PROBLEMA DE SEQUENCIAMENTO DE PADRÕES PARA MINIMIZAR A QUANTIDADE MÁXIMA DE PILHAS ABERTAS

Alexandre Fonseca
Aluno da Universidade de Taubaté - Bolsa PIBIC / CNPQ
Orientador: Dr. Horacio Hideki Yanasse, Pesquisador,
LAC - Laboratório Associado de Computacão e Matemática Aplicada

O objetivo deste projeto é estudar o problema de sequenciamento de padrões de corte, de forma a minimizar a quantidade máxima de pilhas abertas ao longo do processo de corte. Cada tipo de painel cortado é armazenado em uma pilha que permanece aberta até que todos os painéis daquele tipo sejam cortados. Cada padrão escolhido na sequência é cortado completamente. Como consequência, várias pilhas serão abertas e não necessariamente serão completadas. Precisamos dispor de um espaco físico grande suficiente para armazenar estas pilhas incompletas até que sejam concluídas na medida que os padrões vão sendo cortados. Queremos minimizar o tamanho deste espaço físico, e, para isso, minimizamos a quantidade máxima de pilhas abertas a cada momento.

Pretendemos neste trabalho implementar e testar um método exato para solucionar este problema, ou seja, um método que determina a melhor sequência de corte destes padrões a fim de atingirmos nosso objetivo. Focalizaremos nosso trabalho em uma método de busca em árvore. Uma análise de todas as possibilidades não seria muito prático, pois, em situações onde as quantidades de painéis e padrões forem grandes, este método degeneraria numa busca exaustiva(Explosões Combinatoriais). O direcionamento mais adequado da busca em árvore é o desafio deste trabalho, pois temos que viabilizar o processo de busca em árvore em um tempo computacional razoável, obedecendo a limitacão do uso de memória da máquina. A busca em árvore está sendo definida segundo a metodologia conhecida como Branch-and-Bound. O percorrimento da árvore deverá ser feito segundo a estratégia de Pesquisa em Profundidade Primeiro(PPP). Porém, a estratégia PPP é cega, o que pode conduzir a buscas desnecessárias. Assim, introduzimos um critério guloso de escolha para o próximo nó da árvore a ser analisado. Segundo este procedimento, poderemos encontrar uma solucão ótima na primeira folha obtida na expansão desta árvore, ou, possivelmente uma solução muito boa, a qual nos dá um referencial para excluirmos outros caminhos na árvore que apresentem, a priori, uma solucão igual ou pior. Evitaríamos assim a análise de diversas ramificações da árvore.

Na figura 1, exemplificamos a escolha do menor LB do mais recente nível usando a estratégia PPP até chegarmos a primeira folha da árvore (nó 10). Os nós 1, 2, 4, 6, 7 e 8 não precisariam mais serem analisados visto que seus respectivos LB são maiores ou iguais ao LB do nó 10 (que é um nó folha, portanto um solucão viável para o problema). Portanto a sequência de corte dos padrões que proporciona a conclusão dos painéis 3, 1, 4, 2 (nesta ordem) seria a solução ótima do nosso problema (Veja na figura 2).

Ministério da Ciência e Tecnologia
Instituto Nacional de Pesquisas Espaciais - *INPE*
Laboratório Associado de Computação e Matemática Aplicada - *LAC*
PIBIC-CNPq

Relatório de Atividades

PROBLEMA DE SEQUENCIAMENTO DE PADRÕES PARA MINIMIZAR A QUANTIDADE MÁXIMA DE PILHAS ABERTAS

Bolsista: *Alexandre Fonseca*
Orientador: *Dr. Horacio Hideki Yanasse*

Período: *02/96 à 06/96*

São José dos Campos - SP

Atividades

Primeiramente estudei o Relatório de Pesquisa LAC-007/95 publicado pelo Dr. Horácio Hideki Yanasse intitulado "On A Pattern Sequencing Problem To Minimize The Maximum Number Of Open Stacks" onde é apresentado o tema básico de estudo deste projeto.

Vários algoritmos poderiam ser utilizados para a solução deste problema, porém escolhi, a princípio, um algoritmo Branch-and-Bound que está sendo implementado com êxito.

O motivo pelo qual escolhi este algoritmo foi devido a seu possível comportamento em casos mais complexos, podendo solucionar problemas maiores em relação a um algoritmo que utiliza um método enumerativo exaustivo, por exemplo. Seu efeito de eliminar ramos não promissores pode "economizar" tempo no processamento e memória do micro-computador podendo evitar a situação indesejada que é a explosão combinatorial.

A linguagem de programação que estou utilizando é a linguagem C por ser de fácil manutenção e devido a sua portatibilidade entre ambientes mais utilizados UNIX e DOS. Estou enfatizando sua implementação em ambiente DOS para ser executado em micro-computadores. Minha preocupação principal neste projeto é a eficiência do algoritmo podendo ficar como tema de um próximo projeto uma interface gráfica visto que hoje também se faz necessário um ambiente gráfico mais amigável ao usuário.

Meu aprendizado neste trabalho está acontecendo principalmente durante a elaboração do algoritmo, pois estou tendo que antecipar situações críticas de estouro de memória RAM da máquina, armazenamento de matrizes e registros em arquivos no winchester. Consequentemente estou aprendendo técnicas refinadas de programação para poder atender a esta necessidade. Outra grande experiência foi o acontecimento do 2º SICINPE no período de 27 a 28 de maio onde apresentei meu projeto e pude adquirir importantes conhecimentos na elaboração e apresentação de uma palestra científica.

Gostaria de citar a grande importância do auxílio do meu orientador, o Dr. Horacio Hideki Yanasse, que está sendo imprecindível para o sucesso deste trabalho, pois a todo momento que é necessário me orienta e incentiva com bastante propriedade. Enfim, posso afirmar que estou satisfeito com os conhecimentos que estou adquirindo nesta área de pesquisa, a qual tenho interesse e considero bastante gratificante.

Outras Considerações

O algoritmo está sendo implentado num rítmo satisfatório, visto que iniciei os estudos a respeito no começo do mês de fevereiro deste ano. O início da bolsa nesta data foi devido a conclusão do curso de graduação do antigo bolsista Marcelo Saraiva Limeira e que por isso teve o desligamento automático do PIBIC. Mesmo com pouco tempo de estudo sobre o assunto tratado aqui, apresentei o projeto em andamento no 2º SICINPE acontecido no INPE entre 27 e 28 de maio de 1996, fato que exigiu e incentivou um maior aprofundamento teórico neste assunto. Planejo para o próximo período terminar a implementação do algoritmo e fazer uma bateria de testes para poder compará-lo com outros métodos.

Anexado a este relatório de atividades existe um manuscrito mais técnico sobre o trabalho que está sendo desenvolvido.

São José dos Campos, 05 de julho de 1996.

Anexo

PROBLEMA DE SEQUENCIAMENTO DE PADRÕES PARA MINIMIZAR A QUANTIDADE MÁXIMA DE PILHAS ABERTAS

Neste projeto aborda-se o sequenciamento de padrões de corte visando minimizar a quantidade máxima de pilhas abertas durante o processo de corte.

Objetivo

Implementar e testar em um micro-computador, um algoritmo tipo Branch-and-Bound para resolução de um problema de sequenciamento de padrões de corte. Efetuar comparações com outros métodos.

Introdução

Suponha que temos peças retangulares de um certo material de onde queremos extrair pedaços menores - que chamaremos de ***painéis*** - para atender uma certa demanda. A demanda pode ser de quaisquer quantidades de painéis que podem ter tamanhos diferentes. Após sabermos quantas peças precisaremos cortar e quais painéis serão extraídos de cada peça, teremos em mãos o que chamaremos de ***padrões de corte. Padrões de corte*** são as disposições de como os painéis deverão ser cortados das peças. Praticamente o padrão de corte indica como, quantos e quais painéis serão obtidos no corte de uma determinada peça. Vamos supor que temos uma determinada quantidade de Padrões de corte já definidos.

O objetivo deste projeto é estudar o problema de sequenciamento de padrões de corte de forma a minimizar a quantidade máxima de pilhas abertas ao longo do processo de corte de todas as peças necessárias para atender as quantidades desejadas dos painéis. Cada painel diferente que vai sendo cortado é colocado em uma pilha. Cada padrão de corte poderá conter diferentes tipos de painéis. Cada padrão escolhido na sequência deve ser cortado completamente. Como consequência destes cortes, várias pilhas (pertencentes a painéis diferentes) serão abertas e não necessariamente serão completadas. Precisamos eventualmente dispor de um espaço físico muito grande, o suficiente para armazenar estas pilhas ainda incompletas até que sejam concluídas na medida que os padrões vão sendo cortados. Queremos minimizar a requisição deste espaço físico, e, para isso, desejamos minimizar a quantidade máxima de pilhas abertas (painéis incompletos) a cada momento. Pretendemos neste trabalho encontrar uma solução ótima para o problema, ou seja, uma sequência de corte destes padrões que satisfaça nosso objetivo. Um método de busca em árvore será a abordagem a ser tratada. Uma análise de todas as possibilidades não seria muito prático, pois em situações onde as quantidades de painéis e padrões forem grandes, este número a analisar seria extremamente grande (explosões combinatoriais). Temos que tentar viabilizar uma busca em árvore em tempo computacional razoável e com uma certa

coerência no uso de memória da máquina. A busca em árvore está sendo feita segundo a metodologia conhecida como branch-and-bound. O percorrimento da árvore deverá ser segundo o método PPP (Pesquisa Profundidade Primeiro). Uma busca feita apenas pela PPP é cega, podendo torná-la exaustiva. É adotado um critério guloso de escolha em cada nó da árvore, pois, segundo este critério, na melhor das hipóteses, poderemos encontrar uma solução ótima na primeira folha obtida. Caso isto não ocorra, encontraremos uma solução provavelmente muito boa, a qual nos dá um referencial para excluirmos outros caminhos que apresentem, a priori, uma solução igual ou pior, reduzindo o número de ramificações a serem analisadas.

O Método Branch-and-Bound

A técnica Branch-and-Bound envolve um bem estruturado sistema de busca num espaço de todas as possíveis soluções.

Usualmente o espaço de todas as soluções possíveis é dividida em ramificações (branching) cada vez menores e um limitante inferior (para uma minimização do problema) é calculado para um sub-conjunto de soluções para cada ramificação (bound). Antes de cada ramificação, os ramos que excederem este limitante inferior são eliminados de futuras ramificações. Assim um grande grupo de ramificações pode ser excluído sem a necessidade de examinação. Claro que o sucesso da técnica depende da quantidade de possíveis soluções examinadas até que uma solução ótima seja encontrada.

O algoritmo Branch-and-Bound tem sido utilizado com um certo grau de sucesso para a solução de problemas de programação inteira, problema do caixeiro viajante, problema da mochila e uma variedade de outros problemas que possuem uma quantidade limitada (na maioria dos casos grande) de possíveis soluções.

O princípio geral da técnica Branch-and-Bound consiste em um procedimento de ramificação, limitação e eliminação de ramificações.

Todo nó que apresentar seu limitante inferior maior que o limitante superior pode ser eliminado e o nó com menor limitante inferior será o representante do próximo painel a ser completado (problema de minimização). Analisaremos duas regras de decisão no algoritmo Branch-and-Bound:

1. Ramificar o nó de menor limitante
2. Ramificar o nó do nível mais recente com menor limitante.

A regra de decisão número 1, basicamente diz que entre todas as possíveis soluções de todos os níveis da análise, o próximo nó a ser analisado seria o que apresentasse o menor limitante independente do nível que este nó pertencesse. Espera-se com esta política que a primeira solução encontrada seja bem próxima da ótima. A possível desvantagem desta política seria uma maior utilização de memória e um longo tempo na busca até chegar-se a alguma solução.

A outra regra de decisão (número 2), basicamente escolhe para ramificação posterior, entre os nós do nível mais recente, aquele que apresenta o menor limitante inferior. Esta política tem como possível vantagem a rápida obtenção de uma boa solução podendo esta ser ótima.

O teste computacional destes dois métodos será feito com o objetivo de justamente observar as vantagens e desvantagens em relação à ocupação de memória e tempo de execução da busca.

Para cada problema tratado podemos utilizar algoritmos Branch-and-Bound diferentes de acordo com as necessidades específicas do problema.

Algoritmo para Resolução do Problema

Podemos observar que qualquer sequência do corte dos padrões considerados no problema determina em correspondência uma certa sequência na conclusão dos painéis. Para qualquer sequência de painéis a serem concluidos existe pelo menos uma sequência de corte de padrões que a produz. Se soubermos a sequência ótima dos tipos diferentes dos painéis a serem concluidos consequentemente saberemos qual ou quais sequências de padrões são válidas para que isto seja possível.

O algoritmo a ser implementado procura determinar esta sequência ótima em que os painéis devem ser completados. Conforme os padrões são cortados, a pilha de algum painel vai ser concluida em primeiro lugar.

Tentamos identificar então qual painel deve ter sua pilha concluida em primeiro lugar. Ao descobrirmos o primeiro, tentamos identificar o segundo e assim sucessivamente até encontrar uma solução ótima.

É óbvio que esta procura deve ser feita segundo um critério.

Todas informações dos nós são armazenadas em registros e matrizes para que possam ser recuperadas se o atual aprofundamento da busca não resultar em uma soluçào ótima. Isto significa que podem existir nós de outros níveis ainda mais promissores que a solução obtida pela atual ramificação. Este procedimento de comparação das soluções encontradas é repetido até que uma solução ótima seja encontrada.

Conclusão

O Método

- O método de Branch-and-Bound pode diminuir consideralvelmente o tempo de busca e a quantidade de espaço de memória que seriam ocupados por dados desnecessários.

A Memória

- Mesmo em problemas maiores onde a Solução Ótima demoraria mais para ser encontrada, a técnica de armazenamento de dados em winchester asseguraria que não houvesse estouro de memória RAM do equipamento.

A Velocidade

- Em casos de problemas maiores seria necessário, a princípio, um equipamento razoavelmente rápido para se obter uma solução dentro de um prazo razoável. Em caso de não disponibilidade de uma máquina rápida, poder-se-ia relaxar a busca o que não garantiria que a solução encontrada seja a ótima.

A Linguagem

- A implementação do algoritmo feita com a linguagem C em forma de funções torna fácil a manutenção do programa e garante a portatibilidade da função se for desejada sua utilização em um outro algoritmo.

OBS: Testes precisos ainda não foram feitos pelo fato de que o projeto está em andamento.

Referências:

.Fischetti, M. ; Toth, P. ; Vigo, D. Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. **Operations Research**, 42(5):846-859, Sept.-Oct. 1994.

.Gendron, B.; Crainic T.G. Parallel branch-and-bound algorithms: survey and synthesis. **Operation Research**, 42(6):1042-1066, Nov.-Dec. 1994.

.Gillet, B.E. **Introduction to operations research: algorithm aproach.** New York, Mc. Graw Hill, 1976. 617p.

.Yanasse, H.H. **On a pattern sequencing problem to minimize the maximum number of open stacks.** São José dos Campos. INPE, 1995. 21p. (Relatório de Pesquisa LAC-007/95).

.Schildt, H. **Linguagem C: guia do usuário.** São Paulo, Mc. Graw Hill, 1986. 255p.

.Schildt, H. **C avançado: guia do usuário.** 2.ed. São Paulo. Mc. Graw Hill, 1989. 335p.

.Schildt, H. **C completo e total.** São Paulo. Mc. Graw Hill, 1990, 889p.

.Whitehouse, G.E.; Wechsler, B.L. **Applied operations research: a survey.** New York, Wiley, 1976. 434p.

PROBLEMA DA SEQÜÊNCIA DE PADRÕES PARA MINIMIZAR O MÁXIMO DE PILHAS ABERTAS

Alexandre Fonseca
Aluno da Universidade de Taubaté - Bolsa PIBIC / CNPq
Orientador: Dr. Horácio Hideki Yanasse, Pesquisador,
LAC - Laboratório Associado de Computação e Matemática Aplicada

O objetivo deste projeto é estudar o problema de seqüenciamento de padrões de corte de forma a minimizar a quantidade máxima de pilhas abertas ao longo do processo de corte. Pilhas são abertas para armazenar os diferentes painéis que são obtidos a cada padrão cortado. Cada padrão na seqüência é cortado completamente. Como conseqüência destes cortes, várias pilhas (correspondentes a painéis diferentes) serão abertas e não necessariamente serão completadas. Precisamos eventualmente dispor de um espaço físico muito grande, o suficiente para armazenar estas pilhas ainda incompletas até que sejam concluídas à medida que os padrões vão sendo cortados. Queremos minimizar a requisição deste espaço físico, e, para isso, teremos que minimizar a quantidade máxima de pilhas abertas (painéis incompletos) a cada momento. Pretendemos neste trabalho implementar e testar um algoritmo que determine uma solução ótima para este problema, ou seja, determina uma seqüência de corte dos padrões que atinja o nosso objetivo.

Existem grandes dificuldades em se determinar uma seqüência ótima para este problema, pois temos que possibilitar uma busca em tempo computacional viável e com uma certa coerência no uso de memória da máquina. Um método de busca em árvore parece ser uma tentativa interessante para a solução deste problema. Uma análise de todas as possibilidades não seria muito prático, pois em situações onde as quantidades de painéis e padrões forem grandes, a solução degeneraria numa solução exaustiva(Explosões Combinatoriais). A busca em árvore está sendo definida segundo a metodologia conhecida como Branch-and-Bound. O percorrimento da árvore é feita segundo o método PPP(Pesquisa Profundidade Primeiro). Uma busca feita apenas pela PPP é cega, podendo tornar a busca exaustiva. Possibilitamos, por esta razão, um critério guloso de escolha em cada nível da árvore. Segundo esta estratégia, na melhor das hipóteses, poderemos encontrar uma solução ótima na primeira folha obtida, ou, pelo menos, encontraremos uma solução provavelmente muito boa que nos dá um referencial para excluirmos outros caminhos que apresentem, a priori, uma solução igual ou pior. Teríamos, desta forma, menos ramificações a serem analisadas.

O principal desafio deste algoritmo foi sua implementação prevendo a solução para problemas relativamente grandes. Isto se deve pela alta requisição de memória à máquina, pois quanto maior o problema inicial, exponencialmente maior é o consumo de memória e tempo. Eventualmente, vários artifícios podem ser utilizados para solucionar este problema, tais como relaxamento na busca da solução ou armazenamento de variáveis em disco rígido. Foi utilizado um recurso bastante conhecido e útil em programação que é a alocação dinâmica de variáveis. Essa decisão foi tomada para evi-

Ministério da Ciência e Tecnologia
Instituto Nacional de Pesquisas Espaciais - ***INPE***
Laboratório Associado de Computação e Matemática Aplicada - **LAC**
PIBIC-CNPq

Relatório de Atividades

PROBLEMA DE SEQUENCIAMENTO DE PADRÕES PARA MINIMIZAR A QUANTIDADE MÁXIMA DE PILHAS ABERTAS

Bolsista: *Alexandre Fonseca*
Orientador: *Dr. Horacio Hideki Yanasse*

São José dos Campos - SP

Atividades

Primeiramente estudei o Relatório de Pesquisa LAC-007/95 publicado pelo Dr. Horácio Hideki Yanasse intitulado "On A Pattern Sequencing Problem To Minimize The Maximum Number Of Open Stacks" onde é apresentado o tema básico de estudo deste projeto.

Vários algoritmos poderiam ser utilizados para a solução deste problema, porém escolhi, a princípio, um algoritmo tipo Branch-and-Bound.

Para entender melhor este tipo de algoritmo e poder realizar a sua implementação com êxito, estudei alguns artigos e capítulos de livros.

.Fischetti, M. ; Toth, P. ; Vigo, D. Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. **Operations Research**, 42(5):846-859, Sept.-Oct. 1994.

.Gendron, B.; Crainic T.G. Parallel branch-and-bound algorithms: survey and synthesis. **Operation Research**, 42(6):1042-1066, Nov.-Dec. 1994.

.Gillet, B.E. **Introduction to operations research: algorithm aproach.** New York, Mc. Graw Hill, 1976. 617p.

.Whitehouse, G.E.; Wechsler, B.L. **Applied operations research: a survey.** New York, Wiley, 1976. 434p.

O motivo pelo qual escolhi este algoritmo foi devido a seu possível comportamento em casos mais complexos, podendo solucionar problemas maiores em relação a um algoritmo que utiliza um método enumerativo exaustivo, por exemplo. Seu efeito de eliminar ramos não promissores pode "economizar" tempo no processamento e memória do micro-computador podendo evitar a situação indesejada que é a explosão combinatorial.

A linguagem de programação que estou utilizando é a linguagem C por ser de fácil manutenção e devido a sua portatibilidade entre ambientes operacionais. Estou enfatizando sua implementação em ambiente DOS para ser executado em micro-computadores. Para

melhor utilizar esta linguagem, aperfeiçoei meu conhecimento estudando as seguintes referências.

.Schildt, H. **Linguagem C: guia do usuário.** São Paulo, Mc. Graw Hill, 1986. 255p.

.Schildt, H. **C avançado: guia do usuário.** 2.ed. São Paulo. Mc. Graw Hill, 1989. 335p.

.Schildt, H. **C completo e total.** São Paulo. Mc. Graw Hill, 1990, 889p.

Minha preocupação principal neste projeto é a eficiência do algoritmo podendo ficar como uma próxima etapa deste projeto uma interface gráfica visto que hoje também se faz necessário um ambiente gráfico mais amigável ao usuário.

Meu aprendizado neste trabalho tem acontecido principalmente durante a elaboração e implementação do algoritmo, pois estou tendo que antecipar situações críticas de estouro de memória RAM da máquina. Consequentemente estou aprendendo técnicas refinadas de programação para poder atender a esta necessidade. Outra grande experiência durante este período foi minha participação no 2º SICINPE no período de 27 a 28 de maio onde apresentei meu projeto e pude adquirir importantes conhecimentos na elaboração e apresentação de uma palestra científica.

Gostaria de citar a grande importância do auxílio do meu orientador, o Dr. Horacio Hideki Yanasse, que está sendo imprescindível para o sucesso deste trabalho, pois a todo momento que é necessário me orienta e incentiva com bastante propriedade. Enfim, posso afirmar que estou satisfeito com os conhecimentos que estou adquirindo nesta área de pesquisa, a qual tenho interesse e considero bastante gratificante. No anexo, apresento uma breve descrição do problema e o código documentado do que foi implementado até o momento.

São José dos Campos, 20 ~~de janeiro de~~ 1997

PROBLEMA DE SEQUENCIAMENTO DE PADRÕES PARA MINIMIZAR A QUANTIDADE MÁXIMA DE PILHAS ABERTAS

Neste projeto aborda-se o sequenciamento de padrões de corte visando minimizar a quantidade máxima de pilhas abertas durante o processo de corte.

Objetivo

Implementar e testar em um micro-computador, um algoritmo tipo Branch-and-Bound para resolução de um problema de sequenciamento de padrões de corte.

Introdução

Suponha que temos peças retangulares de um certo material de onde queremos extrair pedaços menores - que chamaremos de ***painéis*** - para atender uma certa demanda. A demanda pode ser de quaisquer quantidades de painéis que podem ter tamanhos diferentes. Após sabermos quantas peças precisaremos cortar e quais painéis serão extraídos de cada peça, teremos em mãos o que chamaremos de ***padrões de corte. Padrões de corte*** são as disposições de como os painéis deverão ser cortados das peças. Praticamente o padrão de corte indica como, quantos e quais painéis serão obtidos no corte de uma determinada peça. Vamos supor que temos uma determinada quantidade de padrões de corte já definidos.

O objetivo deste projeto é estudar o problema de sequenciamento de padrões de corte de forma a minimizar a quantidade máxima de pilhas abertas ao longo do processo de corte de todas as peças necessárias para atender as quantidades desejadas dos painéis. Cada painel diferente que vai sendo cortado é colocado em uma pilha. Cada padrão de corte poderá conter diferentes tipos de painéis. Cada padrão escolhido na sequência deve ser cortado completamente. Como consequência destes cortes, várias pilhas (pertencentes a painéis diferentes) serão abertas e não necessariamente serão completadas. Precisamos eventualmente dispor de um espaço físico muito grande, o suficiente para armazenar estas pilhas ainda incompletas até que sejam concluídas na medida que os padrões vão sendo cortados. Queremos minimizar a requisição deste espaço físico, e, para isso, desejamos minimizar a quantidade máxima de pilhas abertas (painéis incompletos) a cada momento. Pretendemos neste trabalho encontrar uma solução ótima para o problema, ou seja, uma sequência de corte destes padrões que satisfaça nosso objetivo. Um método de busca em árvore será a abordagem a ser tratada. Uma análise de todas as possibilidades não seria muito prático, pois em situações onde as quantidades de painéis e padrões forem grandes, este número a analisar seria extremamente grande (explosões combinatoriais). Temos que

tentar viabilizar uma busca em árvore em tempo computacional razoável e com uma certa coerência no uso de memória da máquina. A busca em árvore está sendo feita segundo a metodologia conhecida como branch-and-bound. O percorrimento da árvore deverá ser segundo a estratégia PPP (Pesquisa Profundidade Primeiro). Uma busca feita apenas pela PPP é cega, podendo torná-la exaustiva. É adotado um critério guloso de escolha em cada nó da árvore, pois, segundo este critério, na melhor das hipóteses, poderemos encontrar uma solução ótima na primeira folha obtida. Caso isto não ocorra, encontraremos uma solução provavelmente muito boa, a qual nos dá um referencial para excluirmos outros caminhos que apresentem, a priori, uma solução igual ou pior, reduzindo o número de ramificações a serem analisadas.

O Método Branch-and-Bound

A técnica Branch-and-Bound envolve um bem estruturado sistema de busca num espaço de todas as possíveis soluções.

Usualmente o espaço de todas as soluções possíveis é dividida em ramificações (branching) cada vez menores e um limitante inferior (para uma minimização do problema) é calculado para um sub-conjunto de soluções para cada ramificação (bound). Antes de cada ramificação, os ramos que excederem a melhor solução corrente são eliminados de futuras ramificações. Assim um grande grupo de ramificações pode ser excluído sem a necessidade de examinação. Claro que o sucesso da técnica depende da quantidade de possíveis soluções examinadas até que uma solução ótima seja encontrada.

O algoritmo Branch-and-Bound tem sido utilizado com um certo grau de sucesso para a solução de problemas de programação inteira, problema do caixeiro viajante, problema da mochila e uma variedade de outros problemas que possuem uma quantidade limitada (na maioria dos casos, muito grande) de possíveis soluções.

O princípio geral da técnica Branch-and-Bound consiste em um procedimento de ramificação, limitação e eliminação de ramificações.

Todo nó que apresentar seu limitante inferior maior ou igual que o limitante superior corrente pode ser eliminado e o nó com menor limitante inferior será o representante escolhido do próximo painel a ser completado (problema de minimização). Analisaremos duas regras de decisão no algoritmo Branch-and-Bound:

1. Ramificar o nó de menor limitante
2. Ramificar o nó do nível mais recente com menor limitante.

A regra de decisão número 1, basicamente diz que entre todas as possíveis soluções de todos os níveis da análise, o próximo nó a ser analisado seria o que apresentasse o menor limitante independente do nível que este nó pertencesse. Espera-se com esta política que a primeira solução encontrada seja bem próxima da ótima. A possível desvantagem desta política seria uma maior utilização de memória e uma busca mais extensiva até chegar-se a alguma solução.

A outra regra de decisão (número 2), basicamente escolhe para ramificação posterior, entre os nós do nível mais recente, aquele que apresenta o menor limitante inferior. Esta política tem como possível vantagem a rápida obtenção de uma boa solução podendo esta ser ótima.

Algoritmos Branch-and-Bound diferentes são utilizados de acordo com as necessidades específicas de cada problema tratado.

Algoritmo para Resolução do Problema

Podemos observar que qualquer sequência do corte dos padrões considerados no problema determina em correspondência uma certa sequência na conclusão dos painéis. Para qualquer sequência de painéis a serem concluídos existe pelo menos uma sequência de corte de padrões que a produz. Se soubermos a sequência ótima dos tipos diferentes dos painéis a serem concluidos consequentemente saberemos qual ou quais sequências de padrões são válidas para que isto seja possível.

O algoritmo a ser implementado procura determinar esta sequência ótima em que os painéis devem ser completados. Conforme os padrões são cortados, a pilha de algum painel vai ser concluida em primeiro lugar.

Tentamos identificar então qual painel deve ter sua pilha concluida em primeiro lugar. Ao descobrirmos o primeiro, tentamos identificar o segundo e assim sucessivamente até encontrar uma solução ótima.

É óbvio que esta procura deve ser feita segundo um critério.

Todas informações dos nós são armazenadas em registros e matrizes para que possam ser recuperadas se o atual aprofundamento da busca não resultar em uma solução ótima. Isto significa que podem existir nós de outros níveis ainda mais promissores que a solução obtida pela atual ramificação. Este procedimento de comparação das soluções encontradas é repetido até que uma solução ótima seja encontrada.

Conclusão

O Método

- O método de Branch-and-Bound pode diminuir consideralvelmente o tempo de busca e a quantidade de espaço de memória que seriam ocupados por dados desnecessários.
- Vários critérios de escolha podem ser utilizados num mesmo algoritmo, dependendo da necessidade do usuário.

A Memória

- Mesmo em problemas maiores onde a Solução Ótima demoraria mais para ser encontrada, a técnica de armazenamento de dados em winchester ou alocação dinâmica de variáveis poderia assegurar que não houvesse estouro de memória RAM do equipamento.

A Velocidade

- Em casos de problemas maiores seria necessário, a princípio, um equipamento razoavelmente rápido para se obter uma solução dentro de um prazo razoável. Em caso de não disponibilidade de uma máquina rápida, poder-se-ia relaxar a busca o que não garantiria que a solução encontrada seja a ótima. Foi conseguido resoluções de problemas de tamanho razoavel com equipamentos como um PC 486 DX2 66Mhz sem a exigência de máquinas mais rápidas.

A Linguagem

- A implementação do algoritmo feita com a linguagem C em forma de funções torna fácil a manutenção do programa e garante a portatibilidade da função se for desejada sua utilização em um outro algoritmo.

Bibliografia

.Fischetti, M. ; Toth, P. ; Vigo, D. Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. **Operations Research**, 42(5):846-859, Sept.-Oct. 1994.

.Gendron, B.; Crainic T.G. Parallel branch-and-bound algorithms: survey and synthesis. **Operation Research**, 42(6):1042-1066, Nov.-Dec. 1994.

.Gillet, B.E. **Introduction to operations research: algorithm aproach.** New York, Mc. Graw Hill, 1976. 617p.

.Yanasse, H.H. **On a pattern sequencing problem to minimize the maximum number of open stacks.** São José dos Campos. INPE, 1995. 21p. (Relatório de Pesquisa LAC-007/95).

.Schildt, H. **Linguagem C: guia do usuário.** São Paulo, Mc. Graw Hill, 1986. 255p.

.Schildt, H. **C avançado: guia do usuário.** 2.ed. São Paulo. Mc. Graw Hill, 1989. 335p.

.Schildt, H. **C completo e total.** São Paulo. Mc. Graw Hill, 1990, 889p.

.Whitehouse, G.E.; Wechsler, B.L. **Applied operations research: a survey.** New York, Wiley, 1976. 434p.

Anexo - *Código*

```
//-------------------------------------------------------
//ASSUNTO:MINIMIZACAO DO MAXIMO POSSIVEL DE PILHAS ABERTAS
//-------------------------------------------------------
//-------------------------------------------------------
// ESTRUTURA PRINCIPAL:
// As variaveis P_abertas e P_fechadassao a essencia do
// sistema, pois tornam cada no' da arvore inteligente",
// ou seja com informacoes suficientes para que, a partir
// de qualquer no possamos iniciar a analise novamente.
//
//-------------------------------------------------------
//P_abertas/P_fechadas /            Situacao
/
//---------/-----------/-----------------------------------
//    0    /     0     / Este Painel ainda nao foi tocado
//    0    /     1     / Este Painel foi concluido
//    1    /     0     / A pilha do painel esta aberta
//    1    /     1     / NAO EXISTE ESTA COMBINACAO
//-------------------------------------------------------

#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#include<Ctype.h>
#include<DOS.H>

#define NMAX 340  //NUMERO MAXIMO DE NOS DA ARVORE
#define PMAX 22   //NUMERO DE PADROES DO PROBLEMA
#define TMAX 25   //NUMERO DE PAINEIS DO PROBLEMA


//-------------------------------------------------------
// DELCARACAO DA ESTRUTURA DE UM NO (GLOBAIS)
//-------------------------------------------------------
  struct no {
    int LB;                 // QUANTIDADE DE PILHAS ABERTAS
    int Step;               // ETAPA DA BUSCA NA ARVORE
    int Painel;             // NUMERO DO PAINEL QUE ESTE NO'
                            //REPRESENTA
    int P_fechadas[TMAX];// PILHA DOS PAINEIS QUE JA FORAM
                            //CONCLUIDOS
    int P_abertas[TMAX]; // PILHA DOS PAINEIS QUE AINDA
                            //NAO FORAM FINALIZADOS
    int Padrao[PMAX];       // PILHA DE BITS INDICANDO QUAIS
                            //PADROES FORAM...
                            // ...CORTADOS PARA A CONCLUSAO
                            //DESTE PAINEL
    int M;                  // VALOR BASE DE ESCOLHA DO
```

```
                                // PAINEL A SER CONCLUIDO
    int Choose;                 // INDICE QUE INDICA SE ESTE NO
                                //FAZ PARTE DA SOLUCAO
                                // ...OTIMA (0 P/ NAO E 1 P/ SIM)
    } NUM_NO[NMAX];

//-------------------------------------------------------
// DECLARACAO DAS MATRIZES (GLOBAIS)
//-------------------------------------------------------
  int B[TMAX][TMAX];    // MATRIZ AUXILIAR
  int M[PMAX][TMAX]={   // MATRIZ DOS PADROES DE CORTE

(DADOS INICIAIS)

//-------------------------------------------------------
    1,1,0,0,0,1,1,0,0,1,0,1,0,0,0,1,0,0,1,0,0,0,0,1,0,
    1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,1,0,1,0,1,1,1,1,0,1,
    0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,
    1,0,0,0,1,1,1,0,1,1,1,0,0,0,1,0,1,0,0,0,0,0,0,0,1,
    1,1,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,
    0,0,0,1,0,1,0,0,0,0,1,0,0,1,1,0,1,0,0,0,0,1,1,1,0,
    1,1,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
    0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,
    0,0,0,0,1,0,1,0,0,1,0,1,1,0,1,0,1,0,0,1,0,1,0,1,1,
    0,1,0,1,0,1,1,0,0,0,1,1,0,0,1,0,0,1,0,0,0,1,1,0,1,
    0,0,0,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,1,0,
    0,1,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,
    1,1,0,0,1,0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,
    0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,1,1,1,0,
    0,0,1,0,0,1,0,1,0,1,1,0,0,0,1,0,1,0,0,0,0,0,0,0,1,
    1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,
    0,1,1,0,1,0,0,0,1,1,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,
    0,1,0,1,0,0,0,0,0,1,0,1,0,1,1,0,0,0,1,1,0,1,0,0,0,
    0,1,1,0,1,0,1,0,0,1,0,1,1,0,1,0,1,0,0,1,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,
    0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,
    1,0,1,1,0,0,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0

//-------------------------------------------------------
    };

//-------------------------------------------------------
// DECLARACAO DAS VARIAVEIS (GLOBAIS)
//-------------------------------------------------------

  int escolhido=0;      // Indice do NO' que foi escolhido
                        //para ter seu painel
                        // ...correspondente concluido.
```

```
  int pre_escolhido=0;// Escolha parcial do NO'ate' que
                      // termine as comp. dos M.
  int simulado=0;     // Indice do PAINEL que tera'seu
                      // corte completo simulado.
  int Ultimo_no=0;
  int Numeracao=1; // N§ DO ULTIMO NO' QUE FOI UTILIZADO
                   // NA ARVORE ATE' AGORA
  int Step=0;
  char Resp='Y';
  int LB=0;
  int aux;
  int opcao=1;
  int PAINEIS [TMAX];  // CONTABILIZA OS PAINEIS QUE JA
                       // FORAM CONCLUIDOS
                       // ... EM QUALQUER ETAPA.(VETOR DE
                       // BITS)

  int PADROES [PMAX];  // auxiliar

// INICIALIZANDO MATRIZES, VETORES E REGISTROS (IGUAL A
// ZERO)
//-------------------------------------------------------
  void inicialize()
//-------------------------------------------------------
  {
    int i,j;
    escolhido=0;
    pre_escolhido=0;
    simulado=0;
    Ultimo_no=0;
    Numeracao=1;
    Step=0;
    Resp='Y';
    LB=0;

  for (j=0;j<=TMAX;j++)
    for (i=0;i<=TMAX;i++)
       {
      PADROES[i]=0;                // INICIALIZACAO DO NO 0.
      PAINEIS[i]=0;
      NUM_NO[0].P_abertas[j]=0;
      NUM_NO[0].P_fechadas[j]=0;
       }
      NUM_NO[0].Painel=9999;
      NUM_NO[0].M=-9999;
      NUM_NO[0].Step=0;
```

```
// ROTINA DE TELA : EXIBE UMA MATRIZ
//-------------------------------------------------------
  void display(int C[TMAX][TMAX],int V)
//-------------------------------------------------------
  {
    int i,j;
    int PMAXaux=0;
    if (V==1) PMAXaux=TMAX;
    else PMAXaux=PMAX;
    for (i=0;i<=PMAXaux-1;i++)
      for (j=0;j<=TMAX-1;j++)
     {
       if (i==j) textcolor(GREEN);
       else textcolor(YELLOW);
       cprintf("%i  ",C[i][j]);
       if (j==TMAX-1)  printf("\n");
     }
      textcolor(WHITE);
  }


// ALIMENTA O VETOR DE BITS QUE INFORMA QUAIS PAINEIS
// FORAM CONCLUIDOS
//-------------------------------------------------------
   void closing(int NO)
//-------------------------------------------------------
  {
    int i,j;
    for(i=0;i<=TMAX;i++)
      if(NUM_NO[NO].P_fechadas[i]==1)  PAINEIS[i]=1;
  }


// ROTINA DE TELA : AGUARDA UM COMANDO DO USUARIO
//-------------------------------------------------------
  void wait(int escolha)
//-------------------------------------------------------
   {

     if (escolha==1)
      {
     gotoxy(52,21);
     textcolor(CYAN);
     cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ  ");
     gotoxy(52,22);
     cprintf("ÛÛÛÛ Continuar ? (S/N) ÛÛÛÛ  ");
     gotoxy(52,23);
     cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß ");
     textcolor(WHITE);
```

```
        gotoxy(79,23);
        Resp=getch();
         }
        if (escolha==2)
          {
         textcolor(CYAN);
         gotoxy(44,21);
         cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ ");
         gotoxy(44,22);
         cprintf("ÛÛÛÛ Tecle algo para continuar ÛÛÛÛ ");
         gotoxy(44,23);
         cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß ");
         textcolor(WHITE);
         gotoxy(79,23);
         aux=getch();
          }

    }


// ROTINA DE TELA : EXIBE O MENU PRINCIPAL DE OPCOES PARA
// ESCOLHA DO USUARIO
//-----------------------------------------------------------
  void menu_principal()
//-----------------------------------------------------------
    {
     clrscr();
     textcolor(CYAN);

cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
     textcolor(WHITE);
     cprintf("ÛÛÛÛ  INPE  - Instituto Nacional de Pesquisas
Espaciais  ÛÛÛ PIBIC ÛÛÛ CNPq ÛÛÛÛ");
     textcolor(CYAN);

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßß\n");
     textcolor(CYAN);
     gotoxy(11,5);

cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜ  ");
     gotoxy(11,6);
     textcolor(WHITE);
     cprintf("   PROBLEMA DE MINIMIZACAO DO MAXIMO DE
PILHAS ABERTAS");
     textcolor(CYAN);
```

```
    gotoxy(11,7);

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßß\n ");
    textcolor(WHITE);
    gotoxy(24,11);
    cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    gotoxy(34,12);
    cprintf("MENU PRINCIPAL");
    gotoxy(24,13);
    cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    textcolor(YELLOW);
    gotoxy(25,15);
    cprintf("1. EXIBIR A MATRIZ PROBLEMA.");
    gotoxy(25,16);
    cprintf("2. EXIBIR SOLUCAO OTIMA.");
    gotoxy(25,17);
    cprintf("3. EXIBIR RESOLUCAO PASSO A PASSO.");
    gotoxy(25,18);
    cprintf("4. SAIR.");
    gotoxy(24,20);
    textcolor(WHITE);
    cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    gotoxy(25,22);
    cprintf("OPCAO ? =>");
    scanf("%i",&opcao);

    }


// ROTINA DE TELA : EXIBE A TELA DE APRESENTACAO DO
// PROGRAMA
//----------------------------------------------------------
  void tela_apresent()
//----------------------------------------------------------
   {
    clrscr();
    textcolor(CYAN);

cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    textcolor(WHITE);
    cprintf("ÛÛÛÛ  INPE  - Instituto Nacional de Pesquisas
Espaciais  ÛÛÛ PIBIC ÛÛÛ CNPq ÛÛÛÛ");
    textcolor(CYAN);

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß\n");
    textcolor(CYAN);
```

```
    gotoxy(11,10);

cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜ  ");
    gotoxy(11,12);
    textcolor(WHITE+BLINK);
    cprintf("    PROBLEMA DE MINIMIZACAO DO MAXIMO DE
PILHAS ABERTAS");
    textcolor(CYAN);
    gotoxy(11,14);

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßß\n ");
    textcolor(WHITE);
    gotoxy(11,15);
    printf("                              V.1.1\n");
    textcolor(YELLOW);
    gotoxy(1,20);

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß  ");
    gotoxy(2,21);
    cprintf("ORIENTADOR: Horacio Hideki Yanasse\n");
    gotoxy(2,22);
    cprintf("BOLSISTA  : Alexandre Fonseca\n");
    textcolor(WHITE);
    gotoxy(79,23);
    }


// ROTINA DE TELA : INFORMACAO
//---------------------------------------------------------
  void tela1()
//---------------------------------------------------------
   {
    clrscr();
    textcolor(YELLOW);

cprintf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    cprintf("ÛÛÛÛ                 SIMULACAO DE CONCLUSAO DE
CADA PAINEL                ÛÛÛÛ");

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    textcolor(WHITE);
   }
```

```
// ROTINA DE TELA : INFORMACAO
//-----------------------------------------------------
  void tela2()
//-----------------------------------------------------
   {
    clrscr();
    textcolor(GREEN);

cprintf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ \n");
    cprintf("ÛÛÛÛ                     ESCOLHA DO PAINEL QUE TERA
SUA PILHA CONCLUIDA                ÛÛÛÛ \n");

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß \n");
    textcolor(WHITE);
   }

// ROTINA DE TELA : INFORMACAO
//-----------------------------------------------------
  void tela3()
//-----------------------------------------------------
   {
    clrscr();
    textcolor(CYAN);

cprintf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
    cprintf("ÛÛÛÛ                          SOLUCAO OTIMA DO
PROBLEMA                             ÛÛÛÛ");

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    textcolor(WHITE);
   }


// ROTINA DE TELA : EXIBE A O MOMENTO EM QUE UM PAINEL FOI
// ESCOLHIDO E CONCLUIDO.
//-----------------------------------------------------
  void tela4()
//-----------------------------------------------------
   {
    int i,j;
    clrscr();
    textcolor(YELLOW+BLINK);

cprintf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
```

```
cprintf("ÛÛÛÛ                                  SITUACAO ATUAL
ÛÛÛÛ");

cprintf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");    printf("\n");
    textcolor(WHITE);

    gotoxy(1,6);

printf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
      printf("ÛÛÛÛ PAINEIS FECHADOS ÛÛÛÛ\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
      gotoxy(28,7);
      for(i=0;i<=TMAX-1;i++)
     {
       printf("%i ",PAINEIS[i]);
     }

    gotoxy(1,9);

printf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
      printf("ÛÛÛÛ PADROES          ÛÛÛÛ\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
      gotoxy(28,10);
     for(j=0;j<=PMAX-1;j++)
       {
        printf("%i ",NUM_NO[escolhido].Padrao[j]);
       }

     gotoxy(1,13);

printf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
     printf("                                  NO ESCOLHIDO
=> %i\n",escolhido);
     printf("                                  PAINEL ESCOLHIDO
=> %i\n",NUM_NO[escolhido].Painel);

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß \n");

   }
```

```
// ROTINA DE TELA : INFORMACAO.
//-----------------------------------------------------------
  void tela5()
//-----------------------------------------------------------
    {

printf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
     printf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
PASSO => %i ÛÛÛÛ LB atual=> %i ÛÛÛÛ",Step-
1,NUM_NO[escolhido].LB);

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");

    }


// ALIMENTA OS VETORES DE BITS DE CONTROLE DOS NOS E
// VERIFICA SE ALGUM OUTRO PAINEL E' TAMBEM CONCLUIDO.
//-----------------------------------------------------------
  void Verify_other(int D[TMAX][TMAX])
//-----------------------------------------------------------
   {
     int T=simulado;
     int opened=0;
     int closed=0;
     int j,l,m,n,w,x;
     if (opcao==3)
       {
      tela1();
      printf("  CONCLUIU         => %i\n",T);
      printf("  CONCLUI TAMBEM   => ");
       }
     for (j=0;j<=TMAX-1;j++)         // Inicializa o novo
       {                             // vetor de pilhas
                                     // abertas e fechadas.
      NUM_NO[Numeracao].P_abertas[j]=0;
      NUM_NO[Numeracao].P_fechadas[j]=0;
if ((D[j][T]<=D[T][T]) && (D[j][T]>=D[j][j]) && (j!=T))
        {
         if(PAINEIS[j]==0)
// Informa ao novo no' quais outros paineis
// tambem foram concluidos.
          if (opcao==3)  printf("%i - ",j);
           NUM_NO[Numeracao].P_abertas[j]=0;
           NUM_NO[Numeracao].P_fechadas[j]=1;
           closed++;
```

```
       }else
         {
// Informa ao novo no' quais pilhas ja' estao abertas e
// NAO concluidas.

        if (D[j][T]>0)
          {
            NUM_NO[Numeracao].P_abertas[j]=1;
            NUM_NO[Numeracao].P_fechadas[j]=0;
            opened++;
          }
         }
}
    printf("\n");

// Informa que o painel simulado teve sua pilha concluida.

    NUM_NO[Numeracao].P_abertas[simulado]=0;
    NUM_NO[Numeracao].P_fechadas[simulado]=1;
    NUM_NO[Numeracao].Painel=simulado;
    opened++ ; closed++ ;

    // Calcula o valor de M
    NUM_NO[Numeracao].M= closed - opened;
                                // (Termo de comparacao
                                //  para a escolha do
                                //  proximo no').
    NUM_NO[Numeracao].LB=opened;
    NUM_NO[Numeracao].Step=Step;

  }

// CALCULANDO A MATRIZ BASE
//---------------------------------------------------------
  void Lower_Bound(int C[PMAX][TMAX])
//---------------------------------------------------------
  {
    int i,j,aux,cont1;
     for (i=0;i<TMAX;i++)
      for (j=0;j<PMAX;j++)
     {
       if (C[j][i]==1)
         {
           for(aux=0;aux<TMAX;aux++)
           B[i][aux] = B[i][aux] + C[j][aux];
         }
      }
     NUM_NO[cont1].Step = Step;
```

```
      cont1++;
  }


// ANALISA E ESCOLHE (POR UM CRITERIO) O PROXIMO NO' PARA
// A RAMIFICACAO
//-------------------------------------------------------
  void Sort()
//-------------------------------------------------------
   {
   int i;
   for (i=1;i<=Numeracao;i++) //PERCORRE OS NOS DA ARVORE
     {
       // GARANTE QUE A COMPARACAO SERA' FEITA COM NOS
       // DESTE PASSO
      if (NUM_NO[pre_escolhido].Step==Step)
      {
        if (NUM_NO[i].M > NUM_NO[pre_escolhido].M)
          {
// Indice do NO' escolhido para ter sua pilha concluida.
            pre_escolhido=i;
          }

          }else
            {
           if
((NUM_NO[i].Step==Step)&&(NUM_NO[i+1].Step==Step))
              {
                if(NUM_NO[i-1].Step==Step)
                  {
// Esta' seguindo como criterio de escolha o maior M
// ( M = [pilhas concluidas - pilhas que ficaram abertas])
                if ((NUM_NO[i].M > NUM_NO[i+1].M) &&
                  (NUM_NO[i].M >= NUM_NO[pre_escolhido].M))
                   {
// Indice do NO' escolhido para ter sua pilha concluida.
                     pre_escolhido=i;
                   }
                  }else pre_escolhido=i;
              }
            }
          }
   }


//-------------------------------------------------------
  void display_step()
//-------------------------------------------------------
  {
```

```
    int i,j,l,m;
    gotoxy(1,6);

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    printf("ÛÛÛÛ PAINEIS ABERTOS  ÛÛÛÛ\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    gotoxy(28,7);
    for(j=0;j<=TMAX-1;j++)
      {
        printf("%i ",NUM_NO[Numeracao].P_abertas[j]);
      }

    gotoxy(1,9);

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    printf("ÛÛÛÛ PAINEIS FECHADAS ÛÛÛÛ\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    gotoxy(28,10);
    for(l=0;l<=TMAX-1;l++)
      {
        printf("%i ",NUM_NO[Numeracao].P_fechadas[l]);
      }
    gotoxy(1,12);

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    printf("ÛÛÛÛ PADROES          ÛÛÛÛ\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
    gotoxy(28,13);
    for(m=0;m<=PMAX-1;m++)
      {
        printf("%i ",NUM_NO[Numeracao].Padrao[m]);
      }
    gotoxy(1,15);

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
    printf("ÛÛÛÛ  NO  ÛÛÛ PAINEL ÛÛÛ STEP ÛÛÛ  M  ÛÛÛ  LB
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
```

```
printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
   printf("      %i          %i          %i         %i       %i
\n",Numeracao,NUM_NO[Numeracao].Painel,Step,
NUM_NO[Numeracao].M, NUM_NO[Numeracao].LB);

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ \n");
  }


// CRIA, ATRIBUI, TESTA, ESCOLHE E CONCLUI UM NO'
//----------------------------------------------------------
  void Atribui_novos_nos()
//----------------------------------------------------------
// escolhido... NUMERO DO NO' QUE FOI ESCOLHIDO PARA TER
// SEU PAINEL  CORRESPONDENTE CONCLUIDO.
// NUM_NO[escolhido].P_abertas[]... VETOR PERTENCENTE AO
// NO' ESCOLHIDO  QUE CONTEM OS PAINEIS AINDA nao
// CONCLUIDOS.

  {
// VARIAVEIS LOCAIS
//----------------------------------------------------------
  int i,j,l,m,Item,w,v,s,x;
  //ATRIBUINDO OS PAINEIS NAO CONCLUIDOS AOS NOVOS NOS
  if (escolhido>=0)
   {
     for (l=0;l<TMAX;l++)
      {
     if ((NUM_NO[escolhido].P_fechadas[l]==0) &&
                                          (PAINEIS[l]==0))
       {                             // Assim incluiremos as
         simulado=1;                 // pilhas que ja' estao
         clrscr();                   // abertas e as que
         NUM_NO[Numeracao].Choose=0; //ainda nao foram
                                     // tocadas
         for(w=0;w<=TMAX-1;w++)
           {
// Armazenam os indices dos Padroes que precisariam serem
// cortados p/conclusao deste painel
           if(M[w][l]==1)
             {
               NUM_NO[Numeracao].Padrao[w]=1;
             }else  NUM_NO[Numeracao].Padrao[w]=0;

           }
```

```
          Verify_other(B);
          if (opcao==3)  display_step();
// FAZ A ANALISE PARA ELEGER O MELHOR NO' DESTA ETAPA
          Sort();
          if (opcao==3) wait(2);
          Numeracao=Numeracao+1;
        }
      }
      escolhido=pre_escolhido;
// Inclui o no' escolhido na solucao
      NUM_NO[escolhido].Choose=1;
      closing(escolhido);
      Step++;
    }
  }


// EXIBE A SOLUCAO OTIMA DO PROBLEMA
//----------------------------------------------------------
  void Solution()
//----------------------------------------------------------
  {
   int i,j;
   tela3();
   printf("\n");
   printf("  O CORTE DOS PADROES DEVEM SEGUIR A SEGUINTE
ORDEM:\n");

printf("ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ
ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜ");
   textcolor(CYAN);
   cprintf("                        SEQUENCIA DE PADROES");
   printf("\n  ");
   for(i=0;i<=Numeracao-1;i++)
    {
     if(NUM_NO[i].Choose==1)
      {
       if(NUM_NO[i].LB>LB) LB=NUM_NO[i].LB;
       for(j=0;j<=PMAX-1;j++)
     {
      if(NUM_NO[i].Padrao[j]==1)
       {
        if(PADROES[j]==0)
         {
          textcolor(YELLOW);
          cprintf("%i - ",j);
          textcolor(WHITE);
         }
        PADROES[j]=1;
```

```
          }
         }
        }
      }
   printf("\n");

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
   printf("NUMERO MAXIMO DE PILHAS ABERTAS => %i\n",LB-1);

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
   printf("NUMERO TOTAL DE PADROES ANALISADOS => %i\n",
PMAX);

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
   printf("NUMERO DE NOS DA ARVORE  => %i\n",Numeracao);

printf("ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
   textcolor(YELLOW+BLINK);
   cprintf("                    OBSERVACAO");
   printf("\n  OS OUTROS PADROES PODEM SER CORTADOS EM
QUALQUER ORDEM, POIS NAO\n");
   printf("  MUDARIA A SOLUCAO OTIMA.\n");
   textcolor(WHITE);

printf("\nßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß");
 }


// TESTA SE TODAS AS PILHAS FORAM CONCLUIDAS PARA
FINALIZAR A ANALISE
//----------------------------------------------------------
  void test_end()
//----------------------------------------------------------
  {
    int i, cont=0;
    for (i=0;i<=TMAX-1;i++)
      if (PAINEIS[i]==0) cont++;
      if (cont<=1)
     {
      Resp='n';
      Solution();
     }
  }
```

```
// ALTERA A MATRIZ BASE CONFORME OS PAINEIS VAO SENDO
// CONCLUIDOS
//-------------------------------------------------------
  void modify()
//-------------------------------------------------------
 {
   int i,j;
   for (i=0;i<=TMAX-1;i++)
     if (NUM_NO[escolhido].P_fechadas[i]==1)
       {
      for(j=0;j<=TMAX-1;j++)
        {
         if(i!=j)
          {
            B[i][j]=0;
            B[j][i]=0;
          }
        }
       }
 }

//-------------------------------------------------------
  void main()
//-------------------------------------------------------
 {
  int Painelteste,r;
  textbackground(BLUE);
  tela_apresent();
  wait(2);
  do
   {
    menu_principal();
    if (opcao==1)
     {
       clrscr();
       display(M,2);      // EXIBE A MATRIZ PROBLEMA
       wait(2);
     }

    if (opcao==2)      // EXIBE A SOLUCAO DO PROBLEMA
      {
     inicialize();
     escolhido=0;
     Lower_Bound(M);
     do
      {
       clrscr();
       pre_escolhido=0;escolhido=0;
```

```
      Atribui_novos_nos();
      modify();
      test_end();
     }while (Resp!='n');
     wait(2);
     }
   if (opcao==3)     // EXIBE PASSO A PASSO, A SOLUCAO DO
                     // PROBLEMA
     {
    inicialize();
    escolhido=0;
    Lower_Bound(M);
    do
      {
       clrscr();
       pre_escolhido=0;escolhido=0;
       Atribui_novos_nos();
       modify();
       test_end();
       if (Resp!='n')
         {
           tela4();
           tela5();
         }
       wait(2);
      }while (Resp!='n');
     }
}while (opcao!=4);
clrscr();
}
```