Otimização Combinatória: Trabalho Final (2023/2)

Henrique Becker

24 de dezembro de 2023

1 Instruções gerais

1. O trabalho final consiste em 4 entregáveis:
   (a) formulação inteira (0.5pt),
   (b) meta-heurística (0.8pt),
   (c) relatório/experimentos (0.6pt),
   (d) e apresentação (0.6pt).
2. O trabalho deve ser realizado em trios.
3. A formação dos trios fica a critério e responsabilidade dos alunos.
4. A escolha do problema e meta-heurística é responsabilidade dos trios.
   (a) Não poderá ser escolhido um mesmo par de problema e meta-heurística já escolhido por outro trio.
   (b) Uma lista dos pares já escolhidos está disponível em: `https://link.inf.ufrgs.br/45-QYHq`
   (c) Assim que um trio decidir por um par problema/meta-heurística, este deve ser imediatamente informado por e-mail para o professor (hbecker@inf.ufrgs.br) com assunto '[OC TF] Escolha' e o número do cartão da UFRGS dos três envolvidos.
   (d) Este e-mail será respondido pelo professor confirmando ou não a possibilidade (após atualizar a planilha).

5. A mesma exata nota será compartilhada pelos 3 membros do trio.

   (a) É responsabilidade de cada aluno de formar um trio com outros alunos que irão contribuir com o trabalho.

6. O entregável **formulação inteira** consiste em:

   (a) Um código-fonte em qualquer linguagem (com instruções claras para compilação/instalação/execução em Linux) que toma um arquivo no formato de entrada dado, a semente de aleatoriedade, e o limite de tempo (1h por default), e utiliza uma formulação inteira e o solucionador GLPK para resolver o problema.

   (b) Uma execução deve escrever escrever na saída padrão as informações usadas para montar a tabela do relatório (melhor solução encontrada, tempo, etc...).

   (c) Essa formulação deve ser exatamente a mesma utilizada para escrita do entregável **relatório/experimentos**.

7. O entregável **meta-heurística** consiste em:

   (a) Um código-fonte em qualquer linguagem (com instruções claras para compilação/instalação/execução em Linux) que toma um arquivo no formato de entrada dado e outros parâmetros (veja abaixo) e executa a meta-heurística.

   (b) Uma execução deve escrever escrever na saída padrão as informações usadas para montar a tabela do relatório (solução inicial, melhor solução encontrada, tempo, etc...).

   (c) O código deve implementar a meta-heurística escolhida. Cada meta-heurística tem lacunas que variam com o problema e decisões de implementação a critério dos alunos, mas também tem partes que caracterizam ela como aquela meta-heurística (estas devem estar presentes no código).

   (d) Além do nome do arquivo de entrada, o programa deve tomar na linha de comando quaisquer outros parâmetros os quais os valores foram variados nos experimentos (pode ser por meio de argumentos posicionais). Dentre esses argumentos, obrigatoriamente, para todas as meta-heurísticas, devem se encontrar a semente de

aleatoriedade e o valor que controla o critério de parada (número de iterações, ou qualquer outro escolhido).

(e) Esse programa deve ser exatamente o mesmo utilizado para escrita do entregável **relatório/experimentos**.

8. O entregável **relatório/experimentos** consiste em:

   (a) Descrição clara e completa da formulação inteira empregada.

   (b) Descrição clara e completa da meta-heurística desenvolvida:

      i. Escolha de representação do problema.
      ii. Construção da solução inicial.
      iii. Principais estruturas de dados
      iv. Vizinhança e a estratégia de escolha de vizinhos (quando aplicável).
      v. Processo de recombinação e factibilização (quando aplicável).
      vi. Parâmetros do método (valores usados nos experimentos).
      vii. Critério de parada (NÃO pode ser tempo).

   (c) Tabela dos resultados de 10 execuções da meta-heurística sobre cada uma das instâncias (i.e., cada linha representa um par de uma instância e de uma semente de aleatoriedade, a qual é diferente para cada execução, OU uma tabela com cabeçalho + 10 linhas para cada instância) com as seguintes colunas:

      i. Valor da solução inicial da meta-heurística: S_i.
      ii. Valor da melhor solução encontrada pela meta-heurística: S_h.
      iii. Tempo de execução da meta-heurística (segundos): *H T (s)*.
      iv. Valor da solução encontrada pela formulação: S_f.
      v. Caso termine por limite de tempo, o limite superior: U_f.
      vi. Tempo de execução da formulação (segundos): *F T (s)*.

   (d) Análise dos resultados (resultados variam muito com a semente de aleatoriedade? é melhor que a formulação?).

   (e) Conclusões e bibliografia utilizada.

9. O entregável **apresentação** consiste em:

   (a) Uma apresentação para turma de *no máximo* 15 minutos.

(b) O objetivo da apresentação é apresentar a implementação e os resultados; o professor e a turma podem fazer perguntas.

(c) Cada membro do grupo deve participar e comentar o que fez.

2 Meta-heurísticas

Cada trio deve escolher uma das 7 opções de meta-heurística abaixo. Devido a similaridade, algoritmos genéticos e meméticos contam como a mesma opção (e, consequentemente, um grupo não pode pegar genético para um problema e outro grupo pegar memético para o mesmo problema).

- Simulated Annealing
- Late Acceptance Hill Climbing
- GRASP
- Busca Tabu
- VNS
- Busca Local Iterada
- Algoritmos genéticos/meméticos

Lembrando que:

- Critérios como qualidade das soluções encontradas e eficiência das implementações serão considerados na avaliação (ex: quando fizer uma modificação na solução, recalcular a diferença com o vizinho e não a solução inteira novamente).
- Comentários no código ajudam na correção.
- Uma boa vizinhança é aquela que permite alcançar qualquer solução (em múltiplos passos, não um único passo).
- Vizinhos com mesmo valor de solução: utilizar escolhas aleatórias como critério de desempate podem trazer bons resutados.

3 Problemas

Bin Packing	Multidimensional 0-1 Knapsack Problem	Set Packing

1. **Bin Packing**

Instância A instância é composta da capacidade dos cestos, da quantidade de items a serem guardados nos cestos, e os pesos dos ditos items.

Objetivo Determinar uma atribuição de objetos a cestos do tamanho especificado de forma a respeitar a capacidade de cada cesto, e minimizar o número de cestos necessários[Delorme et al. 2014, p. 1–2].

2. **Multidimensional 0-1 Knapsack**

Instância A instância é composta das: capacidades de uma única mochila com múltiplas dimensões, o valor/lucro de cada item, o peso de cada item em cada uma das dimensões do knapsack.

Objetivo Determinar quais items devem ser colocados dentro da mochila de forma a respeitar a capacidade de cada dimensão, e maximizar o valor dos itens dentro da mochila. Observar que há somente um de cada um dos itens descritos, logo não se pode adicionar várias cópias de um mesmo item. Dois itens que não são o mesmo podem ter mesmo peso e valor (mas depende da instância para isso ocorrer).

3. **Set Packing**

Instância A instância é composta de: um valor inteiro N, e M conjuntos de números inteiros entre 1 e N.

Objetivo Determinar a maior seleção de conjuntos possível em que dois conjuntos distintos não compartilhem um mesmo número.

4 Instâncias

Um arquivo *.tar* composto pelas instâncias abaixo listadas de cada problema está disponível nos links a seguir: `www.inf.ufrgs.br/~hbecker/instancias_oc_tf/selected_bpp_instances.tar` (Bin Packing), `www.inf.ufrgs.br/~hbecker/instancias_oc_tf/selected_mdkp_instances.tar` (Multidimen-

sional Knapsack) e `www.inf.ufrgs.br/~hbecker/instancias_oc_tf/selected_spp_instances.tar` (Set Packing).

O formato das instâncias para cada problema é:

1. Bin Packing: O arquivo (com terminação ".txt") possui na primeira linha somente o número de items N. A segunda linha do arquivo possui somente a capacidade C a ser usada para todos os cestos. Todas as linhas posteriores até o final do arquivo possuem cada uma somente o peso de um dos items. O arquivo sempre possui, portanto, N+2 linhas, as quais da terceira (inclusive) a última (inclusive) possuem o peso de um único item cada. Nenhum peso de item é superior a C. Todos valores são inteiros (e o maior número é 61120).

2. Multidimensional 0-1 Knapsack Problem: O arquivo (com terminação ".dat") possui quebras de linhas arbitrárias, somente o índice do valor dentro do arquivo (se ele é o terceiro ou quadragésimo segundo) é relevante. O primeiro valor do arquivo é o número de items existentes (N). O segundo é o número de dimensões que os itens e o knapsack possuem (M). O terceiro é a solução ótima da instância (Z), ou zero se a solução ótima é desconhecida. Os próximos N números são os valores dos itens (os itens dentro do knapsack devem perfazer a maior soma possível desses valores). A partir desse ponto, por M vezes, cada N valores representam os pesos de todos os items em uma das dimensões. Ou seja, o próximo valor representa o peso do primeiro item na primeira dimensão, o depois dele representa o peso do segundo item na primeira dimensão. Após N valores, os próximos N indicam o peso dos itens na segunda dimensão, até os N valores que indicam o peso dos itens na M-ésima dimensão. Os últimos M valores (que vem após essa sequência de $N \times M$ valores) são as capacidades do knapsack nas M dimensões. O arquivo deve conter, portanto, $3+N+N\times M+M$ valores numéricos inteiros separados por brancos (espaço(s) e/ou quebra(s) de linha).[1]

3. Set Packing Problem: O arquivo (com terminação ".msc") segue na primeira linha o padrão "p set N M", onde N e M são, respectivamente, o número de itens distintos e o número de conjuntos. O número de itens distintos é também o maior número que pode aparecer no

[1] Caso ainda esteja com dúvida a descrição original (em inglês) do formato do arquivo se encontra em: `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`. Os parágrafos relevantes são os referentes ao formato do arquivo "mknap1".

restante do arquivo (uma vez que número identificador do item varia de $1..N$). Após a primeira linha existem M linhas cada uma delas começa com "s " (um 's' minúsculo e um espaço) e o resto da da linha até a quebra é composto de números separados por espaços. Cada uma dessas linhas representa um dos conjuntos e pode possuir um número variado e não préviamente especificado de elementos (sempre contêm ao menos um elemento). O objetivo é selecionar a maior quantidade destas linhas (conjuntos) sem selecionar o mesmo número (item) mais de uma vez. Todos números são inteiros.

Tabela 1: Instâncias do Bin Packing a serem consideradas nos testes computacionais.

Nome do arquivo	#items
1002_80000_NR_27.txt	1002
Falkenauer_u1000_00.txt	1000
402_10000_DI_18.txt	403
BPP_500_300_0.2_0.8_6.txt	500
Hard28_BPP645.txt	160
BPP_100_150_0.1_0.7_0.txt	100
Falkenauer_t60_00.txt	60
N1W1B1R0.txt	50

Tabela 2: Instâncias do Multidimensional 0-1 Knapsack a serem consideradas nos testes computacionais.

Nome do arquivo	#items	#dimensões
hp1.dat	28	4
weish30.dat	90	5
pb7.dat	37	30
sento1.dat	60	30
OR5x100-0.25_1.dat	100	5
OR5x100-0.75_1.dat	100	5
OR30x500-0.25_1.dat	500	30
OR30x500-0.75_1.dat	100	30

Tabela 3: Instâncias do Set Packing a serem consideradas nos testes computacionais.

Nome do arquivo	#items distintos	#conjuntos	#total items
v100qs500ts20.msc	100	500	10000
v200qs500ts20.msc	200	500	10000
v1000qs500ts20.msc	1000	500	10000
v500qs100ts100.msc	500	100	10000
v1000qs100ts100.msc	1000	100	10000
v50000qs100ts100.msc	50000	100	10000
frb30-15-1.msc	17827	450	35654
frb30-15-5.msc	17794	450	35588

Referências

[Delorme et al. 2014] Delorme, M., Iori, M., and Martello, S. (2014). Bin packing and cutting stock problems: Mathematical models and exact algorithms. In *DECISION MODELS for SMARTER CITIES*.