PCC170 - Projeto e Análise de Experimentos Computacionais

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é parcialmente baseado no conteúdo de

- Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. *The irace package: Iterated Racing for Automatic Algorithm Configuration*. Operations Research Perspectives, 2016.
- Manuel López-Ibáñez, Leslie Pérez Cáceres, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. *The irace package: User Guide*. Version 3.5, 2021.

Licença

Definição de parâmetros

Introdução

Algoritmos para lidar com problemas computacionalmente difíceis têm vários parâmetros que influenciam seu comportamento de busca.

Esses algoritmos incluem algoritmos exatos, como branch-and-bound, pacotes para programação inteira e algoritmos aproximados, como algoritmos de busca local estocástica.

Os parâmetros podem ser classificados grosseiramente em parâmetros **numéricos** e **categóricos**.

Definição de parâmetros

Parâmetros numéricos e categóricos

Exemplos de parâmetros numéricos são a aspiração na busca tabu ou a taxa de evaporação de feromônio em colônia de formigas.

Algoritmos podem também ser compostos por um conjunto de componentes intercambiáveis, e.g.:

- Diferentes estratégias de ramificação no *branch-and-bound*;
- Diferentes operadores de cruzamento em algoritmos evolucionários;
- Diferentes algoritmos de busca local.

Esses componentes intercambiáveis geralmente são representados como parâmetros categóricos.

Definição de parâmetros

Introdução

A escolha de uma configuração apropriada de parâmetros é em si um problema de otimização difícil.

Chamamos esse problema também de **problema de configuração de algoritmos**.

Este problema é tipicamente um problema estocástico, seja pela característica do algoritmo, que pode incluir decisões aleatórias, seja por causa da instância tratada a cada instante.

A depender das decisões tomadas por nós, calibrar os parâmetros de um algoritmo pode se tornar mais custoso do que resolver o problema para o qual o algoritmo foi projetado.

Introdução

Um aspecto crucial do problema de configuração de algoritmo é que ele é um problema de **generalização**, como ocorre em aprendizado de máquina.

Com base em um conjunto de instâncias de treinamento, o objetivo é encontrar configurações de algoritmo de alto desempenho que tenham bom desempenho em um conjunto instâncias não conhecidas, não disponíveis ao decidir sobre os parâmetros do algoritmo.

Assim, uma suposição que é feita tacitamente é que o conjunto de instâncias de treinamento é representativo das instâncias que o algoritmo enfrenta uma vez empregado na fase de produção.

Definição de parâmetros

Overfitting

O *overfitting* é um conceito em ciência de dados, que ocorre quando um modelo estatístico se ajusta exatamente aos seus dados de treinamento.

Quando isso acontece, o algoritmo infelizmente não pode funcionar com precisão contra dados não conhecidos, anulando seu propósito.

Para prevenir esse tipo de comportamento, parte do conjunto de dados de treinamento é normalmente reservado como o "conjunto de teste".

Se os dados de treinamento tiverem uma taxa de erro baixa e os dados de teste tiverem uma taxa de erro alta, isso indica um *overfitting*.

Definição de parâmetros

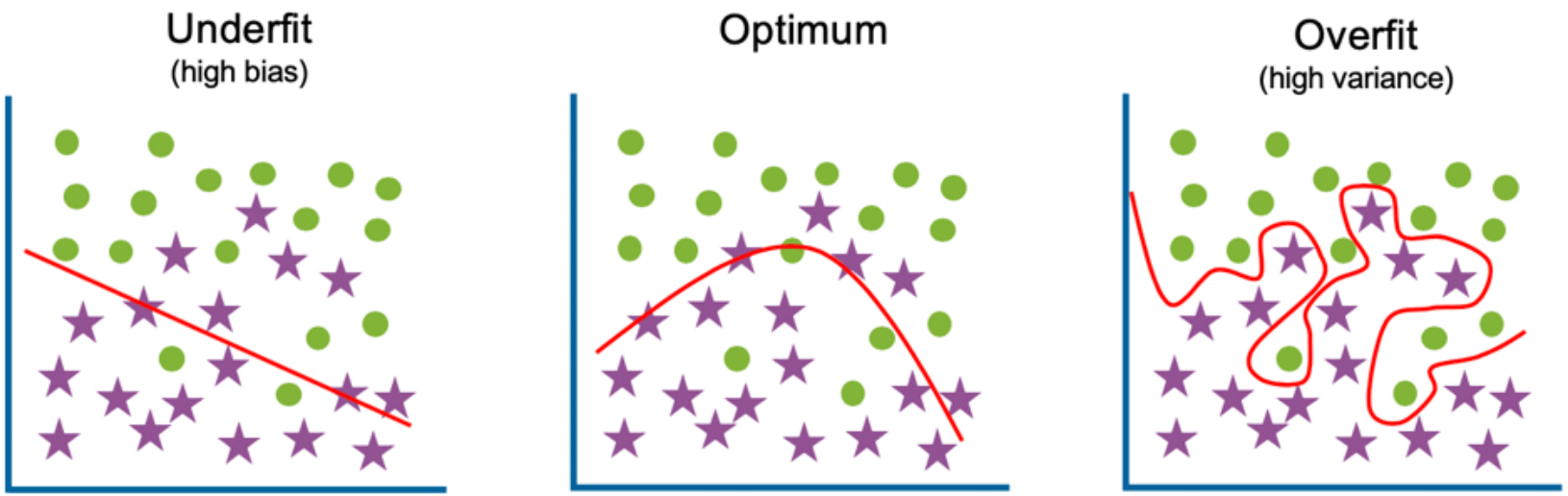
Underfitting

Uma resposta lógica de prevenção ao *overfitting* seria pausar o processo de treinamento mais cedo, ou reduzir a complexidade do modelo eliminando entradas menos relevantes.

No entanto, se pausarmos muito cedo ou excluirmos muitos recursos importantes, poderemos encontrar o problema oposto.

O *underfitting* ocorre quando o modelo não foi treinado o suficiente ou as variáveis de entrada não são significativas o suficiente para determinar uma relação significativa entre as variáveis de entrada e saída.

Definição de parâmetros



fonte: `https://bityli.com/muLkk`

Definição de parâmetros

Quais parâmetros calibrar?

Os parâmetros que podem ser calibrados incluem:

- Número de iterações sem melhorias;
- Percentual de aplicação de busca local;
- Percentual de aplicação de perturbação;
- Taxas de mutação e cruzamento;
- Percentual de elitismo;
- Etc.

Definição de parâmetros

Quais parâmetros calibrar?

Outras aplicações menos ortodoxas incluem:

- Determinação de limitante inferior para o número máximo de iterações;
- Determinação da utilização ou não de componentes.

Definição de parâmetros

Observações

Os valores de parâmetros devem ser os mesmos para todas instâncias, no entanto, há flexibilidade quando se trata de instâncias com estruturas ou dimensões muito discrepantes.

Normalmente, o conjunto total de instâncias deve ser dividido em dois: um menor para calibrar os parâmetros e outro maior para avaliar o desempenho do método computacional.

Definição de parâmetros

Software

Diferentes softwares podem ser utilizados para configuração de parâmetros em métodos computacionais:

- **irace**[a];
- SMAC;
- Bonesa;
- PARAMILS;
- GGA++;
- Revac;
- CRS-*Tuning*;
- *SPOT Sequential Parameter Optimization Tool*.

[a] `https://mlopez-ibanez.github.io/irace`

Definição de parâmetros

irace

O *irace* é um pacote do *R* que implementa um procedimento de corrida iterada, extensão do *Iterated F-race*.

O uso principal é a configuração automática de algoritmos de otimização e decisão, i.e., encontrar as configurações mais apropriadas de um algoritmo para um conjunto de instâncias de um problema.

Definição de parâmetros

irace

Na fase de **treinamento** do ajuste *offline*, uma configuração de algoritmo deve ser determinada em uma quantidade limitada de tempo que otimiza alguma medida de desempenho do algoritmo.

A configuração final do algoritmo é então implantada em uma fase de **produção**, em que o algoritmo é usado para resolver instâncias não vistas anteriormente.

Definição de parâmetros

Procedimentos de corrida

Os procedimentos de corrida, em geral, avaliam um determinado conjunto de configurações candidatas iterativamente em um fluxo de instâncias.

Assim que forem reunidas evidências estatísticas suficientes contra algumas configurações candidatas, estas são eliminadas e a corrida continua apenas com as sobreviventes.

Definição de parâmetros

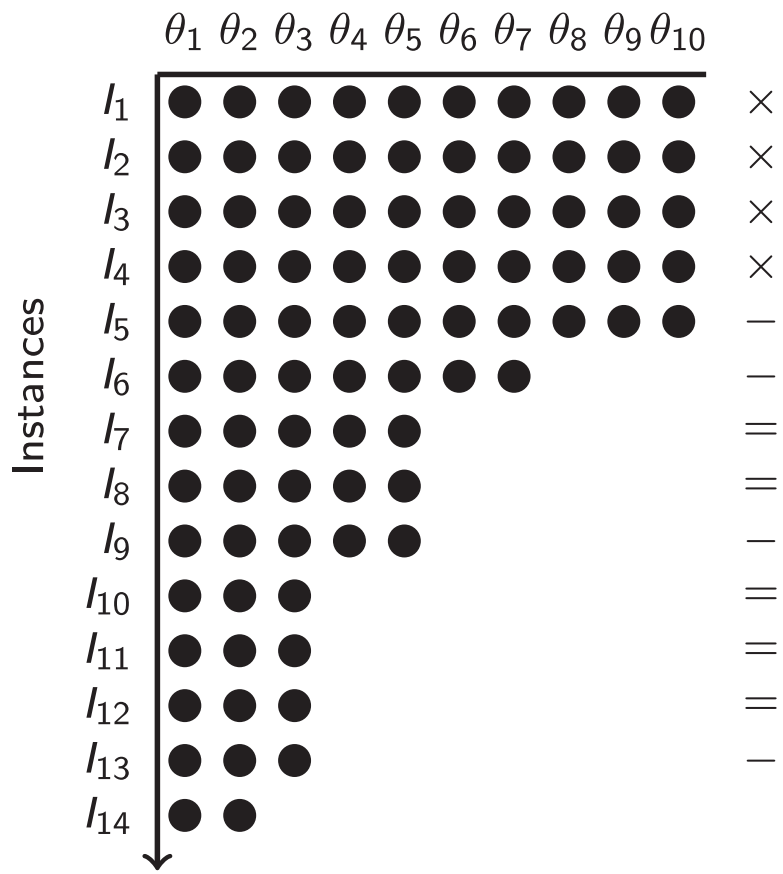


Fig. 1. Racing for automatic algorithm configuration. Each node is the evaluation of one configuration on one instance. '× ' means that no statistical test is performed, ' – ' means that the test discarded at least one configuration, ' = ' means that the test did not discard any configuration. In this example, $T^{\text{first}} = 5$ and $T^{\text{each}} = 1$.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

Funcionamento geral

O *irace* realiza uma amostragem de novas configurações de parâmetros a cada iteração.

A cada amostragem, as melhores configurações são escolhidas e as demais descartadas, após testes estatísticos.

A amostragem é atualizada para se concentrar na região de busca das melhores configurações.

Ao final, um conjunto de configurações não dominantes entre si é apresentada.

Definição de parâmetros

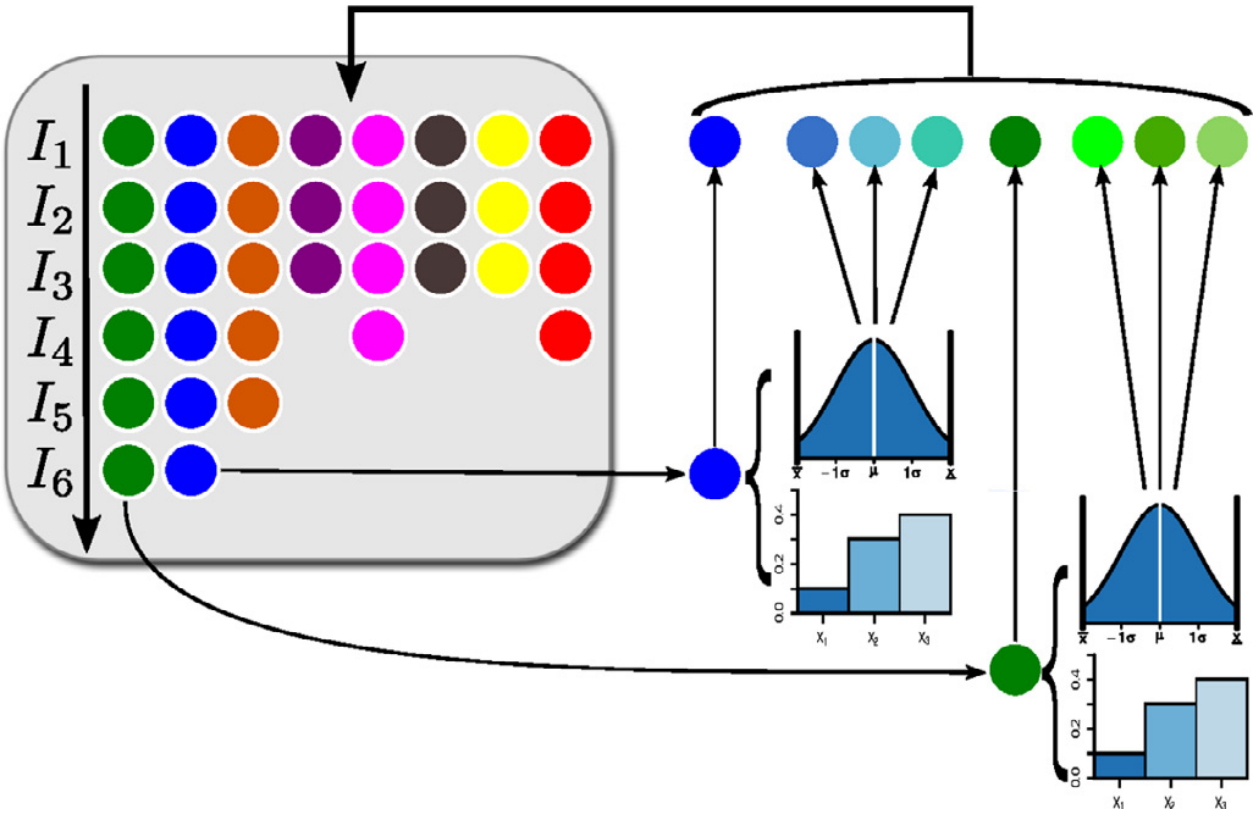


Fig. 2. Scheme of the iterated racing algorithm.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

irace

A entrada do *irace* contém:

- A definição de espaço de parâmetros correspondente ao algoritmo de destino (*target*);
- O conjunto de instâncias para as quais os parâmetros devem ser ajustados (*training instances*);
- O conjunto de opções para *irace* que definem o cenário de configuração (*parameter space*).

Definição de parâmetros

irace

O *irace* procura no espaço de busca de parâmetros por configurações de bom desempenho executando o algoritmo de destino em diferentes instâncias e com diferentes configurações de parâmetros.

Um `targetRunner` deve ser fornecido para executar o algoritmo de destino com uma configuração de parâmetro (ϕ) e instância (i) específicos.

O *targetRunner* atua como uma interface entre a execução do algoritmo e o *irace*: ele recebe a instância e a configuração como argumentos e deve retornar a avaliação da execução do algoritmo alvo.

Definição de parâmetros

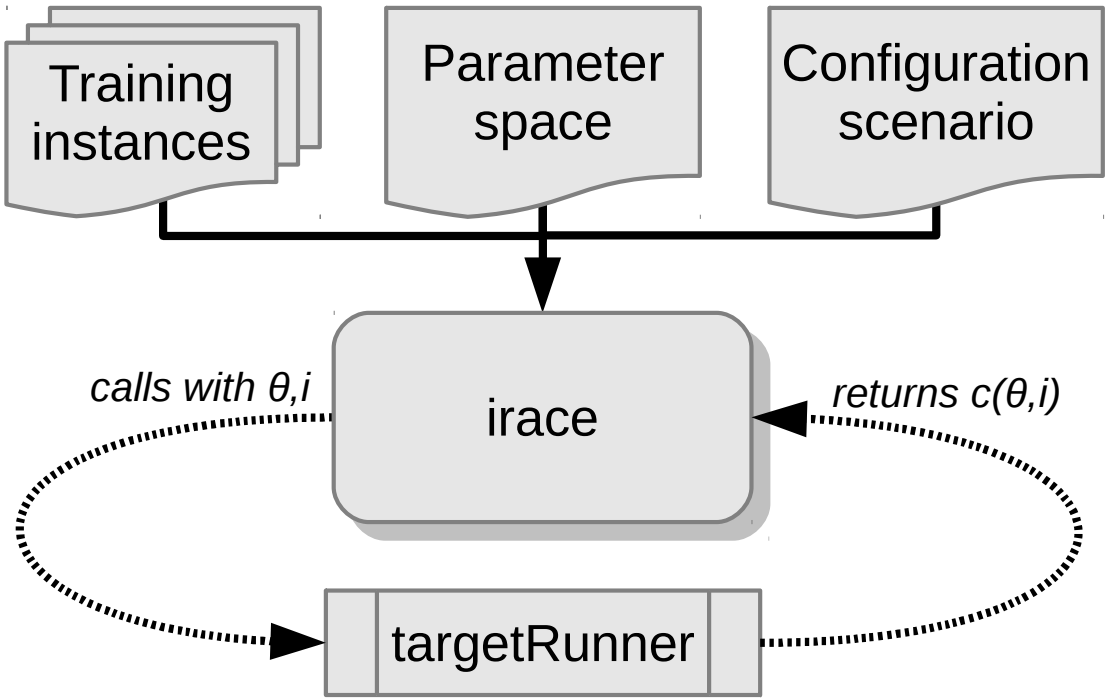


Figure 1: Scheme of **irace** flow of information.

fonte: `https://bityli.com/EZlut`

Definição de parâmetros

Organização

O *irace* é executado por um script de terminal chamado *target-runner*, sem modo gráfico.

A organização dos dados é realizada de acordo com arquivos de configuração e uma estrutura de diretórios.

Definição de parâmetros

Organização

Os principais arquivos são:

scenario.txt: Configurações do *irace*, como onde estão localizados os demais arquivos e diretórios e opções da corrida;

parameters.txt: Define os parâmetros por nome, flag, tipo, possíveis valores e condições;

forbidden.txt: Define condições proibidas para combinação de parâmetros.

Definição de parâmetros

Organização

A estrutura de diretórios pode incluir:

- **arena:** Diretório em que o código será executado;
- **bin:** Contém o executável do método a ser ajustado;
- **instances:** Contém as instâncias de treinamento;
- **source:** Contém o código fonte do método a ser ajustado, não é utilizado pelo *irace*.

Definição de parâmetros

scenario.txt

parameterFile: Indica qual é o arquivo com a descrição dos parâmetros;

execDir: Indica qual é o diretório com o algoritmo a ser executado;

trainInstancesDir: Indica qual é o diretório com as instâncias de treinamento;

maxExperiments: Indica o número máximo de execuções do `targetRunner`;

digits: Indica o número de casas decimais para parâmetros reais;

parallel: Indica quantos `targetRunner` serão invocados em paralelo.

Definição de parâmetros

parameters.txt

Os parâmetros podem ser dos tipos categórico (`c`), inteiro (`i`), ordinal (`o`) ou real (`r`).

É necessário tomar cuidado especialmente com parâmetros cujos valores são especificados em intervalos, que podem tornar a calibragem excessivamente lenta.

Caso haja alguma condição lógica para o parâmetro, esta deve ser especificada após um *pipe* (| condicao).

Definição de parâmetros

```
# name      switch            type values    [| conditions]
algorithm   "--"              c    (as, mmas, eas, ras, acs)
ls          "--localsearch "  c    (0, 1, 2, 3)
alpha       "--alpha "        r    (0.01, 5.00)
beta        "--beta "         r    (0.01, 10.00)
rho         "--rho  "         r    (0.00, 1.00)
ants        "--ants "         i    (5, 100)
nnls        "--nnls "         i    (5, 50)     | ls %
q0          "--q0 "           r    (0.0, 1.0)  | algorithm == "acs"
dlb         "--dlb "          c    (0, 1)      | ls %
rank        "--rasranks "     i    (1, 100)    | algorithm == "ras"
eants       "--elitistants "  i    (1, 750)    | algorithm == "eas"
```

Fig. 3. Parameter file (`parameters.txt`) for tuning ACOTSP. The first column is the name of the parameter; the second column is a label, typically the command-line switch that controls this parameter, which irace will concatenate to the parameter value when invoking the target algorithm; the third column gives the parameter type (either *integer*, *real*, *ordinal* or *categorical*); the fourth column gives the range (in case of numerical parameters) or domain (in case of categorical and ordinal ones); and the (optional) fifth column gives the condition that enables this parameter.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

forbidden.txt

Utiliza operadores lógicos `==`, `!=`, `>=`; `<=`, `<`, `>`, `&`, `|`, `!`, `%in%` para relacionar parâmetros.

Combinações de parâmetros que satisfaçam as expressões lógicas deste arquivo **não serão** avaliadas pelo *irace*.

Por exemplo:

- `(ls == 1 & rho == 0.00)`
- `(alpha < 0.10 & beta > 9.00)`

Definição de parâmetros

Código fonte

O código a ser calibrado deve retornar apenas um valor (inteiro ou real) pela saída padrão indicando o valor da solução obtida, que será *minimizado*.

Adicionalmente, o código a ser calibrado deve ter a capacidade de receber os parâmetros por linha de comando.

```
./run instances/instancia1.txt --as --localsearch 2 --alpha
0.01 --beta 5.00 --rho 0.05 --ants 50 --nnls 27 --dlb 0
```

É uma boa prática definir valores padrão para cada parâmetro dentro do código e depois substituí-los pelos valores informados.

Definição de parâmetros

```
int main(int argc, char* argv[]) {
  //...
  vector<string> arguments(argv + 1, argv + argc);

  //...
  for(int i=1; i<arguments.size(); i+=2)
  {
      if(arguments[i]== "--p")
        p = n*stoi(arguments[i+1], &sz);
      else if(arguments[i]== "--pe")
          pe = stod(arguments[i+1], &sz);
        else if(arguments[i]== "--pm")
            pm = stod(arguments[i+1], &sz);
          else if(arguments[i]== "--rhoe")
              rhoe = stod(arguments[i+1], &sz);
            else if(arguments[i]== "--K")
                K = stoi(arguments[i+1], &sz);
                 else if(arguments[i]== "--X_INTVL")
                X_INTVL = stoi(arguments[i+1], &sz);
              else if(arguments[i]== "--MAX_GENS")
                  MAX_GENS = stoi(arguments[i+1], &sz);
                else if(arguments[i]== "--X_NUMBER")
                    X_NUMBER = stoi(arguments[i+1], &sz);
  }
```

Definição de parâmetros

Saídas parciais

A cada iteração, o *irace* fornece uma saída parcial indicando o status da corrida.

As informações são referentes a quantidade de testes realizados, quantidade de testes restantes e número de configurações consideradas.

Por fim, é apresentada uma tabela com o progresso da corrida, indicando o descarte de configurações, entre outros.

Definição de parâmetros

```
#-------------------------------------------------------------------------------
# irace: An implementation in R of (Elitist) Iterated Racing
# Version: 3.5.ec6f3e5
# Copyright (C) 2010-2020
# Manuel Lopez-Ibanez     <manuel.lopez-ibanez@manchester.ac.uk>
# Jeremie Dubois-Lacoste
# Leslie Perez Caceres    <leslie.perez.caceres@ulb.ac.be>
#
# This is free software, and you are welcome to redistribute it under certain
# conditions.  See the GNU General Public License for details. There is NO
# WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
#
# irace builds upon previous code from the race package:
#     race: Racing methods for the selection of the best
#     Copyright (C) 2003 Mauro Birattari
#-------------------------------------------------------------------------------
# installed at: /home/manu/R/x86_64-pc-linux-gnu-library/3.4/irace
# called with: --parallel 2
== irace == WARNING: A default scenario file'./scenario.txt'has been found and will be read
```

Saída parcial do *irace* 1/5.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

```
# Read 1 configuration(s) from file '/home/manu/work/irace/git/examples/vignette-example/default.txt'
# 2020-05-02 13:01:18 BST: Initialization
# Elitist race
# Elitist new instances: 1
# Elitist limit: 2
# nbIterations: 5
# minNbSurvival: 5
# nbParameters: 11
# seed: 432450872
# confidence level: 0.95
# budget: 1000
# mu: 5
# deterministic: FALSE
```

Saída parcial do *irace* 2/5.
fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

```
# 2020-05-02 13:01:18 BST: Iteration 1 of 5
# experimentsUsedSoFar: 0
# remainingBudget: 1000
# currentBudget: 200
# nbConfigurations: 33
# Markers:
     x No test is performed.
     c Configurations are discarded only due to capping.
     - The test is performed and some configurations are discarded.
     = The test is performed but no configuration is discarded.
     ! The test is performed and configurations could be discarded but elite configurations are preserved.
     . All alive configurations are elite and nothing is discarded
```

Saída parcial do *irace* 3/5.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

```
+-+-----------+-----------+-----------+---------------+-----------+--------+-----+----+------+
| |   Instance|      Alive|       Best|      Mean best| Exp so far|  W time|  rho|KenW|  Qvar|
+-+-----------+-----------+-----------+---------------+-----------+--------+-----+----+------+
|x|          1|         33|         17|    23317695.00|         33|00:02:52|   NA|  NA|    NA|
|x|          2|         33|         17|    23371506.00|         66|00:02:52|+0.93|0.97|0.0087|
|x|          3|         33|         17|    23199885.00|         99|00:02:53|+0.94|0.96|0.0078|
|x|          4|         33|         17|    23170261.00|        132|00:02:53|+0.92|0.94|0.0086|
|-|          5|          2|         17|    23150499.80|        165|00:02:52|+0.20|0.36|0.4000|
+-+-----------+-----------+-----------+---------------+-----------+--------+-----+----+------+
Best-so-far configuration:          17    mean value:     23150499.80
Description of the best-so-far configuration:
   .ID. algorithm localsearch  alpha   beta    rho ants nnls q0 dlb rasrank elitistants .PARENT.
17   17      mmas           3 4.0354 3.5637 0.2046   31   23 NA   0      NA           NA       NA
```

Saída parcial do *irace* 4/5.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

```
# 2020-05-02 13:15:44 BST: Elite configurations (first number is the configuration ID;
   algorithm localsearch  alpha   beta    rho ants nnls q0 dlb rasrank elitistants
17      mmas           3 4.0354 3.5637 0.2046   31   23 NA   0      NA           NA
15       eas           3 2.1493 7.5612 0.5756   41   10 NA   1      NA          301
# 2020-05-02 13:15:44 BST: Iteration 2 of 5
# experimentsUsedSoFar: 165
# remainingBudget: 835
```

Figure 6: Sample text output of **irace**.

Saída parcial do *irace* 5/5.
fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

Saída final

Ao final da execução o *irace* apresenta um conjunto de configurações não dominantes entre si.

Isto significa que, apesar de as configurações serem diferentes, os resultados gerados não possuem diferença estatística relevante.

Podemos escolher a que for mais adequada, e.g., em relação ao tempo de execução.

Outro exemplo é a calibração do número máximo de execuções: o menor valor para o parâmetro dentre as configurações não dominantes entre si é um limitante inferior.

Definição de parâmetros

```
# Best configurations (first number is the configuration ID)
    algorithm  ls alpha beta  rho ants nnls   q0 dlb rank eants
530       acs   3  1.93 8.27 0.36   21   10 0.78   1   NA    NA
635       acs   3  2.52 9.69 0.29   42   11 0.83   1   NA    NA
552       acs   3  1.81 7.86 0.25   20   16 0.72   1   NA    NA
741       acs   3  1.78 5.74 0.23   31   13 0.64   1   NA    NA
700       acs   3  1.85 5.97 0.23   34   12 0.68   1   NA    NA
```

Saída final do *irace*.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

```
# Best configurations as commandlines (first number is the configuration ID)
530  --acs --localsearch 3 --alpha 1.93 --beta 8.27 --rho  0.36 \
     --ants 21 --nnls 10 --q0 0.78 --dlb 1
635  --acs --localsearch 3 --alpha 2.52 --beta 9.69 --rho  0.29 \
     --ants 42 --nnls 11 --q0 0.83 --dlb 1
552  --acs --localsearch 3 --alpha 1.81 --beta 7.86 --rho  0.25 \
     --ants 20 --nnls 16 --q0 0.72 --dlb 1
741  --acs --localsearch 3 --alpha 1.78 --beta 5.74 --rho  0.23 \
     --ants 31 --nnls 13 --q0 0.64 --dlb 1
700  --acs --localsearch 3 --alpha 1.85 --beta 5.97 --rho  0.23 \
     --ants 34 --nnls 12 --q0 0.68 --dlb 1
```

Saída final do irace.

fonte: `https://bityli.com/WVzhn`

Definição de parâmetros

Observação

Quando o valor selecionado for igual ao maior ou menor valor disponibilizado para um parâmetro, pode ser um indicativo de que a faixa de valores fornecidos é inadequada.

Nestes casos, a calibração deve ser refeita, com intervalos maiores.

Definição de parâmetros

irace

No material de apoio há a pasta *Irace* com material relativo a este pacote.

Há uma subpasta *tuning*, que possui um *setup* do *irace* pronto para calibrar os parâmetros de um código BRKGA de exemplo.

Definição de parâmetros

irace

A execução do *irace* pode levar semanas, e eventualmente a execução pode ser interrompida, e.g., quedas de energia.

Nestes casos é possível resumir a execução a partir do arquivo `irace-backup.Rdata`.

```
> irace --recovery-file ./irace-backup.Rdata
```

Observações sobre outros parâmetros

Quantidade de repetições

A quantidade de testes necessários por instância também é um parâmetro de um experimento, e varia de acordo com os objetivos do mesmo.

Um erro que não se deve cometer é realizar apenas um teste com cada instância, pois pode-se chegar a conclusões erradas e até mesmo tornar o experimento irreprodutível.

Métodos que possuem componentes com aleatoriedade podem dar resultados diferentes ao fazer vários testes com uma única instância.

Para reduzir a variância dos resultados, o código pode ser testado sobre cada instância com sementes de inicialização diferentes.

Observações sobre outros parâmetros

Critério de parada

Não se deve utilizar tempo como critério de parada único de um método ou experimento.

Utilizar o tempo como critério de parada e relatar que o tempo de execução foi 60 minutos, por exemplo, pode ser insignificante.

Estas comparações não são reproduzíveis, pois se os testes forem executados em um computador dez vezes mais rápido, a solução poderá ser diferente e algumas conclusões podem mudar drasticamente.

Observações sobre outros parâmetros

Critério de parada

Para fazer comparações justas, pode-se utilizar como critério de parada alguma medida combinatorial, como o número de vizinhos encontrados ou a quantidade de passos de *branching*.

O algoritmo fica bem definido e o tempo de execução e a qualidade da solução podem ser medidas com um cálculo combinatorial, o qual poderá ser reproduzido futuramente.

Observações sobre outros parâmetros

Critério de parada

Não se deve usar também o valor de uma solução ótima como critério de parada.

Este erro é encontrado em alguns trabalhos com metaheurísticas em que os algoritmos não têm nenhum critério ou método para verificar a otimalidade, e procuram uma boa solução até algum critério de parada ser atingido.

Se esta abordagem for utilizada, só poderão ser testadas instâncias com valores ótimos já conhecidos, e para as instâncias em que estes valores são desconhecidos este critério não poderia ser utilizado.

Para as instâncias em que os valores são conhecidos esse método não refletirá o desempenho do algoritmo de fato.

Leitura recomendada

- Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. *The irace package: Iterated Racing for Automatic Algorithm Configuration*. Operations Research Perspectives, 2016.
- Manuel López-Ibáñez, Leslie Pérez Cáceres, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. *The irace package: User Guide*. Version 3.5, 2021.

Dúvidas?