ANÁLISE EXPERIMENTAL DE ALGORITMOS ONLINE

Aluno: Lucas Pereira Francisco de Azevedo
Orientador: Marcus Vinicius Soledade Poggi de Aragão

Introdução

A pesquisa em Algoritmos Online trata da resolução de problemas em que a informação sobre a instância a ser resolvida é obtida ao longo do tempo, ou seja, o antagônico dos algoritmos offline onde toda a instancia está disponível antes do início da resolução do problema, como em ordenações e buscas. Um exemplo clássico é o problema do k-servidores em um espaço métrico, i.e. dados n locais e as distâncias entre estes locais, os servidores (k em número) devem ser deslocados para atender os pedidos de serviço em locais específicos que chegam ao longo do tempo. Neste problema, deseja-se determinar a atribuição dos servidores de modo a minimizar o deslocamento total.

Existe hoje uma extensa literatura sobre a análise teórica dos algoritmos utilizados para resolver problemas online. Esta tem como foco determinar a competitividade dos algoritmos, i.e. a razão entre o valor da solução obtida pelo algoritmo e o valor da solução ótima considerando-se que fosse possível conhecer por completo a instância no início.

Entretanto, existe uma escassez de trabalhos com o intuito de avaliar experimentalmente a performance dos algoritmos online propostos na literatura. Para tal é necessário um trabalho de desenvolvimento de um ambiente onde seja possível simular categorias de instâncias dos problemas de interesse, assim como resolvê-las pelos diferentes algoritmos online propostos. É neste contexto que este trabalho está situado. Com os resultados da análise prática dos algoritmos, será possível aplicar tais algoritmos em contextos reais, de forma a otimizar processos que não se enquadram nos requisitos impostos nas instâncias teóricas dos problemas. Outra possibilidade é utilizar técnicas ainda pouco exploradas que podem ser úteis e, principalmente, viáveis num contexto prático, ao contrário do que se assume nos contextos teóricos. Com isso, é possível encontrar algoritmos melhores para problemas específicos.

A qualidade de um algoritmo online é geralmente avaliada usando uma análise de competitividade. A idéia da competitividade é comparar a saída gerada por um algoritmo online à saída produzida por um algoritmo offline ótimo. Um algoritmo offline ótimo é um algoritmo que conhece toda a entrada de dados e pode computar uma saída ótima. Quanto melhor um algoritmo online se aproximar da solução ótima, mais competitivo é o algoritmo.

No prosseguimento do projeto, que se justifica principalmente pelo foco da maioria dos trabalhos publicados ser a análise dos problemas a partir de hipóteses que não se adequam à realidade (por exemplo, todas as análises e algoritmos para o problema de Load Balancing desconsideram o tempo entre chegadas dos Jobs, assumindo que um novo job chega imediatamente após o job anterior; o que não é real, além do fato de que a análise de competitividade torna-se complicada mesmo para tempos constantes entre chegadas). Outros pontos considerados neste prosseguimento são o "look ahead" (i.e., definir uma tempo de espera para uma decisão) e a utilização de algoritmos de maior complexidade para definir a decisão (os utilizados e analisados são habitualmente de complexidade constante ou, no máximo, linear).

Objetivos

Os objetivos são desenvolver um ambiente que permita avaliar experimentalmente diferentes algoritmos (online) para um dado problema e fazer avaliações e comparações de algoritmos propostos na literatura para diferentes problemas. Este projeto consiste em, primeiramente, desenvolver este ambiente para um problema clássico simples como o dos k-servidores.

Problemas Tratados: *Load Balancing*

O problema de *load balancing* consiste em m máquinas iguais que devem realizar uma sequência J de tarefas. Cada tarefa j possui um tempo de execução próprio e deve ser atribuida a uma máquina assim que é criada, ou seja a decisão da máquina para a tarefa deve ser imediata. O objetivo final é terminar todas as tarefas no menor tempo possível.

Neste caso foi analizado o algoritmo Greedy, o qual consite em atribuir a tarefa que acaba de chegar à máquina menos carregada. Entretanto, mas a literatura têm poucos algoritmos eficientes para obter o ótimo offline para o problema de *load balance*. Por esse motivo, foi desenvolvido um algoritmo baseado no problema *bin packing*, que consiste em armazenar objetos de tamanhos variados no menor número de containers. No nosso caso, dever-se-ia achar um lower bound [2] para o tamanho dos containers de forma que se pudesse guardar todos os objetos (tarefas) numa quantidade fixa de containers (máquinas).

O processo de determinar o tamanho dos contêineres consiste primeiramente em encontrar o máximo entre a média do tempo das tarefas e o tempo da maior tarefa, ou seja, um limite inferior. O valor encontrado é então testado no bin packing. Caso as tarefas não "caibam" nas máquinas, um novo valor é gerado num esquema de busca binária, ou seja, o tamanho atual dos contêineres é dobrado e novamente testado no bin packing. No caso de as tarefas caberem nas máquinas, o que seria um limite superior, o valor é dividido pela metade. Esse processo segue até que seja determinado um limite justo.

Foi implementado um protótipo de um simulador que reproduz o load balance. O programa foi escrito em C++ e gera cinco seqüências pseudo-aleatórias de tarefas, testando-as no algoritmo implementado. As seqüências consistem de 100 tarefas com tempo de execução variando entre 30 e 60. As máquinas são simuladas numa configuração simétrica, ou seja, todas têm o mesmo poder de processamento, com 5 máquinas.

A competitividade observada foi 1,028-competitivo contra 1,8-competitivo teórica. A grande disparidade dos resultados pode ser explicada pela forma como é provada a competitividade teórica. O simulador testa instâncias randômicas e tendem a um comportamento de caso médio. Os resultados teóricos são baseados num estudo do pior caso. Um pior caso no load balance seria uma distribuição homogênea dos tempos das n-1 primeiras tarefas e uma n-ésima tarefa imensamente grande. Isso causaria desequilíbrio na distribuição das tarefas por parte do algoritmo greedy. O algoritmo ótimo offline nesse caso distribuiria as n-1 primeiras tarefas de forma que fosse reservado o máximo de espaço possível para a n-ésima tarefa, o que poderia até mesmo resultar numa máquina ociosa durante todo o tempo antes da chegada na n-ésima tarefa, fato perfeitamente razoável.

Instância	Offline	Greedy
A	887	912
B	910	933
C	900	923
D	910	933
E	938	953

Na abordagem que considera os tempos entre as chegadas das tarefas foram experimentados o mesmos algoritmos, e comparado com valor offline obtido como se os tempos entre as chegadas das tarefas não existissem. Claramente a competitividade obtida foi uma estimativa superior válida, mas que pode estar muito superior à competitividade real.

Esta dificuldade na estimativa da competitividade era esperada. Isto decorre inclusive da dificuldade de resolução do problema offline contemplando tempos não nulos entre as tarefas. Ilustrativamente, se usada a mesma ordem de atribuição de tarefas às máquinas calculada pelo algoritmo offline numa situação em que há tempo de chegada entre as tarefas, em algumas situações o algoritmo greedy apresentará competitividade menor do que 1, o que é absurdo.

Foram usadas as mesmas seqüências pseudo-aleatórias das simulaçõs anteriores, porém executando o algoritmo greedy em três novas situações. Na primeira situação, foi usado um tempo de chegada constante igual a 1 ciclo de execução. Na segunda situação, foram usados tempos de chegada numa distrubuição uniforme U, com tempos variando de 0 a 10. Na terceira situação, foi usada uma distribuição exponencial, obtida a partir da transformação da distribuição uniforme U. Para gerar números aleatórios com distribuição exponencial a partir de números aleatórios de distribuição uniforme, utilizou-se a equação (2) [4] obtida através da inversa da função de distribuição exponencial dada pela equação (1):

$$F_X(x) = 1 - e^{-\lambda x}, x \geq 0 \quad (1)$$

$$X = -\frac{\ell n(U)}{\lambda} \quad (2),$$

onde $\lambda > 0$ é a taxa de variação da exponencial. Em nossa simulação, foi utilizado $\lambda = 0{,}5$.

Instância	Constante	Uniforme	Exponencial
A	915	940	916
B	936	947	943
C	926	945	926
D	935	951	939
E	956	975	957

Metodologia

O processo de pesquisa se fundamentou no entendimento dos problemas tratados e na implementação de protótipos que permitissem apreciar resultados acerca da aplicação prática dos algoritmos propostos. Tais protótipos fazem uso das mesmas restrições, impostas nas demonstrações de competitividade e de limites, presentes na literatura.

Conclusões

A literatura em torno dos algoritmos online se baseia em instâncias bastante restritas e povoadas de suposições que simplificam o processo de demonstrar a competitividade e os upper e lower bounds dos algoritmos. Porém, isso limita a aplicação de tais algoritmos em situações mais reais, onde ainda existe pouco esforço de pesquisa. No caso da avaliação experimental de problemas com características mais próximas de problemas reais, fica ainda mais difícil prover estimativas precisas.

Referências

1 – ALBERS, S. Online algorithms: a survey. **Mathematical Programming**, v.97, 1-2, p.3-27. 2003.

2 – FEKETE, S. P., SCHEPERS, J. New classes of lower bounds for bin packing problems. **Mathematical Programming**, v.91. 2001.

3 – BANKS, J., CARSON II, J.S., NELSON, B.L., Discrete-Event Systems Simulation, Prentice Hall, Upper Saddle River, NJ, USA, 1999.

4 – GARCIA, A. L., Probability and Random processes for Electrical Engineering, Addison-Wesley Publishing Company, New York, NY, USA, 1989, 583 p.