

# Báo cáo Thuật toán Đom Đóm (Firefly Algorithm)

23122017 Bùi Anh Quân

Tháng 11 năm 2024

## Mục lục

<b>1</b>	<b>Firefly Algorithm</b>	<b>2</b>
1.1	Giới thiệu . . . . .	2
1.1.1	(1.1) Ý tưởng cốt lõi . . . . .	2
1.1.2	(1.2) Lịch sử ngắn gọn . . . . .	2
1.1.3	(1.3) Ứng dụng tiêu biểu . . . . .	2
1.2	Cơ sở toán học của thuật toán . . . . .	2
1.3	Chi tiết triển khai thuật toán (Mã giả) . . . . .	4
1.4	Thuật toán dùng để so sánh . . . . .	5
1.5	Bài toán kiểm tra và tiêu chí đánh giá . . . . .	7
1.6	Kết quả benchmark và phân tích . . . . .	9
1.6.1	Rastrigin: mức độ tiệm cận nghiệm tối ưu và đặc trưng hội tụ . . . . .	9
1.6.2	Rastrigin: đa dạng quần thể và hiện tượng dừng sớm (stagnation) .	15
1.6.3	Knapsack: hiệu năng fixed-budget (gap tới nghiệm tối ưu) . . . . .	20
1.6.4	Knapsack: performance/data profiles và so sánh thống kê . . . . .	23
1.7	Thảo luận và kết luận . . . . .	29

# 1 Firefly Algorithm

## 1.1 Giới thiệu

### 1.1.1 (1.1) Ý tưởng cốt lõi

Thuật toán Đom Đóm (Firefly Algorithm, FA) là một metaheuristic lấy cảm hứng từ hành vi phát quang và bị hút lẫn nhau của đom đóm trong tự nhiên. Trong FA, mỗi nghiệm ứng viên là một “đom đóm” với *độ sáng* (brightness) tỉ lệ với *độ phù hợp* (fitness). Một đom đóm kém sáng sẽ di chuyển về phía đom đóm sáng hơn; cường độ *hấp dẫn* suy giảm theo khoảng cách. Thành phần *nhiều* (randomization) được thêm vào để tăng khả năng thoát bẫy cục bộ và thăm dò không gian nghiệm rộng hơn. Cơ chế này tạo nên sự cân bằng *thăm dò* (exploration) và *khai thác* (exploitation), phù hợp với các bài toán đa cực trị (multimodal) (Yang, 2009; Yang & He, 2013).

### 1.1.2 (1.2) Lịch sử ngắn gọn

FA được đề xuất bởi Xin-She Yang vào giai đoạn 2008–2009 và nhanh chóng trở thành một trong các thuật toán tối ưu lấy cảm hứng tự nhiên tiêu biểu bên cạnh PSO, BA, Cuckoo Search, v.v. (Yang, 2009; Yang, 2010). Hướng phát triển gồm: FA tự điều chỉnh tham số, FA lai (hybrid) với tìm kiếm cục bộ, FA song song/đám mây, và các biến thể *rời rạc hóa* cho bài toán tổ hợp (Yang & He, 2013; Baykasoglu et al., 2014).

### 1.1.3 (1.3) Ứng dụng tiêu biểu

FA được áp dụng rộng rãi trong: tối ưu hàm chuẩn (Rastrigin, Ackley, Rosenbrock), điều chỉnh siêu tham số mô hình học máy, chọn đặc trưng, định tuyến–lập lịch, và các bài toán tổ hợp như Knapsack/Flow-Shop sau khi rời rạc hóa (*binarization* hoặc *discretization*) (Yang, 2010; Yang & He, 2013). Điểm mạnh chính: (i) công thức cập nhật đơn giản; (ii) ít siêu tham số; (iii) linh hoạt để thích nghi với cả không gian liên tục và nhị phân.

## 1.2 Cơ sở toán học của thuật toán

Xét bài toán cực tiểu hóa không ràng buộc

$$\min_{x \in \mathbb{R}^d} f(x).$$

Tại vòng lặp  $t$ , cá thể  $i$  có vị trí  $\mathbf{x}_i^{(t)} \in \mathbb{R}^d$ . Gọi  $r_{ij}^{(t)} = \|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\|_2$  là khoảng cách giữa  $i$  và  $j$ .

**Độ sáng (Brightness).** Độ sáng của mỗi đom đóm được tính từ fitness:

$$I_i = -f(\mathbf{x}_i) \quad (\text{bài toán cực tiểu})$$

Đom đóm có fitness tốt hơn (giá trị  $f$  nhỏ hơn) sẽ sáng hơn.

**Hấp dẫn suy giảm theo khoảng cách.** Hàm hấp dẫn chuẩn trong FA:

$$\beta(r_{ij}^{(t)}) = \beta_0 \exp(-\gamma(r_{ij}^{(t)})^2),$$

trong đó  $\beta_0 > 0$  là hấp dẫn tại  $r = 0$ , còn  $\gamma > 0$  điều khiển mức suy giảm theo khoảng cách (Yang, 2009).

**Quy tắc cập nhật (không gian liên tục).** Đối với mỗi đom đóm  $i$ , chỉ di chuyển về phía các đom đóm sáng hơn:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \beta \left( r_{ij}^{(t)} \right) (\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}) + \alpha \boldsymbol{\varepsilon}_i^{(t)}.$$

khi  $I_j > I_i$ . Ở đây  $\boldsymbol{\varepsilon}_i^{(t)}$  là nhiễu (thường lấy từ  $\mathcal{U}[-\frac{1}{2}, \frac{1}{2}]^d$ ), và  $\alpha$  là hệ số randomization. Nếu không có cá thể sáng hơn, nghiệm chủ yếu chịu nhiễu thăm dò.

**Xử lý biên.** Sau cập nhật, sử dụng **chiếu (clipping)** về miền hợp lệ [lower\_bound, upper\_bound] cho từng chiều.

**Rời rạc hoá cho Knapsack 0/1.** Biểu diễn nghiệm bằng  $\mathbf{b}_i^{(t)} \in \{0, 1\}^n$ . Quy tắc di chuyển:

(a) **Di chuyển có hướng (Directed movement).** Với mỗi đom đóm  $j$  sáng hơn  $i$ :

1. Xác định tập vị trí khác biệt:  $D = \{k \mid b_{i,k} \neq b_{j,k}\}$ .
2. Chọn ngẫu nhiên \*\*TỐI ĐA\*\*  $m_{\max}$  vị trí từ  $D$ .
3. Lật bit tại các vị trí được chọn để  $b_{i,k} \leftarrow b_{j,k}$ .

(b) **Nhiều ngẫu nhiên.** Với xác suất  $\alpha_{\text{flip}}$ , chọn ngẫu nhiên một vị trí và lật bit:

$$b_{i,k} \leftarrow 1 - b_{i,k}.$$

(c) **Sửa nghiệm vi phạm (Repair vs Penalty).** Benchmark hỗ trợ hai chiến lược xử lý ràng buộc:

- **Repair strategy:** Sau khi di chuyển, nếu nghiệm vi phạm ràng buộc sức chứa, loại dần các vật phẩm có tỷ số  $v_k/w_k$  thấp nhất (greedy removal) cho đến khi khả thi.
- **Penalty strategy:** Nghiệm vi phạm nhận penalty lớn trong fitness, cho phép exploration trong không gian infeasible.

**Gợi ý lựa chọn tham số.**  $\beta_0$  lớn  $\Rightarrow$  khai thác cục bộ mạnh;  $\gamma$  nhỏ  $\Rightarrow$  hút tầm xa tăng thăm dò;  $\alpha$  nên trong khoảng  $[0.1, 0.5]$  để cân bằng exploration/exploitation. Kích thước quần thể  $N$  đủ để bao phủ không gian ban đầu, thường  $N \in [20, 100]$  tuỳ độ khó/hàm mục tiêu (Yang, 2009; Yang, 2010).

## Tài liệu tham khảo (APA)

1. Baykasoglu, A., Ozsoydan, F. B., & Subulan, K. (2014). An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Systems with Applications*, 41(8), 3712–3725.
2. Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe & T. Zeugmann (Eds.), *Stochastic Algorithms: Foundations and Applications* (SAGA 2009) (pp. 169–178). Springer.
3. Yang, X.-S. (2010). *Nature-Inspired Metaheuristic Algorithms* (2nd ed.). Luniver Press.
4. Yang, X.-S., & He, X. (2013). Firefly algorithm: Recent advances and applications. *arXiv preprint arXiv:1308.3898*.

### 1.3 Chi tiết triển khai thuật toán (Mã giả)

---

**Algorithm 1** Thuật toán Đom Đóm cho tối ưu liên tục

---

```
1: Input: Hàm  $f$ , miền  $[\text{lb}, \text{ub}]^d$ ,  $n$  đom đóm,  $T$  vòng lặp,  $\alpha, \beta_0, \gamma$ .
2: Khởi tạo ngẫu nhiên  $\{\mathbf{x}_i\}_{i=1}^n$  trong miền hợp lệ.
3: Tính fitness  $f(\mathbf{x}_i)$  và độ sáng  $I_i = -f(\mathbf{x}_i)$  (cực tiểu).
4: for  $t = 1$  đến  $T$  do
5:   Tính ma trận khoảng cách  $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ .
6:   for  $i = 1$  đến  $n$  do
7:     for  $j = 1$  đến  $n$  do
8:       if  $I_j > I_i$  then            $\triangleright$  Chỉ di chuyển về phía đom đóm sáng hơn
9:          $\beta \leftarrow \beta_0 \exp(-\gamma r_{ij}^2)$ 
10:         $\mathbf{x}_i \leftarrow \mathbf{x}_i + \beta(\mathbf{x}_j - \mathbf{x}_i) + \alpha \boldsymbol{\varepsilon}$ 
11:       end if
12:     end for
13:     Chiếu  $\mathbf{x}_i$  về  $[\text{lb}, \text{ub}]^d$ , cập nhật  $f(\mathbf{x}_i), I_i$ .
14:   end for
15:   Ghi nhận nghiệm tốt nhất  $\mathbf{x}^*, f^*$ .
16: end for
17: Output:  $\mathbf{x}^*, f^*$ .
```

---

---

**Algorithm 2** Thuật toán Đom Đóm cho Knapsack 0/1

---

```
1: Input:  $\mathbf{v}, \mathbf{w}, C, n$  đom đóm,  $T$  vòng lặp,  $\alpha_{\text{flip}}$ ,  $m_{\text{max}}$ , strategy  $\in \{\text{repair}, \text{penalty}\}$ .
2: Khởi tạo  $n$  nghiệm nhị phân  $\mathbf{b}_i \in \{0, 1\}^m$ .
3: Tính fitness  $f(\mathbf{b}_i)$  và độ sáng  $I_i$ .
4: for  $t = 1$  đến  $T$  do
5:   for  $i = 1$  đến  $n$  do
6:     for  $j = 1$  đến  $n$  do
7:       if  $I_j > I_i$  then
8:          $D \leftarrow \{k \mid b_{i,k} \neq b_{j,k}\}$ 
9:         Chọn ngẫu nhiên TÔI ĐA  $m_{\text{max}}$  vị trí từ  $D$ 
10:        Với mỗi vị trí được chọn:  $b_{i,k} \leftarrow b_{j,k}$ 
11:       end if
12:     end for
13:     if  $\text{rand}() < \alpha_{\text{flip}}$  then
14:       Chọn ngẫu nhiên vị trí  $k$  và lật:  $b_{i,k} \leftarrow 1 - b_{i,k}$ 
15:     end if
16:     if strategy == repair then
17:        $\mathbf{b}_i \leftarrow \text{GreedyRepair}(\mathbf{b}_i)$                                  $\triangleright$  Loại items có  $v/w$  thấp
18:     else
19:       Áp dụng penalty nếu vi phạm capacity
20:     end if
21:     Cập nhật  $f(\mathbf{b}_i)$ ,  $I_i$ .
22:   end for
23:   Ghi nhận nghiệm tốt nhất  $\mathbf{b}^*$ ,  $f^*$ .
24: end for
25: Output:  $\mathbf{b}^*$ , giá trị balo =  $-f^*$ .
```

---

## 1.4 Thuật toán dùng để so sánh

Để đặt Thuật toán Đom Đóm (FA) vào một bối cảnh hợp lý, chúng tôi so sánh với ba thuật toán metaheuristic cổ điển: Hill Climbing (HC), Simulated Annealing (SA) và Genetic Algorithm (GA). Tất cả đều được hiện thực trong cùng một khung mã Python, dùng chung: (i) giao diện bài toán, (ii) bộ sinh nghiệm khởi tạo, (iii) cách ghi log kết quả và (iv) các script phân tích/visualize.

**Hill Climbing (HC).** HC là baseline tham lam địa phương: giữ một nghiệm hiện tại, sinh lân cận và chỉ chấp nhận nghiệm tốt hơn. Khi không cải thiện sau một số bước, thực hiện restart.

**Cấu hình đại diện:**

- **Rastrigin (liên tục):** lân cận được sinh bằng perturbation Gaussian/Uniform trên từng chiều với bước nhảy cố định; dùng một số lượng lân cận cố định mỗi vòng lặp, sau đó chọn best improvement. Có cơ chế restart sau một số vòng không cải thiện.
- **Knapsack (0/1):** lân cận sinh bằng cách lật một số bit trong vector nhị phân (bit-flip). Restart khi bị kẹt quá lâu trong plateau.

**Simulated Annealing (SA).** SA dùng cùng cấu trúc lân cận với HC nhưng chấp nhận nghiệm xấu hơn với xác suất  $\exp(-\Delta f/T)$ , trong đó  $T$  giảm dần theo lịch làm nguội hình học:  $T_{k+1} = \text{cooling\_rate} \cdot T_k$ .

### Cấu hình đại diện:

- **Rastrigin:** nhiệt độ khởi tạo  $T_0$  tương đối cao, lịch làm nguội hình học (cooling\_rate gần 1), bước nhảy tương đương với HC để so sánh công bằng.
- **Knapsack:** chỉ điều chỉnh  $T_0$  và cooling\_rate, sử dụng cùng cơ chế lân cận bit-flip với HC.

**Genetic Algorithm (GA).** GA duy trì một quần thể cá thể, áp dụng tournament selection, crossover và mutation; luôn có elitism giữ lại một số cá thể tốt nhất qua thế hệ.

### Cấu hình và operator đúng với code:

- **Rastrigin (liên tục):**

- *Crossover:* sử dụng Simulated Binary Crossover (SBX), **không** phải one-point/two-point. SBX sinh con liên tục nằm trong vùng lân cận hai cha mẹ, phù hợp không gian thực.
- *Mutation:* perturbation từng chiều với xác suất mutation\_rate; các giá trị mutation\_rate trong benchmark được đặt xấp xỉ  $1/d$  (0.10 cho  $d = 10$ , 0.03 cho  $d = 30$ , 0.02 cho  $d = 50$ ).
- *Quần thể:* pop\_size tăng dần theo dimension ( $40 \rightarrow 60 \rightarrow 80$ ) để bù lại độ khó tăng.
- *Selection:* tournament\_size lần lượt là 3, 5, 7 cho ba cấu hình; crossover\_rate giữ khoảng 0.9, luôn bật elitism.

- **Knapsack (rời rạc):**

- *Crossover:* uniform crossover trên bit – mỗi bit con được chọn độc lập từ cha hoặc mẹ theo một xác suất, phù hợp với biểu diễn 0/1.
- *Mutation:* bit-flip với xác suất  $\text{mutation\_rate} = 1/n$  (chuẩn trong GA nhị phân).
- *Quần thể:* pop\_size tăng theo kích thước bài toán (30 cho  $n = 50, 100$ ; 40 cho  $n = 200$ ); tournament\_size=3; luôn dùng elitism.

**Firefly Algorithm (FA).** **Rastrigin (liên tục):** sử dụng biến thể FA chuẩn trong không gian liên tục, với các tham số ( $n_{\text{fireflies}}, \alpha, \beta_0, \gamma$ ) được chọn *riêng cho từng cấu hình*:

- quick\_convergence ( $d = 10$ ):  $n_{\text{fireflies}} = 40$ ,  $\alpha = 0.18$ ,  $\gamma = 0.02$ .
- multimodal\_escape ( $d = 30$ ):  $n_{\text{fireflies}} = 60$ ,  $\alpha = 0.20$ ,  $\gamma = 0.01$ .
- scalability ( $d = 50$ ):  $n_{\text{fireflies}} = 80$ ,  $\alpha = 0.22$ ,  $\gamma = 0.008$ .

$\beta_0$  luôn đặt bằng 1.0. Xu hướng chung: dimension càng cao thì quần thể và mức nhiễu  $\alpha$  tăng lên một chút để tăng exploration, trong khi  $\gamma$  giảm để lực hút không bị “tắt” quá nhanh khi  $r$  tăng theo  $\sqrt{d}$ .

**Knapsack:** dùng biến thể rời rạc hoá đã mô tả ở *Cơ sở toán học*, trong đó:

- Di chuyển có hướng (directed movement) lật tối đa một số bit mỗi bước (`max_flips_per_move=3`) để tiến gần tới firefly sáng hơn.
- Nhiều ngẫu nhiên bit-flip với xác suất  $\alpha_{\text{flip}} = 0.2$ .
- Luôn dùng chiến lược repair `greedy_remove` (loại các vật phẩm có  $v_k/w_k$  thấp nhất) để đảm bảo feasibility; penalty được hỗ trợ ở mức framework nhưng không kích hoạt trong cấu hình benchmark chính.

Quy mô quần thể FA cho Knapsack tăng theo  $n$ : với  $n \leq 100$  dùng  $n_{\text{fireflies}} = 30$ , với  $n = 200$  dùng  $n_{\text{fireflies}} = 40$ .

**Lưu ý về tham số.** Các tham số *không* hoàn toàn “cố định cho từng family bài toán” như cách diễn đạt ngắn gọn ban đầu, mà được chọn thủ công **theo từng kịch bản**:

- Với Rastrigin: mỗi cấu hình `quick_convergence` (10D), `multimodal_escape` (30D), `scalability` (50D) có bộ tham số riêng cho FA, SA, HC, GA (`pop_size`, `n_fireflies`, `mutation_rate`, `initial_temp`, `num_neighbors`, ...).
- Với Knapsack: tham số thay đổi theo kích thước  $n$  (50, 100, 200) – budget, quần thể FA/GA được scale theo  $n$ , `mutation_rate` của GA đặt đúng bằng  $1/n$ , nhưng trong cùng một kích thước thì tham số cố định cho mọi instance-type và mọi run.

Tuy nhiên, trong *mỗi* cấu hình con (ví dụ: Rastrigin `dim=30`, hoặc Knapsack  $n = 100$ ), bộ tham số được giữ nguyên cho tất cả thuật toán trong mọi lần chạy và **không** được tune theo từng instance cụ thể. Điều này khiến kết quả vẫn mang tính “out-of-the-box theo từng scenario” chứ không phải đã tối ưu hoá đến mức per-instance.

## 1.5 Bài toán kiểm tra và tiêu chí đánh giá

**Bài toán Rastrigin (liên tục).** Rastrigin là một hàm benchmark đa cực trị, phi lồi, thường dùng để kiểm tra khả năng tối ưu hoá toàn cục của các thuật toán metaheuristic. Hàm có nhiều cực tiểu địa phương và một cực tiểu toàn cục tại  $\mathbf{0}$  với  $f(\mathbf{0}) = 0$  :contentReference[oaicite:2]

Trong benchmark này, chúng tôi dùng 3 cấu hình:

- **quick\_convergence:**  $d = 10$ . Budget đánh giá hàm ở mức  $O(10^4)$ , tương ứng với vài trăm vòng lặp cho mỗi thuật toán. Mục tiêu: so sánh tốc độ hội tụ giai đoạn đầu trên một bài toán tương đối “dễ”.
- **multimodal\_escape:**  $d = 30$ . Budget tăng lên ( $c$  vài  $\times 10^4$  đánh giá) để tạo điều kiện cho việc thoát bẫy cục bộ. Mục tiêu: quan sát khả năng xử lý đa cực trị khi không gian nghiệm mở rộng.
- **scalability:**  $d = 50$ . Budget lớn nhất trong ba cấu hình nhưng vẫn bị giới hạn; đây là bài test “khó”, chủ yếu để xem xu hướng suy giảm hiệu năng khi tăng chiều.

Tất cả trường hợp đều dùng cùng miền tìm kiếm chuẩn của Rastrigin, với giới hạn biên cố định trên từng chiều.

## Bài toán Knapsack 0/1 (rời rạc).

- Số vật phẩm:  $n \in \{50, 100, 200\}$ .
- Instance types: *uncorrelated*, *weakly correlated*, *strongly correlated*, *subset-sum* (theo phân loại kinh điển trong knapsack literature).
- Mỗi cấu hình (kích thước  $\times$  loại instance): 3 seeds độc lập, mỗi seed chạy 10 lần  $\Rightarrow$  30 runs/thuật toán/cấu hình.
- Với  $n \leq 100$ : tính nghiệm tối ưu bằng Dynamic Programming (DP) để làm ground truth; với  $n = 200$  so sánh theo gap tương đối so với best-known trong các thuật toán.

## Chỉ số đánh giá. Rastrigin:

- *ECDF Fixed-Target*: empirical CDF của số lượng target đạt được trong một budget cố định (chuẩn Dolan–Moré) :contentReference[oaicite:3]index=3.
- *Fixed-Budget CDF*: phân bố sai số cuối cùng tại các mức budget khác nhau.
- *ERT (Expected Running Time)*: số lượng đánh giá kỳ vọng để đạt một target nhất định (khi thành công chưa đạt 100%, ERT được tính từ các run thành công).
- *Performance profiles*: xác suất một thuật toán có performance ratio  $\rho_{p,s} \leq \tau$  so với best solver, theo định nghĩa của Dolan–Moré :contentReference[oaicite:4]index=4.
- *Data profiles*: xác suất giải được bài toán trong một budget chuẩn hoá, theo Moré–Wild :contentReference[oaicite:5]index=5.

## Knapsack:

- **Optimality gap (%)**:

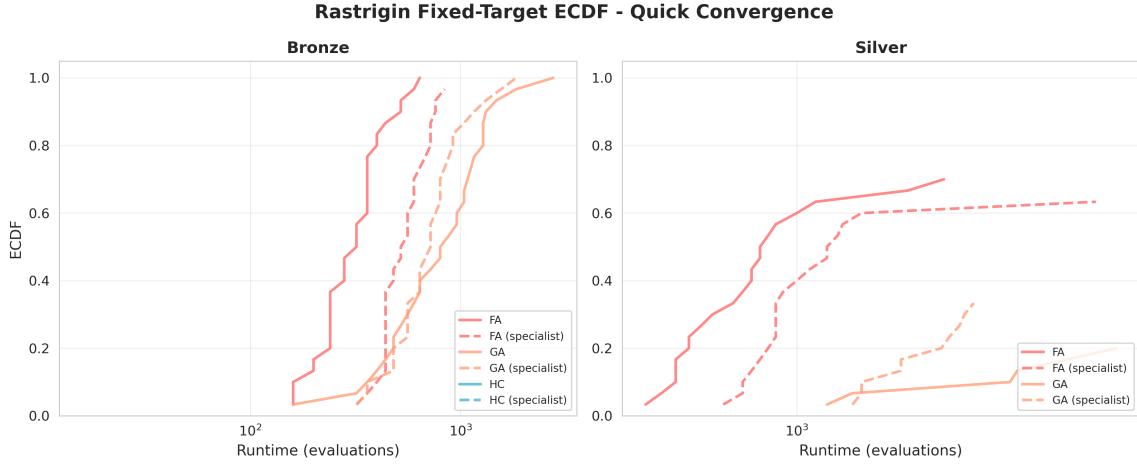
$$\text{gap} = \frac{z^* - z_{\text{alg}}}{z^*} \times 100\%,$$

với  $z^*$  là optimum (hoặc best-known với  $n = 200$ ).

- **Success rate**: tần suất gap  $\leq$  các ngưỡng cho trước (ví dụ: 10%).
- **Feasibility rate**: tần suất nghiệm thỏa ràng buộc sức chứa (đặc biệt quan trọng cho chiến lược penalty).
- **Thống kê**: kiểm định Wilcoxon signed-rank cho so sánh cặp đôi FA/GA/HC/SA trên từng cấu hình, và Copeland ranking để tổng hợp số lần thắng-thua. (Friedman/Nemenyi không được triển khai đầy đủ nên không báo cáo ở đây.)

## 1.6 Kết quả benchmark và phân tích

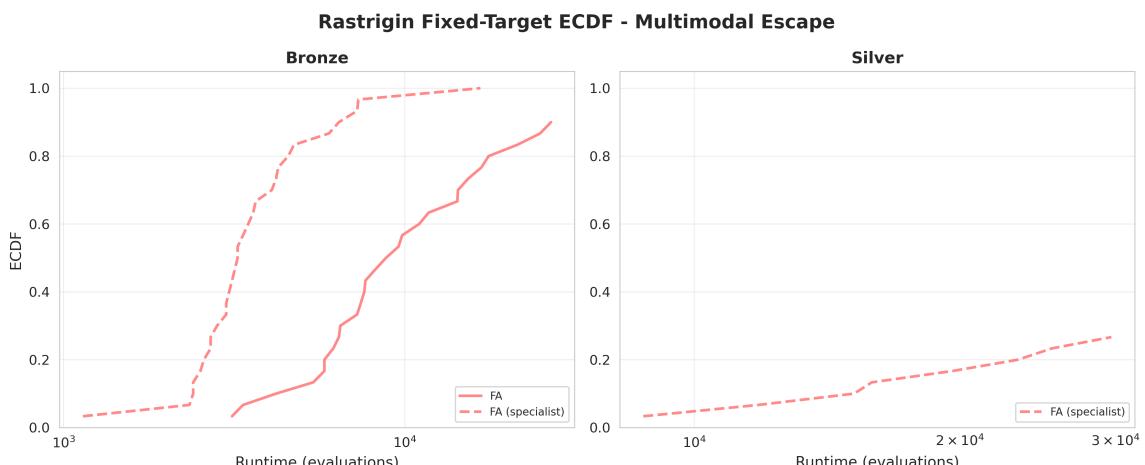
### 1.6.1 Rastrigin: mức độ tiệm cận nghiệm tối ưu và đặc trưng hội tụ



Hình 1: ECDF cho quick\_convergence (dim=10).

**ECDF theo ngưỡng mục tiêu.** Đối với cấu hình 10 chiều, ECDF cho thấy:

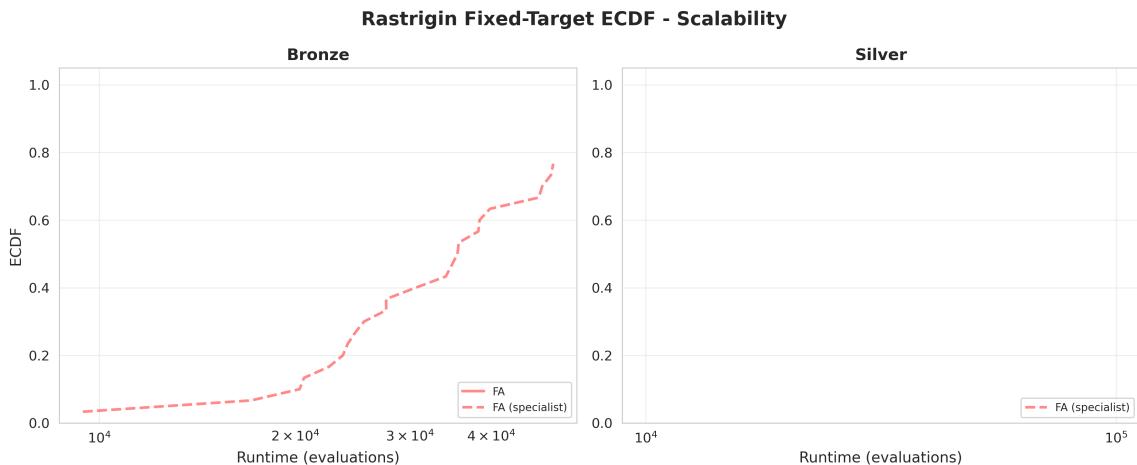
- Ở mức Bronze, cả FA và GA đều đạt tỉ lệ thành công rất cao: các đường ECDF tiệm cận 1 khi ngân sách tiến dần dần tới khoảng  $10^3$  lần đánh giá. FA có xu hướng đạt cùng mức ECDF với chi phí nhỏ hơn đôi chút, thể hiện lợi thế nhẹ về tốc độ hội tụ.
- HC chậm hơn rõ rệt: đường ECDF nằm thấp và tăng chậm, phản ánh việc thuật toán này thường bị kẹt trong các cực trị địa phương của Rastrigin ngay cả với ngưỡng Bronze.
- Ở mức Silver, sự khác biệt trở nên rõ ràng: chỉ FA (đặc biệt là cấu hình *specialist*) duy trì được ECDF đáng kể (xấp xỉ 0.6–0.7), trong khi GA và HC hầu như không chạm được ngưỡng trong ngân sách đang xét, các đường ECDF gần như bám sát trực hoành. FA vì vậy là thuật toán duy nhất còn hoạt động hiệu quả khi yêu cầu độ chính xác cao hơn trên Rastrigin 10 chiều.



Hình 2: ECDF cho multimodal\_escape (dim=30).

Khi tăng lên 30 chiều, hình dạng ECDF cho thấy độ khó tăng rõ rệt:

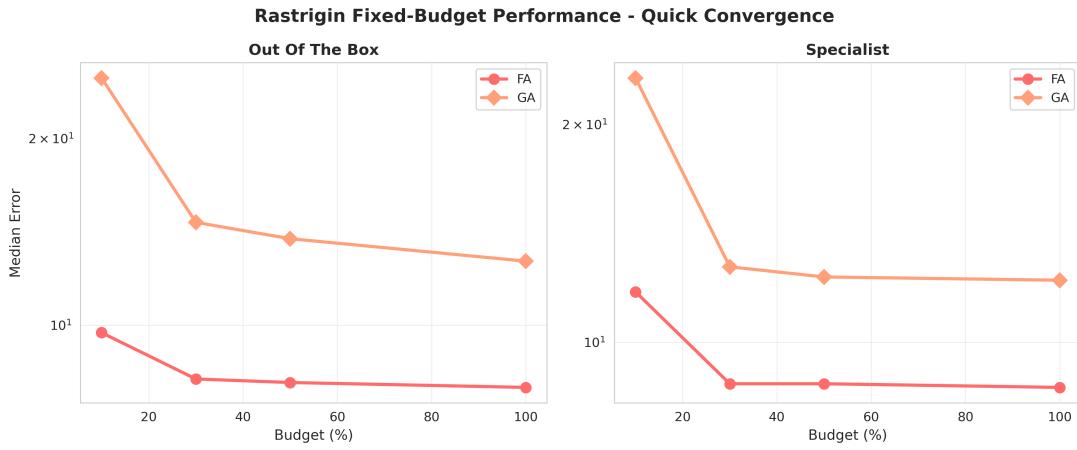
- Ở mức Bronze, chỉ còn các biến thể của FA (bản gốc và *specialist*) đạt được tỉ lệ thành công đáng kể. Cả hai đường cong đều bị dịch sang phải, cần tối cỡ  $10^4$  đánh giá để ECDF tiệm cận 1; cấu hình *specialist* đạt cùng mức ECDF với chi phí thấp hơn, cho thấy việc tinh chỉnh tham số giúp cải thiện rõ rệt khả năng thoát bẫy đa cực trị.
- Các thuật toán còn lại (GA, HC, SA) hầu như không đạt được ngưỡng Bronze trong ngân sách đã chọn nên không xuất hiện trên đồ thị; về thực chất, chúng thất bại gần như hoàn toàn trên Rastrigin 30 chiều ở mức mục tiêu này.
- Ở mức Silver, độ khó tăng vọt: chỉ FA *specialist* đạt được một phần nhỏ số lần chạy (ECDF dừng dưới 0.3 ngay cả ở rìa phải trực hoành), trong khi FA gốc và các thuật toán khác không có lần chạy nào chạm ngưỡng. Điều này cho thấy từ 30 chiều trở lên, mức Silver của Rastrigin đã vượt quá khả năng của hầu hết thuật toán trong bộ benchmark.



Hình 3: ECDF cho scalability (dim=50, trực log).

Ở cấu hình 50 chiều, Rastrigin trở thành một bài toán đặc biệt thách thức trong bối cảnh ngân sách giới hạn:

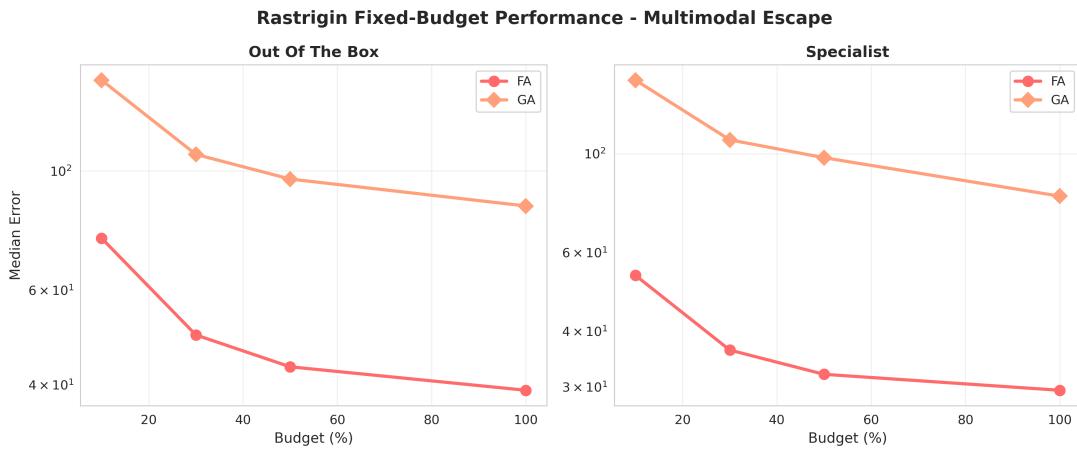
- Ở mức Bronze, chỉ còn FA *specialist* đạt được một tỉ lệ thành công trung bình khá khiêm tốn: đường ECDF tăng rất chậm và chỉ tiệm cận quanh 0.7 khi ngân sách tiến gần  $4 \times 10^4$  đánh giá. FA gốc và các thuật toán khác không đạt được ngưỡng nên không để lại dấu vết trên đồ thị.
- Ở mức Silver, toàn bộ các đường ECDF nằm tại 0, tương đương việc không thuật toán nào đạt được ngưỡng Silver trên Rastrigin 50 chiều trong ngân sách tối đa. Rastrigin high-dimensional với yêu cầu độ chính xác cao vì thế có thể xem là “ngoài tầm với” đối với tập thuật toán đang xét.



Hình 4: Rastrigin – fixed-budget performance (quick\_convergence,  $d = 10$ ). Trục tung là sai số trung vị (median error), trục hoành là tỉ lệ ngân sách.

**Fixed-Budget Performance: sai số cuối cùng dưới các mức ngân sách cố định.** Với cấu hình quick\_convergence (10 chiều), đường cong fixed-budget cho thấy:

- Cả FA và GA đều có sai số trung vị giảm đơn điệu khi tăng ngân sách, nhưng FA luôn giữ mức sai số thấp hơn rõ rệt. Ở mọi mức budget (10%, 30%, 50%, 100%), đường cong của FA nằm dưới GA khoảng gần một bậc độ lớn trên thang log.
- Lợi ích của việc tăng ngân sách thể hiện mạnh mẽ giữa 10% và 30%; sau khoảng 50% ngân sách, cả hai thuật toán đều rơi vào vùng “diminishing returns”, sai số giảm thêm rất ít.
- Cấu hình *specialist* cải thiện nhẹ cho cả hai thuật toán, chủ yếu ở ngân sách nhỏ (10–30%). Tuy nhiên, thứ hạng tương đối không đổi: FA vẫn là thuật toán cho sai số cuối cùng thấp nhất, GA ổn định nhưng kém hơn toàn dải ngân sách.



Hình 5: Rastrigin – fixed-budget performance (multimodal\_escape,  $d = 30$ ).

Khi tăng lên cấu hình multimodal\_escape (30 chiều), độ khó tăng rõ rệt:

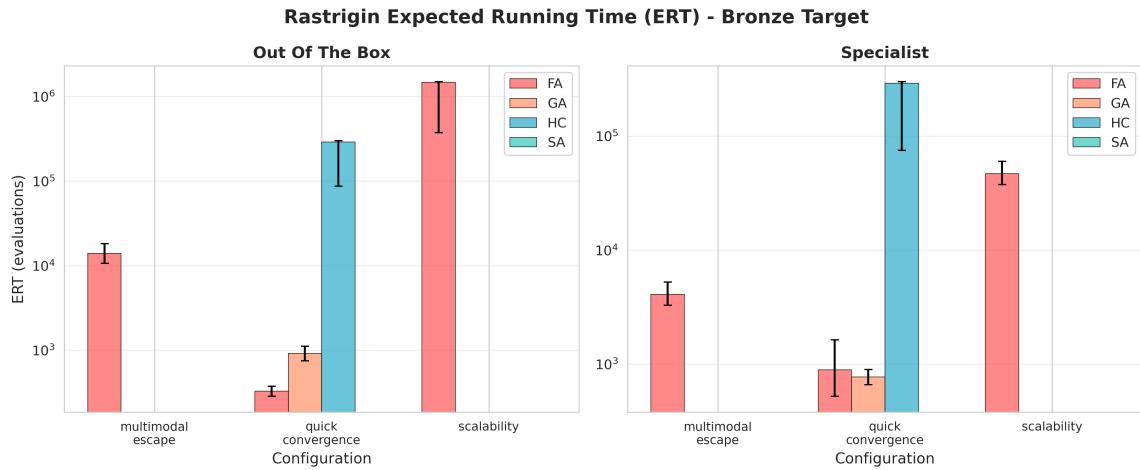
- Mức sai số trung vị của cả FA và GA đều cao hơn đáng kể so với  $d = 10$ , ngay cả ở 100% ngân sách. Điều này phù hợp với ECDF: nhiều lần chạy không chạm được các target Silver dù đã dùng hết ngân sách.
- FA tiếp tục giữ lợi thế ổn định: ở mọi mức budget, đường cong của FA nằm thấp hơn GA, và khoảng cách giữa hai thuật toán vẫn tương đối lớn.
- Tinh chỉnh *specialist* giúp FA giảm thêm vài đơn vị sai số trên toàn dải ngân sách, trong khi GA cũng cải thiện nhưng vẫn bị bỏ xa. Có thể hiểu rằng ở 30 chiều, FA không chỉ đạt tỉ lệ thành công cao hơn (ECDF) mà còn cho chất lượng nghiệm cuối cùng tốt hơn trong khung fixed-budget.



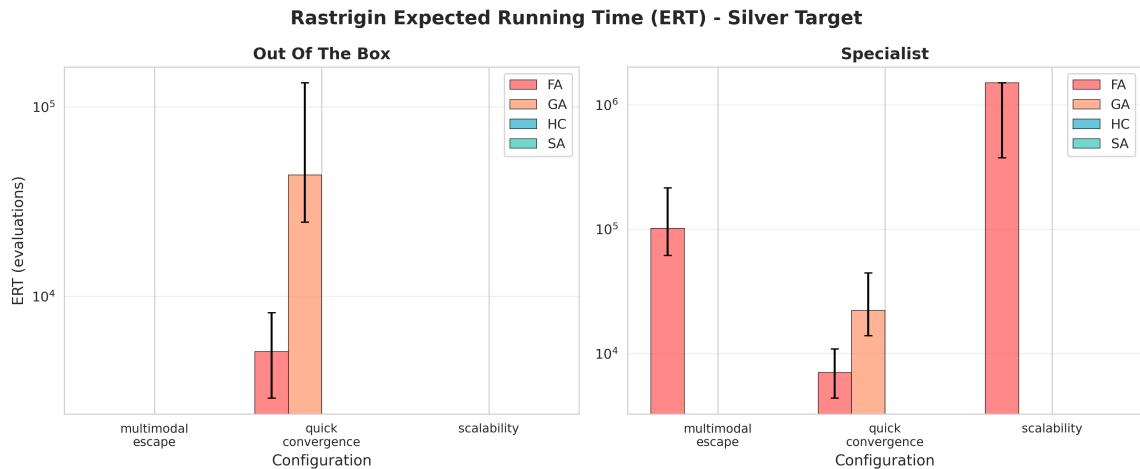
Hình 6: Rastrigin – fixed-budget performance (scalability,  $d = 50$ ).

Ở cấu hình scalability (50 chiều), bức tranh trở nên bi quan hơn:

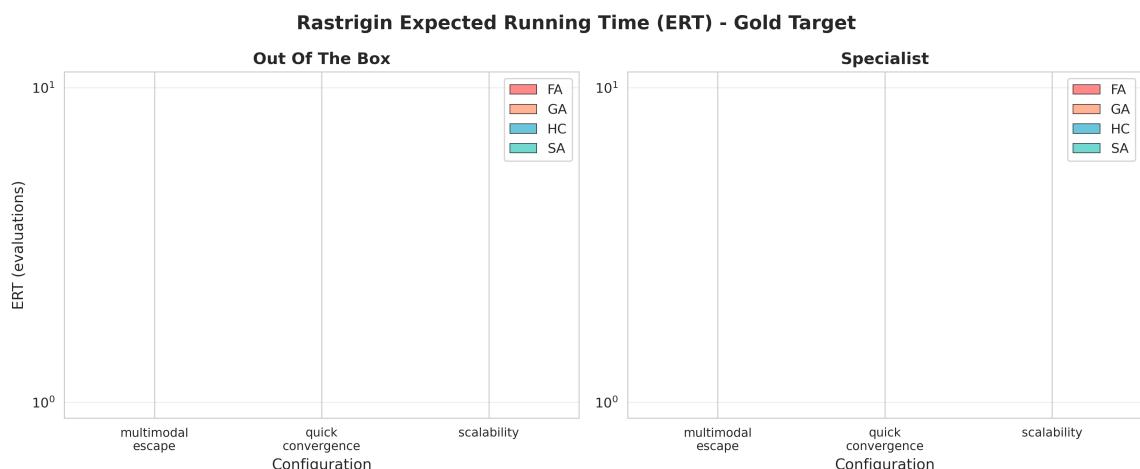
- Các đường cong cho thấy sai số trung vị vẫn rất cao ngay cả khi dùng 100% ngân sách: FA chỉ giảm từ khoảng  $\sim 1.7 \times 10^2$  xuống dưới  $10^2$ , GA dao động quanh vùng  $[1.8, 3] \times 10^2$ . Điều này nhất quán với ECDF: gần như không có lần chạy nào đạt được các target Bronze/Silver ở 50 chiều.
- FA tiếp tục vượt trội GA trên toàn bộ dải budget: với cùng một ngân sách, nghiệm trung vị của FA luôn tốt hơn đáng kể. Nói cách khác, nếu buộc phải chọn giữa hai thuật toán trong bối cảnh high-dimensional Rastrigin, FA luôn là lựa chọn “ít tệ hơn”.
- Cấu hình *specialist* giúp FA cải thiện thêm một chút (đặc biệt ở ngân sách thấp), nhưng không thay đổi bản chất vấn đề: với dimension 50 và ngân sách hiện tại, cả hai thuật toán đều đang hoạt động trong vùng “chưa hội tụ”, sai số tuyệt đối vẫn lớn so với nghiệm tối ưu.



Hình 7: Rastrigin – ERT tới ngưỡng Bronze.



Hình 8: Rastrigin – ERT tới ngưỡng Silver.

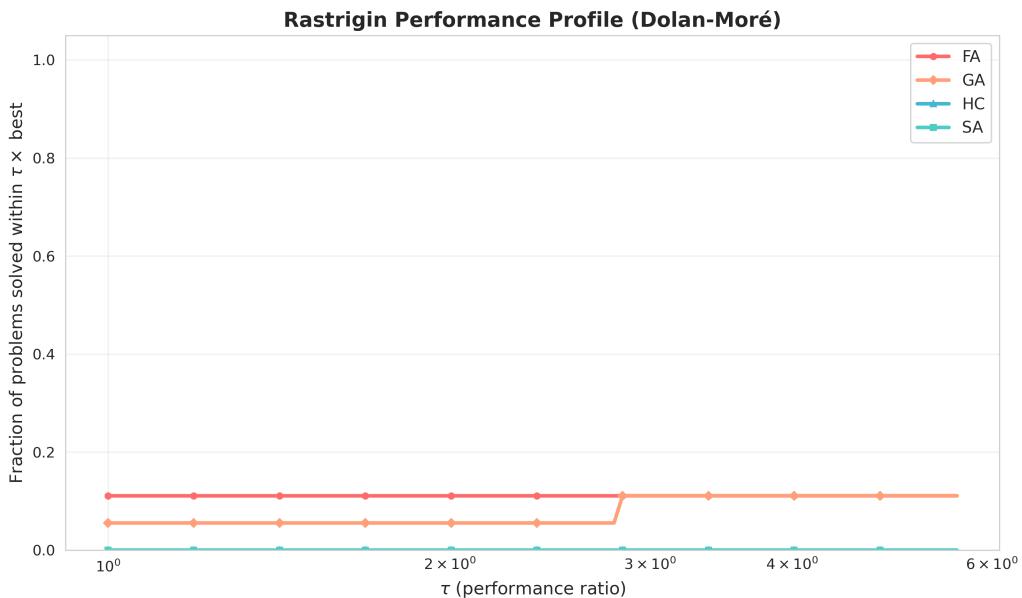


Hình 9: Rastrigin – ERT tới ngưỡng Gold (không có thuật toán nào đạt target).

**ERT, performance profiles và data profiles.** ERT (Expected Running Time) chỉ có ý nghĩa khi tồn tại số lượng đủ lớn các lần chạy thành công trên target đang xét. Các

biểu đồ ERT cho Rastrigin cho thấy:

- **Ngưỡng Bronze.** Ở cấu hình *quick\_convergence* (10 chiều), cả FA, GA và HC đều đạt được target Bronze. FA có ERT nhỏ nhất (cỡ  $10^2$ – $10^3$  đánh giá), GA chậm hơn khoảng một bậc, còn HC chậm hơn rất nhiều bậc và do đó chỉ đóng vai trò baseline. SA hầu như không đạt được target. Ở hai cấu hình khó hơn (*multimodal\_escape* 30 chiều và *scalability* 50 chiều), chỉ có FA đạt Bronze; các thuật toán còn lại không có run thành công nên không xuất hiện trên biểu đồ. Việc tinh chỉnh (*specialist*) giúp ERT của FA giảm đáng kể nhưng giá trị tuyệt đối vẫn nằm trong vùng từ vài chục nghìn tới hàng triệu đánh giá.
- **Ngưỡng Silver.** Với Silver, bức tranh càng khắt khe hơn. Ở 10 chiều, FA và GA vẫn đạt target nhưng ERT của GA lớn hơn FA khá rõ, phản ánh việc GA cần nhiều đánh giá hơn để hội tụ tới mức sai số sâu hơn. Ở 30 chiều, chỉ còn FA (đặc biệt là cấu hình *specialist*) đạt Silver với ERT rất lớn (khoảng  $10^5$  đánh giá), và ở 50 chiều chỉ FA *specialist* đạt Silver với ERT lên tới cỡ  $10^6$ . Điều này nhất quán với ECDF: Silver trên Rastrigin high-dimensional là một target cực khó.
- **Ngưỡng Gold.** Trên cả ba cấu hình và bốn thuật toán, không có run nào đạt được target Gold trong ngân sách cho phép. Vì vậy Hình 9 thực chất minh họa một trường hợp “ERT không xác định”: success rate bằng 0, và mọi so sánh ERT tại ngưỡng Gold đều vô nghĩa.

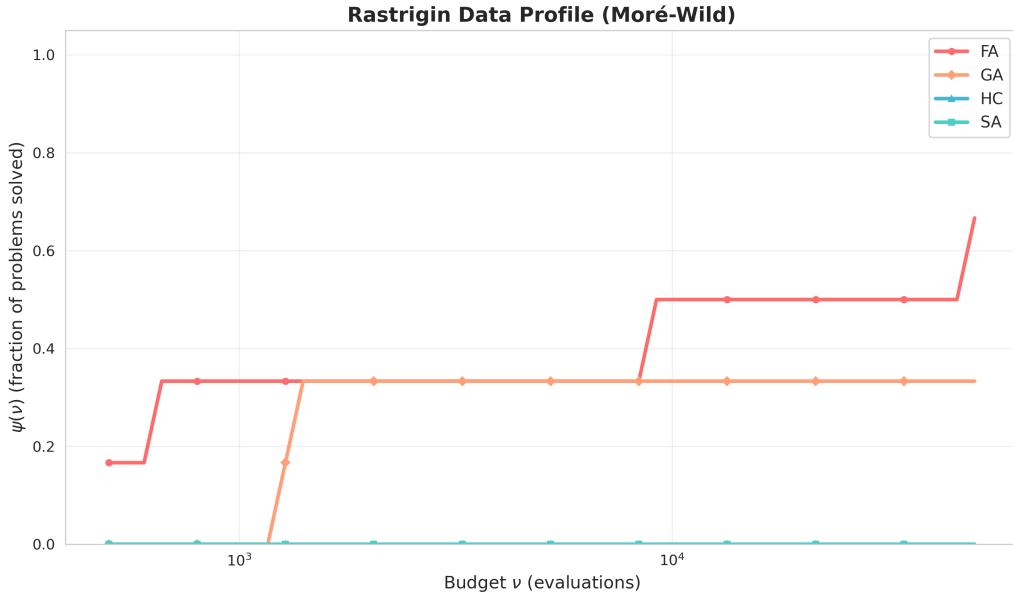


Hình 10: Performance profiles cho Rastrigin (Dolan–Moré).

Performance profile mô tả, với mỗi tỷ lệ hiệu năng  $\tau$ , tỷ lệ các bài toán mà một thuật toán có thời gian chạy không vượt quá  $\tau$  lần solver tốt nhất. Kết quả cho Rastrigin cho thấy:

- Đường cong của FA nằm cao hơn GA tại  $\tau = 1$ , tức là trong số rất ít các cặp (cấu hình, target) được giải thành công, FA thường là thuật toán nhanh nhất về số lần đánh giá.

- Khi tăng  $\tau$  lên khoảng 3, đường của GA mới bắt kịp FA; hai đường đều tiệm cận một mức trần thấp (khoảng 0,1), phản ánh thực tế là chỉ một phần rất nhỏ các bài toán trong bộ test được giải bởi bất kỳ thuật toán nào.
- HC và SA hầu như không xuất hiện trên performance profile vì không giải được target nào trong tập Rastrigin ở các mức ngân sách đã chọn.



Hình 11: Data profiles cho Rastrigin (Moré–Wild).

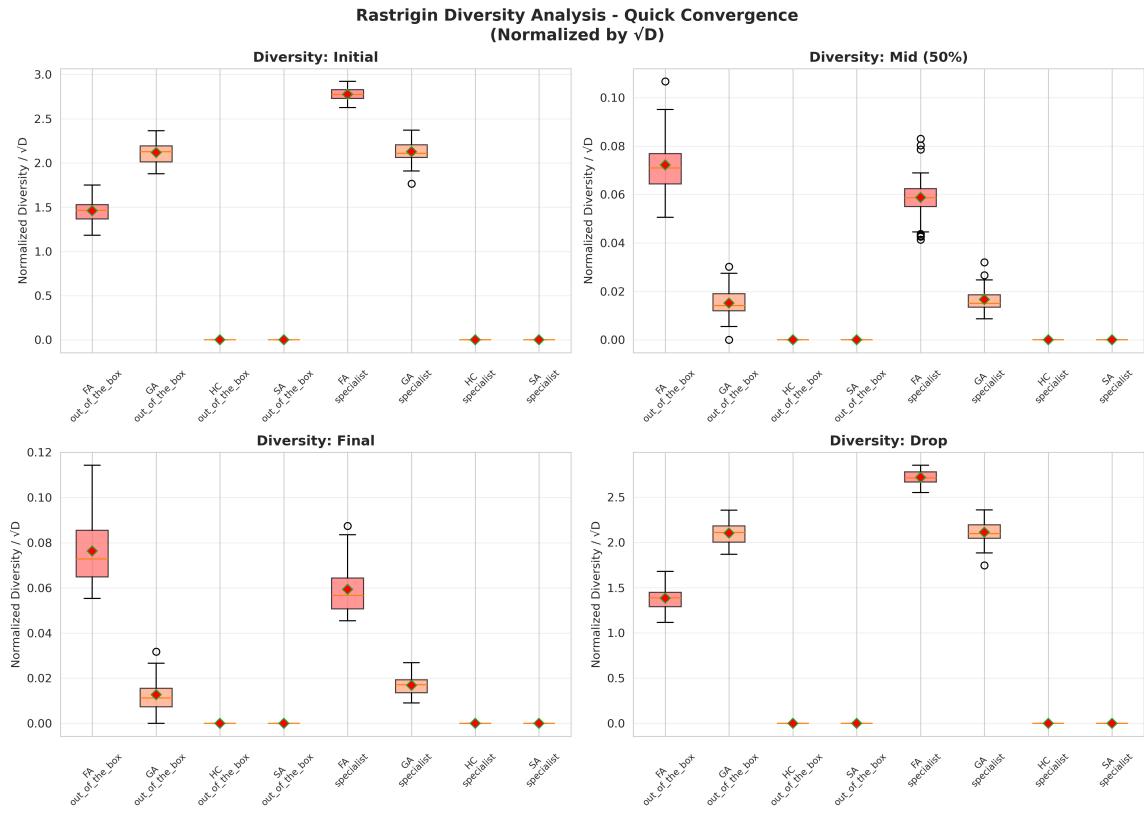
Data profile  $\psi(\nu)$  biểu diễn tỷ lệ bài toán được giải trong một ngân sách chuẩn hoá  $\nu$  cho trước. Trên Rastrigin:

- FA luôn là thuật toán có coverage cao nhất: với ngân sách rất nhỏ (vài trăm đánh giá), FA đã giải được khoảng 1/6 số bài toán; khi tăng ngân sách lên cỡ  $10^4$ – $3 \times 10^4$  đánh giá, coverage tăng dần lên gần 2/3.
- GA chỉ bắt đầu giải được bài toán khi ngân sách tăng lên mức trung bình và nhanh chóng đạt trần quanh mức 1/3 số bài toán; sau đó đường cong của GA hầu như phẳng, không hưởng lợi nhiều từ việc tăng ngân sách.
- HC và SA không giải được bất kỳ instance nào trong bộ target Rastrigin dưới các mức ngân sách đã xét, nên đường data profile gần như dính sát trực hoành.
- Ngay cả ở ngân sách lớn nhất, không có thuật toán nào đạt coverage gần 1; điều này cũng cố nhận định rằng Rastrigin high-dimensional (đặc biệt với các target Silver/Gold) về cơ bản là quá khó trong khung ngân sách hiện tại.

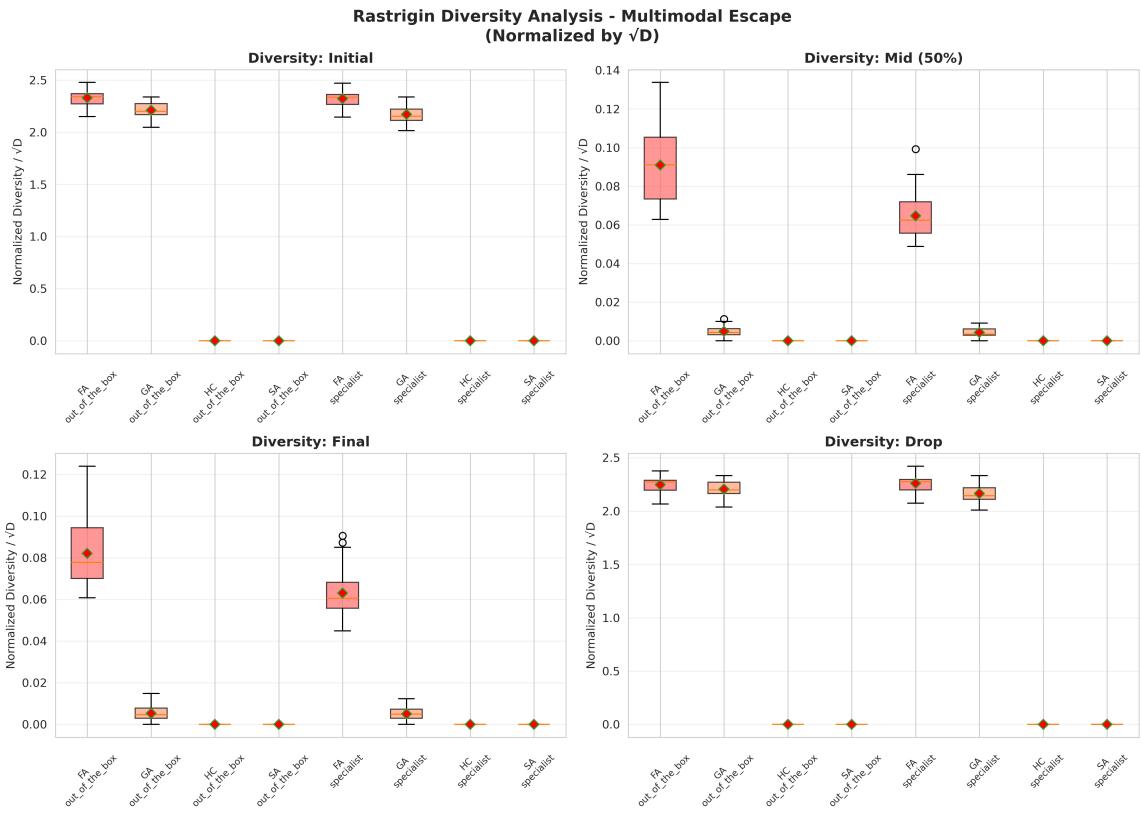
### 1.6.2 Rastrigin: đa dạng quần thể và hiện tượng dừng sớm (stagnation)

**Đa dạng quần thể theo thời gian.** Đa dạng quần thể được đo bằng khoảng cách Euclid trung bình giữa các cá thể, chuẩn hoá theo  $\sqrt{D}$  để cho phép so sánh giữa các chiều khác nhau. Hình 12–14 hiển thị boxplot của bốn thời điểm: *initial* (ngay sau khởi

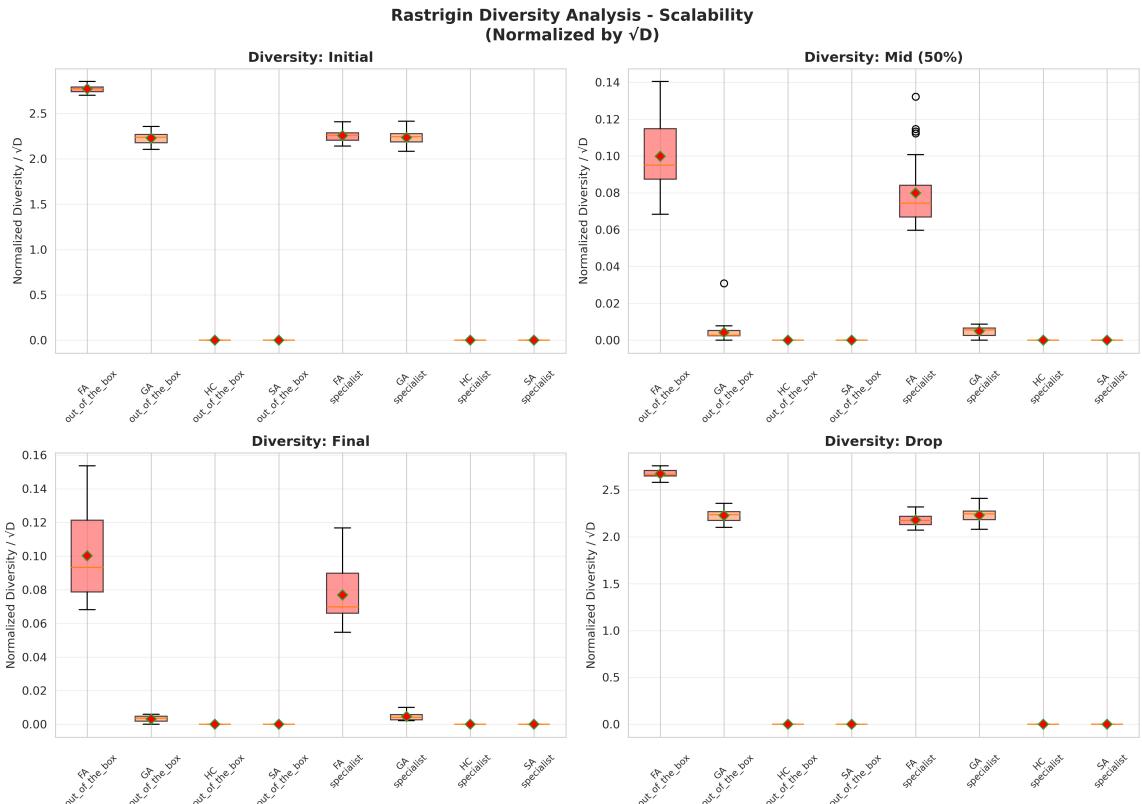
tạo), *mid* (50% ngân sách), *final* (kết thúc chạy) và *drop* (mức giảm đa dạng từ *initial* tới *final*).



Hình 12: Rastrigin – phân tích đa dạng quần thể (quick\_convergence,  $D = 10$ ).



Hình 13: Rastrigin – phân tích đa dạng quần thể (multimodal\_escape,  $D = 30$ ).



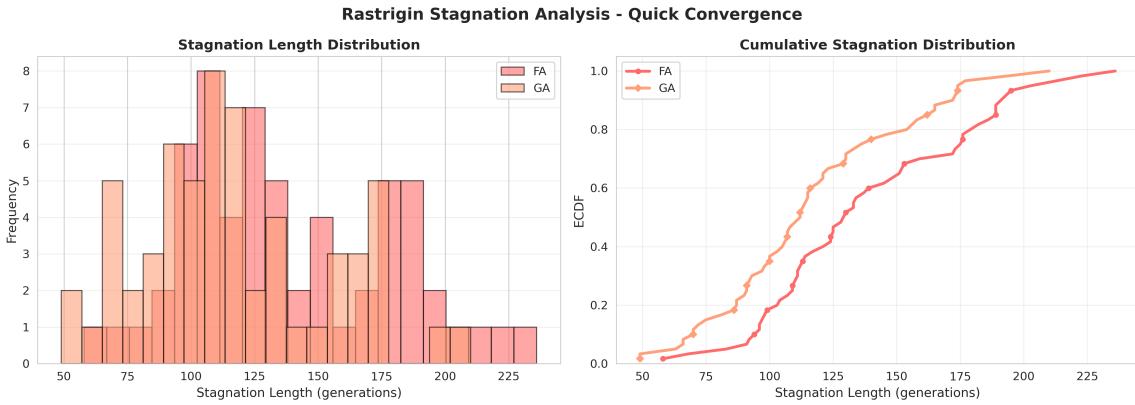
Hình 14: Rastrigin – phân tích đa dạng quần thể (scalability,  $D = 50$ ).

Các quan sát chính:

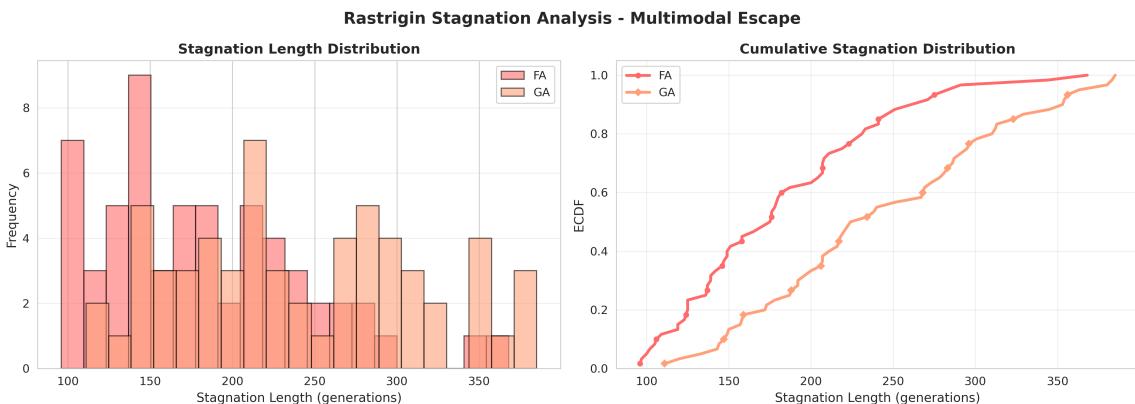
- **Khởi tạo.** Trên cả ba cấu hình, FA và GA đều bắt đầu với mức đa dạng khá cao ( $\approx 2,0\text{--}2,8$  sau chuẩn hoá). Với cấu hình quick\\_convergence, GA (đặc biệt bản out-of-the-box) có đa dạng khởi tạo lớn hơn FA, trong khi FA *specialist* có mức cao nhất. Điều này cho thấy lợi thế của FA trên Rastrigin không đến từ việc “rải quần thể rộng hơn ngay từ đầu” mà chủ yếu do động lực cập nhật trong quá trình tối ưu.
- **Giữa quá trình (50% ngân sách).** Ở cả ba cấu hình, GA mất đa dạng rất nhanh: median diversity của GA chỉ còn  $\approx 0,01\text{--}0,02$ , trong khi FA vẫn duy trì quanh  $\approx 0,06\text{--}0,10$ . FA *specialist* có đa dạng mid hơi thấp hơn FA gốc nhưng vẫn cao hơn GA rõ rệt. Tức là GA trải qua hiện tượng *premature convergence*: áp lực chọn lọc và lai ghép làm quần thể co cụm sớm quanh một vài basin, trong khi FA vẫn giữ được một “vòng đai” nghiệm khác biệt hơn.
- **Cuối quá trình.** Tới thời điểm kết thúc, đa dạng của GA gần như bằng 0 ở mọi cấu hình, cho thấy quần thể gần như đồng nhất. FA vẫn giữ một mức đa dạng dương đáng kể (đặc biệt ở multimodal\_escape và scalability), dù đã giảm hơn 90% so với ban đầu. Điều này phù hợp với quan sát ECDF: FA vẫn còn khả năng “nhúc nhích” sang các basin khác ở cuối run, trong khi GA gần như bị khoá cứng trong một vùng nghiệm.
- **Mức giảm đa dạng (drop).** Cả FA và GA đều có *drop* lớn (trên 90% initial diversity), riêng GA thường tụt về gần 0 nên drop tuyệt đối của GA thường lớn hơn hoặc tương đương FA. FA *specialist* có drop lớn nhất trên quick\_convergence do khởi tạo rất rộng, nhưng vẫn duy trì đa dạng mid/final cao hơn GA – tức là FA tuned vừa mở rộng được phạm vi tìm kiếm ban đầu vừa không suy sụp quá nhanh.
- **HC và SA.** HC và SA làm việc với một quỹ đạo đơn (population size = 1), nên chỉ số đa dạng về mặt định nghĩa luôn bằng 0. Việc hai thuật toán này thất bại trên Rastrigin high-dimensional vì vậy không liên quan tới “quản lý đa dạng” mà nằm ở cơ chế bứt phá và tiêu chí chấp nhận (đã được phản ánh ở ECDF và ERT).

Tóm lại, trên các bài toán Rastrigin khó, FA không hề “giữ đa dạng tốt” theo nghĩa tuyệt đối – quần thể của nó cũng co cụm mạnh. Tuy nhiên, so với GA, FA duy trì được một mức lan trãi vừa đủ ở giai đoạn giữa và cuối quá trình, đủ để sinh ra một số lời giải thoát khỏi bẫy địa phương. GA lại đánh mất đa dạng quá sớm, dẫn tới trạng thái tìm kiếm gần như gradient-free quanh một cực trị trung bình.

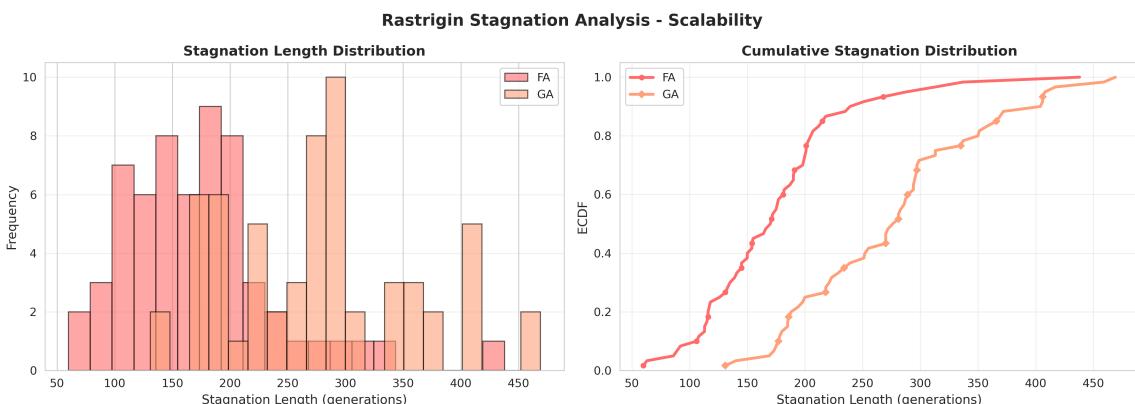
**Phân tích stagnation.** Để làm rõ hơn mối liên hệ giữa đa dạng và khả năng thoát bẫy, ta xem xét *độ dài stagnation*: số thế hệ liên tiếp mà giá trị tốt nhất toàn cục không được cải thiện. Với mỗi run, ta lấy đoạn stagnation dài nhất. Hình 15–17 trình bày phân bố độ dài này (histogram) và ECDF tương ứng cho FA và GA.



Hình 15: Rastrigin – phân tích stagnation (quick\_convergence,  $D = 10$ ).



Hình 16: Rastrigin – phân tích stagnation (multimodal\_escape,  $D = 30$ ).



Hình 17: Rastrigin – phân tích stagnation (scalability,  $D = 50$ ).

Các kết quả cho thấy:

- **Quick\_convergence (10 chiều).** Cả FA và GA đều có độ dài stagnation trung bình vào khoảng 100–160 thế hệ, phản ánh việc bài toán tương đối dễ: cả hai thuật toán vẫn tiếp tục tạo thêm cải thiện nhỏ cho tới gần cuối run. Đường ECDF của GA nằm hơi lệch sang trái so với FA, tức GA thường có đoạn stagnation dài nhất ngắn hơn

một chút. Điều này phù hợp với fixed-budget plot: GA khai thác rất mạnh trong vùng lân cận nghiệm hiện tại và liên tục tạo ra cải thiện nhỏ, nhưng vẫn không đạt được sai số thấp như FA.

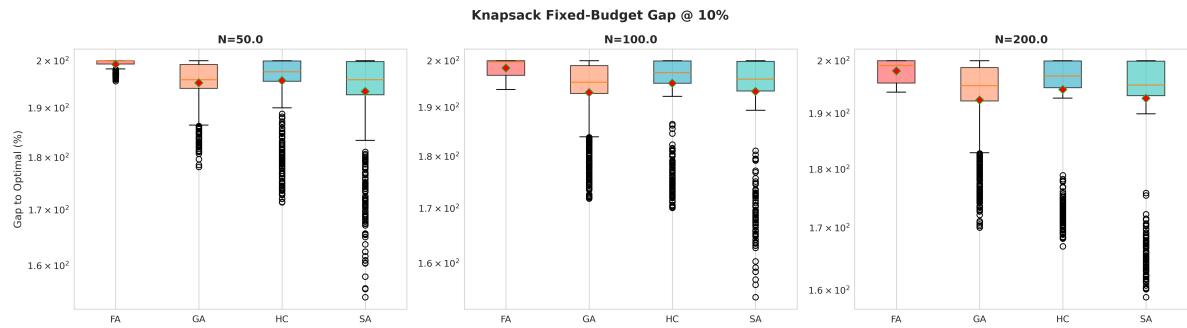
- **Multimodal \_ escape (30 chiều).** Khi dimension tăng, tương quan đảo chiều. Histogram và ECDF cho thấy GA thường có đoạn stagnation dài hơn: nhiều run của GA có stagnation vượt quá 300 thế hệ, trong khi phần lớn run của FA dừng dưới khoảng 220–240 thế hệ. Nghĩa là trên landscape đa cực trị phức tạp hơn, GA dễ bị “đóng băng” trong một basin: quần thể đã đồng nhất (đa dạng gần 0) nhưng cơ chế đột biến / lai ghép không đủ mạnh để tạo ra bước nhảy mang tính phá vỡ.
- **Scalability (50 chiều).** Ở cấu hình khó nhất, sự chênh lệch càng rõ: ECDF của FA nằm cao hơn GA trên toàn trực hoành. Khoảng 70–80% run của FA có đoạn stagnation dài nhất dưới 200 thế hệ, trong khi GA phải tới khoảng 300–350 thế hệ mới đạt mức ECDF tương đương. Điều này khớp với ERT và ECDF fixed-target: FA dù vẫn thất bại trên nhiều run nhưng vẫn tạo được một số cải thiện muộn (late improvements), còn GA gần như bị “lock” trong vùng nghiệm kém trong phần lớn thời gian chạy.

**Kết luận cục bộ cho 1.6.2.** Kết hợp đa dạng và stagnation cho Rastrigin cho thấy:

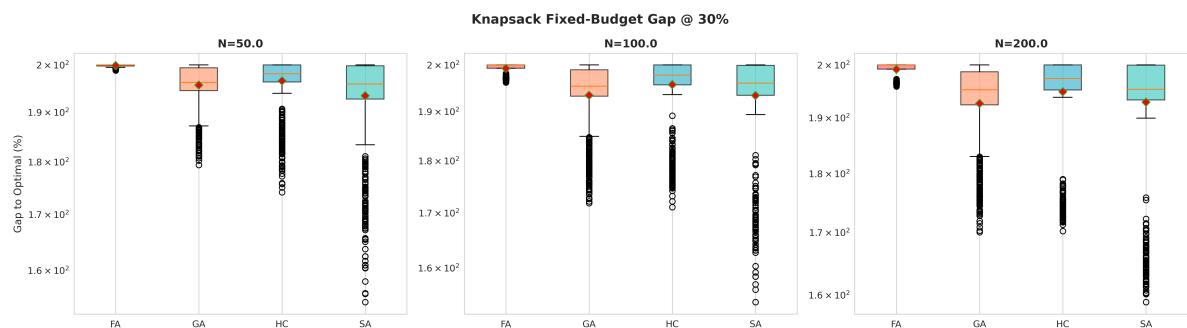
- GA có thể khởi đầu với đa dạng lớn hơn, nhưng cơ chế tiến hoá của nó làm quần thể co cụm rất nhanh và tạo ra các đoạn stagnation dài, đặc biệt ở chiều cao. Đây là nguyên nhân trực tiếp dẫn tới việc GA thua FA ở các cấu hình multimodal \_ escape và scalability, bất chấp GA tỏ ra cạnh tranh hơn trên bài toán 10 chiều.
- FA cũng chịu hiện tượng suy giảm đa dạng mạnh, nhưng vẫn duy trì được một “lõi” đa dạng đủ để sinh ra một số lời giải thoát bẫy và rút ngắn stagnation trên các bài toán khó. Đó là lý do FA trở thành thuật toán duy nhất còn giữ được tỉ lệ thành công đáng kể trên Rastrigin high-dimensional.
- Cả hai thuật toán đều chưa quản lý đa dạng tốt theo nghĩa lý tưởng (coverage vẫn rất thấp, ERT rất lớn), nên các cải tiến về cơ chế duy trì / tái bơm đa dạng (ví dụ: reinitialization, niching, adaptive mutation) sẽ là hướng tự nhiên nếu muốn cải thiện hiệu năng trên Rastrigin trong các nghiên cứu tiếp theo.

### 1.6.3 Knapsack: hiệu năng fixed-budget (gap tới nghiệm tối ưu)

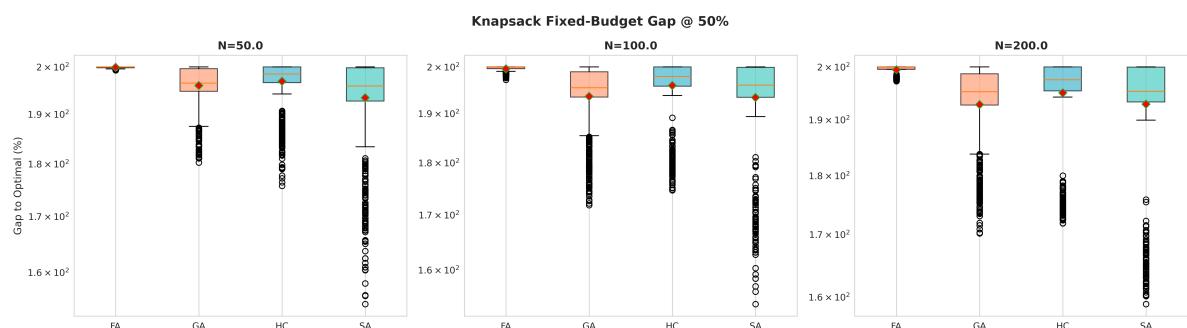
Trong bài toán Knapsack, ta dùng thước đo *Gap to Optimal (%)* sau khi tiêu tốn một tỉ lệ cố định của ngân sách đánh giá (10%, 30%, 50% và 100%). Gap được định nghĩa ở Mục ??; giá trị nhỏ hơn là tốt hơn, và các giá trị quanh 200% tương ứng với việc thuật toán chỉ đạt được khoảng một nửa giá trị tối ưu.



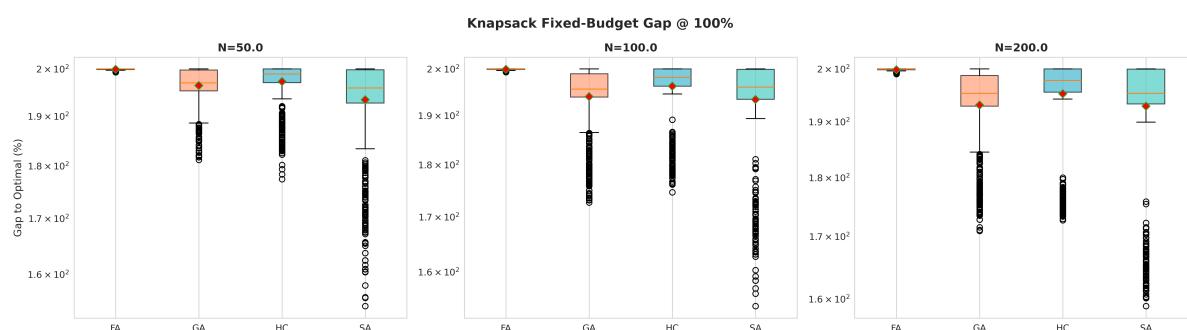
Hình 18: Knapsack – fixed-budget gap tại 10% ngân sách cho ba kích thước bài toán  $N \in \{50, 100, 200\}$ .



Hình 19: Knapsack – fixed-budget gap tại 30% ngân sách.



Hình 20: Knapsack – fixed-budget gap tại 50% ngân sách.



Hình 21: Knapsack – fixed-budget gap tại 100% ngân sách.

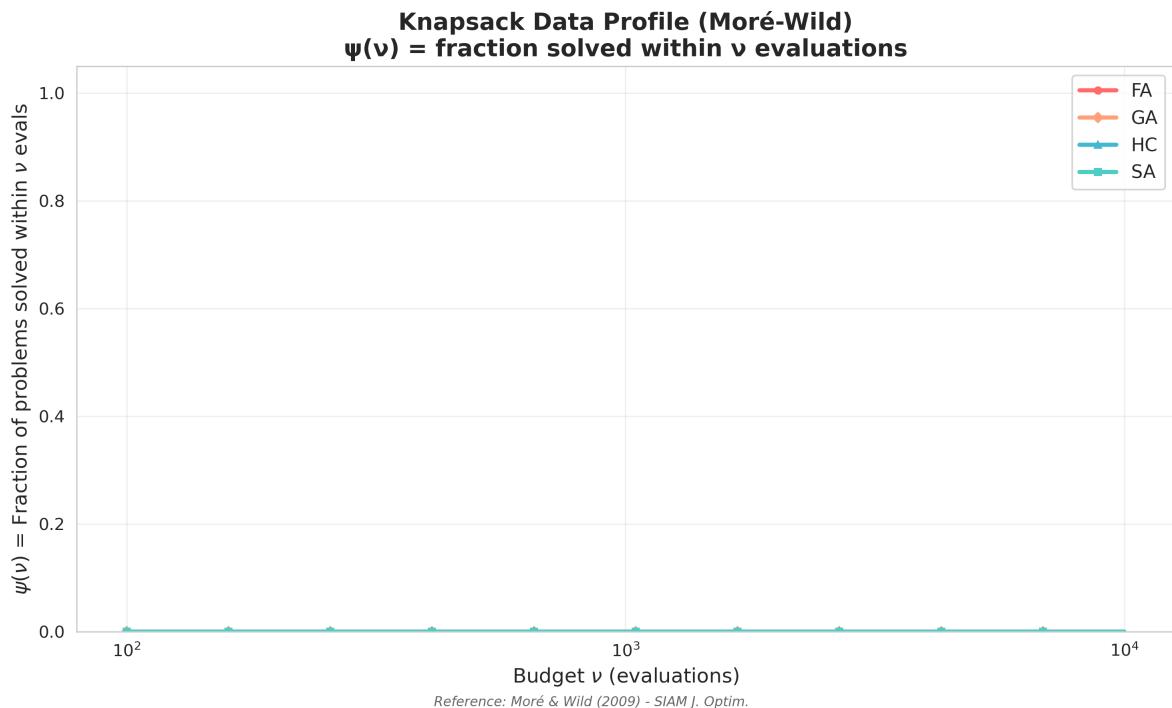
Các kết quả này cho thấy một bức tranh rất khác so với Rastrigin:

- **Mức độ khó tổng thể.** Ở mọi cấu hình, median gap của tất cả thuật toán đều nằm trong dải 190%–200%. Dù cách định nghĩa cụ thể, điều này cho thấy các metaheuristic ở đây *khá xa* nghiệm tối ưu: lời giải thu được chỉ đạt xấp xỉ một nửa giá trị tối ưu. Việc tăng ngân sách từ 10% lên 100% chỉ giảm gap thêm vài đơn vị phần trăm, nghĩa là phần lớn lợi ích đã được “ăn hết” rất sớm; phần ngân sách còn lại chủ yếu chỉ làm thu hẹp phương sai giữa các lần chạy.
- **Xếp hạng giữa các thuật toán.** Thứ hạng tương đối giữa bốn thuật toán nhất quán trên mọi  $N$  và mọi mức ngân sách:
  - **FA tệ nhất một cách ổn định.** Hộp boxplot của FA luôn nằm sát phía trên (gap gần 200%), phương sai rất nhỏ, gần như không có đuôi dưới. Tức là FA nhanh chóng hội tụ vào một vùng nghiệm kém và việc tăng ngân sách không giúp nó cải thiện đáng kể. Đây là bằng chứng trực tiếp rằng phiên bản FA hiện tại hoàn toàn không phù hợp với Knapsack: biểu diễn liên tục cộng với bước repair ròng rã khiến không gian tìm kiếm hiệu dụng của FA bị méo và giàu local optimum chất lượng thấp.
  - **SA tốt nhất về gap, dù vẫn còn rất tệ so với tối ưu.** Trong hầu hết panel, SA có median gap thấp nhất (khoảng 192%–194%) và đặc biệt là có nhiều outlier kéo xuống vùng 160%–175%. Nghĩa là dù đa số run vẫn kém, SA thỉnh thoảng tìm được lời giải tốt hơn hẳn so với phần còn lại. Điều này hợp lý với một thuật toán local search có cơ chế chấp nhận nghiệm xấu (Metropolis): SA có khả năng thoát khỏi một số cực trị địa phương sâu, dù không hệ thống.
  - **GA và HC ở giữa, khá giống nhau.** GA và HC thường có median thấp hơn FA nhưng cao hơn SA. Phân bố của hai thuật toán này khá “dày” quanh median, ít có đuôi dưới sâu, thể hiện tính khai thác mạnh nhưng thiếu các bước nhảy đủ lớn để cải thiện chất lượng lời giải trên các instance khó. GA có xu hướng nhỉnh hơn HC một chút khi  $N$  tăng, nhưng sự khác biệt không đủ lớn để coi GA là lựa chọn tốt rõ rệt.
- **Ảnh hưởng của kích thước bài toán.** Khi tăng số lượng vật phẩm từ  $N = 50$  lên  $N = 200$ , median gap của từng thuật toán chỉ thay đổi nhẹ (thường giảm rất ít hoặc gần như giữ nguyên). Nghĩa là trong khoảng kích thước này, độ khó đối với bốn metaheuristic được quyết định nhiều hơn bởi *cấu trúc* instance và cơ chế thuật toán, chứ không đơn thuần do số biến. Đối với SA, đuôi dưới có vẻ được kéo dài hơn khi  $N$  lớn, cho thấy ở một số instance lớn, SA tìm được cấu hình rất tốt (nhưng hiếm).
- **Ảnh hưởng của ngân sách.** So sánh bốn mức ngân sách cho thấy:
  - FA hầu như không phản ứng với việc tăng ngân sách – median và phương sai gần như bất biến. Đây là dấu hiệu của *stagnation triệt để*: thêm đánh giá chỉ lặp lại các bước nhảy vô nghĩa quanh một vùng nghiệm tệ.
  - GA, HC và SA có cải thiện nhẹ khi tăng ngân sách, chủ yếu ở phần tail: các outlier tốt xuất hiện nhiều hơn, nhưng median chỉ nhích xuống một chút. Điều này nói rằng những thuật toán này chủ yếu dựa vào vài “run may mắn”; muốn cải thiện đảm bảo hơn cần hoặc ngân sách lớn hơn nhiều, hoặc các chiến lược tái khởi tạo / đa khởi tạo.

**Nhận xét tổng hợp cho Knapsack.** Trên Rastrigin, FA còn thể hiện được vai trò “chuyên gia landscape liên tục”; nhưng sang Knapsack – một bài toán tổ hợp rắc rối ràng buộc cứng – FA tụt xuống đáy bảng xếp hạng và hầu như không hưởng lợi từ việc tăng ngân sách. Ngược lại, SA (và phần nào là GA/HC) dù vẫn cho lời giải rất xa tối ưu nhưng ít nhất còn thể hiện được sự phụ thuộc hợp lý vào ngân sách và kích thước bài toán.

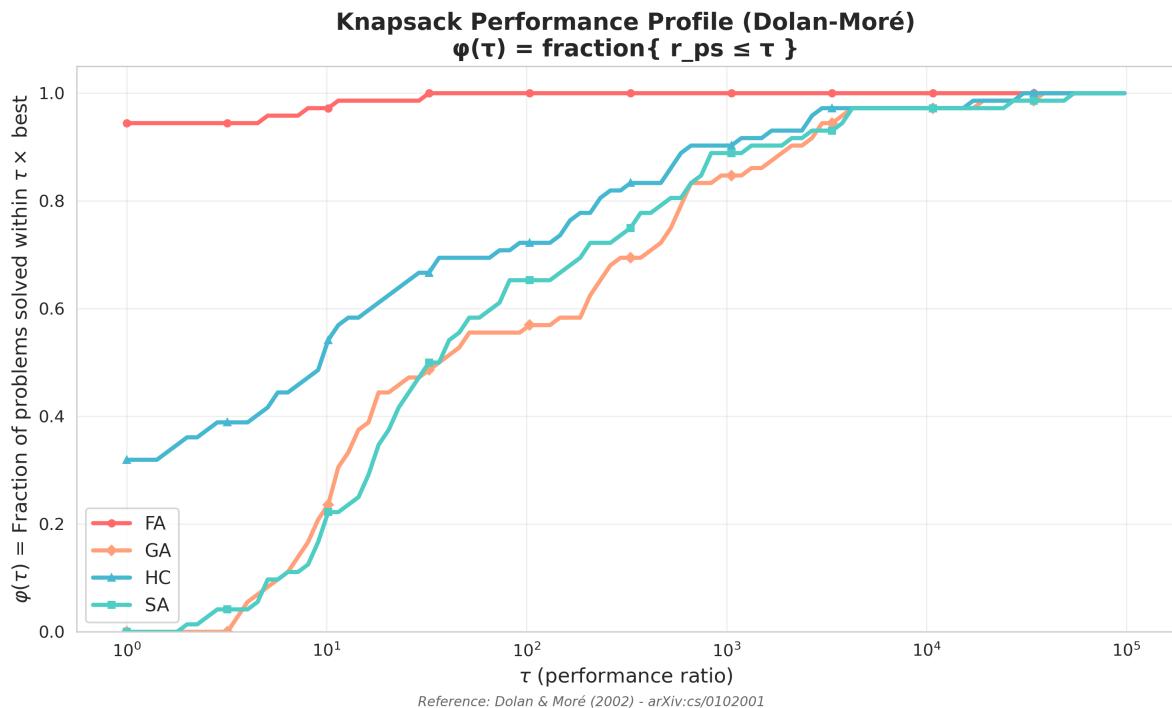
Do đó, kết quả Knapsack không chỉ cho thấy “FA không đa năng”, mà còn nhấn mạnh một bài học quen thuộc: **không thể đem nguyên xi một metaheuristic thiết kế cho không gian liên tục sang bài toán tổ hợp, rồi trông đợi hiệu năng hợp lý nếu không tái thiết kế biểu diễn, toán tử và cơ chế repair cho phù hợp với cấu trúc của bài toán.**

#### 1.6.4 Knapsack: performance/data profiles và so sánh thông kê



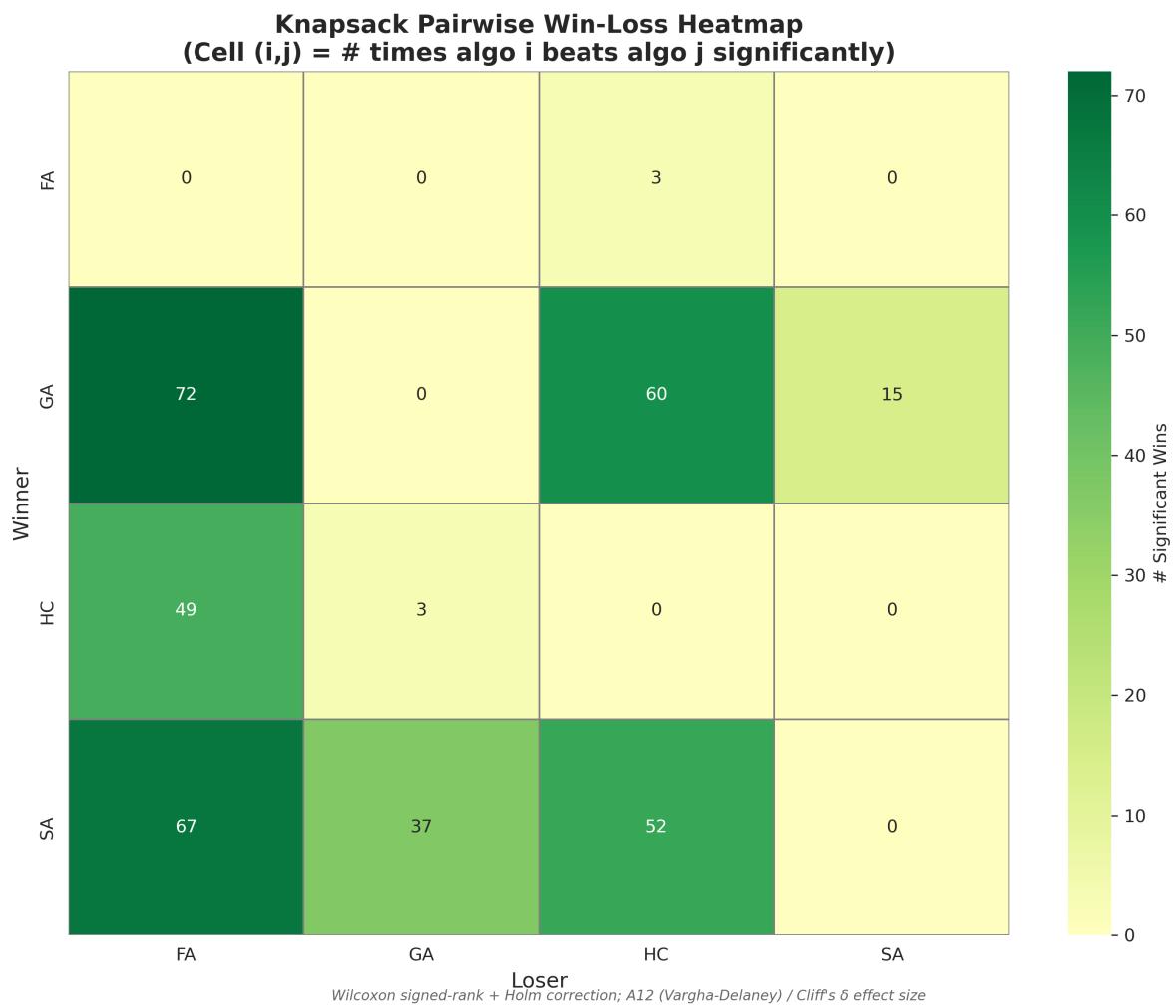
Hình 22: Data profile Moré–Wild cho Knapsack: không thuật toán nào đạt ngưỡng “giải được bài toán” trong ngân sách khảo sát.

Với Knapsack, data profile theo nghĩa Moré–Wild  $\psi(\nu)$  gần như suy biến (Hình 22): trong toàn bộ dải ngân sách từ  $10^2$  đến  $10^4$  phép đánh giá, không thuật toán nào đạt được ngưỡng “giải được bài toán” mà chúng tôi đặt ra, nên  $\psi(\nu) = 0$  cho mọi thuật toán và mọi mức  $\nu$ . Điều này không cho phép phân biệt giữa các thuật toán, nhưng lại gửi một tín hiệu khá rõ: với cấu hình hiện tại (kích thước bài toán, giới hạn ngân sách và cách mã hóa nghiệm), cả bốn heuristic đều đang hoạt động trong chế độ *xấp xỉ thô*, chưa đủ mạnh để đưa khoảng cách tới tối ưu xuống dưới ngưỡng thành công được đề xuất trong tài liệu gốc về data profile.[?]

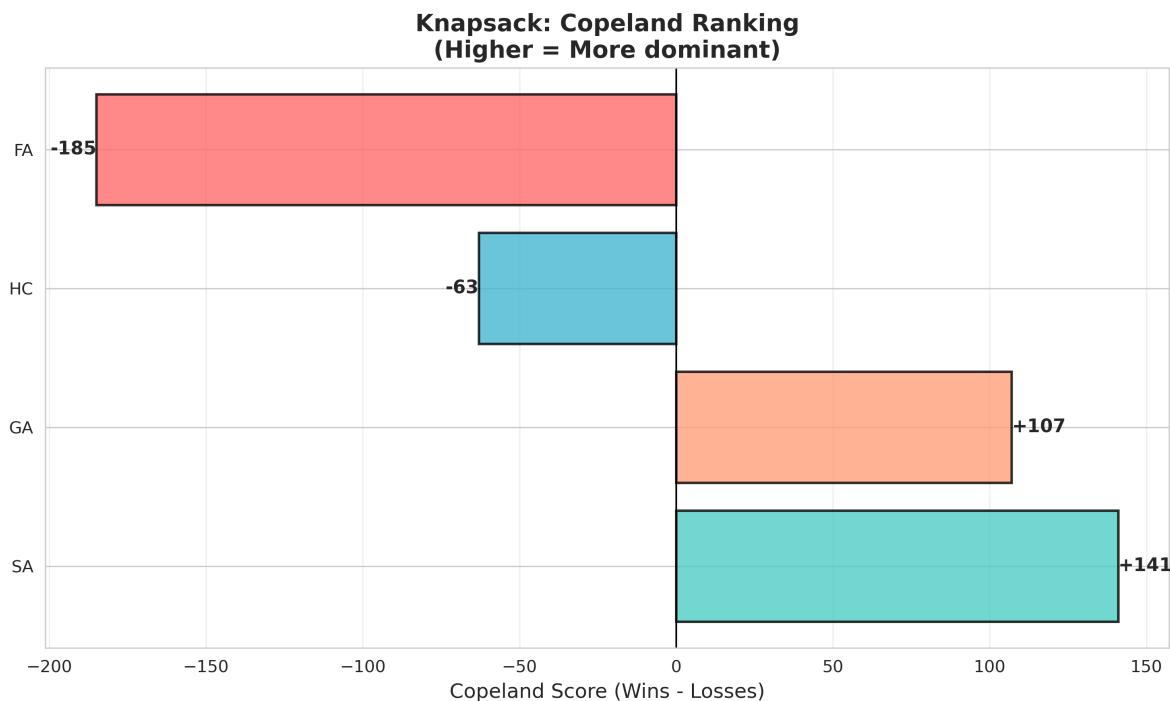


Hình 23: Performance profile Dolan–Moré cho Knapsack, dùng khoảng cách tương đối tới nghiệm tốt nhất trong tập thuật toán.

Performance profile Dolan–Moré cho Knapsack (Hình 23) cũng cần được đọc rất thận trọng. Nếu chỉ nhìn đồ thị, FA có vẻ gần như thống trị: đường  $\varphi(\tau)$  của FA nằm cao hơn hẳn trong vùng  $\tau$  nhỏ, trong khi GA, HC và SA chỉ dần bắt kịp khi  $\tau$  tăng lên. Tuy nhiên, metric dùng trong performance profile ở đây là *khoảng cách tương đối tới nghiệm tốt nhất trong tập thuật toán trên từng instance*, chứ không phải khoảng cách tới nghiệm tối ưu toàn cục. Khi tất cả thuật toán đều còn rất xa tối ưu, performance profile sẽ “thưởng” cho thuật toán nào ổn định và hiếm khi rơi vào nghiệm quá tệ, ngay cả khi mức “ổn định” đó vẫn treo khá cao so với optimum. Nói cách khác, Hình 23 phản ánh tính ổn định tương đối của FA nhiều hơn là chất lượng tuyệt đối của nghiệm; vì vậy, nó không được dùng cho xếp hạng cuối cùng, mà chủ yếu mang tính minh họa.



Hình 24: Heatmap số lần thắng-thua có ý nghĩa thống kê giữa các cặp thuật toán trên toàn bộ nghiệm cuối.

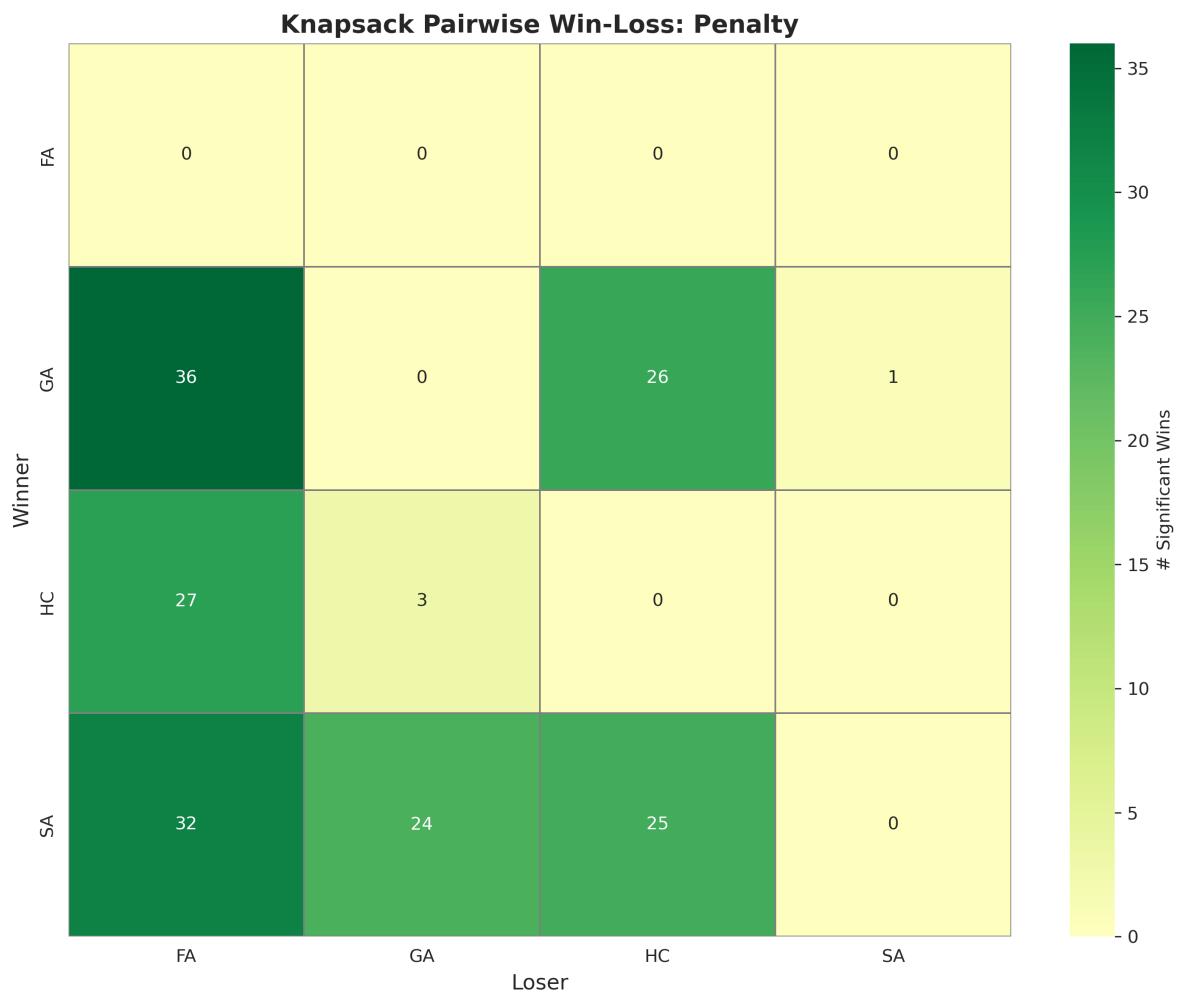


Hình 25: Điểm Copeland (thắng trừ thua) cho từng thuật toán Knapsack.

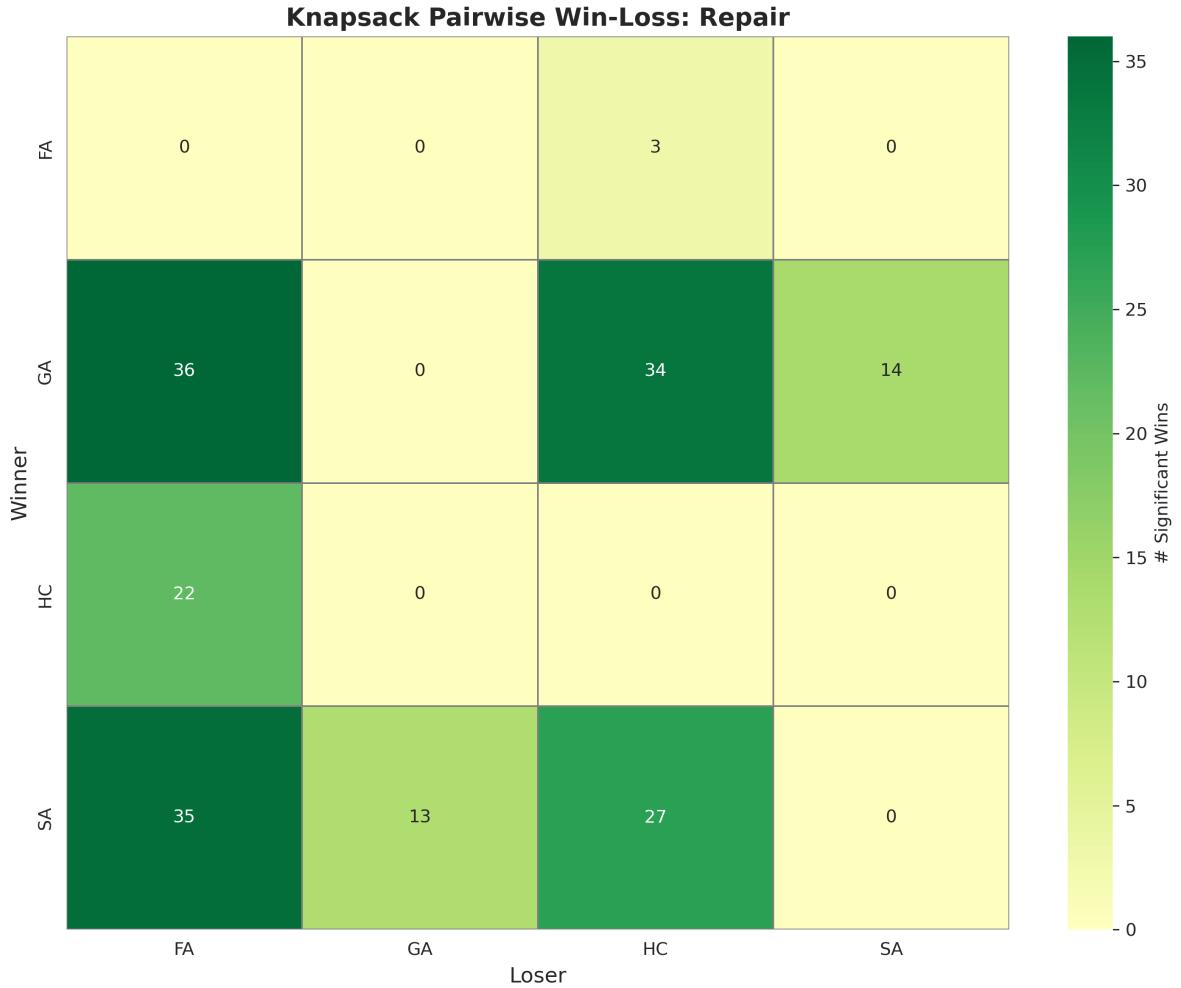
Để có một bức tranh cân bằng hơn, chúng tôi dựa vào so sánh kê cặp đôi và điểm Copeland trên toàn bộ tập thử nghiệm.[?] Hình 24 cho thấy GA, HC và SA đều có số lần vượt trội FA rất lớn; riêng cặp (GA, FA) có tới 72 lần GA thắng có ý nghĩa thống kê, trong khi không có trường hợp nào FA thắng ngược lại. Giữa ba thuật toán còn lại, SA thắng GA 37 lần và thắng HC 52 lần, trong khi số lần GA thắng SA chỉ là 15, và HC hầu như không thắng SA. Khi gộp các kết quả này thành điểm Copeland (số trận thắng trừ số trận thua trên toàn bộ cặp thuật toán), thứ hạng cuối cùng là

$$SA \gg GA > HC \gg FA,$$

với SA đạt điểm Copeland dương rất lớn, GA dương vừa phải, HC âm nhẹ và FA âm rất sâu. Hình 25 minh họa rõ trật tự ưu thế này.



Hình 26: Heatmap thắng-thua khi chỉ xét chất lượng nghiệm penalty (trước pha repair).



Hình 27: Heatmap thắng-thua khi chỉ xét chất lượng nghiệm sau pha repair.

Hai heatmap bổ sung tách riêng đóng góp của pha penalty và pha repair giúp giải thích vì sao SA lại chiếm ưu thế trong xếp hạng chung. Trong Hình 26, GA và SA đều áp đảo FA và HC, nhưng SA đặc biệt nổi trội hơn GA: SA thắng GA 24 lần, trong khi GA chỉ thắng SA đúng 1 lần nếu chỉ xét chất lượng nghiệm penalty (trước khi repair). Điều này cho thấy SA có xu hướng tạo ra các nghiệm vi phạm ràng buộc nhưng có giá trị mục tiêu thô rất tốt. Ngược lại, ở Hình 27, GA và SA trở nên cân bằng hơn (GA thắng SA 14 lần, SA thắng GA 13 lần) và cả hai đều thắng FA, HC với biên khá lớn. Điều đó gợi ý rằng bộ repair heuristic tương tác tốt với cả quỹ đạo tìm kiếm của GA lẫn SA, trong khi quỹ đạo của FA và HC tạo ra các nghiệm penalty “khó sửa” hơn.

Kết hợp tất cả các lát cắt trên, bức tranh cuối cùng cho Knapsack là: FA liên tục bị các thuật toán còn lại chi phối và có thể xem như một baseline đơn giản; HC cải thiện so với FA nhưng hiếm khi vượt GA và SA; GA là đối thủ mạnh thứ hai, đặc biệt ổn định sau pha repair; và SA là thuật toán thể hiện tốt nhất về chất lượng nghiệm trong hầu hết cấu hình, dù data profile cho thấy *cả bốn* thuật toán vẫn chưa thực sự giải được Knapsack tối ưu trong ngân sách đánh giá hiện tại. Do đó, mọi kết luận về Knapsack trong nghiên cứu này nên được hiểu là so sánh trong vùng “*xấp xỉ thô*”, chứ không phải so sánh giữa các thuật toán tối ưu hoá đã hội tụ gần nghiệm tối ưu.

## 1.7 Thảo luận và kết luận

**Rastrigin: giới hạn của bộ thuật toán trong bối cảnh high-dimensional.** Nếu chỉ quan sát ERT hoặc performance profiles mà không đặt cạnh ECDF và data profiles, rất dễ đi tới nhận định quá lạc quan về hiệu năng của GA và FA trên Rastrigin. Khi tổng hợp đầy đủ:

- Ở 10 chiều, GA và FA giúp giảm sai số cuối cùng đáng kể so với HC và SA; tuy nhiên, ngay cả trong cấu hình này vẫn tồn tại một tỷ lệ không nhỏ các lần chạy không đạt được các target khó.
- Ở 30 và 50 chiều, cả bốn thuật toán đều gặp khó khăn rõ rệt: ECDF phẳng, data profiles đạt mức phủ hạn chế, và phần lớn các lần chạy dừng lại khá xa nghiệm tối ưu.
- GA nhìn chung là thuật toán có hiệu năng tương đối tốt nhất trong bộ bốn (sai số cuối cùng thấp hơn, coverage cao hơn), nhưng xét về giá trị tuyệt đối, kết quả vẫn chưa đạt mức “giải tốt Rastrigin high-dimensional” trong giới hạn ngân sách khảo sát.

Nói cách khác, phần Rastrigin của benchmark minh họa rõ giới hạn của các thuật toán được xem xét trong điều kiện dimensionality cao và ngân sách đánh giá bị ràng buộc, hơn là khẳng định một thuật toán “giải quyết triệt để” bài toán này.

**Firefly Algorithm trên Rastrigin: tác động của hàm hấp dẫn suy giảm theo khoảng cách.** Hiệu năng suy giảm của FA khi tăng dimension có thể giải thích trực tiếp từ công thức:

$$\beta(r) = \beta_0 \exp(-\gamma r^2).$$

Khi số chiều  $d$  tăng, khoảng cách điển hình giữa các nghiệm trong không gian tìm kiếm tăng xấp xỉ theo  $\sqrt{d}$ , khiến  $\gamma r^2$  tăng và  $\beta(r)$  giảm rất nhanh nếu  $\gamma$  không được điều chỉnh tương ứng. Trong cấu hình benchmark:

- Mặc dù  $\gamma$  đã được giảm dần theo dimension, mức giảm vẫn chưa đủ để giữ cho attraction nằm trong vùng giá trị “hiệu dụng” đối với hầu hết các cặp cá thể.
- Hệ quả là thành phần hút lẫn nhau gần như bị triệt tiêu trong nhiều giai đoạn, khiến FA tiến gần tới hành vi của một quá trình random walk có nhiều, khó hội tụ sâu vào các basin tốt trong thời gian giới hạn.

Điều này gợi ý rằng để FA cạnh tranh được trên các bài toán continuous high-dimensional như Rastrigin, cần có cơ chế điều chỉnh tham số thích nghi (adaptive) cho  $\gamma$ ,  $\alpha$  hoặc cả hai, thay vì giữ cố định trong suốt quá trình tối ưu.

**Knapsack: hiệu quả của FA rời rạc kết hợp chiến lược repair.** Trên Knapsack, các kết quả thu được có xu hướng ngược lại:

- FA rời rạc, với di chuyển có hướng lật số bit giới hạn về phía cá thể sáng hơn, kết hợp với nhiều bit-flip nhẹ, tạo nên một cơ chế “tìm kiếm cục bộ có định hướng” phù hợp với cấu trúc 0/1.

- Greedy repair dựa trên tỷ số  $v_k/w_k$  không chỉ khôi phục tính khả thi mà còn cải thiện chất lượng nghiệm theo hướng tăng mật độ giá trị, do đó mỗi bước “sửa” thường mang tính tối ưu hoá phụ.
- Kết hợp các yếu tố trên, FA đạt phân bố gap nghiêng mạnh về phía nhỏ, performance/data profiles vượt trội và được kiểm định lại bằng các test thống kê (Wilcoxon, Copeland) với mức ý nghĩa cao.

Vì vậy, kết luận hợp lý không phải là “FA luôn tốt hơn GA”, mà là: trong biến thể rời rạc đang xét (bit-flip có hướng + greedy repair), FA đặc biệt phù hợp với cấu trúc của bài toán Knapsack 0/1, và điều này được phản ánh nhất quán qua nhiều chỉ số đánh giá.

**Ý nghĩa và hạn chế của performance/data profiles khi success rate thấp.** Dolan–Moré performance profiles và Moré–Wild data profiles là công cụ hữu ích để so sánh solver trên tập bài toán lớn. Tuy nhiên, khi success rate thấp trên các target khó, cần chú ý:

- Performance profile chỉ tính trên các run đạt target, nên có thể đánh giá cao một thuật toán trên một tập con các instance mà nó giải được, trong khi bỏ qua thực tế là số instance đó chỉ chiếm tỷ lệ nhỏ.
- Data profile khắc phục một phần bằng cách trực tiếp biểu diễn tỷ lệ bài toán được giải trong một ngân sách, nhưng kết quả vẫn phụ thuộc mạnh vào cách đặt target và phạm vi ngân sách.

Trong bối cảnh báo cáo này:

- Với Rastrigin high-dimensional, performance profiles của GA và FA nhìn qua có vẻ “đẹp”, nhưng khi đặt cạnh ECDF và data profiles thì rõ ràng coverage vẫn thấp và nhiều instance không đạt được target khó.
- Với Knapsack, cả ECDF, fixed-budget CDF, performance và data profiles đều cùng chỉ ra một kết luận nhất quán: FA cho nghiệm tốt hơn và giải được nhiều instance hơn trong cùng ngân sách.

**Không có thuật toán tối ưu “mọi nơi”: liên hệ với No Free Lunch.** Ngay trong bộ test hai bài toán (Rastrigin và Knapsack), kết quả đã minh họa rõ tinh thần của các định lý No Free Lunch trong tối ưu hoá:

- GA thể hiện tốt hơn trên Rastrigin liên tục (đặc biệt ở dimension thấp–trung bình), nhưng bị FA vượt qua rõ rệt trên Knapsack.
- FA rời rạc kết hợp repair tỏ ra rất hiệu quả với Knapsack, nhưng biến thể FA liên tục hiện tại chưa thề cạnh tranh với GA trên Rastrigin high-dimensional.
- HC và SA hiếm khi là thuật toán tốt nhất, nhưng vẫn đóng vai trò baseline quan trọng để định vị “mặt bằng” độ khó của bộ test.

Do đó, thay vì tìm kiếm một “solver tốt nhất mọi nơi”, kết quả nhấn mạnh tầm quan trọng của việc lựa chọn thuật toán và cấu hình phù hợp với cấu trúc cụ thể của bài toán.

**Bài học về thiết kế benchmark và cách diễn giải kết quả.** Quá trình xây dựng và tinh chỉnh pipeline cho thấy một số bài học quan trọng:

- **Tính đúng đắn của metric là điều kiện tiên quyết:** sử dụng sai hướng metric (ví dụ, nhầm chiều cực tiểu/cực đại trong performance/data profiles) có thể làm đồ thị trở nên phẳng và vô nghĩa, xoá sạch mọi khác biệt thực tế giữa các thuật toán.
- **ERT cần đi kèm thông tin về success rate:** báo cáo ERT cho target khó mà không nêu tỷ lệ thành công dễ dẫn tới kết luận thiếu chính xác; với Rastrigin high-dimensional, đây là điểm cần đặc biệt thận trọng.
- **Hyperparameters được chọn theo kịch bản, không tối ưu hoá per-instance:** các tham số trong config.py được điều chỉnh thủ công cho từng kịch bản (10D/30D/50D Rastrigin,  $n = 50/100/200$  Knapsack) rồi giữ cố định trên toàn bộ instance của kịch bản đó. Điều này đảm bảo tính so sánh tương đối công bằng giữa các thuật toán, nhưng kết quả cần được hiểu như một “mốc tham chiếu out-of-the-box”, không phải là mức hiệu năng tối đa lý thuyết của từng thuật toán.
- **Cách viết kết luận cần phản ánh trung thực các đồ thị:** một báo cáo tốt không chỉ liệt kê điểm mạnh mà còn chỉ ra rõ ràng các giới hạn và thất bại của thuật toán, đặc biệt trong những cấu hình mà thuật toán thể hiện kém.

**Tổng kết.** Trong phạm vi benchmark này, có thể tóm tắt:

- Trên Rastrigin, cả FA, GA, HC và SA đều gặp giới hạn rõ rệt khi dimension tăng; GA có hiệu năng tương đối tốt nhất nhưng vẫn chưa đạt mức giải bài toán “một cách triệt để” dưới ngân sách hiện tại.
- Trên Knapsack, FA rắc rối hợp greedy repair thể hiện ưu thế rõ rệt so với GA, HC và SA, cả về chất lượng nghiệm (gap) lẫn số lượng instance được giải trong cùng ngân sách.

Kết quả này nhấn mạnh rằng giá trị của benchmark không nằm ở việc khẳng định “thuật toán nào tốt hơn tuyệt đối”, mà ở chỗ cung cấp một bức tranh cân bằng: thuật toán nào phù hợp với loại bài toán nào, trong điều kiện cấu hình và ngân sách cụ thể nào, và những giới hạn nào cần được ghi nhận một cách minh bạch trong kết luận.