Análise de algoritmos guloso e dinâmico para resolução de problemas de escalonamento job-shop

Ewerton Carlos de Araújo Assis

27 de Junho de 2013

Resumo

O presente trabalho, que comporá nota parcial na disciplina de Análise e Projeto de Algoritmos, ministrada pelo professor Leonardo Alves, no Instituto de Informática da Universidade Federal de Goiás, tem como objetivo apresentar soluções algorítmicas para problemas de escalonamento job-shop e a análise destas soluções em termos de sua complexidade computacional (complexidade de tempo). O presente trabalho não tem como finalidade correlacionar ou estabelecer a qualidade das soluções apresentadas — as quais foram obtidas em literatura especializada —, muito menos estabelecer a corretude das mesmas.

1 Introdução

Problemas de escalonamento job-shop, como serão apresentados e definidos a seguir, é uma classe de problemas numericamente intratáveis, NP-difícil [**?**], que têm, portanto, várias heurísticas e metaheurísticas para solucioná-lo. Dentre elas, são apresentadas duas heurísticas comumente utilizadas para solucionar problemas difíceis na computação: algoritmos gulosos e algoritmos dinâmicos. Como não é intenção do presente trabalho realizar uma análise abrangente de todos os algoritmos que utilizam essas abordagens, dois algoritmos são apresentados: um algoritmo baseado em programação dinâmica [**?**]; e um algoritmo baseado em estratégia gulosa, construído a partir de outras soluções para problemas de escalonamento. Ambas as estratégias serão definidas e apresentadas, com suas respectivas complexidades computacionais.

O presente trabalho divide-se, portanto, em: definição do que se trata um problema de escalonamento job-shop (seção **??**); como é constituída uma solução baseada em programação dinâmica (seção **??**); como é constituída uma solução baseada em estratégia gulosa (seção **??**); e, por fim, como estas duas abordagens influenciam positivamente para a obtenção de uma solução para um problema NP-difícil (seção **??**).

2 Definição do problema de escalonamento job-shop

De forma simples, um problema de escalonamento job-shop pode ser definido como: "dadas n tarefas (ou *jobs*) $\{J_1, J_2, ..., J_n\}$ a serem processadas por m máquinas $\{M_1, M_2, ..., M_m\}$, cada tarefa deve ser processada por cada máquina uma única vez. O processamanto de uma tarefa em uma máquina é denominado operação: a operação da tarefa i na máquina j é denotada por o_{ij}. Restrições tecnológicas determinam a ordem de processamento de cada tarefa através das m máquinas. As tarefas em um problema de escalonamento job-shop não compartilham uma mesma ordem de processamento através das máquinas, (...). O ambiente analisado em um problema de escalonamento job-shop é geralmente denotado por oficina (*workshop*).

"Cada tarefa i (J_i) apresenta um tempo de processamento para cada máquina, i.e. para cada operação o_{ij}, operação da tarefa i na máquina j, há um tempo de processamento p_{ij}. Qualquer tempo de ajuste ou de carregamento (*set-up time*) da máquina para processar a operação o_{ij} é incluído em p_{ij}. Além do tempo de processamento para cada operação, cada tarefa i (J_i) pode apresentar um tempo de lançamento, denotado por r_i, que determina a partir de quanto tempo ou em qual momento a tarefa J_i estará disponível para processamento pelas m máquinas da oficina.

"A partir dessas restrições e definições surge a necessidade de construir um escalonador de tarefas de forma a agendar as n tarefas através das m máquinas de forma que o tempo total de processamento seja o menor possível. (...).

"O problema de escalonamento job-shop é considerado um problema numericamente intratável, NP-difícil (para $m \geqslant 2$) [**?**] e que apresenta um limite superior de $(n!)^m$ soluções possíveis; portanto, objetivamente, a enumeração de todas as soluções possíveis para instâncias com dimensão relevante (por exemplo, 10×15 ou 10 tarefas, 15 máquinas) é impraticável [**?**]." (Definição integralmente apresentada em [**?**])

3 Solução baseada em programação dinâmica

Uma solução baseada em programação dinâmica pode ser definida rudemente como "solução recursiva apoiada por uma tabela de subsoluções". Uma instância do problema é subdividida em várias subinstâncias menores, que são resolvidas, e a partir da combinação das soluções menores, uma solução geral para a instância original é conquistada. Como é geralmente feito o uso de uma tabela para estabelecer a solução das subinstâncias e da instância original, a complexidade da solução relaciona-se ao tamanho da tabela construída.

Soluções em programação dinâmica são, a partir dessa constatação inicial, inerentemente recursivas: o problema deve ser modelado como um problema recursivo, no qual a solução de dada instância depende da solução de subinstâncias menores da instância original. Soluções baseadas em programação dinâmica têm como complexidade fórmulas de recorrência, dada essa característica recursiva.

Como o problema é modelado a partir de uma abordagem recursiva, com o fim de evitar que subinstâncias sejam resolvidas mais de uma vez, a solução baseada em programação dinâmica toma como apoio uma tabela, que armazena as soluções conquistadas para as subinstâncias tratadas nas iterações (recurções) do algoritmo.

3.1 Algoritmo dinâmico para problemas de escalonamento job-shop

O algoritmo dinâmico análisado para solucionar problemas de escalonamento job-shop tem por base o algoritmo apresentado em "Exponentially better than brute force: solving the job-shop scheduling problem optimally by dynamic programming.", que não foi copiado aqui dado a complexidade de definições que são apresentadas no artigo. Seja $p_{max} = \max_{ij} p_{ij}$, o algoritmo apresentado tem como complexidade $\mathcal{O}((2p_{max})^{n-1}(n\sqrt{n} + (2p_{max})^2)(m+1)^n n)$.

A ideia básica na construção da solução, que tem por objetivo criar uma agenda que tenha um tempo de execução mínimo, mas não ótimo, a partir das ideias de Held and Karp para problemas de sequenciamento, apresentada no artigo "*A dynamic programming approach to sequencing problems*". A ideia, então, é criar uma sequência de operações a partir de definições, teoremas e corolários feitos a partir da descrição do problema de escalonamento job-shop.

4 Solução baseada em estratégia gulosa

Uma solução baseada em estratégia gulosa parte como princípio um problema que foi modelado recursivamente: uma instância do problema é resolvida a partir da solução de subinstâncias da instância original. Assim, a cada iteração (ou recurção, propriamente dito) o algoritmo escolhe como subsolução aquela que for mais interessante em dada iteração — gula, nesse contexto, relaciona-se à ganância pela melhor solução em dado momento.

Diferentemente da abordagem baseada em programação dinâmica (seção **??**), que mantém um panorama geral das subinstâncias que já foram conquistadas e resolvidas e que influenciam na obtenção da solução da instância original, na abordagem gulosa não existe um panorama geral: decisões são tomadas em microcontextos; no caso, na iteração corrente. Assim, existe um comportamento cego e de "*(micro)*-exploração" no espaço de busca da solução — *exploitation* —, antagônico a "*(macro)*-exploração" — *exploration*.

4.1 Algoritmo guloso para problemas de escalonamento job-shop

A seguir é apresentada uma solução baseada em estratégia gulosa para problemas de escalonamento job-shop. A intenção do presente algoritmo é construir uma agenda que seja factível; contudo, nenhum compromisso com critérios de

otimalidade é estabelecido. Em seguida é feita uma análise de complexidade de tempo do algoritmo.

Algorithm 1: Estratégia gulosa: solução para escalonamento job-shop

Entrada: Instância de problema com n tarefas e m máquinas
Assuma: $\Omega : \{ (o_{ij}, p_{ij})$, t.q. o_{ij} é a operação da tarefa i na máquina j e p_{ij} é o tempo de processamento da tarefa i na máquina $j\}$
Saída: Sequência T, uma tupla com a ordem de operações a serem realizadas no *workshop*

```
begin
    while |Ω| > 0 do
        o' ← Retira (o_ij, p_ij) ∈ Ω, com p_ij mínimo do conjunto
        Adiciona o elemento o_ij da tupla o' na tupla T
    end
    return T
end
```

4.2 Análise do algoritmo guloso

Como o tamanho do conjunto $\Omega = n \times m$, temos que a complexidade desse algoritmo é $\mathcal{O}(nm)$. Nenhuma garantia quanto à otimalidade da solução é assumida; o único intuito é obter uma agenda de execução das tarefas que tem por base uma estretégia gulosa.

Se fosse aplicado o método de força bruta para obter uma solução ótima a partir da análise de todas as sequências criadas por esse algoritmo, essa solução teria complexidade de tempo na ordem de $\mathcal{O}(nm(n!^m))$, dado que o número de soluções possíveis para esse problema está na ordem de $n!^m$ soluções [**?**].

5 Conclusão

As soluções apresentadas têm como propósito elucidar como soluções algorítmicas baseadas em abordagem gulosa e em programação dinâmica devem orientar-se, a partir de um cenário-problema apresentado: problemas de escalonamento job-shop.

Ambas as abordagens e soluções apresentadas têm como propósito resolver um problema de escalonamento NP-difícil, no qual soluções triviais, por força bruta, não são interessantes. A solução gulosa, diferentemente da solução dinâmica, tem como propósito construir uma agenda que seja possível e interessante como solução, embora não exista qualquer compromisso de que a solução-agenda encontrada seja de fato ótima (a otimalidade da solução não é garantida nem intencionada). Já a solução dinâmica tenta inferir a partir de

subsoluções uma solução-agenda que seja interessante e relativamente próxima ao critério de otimalidade.

As abordagens apresentadas são interessantes em cada contexto analisado: se for interessante obter ao menos uma solução-agenda para o problema, que não tenha compromisso com otimalidade, a solução gulosa é apropriada nesse contexto. No entanto, quando uma solução ótima é almejada, uma solução baseada em programação dinâmica poderá ser obtida; ou mesmo poderá ser feito o uso de outras abordagens para obter uma solução [**?**].