PLANOS DE PROCESSOS CONTENDO ALTERNATIVAS: UMA IMPLEMENTAÇÃO VOLTADA PARA A INTERNET

João Carlos Espíndola Ferreira
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, 88040-900, Florianópolis, SC, Tel: (0xx48) 331-9387 ramal 212, e-mail: jcarlos@emc.ufsc.br

Gabriel Fernando Andriolli
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, 88040-900, Florianópolis, SC, Tel: (0xx48) 331-9387 ramal 240, e-mail: gabriolli@bol.com.br

Abstract. *This paper aims at describing the CAPP module of a software system that is being developed to support the remote manufacturing of parts via Internet. In this CAPP system, the Halevi Matrix method is applied, in which process plans with pre-planned alternatives are considered. The presence of alternatives in a process plan is important because a new alternative may be selected immediately by the software or by the process planner in the case of a machine or a cutting tool being unavailable (e.g. machine under maintenance, broken tool, etc). Thus this is considered a convenient method to determine the machine routing for manufacturing a batch of parts, resulting in a feasible schedule for the use of the machines. The Halevi Matrix method is composed of two steps: (i) generation of the Halevi Universal Matrix based on the features present in the part (e.g. hole, slot); (ii) linearization of the Matrix, taking into account the machines available, which results in the final process plan. This method was implemented by the authors using Java Server Pages (JSP), and an example is presented in order to illustrate the method.*

Keywords. *Process Planning, Remote Manufacturing, Internet.*

1. INTRODUÇÃO

A Internet tem mudado a vida das pessoas em todo o mundo, e uma de suas principais características é a redução ou até eliminação das distâncias entre pessoas/empresas localizadas geograficamente distantes entre si. Conseqüentemente, a Internet contribui diretamente para a globalização. Algumas das atividades que têm sido efetuadas através da Internet são:

- busca de informações em geral (Google, 2002);
- compra e venda de produtos (Amazon, 2002);

Além de disponibilizar informações para os clientes mais rapidamente, os preços de produtos vendidos através do comércio eletrônico são em geral mais reduzidos, devido à eliminação de despesas com aluguel de pontos comerciais, e com a quantidade de funcionários.

Além destas aplicações, a Internet tem sido utilizada na programação e monitoramento de equipamentos de manufatura a distância. Por exemplo, pode-se enviar informações para um robô real localizado bem distante para que ele efetue determinadas atividades (Telerobot, 1998).

Uma outra contribuição da Internet para a área de manufatura é a disponibilização de um software de CAD/CAM para ser utilizado por um usuário remoto (Cybercut, 2000).

Uma vez que a Internet permite a aproximação virtual entre pessoas/empresas, ela pode também ser utilizada como tecnologia para permitir a fabricação de peças a distância (Ferreira e Andriolli, 2001). Este tipo de fabricação é motivado pelo fato do cliente não necessariamente possuir os equipamentos e acessórios para a fabricação do produto. Do ponto de vista da empresa que executa a manufatura, ela ao mesmo tempo em que entra

diretamente em contato com o cliente, ao responder prontamente à solicitação quanto à qualidade do produto e tempo de entrega, ela pode assim não somente manter o seu nicho de mercado, mas também aumentá-lo.

O presente artigo tem por objetivo descrever o módulo de CAPP, que faz parte de um sistema computacional sendo desenvolvido para dar suporte à fabricação de peças a distância via Internet. Neste sistema CAPP aplica-se o método da Matriz de Halevi, em que são considerados planos de processos contendo alternativas pré-planejadas. A presença de alternativas no plano de processo é importante porque uma nova alternativa poderá ser selecionada imediatamente pelo software no caso de uma máquina não estar disponível (p.ex. ocupada ou em manutenção), ou então no caso de uma ferramenta não estar disponível. Isto torna este método conveniente para a determinação do roteiro de fabricação de um lote de peças (que inclui as máquinas-ferramenta selecionadas, e a sua seqüência), resultando assim numa boa agenda para o uso das máquinas. Este método foi implementado no GRIMA/GRUCON pelos autores usando Java Server Pages (JSP) (Sun, 2001), e um exemplo é apresentado no final do artigo para ilustrar o método.

2. PLANOS DE PROCESSOS CONTENDO ALTERNATIVAS

Tradicionalmente as decisões sobre planos de processo para a fabricação de peças são tomadas sem considerar aspectos dinâmicos do chão de fábrica, e desta forma introduz-se restrições artificiais que poderão dificultar a fase de agendamento. Algumas situações que podem ocorrer no chão de fábrica que podem atrasar a fabricação são: indisponibilidade da máquina por estar ocupada fabricando outro lote, ou então por estar quebrada ou em manutenção; pode ser também que uma ou mais ferramentas não estejam disponíveis. Para evitar estes problemas, pode-se atrasar a determinação do plano de processos final, isto é, as possíveis alternativas neste caso não são descartadas, e sim consideradas até o momento da efetiva fabricação do lote no chão de fábrica. Alguns trabalhos foram desenvolvidos no passado que consideraram a presença de alternativas em planos de processos, dentre os quais incluem-se: Wilhelm e Shin (1985), Kruth e Detand (1992); Xirouchakis et al. (1999) e Ferreira e Wysk (2001).

Um fator importante em pesquisas para a geração de planos de processo contendo alternativas é a forma de representação destes planos de processo. Alguns métodos para representar-se planos de processo contando alternativas são os seguintes:

- Petri-nets (Wilhelm e Shin, 1985; Kruth e Detand, 1992);
- Grafos E/OU (Ferreira e Wysk, 2001);
- Grafos direcionados (Catron e Ray, 1991);
- Matrizes (Halevi, 1999).

No presente projeto propõe-se a utilização de matrizes para a representação de planos de processos.

3. O MÉTODO DE HALEVI

Para a geração dos planos de processo contendo alternativas, propõe-se a aplicação do método de Halevi (1999), que é composto pelos seguintes passos:

- Geração da matriz universal de operações (chamada aqui de Matriz Universal de Halevi), onde operações são selecionadas sem considerar máquinas específicas.
- Linearização da matriz, considerando as máquinas disponíveis. A linearização é feita aplicando-se um método em que linhas da matriz são reordenadas considerando-se o custo de execução das operações nas máquinas, e a precedência entre as operações (Halevi, 1999).

De posse do plano de processos linearizado, as informações neste plano, que correspondem a operações, máquinas e ferramentas, podem ser enviadas para o software de controle do sistema de manufatura via Internet. Estes passos serão descritos abaixo.

3.1. Geração da Matriz Universal de Halevi

Considerando-se uma peça de alumínio na qual há um furo passante de diâmetro 10mm, com rugosidade R_a igual a 1,65μm, e tolerância de posição igual a 0,04mm, pode-se usinar este furo de várias maneiras, sendo que a furação com uma broca sólida de metal duro possa bastar para dar a precisão e o acabamento necessários. Uma outra alternativa possível seria efetuar a furação com uma broca helicoidal de aço rápido com diâmetro um pouco menor que 10mm, e depois o acabamento poderia ser feito através de diferentes operações, como por exemplo mandrilamento ou alargamento (Halevi e Weill, 1995).

Visto que pode-se ter muitas seqüências possíveis de operações para usinar uma mesma feature, e que uma peça pode conter várias features, percebe-se a dificuldade de se planejar a fabricação de uma peça de maneira econômica.

É importante que estas seqüências de operações sejam armazenadas, e isto é feito numa matriz onde cada linha descreve uma operação em termos dos seguintes parâmetros: diâmetro que ela vai usinar, comprimento, profundidade de corte, avanço, velocidade de corte, força, potência e tempo. Essa matriz de operações é chamada de Matriz Universal de Halevi (1999), e um exemplo desta matriz é mostrado na tabela 1.

Tabela 1. Exemplo de Matriz Universal com 4 operações

Ferramenta	*Diâmetro (mm)*	*Comprim. (mm)*	*Profund. (mm)*	*Avanço (mm/rot)*	*Velocidade (m/min)*	*Força (N)*	*Potência (kW)*	*Tempo (min)*
Barra mandrilar	20.00	31.00	0.615	0.311	109.702	55.141	0.101	0.057
Br. sólida metal duro	18.77	34.42	9.385	0.298	46.7360	637.727	0.497	0.146
Barra mandrilar	20.00	21.00	1.127	0.088	141.582	39.2050	0.093	0.106
Br. inserto metal duro	17.75	21.00	8.873	0.067	100.739	78.9990	0.133	0.173

Para a geração da Matriz Universal de Halevi (1999), foi implementada uma classe em Java, denominada *Universal.java*, e para construir-se um objeto desta classe é necessário informar a peça para a qual se deseja gerar o plano de processos. Essa informação é passada através de um número de identificação da peça (ID), e então as informações referentes à peça, como matéria-prima e *features*, são obtidas do banco de dados de *features*, que foi implementado em MySQL (2001).

Para ilustrar o funcionamento da classe para a geração da Matriz Universal de Halevi, será considerada a peça ilustrada na figura 1, que possui dois furos que originam-se em faces diferentes do bloco prismático.

As tabelas 2 e 3 mostram as informações contidas no banco de dados para a peça da figura 1. As consultas em SQL estão no topo dessas figuras. Deve-se notar o número de identificação da peça (*Peca_id*), que neste caso é igual a 4.

A partir da tabela 2 percebe-se que a peça ilustrada na figura 1 possui duas *features* com ID igual a 3, e este identificador corresponde a um furo simples. Para um destes furos tem-se o atributo igual a 8, enquanto para o outro furo o atributo é igual a 10. O atributo dado à *feature* relaciona a mesma com os dados mostrados na tabela 3, onde são detalhadas as características da *feature*.

Posteriormente o software gera, para cada feature, um conjunto das 5 melhores Matrizes Universais de Halevi.

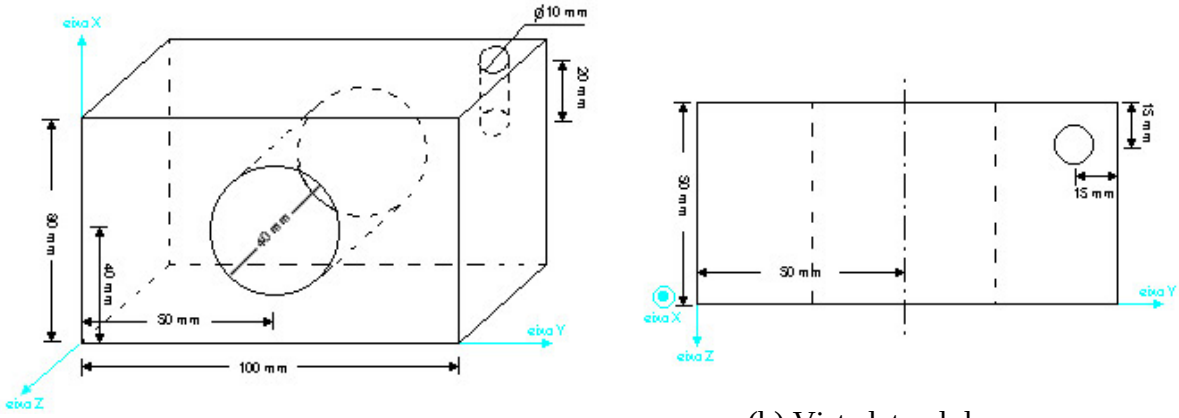
Para o furo de diâmetro a 40mm na peça da figura 1, o seu comprimento é de 50mm, e ele é passante (ver AtribFeat_id=8 na tabela 3). Além disso, a tabela 3 fornece informações sobre a rugosidade e tolerâncias deste furo.

Para gerar a matriz universal para essa *feature*, o código segue basicamente o método descrito por Halevi e Weill (1995). Descreveremos esse código abaixo.

Estágio 1: É necessário construir dois *arrays* de ferramentas, e isto é feito para cada *feature*. Estes *arrays* são:

TLS: Ferramentas que podem participar no processo de usinagem da *feature*.

TLSM: Ferramentas que podem dar o acabamento na *feature*;



(a) Peça em perspectiva (b) Vista lateral da peça

Figura 1. Peça usada como exemplo para a geração da Matriz Universal de Halevi

Tabela 2. Dados da tabela ¨detalhesPeca¨ do banco de dados

*mysql> select * from detalhespeca where Peca_id=4;*

Peca_id	Feature_id	AtribFeat_id	PosX	PosY	PosZ	DirX	DirY	DirZ
4	3	8	40	50	0	0	0	-1
4	3	10	80	85	35	-1	0	0

Tabela 3. Dados da tabela ¨AtribFeatAxiss¨ do banco de dados

*mysql> select * from atribfeataxiss*

AtribFeat_id	Diam	TolDiam	Comp	TolComp	Rugosidade	TolGeom	Cego
1	20	0.5	30	0.1	1	0.03	0
2	44	6.5	40	0.5	999	0.1	1
3	2	0.08	8	0.5	6	0.05	1
4	20	1	30	0.5	3.25	0.01	0
5	12	0.5	20	4.5	999	999	0
6	5	4.75	60	0.5	5.2	999	0
7	20	0.4	30	0.5	12.5	0.1	0
8	40	1	50	0.5	4	0.05	0
9	20	0.5	20	0.5	1	0.01	1
10	10	0.5	30	0.5	6	0.05	1

Para construir estes dois *arrays*, consulta-se dentre as ferramentas disponíveis no banco de dados aquelas que possam ser usadas na usinagem da *feature*, e elas são armazenadas nos respectivos *arrays*. A análise da capacidade de cada ferramenta é feita na própria classe que define a ferramenta.

Para a peça da figura 1, os *arrays* ¨TLS¨ e ¨TLSM¨ gerados são mostrados abaixo.

TLS = [BrocaHelicAçoRápido, BrocaInsertoMetalDuro, BrocaSólidaMetalDuro, FresaInsertoMetalDuro, FresaSólidaMetalDuro, AlargadorAcabamento, AlargadorSemiAcabamento, AlargadorDesbaste, BarraMandrilar]

TLSM = [BrocaSólidaMetalDuro, FresaSólidaMetalDuro, AlargadorDesbaste, BarraMandrilar]

Estágio 2: Neste estágio é construído um *array* das ferramentas necessárias e suficientes para produzir a *feature* do começo ao fim, chamado de ¨TLSW¨. A estratégia usada para se obter esta seqüência de ferramentas (que relacionam-se diretamente a operações) consiste em começar pela última ferramenta (contida em ¨TLSM¨) e ir ¨preenchendo¨ a *feature* com ferramentas capazes de participar do processo (contidas em ¨TLS¨) até encontrar uma que seja capaz de iniciar a usinagem.

Primeiramente, as ferramentas contidas no *array* TLSM são lidas seqüencialmente, e a primeira ferramenta lida é guardada no *array* TLSW. Verifica-se então se essa ferramenta é capaz de iniciar a usinagem da *feature*. Se a resposta for afirmativa, o *array* TLSW será considerado completo, possuindo portanto ferramenta(s) capazes de iniciar e terminar a usinagem da *feature*.

Se não houver ferramenta em TLSM capaz de iniciar a usinagem, calcula-se a máxima profundidade que esta ferramenta pode usinar. Procura-se então em TLS uma ferramenta

que possa efetuar a usinagem numa menor profundidade de corte. Assim, uma ferramenta com essa característica pode preceder a ferramenta anterior, sendo então guardada em TLSW. Então, verifica-se se esta ferramenta tem a capacidade de iniciar o processo, e se a primeira operação puder ser feita com ela, a seqüência de ferramentas (e operações) em TLSW será capaz de realizar a usinagem da *feature*, e o estágio será considerado concluído. Caso a resposta seja negativa, novamente são calculadas as profundidades de corte até se encontrar uma operação capaz de iniciar a usinagem considerando-se então a seqüência resultante como sendo capaz de usinar a *feature* por completo.

Alguns dos possíveis *arrays* TLSW obtidos através do método de Halevi para a *feature* do nosso exemplo são mostrados abaixo. Estes *arrays* são baseados nos *arrays* TLS e TLSW mostrados anteriormente. Observa-se que o método de Halevi não considera todas as seqüências possíveis, e será mostrado neste artigo que elas poderiam ser interessantes do ponto de vista de otimização.

$$TLSW = \begin{cases} [FresaSólidaMetalDuro – BrocaHelicAçoRápido\ ] \\ [AlargadorDesbaste – BrocaHelicAçoRápido] \\ [AlargadorDesbaste – BrocaInsertoMetalDuro] \\ [BarraMandrilar – BrocaSólidaMetalDuro] \\ [BrocaSólidaMetalDuro] \end{cases}$$

Estágio 3: Neste estágio calcula-se as condições de corte e detalha-se todas as operações necessárias para produzir a *feature*. A entrada para esse estágio é o *array* TLSW.

O método implementado pelos autores para este estágio é diferente daquele proposto por Halevi, pois a entrada é uma árvore de ferramentas (em vez de um *array*). Nesta árvore, a ferramenta que está em sua raiz é capaz de efetuar o acabamento da *feature* (advinda de TLSM), e cada ramo a partir dela corresponde a uma seqüência capaz de usinar a *feature* por completo, ou seja, cada ramo representa uma futura matriz universal para a *feature*. Percebe-se portanto que cada ramo da árvore pode ser comparado a um *array* TLSW.

Gera-se uma árvore para cada operação do *array* TLSM. Desta forma, são consideradas todas as possibilidades mais promissoras para se usinar a *feature*.

Após ser gerada uma árvore para uma determinada ferramenta do *array* TLSM, a próxima árvore (referente à próxima ferramenta de TLSM) só será construída após ter-se gerado as matrizes universais para a primeira ferramenta.

Para o exemplo deste artigo, tem-se quatro ferramentas no *array* TLSM, resultando em quatro árvores. A figura 2 mostra como seria a representação de todos os *arrays* TLSW para a *feature* furo com diâmetro igual a 40mm. Nota-se que a raiz de cada árvore corresponde a uma ferramenta capaz efetuar o acabamento da *feature*, e que nas extremidades (folhas) tem-se ferramentas capazes de iniciar a usinagem. Assim, cada ramo corresponde a uma possível seqüência de ferramentas (e suas operações correspondentes).

No caso da figura 2(d), a árvore é representada somente pela raiz, e neste caso esta ferramenta é capaz de usinar a *feature* por completo, satisfazendo os requisitos do projeto.

Após a construção da árvore, calcula-se as condições de corte para cada ferramenta (operação) da árvore, e estes ramos são organizados nas diversas matrizes universais. Estas matrizes são objetos Java do tipo *java.util.Vector*, e elas são armazenadas na variável ¨*matrizes*¨ e posteriormente na variável ¨*todasMatrizes*¨ (figura 3).

Na figura 3 a variável ¨*matAux*¨ representa uma matriz universal qualquer referente a uma determinada *feature.* Nesse caso específico, tem-se dois objetos do tipo ¨*matrizes*¨, o que indica que as duas *features* da peça da figura 1 podem ser usinadas.

Finalmente, o *Vector* ¨*todasMatrizes*¨ representa a peça inteira do ponto de vista das matrizes universais, pois considera todas as *features* possíveis de serem usinadas. Na figura 3, a *feature* ¨0¨ representa o furo de menor diâmetro, enquanto a *feature* ¨1¨ representa a o furo de maior diâmetro (ver figura 1).

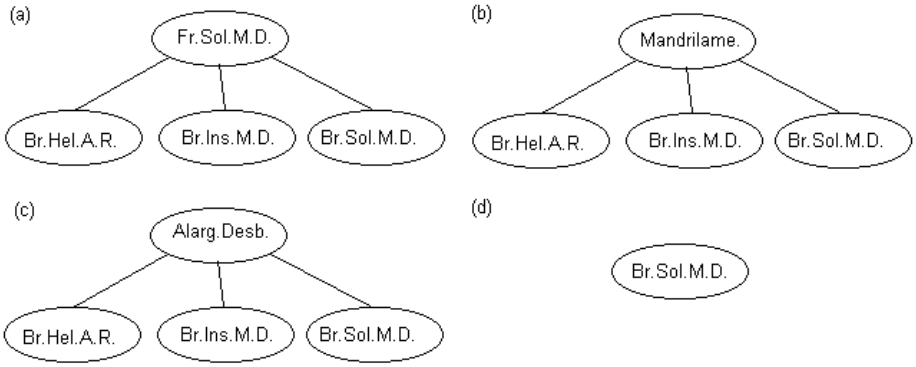


Figura 2. Os *arrays* TLSW são representados por árvores no método implementado pelos autores no GRIMA/GRUCON

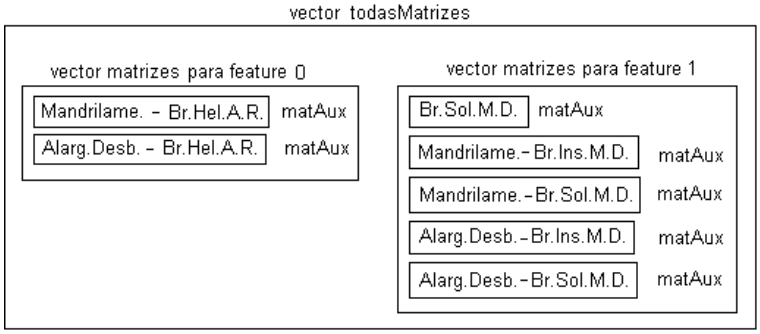


Figura 3. Variáveis envolvidas na representação das Matrizes Universais de Halevi

O próximo passo consiste em combinar as *n* melhores matrizes universais de cada *feature* de modo a se obter as *m* melhores para a peça inteira. A ordenação e feita da seguinte maneira:

(i) Faz-se combinações entre as opções de matrizes universais para cada *feature*;
(ii) Ordena-se as matrizes de acordo com o tempo;
(iii) Redimensiona-se o *Vector ¨universalPeca¨* para suportar o número *m* de matrizes especificado.

Assim, o algoritmo faz um *loop* para cada árvore (isto é, para cada ferramenta de TLSM), tendo-se então as diversas matrizes universais para uma *feature*, e estas são armazenadas na variável *¨matrizes¨*. O mesmo procedimento é aplicado para todas as *features*, chegando-se ao final nas melhores matrizes universais para todas as *features* (ver figura 3).

Para a *feature* ¨1¨ na figura 3, tem-se na primeira linha *BrocaSólidaMetalDuro*, que é suficiente para usinar o furo. A segunda e terceira linhas correspondem a uma matriz universal vinda da árvore da figura 2(a), e para as linhas seguintes aplica-se a mesma lógica, isto é, representam aos pares uma matriz universal oriunda de uma ramo das árvores.

As matrizes universais representadas na figura 3 são ordenadas de acordo com o tempo, e as cinco melhores são selecionadas. O vetor de matrizes resultante é então guardado na variável *¨todasMatrizes¨*, e a próxima *feature* é avaliada.

3.2. Linearização da Matriz Considerando as Máquinas Disponíveis

Neste passo, dada uma Matriz Universal de Halevi e tendo-se as informações das máquinas (disponíveis no banco de dados), busca-se determinar em que máquina e em qual seqüência cada operação dessa matriz universal será executada. Essa informação é dada na forma de uma outra matriz que representa uma linearização da Matriz Universal introduzida. Essa matriz linearizada apresenta as operações que serão executadas em cada

máquina, e elas são ordenadas respeitando-se as precedências entre as operações. Ela apresenta também os custos e os tempos de execução considerando os eventuais penalidades ("*penalties*") decorrentes de trocas de máquina. A tabela 4 ilustra uma matriz linearizada.

Tabela 4. Exemplo de uma Matriz Universal linearizada

Operação				
Nova	Antiga	Máquina	Custo ($)	Tempo (min)
1	1-2-3-4	3	2.569	0.856
Total			2.569	0.856
1 penalty			1x0.21	1x0.03
Total			2.759	0.886

Nesta tabela tem-se quatro ferramentas iniciais (1, 2, 3 e 4) fornecidas pela matriz universal. As quatro operações referentes a estas ferramentas deverão ser executadas nessa mesma seqüência, e todas na máquina 3, podendo assim ser representada como uma única operação do ponto de vista de máquinas que participarão no processo da fabricação da peça. Os custos e os tempos são mostrados, bem como as penalidades.

4. INTERFACE COM O USUÁRIO

Foi desenvolvida uma interface gráfica através da qual o usuário pode introduzir uma peça e receber as matrizes universais para ela. A interface gráfica foi desenvolvida em HTML, e o processamento é feito com tecnologia JSP. O *link* para este programa é http://einstein.lmp.ufsc.br:8080/jsp/nosso/halevi.

Exemplos de matrizes universais geradas para a peça são ilustrados na figura 4, e matrizes auxiliares, as quais são detalhadas em (Halevi, 1999) são mostradas na figura 5. A matriz linearizada é mostrada na figura 6.

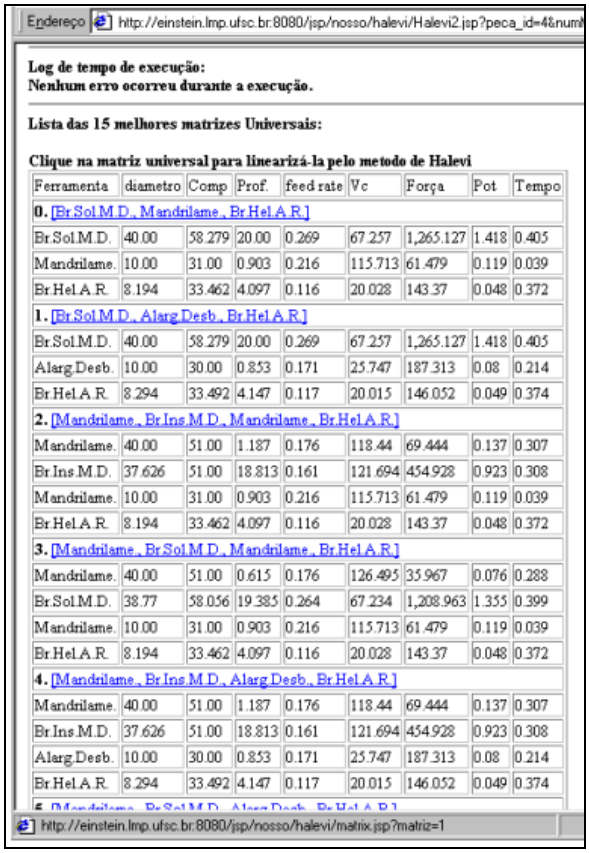


Figura 4. Matrizes Universais de Halevi geradas pelo sistema CAPP

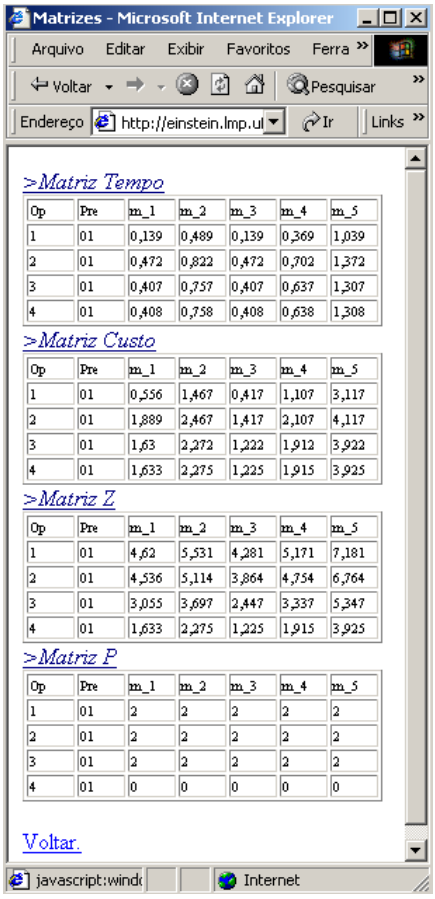


Figura 5. Matrizes intermediárias (Halevi, 1999)

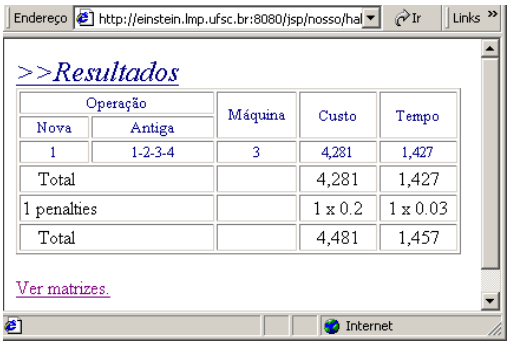


Figura 6. Matriz linearizada

5. CONCLUSÕES

O programa apresentado neste artigo visa dar suporte à fabricação de peças a distância. A abordagem proposta busca o desenvolvimento de módulos de software na linguagem Java e HTML para a tomada de decisões sobre o planejamento do processo para a fabricação de um lote de peças. Tais decisões incluem a seleção e seqüenciamento das operações, as máquinas onde elas serão executadas, e as ferramentas a serem utilizadas. O plano de processos para as peças inclui alternativas, e ele é representado através de matrizes. O método de Halevi apresentou bons resultados, e utilizou-se estrutura de árvore para a geração da Matriz Universal de Halevi.

Pretende-se no futuro incluir outras *features* no software, como cavidades e ranhuras, e no momento está sendo desenvolvido um módulo para a geração do programa de comando numérico (código "G") para a usinagem da peça. Este programa poderá ser enviado via Internet para o comando da máquina.

7. REFERÊNCIAS

(Amazon, 2002) Amazon, "Amazon.com – Earth´s Biggest Selection", http://www.amazon.com, 2002

(Catron e Ray, 1991) Catron, B.A. e Ray, S.R., "ALPS: A Language for Process Specification", International Journal of Computer Integrated Manufacturing, Vol. 4, No. 2, 1991, págs 105-113

(Cybercut, 2000), Cybercut, "Cybercut Project", http://cybercut.berkeley.edu/html/design/webcad_user.htm, 2000

(Ferreira e Wysk, 2001) Ferreira, J.C.E. e Wysk, R.A., "An Investigation of the Influence of Alternative Process Plans on Equipment Control", Journal of Manufacturing Systems, Vol. 19, No. 6, 2001, págs 393-406

(Ferreira e Andriolli, 2001) Ferreira, J.C.E. e Andriolli, G.F., "Uma Metodologia para a Fabricação de Peças à Distância", XVI Congresso Brasileiro de Engenharia Mecânica (COBEM 2001), Uberlândia, MG, novembro de 2001

(Google, 2002) Google, "Google Brasil", http://www.google.com.br, 2002

(Halevi e Weill, 1995) Halevi, G. e Weill, R.D., "Principles of Process Planning: A Logical Approach", Chapman & Hall, 1995

(Halevi, 1999) Halevi, G., "Restructuring the Manufacturing Process: Applying the Matrix Method", St. Lucie Press, 1999

(Kruth e Detand, 1992) Kruth, J.P. e Detand, J., "A CAPP System for Nonlinear Process Plans", Annals of the CIRP, Vol. 41, No. 1, 1992, págs 489-492

(MySQL, 2001) MySQL, "MySQL", http://www.mysql.com, 2001

(Sun, 2001) Sun, "JavaServer Pages ™", http://java.sun.com/products/jsp, 2001

(Telerobot, 1998) Telerobot Project, "Australia's Telerobot on the Web", http://telerobot.mech.uwa.edu.au, 1998

(Wilhelm e Shin, 1985) Wilhelm, W.E. e Shin, H.-M., "Effectiveness of Alternate Operations in a Flexible Manufacturing System", International Journal of Production Research, Vol. 23, No. 1, 1985, págs 65-79

(Xirouchakis et al., 1998) Xirouchakis, P., Kiritsis, D. e Persson, J.-G., "A Petrinet Technique for Process Planning Cost Estimation", Annals of the CIRP, Vol. 47, No. 1, 1998, págs 427-430