Relatório da segunda lista de exercícios

Herberth Amaral

Departamento de Ciência da Computação
Universidade Estadual de Montes Claros
herberthamaral@gmail.com
15 de outubro de 2015

1 Introdução

Este trabalho contempla as análises feitas na segunda lista de exercícios da disciplina de Computação Evolutiva. Foram implementados e analisados três algoritmos evolutivos: recozimento simulado (*simulated annealing - SA*), evolução diferencial (*differential evolution - DE*) e estratégia evolutiva (*evolution strategy - ES*) do tipo $\mu + \lambda$. Os testes foram executados com critérios de parada baseado em número de iterações e em tempo e utilizaram a função rastrigin (com e sem restrições e com 3, 5 e 10 dimensões) como *benchmark*. Os resultados mostram que o algoritmo de evolução diferencial apresentou melhor performance que os demais.

2 Desenvolvimento

Os algoritmos foram implementados utilizando a linguagem de programação *Python* e utilizando os parâmetros mostrados à seguir.

2.1 Todos os algoritmos:

1. **Critério de parada**: 10.000 iterações e 20 segundos de execução;
2. **Problema**: Rastrigin ($\underset{x}{argmin}(f(x)) = 10n + \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i)]| - 5.12 \leq x \leq 5.12$)com 3, 5 e 10 dimensões, com e sem restrições;
3. **Restrições**: $g_i(x) = sen(2\pi x_i) + 0.5 \leq 0, \forall i \in \{1, 2, ..., ng\}$ e $h_j(x) = sen(2\pi x_j) + 0.5 = 0, \forall j \in \{1, 2, ..., nh\}$
4. **Valores de n (dimensões)**: 3, 5 e 10

2.2 Recozimento simulado:

1. **Temperaturas inicial e final:** $T_0 = 1e5, T_f = 1e-10$
2. **Taxa de aprendizado:** $\alpha = 0.995$

2.3 Evolução diferencial:

1. **População**: 20
2. **Recombinação**: 0.995

2.4 Estratégia evolutiva:

1. μ: 20
2. λ: 5
3. σ_i: 20
4. **Operador de recombinação:[1]**

$$\forall l = 1, \ldots, \lambda : \quad \mathbf{a}_l \leftarrow \begin{cases} \sigma_l \leftarrow \langle \sigma \rangle \mathbf{e}^{\tau \mathrm{N}_l(0,1)}, \\ \mathbf{y}_l \leftarrow \langle \mathbf{y} \rangle + \sigma_l \mathbf{N}_l(\mathbf{0}, \mathbf{1}), \\ F_l \leftarrow F(\mathbf{y}_l), \end{cases} \tag{1}$$

3 Resultados

3.1 Recozimento simulado

Foi possível observar que, diferentemente dos outros algoritmos, o recozimento simulado não apresenta uma "tendência de queda": inicia-se com um certo *fitness* e logo após algumas iterações esse mesmo fitness piora para só depois voltar ao nível normal e finalmente ir abaixo dele. Isso pode ser explicado pelo fato que o algoritmo de recozimento simulado aceita e explora soluções piores quando a temperatura ainda está alta e passa a ter menos tolerância a resultados piores a medida que a temperatura decai. Tal evolução pode ser vista nas imagens abaixo:

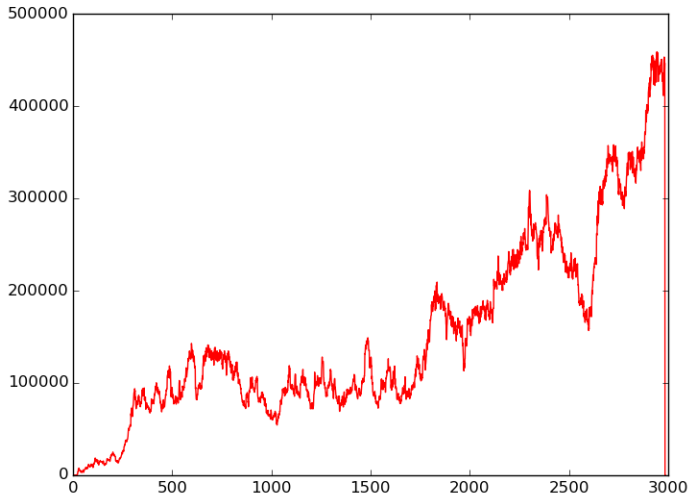

Figura 1: *Evolução do fitness do algoritmo de recozimento simulado com $n = 3$ com critério de parada por iterações*

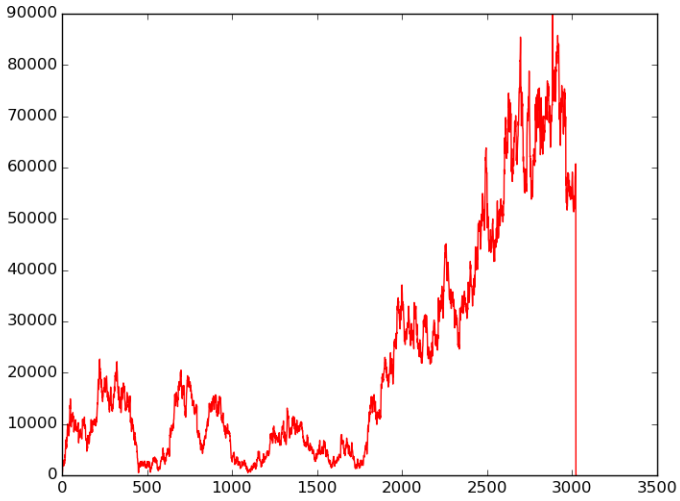

Figura 2: *Mais um exemplo de evolução do fitness do algoritmo de recozimento simulado*

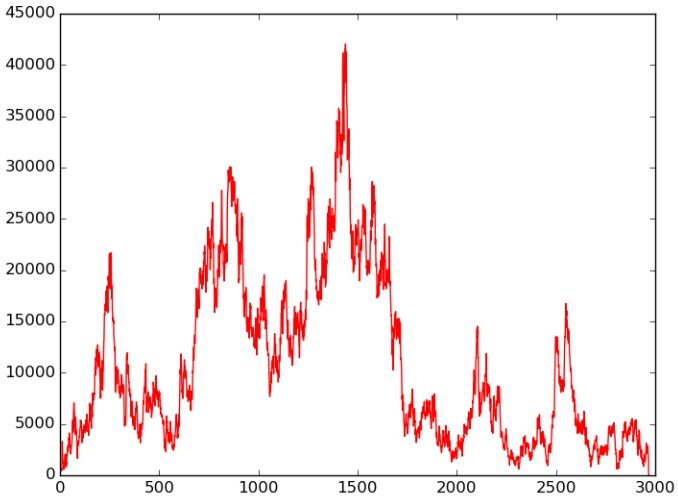

Figura 3: *Mais um exemplo de evolução do fitness do algoritmo de recozimento simulado*

3.2 Evolução diferencial

Pelos gráficos foi possível observar que o algoritmo de evolução diferencial também explora bem seu espaço de busca, porém de forma diferente do recozimento simulado. Ao invés de ter uma tendência de alta para só depois baixar, o algoritmo apresenta uma tendência de baixa o tempo inteiro, mas com um alto desvio padrão por janela de tempo no início que diminui no final.

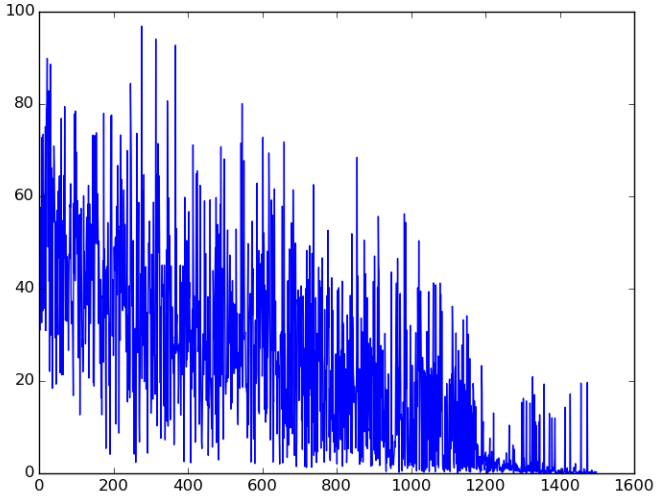

Figura 4: *Exemplo de evolução do fitness do algoritmo de evolução diferencial com critério de iterações para parada*

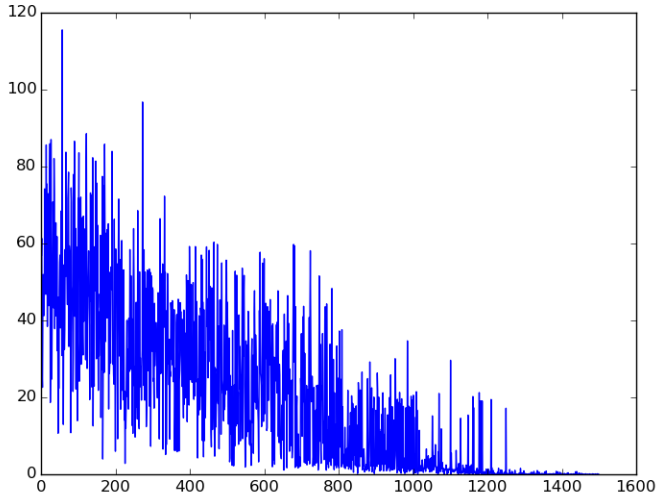

Figura 5: *Exemplo de evolução do fitness do algoritmo de evolução diferencial com critério de tempo para parada*

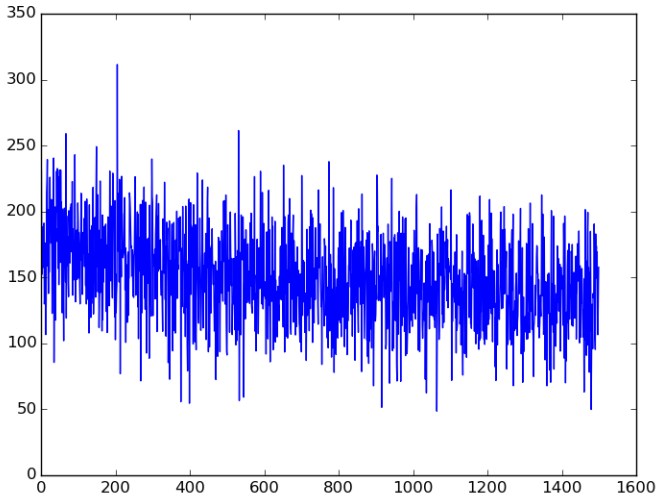

Figura 6: *Exemplo de evolução do fitness do algoritmo de evolução diferencial com critério de tempo para parada em 10 dimensões.*

3.3 Estratégia evolutiva

Os algoritmos de estratégia evolutiva são os que menos exploraram o espaço de busca e tiveram uma convergência rápida (não necessariamente prematura). A convergência rápida pode ser explicada por causa das taxas de mutação baseadas no desvio padrão que decaem relativamente rápido. O algoritmo de estratégia evolutiva foi o único em que o critério de tempo não fez sentido por causa da convergência rápida.

Tentou-se inserir pertubações quando o desvio padrão das dez últimas iterações fosse zero. O espaço de busca foi melhor explorado, mas os resultados pioraram.

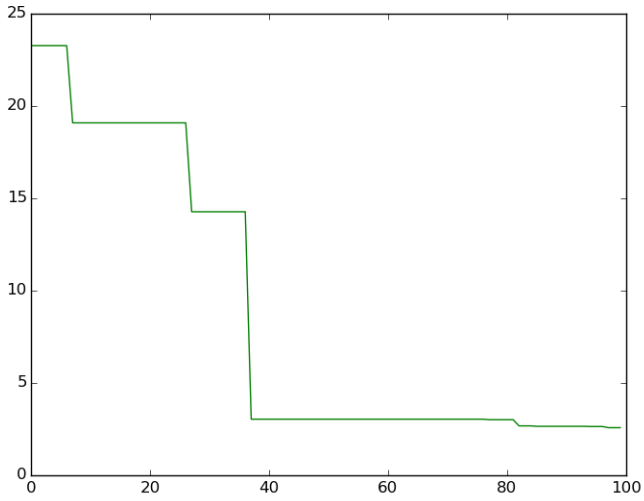

Figura 7: *Exemplo de evolução do fitness do algoritmo de estratégia evolutiva* $\mu + \lambda$

Para melhor visualização, as figuras abaixo contemplam a evolução do *fitness* somente das 100 primeiras iterações.

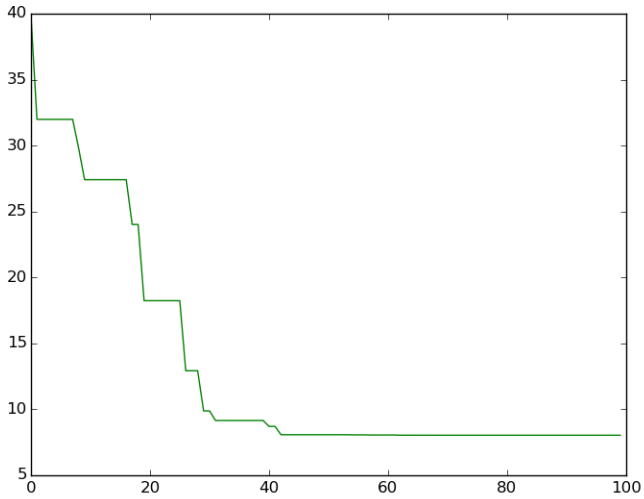

Figura 8: *Mais um exemplo de evolução do fitness do algoritmo de estratégia evolutiva $\mu + \lambda$*

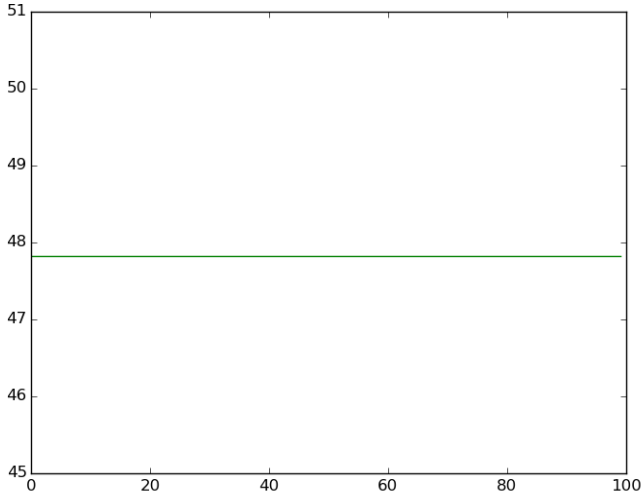

Figura 9: *Neste caso, o algoritmo ficou preso num mínimo local*

3.4 Comparativo dos três algoritmos em vista do critério de tempo

Os resultados mostram a maior eficiência do algoritmo de evolução diferencial. Vale ressaltar que a convergência se dá muito antes do tempo completar, não sendo necessário um tempo de execução tão alto para seu caso (20 segundos).

Algoritmo	**Valor médio (DP)**	**Valor mínimo**
Recozimento simulado	0.97 (1.21)	0.012
Estratégia evolutiva	1.91 (1.094)	0.060
Evolução diferencial	1.62e-09 (1.76e-09)	3.17e-10

Referências

[1] BEYER, Hans-Georg (2007), *Evolution strategies*. Disponível em `http://www.scholarpedia.org/article/Evolution_strategies`. Acessado em 14/10/2015.

[2] GUIMARÃES, Frederico Gadelha, *Algoritmos de evoluçao diferencial para otimização e aprendizado de máquina*. Anais do IX Congresso Brasileiro de Redes Neurais/Inteligência Computacional (IX CBRN) - 2009.