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

1. Diretrizes práticas para projetar um experimento - Parte 3
   - A theoretician's guide to the experimental analysis of algorithms

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

A theoretician's guide to the experimental analysis of algorithms

Sugestão 6 - Use códigos *benchmark* para medir velocidades

O *benchmark* é distribuído como código-fonte portátil, que preferencialmente envolve estruturas de dados e operações algorítmicas um pouco semelhantes às usadas nos algoritmos estudados.

Você compila e executa o código de referência na mesma máquina e com o mesmo compilador usado para as implementações que está estudando, relatando seus tempos de execução para um conjunto especificado de instâncias de teste publicamente disponíveis, de tamanhos variados.

Futuros pesquisadores poderão calibrar suas próprias máquinas da mesma maneira e, com base nos dados de referência relatados no artigo original, tentar normalizar os resultados antigos para suas máquinas atuais.

A theoretician's guide to the experimental analysis of algorithms

Implicância 14 - As instâncias perdidas

Você deve fazer o seu melhor para utilizar instâncias às quais os pesquisadores subsequentes terão acesso.

Atualmente, a solução padrão é garantir que as instâncias (ou seus geradores) estejam disponíveis na Web.

Implicância 14 - As instâncias perdidas

Outra alternativa para resolver as duas irritações acima é simplesmente disponibilizar o código-fonte do seu algoritmo. Qual a melhor maneira de garantir a comparabilidade?

É claro que pode haver obstáculos para fazer o código disponível. Dependendo das políticas do seu empregador, pode ou não ser possível liberar seu código ao público.

Armadilha 6 - O código perdido

Se você perder seu código, perde sua melhor abordagem para obter comparabilidade entre ele e futuros algoritmos projetados por você e por outras pessoas.

Se você perder os dados nos quais os resumos relatados em seus artigos se baseiam, perde a chance de voltar e responder a perguntas mais detalhadas levantadas por leitores ou futuros artigos.

Então, por que alguém permitiria que isso acontecesse? Posso citar várias maneiras de uma experiência pessoal amarga, desde falta de versionamento a ausência de *backup*.

Princípio 8 - Conte a história completa

Se você é excessivamente seletivo, pode não conseguir chegar a conclusões precisas ou apoiá-las adequadamente.

Qualquer escala ou normalização de medições deve ser cuidadosamente explicada para que as médias brutas possam ser regeneradas, se desejado.

Uma parte importante diz respeito a resultados anômalos. Você não deve escondê-los, omitindo-os ou deixando de apontá-los quando estiverem presentes.

Princípio 8 - Conte a história completa

Por razões de honestidade intelectual, é claro que se deve incluir resultados que são "anômalos" apenas por serem inconsistentes com as conclusões que você deseja tirar (por exemplo, resultados para casos em que o código que você deseja defender tem desempenho inferior).

No entanto, deve-se também relatar qualquer fenômeno estranho (mas reprodutível) encontrado nos experimentos.

Idealmente, deve-se também fornecer explicações para tais anomalias, mas mesmo se você não tiver nenhuma, o leitor deve saber de sua existência.

Anomalias "inexplicáveis" na verdade podem eventualmente ser a chave para grandes *insights* sobre algoritmos, implementações ou dados de teste.

Implicância 16 - Anomalias desapercebidas

A pior anomalia é aquela que você não percebe.

Infelizmente, muitos artigos as contêm, deixando o leitor se perguntar se é uma anomalia verdadeira ou simplesmente um erro tipográfico (ou evidência de um *bug* de implementação).

Anomalias são importantes. Procure por elas.

Implicância 16 - O critério de parada ex post facto

Alguns autores não relatam o tempo total de execução, mas apenas o tempo até a melhor solução ser encontrada.

Na verdade, eles estão investigando um algoritmo clarividente que conhece antecipadamente o melhor valor de solução que jamais verá e, portanto, pára assim que uma solução com esse valor é encontrada.

Isso não é algo que podemos fazer no mundo real.

Implicância 16 - O critério de parada ex post facto

Uma abordagem melhor adotada por muitos autores é informar, além dos tempos de execução completos, o número de etapas/iterações realizadas antes que a melhor solução fosse encontrada.

Essas informações podem ajudar a determinar regras gerais sobre como definir o parâmetro que governa o número total de etapas/iterações a serem executadas.

A theoretician's guide to the experimental analysis of algorithms

Crie conclusões bem justificadas e procure por explicações

O objetivo de realizar experimentos com algoritmos é, presumivelmente, aprender algo sobre o desempenho deles, e assim um bom artigo experimental terá conclusões para declarar e apoiar.

Vale a pena enfatizar, pois, com base nos envios de conferências que eu já vi, muitos iniciantes aparentemente não percebem isso.

Implicância 19 - Dados sem interpretação

Não basta simplesmente executar testes, tabular seus resultados e deixar ao leitor o trabalho de encontrar as implicações dos dados.

Se não houver implicações que você possa encontrar ou expressar, então você fez os experimentos errados e ainda não deveria escrever um artigo, pois foi reprovado no teste de "novidade".

Implicância 19 - Dados sem interpretação

No mínimo, deve-se ser capaz de resumir padrões nos dados. Idealmente, também é possível propor conjecturas mais gerais que os dados suportam e que pesquisas subsequentes podem tentar confirmar ou negar.

Observe que a conclusão pode ter que ser "os dados são inconclusivos", mas se você configurou seu estudo para responder a perguntas interessantes, poderá obter uma resposta de algum tipo.

Você também precisa criar um argumento convincente de que seus dados realmente suportam as respostas que você declara.

Implicância 20 - Conclusões sem suporte

É surpreendente a frequência com que os artigos afirmam conclusões que não são bem suportadas pelos dados que apresentam, seja por causa de exceções gritantes que não comentam, de tendências que não percebem ou de alguma outra falha.

Como exemplo da segunda falha, considere um artigo que li recentemente que afirmava que dois algoritmos eram igualmente bons porque cada um era melhor na metade do tempo, mas não percebeu que um era melhor apenas nas instâncias menores do banco de ensaio e o outro foi melhor (aumentando as margens) à medida que o tamanho da instância aumentou.

Princípio 10 - Apresente seus dados de maneiras informativas

A melhor maneira de apoiar suas conclusões (e às vezes também a maneira mais fácil de derivá-las) é exibir seus dados de maneira a destacar as tendências que exibe, as distinções que faz e assim por diante.

Existem muitas técnicas de exibição boas, dependendo dos tipos de pontos que se deseja destacar.

A theoretician's guide to the experimental analysis of algorithms

Implicância 22 - Tabelas sem ../Figuras

Tabelas por si só são geralmente uma maneira muito ineficiente de contar sua história.

A maioria dos leitores (e participantes da conferências) odeia apresentações em que a linha de destaque é uma tabela grande com várias colunas em letras pequenas.

Se houver alguma maneira gráfica de resumir os dados e revelar sua mensagem, é quase sempre preferível a uma tabela sozinha.

Implicância 23 - ../Figuras sem tabelas

Por outro lado, embora as imagens possam contar sua história mais rapidamente, geralmente são uma maneira ruim de apresentar os detalhes de seus resultados.

Não quero ter que criar uma régua para estimar o valor da solução que seu algoritmo encontrou para que eu possa compará-la com os resultados do meu próprio algoritmo.

Portanto, um bom artigo deve conter ../Figuras e tabelas, embora fique à vontade para colocar as tabelas em um apêndice, para que os leitores que desejam apenas ver o quadro geral não precisem lidar com eles.

A theoretician's guide to the experimental analysis of algorithms

Implicância 23 - ../Figuras que produzem poucos *insights*

Embora as imagens possam adicionar muito ao nosso entendimento dos dados, nem todas as imagens são igualmente úteis.

É preciso ter cuidado ao descobrir o que exibir e como exibi-lo, e sem esse cuidado podemos obter números que nos dizem muito pouco.

Implicância 28/29 - Tabelas mal estruturadas

Nenhuma inferência é justificada quando os dados são ordenados lexicograficamente, a menos que se queira sugerir seriamente que o desempenho algorítmico seja influenciado pela classificação alfabética do nome de uma instância.

O perigo não é (como na discussão anterior) sugerir tendências falsas, mas obscurecer as verdadeiras.

A ordenação de instâncias por tamanho faz muito mais sentido do que ordená-las por nome, pois é possível identificar tendências mais prontamente no tempo ou na qualidade da solução que se correlacionam com o tamanho.

A theoretician's guide to the experimental analysis of algorithms

Implicância 32 - Comparar maçãs com laranjas

Meu principal exemplo desse problema é a tabela que apresenta tempos de execução para vários algoritmos em um determinado conjunto de instâncias, mas para cada algoritmo as entradas foram retiradas de um artigo anterior cujos experimentos foram realizados em uma máquina diferente (e muitas vezes muito mais lenta).

Mesmo quando a legenda ou o texto continha a ressalva de que máquinas diferentes estavam envolvidas, um leitor que folheava o artigo provavelmente seria seriamente enganado sobre os tempos de execução relativos dos algoritmos.

Os leitores que desejam uma comparação mais precisa ficam com a tarefa de normalizar os tempos de execução para si mesmos, fazendo disso outra ilustração do problema de fazer o leitor fazer a aritmética.

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?