MAC 6711 - Tópicos de Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2023

Lista 9

1. Prove que o algoritmo modificado da aula 23 para o MAXCUT é uma $(2+\frac{\epsilon}{2})$-aproximação. (Argumente que ele consome tempo polinomial e que atinge essa razão de aproximação.)

2. Considere o problema de encontrar uma configuração estável em uma rede neural de Hopfield, no caso especial em que todos os pesos são positivos. Isso corresponde ao problema MAXCUT.

   Suponha que o grafo dado é bipartido e conexo, com bipartição, digamos, X e Y. Existe uma configuração "ótima" natural para esta rede de Hopfield, em que todos os vértices de X recebem sinal $+$ e todos os vértices de Y recebem sinal $-$, deixando com isso todas as arestas do grafo boas.

   A pergunta é: Nesse caso especial, quando uma configuração ótima é tão clara, o algoritmo que vimos na aula sempre vai encontrar essa configuração? Dê uma prova de que isso é verdadeiro, ou apresente um exemplo de grafo bipartido conexo, estado inicial, e execução do algoritmo que termine numa configuração em que nem todas as arestas são boas.

3. Lembre-se que para o problema em que o objetivo é maximizar alguma quantidade subentendida, o algoritmo do *gradiente descendente* tem uma versão análoga natural invertida, na qual repetidamente move-se da solução atual para uma de valor estritamente maior. Chamamos essa versão do algoritmo de *gradiente ascendente*.

   Por simetria, as observações feitas sobre o gradiente descendente se aplicam para o gradiente ascendente: para muitos problemas pode-se terminar num ótimo local que não é muito bom. Mas algumas vezes encontramos problemas para os quais uma busca local produz soluções com fortes garantias de qualidade, como no caso do MAXCUT, por exemplo. Vamos considerar agora o problema do emparelhamento bipartido e mostrar que esse fenômeno se aplica a ele.

   Para isso, considere o seguinte algoritmo gradiente ascendente para encontrar um emparelhamento num grafo bipartido.

   > Enquanto existe uma aresta com pontas desmarcadas, adicione uma tal aresta ao emparelhamento corrente. Quando não há mais arestas assim, termine com um emparelhamento localmente ótimo.

   (a) Dê um exemplo de um grafo bipartido G para o qual esse algoritmo gradiente ascendente não retorna um emparelhamento de tamanho máximo.

   (b) Sejam M e M' emparelhamentos num grafo bipartido G. Suponha que $|M'| > 2|M|$. Mostre que existe uma aresta $e' \in M'$ tal que $M \cup \{e'\}$ é um emparelhamento em G.

   (c) Use (b) para concluir que qualquer emparelhamento localmente ótimo devolvido pelo algoritmo gradiente ascendente num grafo bipartido G tem pelo menos metade do tamanho de um emparelhamento máximo em G.

4. Suponha que você está dando consultoria para uma empresa de biotecnia que efetua experimentos em duas caras máquinas de ensaio de alto rendimento, idênticas, que chamaremos de M_1 e M_2. Cada dia, eles têm um número de tarefas que precisam executar, e cada tarefa deve ser atribuída a uma das duas máquinas. O problema que eles precisam resolver é como atribuir as tarefas às máquinas para manter a carga das duas máquinas balanceada a cada dia. O problema é descrito da seguinte maneira. Existem n tarefas, e cada tarefa j tem um tempo de processamento t_j. Eles precisam particionar as tarefas em dois grupos, A e B, onde A será atribuído à máquina M_1 e B à M_2. O tempo necessário para processar todas as tarefas nas duas máquinas é $T_1 = \sum_{j \in A} t_j$ e $T_2 = \sum_{j \in B} t_j$, O problema é garantir que as duas máuinas trabalham grosseiramente a mesma quantidade de tempo, ou seja, minimizar $|T_1 - T_2|$.

   Um consultor anterior mostrou que o problema é NP-difícil. Agora eles estão procurando por um bom algoritmo de busca local. Eles propuseram o seguinte. Comece atribuindo tarefas às duas máquinas arbitrariamente (por exemplo, tarefas de 1 a $n/2$ para M_1 e o restante à M_2). Os movimentos locais são mover uma tarefa de uma máquina para a outra, e apenas fazemos o movimento se ele diminui o valor absoluto da diferença entre os tempos de processamento das máquinas. Você foi contratado para responder as seguintes questões:

   (a) Quão boa é a solução produzida? Assuma que não existe tarefa que domina o tempo total de processamento, ou seja, $t_j \leq \frac{1}{2} \sum_{i=1}^{n} t_i$ para todo j. Prove que, para toda solução localmente ótima, o tempo das duas máquinas operarem está grosseiramente balanceado: $\frac{1}{2} T_1 \leq T_2 \leq 2T_1$.

   (b) Em seguida, pense no tempo de execução do algoritmo: Quantas vezes tarefas serão movidas de uma máquina para a outra? Proponha a seguinte variante pequena modificação no algoritmo. Se, numa mudança local, várias tarefas podem ser movidas de uma máquina para a outra, então o algoritmo escolhe sempre, dentre estas, mover a tarefa j com t_j maior possível. Mostre que, com essa pequena alteração, cada tarefa é movida no máximo uma vez pelo algoritmo, e que portanto o algoritmo efetua no máximo n movimentos até chegar num ótimo local.

   (c) Finalmente, eles se perguntam se deveriam buscar algoritmos melhores. Dê um exemplo um exemplo onde o algoritmo acima não produz uma solução ótima.