EEE933 - Estudo de Caso 04

Comparação de desempenho de operadores em um algoritmo de otimização

Apresentação

Algoritmos baseados em populações são uma alternativa comum para a solução de problemas de otimização em engenharia. Tais algoritmos normalmente consistem de um ciclo iterativo, no qual um conjunto de soluções-candidatas ao problema são repetidamente sujeitas a operadores de variação e seleção, de forma a promover uma exploração do espaço de variáveis do problema em busca de um ponto de ótimo (máximo ou mínimo) de uma dada função-objetivo.

Dentre estes algoritmos, um método que tem sido bastante utilizado nos últimos anos é conhecido como *evolução diferencial* (DE do inglês *differential evolution*)(Storn and Price 1997). De forma simplificada, este método é composto pelos seguintes passos:

0. Entrada: $N,\ n_{iter},\ recpars,\ mutpars$
1. $t \leftarrow 0$
2. $X_t \leftarrow \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ (população inicial)
3. $\vec{f_t} \leftarrow f(X_t)$
4. Enquanto $(t < n_{iter})$
    1. $V_t \leftarrow$ mutação$(X_t,\ mutpars)$
    2. $U_t \leftarrow$ recombinação$(X_t,\ V_t,\ recpars)$
    3. $\vec{j_t} \leftarrow f(U_t)$
    4. $(X_{t+1},\ \vec{f_{t+1}}) \leftarrow$ seleção$(X_t,\ U_t,\ \vec{f_t},\ \vec{j_t})$
    5. $t \leftarrow t + 1$
5. Saída: $(X_t,\ \vec{f_t})$

Suponha que um pesquisador está interessado em investigar o efeito de diferentes operadores de *recombinação* (etapa **4.2**) no desempenho do algoritmo para uma dada classe de problemas. Para tal, o mesmo implementa o algoritmo e diversos operadores de forma padronizada, na forma do pacote ExpDE(Campelo and Botelho 2016), cuja versão de desenvolvimento encontra-se disponível no Github.

Atividades

Como forma de análise preliminar deste problema, cada grupo terá como tarefa a comparação experimental de alguns operadores de recombinação, em um único problema de teste. O objetivo deste estudo é responder às perguntas:

Há alguma diferença no desempenho médio do algoritmo quando equipado com estes diferentes operadores, para o problema de teste utilizado? Caso haja, qual o melhor operador em termos de desempenho médio (quanto menor o valor retornado, melhor o algoritmo), e qual a magnitude das diferenças encontradas? Há algum operador que deva ser recomendado em relação aos demais?

Os seguintes parâmetros experimentais são dados para este estudo:

- Mínima diferença de importância prática entre qualquer par de algoritmos (padronizada, em termos do coeficiente d de Cohen): $(d^* = \delta^*/\sigma) = 0.25$
- Significância desejada: $\alpha = 0.05$
- Potência mínima desejada (para o caso $d = d^*$): $\pi = 1 - \beta = 0.85$

Informações operacionais

Para instalar e carregar o pacote no computador, os seguintes comandos podem ser utilizados:

```
install.packages("ExpDE")
library(ExpDE)
```

Os parâmetros fixos do algoritmo (ao longo de toda a experimentação) são dados por:

```
selpars  <- list(name  = "selection_standard")
stopcrit <- list(names = "stop_maxeval", maxevals = 60000, maxiter = 1000)
probpars <- list(name  = "sphere", xmin  = -seq(1,20), xmax  = 20 + 5 * seq(5, 24))
```

Os operadores de recombinação que deverão ser comparados por cada grupo passaram por uma etapa anterior de ajuste de parâmetros. Suas configurações são dadas por:

```
# Equipe *We R Group*

## Arithmetic recombination
recpars1 <- list(name = "recombination_arith")
mutpars1 <- list(name = "mutation_rand", f = 4)
popsize1 <- 300

## Binomial recombination
recpars2 <- list(name = "recombination_bin", cr = 0.7)
mutpars2 <- list(name = "mutation_best", f = 3)
popsize2 <- 300

## BLX-alpha recombination
recpars3 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars3 <- list(name = "mutation_rand", f = 4)
popsize3 <- 230

## Eigenvector-based binomial recombination
recpars4 <- list(name = "recombination_eigen", othername = "recombination_bin", cr = 0.9)
mutpars4 <- list(name = "mutation_best", f = 2.8)
popsize4 <- 85
```

```
# Equipe *Corcel Team*

## Exponential Recombination
recpars1 <- list(name = "recombination_exp", cr = 0.6)
mutpars1 <- list(name = "mutation_best", f = 2)
popsize1 <- 130

## Geometric recombination
recpars2 <- list(name = "recombination_geo", alpha = 0.6)
mutpars2 <- list(name = "mutation_rand", f = 1.2)
popsize2 <- 70

## Linear BGA recombination
recpars3 <- list(name = "recombination_lbga")
mutpars3 <- list(name = "mutation_rand", f = 4.5)
```

```
popsize3 <- 300

## BLX-alpha-beta recombination
recpars4 <- list(name = "recombination_blxAlphaBeta", alpha = 0.1, beta = 0.4)
mutpars4 <- list(name = "mutation_rand", f = 3)
popsize4 <- 80
```

```
# Equipe *Grupo 2*

## Flat recombination
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0, beta = 0)
mutpars1 <- list(name = "mutation_rand", f = 4)
popsize1 <- 200

## Linear recombination
recpars2 <- list(name = "recombination_linear")
mutpars2 <- list(name = "mutation_rand", f = 1.5)
popsize2 <- 250

## Min-max recombination
recpars3 <- list(name = "recombination_mmax", lambda = 0.25)
mutpars3 <- list(name = "mutation_best", f = 4)
popsize3 <- 375

## N-point recombination
recpars4 <- list(name = "recombination_npoint", N = 17)
mutpars4 <- list(name = "mutation_rand", f = 2.2)
popsize4 <- 225
```

```
# Equipe *Mythbusters*

## One-point recombination
recpars1 <- list(name = "recombination_onepoint", K = 17)
mutpars1 <- list(name = "mutation_best", f = 2.4)
popsize1 <- 225

## P-best recombination
recpars2 <- list(name = "recombination_pbest", cr = 0.25)
mutpars2 <- list(name = "mutation_rand", f = 3.5)
popsize2 <- 325

## SBX recombination
recpars3 <- list(name = "recombination_sbx", eta = 90)
mutpars3 <- list(name = "mutation_best", f = 4.5)
popsize3 <- 200

## Wright recombination
recpars4 <- list(name = "recombination_wright")
mutpars4 <- list(name = "mutation_best", f = 4.8)
popsize4 <- 113
```

```
# Equipe *Ta tranquilo ta lascado*

## Eigenvector-based binomial recombination
recpars1 <- list(name = "recombination_eigen", othername = "recombination_bin", cr = 0.9)
mutpars1 <- list(name = "mutation_best", f = 2.8)
popsize1 <- 85

## SBX recombination
recpars2 <- list(name = "recombination_sbx", eta = 90)
mutpars2 <- list(name = "mutation_best", f = 4.5)
popsize2 <- 200

## Linear recombination
recpars3 <- list(name = "recombination_linear")
mutpars3 <- list(name = "mutation_rand", f = 1.5)
popsize3 <- 250

## BLX-alpha-beta recombination
recpars4 <- list(name = "recombination_blxAlphaBeta", alpha = 0.1, beta = 0.4)
mutpars4 <- list(name = "mutation_rand", f = 3)
popsize4 <- 80
```

```
# Equipe *Time 7*

## One-point recombination
recpars1 <- list(name = "recombination_onepoint", K = 17)
mutpars1 <- list(name = "mutation_best", f = 2.4)
popsize1 <- 225

## Min-max recombination
recpars2 <- list(name = "recombination_mmax", lambda = 0.25)
mutpars2 <- list(name = "mutation_best", f = 4)
popsize2 <- 375

## P-best recombination
recpars3 <- list(name = "recombination_pbest", cr = 0.25)
mutpars3 <- list(name = "mutation_rand", f = 3.5)
popsize3 <- 325

## Geometric recombination
recpars4 <- list(name = "recombination_geo", alpha = 0.6)
mutpars4 <- list(name = "mutation_rand", f = 1.2)
popsize4 <- 70
```

```
# Equipe *Team 5*

## Flat recombination
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0, beta = 0)
mutpars1 <- list(name = "mutation_rand", f = 4)
popsize1 <- 200

## Exponential Recombination
recpars2 <- list(name = "recombination_exp", cr = 0.6)
```

```
mutpars2 <- list(name = "mutation_best", f = 2)
popsize2 <- 130

## BLX-alpha recombination
recpars3 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars3 <- list(name = "mutation_rand", f = 4)
popsize3 <- 230

## Wright recombination
recpars4 <- list(name = "recombination_wright")
mutpars4 <- list(name = "mutation_best", f = 4.8)
popsize4 <- 113
```

Cada observação individual do desempenho do algoritmo com um dado operador pode ser obtida através dos comandos abaixo:

```
out <- ExpDE(popsizeX, mutparsX, recparsX, selpars, stopcrit, probpars)
out$Fbest
```

onde *popsizeX*, *mutparsX* a *recparsX* devem ser substituídos pelas variáveis apropriadas.

Outras definições

Este estudo de caso consiste das seguintes etapas:

1. Formulação das hipóteses de teste;
2. Cálculo do tamanho amostral;
3. Coleta e tabulação dos dados;
4. Teste das hipóteses;
5. Estimação dos tamanhos de efeito e dos intervalos de confiança;
6. Verificação das premissas dos testes;
7. Derivação de conclusões;
8. Discussão sobre possíveis limitações do estudo e sugestões de melhoria.

Lembre-se que as conclusões devem ser colocadas no contexto da pergunta técnica de interesse.

Relatório

Cada grupo deverá entregar um relatório detalhando o experimento e a análise dos dados. O relatório será avaliado de acordo com os seguintes critérios:

- Obediência ao formato determinado (ver abaixo);
- Reproducibilidade dos resultados;
- Qualidade técnica;
- Estrutura da argumentação;
- Correto uso da linguagem (gramática, ortografia, etc.);

O relatório deve *obrigatoriamente* ser produzido utilizando R Markdown, e deve conter todo o código necessário para a reprodução da análise obtida, embutido na forma de blocos de código no documento. Os grupos devem enviar:

- O arquivo **.Rmd** do relatório.
- O arquivo de dados utilizado.

O arquivo **.Rmd** deve ser capaz de ser compilado em um pdf sem erros, e deve assumir que o arquivo de dados se encontra no mesmo diretório do arquivo do relatório. Modelos de estudos de caso estão disponíveis aqui e aqui.

Importante: Salve seu arquivo **.Rmd** em UTF-8 (para evitar erros na compilação em outros sistemas).
Importante: Inclua no relatório os papéis desempenhados por cada membro da equipe (Relator, Verificador etc.)

Relatórios serão aceitos em português, inglês ou espanhol.

Entrega

Os arquivos deverão ser enviados via *e-mail* para o endereço fcampelo@ufmg.br. O título do e-mail deve seguir o padrão "**[EEE933_2016-1_EC04] Nome_da_equipe**" (sem as aspas). A data-limite para o recebimento dos arquivos é **segunda-feira (16/05) às 11:00h**

Referências

Campelo, Felipe, and Moises Botelho. 2016. "Experimental Investigation of Recombination Operators for Differential Evolution." In *Proc. Genetic and Evolutionary Computation Conference - GECCO'2016*. Denver, CO – To appear.

Storn, Rainer, and Kenneth Price. 1997. "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *J. of Global Optimization* 11 (4): 341–59.