PCC170 - Projeto e Análise de Experimentos Computacionais

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é baseado no artigo

- Johnson, D. S. (2002). A theoretician's guide to the experimental analysis of algorithms. Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges, 59, 215-250.

Licença

Sugestão 5 - Utilize programas auto documentados

Salvar dados em arquivos e diretórios com nomes descritivos pode ajudar, assim como a construção de arquivos LEIA-ME, fornecendo mapas de diretórios e outras informações.

No entanto, dados que não podem ser interpretados com precisão após o fato são dados inúteis.

Portanto, é desejável ter arquivos de saída que contenham todas as informações que você gostaria de saber sobre o experimento que os gerou.

Sugestão 5 - Utilize programas auto documentados

Isso inclui não apenas métricas de desempenho, como tempo de execução e qualidade da solução, mas também:

- O nome (e versão) do algoritmo usado;
- A máquina na qual ele foi executado (e a data, para ajudar no caso de atualização da máquina);
- O nome da instância resolvida;
- As configurações de todos os parâmetros ajustáveis;
- Quaisquer medidas (tempos das operações, contagens, valores intermediários da solução) que você julgue interessantes posteriormente.

Princípio 5 - Utilize implementações razoavelmente eficientes

Este é surpreendentemente um princípio um tanto controverso.

A eficiência tem um custo no esforço de programação e há várias situações nas quais os pesquisadores argumentam que eles deveriam ter permissão para se contentar com menos.

A theoretician's guide to the experimental analysis of algorithms

Implicância 5 - Alegar tempo/habilidade de programação inadequados como desculpa

Em alguns casos, os pesquisadores simplesmente afirmam que suas implementações provavelmente seriam competitivas com as de algoritmos anteriores, se tivessem apenas tempo ou habilidade para usar os mesmos mecanismos de aceleração.

A theoretician's guide to the experimental analysis of algorithms

Princípio 6 - Garanta a reprodutibilidade

Como em todos os estudos científicos, uma parte essencial de um trabalho experimental é a reprodutibilidade dos resultados.

Mas o que significa "reprodutibilidade" no contexto da análise experimental de algoritmos?

No sentido estrito, significa que se você executasse o mesmo código nas mesmas instâncias na mesma combinação máquina/compilador/sistema operacional/carga do sistema, obteria o mesmo tempo de execução, contagem de operações, qualidade da solução (ou as mesmas médias, no caso de um algoritmo aleatório).

Princípio 6 - Garanta a reprodutibilidade

Ao reproduzir esse estudo, um pesquisador deve usar os mesmos métodos básicos, mas normalmente usará aparelhos diferentes, materiais semelhantes, mas distintos, e possivelmente diferentes técnicas de medição.

Diz-se que ele reproduziu os resultados originais se os dados obtidos forem consistentes com os dos experimentos originais e apoiarem as mesmas conclusões.

Dessa maneira, ele ajuda a confirmar que os resultados do experimento original (e as conclusões tiradas deles) são independentes dos detalhes precisos do próprio experimento.

Princípio 6 - Garanta a reprodutibilidade

Isso tem implicações tanto na forma como você realiza seus experimentos quanto na forma de relatá-los.

Seus experimentos precisam ser extensos o suficiente para garantir que suas conclusões sejam verdadeiras e não artefatos de sua configuração experimental (as máquinas, compiladores e geradores de números aleatórios que você usa, as instâncias específicas que você testa, etc.)

Ao reportar seus resultados, você deve descrever os algoritmos, instâncias de teste, ambiente computacional, resultados, etc. com detalhes suficientes para que um leitor possa, pelo menos em princípio, realizar experimentos semelhantes que levem às mesmas conclusões básicas.

Implicância 6 - O código que não bate com a descrição no artigo

Com demasiada frequência, descobri que o código fornecido por um autor não implementa com precisão o algoritmo descrito no artigo correspondente.

As diferenças podem variar de especificações de entrada incorretas a problemas mais sérios, como etapas ausentes ou adicionadas.

Normalmente, esse é um erro honesto, devido à baixa manutenção de registros por parte do autor, mas não é menos frustrante.

Implicância 6 - O código que não bate com a descrição no artigo

Obviamente, se as conclusões do trabalho original se referem ao desempenho de um código específico em vez do algoritmo que ele implementa, a reprodutibilidade exige que o autor forneça acesso ao seu código.

Da mesma forma, se estivermos lidando com um artigo sobre "corrida de cavalos", em que a principal conclusão é que o algoritmo/implementação A vence o algoritmo/implementação B em um determinado conjunto de instâncias, a reprodutibilidade exige que o autor forneça acesso ao conjunto de instâncias utilizado.

Ambos os casos, no entanto, são mais como testes de produtos do que ciência e não são nosso foco principal aqui.

Implicância 7 - Padrões de comparação irreprodutíveis

Suponha que você esteja avaliando experimentalmente algoritmos e não tenha restringido a atenção às instâncias para quais o valor ideal da solução é conhecido.

Você é confrontado com a questão de como medir a qualidade relativa de uma solução.

Algumas repostas possuem desvantagens de reprodutibilidade associadas, algumas delas fatais.

A theoretician's guide to the experimental analysis of algorithms

Implicância 8 - Utilizar tempo de execução como critério de parada

Se alguém usar uma máquina com um processador ou sistema operacional diferente, ou apenas uma implementação mais/menos eficiente do algoritmo na mesma máquina, é possível obter soluções com um nível de qualidade distintamente diferente.

Assim, definir um algoritmo dessa maneira não é aceitável para um artigo científico.

Implicância 8 - Utilizar tempo de execução como critério de parada

Então, o que se deve fazer se quiser fazer comparações "justas" reproduzíveis?

Uma solução é projetar seus códigos para que uma contagem combinatória prontamente mensurável (número de vizinhanças pesquisadas, número de etapas de *branching* etc.) seja usada como critério de parada

Implicância 9 - Utilizar a solução ótima como critério de parada

Os algoritmos recebem a instância e o valor ideal da solução (se conhecidos) e são interrompidos antecipadamente caso seja encontrada uma solução com o valor ótimo.

Os algoritmos geralmente têm um critério de parada de *backup* para instâncias com otimização desconhecida e para execuções que não conseguem encontrar uma otimização conhecida.

Portanto, os testes para instâncias com ótimos conhecidos não refletem o desempenho na prática, onde o benefício da parada precoce nunca será sentido.

A theoretician's guide to the experimental analysis of algorithms

Implicância 10 - Parâmetros ajustados manualmente

Muitas heurísticas têm parâmetros que precisam ser definidos antes que o algoritmo possa ser executado.

Por exemplo, em um algoritmo de otimização local de inicialização múltipla, é necessário especificar o número de partidas.

Metaheurísticas mais elaboradas, como recozimento simulado, busca tabu, algoritmos genéticos etc. podem vir com conjuntos inteiros de parâmetros ajustáveis.

Implicância 10 - Parâmetros ajustados manualmente

Para que o algoritmo seja bem definido, é necessário fixar esses parâmetros ou defini-los de uma maneira que dependa de aspectos mensuráveis da instância que está sendo tratada.

O ajuste manual significa que o algoritmo é mal especificado, pois o processo de configuração de parâmetros não é explicado em detalhes.

Como os tempos de execução relatados normalmente não incluem o tempo gasto na determinação dos parâmetros, isso leva a uma subestimação séria do tempo de computação necessário para aplicar o algoritmo a uma nova instância/classe de instâncias.

Implicância 10 - Parâmetros ajustados manualmente

Se configurações diferentes de parâmetros devem ser usadas para diferentes instâncias, o processo de ajuste deve ser bem definido e algorítmico.

O algoritmo de ajuste deve ser descrito no documento e o tempo para o ajuste deve ser incluído em todos os tempos de execução relatados.

Implicância 11 - O estudo de execução única

A menos que um estudo cubra uma ampla variedade de instâncias (e um número suficientemente grande de execuções no caso de algoritmos aleatórios), as conclusões tiradas podem muito bem estar erradas e, portanto, improdutivas por esse motivo.

A quantidade de dados necessários variará, é claro, com os detalhes das conclusões.

A theoretician's guide to the experimental analysis of algorithms

Implicância 12 - Utilizar o melhor resultado encontrado como critério de avaliação

A melhor solução encontrada é uma amostra da cauda de uma distribuição e, portanto, menos provável de ser reproduzível que a média.

Segundo, em tabelas, os tempos de execução relatados são geralmente para uma única execução do algoritmo, e não para todo o conjunto de execuções que produziu o "melhor" relatado.

Assim, o tempo para obter essa resposta é obscurecido. De fato, se, como geralmente ocorre, o número de execuções não for claramente indicado, não há como determinar o tempo de execução.

A theoretician's guide to the experimental analysis of algorithms

Princípio 7 - Garanta a comparabilidade

Este princípio é essencialmente o inverso do princípio de amarrar seu artigo à literatura.

O princípio anterior se referia à literatura passada. Este se refere ao futuro.

Você deve escrever seus artigos (e, na medida do possível, disponibilizar publicamente dados, códigos e instâncias relevantes de maneira a longo prazo) para que futuros pesquisadores (inclusive você) possam comparar com precisão os resultados dos novos algoritmos/instâncias aos seus resultados.

A theoretician's guide to the experimental analysis of algorithms

Aborrecimento favorio 13 - A máquina não especificada

Embora a maioria dos artigos mencione pelo menos a máquina na qual seus experimentos foram executados, eles geralmente omitem outros fatores importantes, como velocidade do processador, sistema operacional e linguagem/compilador.

Uma dificuldade que só piora com o passar do tempo e as máquinas ficam desatualizadas e esquecidas.

Leitura recomendada

- Johnson, D. S. (2002). *A theoretician's guide to the experimental analysis of algorithms*. Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges, 59, 215-250.

Dúvidas?