MAC0325 – Otimização Combinatória

Escalonamento de Tarefas

Daniel Augusto Cortez

3 de dezembro de 2013

Resumo

Relatório descrevendo a implementação e os resultados do projeto da disciplina MAC0325 - Otimização Combinatória, IME-USP 2013 (Escalonamento de Tarefas).

1 Introdução

Neste projeto, estuda-se o problema de se escalonar um determinado número de tarefas a um conjunto de máquinas idênticas e paralelas. Sejam m o número de máquinas e n o número de tarefas. Seja d_i a duração da tarefa i. Cada tarefa deve ser processada por exatamente uma máquina, sem interrupição. Objetiva-se minimizar o tempo máximo de ocupação das máquinas, o chamado *makespan*. O problema é denotado na literiarura [1] por $Pm\,||\,C_{max}$.

Seja $x_{ij} \in \{0, 1\}$ a variável de decisão que assume valor 1 se a tarefa i for atribuída a máquina j e 0 caso contrário. O problema de minimização do *makespan* pode ser formulado como

$$\begin{aligned} \text{minimizar} \quad & C_{max} \\ \text{sujeito a} \quad & \sum_{i=1}^{n} x_{ij} d_i \leq C_{max}\,, \;\; 1 \leq j \leq m \\ & \sum_{j=1}^{m} x_{ij} = 1\,, \;\; 1 \leq i \leq n\,. \end{aligned} \tag{1.1}$$

$Pm\,||\,C_{max}$ pode ser reduzido ao problema da 3-partição sendo, portanto, de natureza $\mathcal{NP}$-difícil. Um método heurístico simples para resolução aporoximada do problema de escalonamento foi desenvolvido por Graham na década de 60 [2]. O método, chamado de *List Scheduling Algorithm*, consiste na atribuição da próxima

tarefa disponível a máquina de menor ocupação. É bem sabido [3] que esse algoritimo resulta em uma $(2 - 1/m)$-aproximação

A heurística de Graham pode ser melhorada fazendo-se uso de uma estratégia gulosa. Primeiro ordena-se as tarefas em ordem decrescente no valor de suas durações. Em seguida, faz-se a atribuição das tarefas nessa ordem a máquina de menor ocupação. Tal procedimento é conhecido pelo nome de *Longest Processing Time* (LPT) e resulta em uma $(4/3 - 1/3m)$-aproximação [3]. Tal razão de aproximação é justa. De fato, considere uma instância com 4 máquinas e 9 tarefas, cujas durações são mostradas na tabela abaixa:

tarefas	1	2	3	4	5	6	7	8	9
d_i	7	7	6	6	5	5	4	4	4

Escalonando as tarefas de acordo com a heurística LPT resulta em um *makespan* igual a 15. Entretanto, para esse conjunto de tarefas um escalonamento ótimo pode ser encontrado com $C_{max} = 12$.

Neste projeto a heurística LPT é estudada empiricamente, comparando seus resultados com valores exatos obtidos pela resolução de (1.1) pelo otimizador CPLEX. Os resultados indicam, para as instâncias consideradas, que a heurística se comporta de forma muito melhor do que o esperado pela análise de pior caso.

2 Implementação

Na implementação realizada, uma classe básica `Scheduler` foi projetada como interface para definir os métodos que as classes derivadas `LptScheduler` e `CplexScheduler` devem implementar. Tais métodos permitem a caracterização completa da solução gerada para uma instância do $Pm\,||\,C_{max}$ passada, como, por exemplo, o valor do *makespan* resultante e a impressão do escalonamento final. As classes `LptScheduler` e `CplexScheduler` implementam o procedimento de solução via heurística LPT e via otimzador CPLEX, respectivamente.

A comparação dos resultados é realizada através do cálculo da razão LPT/OPT entre os valores de *makespan* obtidos pelas duas metodologias. É claro que uma mesma instância do problema deve ser passada para as respectivas classes. Tal instância, na implementação, é representada por uma lista de máquinas e por uma lista de tarefas, podendo ser lida a partir de um arquivo de entrada padrão[1], ou ainda gerada aleatoriamente.

A classe responsável pela geração das instância aleatórias é `Generator`. Na instanciação do objeto associado, deve-se informa o número de máquinas e o número de tarefas pretendidas. O método `generate(min, max)` gera as tarefas com durações inteiras *uniformemente distribuídas* no intervalo fechado [min, max]. O intuito da

[1]Confira o arquivo README para obter informações sobre o arquivo de entrada e sobre as forma de utilização do programa.

geração aleatória das instâncias é estudar o comportamento da heurística no caso **médio**.

Na implementação da heurística LPT, as máquinas são inseridas em uma fila de prioridades (*heap*) que considera o valor de ocupação da máquina. Com isso, a operação de se obter a máquina de menor ocupação é realizada em $O(\log m)$, resultando em uma complexidade total para o algoritmo de $O(n \log m)$.

3 Experimentos Numéricos

A implementação realizada em C++ foi testada em uma máquina com processador 1.6 GHz Intel Core i5 com 4GB de RAM, rodando o sistema operacional MacOS 10.7.2.

Instâncias aleatórias foram geradas para diversos valores de m e n, medindo-se 5 vezes o valor da razão LPT/OPT entre os *makespans* das soluções encontradas. As durações das tarefas apresentam valores inteiros com distribuição uniforme no intervalo $[0, 1000]$. A Tabela 3.1 mostra os resultados médios obtidos e a Figura 3 traz a representação gráfica das médias com um intervalo de confiança de 67% (um desvio-padrão).

As piores instâncias estudadas em termos de tempo de processamento, curiosamente, não ocorreram com aquelas com maior número de variáveis. Na verdade, elas foram aquelas com $m = 16$ e $n = 40, 80$. Nesses casos, o algoritmo *branch-and-bound* utilizado pelo CPLEX teve duração limitada a 3 minutos, de forma que a solução apresentada não representa o valor ótimo, embora estivesse muito próxima dela (o processamento foi interrompido com gaps da ordem de 0.5%-1.0%).

	$m = 2$	$m = 4$	$m = 8$	$m = 16$
$n = 10$	1.00756	1.04135	1.00000	1.00000
$n = 20$	1.00198	1.01480	1.03009	1.00000
$n = 40$	1.00040	1.00550	1.02280	1.04043
$n = 80$	1.00030	1.00132	1.00560	1.02373
$n = 160$	1.00004	1.00031	1.00154	1.00514
$n = 320$	1.00002	1.00007	1.00028	1.00077

Tabela 3.1: Valores médios da razão LPT/OPT para diversas instâncias do problema $Pm\,||\,C_{max}$. As médias forma calculadas através de 5 instâncias geradas aleatoriamente para cada valor de m e n.

A análise dos experimentos mostra que o comportamento da heurística no caso médio é bem melhor do que o esperado pela análise de pior caso. De fato, a razão de aproximação $(4/3 - 1/3m)$, que para $m = 2$ corresponde a aproximadamente 1.167, não é verificada em nenhum experimento. Um cálculo simples a partir dos dados da Tabela 3.1 mostra que os valores da razão LPT/OPT obtidos estão, na média,

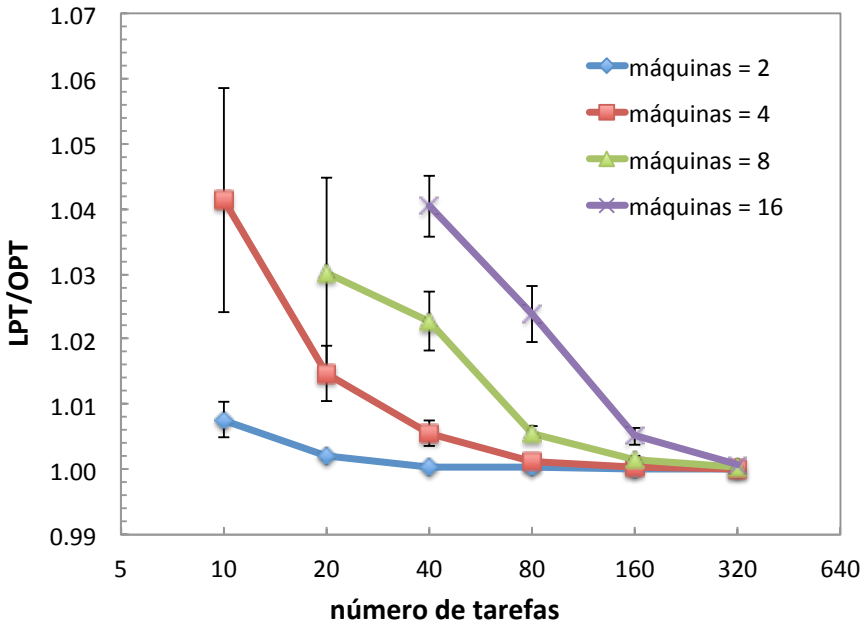

Figura 3.1: Representação gráfica dos valores médios da razão LPT/OPT apresentados na Tabela 3.1. Os intervalos de confiança correspondem a um desvio-padrão.

algo em torno de 80% menores do que a razão dada pela análise de pior caso, o que é um resultado muito bom.

Fica claro também, analisando o comportamento apresentado no gráfico da Figura 3, que a heurística se comporta melhor quanto maior o valor de n (número de tarefas) e quanto menor o valor de m (número de máquinas). Em particular, para valores pequenos de n, o intervalo de confiança é maior, indicando que o algoritmo não tem um comportamento tão regular para esse tipo de instância. Note ainda que o exemplo de pior caso dado na Seção 1 ocorre para n e m pequenos.

Em termos de tempo de processamento, as instância puderam ser otimzadas pelo CPLEX de forma rápida. Como mencionado, os piores casos ocorreram para $m = 16$ e $n = 40, 80$, onde limitou-se o tempo de solução em 3 minutos, permitindo ainda a obtenção de excelentes resultados.

Alguns testes com instâncias grandes também foram realizados para explorar os limites de processamento do otimizador. Uma instância com 100 máquinas e 1000 tarefas, resultando em um modelo com 10^5 variáveis binárias, foi processada pelo CPLEX durante 6 horas. A solução ótima não foi encontrada. Ao final da execução o gap da solução era de 0.62%, resultando em um valor da função objetivo 0.08% maior do que a fornecida pela heurística LPT (que rodou durante frações de segundo). O arquivo de saída para esse teste pode ser encontrado em `pior_caso.out`.

4 Conclusões

Neste projeto implementou-se, de forma eficiente, a heurística LPT para o problema do escalonamento de tarefas em máquinas paralelas idênticas e fez-se a comparação com os resultados obtidos pelo otimizador CPLEX que resolve o modelo de programação linear inteiro (1.1).

Os resultados obtidos indicam que a heurística se comporta muito bem no caso médio, com valores da razão LPT/OPT da ordem de 80% menores do que o limite dado pela análise de pior caso, chegando a soluções muito próximas da ótimo.

O trabalho foi muito válido pois além da análise do comportamento da heurística estudada, ele possibilitou a aplicação da API C++ do CPLEX para resolver um problema de programação linear inteiro. Ele ainda possibilitou que o limite de processamento dessa ferramenta fosse explorado.

Referências

[1] M. L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Third Edition, Springer.

[2] R. L. Graham. *Bounds on multiprocessing timing anomalies.* SIAM Journal of Applied Mathematics, 17: 416-429 (1969).

[3] D. P. Williamson, D. B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press.