Guia Para Orientações
Versão 1.1

Marco Antonio Moreira de Carvalho

13 de julho de 2023

Este documento é parte do material de apoio[1] para orientações do Prof. Marco Antonio Moreira de Carvalho[2], do departamento de Ciência da Computação da Universidade Federal de Ouro Preto. Deliberadamente, não há intenção de que o conteúdo seja tecnicamente (e politicamente) correto ou completo. O conteúdo deste documento é para uso pessoal e pode ser distribuído sob a licença *Creative Commons* CC BY-NC-ND, i. e., só é permitido o *download* e compartilhamento desde que atribuam crédito ao autor, sem autorização para alteração ou utilização para fins comerciais.

[1] https://shorturl.at/esEHN

[2] www.decom.ufop.br/marco

Visão Geral

Minhas áreas de pesquisa preferenciais são pesquisa operacional e algoritmos. Tenho preferência por problemas de *scheduling* com aplicação prática na indústria e engenharia, principalmente os que envolvem permutações e matrizes. O seu tema pode ser desenvolver um método específico para um destes problemas de otimização ou então estudar um método/ferramenta em particular.

Caso seja necessário, você passará por um nivelamento em projeto e análise de algoritmos, teoria dos grafos e também em programação usando C++ e STL.

Seu tema será apresentado brevemente por mim (caso você não tenha um em mente), com alguns exemplos e indicação de propriedades importantes.

If we knew what it was we were doing, it would not be called research, would it?
– Albert Einstein

Um artigo será utilizado como base para você entender o problema e realizar a primeira implementação. Inicialmente, você deve ler o artigo, entender o problema e os exemplos dados, tentando resolver à mão algum exemplo do problema usando o método descrito pelo próprio artigo. Em uma próxima oportunidade, você explicará para mim o problema e como funciona o método que você executou manualmente e como chegou à solução.

Após entender o problema, você realizará a revisão da literatura para se inteirar sobre o que já foi feito a respeito do seu tema e se inspirar também. Boas idéias podem ter passado batidas por outros autores. Revisar o que já foi feito também evita o erro comum de realizar uma pesquisa inteira e depois descobrir o que você fez já existe.

"Oh, you can't help that,' said the cat. 'We're all mad here."
– Lewis Carroll, *Alice in Wonderland*

O restante do trabalho se dará em projeto de algoritmos, imple-

mentações, testes preliminares, revisão das implementações, testes extensivos, análise de dados e escrita de textos. Depois virão as participações em congressos, prêmios de iniciação científica e defesas.

Algumas destas etapas estão descritas nas seções a seguir. Normalmente teremos reuniões semanais de acompanhamento.

Nivelamento em Projeto e Análise de Algoritmos e Teoria dos Grafos

DOMINAR O PROJETO E ANÁLISE DE ALGORITMOS E TEORIA DOS GRAFOS é primordial em minha área de pesquisa. Não há alternativa.

Você terá algumas aulas sobre PAA e teoria dos grafos com introdução de conceitos e técnicas importantes também na pesquisa operacional. Será utilizado o material especifico para isto, verifique a pasta "Nivelamento PAA"[3].

[3] https://shorturl.at/esEHN

Nivelamento em Programação C++ e STL

DOMINAR PELO MENOS UMA LINGUAGEM DE PROGRAMAÇÃO e sua biblioteca padrão é primordial. Eu sugiro fortemente C++, por razões históricas e de desempenho. Entretanto, você poderá utilizar outra linguagem, desde que a domine.

Se necessário, você terá algumas aulas sobre algoritmos, estruturas de dados, grafos e programação em C++. O intuito é que você compreenda e passe a utilizar sem dificuldades os contêineres, iteradores e algoritmos disponíveis na STL. Verifique os padrões de codificação a serem adotados na seção *Implementações*.

Será utilizado o material especifico para isto, verifique a pasta "Nivelamento programação"[4]. O material foi criado e utilizado por mim nos anos de 2014 e 2015 para um projeto de extensão em cooperação com o IFMG campus Ouro Preto para treinamento para Olimpíada Brasileira de Informática, então o nível é introdutório.

[4] https://shorturl.at/esEHN

Inscreva-se no URI, uma plataforma *online* para realização de exercícios de programação com correção automática [5]. Me envie seu *id* de usuário para que eu possa verificar seu progresso.

[5] https://www.urionlinejudge.com.br/judge/

O URI é o que chamamos de juiz automático *online*, muito utilizado em competições de programação. Os problemas são enunciados fora de contexto, sem muita dica a respeito da solução, há uma especificação de como será realizada a entrada e como deve ser feita a saída dos dados. Há também uma entrada de exemplo e a saída correspondente.

Você deve submeter um código que leia a entrada no formato especificado, resolva o problema dentro do limite de tempo estipulado e apresente a resposta no padrão exigido. O código submetido será então testado com diferentes casos de teste e ao final será apresentado o percentual de casos de teste em que seu código obteve sucesso. Todos os seus códigos devem ter 100% de aceitação.

Note que diferentes tipos de erro podem ocorrer no seu código:

- Resposta errada;
- Tempo esgotado;
- Formato errado da saída (erro de apresentação);
- Erro de compilação;
- Erro de execução.

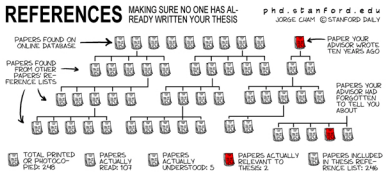
Comece pelos problemas de nível iniciante, revise o material de cada aula passada e posteriormente faça os exercícios indicados no material. Em caso de dúvidas, verifique o fórum de cada problema, no qual os usuários discutem como resolver o problema ou me procure.

Para verificar erros de apresentação, utilize a ferramenta *diff* [6]. Também há uma versão no terminal dos sistemas UNIX.

[6] `https://www.diffchecker.com/diff`

Revisão Bibliográfica

Revisar o que já foi feito evita o erro comum de realizar uma pesquisa inteira e depois descobrir o que você fez já existe. Também é muito útil para posicionar seu trabalho frente ao que já existe.



Repositórios Acadêmicos Online

Utilize os repositórios *online* de citações abaixo para pesquisar os trabalhos que foram publicados sobre o seu tema. Usualmente o *Web of Science* é suficiente, mas outros podem complementar o trabalho de revisão. Todos eles são acessíveis gratuitamente pela rede da UFOP.

Uma vez realizada uma busca sobre o tema, refine por categorias relacionadas a computação, engenharia, pesquisa operacional e *management science*.

Com a busca pronta, crie alertas de publicação. Cada nova publicação que atender a sua pesquisa será avisada a você por *e-mail*, e deste jeito sua revisão estará sempre atualizada.

- *Web of Science* (veja como acessar abaixo)
- *Semantic Scholar* [7]

[7] `https://www.semanticscholar.org`

- *Science Direct* [8]
- *IEEE Explore* [9]
- *Google Acadêmico* [10]
- Portal de Periódicos da CAPES [11]

[8] http://www.sciencedirect.com

[9] http://ieeexplore.ieee.org/Xplore/

[10] https://scholar.google.com.br

[11] https://www.periodicos.capes.gov.br

Originalmente, os repositórios utilizavam autenticação via IP: você se logava a partir da rede da UFOP e depois estava liberado para usar de qualquer outro lugar. Entretanto houve alterações, sendo o acesso a todos os repositórios realizado por meio do Portal de Periódicos da CAPES (veja o link acima). Uma maneira simples de acessar o *Web of Science* é acessar o portal de periódicos da CAPES e na página inicial buscar pela base. Basta clicar no nome da base e você será redirecionado para o site. Ainda é necessário se autenticar com usuário e senha.

O *Semantic Scholar* não exige autenticação, porém, eventualmente não é disponibilizado o PDF do artigo desejado. Entretanto, ele é uma boa ferramenta para levantar a literatura sobre um problema, porque possui apenas literatura em computação e neurociência, o que reduz um pouco os resultados. Além disto, ele classifica os artigos mais influentes de uma determinada busca, oferece artigos relacionados e também extrai palavras chave melhor, ajudando a filtrar os artigos que interessam.

Ao ler os artigos, esteja atento a quantidade de citações que o artigo recebeu (*times cited*) – quanto mais citado, mais importante. Artigos que não são citados não são relevantes. Uma exceção são artigos recentes, que não tem muitas citações eventualmente por serem novidade.

Se possível, baixe os artigos para uso posterior e também baixe a referência no formato bibtex (ou .bib), que te ajudarão posteriormente na hora da escrita. Boas fontes de arquivos .bib são *CiteSeerX*[12], *The Collection of Computer Science Bibliographies*[13] e o próprio Google Acadêmico[14]. Existem também alguns softwares gerenciadores de referências, como *Mendeley*[15] e *Zotero* [16], embora eu mesmo não use nenhum.

[12] http://citeseerx.ist.psu.edu/index

[13] http://liinwww.ira.uka.de/bibliography/index.html

[14] https://scholar.google.com.br

[15] https://www.mendeley.com

[16] https://www.zotero.org

Alguns artigos importantes devem ser identificados na sua revisão:

- O artigo que define o problema/método original;
- O artigo que determina a complexidade computacional do problema;
- Artigos que propõem instâncias para o problema;
- Os artigos cujos métodos propostos superaram todos os outros do estado da arte;

- Os artigos cujos métodos propostos compõem o estado da arte atual.

Ao ler um artigo pela primeira vez, faça uma espécie de "fichamento" para não ter que ler de novo depois:

- Ano do artigo;
- Título do artigo;
- Nome dos autores;
- Onde foi publicado;
- Método utilizado (nome);
- Quais instâncias foram usadas nos experimentos computacionais, tamanho, origem, nível de dificuldade;
- Qualidade dos resultados: com quem comparou, se melhorou ou piorou;
- Observações.

Verifique e baixe todas as instâncias disponíveis na internet (nos sites dos autores, por exemplo) que foram referenciadas nos artigos que você leu. Solicite diretamente a autores as que não estiverem disponíveis. Verifique também a seção *Bibliotecas de Instâncias* no final deste documento para repositórios que armazenam instâncias para variados problemas. Aproveite e sumarize em uma planilha excel o melhor resultado conhecido para cada instância, você os utilizará posteriormente nos experimentos computacionais.

Verifique como escrever a revisão da bibliografia na seção *Estrutura Geral de Textos*.

Programa de Comutação Bibliográfica

Caso não consiga acessar ou encontrar um determinado artigo nos repositórios *online*, tente o Programa de Comutação Bibliográfica – COMUT[17] com o bibliotecário do ICEB. Podem ser obtidos:

- Artigos de periódicos;
- Trabalhos publicados em anais de eventos;
- Teses, dissertações, relatórios e demais tipos de monografias científicas;
- Partes de livros.

Normalmente o COMUT cobra um valor baixo por cada página do documento solicitado.

[17] Uma rede de serviços que permite a obtenção de cópias de documentos técnico-científicos disponíveis em acervos de bibliotecas de todo o Brasil.

Últimos Recursos

As referências bibliográficas também podem ser solicitadas diretamente em bibliotecas de outras instituições.

Implementações

Considerando C++ como a linguagem padrão, seguem abaixo alguns padrões sobre a codificação, compilação e a estrutura do que deve ser implementado inicialmente.

Uma boa fonte de boas práticas em implementação de algoritmos científicos é encontrada em *How Not To Do It* [18]. Resumidamente:

[18] Ian P. Gent, Stuart A. Grant, Ewan MacIntyre, Patrick Prosser, Paul Shaw, Barbara M. Smith and Toby Walsh. Research Report 97.27, School of Computer Studies, University of Leeds. May 1997.

Don't trust yourself: *bugs* acontecem, esteja atento;

Do it fast enough: seu código não precisa ser o mais rápido do mundo, precisa ser rápido o suficiente;

Do use version control: incluindo *backup*;

Do be paranoid: sempre verifique duas vezes;

Do check your solutions: verifique manualmente a consistência das soluções de seus programas;

Do be stupid: não ignore idéias "bobas";

Don't trust your source of random numbers: muito cuidado com sementes, números cíclicos e experimentos em série;

Na pasta "Códigos"[19] do material de apoio há três "esqueletos" de código com implementações parciais dos tópicos abordados nas seções abaixo.

[19] `https://shorturl.at/esEHN`

Os "esqueletos" se referem a três versões diferentes do mesmo código:

Quick and Dirty: Rápido e rasteiro, um gerenciador de testes (*single* e *multi-run*) e um resolvedor em C++ estruturado, usando STL;

Quick and Dirty Paralelo: Um gerenciador de testes *multi-run* e um resolvedor em C++ estruturado, usando STL, porém, executado em um ambiente *multithread*[20]; e

[20] Contribuição valiosa dos alunos Bruno Henrique Miranda dos Santos (bruno_h_m_s@hotmail.com) e Douglas Matuzalém Pontes Belo Lança (douglasvandersar@gmail.com)

Classe: Um gerenciador de testes (*single* e *multi-run*) e resolvedor em C++ utilizando classes e métodos.

Padrões

Os seguintes padrões devem ser adotados:

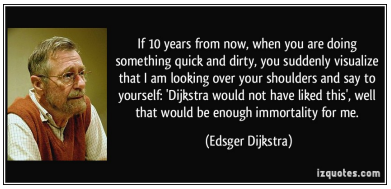
- Linguagem C/C++, padrão C++ 11;
- *flag* de compilação -O3 no g++;
- *flag* de compilação -march=native no g++[21];
- Utilização da STL em sua totalidade (contêineres, algoritmos e iteradores);
- Usar alocação dinâmica de contêineres[22];
- NUNCA use *goto*[23].

[21] Mais informações em: `https://wiki.gentoo.org/wiki/GCC_optimization`.

[22] Utilize o método *.resize()* para isto.

[23] Se você não sabe o que é, continue sem saber.

Programas Híbridos C/C++



Apesar de programarmos em C++, podemos continuar programando de maneira estruturada e usando apenas os objetos que nos interessam da biblioteca padrão – engenharia de *software* não importa neste tipo de pesquisa. Prevalece o "padrão de projeto" *Quick and Dirty*[24]: não é necessário criar classes nos códigos, *getters*, *setters*, nem mesmo separar o código em arquivos diferentes, usar padrões, variáveis globais podem ser utilizadas, etc[25]. Quando competimos com algoritmos que executam em menos de um segundo não podemos nos dar ao luxo de termos *overhead*. Esteja atento à complexidade das suas implementações também.

Apesar de adotarmos o *Quick and Dirty*, um mínimo de validação do código deve ser realizado: Para algumas instâncias pequenas verifique manualmente se o resultado das funções está correto[26].

Utilize como referência o material do nivelamento e também o C++ *reference* [27].

[24] Não confunda com *Quick and Pointless*.

[25] Menos *goto*.

[26] Como leitura da entrada, preenchimento de estruturas, cálculos, etc.

[27] `http://www.cplusplus.com`

Inicialização e Terminação

Todos os seus contêineres devem ser inicializados logo no início da execução do código e limpos ao final. Estabeleça o tamanho de cada contêiner com base nos dados da instância[28] e outros parâmetros, zere todas as variáveis numéricas. Ao final, dê um *.clear()* em todos os contêineres e libere qualquer memória utilizada.

Embora a própria linguagem cuide disto ao construir e destruir objetos, são comuns os erros ocasionados por lixo em variáveis globais em testes extensivos. Nestes experimentos, normalmente o código binário é executado uma única vez, embora o algoritmo em si seja executado diferentes vezes por instância. Desta forma, todas as variáveis e objetos permanecem na memória e podem ser reutilizados com valores antigos.

[28] Utilize o método *.resize()* para isto.

Entrada e Saída

A primeira coisa a implementar é a entrada e saída do problema, padronizando-os.

Para cada arquivo de entrada, gerar um arquivo de saída específico e individual contendo os dados da solução, de modo que seja possível verificar se uma solução está correta ou não. O nome do arquivo de saída deve fazer referência ao nome do arquivo de entrada [29]. O arquivo de saída deve conter:

- Número da execução (no caso de executar várias vezes para uma mesma instância);
- Parâmetros utilizados;
- Valor da função objetivo;
- Tempo de execução;
- Resultado do pré-processamento;
- Resultado final (permutação dos elementos, matrizes, etc.)

Para cada conjunto de testes, gerar um arquivo de resumo de resultados [30], contendo em cada linha:

- Nome da instância;
- Número da execução;
- Valor da função objetivo;
- Tempo de execução.

Estes arquivos serão utilizados posteriormente para geração das demais estatísticas. Confira os arquivos de exemplo na pasta *Exemplos arquivos de saida*.

[29] e.g. metodoX_solucao_instanciaY.txt

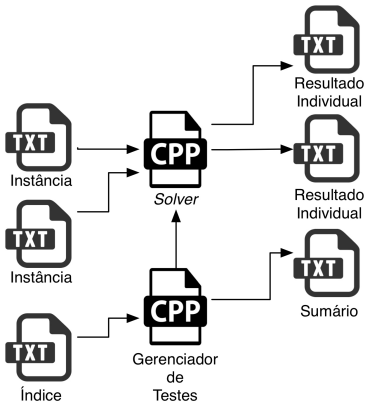


Figura 1: Esquema para entrada e saída.

[30] e.g., metodoX_resumo_resultados_conjuntoY.txt

Função de Avaliação

A função de avaliação literalmente avalia o custo de uma solução calculada por um método. Algumas são simples expressões matemáticas e outras envolvem algoritmos adicionais. Ao implementar a função de avaliação, verifique manualmente se o cálculo está correto para algumas instâncias. Teste para instâncias de conjuntos diferentes e com características diferentes.

Em métodos que envolvem buscas locais, é essencial implementar uma **função de avaliação incremental**: dados uma solução s e um movimento m, uma função $\Delta(s, m)$ avalia somente a transformação da solução s após a aplicação do movimento m, utilizando

informações anteriores. Desta forma, a avaliação é mais leve e consome menos tempo, porém, estas vantagens são obtidas ao preço de maior consumo de memória. Há um exemplo clássico para o método *2-opt*[31]. Na impossibilidade de uma avaliação incremental, uma alternativa é uma avaliação aproximada, que seja mais rápida, porém, imprecisa.

[31] Talbi, El-Ghazali. Metaheuristics: from design to implementation. Vol. 74. John Wiley & Sons, 2009.

Pré-Processamento

Métodos de pré-processamento são aplicados antes de começarmos a resolver uma instância para um problema e tem por finalidade reduzir o tamanho da instância pela remoção de informações redundantes ou irrelevantes e também detectar estruturas específicas que facilitem a solução de um problema.

Modelar um problema usando grafos é uma boa maneira de detectar estruturas específicas. Por exemplo, problemas classificados como NP-difícil podem ter solução em tempo determinístico polinomial caso o grafo resultante seja uma árvore, 1-árvore, grafo completo, etc. Outro exemplo, mais comum ainda, ocorre quando a solução do problema é realizada por uma busca no grafo: se o grafo possuir mais de um componente, então a instância pode ser dividida em duas ou mais de tamanho menor, em que cada problema é um dos componentes do grafo. Uma BFS/DFS serviria para este pré-processamento.

Verifique na sua revisão da literatura se algum método de pré-processamento já foi utilizado. Podemos projetar novos métodos também.

Geração de Números Aleatórios

A biblioteca *random* da STL possui geradores de diversas distribuições de números. Para gerar um número dentro de um intervalo, normalmente utilizamos a distribuição uniforme [32]) ou normal [33]. Entretanto, verifique qual distribuição deve ser utilizada (bernoulli, binomial, geométrica, poisson, exponencial, etc.).

No caso de você precisar selecionar aleatoriamente elementos de um conjunto (um vetor por exemplo) sem poder repetí-los, utilize a função *shuffle* [34], que embaralha o vetor aleatoriamente e depois simplesmente o percorra [35].

Em ambos os casos, utilize o *engine* padrão [36].

[32] Veja o exemplo: `http://www.cplusplus.com/reference/random/uniform_int_distribution/`

[33] Veja o exemplo: `http://www.cplusplus.com/reference/random/normal_distribution/`

[34] Não confundir com *random_shuffle*.

[35] Veja o exemplo em `http://www.cplusplus.com/reference/algorithm/shuffle/?kw=shuffle`

[36] *default_random_engine generator*

Tomada de Tempo

Verifique qual é o padrão de tomada de tempo dos outros métodos utilizados em seus experimentos. Normalmente, o tempo de execução não engloba entrada e saída de dados, somente a execução do

algoritmo em si. Porém, se seu código realiza pré-processamento ao ler a entrada, este o tempo deve ser considerado, uma vez que seu algoritmo já está sendo executado.

Use o objeto *high_resolution_clock* da biblioteca *chrono* da STL [37].

[37] Veja o exemplo em `http://www.cplusplus.com/reference/chrono/high_resolution_clock/now/`

Limitante Inferior Para a Solução

Caso haja um limitante inferior[38] trivial para o valor da solução, de acordo com a sua revisão da literatura, implemente-o. Em métodos que buscas locais ou fases de aprimoramento (*polishing*) são aplicadas sucessivamente, um critério de parada comum é quando a solução obtida for igual ao limitante inferior. Nesse caso, a solução obtida é comprovadamente ótima.

[38] Um limitante inferior é um valor de referência (possivelmente não viável) que é menor ou igual ao valor da solução ótima. Normalmente é obtido por meio da relaxação de alguma restrição.

Projeto de Experimentos Computacionais e Análise de Dados

Tão importante quanto desenvolver um método computacional de qualidade para um problema de interesse é saber apresentá-lo e demonstrar que de fato o método é relevante. Para isso é extremamente importante projetar os experimentos computacionais e saber analisar os dados obtidos.

Novamente, alguns "mandamentos" do *How Not To Do It* [39]:

Do check your health regularly: verifique a consistência das suas soluções em relação às da literatura;

Do look at the raw data: dados resumidos escondem os fatos, dê uma olhada nos dados miúdos;

Do look for good views: procure um ponto de vista interessante para os seus dados;

Do report negative results: não varra a sujeira para debaixo do tapete ou ajuste suas conclusões de acordo com os dados.

[39] Ian P. Gent, Stuart A. Grant, Ewan MacIntyre, Patrick Prosser, Paul Shaw, Barbara M. Smith and Toby Walsh. Research Report 97.27, School of Computer Studies, University of Leeds. May 1997.

Ética e Relato de Experimentos

Nem toda pesquisa resulta em avanço do estado da arte. Ética importa.

O dever de um cientista é apresentar os fatos de uma maneira isenta, e não advogar a favor de uma posição em particular na tentativa de influenciar os outros. Além disto, um cientista deve evitar que seu trabalho seja mal compreendido ou que implique em mais do que deve[40].

[40] Saul I. Gass

Os experimentos computacionais devem ser justos e adotar os mesmos padrões de qualidade para todos os métodos envolvidos.

Nada de dois pesos e duas medidas. Forjar resultados por meio da alteração ou maquiagem de dados no intuito de "obter" resultados esperados ou aceitáveis é uma prática abominável e pode levar à cassação de títulos acadêmicos, bem como à anulação de publicações.

Calibrar parâmetros

Use o *iRace* [41] para calibrar parâmetros de seu código se necessário, por exemplo:

- Número total de iterações em que o código será executado;
- Número de iterações máximo sem melhorias à solução;
- Percentual de aplicação de busca local com aleatoriedade;
- Percentual de aplicação de procedimentos de perturbação;
- Taxas de mutação, cruzamento, elitismo, etc.

Os valores de parâmetros associados ao método computacional devem ser os mesmos para todas instâncias, no entanto, há flexibilidade quando se trata de instâncias com estruturas ou dimensões muito diferentes. Normalmente, o conjunto total de instâncias deve ser dividido em dois: um para calibrar os parâmetros e outro para avaliar o desempenho do método computacional. Caso contrário, pode ocorrer *overfitting*, afetando a robustez do método proposto.

No material de apoio há a pasta *iRace* com material relativo a este pacote. Há uma apresentação de um minicurso introdutório sobre o *iRace*, criado pela Helen Costa, então aluna de doutorado do programa de pós-graduação do DECOM. Há duas subpastas *tuning* e *tuning-BRKGA*, que possuem dois *setups* do *iRace* prontos para calibrar os parâmetros de dois códigos de exemplo.

Gráficos

Use *boxplots* (ou diagramas de caixa) para ilustrar o desempenho dos algoritmos (ver *GNUPlot* e *R* [42]). Um *boxplot* é um gráfico utilizado para avaliar a distribuição empírica de dados, e é poderoso por concentrar muita informação com simplicidade. Pode ser utilizado também para uma comparação visual entre dois ou mais grupos de dados, como os resultados obtidos por dois algoritmos. Apresentando *boxplots* e dados estatísticos são uma maneira eficiente de apresentar e analisar dados.

Um tutorial passo a passo é disponibilizado pela Universidade Federal Fluminense[43]. Também há uma apostila, dos mesmos autores [44].

[41] Veja a seção *Softwares*.

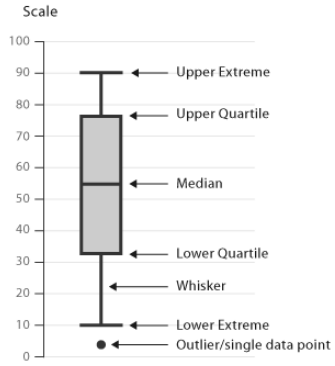


Figura 2: Um *boxplot* comentado.

[42] Veja a seção *Softwares*.

[43] `http://www.uff.br/cdme/conheceboxplot/conheceboxplot-html/conheceboxplot_intro.html`

[44] `http://www.uff.br/cdme/conheceboxplot/conheceboxplot-html/boxplot.pdf`

Análise de Desempenho e Estatística

Ao analisarmos o desempenho de um método computacional usamos indicadores como qualidade da solução, esforço computacional e robustez. Outros critérios podem ser abordados, como facilidade de implementação, simplicidade, facilidade de utilização, etc.

Diferentes tipos de dados estatísticos podem ser retirados dos resultados de um experimento computacional, dependendo se estamos comparando um método com os melhores resultados da literatura (normalmente obtidos por diferentes métodos), com os resultados de métodos específicos da literatura ou com resultados ótimos.

O ***gap*** indica a distância percentual entre o valor obtido para função objetivo e o valor desejado. Cálculo: $100 \times$ (valor_obtido-valor_desejado)/valor_desejado. Podemos calcular o *gap* em relação à melhor solução conhecida, à solução ótima ou em relação a um limitante inferior.

O **esforço computacional** é medido com a análise assintótica, normalmente o pior caso, de cada procedimento que o compõe o método proposto e também o método como um todo.

A **consistência**, em geral, é demonstrada pela insensibilidade do método à alterações na entrada, à valores de parâmetros e à diferentes execuções para uma mesma instância (quando o método contar com componentes aleatórios). Quando as soluções são consistentemente boas, o algoritmo é considerado **robusto**. Para medir a consistência/robustez, utilizamos o desvio padrão, o coeficiente de variação e a taxa de sucesso, definida como:

$$\text{taxa de sucesso} = \frac{\text{número de execuções com sucesso}}{\text{número total de execuções}}$$

O *sucesso* pode ser definido como atingir a solução ótima, ou melhor solução conhecida.

Outros Dados Estatísticos

Analisando-se os resultados para todas as instâncias de um conjunto específico, podemos analisar:

- Percentual de soluções ótimas (ou melhores conhecidas) atingidas;
- Percentual de soluções iguais, melhores ou piores, quando comparado a outro método específico.

Quando os métodos utilizados possuem componentes de aleatoriedade, a cada execução para uma mesma instância podemos ter soluções diferentes. Assim, realizamos diferentes execuções independentes de um mesmo método para uma mesma instância, e podemos analisar:

- Resultados mínimo, médio, máximo, desvio padrão e coeficiente de variação [45];
- Tempo mínimo, médio e máximo de execução, além do desvio padrão;
- Convergência do algoritmo;[46]
- Tempo exigido para geração de uma solução inicial, melhoria percentual obtida na qualidade da solução pela etapa de aprimoramento.

[45] Definido como desvio padrão dividido pela média. Pode ser multiplicado por 100 para ser expresso em porcentagem. Deve ser interpretado como a variabilidade dos dados em relação à média. Quanto menor for o coeficiente mais homogêneo será o conjunto de dados.

[46] Quantas iterações/tempo o algoritmo demora para atingir a melhor solução encontrada por ele?

Métodos iterativos, que executam buscas locais sucessivamente, como ILS, VND e VNS também permitem que outros dados sejam colhidos:

- Número de iterações até atingir o melhor resultado[47];
- Tempo de execução até atingir o melhor resultado.

[47] Note que o algoritmo pode continuar rodando sem melhorar a solução atual.

Comparando Tempos de Execução em Máquinas Diferentes

Não é possível realizar uma comparação justa de tempos de execução de diferentes métodos que foram executados em diferentes arquiteturas. O ideal é executar os métodos no mesmo ambiente computacional.

O *Passmark*[48] é um *site* de *benchmarking* de CPU que contempla muitos ambientes computacionais. Com os dados reportados por ele, é possível obter uma proporção aproximada entre diferentes arquiteturas e com isso realizar uma comparação aproximada dos tempos de execução. Na seção CPU *Benchmarks* há o *Passmark CPU mark*. Encontre uma das CPUs relacionadas aos métodos comparados, adicione a outra CPU na lista de comparação e divida o maior valor de *CPU Mark* pelo menor. O resultado é um fator que deve multiplicar o tempo de execução da CPU de maior *CPU Mark*.

[48] `www.cpubenchmark.net`

Por exemplo, uma CPU 686 Gen tem *CPU Mark* 288, enquanto uma CPU Intel Xeon E5-1650 v3 @ 3.50GHz tem *CPU Mark* 13603. A segunda CPU é $\approx$ 47,23 vezes mais rápida que a primeira, e portanto, o tempo de execução do método rodadao na primeira CPU deve ser multiplicado por este fator.

Time-To-Target Plots

O *Time-To-Target Plots*[49], ou *ttt-plots* são uma maneira de reportar a convergência de um método. A hipótese é a de que o tempo de execução de um método se assemelha a uma distribuição exponencial se o método for executado uma quantidade suficiente de vezes.

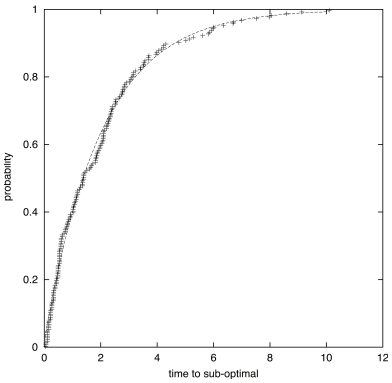


Figura 3: ttt-*plot*.

[49] `http://mauricio.resende.info/tttplots/`

O método é executado várias vezes independentes (e.g., 100) para algumas instâncias selecionadas e é parado quando um resultado aproximado de um alvo é obtido. Normalmente se observa um resultado 5% próximo à melhor solução conhecida para cada instância, ou solução ótima. O ttt-plot indica, dada uma quantidade fixa de tempo, qual a probabilidade que o método possui de gerar uma solução tão boa quanto a do resultado alvo. Quando diferentes métodos são comparados quanto à convergência, a posição relativa dos gráficos na curva indica qual método possui maior probabilidade de obter tal solução.

Há um *script* em linguagem perl que pode ser baixado do site[50] do criador dos *ttt-plots*. É necessário ter o *gnuplot* instalado em seu computador. A versão de 2018 do *script* tinha um problema de retrocompatibilidade com o *gnuplot*, então há uma versão corrigida na pasta *TTT plots* do GitHub.

[50] `http://mauricio.resende.info/tttplots/tttplots.zip`

No terminal, execute o *script* perl com o arquivo de entrada *instancia.dat*. Note que o nome do arquivo é o mesmo nome da instância que foi resolvida, porque este nome será utilizado no título do gráfico. Não se trata do arquivo original da instância. Adicionalmente, não utilize a extensão do arquivo na linha de comando.

No arquivo de entrada, deve haver um número por linha, correspondente ao tempo de execução ou número de iterações que o algoritmo levou para atingir a solução alvo dentro da margem de tolerância. Por exemplo, um método que foi rodado duas vezes, demorando 5 unidades de tempo na primeira execução e 3 unidades de tempo na segunda execução gera o arquivo com o valor 5 na primeira linha e o valor 3 na segunda linha, apenas.

```
1  perl tttplots.pl -f instancia
```

Após a execução do *script*, será gerada uma série de arquivos auxiliares e dois gráficos serão plotados: *instancia-qq.ps* e *input-exp.ps*. O segundo gráfico é o que nos interessa, e mostra duas distribuições sobrepostas: "empirical" e "theoretical". A primeira é a convergência observada do algoritmo e a segunda é a convergência teórica, exponencial.

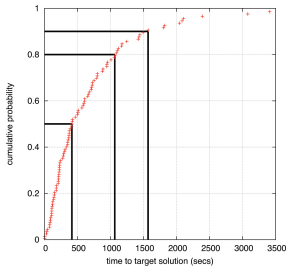


Figura 4: Interpretação de um ttt-*plot*.

A interpretação do *ttt-plot* é feita em relação à curva teórica e em relação ao eixo de probabilidade acumulada. Quanto mais próxima da curva teórica a convergência do algoritmo estiver, mais precisa é a análise da probabilidade acumulada, que indica qual a probabilidade do algoritmo convergir até o tempo indicado no eixo de tempo.

Por exemplo, na figura ao lado (sem a convergência teórica), é possível interpretar que com probabilidade 50% o algoritmo converge antes de 500 unidades de tempo, com probabilidade 80% o algoritmo converge com pouco mais de 1000 unidades de tempo e com probabi-

lidade 90% o algoritmo converge pouco depois de 1500 unidades de tempo,

Caso esteja comparando a convergência de dois métodos, utilize o *script tttplots-compare*[51] com um arquivo separado para cada método. O resultado é apenas a probabilidade de que a convergência do primeiro método seja menor ou igual à convergência do segundo método.

[51] `http://www2.ic.uff.br/~celso/tttplots/index.html`

```
1  perl tttplots-compare.pl -f inputfilename1 inputfilename2
```

Multiple Time-To-Target Plots

Os *Multiple Time-To-Target Plots* (*mttt-plots*) são uma extensão dos *ttp-plots*. Ao invés de realizar a análise a respeito de uma única instância para um problema de otimização, os *mttt-plots* analisam várias instâncias. Ainda não há *script* :(

Testes Estatísticos

É de suma importância analisar e intepretar análises estatísticas de algoritmos. A seguir são apresentadas algumas definições [52] e elencados testes importantes. Todos os testes a seguir podem ser encontrados no *R*[53], e retornam o *p-value*.

[52] Boa parte retirada de `https://goo.gl/DwLZXW`

[53] Veja a seção *Softwares*.

Em uma pesquisa correlacional o pesquisador não influencia nenhuma variável, mas apenas as mede e procura por relações (correlações) entre elas, como pressão sangüínea e nível de colesterol. Duas ou mais variáveis quaisquer estão relacionadas se em uma amostra de observações os valores dessas variáveis são distribuídos de forma consistente. Em outras palavras, as variáveis estão relacionadas se seus valores correspondem sistematicamente uns aos outros para aquela amostra de observações.

Suponha dois métodos sendo comparados: o seu método (MeuMétodo) e um método da literatura (MétodoLiteratura). Inicialmente, os dados devem ser testados quanto ao tipo distribuição, se é normal ou não. Caso seja distribuído normalmente, utiliza-se um teste paramétrico para comparar dois algoritmos. Caso contrário, a distribuição não é normal e um teste não-paramétrico deve ser utilizado para comparar dois algoritmos. Para comparar mais de um algoritmo ao mesmo tempo, é necessário outro tipo de teste. O fluxograma ao lado ajuda a escolher os testes a serem realizados com base no tipo de distribuição.

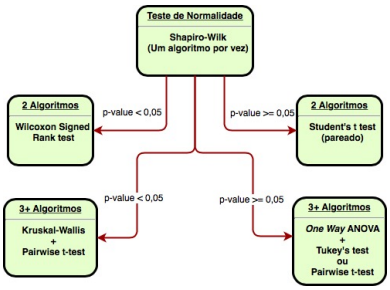


Figura 5: Fluxograma de escolha de testes estatísticos.

Como reportar (1): "A statistical analysis was performed to further compare the proposed MeuMétodo and the MétodoLiteratura method. The average values presented in the previous tables were considered,

as individual solutions for each run of the CS+BRKGA are not available.".

Como reportar (2): "A statistical analysis was performed to further compare the proposed MeuMétodo and the MétodoLiteratura method. The individual solution values for each instance in each run were considered.". Note que se houver os resultados para cada instância, melhor será.

Nível de Significância Estatística

Considere que já tenha sido calculada uma medida da relação entre duas variáveis. A próxima questão é "quão significante é esta relação?"[54]. Especificamente, a significância depende principalmente do tamanho da amostra. Em amostras muito grandes mesmo relações muito pequenas entre variáveis serão significantes, enquanto que em amostras muito pequenas mesmo relações muito grandes não poderão ser consideradas significantes. Assim, para determinar o nível de significância estatística torna-se necessária uma função que represente o relacionamento entre "magnitude" e "significância" das relações entre duas variáveis, dependendo do tamanho da amostra. Tal função diria exatamente "quão provável é obter uma relação de dada magnitude (ou maior) de uma amostra de dado tamanho, assumindo que não há tal relação entre aquelas variáveis na população". Em outras palavras, aquela função forneceria o nível de significância (*p-value*), e isso permitiria conhecer a probabilidade de erro envolvida em rejeitar a idéia de que a relação em questão não existe na população. Esta hipótese alternativa (de que não há relação na população) é usualmente chamada de **hipótese nula**. As funções de probabilidade são complexas e variam, porém, em muitos casos estas são relacionadas a um tipo geral de função que é chamada de **normal**. Por padrão, utiliza-se o nível de significância com valor de 95%.

[54] Por exemplo, 40% da variação global ser explicada pela relação entre duas variáveis é suficiente para considerar a relação significante?

Distribuição Normal

Uma propriedade característica da distribuição normal é que 68% de todas as suas observações caem dentro de um intervalo de 1 desvio padrão da média, um intervalo de 2 desvios padrões inclui 95% dos valores, e 99% das observações caem dentro de um intervalo de 3 desvios padrões da média. Em outras palavras, em uma distribuição normal as observações que tem um valor padronizado de menos do que -2 ou mais do que +2 tem uma freqüência relativa de 5% ou menos (valor padronizado significa que um valor é expresso em termos de sua diferença em relação à média, dividida pelo desvio padrão).

p-value

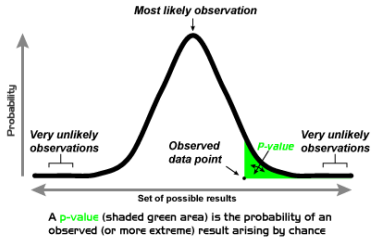


Figura 6: Interpretação visual do *p-value*.

A significância estatística de um resultado é uma medida estimada do grau em que este resultado é "verdadeiro" (no sentido de que seja realmente o que ocorre na população, ou seja no sentido de "representatividade da população"). Mais tecnicamente, o *p-value* representa um índice decrescente da confiabilidade de um resultado. Quanto mais alto o *p-value*, menos se pode acreditar que a relação observada entre as variáveis na amostra é um indicador confiável da relação entre as respectivas variáveis na população. Especificamente, o *p-value* representa a probabilidade de erro envolvida em aceitar o resultado observado como válido, isto é, como "representativo da população". Por exemplo, um *p-value* de 0,05 indica que há 5% de probabilidade de que a relação entre as variáveis, encontrada na amostra, seja um "acaso feliz". Por padrão, um *p-value* de 0,05 é costumeiramente tratado como um "limite aceitável" de erro. Resultados com um *p-value* 0,01 são comumente considerados estatisticamente significantes, e com *p-value* 0,005 ou *p-value* 0,001 são freqüentemente chamados "altamente" significantes. Estas classificações, porém, são convenções arbitrárias e apenas informalmente baseadas em experiência geral de pesquisa.

Shapiro-Wilk Test

O *Shapiro-Wilk* é um teste de normalidade para populações de tamanho até 30 (ou 50, para alguns autores). A hipótese nula é a de que a população é distribuída normalmente. Testes de normalidade são utilizados como preliminar para realização de testes não-paramétricos, que pressupõe normalidade dos dados. Caso a hipótese de normalidade seja rejeitada, é necessária a realização de testes não-paramétricos para comparação dos algoritmos. Caso a hipótese de normalidade seja aceita, é necessária a realização de testes paramétricos para comparação dos algoritmos.

No *R*, crie uma série de dados *MeuMetodo*, atribua os valores e rode o teste.

```
1    > MeuMetodo <- c(1.0, 2.0, 3.0, 4.0)
2    > shapiro.test(MeuMetodo)
```

Se o *p-value* retornado for menor do que um valor crítico dado pelo nível de significância α (normalmente 0,05), então o pressuposto de normalidade é rejeitada no nível de significância α, dado que há evidência de que os dados testados não pertencem a uma população normalmente distribuída. É retornado um valor *W* e o *p-value*. Se o valor de *W* não for pequeno, pode se concluir favoravelmente a respeito da normalidade dos dados. Embora seja reportado, o valor

de *W* não é interpretado.

Como reportar: "We applied the Shapiro-Wilk (Shapiro and Wilk, 1965) normality test, which rejected the null hypothesis stating that the method results could be modeled according to a normal distribution (W = XXX and p = X.0e - X)".

Wilcoxon Signed Rank Test

O *Wilcoxon Signed Rank Test* é um método não-paramétrico para comparação de duas amostras (no nosso caso, pareadas), no sentido de verificar se existem diferenças significativas entre os seus resultados. Por ser não-paramétrico, as distribuições testadas não devem ser normais (hipótese rejeitada, por exemplo, pelo teste de *Shapiro-Wilk*).

No *R*, crie uma série de dados para cada algoritmo (*MeuMetodo* e MétodoLiteratura), atribua os valores de cada um e rode o teste. Note que o parâmetro *less* testa se a mediana do primeiro método é menor do que a do segundo. Em outras palavras, está sendo testado se os dados são significativamente diferentes e se os dados do seu algoritmo são melhores do que os dados do outro algoritmo.

```
1  > MeuMetodo <- c(214, 159, 13, 356, 789, 123)
2  > MetodoLiteratura <- c(159, 135, 123, 543, 12, 345)
3  > wilcox.test(MeuMetodo, MetodoLiteratura, paired=TRUE,
       alternative = "less")
```

Se o *p-value* retornado for menor do que um valor crítico dado pelo nível de significância α (normalmente 0,05), então o pressuposto de inexistência de diferença significativa é rejeitada no nível de significância α, dado que há evidência de que os dados testados diferem. É retornado um valor *V* e o *p-value*. O valor de *V* corresponde à soma dos *ranks* positivos associadas às diferenças com sinal positivo, ou seja, valores de *V* baixos indicam que o primeiro algoritmo é melhor. Embora seja reportado, o valor de *V* não é interpretado.

Como reportar (1): "We applied a Wilcoxon signed-rank test (WSR) (Rey and Neuhäuser, 2011) a non-parametric statistical hypothesis test. This test is used to compare the two sets of solutions to investigate if a significant difference exists between the solutions of MétodoLiteratura and MeuMétodo. The WSR indicated that the MeuMétodo ranks were statistically less significant than the MétodoLiteratura ranks (Z = XXX, p < X.0e - XX). This result suggests that the use of MeuMétodo on this set of instances improves the results found in comparison with MétodoLiteratura.".

Como reportar (2): "We applied the nonparametric Wilcoxon signed-rank test (Rey and Neuhäuser, 2011) in order to investigate whether there is a significant difference between the solution values generated by the MétodoLiteratura and those generated by the

MeuMétodo. The test indicated that there is statistically significant evidence of a difference in the methods' results (V = XX, and p = 0.XXX for a significance level of 0.05)."

Student's t-test

O *Student's t-test* é um método utilizado, entre outras coisas, para comparar duas amostras (no nosso caso, pareadas), no sentido de verificar se existem diferenças significativas entre os seus resultados. As distribuições testadas devem ser normais (hipótese verificada, por exemplo, pelo teste de *Shapiro-Wilk*).

No *R*, crie uma série de dados para cada algoritmo (*MeuMetodo* e MétodoLiteratura), atribua os valores de cada um e rode o teste. Note que o parâmetro *less* testa se a mediana do primeiro método é menor do que a do segundo. Em outras palavras, está sendo testado se os dados são significativamente diferentes e se os dados do seu algoritmo são melhores do que os dados do outro algoritmo.

```
1  > MeuMetodo <- c(214, 159, 13, 356, 789, 123)
2  > MetodoLiteratura <- c(159, 135, 123, 543, 12, 345)
3  > t.test(MeuMetodo, MetodoLiteratura, paired=TRUE, alternative =
       "less")
```

ANOVA

A *Análise de Variância* (ANOVA) é uma coleção de modelos estatísticos que visa, fundalmentamente, verificar se há diferença significativa entre as médias de três ou mais diferentes populações ao mesmo tempo, ou seja, é um teste de comparações múltiplas. Este é um teste paramétrico, portanto, as populações devem possuir distribuição normal.

A hipótese nula é a de que as médias populacionais são iguais, e a hipótese alternativa é a de que pelo menos um das médias é diferente das demais. Note que isso não quer dizer que todas as médias são diferentes. Adicionalmente, o teste não indica qual população possui média diferente.

Quando se trata de comparação de algoritmos, há apenas uma variável sob análise, portanto, o teste é *one-way*.

No *R*, crie uma série de dados *resultados*, atribua os resultados de cada método a esta série e rode o teste. No exemplo a seguir, há cinco métodos (de *A* a *E*) e os resultados para sete instâncias. Desta forma, os sete primeiros valores são referentes ao método *A*, os sete próximos são referentes ao método *B* e assim por diante. O segundo comando dá nomes aos métodos comparados e agrupa os resultados de sete em sete. O quarto comando imprime a associação entre re-

sultados e métodos. O quinto comando imprime um *boxplot*, apenas para comparação visual. O último comando imprime o resultado do teste.

Caso os seus dados sejam em número diferente, é necessário alterar os dados relativos ao primeiro e segundo comandos. Abaixo há um exemplo fictício da comparação simultânea de quatro algoritmos.

```
1  > resultados = c(33.08, 60.83, 66.47, 69.35, 70.59, 72.90, 64.37,
       38.69, 59.57, 62.65, 64.23, 65.61, 66.17, 60.23, 38.64,
      60.33, 64.31, 66.23, 71.54, 71.91, 63.67, 33.66, 56.57,
      59.48, 66.37, 73.35, 68.08, 60.22, 39.20, 60.35, 67.55,
      70.93, 79.60, 84.33, 70.15)
2  > metodos = c(rep("A", 7), rep("B", 7), rep("C", 7), rep("D", 7),
       rep("E", 7))
3  > comparacao = data.frame(resultados, metodos)
4  > comparacao
5  > boxplot(resultados ~ metodos, data=comparacao)
6  > teste = aov(resultados ~ metodos, data=comparacao)
7  > summary (teste)
```

Entre os dados retornados estarão duas colunas *F value* e *Pr*(>*F*). A última é correspondente ao *p-value*. Se o *p-value* for menor do que nível de significância α (normalmente 0,05) é possível concluir que há diferença significativa entre as populações. Caso contrário, não é possível afirmar que há diferença significativa entre os resultados dos métodos. Caso haja diferença significativa, realize um *Tukey's test* para descobrir onde está a diferença.

Tukey's Test

O *Tukey's Test* é um teste de comparações múltiplas, utilizado em conjunto com o ANOVA. O exemplo abaixo utiliza os mesmos dados da seção anterior.

```
1  > teste = aov(resultados ~ metodos, data=comparacao)
2  > TukeyHSD(teste, conf.level = 0.95)
```

Como resultado é apresentada uma tabela cuja coluna *p adj* corresponde ao *p-value* de cada comparação par a par. Desta maneira, *p-value* < 0,05 indica que há diferença significativa entre os resultados do par correspondente à linha da tabela.

Para reportar, é possível gerar um gráfico com as diferenças entre todos os métodos.

```
1  > teste = aov(resultados ~ metodos, data=comparacao)
2  > grafico = TukeyHSD(teste, conf.level = 0.95)
3  > plot (grafico)
```

Kruskal-Wallis Test

O *Kruskal-Wallis Test* é um teste de comparações múltiplas de populações não consideradas normalmente distribuídas. A hipótese nula é de que as populações são idênticas. O exemplo abaixo utiliza os mesmos dados da seção anterior.

```
1  > kruskal.test(resultados ~ metodos, data=comparacao)
```

Como resultado é reportado o *p-value*. Se o *p-value* for menor do que nível de significância α (normalmente 0,05) é possível concluir que há diferença significativa entre as populações. Caso contrário, não é possível afirmar que há diferença significativa entre os resultados dos métodos. Caso haja diferença significativa, realize um *pairwise Wilcoxon test* para descobrir onde está a diferença. O *Kruskal-Wallis Test* seguido do *pairwise Wilcoxon test* é exemplificado abaixo.

```
1  > kruskal.test(resultados ~ metodos, data=comparacao)
2  > pairwise.wilcox.test(resultados, metodos, p.adjust="bonferroni"
       )
```

Computadores para Experimentos Extensivos

Temos à disposição alguns computadores para que os experimentos computacionais longos não sejam rodados nos computadores pessoais dos alunos. Normalmente cada computador é utilizado por apenas um aluno. Caso não haja algum computador designado a você ainda, entre em contato. Os computadores são acessados via SSH.

Para os não familiarizados com o SSH, para acessar uma máquina remotamente via terminal basta executar o comando *usuario@ip*, em que *usuario* é um nome de usuário válido na máquina e *ip* é o ip da máquina. Depois, é necessário informar a senha do usuário e então o terminal se transforma no terminal remoto. Para encerrar a sessão e realizar *logout*, basta o comando *exit*.

Para que algum comando continue executando mesmo após o encerramento da sessão, utilize o *nohup*. No terminal remoto: nohup ./executavel &. Redirecione sua saída para algum arquivo.

Elaboração de Textos

É IMPRESCINDÍVEL FLUÊNCIA EM PORTUGUÊS OU INGLÊS, usar LaTeX [55] e Overleaf (ou outro semelhante).

Verifique o tópico do texto a ser escrito (relatório IC, artigo, monografia ou dissertação) e os arquivos de exemplo para se inspirar

[55] Veja material introdutório em `http://posgraduando.com/introducao-ao-latex-os-primeiros-passos/`. No mesmo site há tutoriais para criar figuras, tabelas e referências bibliográficas.

nas pastas *Exemplos de textos academicos* e *Exemplos de apresentacoes*. Há também guias sobre escrita em PO na pasta *Guidelines*.

Alunos de IC também devem submeter um resumo ao ~~Encontro~~ Encontro de Saberes. Não há modelo ou exemplo. Dada a alta qualificação das revisões realizadas, só posso divulgar as técnicas ancestrais de escrita destes resumos pessoalmente, em um ritual.

Atente para a linguagem utilizada nos arquivos de exemplo, o comprimento das frases (não podem ser muito longas) e parágrafos (não podem ser nem curtos nem longos) e o uso de vírgula [56]. Em português, mantenha o tom formal e impessoal[57]. Utilize verbos na voz ativa [58] e frases em ordem direta [59]. Não utilize superlativos [60] e use adjetivos e advérbios com cautela.

Não procrastine ao escrever, este processo normalmente já dura mais do que se imagina e pode apertar seu cronograma no final.

[56] Escreva frases curtas e simples. Uma frase com muitas vírgulas deve ser substituída por várias frases separadas por ponto

[57] Use a terceira pessoa do singular, não se usam a primeira e a segunda pessoas nas orações em textos acadêmicos.

[58] e.g., sabe-se, entende-se, recomenda-se.

[59] sujeito + verbo + complementos e/ou adjuntos.

[60] "muito", "o mais", "extremamente", "-íssimo", "maior", "bastante", etc

[61] Lei no. 9610/98

[62] Cópia de frases ou parágrafos inteiros, ou levemente alterados.

[63] Reprodução de textos já publicados anteriormente sem a devida referência.

[64] `http://www.scanmyessay.com`

[65] `http://copyspider.com.br`

[66] `http://www.copiaecola.com.br/`

Ética e Plágio

Ética importa. A reprodução indevida de obra acadêmica implica em apropriação indevida no Brasil é crime[61]. Sempre referencie o conteúdo de outros trabalhos que utilizar, seguindo as normas da ABNT. Plágio parcial[62] e autoplágio[63] são tipos comuns.

Existem boas ferramentas livres para detecção de plágio, como *Viper*[64], *Copy Spider*[65] e *Copia e Cola*[66].

Referências e Senso Comum

Na introdução de textos acadêmicos é comum o erro de realizar afirmações sem embasamento suficiente, principalmente em contextos não técnicos. Afirmações que escapem do senso comum devem ser referenciadas, ou seja, atribuídas a algum autor qualificado.

Por exemplo:

"*Cada vez mais as indústrias têm investido na otimização de suas linhas de produção.*"

escapa do senso comum - baseado em quê podemos afirmar isto? Apenas *achismo*? É necessário encontrar os dados e publicações que sustentem nossas afirmações.

Por outro lado, o que é senso comum não precisa ser referenciado:

"*Do ponto de vista de produtividade, é de interesse das indústrias que as linhas de produção sejam otimizadas.*"

Um truque válido é converter afirmações sem citação em afirmações dentro do senso comum.

Estrutura dos Textos

UMA MESMA BASE CONCEITUAL pode ser utilizada para escrever relatórios de IC, artigos para congressos e periódicos, monografias de graduação e dissertações de mestrado,variando apenas no tamanho e profundidade dos capítulos/seções.

No caso de monografias e dissertações, a referência para o formatação é o Guia Para Normalização Bibliográfica de Trabalhos Acadêmicos [67], disponível no site da biblioteca da UFOP.

[67] http://www.sisbin.ufop.br/download/guia.pdf

A seções a seguir se referem à estrutura geral proposta na seção *Estrutura Geral dos Textos*. Verifique os exemplos nos arquivos de apoio[68].

[68] https://shorturl.at/esEHN

Note que existem diferentes estilos de escrita de textos acadêmicos e este guia reflete somente o meu em particular – você tem liberdade para adicionar seu próprio estilo ao texto. Certamente algum membro de banca vai sugerir alterações de estilo :)

Estrutura Relatório IC

Não possui limite de tamanho e possui menor rigor científico. A partir deste relatório o conteúdo de artigos e monografias pode ser lapidado. Contém todos os capítulos da estrutura geral.

Veja o modelo em LaTeX e os arquivos de exemplo[69].

[69] https://shorturl.at/esEHN

Estrutura Artigo

Possui limite de tamanho, normalmente 12[70] páginas, e possui maior rigor científico. É dividido em seções, ao invés de capítulos, que apesar de terem o mesmo conteúdo listado na estrutura geral, possuem nomes diferentes. Possui todos os capítulos da estrutura geral. Veja o modelo em LaTeX e os arquivos de exemplo[71].

[70] Ainda assim, prefiro escrever mais e depois "lapidar" o texto. P. S.: Um artigo não possui 3,48888957e75 páginas.

[71] https://shorturl.at/esEHN

Estrutura Monografia

Não possui limite de tamanho mas normalmente não passa de 40 páginas, incluindo os elementos pré-textuais.

A monografia 1, mais curta (não mais que 20 ou 25 páginas) contém os capítulos de 1 a 4, sendo desejável alguma coisa do capítulo 5. A monografia 1 possui obrigatoriamente o capítulo especial 5,5 que não entra na versão final da monografia. Possui conclusões e trabalhos futuros também,que são relacionados a monografia 2.

A monografia 2 contém os capítulos de 1 a 6. Veja o modelo em LaTeX e os arquivos de exemplo[72].

[72] https://shorturl.at/esEHN

Estrutura Dissertação

Não possui limite de tamanho mas normalmente não passa de 35 páginas na qualificação e 80 páginas na versão final, incluindo os elementos pré-textuais. A profundidade é muito maior em relação a uma monografia.

A qualificação, uma espécie de "dissertação 1", mais curta, contém os capítulos de 1 a 4 e completos e definitivos, além da versão inicial do capítulo 5, com menos dados. A qualificação possui o capítulo especial 5,5 que não entra na versão final da dissertação. Possui conclusões e trabalhos futuros também, que são sobre como a dissertação será terminada.

A versão final da dissertação possui todos os capítulos da estrutura geral.

Veja o modelo em LaTeX e os arquivos de exemplo[73].

[73] https://shorturl.at/esEHN

Estrutura Geral dos Textos

As seções a seguir indicam sugestões de conteúdo para cada capítulo de trabalhos acadêmicos. Elementos pré-textuais (índices, listas de figuras, tabelas, siglas, etc) não são mencionados, mas podem ser incluídos com facilidade pelo LaTeX.

Note que ao dividir o texto em capítulos e seções não se pode colocar dois títulos em sequência sem que haja um parágrafo de texto entre eles, por exemplo, o título do capítulo seguido pelo título da primeira seção. Entre os dois deve ser colocado um parágrafo introdutório do capítulo, que **não** deve possuir apenas um frase curta.

Resumo

O resumo consiste de um único parágrafo corrido e normalmente não ultrapassa 300 palavras. No resumo deve constar, por ordem:

- Contextualização do tema tratado;
- Definição sucinta do tema tratado, incluindo a dificuldade;
- Contribuições do seu trabalho;
- Resumo dos experimentos computacionais reportados: instâncias, métodos da literatura e resultados.

O resumo não pode ser uma cópia de frases da introdução, ou qualquer outra parte do texto.

Abstract

O *abstract* é a versão em inglês do resumo. O google tradutor não consegue fazer isto para você.

Capítulo 1 - Introdução

Contém a introdução propriamente dita:

- Texto inicial com a descrição superficial do problema;
- Motivação – porquê estudar este problema?
- Objetivos – geral e específicos;
- Contribuições esperadas;
- Organização do trabalho.

Capítulo 2 - Revisão da Literatura

Este capítulo é onde você deve traçar um panorama geral da pesquisa já realizada anteriormente e indicar onde o seu trabalho se encaixará. Utilize os artigos listados na revisão da literatura, embora alguns possam não ser incluídos [74]. Organize os trabalhos no texto em uma maneira lógica, seja cronológica, seja por tipo de abordagem (e.g., métodos exatos, heurístico, metaheurísticos, etc.).

Alguns trabalhos devem ficar muito claros ao lermos uma revisão da literatura:

- O artigo que define o problema original;
- O artigo que determina a complexidade computacional do problema;
- Artigos que propõem instâncias para o problema que serão utilizadas no seu trabalho;
- *Surveys*[75];
- Os artigos cujos métodos propostos compõem o estado da arte atual e que serão comparados ao seu nos experimentos.

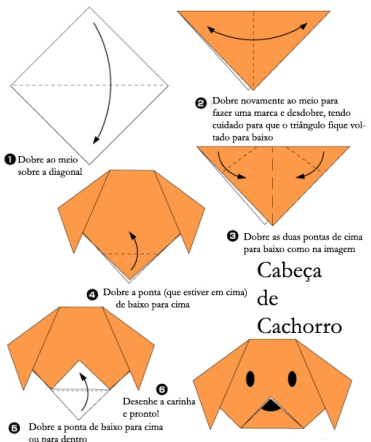
Capítulo 3 - Fundamentação Teórica

Contém a descrição mais pesada do problema e suas propriedades (principalmente matemáticas). Dê exemplos textuais e gráficos (faça o "chinês") de cada ponto interessante:

- O que é uma instância;

[74] Tome cuidado para não exagerar na quantidade de páginas.

[75] Artigos *surveys* são artigos que fazem a revisão bibliográfica de um problema/método específico e são muito interessantes para quem está fazendo uma revisão da literatura. Não é necessário citar cada um dos trabalhos relacionados pelo *survey*, mas sim, recomendar o *survey* como um todo.



- O que é uma solução para um problema;
- Quais são cada uma das restrições e como podem ser violadas;
- Como se calcula a função objetivo.

Eventualmente também podem ser descritos problemas correlatos ao que você trata, ou seja, parecidos ou encontrados no mesmo contexto. Dependendo do nível de profundidade do texto, pode haver uma seção inicial com conceitos básicos de otimização.

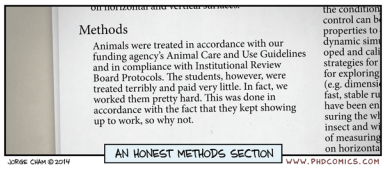
Também deve ser apresentada uma descrição mais pesada da modelagem, algoritmos ou técnicas da literatura, ou seja, que não são contribuições do seu trabalho. Estão inclusas heurísticas de busca local (como 2-*opt*, *swap*) e metaheurísticas[76] (*ILS*, *VND*, *BRKGA*, etc.). Apresente um breve histórico destes componentes, a referência original e o pseudo-código. Descreva sempre um componente por seção e de maneira genérica, sem relacioná-lo ao seu problema tratado.

[76] Veja boas descrições em: Talbi, El-Ghazali. Metaheuristics: from design to implementation. Vol. 74. John Wiley & Sons, 2009.

Capítulo 4 - Metodologia

Contém a descrição detalhada do algoritmo/método utilizado. Descreva brevemente em linhas gerais a categoria do algoritmo (se metaheurística) e depois entrar em detalhes da sua proposta de implementação específica.

É importante garantir a reprodutibilidade do método apresentado por você. Um leitor ao seguir a sua descrição conseguirá obter os mesmos resultados?



Dê exemplos textuais e gráficos (faça o "chinês") de cada etapa do algoritmo (solução inicial, busca local, operadores, etc.).

No caso específico do texto de **monografia 1**, pode não haver informação suficiente para este capítulo. Sendo assim, a metodologia deve conter a descrição do que se pretende na monografia 2, indicando:

- O que se propõe: um novo método, um estudo de um método conhecido, uma nova modelagem/formulação/aplicação, melhorar os resultados da literatura, etc.
- Métodos utilizados para comparação, indicando porquê cada método foi escolhido;
- Forma de comparação (ferramental estatístico a ser utilizado);
- Instâncias utilizadas (eventualmente descrever as características das instâncias com algumas estatísticas e como as mesmas foram geradas, caso as instâncias sejam artificiais);

Capítulo 5 - Experimentos Computacionais

Descreva o projeto dos experimentos:

- Ambiente computacional;
- Instâncias utilizadas (eventualmente descrever as características das instâncias com algumas estatísticas e como as mesmas foram geradas, caso as instâncias sejam artificiais);
- Métodos utilizados para comparação;
- Forma de comparação.

Se você utilizou o *iRace* para calibrar parâmetros, crie uma seção "Experimentos Preliminares" que descreva quais parâmetros foram calibrados e apresente uma tabela com os valores testados (basicamente o arquivo parameters.txt do *iRace*).

Para cada conjunto de instâncias, realizar as comparações, apresentar as estatísticas e gráficos e analisar por meio de comentários.

A maior parte dos tipos de dados analisados se aplica a todos os trabalhos, porém, pode resultar em tabelas grandes demais. Confira abaixo sugestões de colunas para as tabelas apresentadas no capítulo de experimentos computacionais:

- Dimensões das instâncias e outras características das instâncias (número de linhas e colunas de uma matriz, por exemplo);
- Número de instâncias (e);
- Solução ótima (OPT) ou melhor solução conhecida (MSC);
- Valor da solução inicial (S_0);
- Melhor solução encontrada por cada método (S^*);
- Solução média encontrada por cada método (S);
- Tempo médio de execução de cada método em segundos (T);
- *gap* (ou distância percentual) em relação ao melhor resultado conhecido;
- Desvio padrão das soluções para uma mesma instância(σ);

Adicione também uma última linha nas tabelas, com a média dos valores em cada coluna. Além disto, indique valores em negrito quando as melhores soluções foram atingidas pelos métodos propostos.

Capítulo 5,5 - Plano Para Atividades Restantes

Este é um capítulo temporário que serve para ligar a versão preliminar do texto (monografia 1 ou qualificação) e o texto final (monografia 2 ou dissertação final) e consiste em descrever o que ainda será feito para terminar o trabalho.

Deve haver um texto inicial e depois uma tabela indicando as atividades a serem realizadas e seu cronograma. São exemplos de atividades para planos de atividades restantes:

- Realização de experimentos computacionais extensivos, incluindo mais instâncias e métodos;
- Aprimoramento de implementações, incluindo métodos de *polishing* ou etapas adicionais;
- Novas implementações;
- Atualização/manutenção da revisão da bibliografia;
- Análises adicionais de métodos e instâncias.

Capítulo 6 - Conclusões e Trabalhos Futuros

Contém uma geral no que foi feito no trabalho de maneira breve:

- Problema tratado e sua importância;
- Metodologia utilizada;
- Objetivos atingidos (um apanhado em relação aos objetivos geral e específicos elencados na introdução);
- Resultados obtidos e críticas aos mesmos (o que funcionou e o que não funcionou).

Adicione um parágrafo final apontando futuras extensões para o trabalho realizado, seja por experimentar diferentes métodos, incorporar novas técnicas aos métodos existentes, tratar uma versão estendida do problema, etc.

A conclusão não pode ser uma cópia da introdução, ou qualquer outra parte do texto.

Referências

O BibTeX [77] cuida disto para você :)

[77] https://pt.wikipedia.org/wiki/BibTeX

Apêndices

Elemento opcional. São apêndices:

- Formulários e questionários aplicados ou o roteiro da entrevista;
- Planos de ensino e de aula, criados para a aplicação da metodologia proposta;
- Regulamentos e regras criados para a implantação do projeto-piloto.

Anexos

Elemento opcional. São anexos:

- Mapas e documentos cartográficos;
- Leis, estatutos e regulamentos que esclareçam as condições jurídicas da pesquisa;
- Textos e reportagens na íntegra.

Bibliotecas de Instâncias

ABAIXO SÃO LISTADOS alguns repositórios *online* de instâncias para diversos problemas de otimização.

- OR Library[78];
- ESICUP[79];
- DEIS[80];
- Constraint Modelling Challenge 2005[81];
- CsPLib[82];
- SCOOP[83];
- The House of Graphs[84];
- Matrix Market[85];
- Matrix Market (Harwell-Boeing)[86];
- OPTSICOM [87];
- TSPLIB [88];
- TSP[89];

[78] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

[79] https://www.euro-online.org/websites/esicup/data-sets/

[80] http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm

[81] https://ipg.host.cs.st-andrews.ac.uk/challenge/

[82] http://www.csplib.org

[83] http://www.scoop-project.net/view_doc.php?id_sez=8&view_type=1&ord=2

[84] http://hog.grinvin.org/

[85] http://math.nist.gov/MatrixMarket/

[86] http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/

[87] https://grafo.etsii.urjc.es/optsicom/

[88] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[89] http://www.math.uwaterloo.ca/tsp/data/

- DIMACS[90];
- Meu site[91];
- *Kaggle*[92];
- Loggi Benchmark for Urban Deliveries [93];
- Network Repository [94];
- BPPLIB [95];
- Bin Packing [96];
- ekhoda´s list of Optimization Problem Libraries [97];
- 2E-LRP [98];
-
- VRP-REP [99].

[90] http://dimacs.rutgers.edu/Challenges/
[91] http://www.decom.ufop.br/marco/pesquisa/problem-instances/
[92] https://www.kaggle.com/datasets
[93] https://github.com/loggi/loggibud
[94] https://networkrepository.com/index.php
[95] http://or.dei.unibo.it/library/bpplib
[96] https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.htm
[97] https://github.com/ekhoda/optimization_problem_libraries
[98] http://prodhonc.free.fr/Instances/instancesLRP2E_us.htm
[99] http://www.vrp-rep.org/variants.html

Softwares

Abaixo são listados os *softwares* mencionados neste documento e também outros alternativos. A maioria é livre para uso acadêmico e independente de sistema operacional.

- *iRace*: calibrador de parâmetros [100];
- *R*: suíte de estatística [101];
- *Visio*: editor de diagramas (*windows* apenas);
- *GraphViz*: editor de grafos[102];
- *yEd*: editor de diagramas[103];
- *LucidChart*: editor de diagramas[104];
- *GNUPlot*: editor de gráficos vetoriais[105];
- *Asana*: comunicação e gerenciamento de projetos/atividades[106];
- *Overleaf* : edição de textos LaTeX colaborativa [107];
- *Concorde*: *solver* para o problema do caixeiro viajante [108];
- *Google OR-Tools*: suíte de ferramentas para problemas de otimização[109];
- *CPLEX*: pacote de otimização, *solver*[110];
- *Gurobi*: pacote de otimização[111];
- *brkgaAPI*: API em C++ para algoritmos genéticos de chaves aleatórias viciadas [112].

[100] http://iridia.ulb.ac.be/irace/
[101] https://www.r-project.org
[102] http://www.graphviz.org
[103] https://www.yworks.com/products/yed
[104] https://www.lucidchart.com/
[105] http://www.gnuplot.info
[106] https://app.asana.com/
[107] https://www.overleaf.com/
[108] http://www.math.uwaterloo.ca/tsp/concorde/index.html
[109] https://developers.google.com/optimization/
[110] http://www.ibm.com/developerworks/br/downloads/ws/ilogcplex/
[111] https://www.gurobi.com
[112] http://mauricio.resende.info/src/brkgaAPI/

Guias e HowTo's

Concorde

O Concorde é um resolvedor para o clássico Problema do Caixeiro Viajante e alguns problemas de otimizaçãoo relacionados. No material de apoio há a pasta *Concorde* que contém um guia de instalaçãoo e uso do resolvedor Concorde[113] que apresenta como instalar e utilizar o Concorde em ambientes linux-like com os resolvedores CPLEX e QSopt. No texto, há *Minimal Working Examples* (MWEs) da utilizaçãoo do Concorde com os diferentes resolvedores.

[113] Contribuiçãoo valiosa do aluno Túlio Neme de Azevedo (tulioneme10@gmail.com).

Gurobi

No material de apoio há a pasta *Códigos/Gurobi*, em que há um MWE do uso do Gurobi 7.5.2 em C++. O MWE contém um modelo linear de variáveis binárias que exemplifica a criação de ambiente, modelo, função objetivo, variáveis e restrições. Também há a sugestão de um esqueleto de código e um padrão para nomear e acessar variáveis dentro de um modelo.

iRace

No material de apoio há a pasta *iRace* com uma apresentaçãoo de um minicurso introdutório sobre o *iRace* e também um MWE[114].

[114] Contribuição valiosa da Helen Costa, então aluna de doutorado do programa de pós-graduação do DECOM.

Contribua

FALTOU ALGO? Você tem idéias interessantes para adicionar ao documento? Entre em contato.

$$\mathcal{B} > \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} p_i$$