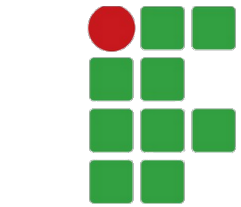
Metaheurísticas

GRASP

Felipe Augusto Lima Reis

felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

GRASP
Greedy Randomized Adaptive Search Procedures

GRASP

- O GRASP (*Greedy Randomized Adaptive Search Procedures*) é uma metaheurística utilizada para solução de problemas de otimização combinatória [Festa and Resende, 2002];

- Foi desenvolvido em 1989 por Feo e Resende [Souza, 2011] [Luzia and Rodrigues, 2009]
  - O algoritmo foi aplicado a diversos problemas de otimização combinatória, como roteamento e *scheduling* (definição de sequência de atividades) [Festa and Resende, 2002].

GRASP

- O GRASP é um processo iterativo, no qual cada iteração é constituída de 2 fases: [Festa and Resende, 2002]
  1. Construção: produz-se uma solução factível, de forma iterativa;
  2. Busca Local: busca-se um ótimo local na vizinhança das soluções construídas;

GRASP

- Após as duas fases, a melhor solução global encontrada é mantida e retornada como resultado;
- O algoritmo busca associar bons aspectos dos algoritmos gulosos, com aqueles existentes no procedimentos aleatórios de construção de soluções [Souza, 2011];
- O GRASP pode ser considerado uma heurística adaptativa, uma vez que os elementos são atualizados a cada iteração na fase de construção [Luzia and Rodrigues, 2009].

Fase 1 - Construção

- Produz-se uma solução factível, de forma iterativa
  - A cada iteração, é escolhido o próximo elemento a ser adicionado à solução;
  - Uma função avalia a lista de candidatos e verifica quais os benefícios gerados por cada um;
  - Os candidatos são ordenados em uma lista dos melhores candidatos, denominada *Restricted Candidate List*[1] (RCL);
  - O algoritmo escolhe aleatoriamente um elemento dessa lista [Festa and Resende, 2002] [Luzia and Rodrigues, 2009];

[1] [Souza, 2011] traduz o nome da lista para Lista Restrita de Candidatos (LRC).

Fase 1 - Construção

- Devido à escolha de elementos aleatória dos melhores candidatos, o GRASP possui uma característica probabilística
  - Essa abordagem permite que diferentes soluções sejam geradas [Luzia and Rodrigues, 2009];
  - Tal procedimento evita que o algoritmo fique preso em soluções ótimas locais.

Fase 2 - Busca Local

- A Fase de Construção, apesar de retornar boas soluções, não garante que a solução gerada seja um ótimo local [Festa and Resende, 2002];
- Com isso, uma segunda fase, de Busca Local, é utilizada para refinar a solução previamente retornada
  - Um método realiza busca do máximo local na vizinhança dos resultados obtidos na fase anterior;
- O refinamento adequado da solução é baseado na escolha de um bom algoritmo de busca local para ser utilizado nesta fase.

GRASP - Versão Paralela

- A versão paralela do GRASP pode ser considerada simples de ser implementada
  - Nessa versão do algoritmo, cada processador pode ser iniciado com sua própria cópia do procedimento;
  - Cada processador também deve conter os dados e uma sequência numérica aleatória independente;
  - As iterações são executadas em paralelo e o melhor resultado local de cada versão é retornado;
  - As soluções são comparadas e a melhor solução global é definida pelo algoritmo [Festa and Resende, 2002] [Luzia and Rodrigues, 2009].

Algoritmo

GRASP

- O pseudo-algoritmo do GRASP pode ser visto abaixo:

```
01: C ← {C1 , ..., Cn } // componentes
02: p ← porcentagem dos componentes serem inclusos em cada iteração
03: m ← tempo para se realizar Hill Climbing
04: Melhor ← 2
05: repita
06:     S ← {} // solução candidata
07:     repita
08:         C' ← elementos em C − S que podem ser postos em S sem torná-la infactível
09:         se C' for vazio, então
10:             S ← {} // tente novamente
11:         senão
12:             C'' ← componentes em C' com o maior valor (ou menor custo) para p%
13:             S ← S ∪ {componente escolhido uniforme e aleatoriamente em C''}
14:     até que S seja uma solução completa
15:     faça m vezes
16:         R ← Variar(Copiar(S))
17:         se Qualidade(R) > Qualidade(S), então
18:             S←R
19:     se Melhor = 2 ou Qualidade(S) > Qualidade(Melhor), então
20:         Melhor ← S
21: até que Melhor seja a solução ideal ou o tempo tenha se esgotado
22: devolva Melhor
```

GRASP

Fonte: [Luzia and Rodrigues, 2009], baseado em [Luke, 2013]

Nesta versão do GRASP, o Hill-Climbing foi o algoritmo escolhido para a fase de busca local.

Vantagens e Desvantagens

Vantagens e Desvantagens

- Vantagens:
  - Implementação simples, facilmente paralelizável;
  - Flexibilidade: pode ser adaptado a diversas situações [Festa and Resende, 2002].
- Desvantagens:
  - A análise formal da qualidade das soluções geradas pelo GRASP é complexa;
  - Solução dependente da lista de candidatos (os resultados serão ruins se a lista for inadequada) [Festa and Resende, 2002].

Referências I

Festa, P. and Resende, M. G. (2002).
Grasp: An Annotated Bibliography, pages 325–367.
Springer US, Boston, MA.

Luke, S. (2013).
Essentials of Metaheuristics (Second Edition).
lulu.com, 2 edition.
[Online]; Disponível em: https://cs.gmu.edu/~sean/book/metaheuristics/.

Luzia, L. F. and Rodrigues, M. C. (2009).
Estudo sobre as metaheurísticas.
[Online]; acessado em 22 de Setembro de 2020. Disponível em:
https://www.ime.usp.br/~gold/cursos/2009/mac5758/LeandroMauricioHeuristica.pdf.

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.