Balanceamento de Carga na Paralelização da Meta-heurística GRASP

Adriana Cesário de Faria Alvim e *Celso Carneiro Ribeiro*

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225, Rio de Janeiro 22453-900
E-mail: {alvim, celso}@inf.puc-rio.br

Palavras-chave: Algoritmos paralelos, balanceamento de carga, meta-heurísticas, GRASP.

Resumo: Compara-se o desempenho de duas estratégias de balanceamento de carga na paralelização da meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*): decomposição *pré-escalonada* (balanceamento estático) ou *auto-escalonada* (balanceamento dinâmico) dos dados. As duas estratégias foram testadas sob a influência dos mesmos fatores externos. O balanceamento de carga dinâmico revelou-se como a melhor alternativa, mostrando significativas melhorias no desempenho da paralelização.

Abstract: We compare two load balancing strategies for the parallelization of the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic: pre-scheduled (static load balancing) or self-scheduling (dynamic load balancing) data partitioning. Both strategies were tested under the influence of the same external factors. Dynamic load balancing is shown to be the most suitable strategy, with significant improvements in terms of elapsed times.

1. Paralelização do GRASP

A meta-heurística GRASP [2] é uma técnica iterativa de amostragem aleatória para a resolução de problemas de otimização combinatória [7, 8]. Cada iteração consiste de duas fases: construção e busca local. Na primeira, uma solução viável é construída, cuja vizinhança será explorada na fase de busca local. A melhor solução entre todas as iterações é guardada como o resultado final. Devido à natureza iterativa do GRASP, o paralelismo de dados é a forma ideal para o seu particionamento. Para decompor os dados, a técnica escolhida deve ter como meta balancear a carga de trabalho e fazer com que a comunicação entre processadores seja mínima e regular. Como cada iteração requer aproximadamente a mesma quantidade de computação (carga de trabalho regular) e cada iteração é independente das demais, uma primeira estratégia consiste em fazer com que cada processador execute o mesmo número de iterações. A divisão de trabalho é determinada em tempo de compilação e, portanto, o balanceamento de carga é estático. Esta técnica é conhecida por *decomposição por bloco pré-escalonada* e sua implementação em um ambiente multi-processador com p processadores é direta. Cada processador deve receber uma cópia do procedimento, os dados de entrada e uma seqüência independente de números pseudo-aleatórios. As K iterações GRASP são divididas entre os p processadores. Cada um dos processadores executa, independentemente e em paralelo, K/p iterações. A melhor solução dentre as melhores encontradas pelos processadores é identificada como a solução encontrada pelo algoritmo. O custo de comunicação é baixo, uma vez que uma única variável global (a melhor solução) é necessária.

Considerando-se o tempo marcado pelo relógio como a medida de desempenho, esta estratégia pode não ser a mais eficiente em um ambiente com processadores não dedicados. Isto porque, embora a carga de trabalho assim distribuída seja uniforme, a carga computacional dos processadores envolvidos pode ser bastante heterogênea. A técnica de

decomposição de dados conhecida como *auto-escalonamento* (ou bolsa de tarefas) é uma alternativa para tratar o problema de desbalanceamento de carga. Um único processador mestre coordena o trabalho realizado por todos os processadores escravos. Determina-se que um certo número *bloco* de iterações representa uma tarefa e que existem *K/bloco* tarefas. Inicialmente, o mestre atribui uma tarefa para cada escravo. A medida em que os escravos terminam suas tarefas, comunicam tais eventos ao mestre e solicitam mais trabalho. Assim, todos os processadores permanecem ocupados, o tempo ocioso de cada processador é reduzido e o tempo necessário para completar o trabalho (as *K/bloco* tarefas) também.

Enquanto na primeira estratégia a carga de trabalho realizada por cada processador é sempre a mesma (exatamente *K/p* iterações), esta carga pode variar muito para os processadores escravos na segunda. O número de operações de comunicação também não é regular. No entanto, o tamanho da mensagem enviada por um processador escravo para comunicar o fim de uma tarefa ao mestre é mínimo, constituindo-se de um único dígito. Cada escravo só precisa enviar a melhor solução por ele encontrada uma única vez, quando o mestre comunicar que não existem mais tarefas a serem realizadas. A especificação detalhada da implementação destas duas estratégias e sua aplicação na solução de problemas de alocação ótima de tráfego em sistemas de satélites TDMA [7] encontra-se em Alvim [1].

2. Balanceamento estático versus dinâmico

Nesta seção, discute-se a metodologia empregada para comparar o desempenho das duas estratégias de balanceamento de carga da meta-heurística GRASP apresentadas anteriormente: estático e dinâmico. Como um dos principais objetivos do processamento paralelo é a redução do tempo real (*wall-clock*) ou do custo necessário para completar um trabalho, o tempo marcado pelo relógio será a medida de performance utilizada nos experimentos.

Um experimento computacional que compara o desempenho de dois algoritmos para uma mesma classe de problemas deve levar em consideração diferentes fatores que afetam os resultados dos testes. Mesmo controlando-se diversos fatores do ambiente, tais como *hardware*, *software* e os problemas-teste utilizados, a repetição de um mesmo experimento pode levar a resultados muito variados. Um fator difícil de ser controlado é a influência de outros programas que competem por recursos compartilhados, tais como os processadores, a rede e leitura/gravação de dados. Uma forma de reduzir a variabilidade dos resultados é executar os experimentos em máquinas dedicadas ou em períodos de cargas leves e uniformes. Porém, quando se deseja comparar a performance de dois algoritmos, tomando-se como medida de desempenho o melhor tempo, é desejável que a carga de trabalho do ambiente computacional dos testes seja representativa do ambiente real onde os algoritmos serão utilizados. Observa-se ainda que a carga real não se repete, nem é reprodutível.

Ao invés de tentar controlar a carga do sistema, uma idéia consiste em forçar as duas aplicações a serem testadas sob a influência dos mesmos fatores externos. O desempenho relativo das aplicações pode ser avaliado executando-as simultaneamente, cada uma com o mesmo número de processadores, porém utilizando diferentes processadores. Usando-se processadores diferentes, as duas aplicações não disputam recursos compartilhados entre si, exceto em termos de comunicação através da rede (o que, no caso do GRASP, devido ao reduzido número de mensagens pequenas, é de menor importância). Além disso, executando-se diversas vezes as duas aplicações simultaneamente, em processadores selecionados aleatoriamente, na média a carga externa que afeta cada estratégia tende a ser a mesma. Pode-se assim fazer a comparação de dois algoritmos aplicados a um mesmo problema com base em seu desempenho médio relativo, utilizando as mesmas máquinas e submetidos a cargas

externas muito similares.

A implementação desta metodologia é simples. São alocados $2p$ processadores para a execução simultânea das duas estratégias, cada uma utilizando metade dos processadores. Os $2p$ processadores são repartidos aleatoriamente entre elas, de modo a evitar distorções provocadas pela alocação sistemática de um mesmo processador à mesma estratégia. Os dois procedimentos são executados simultaneamente, sob a influência da mesma carga externa. Enquanto o procedimento que usa a decomposição pré-escalonada utiliza seus p processadores para a realização de K/p iterações cada um, o procedimento que usa a decomposição auto-escalonada utiliza apenas p - 1 processadores (escravos) para a execução destas iterações, reservando um (mestre) para distribuir as tarefas e coordenar os resultados obtidos.

3. Resultados Computacionais

As duas estratégias foram aplicadas a problemas de decomposição de matrizes no contexto de alocação de tráfego em sistemas de satélites TDMA [7]. A medida utilizada para avaliar a performance foi o tempo marcado pelo relógio. Foram utilizados três conjuntos de matrizes de tráfego que se diferenciam por sua dimensão. Cada um dos três conjuntos de problemas descritos em [1, 7] corresponde a seis matrizes de dimensões iguais a 15, 18 e 21.

Os testes computacionais foram conduzidos numa máquina IBM SP2, configurada com 16 processadores RS6000 mod. 370. Utilizou-se o dispositivo HPS do SP2 para realizar a troca de mensagens entre os processadores pela rede de comunicação. O protocolo de comunicação utilizado foi o *us* da IBM. A comunicação entre processadores foi realizada utilizando-se chamadas de rotinas da biblioteca MPI versão mpich Mpi-1.0.12. As medidas de tempo excluem custos de inicialização e finalização.

Tomou-se o número K de iterações GRASP igual a 9000 e o número *bloco* de iterações por tarefa igual a 500, utilizando-se doze processadores no total. Considera-se como uma amostra de tamanho 60 o conjunto de dez execuções para cada um dos três conjuntos de seis problemas, totalizando 180 observações. Emprega-se um procedimento estatístico para comparar dois sistemas com cargas muito similares [3]. Considera-se o tempo marcado pelo relógio das duas execuções simultâneas do GRASP paralelo como um par de observações, utilizando-se balanceamento estático (A) e balanceamento dinâmico (B). Para cada par computa-se a diferença do tempo e calcula-se um intervalo de confiança para esta diferença. Se o intervalo de confiança inclui zero, os desempenhos das duas estratégias não são diferentes no nível de confiança estabelecido. Detalhes dos testes são relatados em Alvim [1]. A Tabela 1 sumariza os resultados para cada amostra. A primeira coluna apresenta a dimensão dos problemas testados, as colunas indicadas por $\overline{A}$ e $\overline{B}$ fornecem os tempos médios em segundos das 60 execuções, respectivamente com balanceamento de carga estático e dinâmico, enquanto a coluna indicada por $(\overline{A-B})/\overline{A}$ mostra a diferença relativa média. A última coluna fornece o intervalo de confiança de 95% para a média das diferenças.

n	$\overline{A}$	$\overline{B}$	$(\overline{A-B})/\overline{A}$	IC (95%)
15	3064	2597	15%	[279, 655]
18	7486	6403	14%	[595, 1573]
21	15829	13843	13%	[813, 3159]

Tabela 1: Síntese dos resultados de 180 execuções.

Observa-se na Tabela 1 que o intervalo de confiança das três amostras não inclui zero. Portanto, pode-se dizer com 95% de confiança que o balanceamento de carga dinâmico é mais

eficaz do que o balanceamento de carga estático. Os tempos médios de execução da estratégia que usa balanceamento dinâmico foram de 13 a 15% menores do que os da estratégia com balanceamento estático. Em 131 das 180 execuções, a estratégia que usou balanceamento dinâmico foi a mais rápida. Pode-se dizer com 95% de confiança que o balanceamento dinâmico é mais rápido do que o estático e que em 66 a 79% das execuções a estratégia que usar o balanceamento dinâmico será mais rápida.

4. Conclusões

A meta-heurística GRASP vem sendo aplicada com sucesso na solução de problemas complexos de otimização combinatória [2, 7, 8]. Esta caracteriza-se pela independência dos resultados obtidos em cada iteração, o que resulta em implementações paralelas eficientes distribuindo-se as iterações entre os processadores. O custo de comunicação é baixo, visto que uma única variável global é necessária: a melhor solução.

Em um ambiente com processadores não dedicados, a estratégia de paralelização pela distribuição uniforme das iterações (balanceamento de carga estático) pode não ser a melhor opção em termos de balanceamento de carga. De forma alternativa, pode-se distribuir as iterações entre os processadores sob demanda, conforme sua disponibilidade (balanceamento de carga dinâmico). O desenvolvimento e implementação de um experimento possibilitou a comparação do desempenho (medido pelo tempo marcado pelo relógio) destas duas estratégias com base em execuções simultâneas, usando o mesmo número de processadores e sob a influência dos mesmos fatores externos. A análise dos resultados computacionais foi feita sobre três amostras de tamanho igual a 60, totalizando 180 observações. Em todas as amostras, pode-se afirmar com 95% de confiança que o balanceamento de carga dinâmico é mais eficaz que o estático (tempos da ordem de 13 a 15% menores).

As primeiras aplicações paralelas de GRASP surgidas recentemente [5, 6] utilizaram a distribuição uniforme das iterações entre os processadores (balanceamento estático). Os experimentos descritos neste trabalho mostraram que a distribuição dinâmica de iterações é a melhor alternativa para tratar o balanceamento de carga na paralelização do GRASP. Estratégias de paralelização do GRASP baseadas nas conclusões deste trabalho foram aplicadas recentemente com sucesso na solução de problemas de alocação de tráfego em satélites TDMA [1] e de grandes instâncias do problema de Steiner em grafos [4].

Referências

[1] A.C.F. Alvim, *Estratégias de Paralelização da Meta-heurística GRASP*, Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 1998.

[2] T.A. Feo e M.G.C. Resende, "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization* 6 (1995), 109-133.

[3] R. Jain, *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, 1991, New York.

[4] S.L. Martins, C.C. Ribeiro e M.C. Souza, "A Parallel GRASP for the Steiner Problem in Graphs", Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel, Berkeley, 1998, a ser publicado em *Lecture Notes in Computer Science*.

[5] P.M. Pardalos, L. Pitsoulis, T. Mavridou, e M.G.C. Resende, "Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP", *Lecture Notes in Computer Science* 980 (1995), Springer-Verlag, Berlim, 317-331.

[6] P.M. Pardalos, L. Pitsoulis e M.G.C. Resende, "A Parallel GRASP for MAX-SAT Problems", *Lecture Notes in Computer Science* 1180 (1996), Springer-Verlag, Berlim, 575-585.

[7] M. Prais e C.C. Ribeiro, "Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment", *INFORMS Journal on Computing*, 1998, a ser publicado.

[8] M.G.C. Resende e C.C. Ribeiro, "A GRASP for Graph Planarization", *Networks* 29 (1997), 173-189.