Otimização Combinatória: Trabalho Prático 2024/2

Prof. Henrique Becker

30 de Setembro de 2024

1 Instruções gerais

1. O trabalho prático consiste em 4 entregáveis:
   (a) formulação inteira (0.5pt),
   (b) meta-heurística (1pt),
   (c) relatório/experimentos (0.5pt),
   (d) e apresentação (0.5pt).
2. O trabalho completo deve ser entregue até às 23:59 de 2025-01-05 (domingo). Isto é, no dia anterior a primeira aula de apresentações.
   (a) A princípio, serão dedicadas duas aulas para apresentações, dias 2025-01-06 (segunda-feira) e 2025-01-08 (quarta-feira).
   (b) Neste semestre, as apresentações se darão de forma remota por meio do MConf (os links estarão disponíveis no Moodle da disciplina).
   (c) Neste semestre, não haverá prorrogação das datas, um vez que o semestre está encurtado e as datas são as últimas possíveis.
   (d) Os alunos não saberão a ordem das apresentações de antemão; esta será divulgada grupo a grupo durante os dias de apresentação.
   (e) As aulas de apresentação terão chamada e contam para o total de presenças.

3. O trabalho deve ser realizado em trios.

   (a) A formação dos trios fica a critério e responsabilidade dos alunos.

   (b) A mesma exata nota será compartilhada pelos 3 membros do trio.

   (c) É responsabilidade de cada aluno formar um trio com outros alunos que irão contribuir com o trabalho tanto quanto ele próprio.

   (d) Há a possibilidade, mas não a garantia, de que será permitido fazer duplas caso haja dificuldade em formar trios. Entretanto, isso não será permitido nesse primeiro momento, só quando/se for enviado e-mail pelo professor autorizando.

4. O tópico do trabalho é aplicar uma meta-heurística sobre um problema.

   (a) A escolha do tópico é responsabilidade dos trios.

   (b) Não pode ser escolhido o mesmo tópico já escolhido por outro trio.

   (c) Escolher meta-heurística ou problema não descrita aqui exige uma justificativa por parte to grupo e aprovação por parte do professor.

   (d) Uma lista dos tópicos disponíveis se encontra em: `https://link.inf.ufrgs.br/151-ACFo`

   (e) Assim que um trio decidir por um tópico (i.e., par de 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.

   (f) Este e-mail será respondido pelo professor confirmando ou não a possibilidade (após verificar outros e-mails e atualizar a planilha).

5. 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 (para passar ao solucionador[1], não para a RNG da linguagem), e o limite de tempo (em segundos), e utiliza uma formulação inteira e o solucionador HiGHS para resolver o problema.

[1] No caso de Julia/JuMP/HiGHS, a semente de aleatoriedade do solucionador pode ser definida com `set_attribute(model, "random_seed", seed_parametro)`.

(b) O HiGHS é o solucionador recomendado mas outros solucionadores podem ser utilizados caso seja a preferência dos alunos.

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

(d) São critérios de avaliação da implementação da formulação inteira: a sua corretude, a sua legibilidade, o emprego do que aprendido na disciplina, e decisões que levam a formulação a ter mais ou menos variáveis e ser mais apertada ou mais frouxa.

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

6. 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) Após a geração da solução inicial, e toda a vez que implementação encontra uma solução melhor que a melhor conhecida até o momento, esta deve escrever na saída padrão: (i) a quantidade de segundos (com duas casas após a vírgula) desde o começo da execução; (ii) o valor dessa solução; (iii) uma representação da mesma que seja possível de ser compreendida por um ser humano. Além disso, a implementação também deve escrever na saída padrão quaisquer outras informções usadas para montagem das tabelas do relatório.

(c) 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 (idealmente por meio de argumentos posicionais na linha de comando). Dentre esses argumentos, obrigatoriamente, para todas as meta-heurísticas, devem se encontrar a semente de aleatoriedade (exceto se completamente determinística) e o valor que controla o critério de parada (número de iterações, ou qualquer outro escolhido).

(d) O código não pode suportar *somente* o critério de parada por tempo. Os experimentos exigem rodar o código por 5s e 300s,

e para isso é útil um critério por tempo, porém nesse caso é necessário entregar algum outro critério determinístico (como número de iterações) que retorne exatamente a mesma solução.

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

(f) São critérios de avaliação da implementação da meta-heurística: a sua corretude, a sua legibilidade, a adequação a meta-heurística escolhida (vide item acima), e a sua performance (especialmente no que tange a escolha de representação da solução, a exagerada criação novos objetos de solução ao invés de sua alteração, e o recálculo do valor da função objetivo do a partir zero ao invés de por delta).

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

7. 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 *somente* tempo).

   (c) **Execuções da formulação:** O resolvedor toma uma semente de aleatoriedade que influencia o caminho até a solução ótima (quais soluções o resolvedor encontra no meio do caminho, quando encontra as mesmas, quanto tempo leva até a solução ótima, etc...). As sementes podem ser os valores de 1 até 10. Os valores apresentados podem ser médias dos valores soluções encontradas (e médias

dos tempos de execução). Estes valores podem ser apresentados direto na tabela de comparação do item 7e.

(d) **Comparações preliminares:** Com a intenção de apresentar a versão da sua meta-heurística com os melhores resultados possíveis, é esperado que os alunos variem e testem diferentes valores para os diferentes parâmetros. Essa experimentação pode ser exploratória e não envolver um método sistemático e/ou uma tabela de resultados, mas uma breve justificativa deve ser dada para porque se escolheu os valores para cada parâmetro da meta-heurística (caso a meta-heurística seja determinística, os alunos devem apresentar a análise da variação de um desses parâmetros na tabela do item seguinte ainda).

(e) **Execuções finais da meta-heurística** Os experimentos descritos nesse item devem ser repetidos para dois limites de tempo: 5 segundos e 300 segundos (i.e., 5 minutos). Caso a meta-heurística seja determinística, escolher um parâmetro da meta-heurística para variar e comparar os resultados. Caso a meta-heurística não seja determinística, variar a semente de aleatoriedade. Apresentar 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 um valor distinto para a semente ou parâmetro escolhido, e.g., se são 5 instâncias então 50 linhas) com as seguintes colunas:

   i. Nome da instância (ou fazer uma tabela por instância).
   ii. Valor do parâmetro escolhido ou da semente de aleatoriedade.
   iii. Valor da solução inicial da meta-heurística: S_i.
   iv. Valor da melhor solução encontrada pela meta-heurística: S_h.
   v. Tempo de execução da meta-heurística (segundos): *H T (s)*.
   vi. Valor médio da solução encontrada pela formulação: S_f.
   vii. Caso termine por limite de tempo, o limite superior[2]: U_f.
   viii. Tempo médio de execução da formulação (segundos): *F T (s)*.

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

(g) Conclusões e bibliografia utilizada.

[2]Em Julia/JuMP o limite superior é dado pela função `objective_bound`.

8. O entregável **apresentação** consiste em:
   (a) Uma apresentação para turma de *no máximo* 5 minutos **por integrante**.
      i. A apresentação será interrompida se ultrapassar o limite de tempo.
      ii. Caso o tempo acabe consideravelmente antes de o material que se propunha apresentar
   (b) O objetivo da apresentação é apresentar as implementações e os resultados.
   (c) Após o término da apresentação, o professor e os alunos tem 3 minutos para perguntas.
   (d) Cada membro do grupo deve participar e comentar o que fez.
9. Os 4 entregáveis devem ser enviados em um pacote respeitando as seguintes imposições:
   (a) A entrega será via formulário no Moodle da disciplina.
   (b) O pacote deve estar em um dos seguintes formatos: tar, tar.gz, tar.bz2, zip, rar, ou 7z.
   (c) O nome do pacote deve ser: `inf05010_2024-2_<letra da turma>_TP_Grupo-<número do grupo>.<extensão>`
   (d) O relatório e a apresentação de slides (exatamente a mesma a ser usada no dia da apresentação) devem estar em formato PDF, e terem nome, respectivamente: `relatorio.pdf` e `apresentacao.pdf`.
   (e) O código da formulação inteira deve estar numa pasta chamada `formulation` que possui um arquivo README detalhando o processo de compilação/execução, o mesmo vale para o código da meta-heurística porém o nome da pasta deve ser `metaheuristic`.

Algumas dicas:

- Acerca da escolha de vizinhanças: uma boa vizinhança é aquela que permite alcançar qualquer solução *dadas suficientes iterações* (a vizinhança imediata de uma determinada solução não deve conter todas as possíveis soluções).
- No caso de vizinhos com mesmo valor de solução, utilizar escolhas aleatórias como critério de desempate podem trazer bons resultados.

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).

1. Late Acceptance Hill Climbing: 'BURKE, Edmund K.; BYKOV, Yuri. The late acceptance Hill-Climbing heuristic. European Journal of Operational Research, v. 258, n. 1, p. 70–78, 2017.' (`https://doi.org/10.1016/j.ejor.2016.07.012`)

2. Tabu Search: 'Tabu Search: A Tutorial, by Fred Glover (1990), Interfaces.' (`https://www.ida.liu.se/~zebpe83/heuristic/papers/TS_tutorial.pdf`).

3. Iterated Local Search: 'Helena Ramalhinho Lourenço; Olivier Martin; Thomas Stützle. A beginner's introduction to iterated local search. In: Proceeding of the 4th Metaheuristics International Conference. Porto, Portugal: [s.n.], 2001.' (`http://84.89.132.1/~ramalhin/PDFfiles/2001_MIC_FILS.pdf`)

4. GRASP: 'T.A. Feo, M.G.C. Resende, and S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Operations Research, vol. 42, pp. 860-878, 1994' (`http://mauricio.resende.info/doc/gmis.pdf`).

5. Simulated Annealing: 'Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning, by D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Operations research 37 (6), 865-892, 1989.' (`https://pubsonline.informs.org/doi/epdf/10.1287/opre.37.6.865`)

6. VNS: 'A Tutorial on Variable Neighborhood Search, by Pierre Hansen (GERAD and HEC Montreal) and Nenad Mladenovic (GERAD and Mathematical Institute, SANU, Belgrade), 2003.' (`http://www.cs.uleth.ca/~benkoczi/OR/read/vns-tutorial.pdf`)

7. Genetic Algorithm: 'A genetic algorithm tutorial, by D. Whitley, Statistics and computing 4 (2), 65-85.' (`http://link.springer.com/content/pdf/10.1007%252FBF00175354.pdf`)

3 Problemas NP-completos permitidos

Balls and Bins	Organização de Eventos

1. **Balls and Bins**

Instância Uma instância é composta de n recipientes e m bolas. Cada recipiente i ($i \in [n]$) tem duas propriedades: um lower bound l_i e um upper bound u_i, onde sempre $l_i \leq u_i$. As bolas não possuem nenhuma propriedade, sendo indivisíveis e indistinguíveis entre si. É garantido que $\sum_{i \in [n]} l_i \leq m \leq \sum_{i \in [n]} u_i$.

Objetivo Determinar quantas bolas serão depositadas em cada recipiente (não pode restar nenhuma bola fora dos recipientes), de forma a *maximizar* o lucro total, enquanto respeitando as restrições de quantidade mínima e máxima de cada recipiente. O lucro é calculado da seguinte forma: se há uma bola em um recipiente, o valor é 1, se há 2, é 1 + 2, se há 3, é 1 + 2 + 3, e assim por diante (i.e., $(n(n+1))/2$). (Observação: a forma mais imediata de calcular essa função objetivo é não-linear e, portanto, não pode ser utilizada diretamente na formulação.) Uma solução só é válida se a quantidade de bolas dentro do recipiente i fica entre l_i e u_i (ambos valores incluídos, ou seja, se $l_i = 3$ e $u_i = 10$, a quantidade de bolas em i pode ser 3, 10, ou qualquer valor intermediário).

2. **Organização de eventos**

Instância Uma instância é composta de uma quantidade de espaços n todos com a mesma metragem M, e uma quantidade de atrações m onde cada atração tem duas propriedades: a sua temática t_j ($j \in [m]$) e a sua dimensão d_j ($j \in [m]$). Todos valores são inteiros positivos e todos conjunto são base 1 (i.e., as temáticas, por exemplo, começam da 1 e vão até a T).

Objetivo Escolher em que espaço alocar cada atração, de forma a respeitar o tamanho dos espaços (a soma das dimensões de todas as atrações deve ser menor ou igual que a metragem do espaço), e a minimizar a dispersão somada de todas as temáticas. A dispersão de uma temática é calculada como o número de espaços distintos que possuem pelo menos uma atração daquela temática.

3.1 Instâncias dos problemas

Os pacotes com as instâncias de cada problema estão disponíveis em:

- Bins and Balls – `https://link.inf.ufrgs.br/153-aalB`
- Organização de eventos – `https://link.inf.ufrgs.br/152-qkCO`

Estas instâncias são as mesmas referenciadas nas tabelas apresentadas na última página.

O formato das instâncias de cada problema é descrito abaixo:

1. **Bins and balls:** A primeira linha do arquivo informa o número de recipientes (n). A segunda linha do arquivo informa o número de bolas m. Cada linha a partir da terceira linha do arquivo até a linha $n + 2$ possui dois números inteiros positivos: o limite inferior do recipiente i e o limite superior do recipiente i, respectivamente.

2. **Organização de eventos:** A primeira linha do arquivo informa o número de espaços (n). A segunda linha do arquivo informa a metragem M (que é a mesma para todos os espaços). A terceira linha do arquivo informa o número de temáticas diferentes (T). A quarta linha do arquivo informa o número de atrações m. As m linhas seguintes (da linha 5 até a $m + 4$) descrevem cada uma das m atrações. Cada linha descrevendo uma atração contêm, nesta ordem, os seguintes valores (na forma de números inteiros positivos separados por espaços): temática da atração t_j (valor de 1 até T) e a dimensão da atração d_j (valor de 1 até M, mas geralmente muito menor que M).

Algumas informações das instâncias selecionadas de cada problema são descritas abaixo. Em especial, *bkv* significa *best known value* (melhor valor conhecido), e *bound* é um limite do melhor valor possível. Logo, num problema de maximização, um valor maior que *bound* significa que há algo de errado (e num de minimização, um valor menor que *bound* também significa algo de errado). Idealmente, as meta-heurísticas deveria tentar retornar de forma consistente, para o limite de tempo de 300s, valores próximos do *bkv* (ou até melhores que ele, mas nunca melhores que o bound).

Tabela 1: Instâncias do Bins and Balls a serem consideradas nos testes computacionais.

Nome	#bins	#balls	*bkv*	*bound*
01.txt	100	13483	1034999	1035016
02.txt	100	43636	11984785	11984837
03.txt	200	108069	45933095	45984383
04.txt	200	111541	34178023	34178457
05.txt	800	107860	8202021	8202061
06.txt	800	200380	29745831	29752395
07.txt	100	182904	206626542	206914171
08.txt	100	80055	42668196	42668333
09.txt	400	67676	8461348	8461955
10.txt	400	173253	48278994	48352227

Tabela 2: Instâncias do Organização de Eventos a serem consideradas nos testes computacionais.

Nome	n	M	T	m	*bkv*	*bound*
01.txt	100	10	65	285	123	93
02.txt	100	500	7	524	103	97
03.txt	10	100	6	139	12	12
04.txt	15	100	8	216	18	17
05.txt	100	12	81	336	136	102
06.txt	50	400	3	224	50	50
07.txt	10	100	4	135	12	11
08.txt	59	150	3	120	50	48
09.txt	366	150	3	700	308	294
10.txt	70	8	34	154	74	65