GNBG: Trình tạo hàm chuẩn tổng quát cho bài toán Tối ưu số

Nhóm 2 - Tiến hóa đa nhiệm [1]

[1]Trường Công nghệ thông tin và Truyền thông
Đại học Bách khoa Hà Nội

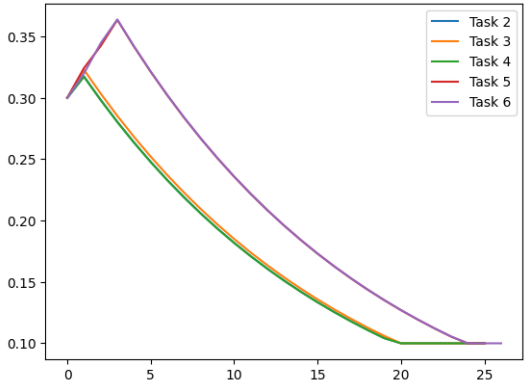
Mục lục

Một số thuật toán dựa trên rmp

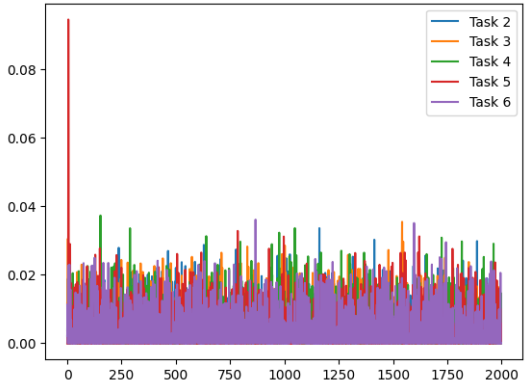
- **MFEA:** Xác suất chuyển giao giữa hai task bất kỳ luôn là cố định.
- **MFEA2:** Cập nhật *rmp* theo phân kỳ Kullback-Leiber giữa hai phân phối xác suất.
- **EME-BI**: Cập nhật *rmp* theo trung bình Lehmer (giống như thuật toán SHADE).

Phân tích EME-BI

Xác suất chuyển giao từ Task k tới Task 1 (trong lớp bài toán Unimodal) được thể hiện trong đồ thị:



Xác suất chuyển giao từ Task k tới Task 1 (trong lớp bài toán Unimodal) được thể hiện trong đồ thị:

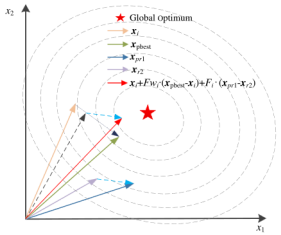


Nhận xét

- Xác suất chuyển giao trong MFEA2 quá nhỏ, hầu như không có sự chuyển giao.
- Xác suất chuyển giao trong EME-BI hầu như giảm, số lần chuyển giao thành công cũng không nhiều, nhưng vẫn tốt hơn MFEA2.
- **Phân bố tài nguyên tính toán ít hơn cho phần chuyển giao, nhiều hơn cho Local Search ?**

- jSO kết hợp nhiều ý tưởng của DE-variants: SHADE, L-SHADE, iL-SHADE, ...
- Qua thực nghiệm của tác giả, jSO vượt qua các biến thể khác trên các lớp hàm Unimodal, Multimodal, ... với số chiều 30.

- $v_i = x_i + FW_i(x_{pbest} - x_i) + F_i(x_{pr1} - x_2)$



Trong đó x_{pr1} được chọn từ quần thể với phân phối xác suất Pr:

$$\mathrm{Pr}_i = \frac{k(NP - i) + 1}{\sum_{j=1}^{NP} [k(NP - j) + 1]}$$

x_2 được chọn ngẫu nhiên từ $Pop + Archive$.

- Thuật toán lưu giữ ba archive: F, CR và archive cá thể lưu giữ những cá thể cũ đã bị thay thế bởi offspring.
-

$$M_F = \begin{cases} M_F, & \text{if } |S_F| = 0 \\ \frac{(\text{mean}_{WL}(S_F) + M_F)}{2}, & \text{if } |S_F| > 0 \end{cases}$$

$$M_{CR} = \begin{cases} \text{end}, & \text{if } M_{CR} = \text{end or } \max(S_{CR} = 0) \\ \frac{(\text{mean}_{WL}(S_{CR}) + M_{CR})}{2}, & \text{otherwise} \end{cases}$$

- Archive cá thể A được tổ chức theo cơ chế FIFO.

Limited Strategy

- Đảm bảo $Cr \geq 0.5$ khi số FEs chưa quá 25%, $Cr \geq 0.25$ khi số FEs từ $25\% - 50\%$.
- Đảm bảo $F \leq 0.7$ khi số FEs chưa quá 25%, $F \leq 0.8$ khi số FEs chưa quá 50%, $F \leq 0.9$ khi số FEs chưa quá 75%.
- $$Fw = \begin{cases} 0.7 \cdot F, & FEs < 0.2 \cdot FEs_{\max} \\ 0.8 \cdot F, & 0.2 \cdot FEs_{\max} \leq FEs < 0.4 \cdot FEs_{\max} \\ 1.2 \cdot F, & FEs \geq 0.4 \cdot FEs_{\max} \end{cases}$$
- p trong $p - best$ giảm tuyến tính.

Selection Strategy

- Ở các phiên bản DE và biến thể của DE trước đây, F và CR được sinh từ hai giá trị M_F, M_{CR} từ archive. Hai giá trị được chọn từ archive hoàn toàn ngẫu nhiên.
- Trong jSO, sự lựa chọn ấy không ngẫu nhiên mà sẽ thiên vị cho những giá trị sinh được F và CR tốt, làm cải thiện fitness cá thể.
-

$$SR_h = \frac{SN_h}{N_h}, h = 1, ..., H$$

$$\mathrm{Pr}_h = \frac{SR_h}{\sum_{i=1}^{H} SR_i}$$

- Ban đầu, xác suất của mỗi phần tử trong archive được chọn đều là $\frac{1}{H}$. Sau đó, bất cứ thế hệ nào mà fitness không cải thiện ta đều sẽ reset lại xác suất như ban đầu.

EME-BI Two-Phase

- **Phase 1 (Trao đổi):** Trao đổi giữa các Task. Cập nhật giá trị rmp.
- **Phase 2 (Khám phá):** Chia quần thể thành K quần thể con dựa trên K task. Mỗi quần thể con thực hiện khám phá một lần.

EME-BI Three-Phase

- **Phase 1 (Trao đổi):** Trao đổi giữa các Task. Cập nhật giá trị rmp.
- **Phase 2 (Khám phá):** Chia quần thể thành K quần thể con dựa trên K task. Mỗi quần thể con thực hiện khám phá **nhiều** lần.
- **Phase 3 (Tìm kiếm):** Local Search trên cá thể tốt nhất **nhiều lần**.

Có thể restart lại toàn bộ thuật toán nếu fitness không cải thiện đáng kể qua T thế hệ, nhưng thử nghiệm có vẻ không hiệu quả ?

New Local Search

- Qua khảo sát các thuật toán tối ưu với số chiều lớn không dựa trên phân rã (non-decomposition), đa số sử dụng Local Search là **MTS-LS1** và **L-BFGS-B**.
- Chiến thuật chọn Local Search dựa trên tỉ lệ cải thiện:

$$LS = \arg\max(lastratio_{MFS}, lastratio_{BFGS})$$

$$ratio = \frac{Fit_{before} - Fit_{after}}{|Fit_{before}|}$$

- **Chiến thuật restart mới:** Ở mỗi thế hệ, ta thực hiện Local Search hai lần: một lần chọn LS theo tỉ lệ cải thiện (như trên) để current-best khám phá thêm nếu như chưa phải cực tiểu địa phương, một lần dùng L-BFGS-B với vector xuất phát ngẫu nhiên (hy vọng may mắn tìm được nghiệm tốt hơn).

Nhận xét

- **Thuật toán khám phá:** Gaussian Mutation + jSO, SHADE
- **Thuật toán tìm kiếm:** MTS-LS1, L-BFGS-B

Pros

- Thuật toán có tốc độ hội tụ rất nhanh về tối ưu địa phương.
- Thuật toán có xác suất cao thoát khỏi điểm tối ưu địa phương (với cỡ 250.000 FEs, thuật toán có 6 lần liên tiếp đến được nghiệm tối ưu toàn cục của f_{16}).
- So sánh với EME-BI hai pha trên 24 tasks, EME-BI ba pha thắng 20 lần, hòa 2 lần, thua 2 lần.
- Giải đa nhiệm hai task giống nhau ?

Nhận xét

- **Thuật toán khám phá:** Gaussian Mutation + jSO, SHADE
- **Thuật toán tìm kiếm:** MTS-LS1, L-BFGS-B

Cons

- Thuật toán vẫn khá nhạy cảm với ngưỡng 10^{-8}, cần khám phá tốt hơn khi đã tìm ra điểm gần basin of attraction của tối ưu toàn cục ?
- L-BFGS-B vốn không tốt với các lớp hàm non-smooth (nghiệm tối ưu mà L-BFGS-B tìm được có gradient xấp xỉ tính được khác 0), nên nó nhạy cảm với đa số các hàm trong GNBG. Tuy nhiên, ở các hàm f_{16}, f_{17} thì L-BFGS-B hoạt động khá tốt, là động cơ để thoát khỏi tối ưu địa phương mà trước đây chưa làm được.
- Mặc dù L-BFGS-B có cơ chế tự dừng, nhưng vẫn lãng phí tài nguyên tính toán ?

Bảng 1: So sánh EME-BI hai pha và ba pha

Function	BFitness	Consumed FEs(EMEBI2, EMEBI3)
F1	-1081.98	25000, 2476
F2	-703.09	50000 (-702.97), 12815 (-703.08)
F3	-357.58	49610, 5904
F4	-382.62	17286, 1476
F5	-337.10	30771, 9335
F6	-186.86	35832, 25968
F7	-912.81	50000 (-912.80), 41818
F8	-656.79	46577, 5916
F9	-884.70	49961, 50000 (không HT)
F10	-604.97	50000 (-604.96), 38090 (-604.97480)
F11	-118.08	49204, 8141
F12	-1002.48	15689, 32627

Bảng 2: So sánh EME-BI hai pha và ba pha

Function	BFitness	Consumed FEs(EMEBI2, EMEBI3)
F13	-216.73	14014, 11554
F14	-194.04	49961, 26135
F15	-234.28	37063, 23831
F16	-5000.00	50000(0 HT), 13700
F17	-5000.00	25487, 9321
F18	-5000.00	14179, 7139
F19	-5000.00	31857, 8335
F20	-100.00	18844, 19640
F21	-50.00	50000(0 HT), 20643
F22	-999.99	16713, 11701
F23	-100.00	42585, 9335
F24	-98.90	24164, 23788

Giải pháp cho bài toán tối ưu nhiều chiều

Ngoài việc giải trực diện bài toán, chúng ta còn vài cách khác sau đây:

- Phân rã không gian
- Phân rã biến
- Giảm chiều

Giảm chiều

Ở khía cạnh lý thuyết, bổ đề **Johnson-Lindenstrauss** chỉ ra rằng, khi giảm chiều quần thể $\mathbb{R}^{N\times d}$ về $\mathbb{R}^{N\times k}$, tốt nhất là

$$d \geq k \geq \frac{8\log N}{\varepsilon^2}$$

với $0 < \varepsilon < 1$, để thông tin ít bị mất mát.
Tuy nhiên, với $d = 30$, thì N lại rất nhỏ. Vậy **giảm chiều không phù hợp ?**

Algorithm 1 Đồng tiến hóa

Require: Tập các biến $\{x_1, x_2, \ldots, x_D\}$

1: Phân rã tập biến thành các tập con $S_1, S_2, \ldots, S_r$
2: Tạo một context vector c
3: **while** not terminateCondition **do**
4: **for** $k = 1$ to r **do**
5: Giải subproblem S_k với context vector
6: **end for**
7: **end while**

Khảo sát các thuật toán phân rã biến

- **Nhận diện đơn điệu:** x_i được gọi là tách rời (không liên kết) với x_j nếu như đã có $f(x_i, x_j, ...) < f\left(x_i^{'}, x_j, ...\right)$ thì cũng phải có $f\left(x_i, x_j^{'}, ...\right) < f\left(x_i^{'}, x_j^{'}, ...\right)$ (1)
  Đại diện cho ý tưởng này là thuật toán FVIL với độ phức tạp $O(4D \log D \times N)$ trong đó D là số chiều, N là số lần kiểm tra tính đúng đắn của (1).
- **Nhận diện phi tuyến:** x_i được gọi là không thể tách rời (liên kết) với x_j nếu như

$$f(x_i + \delta, x_j, ...) - f(x_i, x_j, ...) \neq f\left(x_i + \delta, x_j^{'}, ...\right) - f\left(x_i + \delta, x_j^{'}, ...\right)$$

  → Lớp thuật toán Differential Grouping, độ phức tạp cỡ $O(D^2)$.