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

Biografia

David Stifler Johnson
⋆ 09/12/1945 † 08/03/2016

- Cientista da Computação;
- Especialista em algoritmos e otimização;
- ACM fellow em 1995;
- SIGACT Distinguished Service Prize em 1997;
- AT&T Fellow em 2005;
- SIAM Fellow em 2009;
- Knuth Prize em 2010;
- Membro da National Academy of Engineering em 2016;
- Co-autor de *Computers and Intractability: A Guide to the Theory of NP-Completeness* (1979);
- Até o ano de sua morte, suas obras haviam sido citadas mais de 96 mil vezes.

Diretrizes práticas para projetar um experimento

Artigo sobre análise experimental

A theoretician's guide to the experimental analysis of algorithms.

Conteúdo

- 6 *pitfalls* (armadilhas –"tentações e práticas que podem levar os experimentadores a perdas substanciais de tempo");
- 35 *pet peeves* (literalmente, "implicâncias" – práticas experimentais e de escrita comuns que me parecem particularmente mal orientadas); e
- 8 sugestões relacionados ao projeto e relato de experimentos computacionais.

A theoretician's guide to the experimental analysis of algorithms

Implementações

De muitas maneiras, a implementação de um algoritmo é a parte mais fácil da pesquisa.

A parte difícil é usar essa implementação com sucesso para produzir resultados de pesquisa significativos e valiosos (e publicáveis!).

Talvez haja quatro razões básicas pelas quais valha a pena implementar um algoritmo.

A theoretician's guide to the experimental analysis of algorithms

Razão 1

Para usar o código em uma aplicação específica.

Isso normalmente gera um artigo de aplicação, cujo objetivo é descrever o impacto do algoritmo nesse contexto.

Estamos principalmente interessados na saída do algoritmo, não em sua eficiência.

A theoretician's guide to the experimental analysis of algorithms

Razão 2

Fornecer evidências da superioridade de suas idéias algorítmicas.

Isso geralmente dá origem a artigos sobre corridas de cavalos, onde são publicados resultados para instâncias de *benchmark* padrão para ilustrar a conveniência desse algoritmo em comparação com concorrentes anteriores.

A theoretician's guide to the experimental analysis of algorithms

Razão 3

Para entender melhor os pontos fortes, fracos e operacionais de idéias algorítmicas interessantes na prática.

Essa é a motivação por trás de artigos de análise experimental.

A theoretician's guide to the experimental analysis of algorithms

Razão 4

Gerar conjecturas sobre o comportamento de caso médio de algoritmos sob distribuições de instâncias específicas em que a análise probabilística direta é muito difícil.

Isso leva a artigos experimentais de caso médio.

A theoretician's guide to the experimental analysis of algorithms

Princípios que devem guiar a escrita de artigos experimentais

- Realize experimentos interessantes;
- Amarre seu artigo à literatura;
- Utilize instâncias que suportem conclusões gerais;
- Utilize projetos experimentais eficientes e efetivos;
- Utilize implementações razoavelmente eficientes;
- Garanta a reprodutibilidade;
- Garanta a comparabilidade;
- Reporte a história completa;
- Crie conclusões bem justificadas e procure por explicações;
- Apresente seus dados de maneira informativa.

A theoretician's guide to the experimental analysis of algorithms

Realize experimentos interessantes

Esse princípio se aplica a todos os trabalhos científicos: os resultados devem ser novos e de interesse e/ou usados para uma audiência de leitores de tamanho razoável.

No entanto, os padrões de "interesse" são mais rigorosos para artigos de algoritmos experimentais.

A theoretician's guide to the experimental analysis of algorithms

Armadilha 1 - Lidando com algoritmos dominados

Suponha que você gaste muito tempo implementando e testando um algoritmo que acaba sendo dominado. Nem tudo pode estar perdido.

De fato, grande parte da literatura experimental é dedicada a algoritmos dominados.

Em muitos casos, no entanto, o fato de o algoritmo ter sido dominado não era conhecido no momento em que o artigo foi escrito (ou pelo menos não era conhecido pelo autor ou pelos revisores...).

Implicância 1

Autores e revisores que não fizeram o "dever de casa".

Referências relevantes foram citadas, mas claramente as partes relevantes destas referências não foram lidas.

A theoretician's guide to the experimental analysis of algorithms

Sugestões

1. Pense antes de computar.
2. Use experimentação exploratória para encontrar bons questionamentos.

A theoretician's guide to the experimental analysis of algorithms

Sugestões de questionamentos

- Como os detalhes da implementação, configurações de parâmetros, heurísticas e opções de estrutura de dados afetam o tempo de execução do algoritmo?
- Como o tempo de execução do algoritmo escala com o tamanho da instância e como isso depende da estrutura da instância?
- Como o tempo de execução do algoritmo se compara ao de seus principais concorrentes?
- Como essas comparações são afetadas pelo tamanho e estrutura da instância ou pela arquitetura da máquina, e as diferenças podem ser explicadas em termos de contagem de operações?

A theoretician's guide to the experimental analysis of algorithms

Sugestões de questionamentos

- Quais são as respostas para as perguntas anteriores quando "tempo de execução" é substituído por "qualidade da solução"?
- Qual operação melhor ajuda para explicar o tempo de execução?
- Na prática, quais são os gargalos computacionais do algoritmo e como eles dependem do tamanho e da estrutura da instância?
- Dada uma classe substancialmente nova de instâncias identificada, isso causa alterações significativas no comportamento dos algoritmos estudados anteriormente?

A theoretician's guide to the experimental analysis of algorithms

Princípio 2 - Amarre seu artigo à literatura

Um componente-chave no estabelecimento da novidade de um artigo é colocá-lo no contexto apropriado com relação à literatura sobre o problema que está sendo estudado.

De fato, antes de empreender qualquer projeto experimental, você deve fazer o possível para descobrir e estudar minuciosamente a literatura anterior, se ela existir.

A theoretician's guide to the experimental analysis of algorithms

Princípio 2 - Amarre seu artigo à literatura

Saber o que foi feito pode não apenas impede você de realizar experimentos "desinteressantes", mas também pode sugerir quais devem ser as perguntas interessantes.

Que comportamento precisa ser explicado? Quais algoritmos parecem abertos para melhorias adicionais? Que tipos de instâncias de teste ainda não foram estudadas adequadamente?

A theoretician's guide to the experimental analysis of algorithms

Princípio 2 - Amarre seu artigo à literatura

Idealmente, se seu objetivo é fornecer comparações com um algoritmo estudado anteriormente, você deve obter o código do algoritmo usado nos experimentos anteriores e relatar os resultados desse código em sua própria máquina (ao verificar que esses resultados são consistentes com os relatados anteriormente para o código).

Princípio 2 - Amarre seu artigo à literatura

A opção menos desejável, mas aquela à qual se deve recorrer ocasionalmente ao tentar fornecer comparações com um algoritmo difícil de implementar ou incompletamente especificado da literatura, é simplesmente comparar seus resultados em sua própria máquina e instâncias de teste com as relatadas no artigo anterior.

Normalmente, no entanto, você pode fornecer limites de precisão significativamente precisos em velocidades relativas da máquina para responder pelo menos à questão de saber se os tempos de execução são possivelmente competitivos ou se uma implementação é claramente mais rápida que a outra.

A theoretician's guide to the experimental analysis of algorithms

Princípio 2 - Amarre seu artigo à literatura

Mesmo essas comparações grosseiras são melhores que nada. Os leitores precisam ver como o seu artigo se encaixa na literatura, mesmo que o ajuste seja, na melhor das hipóteses, apenas aproximado.

A theoretician's guide to the experimental analysis of algorithms

Princípio 3 - Utilize instâncias que suportam conclusões gerais

Existem basicamente dois tipos de instâncias de teste disponíveis para os experimentadores: instâncias particulares de aplicações do mundo real e instâncias geradas aleatoriamente.

Observe que instâncias aleatórias não precisam ser "desestruturadas" (como grafos aleatórios e matrizes de distâncias aleatórias) e, de fato, devem preferencialmente ser estruturadas de maneira a refletir alguns aspectos das instâncias do mundo real.

A theoretician's guide to the experimental analysis of algorithms

Implicância 2 - Concentrar-se em instâncias aleatórias desestruturadas

Não apenas as instâncias aleatórias não estruturadas podem nos falar pouco sobre o desempenho no mundo real, mas também podem nos enganar ativamente quanto à dificuldade do problema.

Por exemplo, muitos trabalhos sobre algoritmos de otimização para o TSP assimétrico concentram-se em matrizes de distância com entradas escolhidas independentemente e aleatoriamente em pequenos intervalos como $\{1, 2, \ldots, n\}$, em que n é o número de cidades.

A theoretician's guide to the experimental analysis of algorithms

Implicância 2 - Concentrar-se em instâncias aleatórias desestruturadas

Essas instâncias são particularmente fáceis, pois, à medida que n aumenta, a duração ideal da rota provavelmente será igual ao limite inferior facilmente calculado do problema de atribuição.

Existem autores que orgulhosamente proclamam a capacidade de seus códigos de encontrar soluções ótimas para instâncias desse tipo com milhares de cidades, enquanto os mesmos códigos têm grande dificuldade com instâncias estruturadas do TSPLIB contendo 53 cidades ou menos.

A theoretician's guide to the experimental analysis of algorithms

Implicância 3 - O experimento de milissegundos

Vi mais de um artigo no qual o tempo máximo de execução relatado para qualquer um dos algoritmos testados em qualquer uma das instâncias estudadas é de um segundo ou menos.

Apesar disso, os trabalhos dedicam muito esforço para determinar qual algoritmo é o mais rápido.

Implicância 3 - O experimento de milissegundos

Eu diria que, na maioria das aplicações, se um algoritmo leva um segundo ou menos, o tempo de execução provavelmente é irrelevante, e um algoritmo que leva 0,01 segundos em sua máquina não oferece uma vantagem significativa sobre um que leva 0,1, mesmo que seja 10 vezes mais rápido.

Pode-se argumentar também que esse fator de 10 faria diferença para instâncias maiores, mas, nesse caso, deve-se testar essas instâncias maiores para confirmar que a vantagem persiste à medida que o tamanho da instância aumenta.

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 (ou estejam vinculados a) 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.

A theoretician's guide to the experimental analysis of algorithms

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.

A theoretician's guide to the experimental analysis of algorithms

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

A theoretician's guide to the experimental analysis of algorithms

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.

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?