EEE933 - Design and Analysis of Experiments

Case Study 03

This version was compiled on May 22, 2018

Experiment: Comparação de desempenho de duas configurações de um algoritmo de otimização, parte II

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 duas configurações distintas deste algoritmo em seu desempenho para uma dada classe de problemas de otimização.

Atividades. Como forma de análise deste problema, cada equipe terá como tarefa a comparação experimental de duas configurações em uma classe de problemas, representada por um conjunto de instâncias de teste. O objetivo deste estudo é responder às seguintes perguntas:

Há alguma diferença no desempenho médio do algoritmo quando equipado com estas diferentes configurações, ***para a classe de problemas de interesse****? Caso haja, qual a melhor configuração em termos de desempenho médio (atenção: quanto menor o valor retornado, melhor o algoritmo), e qual a magnitude das diferenças encontradas? Há alguma configuração que deva ser recomendada em relação à outra?*

Os seguintes parâmetros experimentais são dados para este estudo:

- Mínima diferença de importância prática (padronizada): $(d^* = \delta^*/\sigma) = 0.5$
- Significância desejada: $\alpha = 0.05$
- Potência mínima desejada (para o caso $d = d^*$): $\pi = 1 - \beta = 0.8$

Informações operacionais. Para a execução dos experimentos, instale os pacotes `ExpDE` (Campelo and Botelho, 2016) e `smoof` (Bossek, 2017):

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

A classe de funções de interesse para este teste é composta por funções Rosenbrock (Rosenbrock, 1960) (Pohlheim, 2005) de dimensão entre 2 e 150. To generate a Rosenbrock function of a given dimension (p.ex., $dim = 10$) just do:

```
suppressPackageStartupMessages(library(smoof))
dim <- 10
fn <- function(X){
  if(!is.matrix(X)) X <- matrix(X, nrow = 1) # <- if a single vector is passed as X

  Y <- apply(X, MARGIN = 1,
             FUN = smoof::makeRosenbrockFunction(dimensions = dim))

  return(Y)
}

# testing the function on a matrix composed of 2 points
X <- matrix(runif(20), nrow = 2)
fn(X)
```

```
#  [1] 134.5490 198.7516
```

The variable limits for the Rosenbrock function are defined as $-5 \leq x_i \leq 10,\ i = 1, \ldots, n$.
For any given problem dimension dim, the following parameters are set as:

```
dim <- 10 # for instance

selpars  <- list(name  = "selection_standard")
stopcrit <- list(names = "stop_maxeval", maxevals = 5000 * dim, maxiter = 100 * dim)
probpars <- list(name  = "fn", xmin  = rep(-5, dim), xmax  = rep(10, dim))
popsize = 5 * dim
```

As configurações que deverão ser comparadas por cada equipe passaram por uma etapa anterior de ajuste de parâmetros. Suas definições são dadas por:

```
# Equipe A

## Config 1
recpars1 <- list(name = "recombination_arith")
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 2
recpars2 <- list(name = "recombination_bin", cr = 0.7)
mutpars2 <- list(name = "mutation_best", f = 3)
```

```r
# Equipe B

## Config 1
recpars1 <- list(name = "recombination_exp", cr = 0.6)
mutpars1 <- list(name = "mutation_best", f = 2)

## Config 2
recpars2 <- list(name = "recombination_geo", alpha = 0.6)
mutpars2 <- list(name = "mutation_rand", f = 1.2)
```

```r
# Equipe C

## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0, beta = 0)
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 2
recpars2 <- list(name = "recombination_linear")
mutpars2 <- list(name = "mutation_rand", f = 1.5)
```

```r
# Equipe D
## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 4
recpars2 <- list(name = "recombination_eigen",
                 othername = "recombination_bin", cr = 0.9)
mutpars2 <- list(name = "mutation_best", f = 2.8)
```

```r
# Equipe E
## Config 1
recpars1 <- list(name = "recombination_lbga")
mutpars1 <- list(name = "mutation_rand", f = 4.5)

## Config 2
recpars2 <- list(name = "recombination_blxAlphaBeta", alpha = 0.1, beta = 0.4)
mutpars2 <- list(name = "mutation_rand", f = 3)
```

```r
# Equipe F

## Config 1
recpars1 <- list(name = "recombination_mmax", lambda = 0.25)
mutpars1 <- list(name = "mutation_best", f = 4)

## Config 4
recpars1 <- list(name = "recombination_npoint", N = 17)
```

```
mutpars1 <- list(name = "mutation_rand", f = 2.2)
```

```
# Equipe G

## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0, beta = 0)
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 2
recpars2 <- list(name = "recombination_exp", cr = 0.6)
mutpars2 <- list(name = "mutation_best", f = 2)
```

```
# Equipe H

## Config 1
recpars1 <- list(name = "recombination_eigen",
                 othername = "recombination_bin", cr = 0.9)
mutpars1 <- list(name = "mutation_best", f = 2.8)

## Config 2
recpars2 <- list(name = "recombination_sbx", eta = 90)
mutpars2 <- list(name = "mutation_best", f = 4.5)
```

```
# Equipe I

## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 2
recpars2 <- list(name = "recombination_wright")
mutpars2 <- list(name = "mutation_best", f = 4.8)
```

```
# Equipe J

## Config 1
recpars1 <- list(name = "recombination_mmax", lambda = 0.25)
mutpars1 <- list(name = "mutation_best", f = 4)

## Config 2
recpars2 <- list(name = "recombination_geo", alpha = 0.6)
mutpars2 <- list(name = "mutation_rand", f = 1.2)
```

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

```r
suppressPackageStartupMessages(library(ExpDE))
# Run algorithm on problem:
out <- ExpDE(mutpars  = mutparsX,
             recpars  = recparsX,
             popsize  = popsize,
             selpars  = selpars,
             stopcrit = stopcrit,
             probpars = probpars,
             showpars = list(show.iters = "dots", showevery  = 20))

# Extract observation:
out$Fbest
```

onde *mutparsX* e *recparsX* devem ser substituídos pelas variáveis apropriadas (e.g., *mutpars1*, *mutpars2* etc.).

Outras definições. Este estudo de caso consiste das seguintes etapas:

1. Formulação das hipóteses de teste;
2. Cálculo dos tamanhos amostral (quantas instâncias testar? Quantas repetições de cada algoritmo por instância?);
3. Coleta e tabulação dos dados;
4. Teste das hipóteses;
5. Estimação da magnitude da diferença entre os métodos (incluindo intervalo 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 das perguntas técnicas de interesse.

Relatório. Cada equipe 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 (opcionalmente utilizando estilos distintos, como o do presente documento), 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** para geração do relatório.
- O arquivo **.pdf** compilado a partir do **.Rmd**.
- O arquivo de dados utilizado, em formato **.csv**.

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 no repositório da disciplina no github. Caso a equipe deseje utilizar o estilo do presente documento, pode consultar seu código-fonte no repositório (note que o mesmo requer a instalação do pacote *pinp*).

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 ou inglês.

Entrega. Os arquivos relativos a este estudo de caso (pdf + rmd + csv) deverão ser comprimidos em um .ZIP e submetidos via Moodle, na atividade **Case Study 02**, até a data-limite de **Terça-feira, 5 de junho de 2018, às 23:55h**. Após esta data o sistema estará fechado para recebimento.

Importante: Apenas uma submissão por equipe é necessária.

Importante: Relatórios não serão recebidos por e-mail ou em formato impresso.

References

Bossek J (2017). "smoof: Single- and Multi-Objective Optimization Test Functions." *The R Journal*. URL https://journal.r-project.org/archive/2017/RJ-2017-004/index.html.

Campelo F, Botelho M (2016). "Experimental Investigation of Recombination Operators for Differential Evolution." In *Proc. Genetic and Evolutionary Computation Conference - GECCO'2016*.

Pohlheim H (2005). "Examples of Objective Functions." online. URL http://www.geatbx.com/download/GEATbx_ObjFunExpl_v37.pdf.

Rosenbrock HH (1960). "An Automatic Method for Finding the Greatest or least Value of a Function." *Computer Journal*, **3**(3), 175–184.

Storn R, Price K (1997). "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *J. of Global Optimization*, **11**(4), 341–359.