Báo cáo BTL - Tính toán song song:

Bài toán xếp hậu sử dụng GA song song

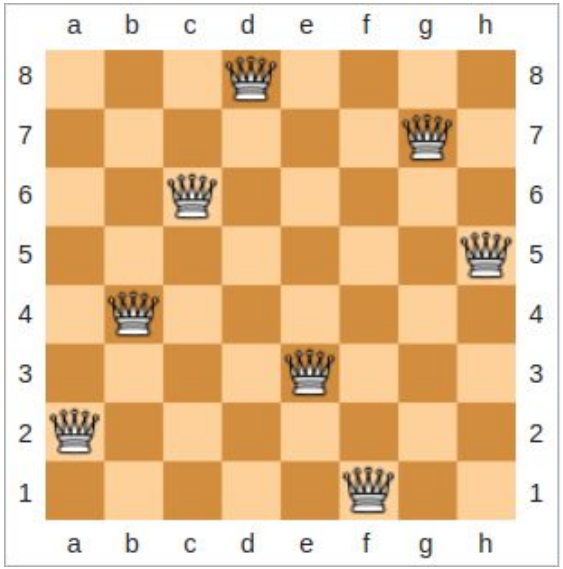
Nguyễn Hồng Ngọc, Nguyễn Văn Linh

Outline

1. Bài toán xếp hậu
2. Thuật toán GA song song
3. Thuật toán GA đa quần thể song song
4. Thử nghiệm

Bài toán xếp hậu

- Cho N quân hậu và một bàn cờ N x N, hãy sắp xếp các quân hậu lên bàn cờ sao cho không có quân hậu nào có thể "ăn" được quân hậu khác
- Mô tả:
  - Input: Số quân hậu N
  - Output: Vị trí các quân hậu trên bàn cờ

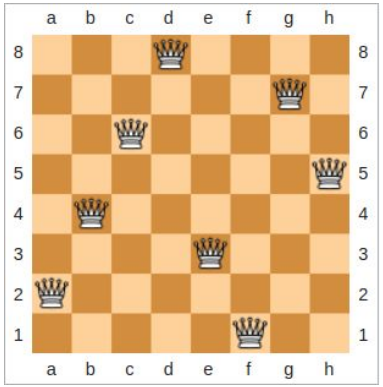


Giải thuật GA song song

- Mã hóa cá thể:
  - Vector (N+1) chiều số nguyên, x_i - tọa độ hàng của con hậu ở cột i
  - Số nguyên cuối cùng lưu giá trị fitness ~ giá trị vi phạm của cách xếp
  - Mục tiêu: fitness min (bằng 0 trong trường hợp lời giải chính xác)

x_1	x_2	x_3	...	...	x_N	*fitness*

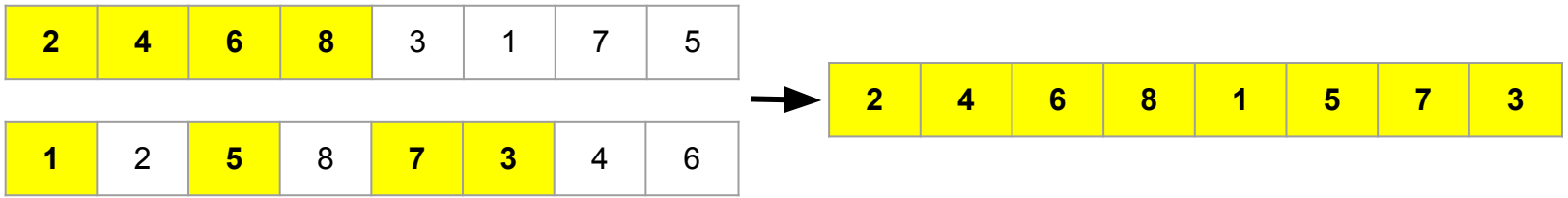
- Ví dụ:



2	4	6	8	3	1	7	5	***0***

Giải thuật GA song song

- Khởi tạo:
  - Cá thể lời giải ít nhất phải là một chuỗi hoán vị của dãy 1,2,3,...,N
  - Khởi tạo ngẫu nhiên: tạo 1 chuỗi hoán vị ngẫu nhiên
- Lai ghép một điểm cắt từ 2 cá thể A và B:
  - Chọn vị trí cắt i trong khoảng [1...N-2]
  - Cách 1: giữ nguyên phần [1...i] của A, phần còn lại sắp xếp theo thứ tự tương ứng trên B
  - Cách 2: ngược lại, giữ phần từ i đến hết



- Đột biến: hoán vị 2 vị trí ngẫu nhiên

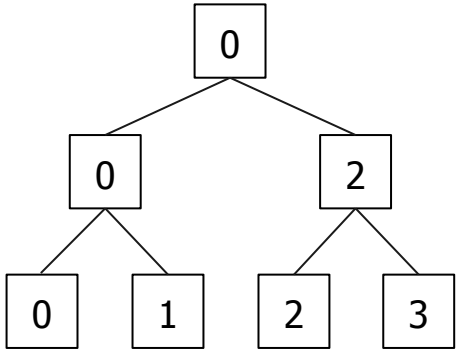
Giải thuật GA song song

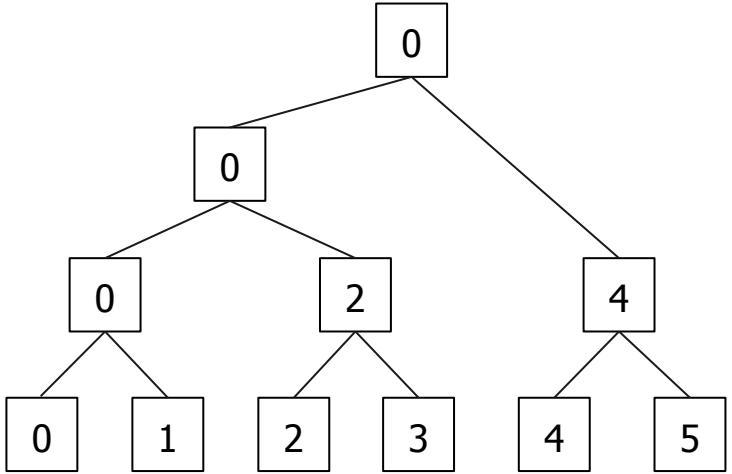
Song song:

- Khởi tạo song song:
  - Mỗi process khởi tạo 1 phần của quần thể (kích thước tổng của quần thể cố định) rồi gửi cho process 0
  - Process 0 thực hiện sắp xếp lại quần thể theo fitness rồi broadcast cho tất cả process khác
  - Mỗi process lưu một copy của toàn quần thể đã sắp xếp
- Lai ghép:
  - Mỗi process đảm nhiệm 1 phần quá trình lai ghép
  - Tính toán số lượng con lai của mỗi process = kích thước quần thể * tỷ lệ lai ghép / số process
  - Mỗi process thực hiện lai ghép song song đến khi đủ số lượng con lai
- Đột biến:
  - Mỗi process tự thực hiện đột biến trên tập con lai của mình
  - Sau quá trình này, mỗi process chứa 1 copy của quần thể gốc và 1 tập con lai riêng
- Chọn lọc:
  - Sắp xếp tập con lai bằng quicksort trên local
  - Merge các tập con lai bằng *parallel merge sort* về tập con lai chung ở process 0
  - Process 0 thực hiện *merge* tập con lai chung với quần thể gốc nhưng chỉ giữ lại *num-pop* cá thể tốt nhất
  - Process 0 broadcast lại quần thể mới lên các process khác

Giải thuật GA song song

Parallel Merge Sort: thực hiện merge song song





Giải thuật GA song song

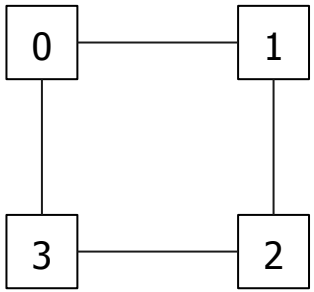
- Điều kiện dừng:
  - Vượt quá số lượng thế hệ tối đa
  - Hoặc giá trị fitness tốt nhất bằng 0 (tìm được lời giải)
  - Hoặc kết quả fitness không giảm sau một số lượng thế hệ nào đó
- Thống kê thời gian:
  - Thời gian chạy của chương trình
  - Thời gian truyền thông: Mỗi process gửi thời gian truyền thông của nó về process 0 để tổng hợp

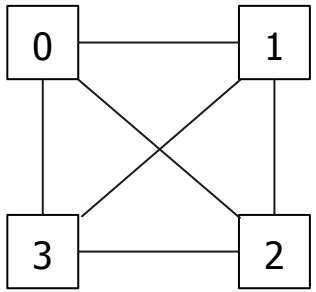
```
commTime -= MPI_Wtime();
communicate with other processes;
commTime += MPI_Wtime();
```

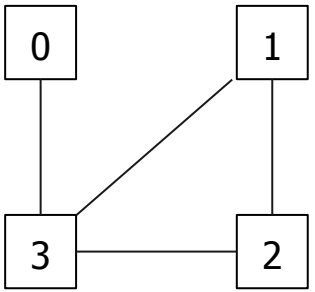
- Thống kê tổng:
  - Do tính ngẫu nhiên gần đúng, thời gian chạy là khác nhau ở mỗi lần chạy với cùng bộ tham số
  - Chạy nhiều lần và thống kê thời gian trung bình

Giải thuật GA đa quần thể song song

- Nhận thấy mức độ song song còn thấp, thuật toán chạy chậm, communication time cao. Khi tốc độ kết nối kém thì chạy rất chậm
- Sử dụng GA đa quần thể để tăng mức độ song song, giảm bớt thời gian truyền thông
- Ý tưởng:
  - Sử dụng tập nhiều quần thể, mỗi quần thể tiến hóa riêng độc lập
  - Theo chu kỳ, các quần thể thực hiện trao đổi cá thể với nhau, gọi là quá trình "di cư"
  - Ở đây, mỗi process sẽ đảm nhiệm 1 quần thể. Việc truyền thông giờ chỉ phải thực hiện theo chu kỳ và chỉ phải truyền một số cá thể di cư







Giải thuật GA đa quần thể song song

- Các tham số bổ sung:
  - MIGRATION_GENERATION: số thế hệ thực hiện chu kỳ di cư
  - PERCENT_INDI_TO_MIGRATION: tỷ lệ cá thể đem đi di cư (ví dụ quần thể có 100 cá thể thì 10 cá thể di cư)
- Quá trình chạy:
  - Các quần thể thực hiện khởi tạo, lai ghép, đột biến, chọn lọc độc lập trên local
  - Khi đến chu kỳ di cư, các quần thể thực hiện di cư như sau:
    - Mỗi quần thể chọn ra các cá thể tốt nhất (số lượng theo tỷ lệ di cư) rồi clone và đưa vào tập **M**
    - Thực hiện Parallel Merge Sort với các tập **M** rồi broadcast tập gộp cho mỗi process
    - Mỗi process thực hiện Merge tập **M** gộp với tập quần thể gốc
  - Nhận thấy, số lần truyền thông cũng như kích thước dữ liệu cần truyền thông đều giảm đi nhiều so với trước

Thực nghiệm

Cài đặt thực nghiệm:

- Cấu hình máy:
  - Máy 1: Intel Core i7, 4 nhân 8 luồng
  - Máy 2: Intel Core i5, 2 nhân 4 luồng
- Kết nối mạng: Wifi LAN - tốc độ 65 ~ 72 Mbit/s
- Phiên bản MPI: 4.0.2
- Ngôn ngữ lập trình: C

Thực nghiệm

Cài đặt thực nghiệm:

- Tham số:

Tham số	GA	Multi Population GA
Kích thước quần thể	1000	1000 (tổng kích thước)
Tỉ lệ lai - đột biến	100% - 50%	100% - 50%
Số thế hệ tối đa	500	500
Số thế hệ dừng khi fitness không đổi	50	50
Tỷ lệ di cư	~	5%
Chu kỳ di cư (thế hệ)	~	5

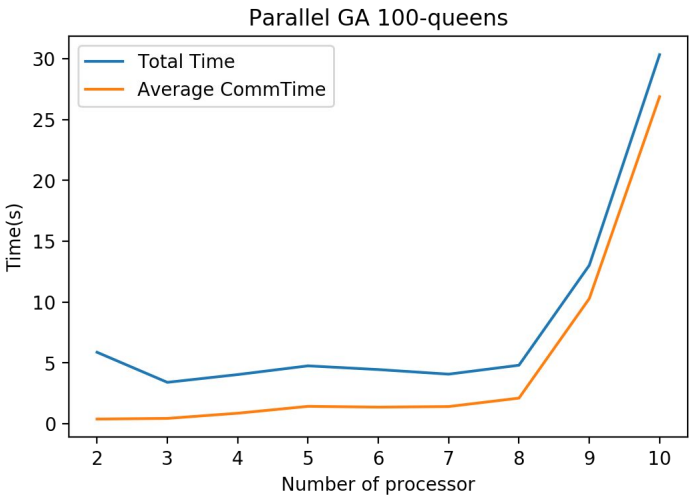
Thực nghiệm

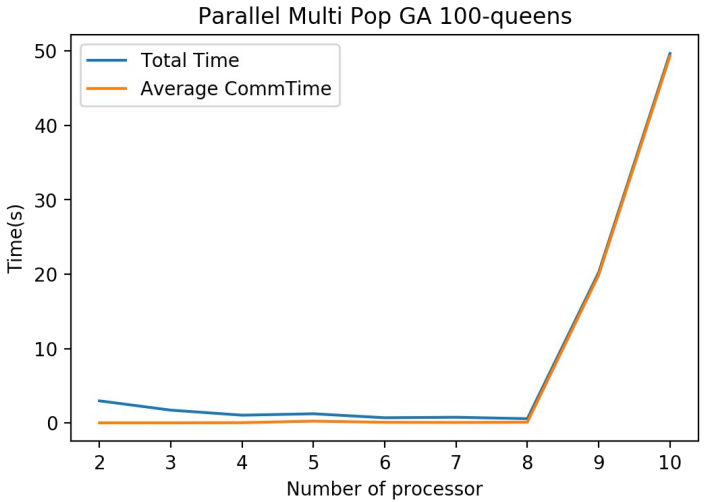
- Kịch bản thực nghiệm:
  - Thực nghiệm trên máy 1, số luồng thay đổi từ 2 đến 10
  - Thực nghiệm trên máy 1 và máy 2, số luồng như sau:

Máy 1	1	2	2	4	4	6	8
Máy 2	1	1	2	2	4	4	4

- Số quân hậu giữ cố định: 100
- Số lần chạy mỗi thuật toán: 10 lần
- Mỗi kịch bản được thực hiện trên mỗi giải thuật để đánh giá:
  - Thời gian chạy trung bình
  - Thời gian truyền thông trung bình trên mỗi processor

Thực nghiệm trên máy 1

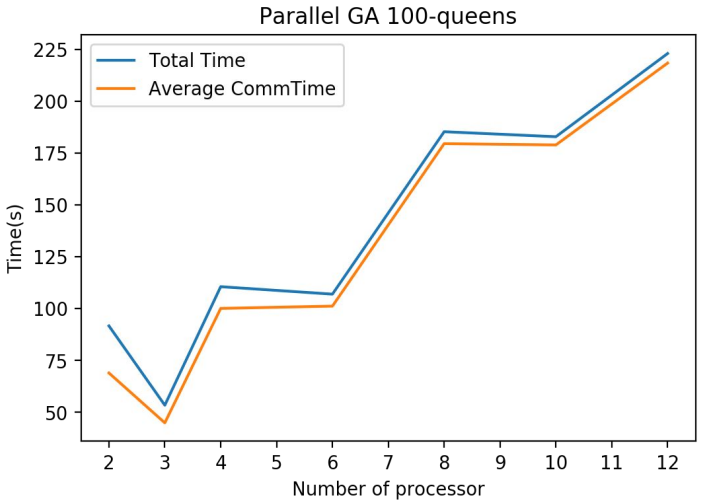


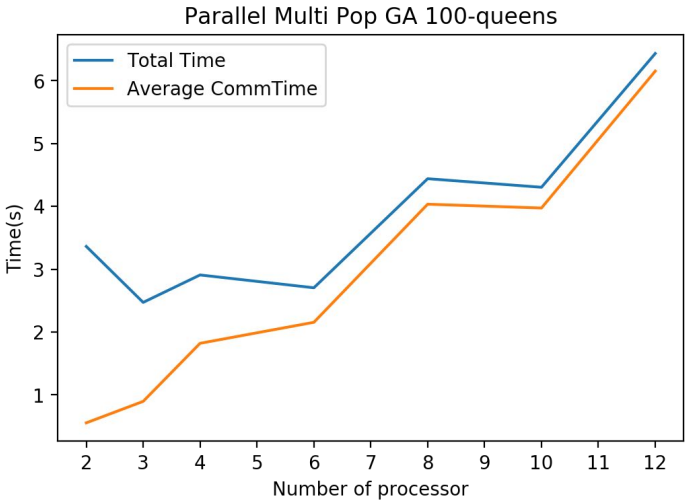


Nhận xét:

- Khi số lượng process tăng lên nhưng không vượt quá số luồng thực là 8: thời gian tổng giảm dần, thời gian truyền thông tăng và thời gian tính toán giảm. Còn khi vượt quá số luồng vật lý thì thời gian truyền thông trở lên rất lớn.
- GA chỉ đạt thời gian tối ưu khi thực hiện trên 3 process, trong khi GA đa quần thể đạt tối ưu khi có 8 process và thời gian cũng nhanh hơn nhiều. Chứng tỏ GA đa quần thể tối ưu song song hơn.

Thực nghiệm trên máy 1 và máy 2





Nhận xét:

- Trong trường hợp 2 máy, thời gian truyền thông do mạng là rất lớn, vì thế thời gian chạy GA bị đẩy lên rất cao. Trong khi đó, GA đa quần thể truyền thông ít hơn nên ít bị ảnh hưởng và thời gian chạy vẫn rất nhỏ
- Tuy nhiên thời gian truyền thông vẫn rất đáng kể (do tốc độ mạng khá thấp) nên GA đa quần thể vẫn chỉ đạt tối ưu ở 3 ~ 6 process


Thank you!
grammarly