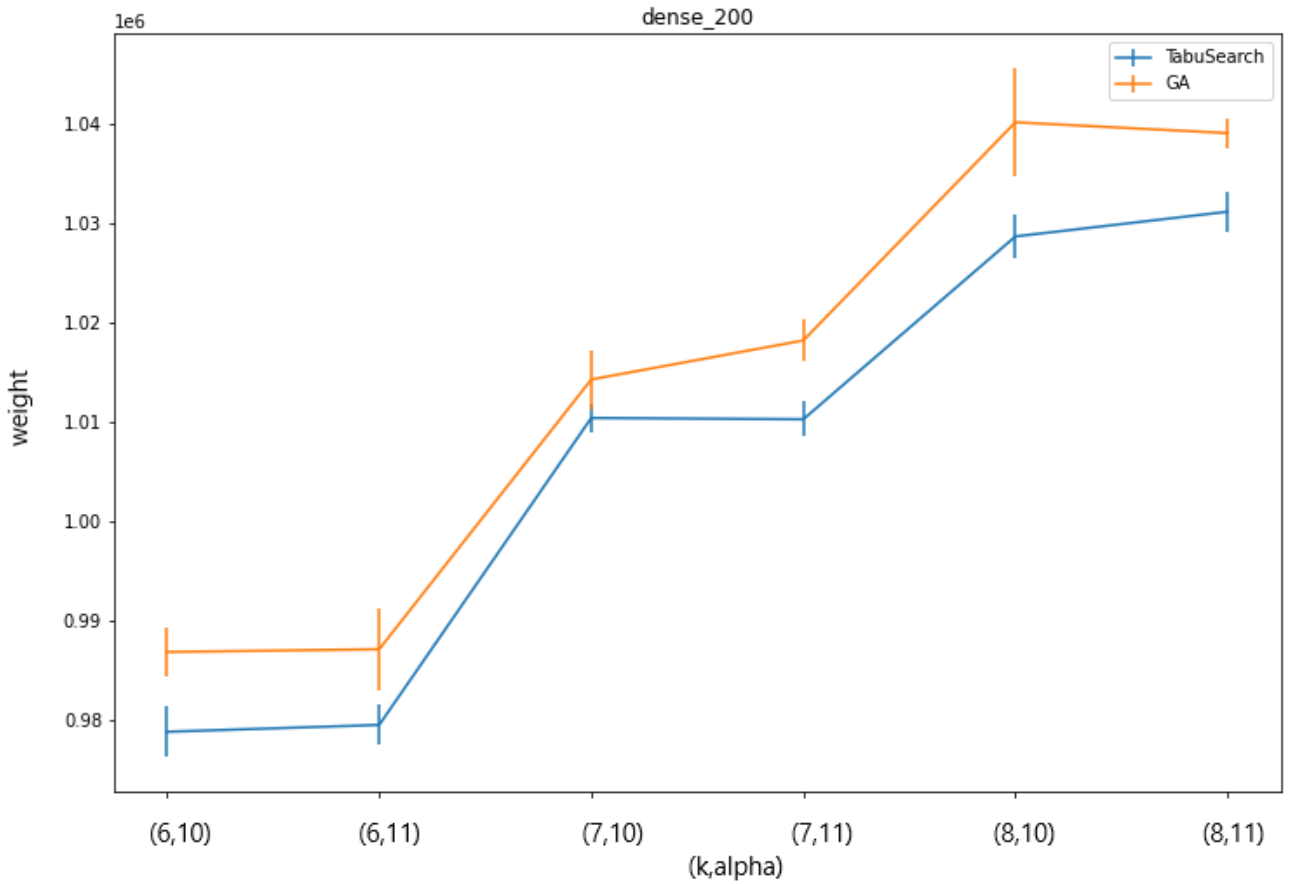
Hình 17: So sánh tính ổn định của TabuSearch và GA trên medium data

Large Data



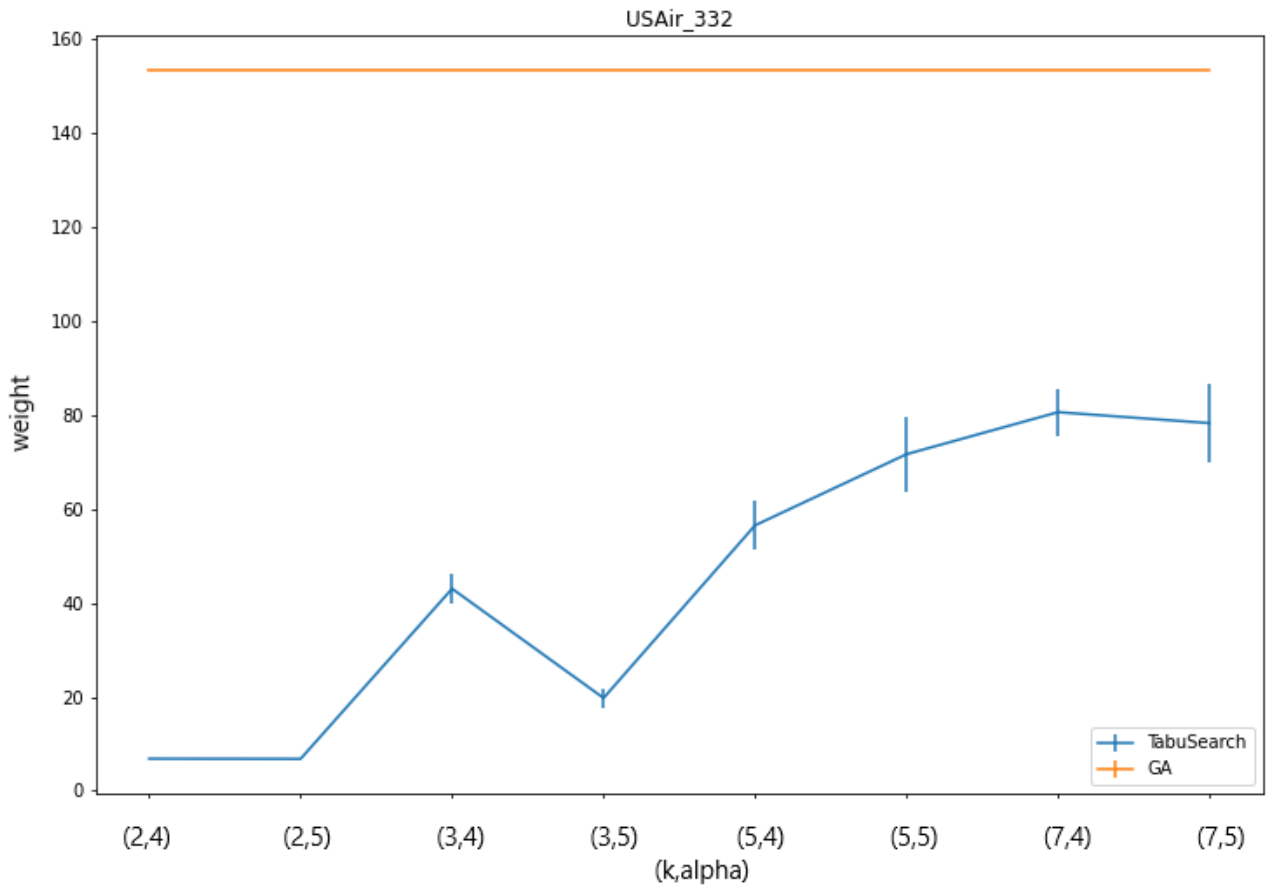
Hình 18: So sánh tính ổn định của TabuSearch và GA trên large data

Huge Data



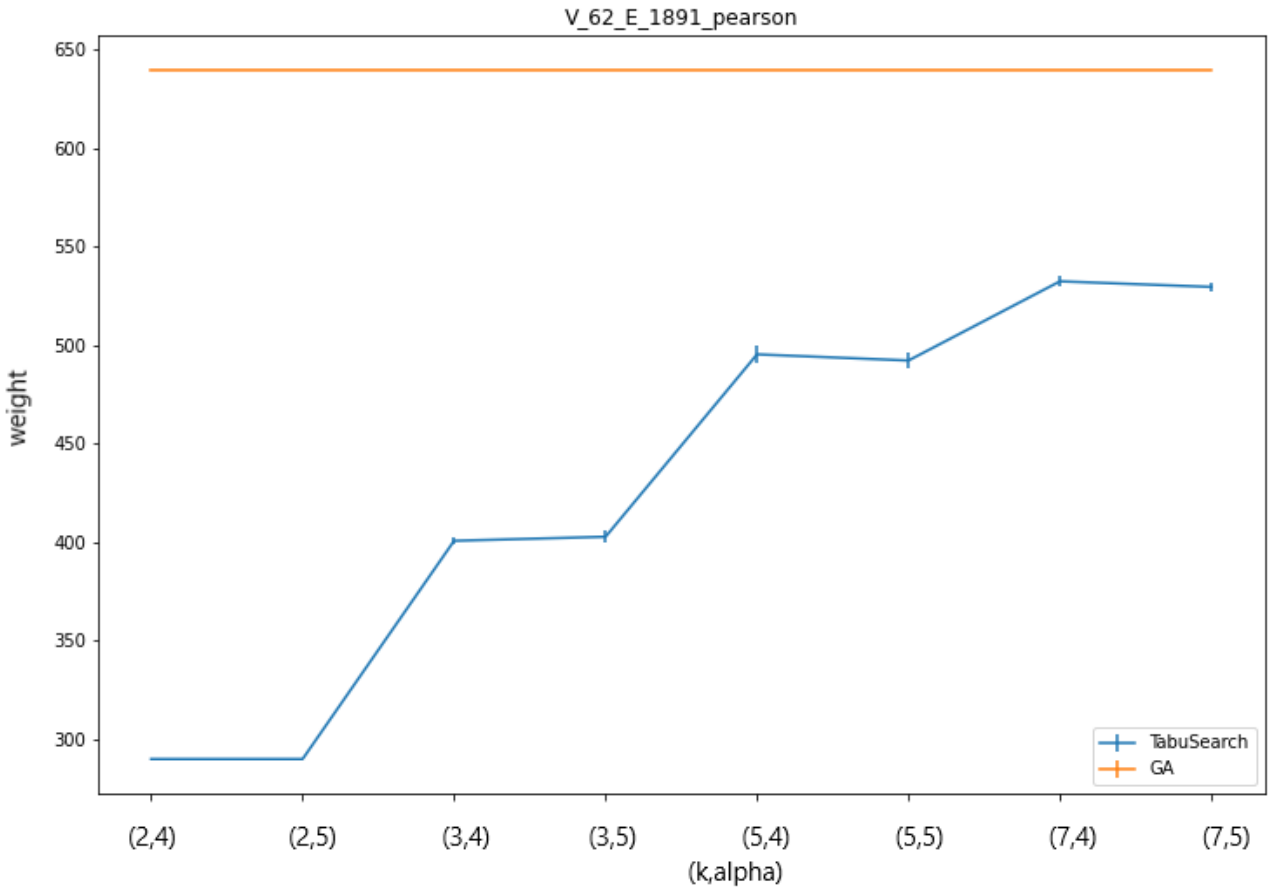
Hình 19: So sánh tính ổn định của TabuSearch và GA trên huge data

Realworld Data



Hình 20: So sánh tính ổn định của TabuSearch và GA trên USAir_332 data

Realworld Data



Hình 21: So sánh tính ổn định của TabuSearch và GA trên V_62_E_1891_pearson data

Kết luận

- Về kết quả
  - Kết quả trên bộ small data cho thấy CP, IP và Tabu Search đều đưa ra được kết quả tối ưu trong thời gian cho phép.
  - Trên các bộ medium và large data cho thấy Tabu Search thể hiện sự vượt trội, GA cũng cho kết quả tương đối tốt, CP và IP đều kém hiệu quả.
  - Trên bộ dữ liệu huge data, kết quả các thuật toán mang lại là gần như nhau, tuy nhiên Tabu Search vẫn trội hơn.
  - Trên bộ dữ liệu thực, Tabu Search và IP cho kết quả tốt, CP không ổn định, GA cho kết quả tồi.

- Về Runtime
  - Trong tất cả các bộ dữ liệu thì Tabu Search đều chạy nhanh hơn cả, GA chậm hơn Tabu Search, còn CP và IP thì chậm hơn khá nhiều.
  - Trên bộ small data thì CP nhanh hơn IP và cả 2 đều cho kết quả tốt.
  - Trên bộ dữ liệu thực, CP chạy nhanh hơn IP nhưng kết quả không chính xác.

- Tổng kết
  - Tabu Search vượt trội hơn về mọi mặt trên các bộ dữ liệu.
  - Trên bộ dữ liệu nhỏ, bên cạnh Tabu Search thì có thể sử dụng CP.
  - Trên bộ medium, large data thì có thể sử dụng GA.
  - Trên bộ huge data thì do kết quả các thuật toán mang lại là xấp xỉ nhau, tuy nhiên Tabu Search chạy nhanh hơn nhiều.
  - Trên bộ dữ liệu thực thì IP cũng là một thuật toán tốt bên cạnh Tabu Search.

Phân chia công việc

- Nguyễn Tiến Long (20180129) : cài đặt Tabusearch và GA, chạy thực nghiệm, làm slide
- Nguyễn Đức Long (20183583) : mô hình hoá IP, chuẩn bị dữ liệu, vẽ biểu đồ so sánh
- Nguyễn Thành Long (20183585) : mô hình hoá và chạy thực nghiệm CP