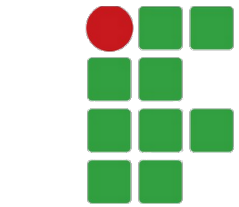
Metaheurísticas

Configuração Automatizada de Metaheurísticas

Baseado no livro *Handbook of Metaheuristics* [Gendreau and Potvin, 2019]

Felipe Augusto Lima Reis

felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

Introdução

Introdução

- A construção de metaheurísticas pode ser demorada e difícil de ser executada;
- Tal condição ocorre devido a uma série de fatores como:
  - Complexidade dos problemas a serem solucionados;
  - Grande número de graus de liberdade[1], devido ao número extenso de parâmetros;
  - Dificuldade de análise do algoritmo devido vieses heurísticos e estocasticidade.

[1] Graus de liberdade correspondem ao número máximo de valores logicamente independentes, ou seja, que têm liberdade para variar, na amostra de dados [Investopedia, 2021].

Introdução

- Tradicionalmente o desenvolvimento de metaheurísticas é feito por meio da construção manual de algoritmos
  - A abordagem é baseada na experiência prévia dos projetistas e requer trabalho extenso de configuração dos parâmetros;
- Nos últimos anos, alguns trabalhos foram propostos para automatização da construção e configuração de algoritmos
  - Alguns resultados mostraram-se muito bem-sucedidos na identificação de algoritmos de alto desempenho e configurações de parâmetros;
  - Os configuradores foram capazes de pesquisar com eficácia espaços de parâmetros grandes e diversos.

Introdução

- Além da configuração automática, os processos automatizados buscam resolver as seguintes desvantagens da configuração manual:
  1. Limitação do número de alternativas de design que são exploradas;
  2. Possibilidade de irreprodutibilidade do processo de desenvolvimento do algoritmo;
  3. Dificuldade em revelar o esforço real que foi dedicado ao desenvolvimento;
  4. Perda de informações sobre decisões de *designs* que foram exploradas e descartadas, pois resultaram em desempenho aparentemente pior.

Introdução

- Uma vantagem conceitual de processos automáticos é a separação clara entre algoritmos e a configuração dos mesmos
  - Tal separação possibilita a criação de metaheurísticas gerais, evitando construção de métodos de propósito específico;
  - Instâncias de treinamento, tipos e domínios, e critérios de encerramento podem ser analisados separadamente;
  - Melhorias futuras dos configuradores, podem melhorar resultados prévios sem alteração da metaheurística original;

Configuração Automática de Algoritmos

Iterated Local Search (ILS)

- Para ilustrar o processo de projeto algoritmos, vamos considerar a metaheuristica ILS.
  - O ILS é baseado em uma arquitetura modular, que possibilita o refinamento do algoritmo para diferentes problemas [Gendreau and Potvin, 2019].

Algorithm Iterated local search

1: $s_0 = \text{GenerateInitialSolution}$
2: $s^* = \text{LocalSearch}(s_0)$
3: **repeat**
4: $\quad s' = \text{Perturbation}(s^*, history)$
5: $\quad s^{*\prime} = \text{LocalSearch}(s')$
6: $\quad s^* = \text{AcceptanceCriterion}(s^*, s^{*\prime}, history)$
7: **until** termination condition met

Fonte: [Gendreau and Potvin, 2019]

Iterated Local Search (ILS)

- Segundo [Souza, 2011], para implementação do ILS são necessários 4 componentes (procedimentos):
  1. Gera Solução Inicial
     - Gera uma solução inicial s_0 para o problema;
  2. Busca Local
     - Retorna uma solução possivelmente melhorada s;
  3. Perturbação
     - Modifica a solução corrente s levando a uma solução intermediária s';
  4. Critério de Aceitação
     - Decide em qual solução a próxima perturbação será aplicada.

Iterated Local Search (ILS)

- O desempenho do ILS em relação à qualidade da solução final e a velocidade de convergência depende do método de busca local escolhido
  - Métodos simples podem ser utilizados, porém outras técnicas mais sofisticadas tendem a obter melhores resultados;
- A variedade de soluções é dependente da intensidade de perturbação do algoritmo
  - A perturbação deve ser forte o suficiente para escapar do ótimo local e possibilitar a exploração de diferentes regiões;
  - No entanto, deve guardar características do ótimo local corrente [Souza, 2011];

Iterated Local Search (ILS)

- O critério de aceitação decide qual solução será utilizada para os passos seguintes
  - Esse critério também auxilia na definição da perturbação a ser aplicada na solução;
  - Critérios de aceitação podem permitir soluções melhores que a atual ou ligeiramente piores [Souza, 2011].

Parametrização ILS

- Parametrização da Perturbação:
  - É utilizado um parâmetro k, em um intervalo $[k_{min}, k_{max}]$, que indica a força da perturbação [Gendreau and Potvin, 2019].

- Parametrização do Critério de Aceitação:
  - Podem ser utilizados critérios probabilísticos, que sempre aceitam soluções candidatas iguais ou melhores que a atual, porém que podem aceitar soluções ligeiramente piores;
  - A probabilidade de aceitar novas soluções pode ser dada por:

$$P(t, R, S) = e^{\left(\frac{Qualidade(R) - Qualidade(S)}{t}\right)}$$

onde,
t: temperatura, R: solução de qualidade inferior, S: solução atual (superior), e e: número de Euler.

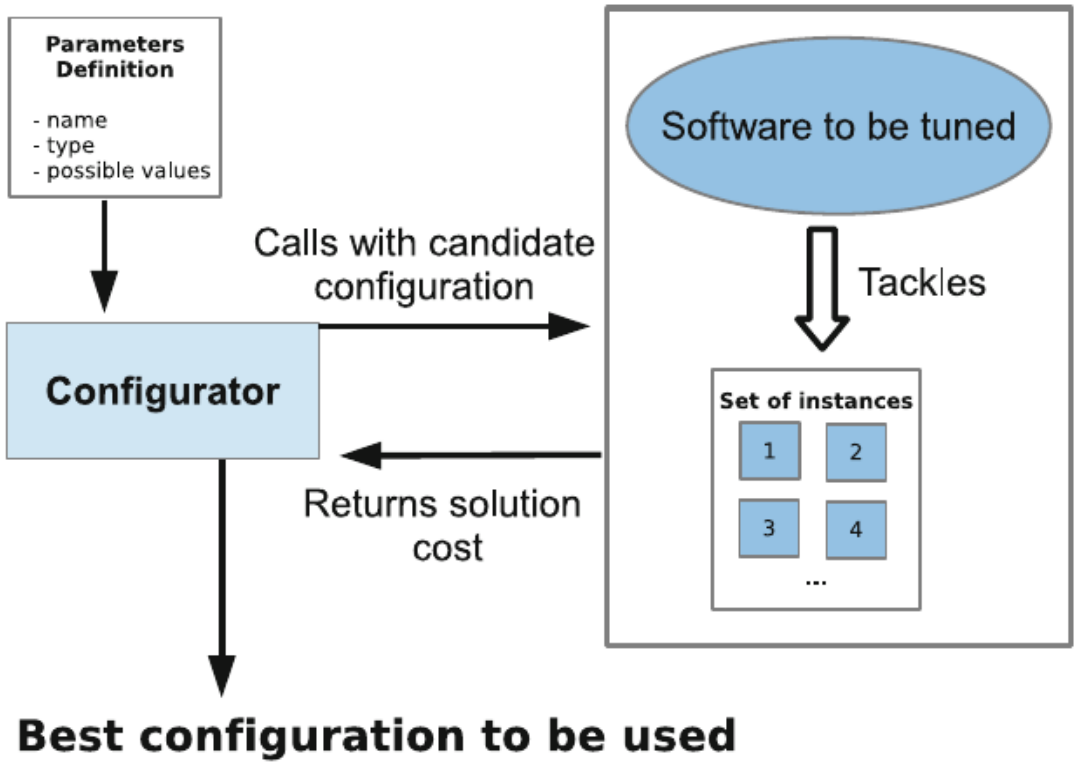
Parametrização - Outros Algoritmos

- Algoritmos populacionais possuem um conjunto distintos de parâmetros a serem configurados, como:
  - Tamanho da população;
  - Taxas de crossover e mutação (se houverem);
  - Número de soluções que permanecerão para gerações seguintes, em algoritmos geracionais;
  - Parâmetros como velocidade e posição, em algoritmos de estado estacionário baseados em enxames.
- Algoritmos construtivos também possuem alguns parâmetros a serem configurados, como:
  - Tamanho de estrutura de dados (ex. Lista Tabu);
  - Tamanho do passo (ex. Hill Climbing).

Processo de Configuração

Processo de Configuração

- O processo de uso de configuradores está representado na figura abaixo:



Fonte: [Gendreau and Potvin, 2019]

Processo de Configuração

- Um configurador recebe como entrada os parâmetros a serem configurados no algoritmo;
- Uma entrada pode incluir informações como:
  - Nomes dos parâmetros, tipos e seus domínios;
  - Métricas de avaliação do desempenho;
  - Dependências entre parâmetros;
  - Combinações proibidas de valores;
  - Quaisquer outras informações relevantes.

Processo de Configuração

- As configurações candidatas são avaliadas em um conjunto de instâncias de treinamento e as avaliações são retornadas ao configurador;
- O processo de geração e avaliação de configurações candidatas é iterado até que um critério de parada seja atingido (ex.: número de iterações).
- A fim de evitar que a configuração caia em "armadilhas"[2] ou faça uso de tentativa-e-erro, processos de treinamento podem utilizar técnicas estatísticas, como testes de hipótese para avaliar a significância estatística.

[2] Tradução direta de *pitfalls*. Significa que o método obtém, aparentemente, um bom resultado, mas essa configuração não evolui adequadamente

Técnicas de Configuração

- São comuns os seguintes tipos de configuração:
  - Otimização Contínua:
    - Utilizado para configuração de parâmetros numéricos;
    - Apesar de funcionar com parâmetros reais, possui melhores resultados com parâmetros inteiros (principalmente quando a faixa de valores é larga);
    - São exemplos as técnicas MADS, CMAES e BOBYQA.
  - Busca Heurística:
    - Utilizada para configuração de parâmetros numéricos e não numéricos;
    - Capaz de configurar parâmetros em algoritmos evolucionários, como os algoritmos genéticos;
    - Destacam-se as técnicas OPAL e ParamILS.

Técnicas de Configuração

- São comuns os seguintes tipos de configuração: *[cont.]*
  - Modelos Substitutivos[3]:
    - Fazem previsões do desempenho de configurações a partir de execuções observadas anteriormente de outras instâncias de problema;
    - Podem ser também denominadas de Otimização Bayesiana;
    - Novas execuções podem ser feitas para melhoria do processo de predição;
    - Destacam-se as técnicas SPOT e SMAC.

[3] Tradução direta de *Surrogate-Model Based Configurators*.

Técnicas de Configuração

- São comuns os seguintes tipos de configuração: *[cont.]*
  - Racing Approaches[4]:
    - Técnicas que selecionam uma melhor configuração entre um conjunto de configurações candidatas;
    - Configurações candidatas iniciais para uma corrida podem ser selecionadas por técnicas experimentais, de forma aleatória ou com base no conhecimento específico do problema;
    - Destacam-se as técnicas F-race e *irace*.

[4] Abordagens baseadas em "corrida" (racing).

ParamILS

ParamILS

- ParamILS é uma framework de configuração automática de algoritmos proposto por Hutter et al., em 2009 [Hutter et al., 2009].
- O ParamILS é um tipo de configurador heurístico para afinação (tuning) de parâmetros e configuração de algoritmos [Hutter and Fawcett, 2021];
- Foi desenvolvido para configuração de algoritmos que buscam solucionar o problema SAT [Hutter et al., 2009].

O ParamILS está disponível online em: `http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/`.

ParamILS

- O ParamILS executa o algoritmo ILS no espaço de parâmetros;
- O algoritmo trata o problema de configuração como uma tarefa de variáveis categóricas
  - Os parâmetros numéricos precisam ser discretizados para serem utilizados pelo método;
- Para a inicialização, é necessário que a entrada tenha uma configuração padrão
  - O algoritmo gera, então, uma pequena quantidade de r configurações aleatórias;
  - A configuração de início é definida como a melhor dentre as $r + 1$ configurações iniciais.

ParamILS

INSTITUTO FEDERAL
Minas Gerais

- O algoritmo do ParamILS pode ser visto abaixo.

Algorithm Framework 1: ParamILS($\theta_0, r, p_{restart}, s$)
Outline of iterated local search in parameter configuration space; the specific variants of ParamILS we study, **BasicILS(N)** and **FocusedILS**, are derived from this framework by instantiating procedure *better* (which compares $\theta, \theta' \in \Theta$). *BasicILS(N)* uses $better_N$ (see Procedure 2), while FocusedILS uses $better_{Foc}$ (see Procedure 3). The neighbourhood $Nbh(\theta)$ of a configuration θ is the set of all configurations that differ from θ in one parameter, excluding configurations differing in a conditional parameter that is not relevant in θ.

Input : Initial configuration $\theta_0 \in \Theta$, algorithm parameters r, $p_{restart}$, and s.
Output : Best parameter configuration θ found.

```
1  for i = 1, ..., r do
2      θ ← random θ ∈ Θ;
3      if better(θ, θ_0) then θ_0 ← θ;
4  θ_ils ← IterativeFirstImprovement (θ_0);
5  while not TerminationCriterion() do
6      θ ← θ_ils;
       // ===== Perturbation
7      for i = 1, ..., s do θ ← random θ' ∈ Nbh(θ);
       // ===== Basic local search
8      θ ← IterativeFirstImprovement (θ);
       // ===== AcceptanceCriterion
9      if better(θ, θ_ils) then θ_ils ← θ;
10     with probability p_restart do θ_ils ← random θ ∈ Θ;
11 return overall best θ_inc found;
12 Procedure IterativeFirstImprovement (θ)
13 repeat
14     θ' ← θ;
15     foreach θ'' ∈ Nbh(θ') in randomized order do
16         if better(θ'', θ') then θ ← θ''; break;
17 until θ' = θ;
18 return θ;
```

Fonte: [Hutter et al., 2009]

ParamILS

- A busca local no ParamILS usa uma vizinhança de troca única, onde os pares de valores de parâmetros são examinados em ordem aleatória
  - Em cada etapa, um movimento seguinte é examinado e a nova configuração é adotada, se o resultado for melhor que o atual;
- Após cada melhoria, a vizinhança é reorganizada de forma aleatória
  - Quando o algoritmo atinge um ótimo local, a configuração local $\theta^{*\prime}$ é comparada ao ótimo global algoritmo θ^{*};
  - A melhor das duas configurações é mantida.

ParamILS

- A perturbação do ParamILS modifica k parâmetros escolhidos aleatoriamente para obter a configuração $\theta^{*\prime}$;
  - Uma nova busca é gerada a partir dessa configuração;
- Uma solução diferente é usar uma solução aleatória ao invés de uma perturbação
  - Essa solução é somente executada em alguns momentos, a partir de uma probabilidade pr (por padrão, $pr = 0.01$).

ParamILS

- Para comparação de configurações, o ParamILS possui duas abordagens diferentes:
  - BasicILS:
    - Todas as configurações são avaliadas no mesmo número máximo de configurações;
    - A configuração que obtém a melhor estimativa de custo é selecionada;
    - Possui potencial desvantagem de exigir uma escolha a priori, podendo desperdiçar configurações abaixo do ideal.
  - FocusedILS:
    - Recomendada pelos autores do método;
    - O número de instâncias em que duas configurações são comparadas é aumentado iterativamente, até que uma configuração domine a outra.

ParamILS

- Para melhoria do desempenho do ParamILS, podem ser utilizados métodos de poda
  - Uma técnica comum é implementar uma técnica de poda denominada *Adaptive Capping*;
  - A poda é usada para encerrar antecipadamente a avaliação de configurações de desempenho potencialmente insatisfatórias.

SMAC

SMAC

- O Sequential Model-based Algorithm Configuration (SMAC) é um configurador que implementa uma pesquisa baseada em modelo substituto do espaço de parâmetros;
- O algoritmo foi proposto por Hutter et al., em 2011 [Hutter et al., 2011].
- Ao contrário do ParamILS, o SMAC lida com parâmetros numéricos e categóricos nativamente, ou seja, sem a necessidade de discretização;
- Segundo [Gendreau and Potvin, 2019], o SMAC é uma das técnicas de configuração automática de algoritmos de melhor desempenho e mais amplamente utilizada.

O algoritmo está disponível online, nas versões Python e Java, em `https://www.ml4aad.org/automated-algorithm-design/algorithm-configuration/smac/`.

SMAC

- O SMAC usa modelagem substitutiva para selecionar um conjunto de configurações usando previsões de desempenho
  - As melhores configurações de acordo com o modelo são selecionadas para avaliação real;
- O modelo é construído usando dados de desempenho gerados durante o processo de pesquisa;
  - A previsão de desempenho é usada para calcular a melhoria esperada;
- O SMAC começa a partir de alguma configuração inicial, normalmente, a configuração padrão do algoritmo
  - Caso inexistam configurações iniciais, o algoritmo inicia uma configuração aleatória.

SMAC

- O modelo substitutivo (*surrogate*) do SMAC usa florestas de decisões aleatórias (conjunto de árvores de decisão);
- O procedimento é executado utilizando as seguintes etapas:
  - Um modelo de *random forest* é aprendido;
  - Um conjunto de configurações candidatas é gerado, a partir de uma lista de configurações de elite;
  - Cada configuração serve como ponto de partida para uma busca local;
  - Cada configuração é avaliada de acordo com o critério de melhoria esperado;
  - O processo resulta em n_{1s} configurações localmente ótimas;

SMAC

- *[cont.]*
  - Um conjunto configurações n_r é criado aleatoriamente, cada uma sendo avaliada de acordo com sua melhoria esperada;
  - Em seguida, o SMAC ordena configurações $n_{1s} + n_r$ de acordo com a melhoria esperada;
  - Estas são executadas na ordem fornecida nas instâncias do problema;
  - O processo de avaliação das configurações utiliza o critério de dominância, assim como o FocusedILS.
- A floresta de decisão pode ser retreinada usando os dados de execução para melhorar as predições.

Referências I

Gendreau, M. and Potvin, J.-Y. (2019).
Handbook of Metaheuristics (Third Edition).
Springer, Cham, 3 edition.

Hutter, F. and Fawcett, C. (2021).
Degrees of freedom.
[Online]; acessado em 20 de Julho de 2021. Disponível em:
http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential model-based optimization for general algorithm configuration.
In Coello, C. A. C., editor, Learning and Intelligent Optimization, pages 507–523, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. (2009).
Paramils: An automatic algorithm configuration framework.
J. Artif. Int. Res., 36(1):267â306.

Investopedia (2021).
Degrees of freedom.
[Online]; acessado em 20 de Julho de 2021. Disponível em:
https://www.investopedia.com/terms/d/degrees-of-freedom.asp.

Referências II

Souza, M. J. F. (2011).
Inteligência computacional para otimização.
[Online]; acessado em 12 de Maio de 2021. Disponível em: http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf.