Uma solução computacional para geração automática de peças com flexibilidade de processo para simulação

José Hamilton Chaves Gorgulho Júnior (UNIFEI) gorgulhojunior@ig.com.br

Eduardo Vila Gonçalves Filho (EESC-USP) evila@sc.usp.br

Resumo

Este trabalho apresenta a metodologia desenvolvida para gerar automaticamente um grande conjunto de peças com variação no número de operações e também com flexibilidade de processo. Inicialmente o texto apresenta as características de um ambiente de manufatura turbulento e os arranjos físicos propostos para atuar nesta condição. Em seguida destaca o arranjo distribuído e apresenta o problema a ser analisado com simulação. O item seguinte trata da flexibilidade de processo. Após a apresentação da metodologia são mostrados os resultados obtidos por meio de uma implementação computacional. No final são apresentados os comentários sobre os resultados obtidos.
Palavras-chave: Simulação; Flexibilidade de processo; Layout.

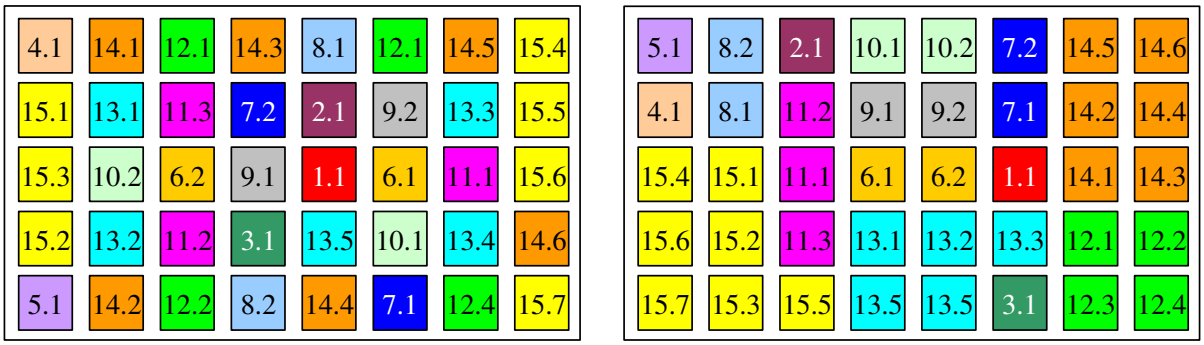
1. Introdução

As indústrias de manufatura de classe mundial estão expostas a um complexo ambiente onde as mudanças nos processos e produtos ocorrem com grande freqüência. Esse ambiente é citado por Rheault, Drolet e Abdulnour (1995) como volátil, turbulento ou altamente dinâmico. Esses autores compilaram as diversas características apresentadas por esse ambiente e que foram citadas por outros pesquisadores:

- Alta variabilidade na demanda e no tamanho dos lotes de produção;
- Alta variabilidade nos tempos de processamento e nos tempos de preparação;
- Demanda parcialmente ou totalmente estocástica;
- Freqüentes mudanças no mix de produtos;
- Variabilidade nas seqüências de produção;
- Forte competição.

Em ambientes com essas características os arranjos físicos clássicos (por produto, por processo, posicional e celular) não atingem um nível de desempenho satisfatório (BENJAAFAR, HERAGU e IRANI, 2002). Sendo assim novos arranjos foram propostos para operarem nas condições citadas, podendo-se citar o *arranjo distribuído* sugerido por Montreuil e Venkatadri (1991), o *arranjo fractal* apresentado por Venkatadri, Rardin e Montreuil (1997) e o *arranjo modular* introduzido por Irani e Huang (2000).

O arranjo distribuído (*distributed layout*) também foi denominado de espalhado (*scattered layout*) e disperso (*dispersed layout*). Caracteriza-se por espalhar as máquinas pelo chão de fábrica de modo a aproximar diferentes tipos de máquinas, ou seja, o objetivo desse arranjo é garantir a proximidade de qualquer estação de trabalho de qualquer processo às estações de trabalho de outros processos para que rotas mais eficientes possam ser criadas em tempo real pelo sistema computadorizado de planejamento e controle da manufatura. A Figura 1 compara um arranjo funcional com um arranjo distribuído.



Arranjo distribuído Arranjo funcional

Figura 1 - Arranjo distribuído e arranjo funcional (adaptado de MONTREUIL e VENKATADRI, 1991)

Montreuil et al. (1993) compararam o desempenho entre arranjos aleatoriamente distribuídos com arranjos distribuídos por um critério de utilização de máquinas. Benjaafar e Sheikhzadeh (2000) e Lahmar e Benjaafar (2005) realizaram comparações entre arranjos funcionais, aleatoriamente distribuídos, parcialmente distribuídos e maximamente distribuídos.

Esses trabalhos apontam para uma grande superioridade do arranjo parcialmente distribuído sobre o funcional e uma pequena vantagem do arranjo maximamente distribuído sobre o parcialmente distribuído. Deve-se ressaltar que nesses trabalhos e também nas demais pesquisas relacionadas com o arranjo distribuído todos os resultados foram obtidos por meio de simulações com dados gerados pelos autores. A única exceção encontrada é o trabalho de Baykasoglu (2003) que usou dados reais, mas sem implementar o arranjo projetado.

2. Descrição do problema

Nenhum dos trabalhos relacionados com o arranjo distribuído leva em consideração peças com flexibilidade de processo. Isso permite levantar a hipótese de que o arranjo maximamente distribuído possa ter um desempenho sensivelmente superior ao parcialmente distribuído se o sistema de programação (*scheduling*) tiver opções de seqüenciamento das operações.

Para trabalhar sobre essa hipótese, além dos arranjos a serem comparados, é necessário que o sistema de simulação tenha disponível conjuntos de peças com flexibilidade de processo. Sem a disponibilidade de dados reais para a pesquisa a solução adotada foi gerá-los de forma automática, pois criar manualmente centenas de peças com flexibilidade no processo de fabricação é uma tarefa muito extensa. O item seguinte discute a representação dessa flexibilidade.

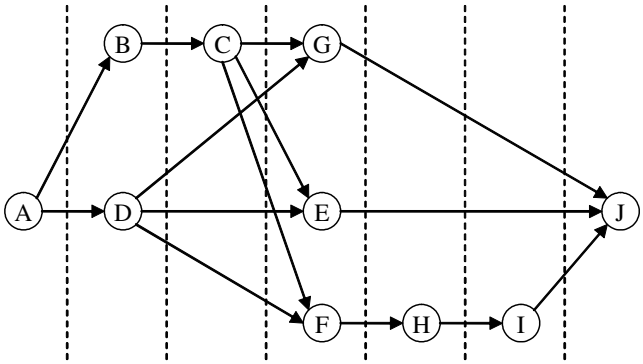
3. Representação do processo com flexibilidade

Para representar um processo com flexibilidade de seqüenciamento das operações optou-se pelo *diagrama de precedência* (*precedence diagram*), como usado por Groover (1987), Hutchinson e Pflughoeft (1994), Borenstein (2000), Sarker e Li (2001) e Rohde e Borenstein (2004). O diagrama de precedência é um tipo de grafo com as seguintes características (GONDRAN, MINOUX e VAJDA, 1984):

- As arestas possuem orientação (grafo orientado ou dirigido);
- Nenhuma aresta parte e chega a um mesmo nó (sem laços ou *loop*);
- Possui apenas um nó sem precedentes (apenas um nó raiz);
- Possui apenas um nó sem descendentes (apenas um nó folha ou terminal).

A Figura 2 mostra do lado esquerdo um exemplo de diagrama de precedência contendo 10 operações. No lado direito têm-se as mesmas relações, mas apresentadas de forma textual. Nota-se que a representação gráfica está dividida em *níveis*. Um procedimento apresentado

por Gondran, Minoux e Vajda (1984), denominado Algoritmo 8, permite calcular o número de níveis e em que nível cada tarefa deve ficar posicionada.



Operação	Precedente
A	-
B	A
C	B
D	A
E	C, D
F	C, D
G	C, D
H	F
I	H
J	E, G, I

Figura 2 – Exemplo de um diagrama de precedência.

As operações dentro de cada nível podem ocorrer em qualquer ordem. Dos 7 níveis do exemplo da Figura 2 apenas um nível possuí 2 operações e outro 3 operações, os demais contém apenas uma operação cada. Com essas informações é possível calcular o número total de seqüências. Usa-se o produtório dos fatoriais do número de operações de cada nível, como mostra a equação 1.

$$NTS = \prod_{i=1}^{nn} (nop(i)!) \qquad (1)$$

Onde: NTS = Número Total de Seqüências
nn = número de níveis
nop(i) = número de operações do nível i

Sendo assim tem-se um total de 12 possíveis seqüências de fabricação para o exemplo da Figura 2 (1! ∗ 2! ∗ 1! ∗ 3! ∗ 1! ∗ 1! ∗ 1!=12). Todas as seqüências estão listadas a seguir.

A B D C E F G H J K	A B D C G E F H J K	A D B C F E G H J K
A B D C E G F H J K	A B D C G F E H J K	A D B C F G E H J K
A B D C F G E H J K	A D B C E F G H J K	A D B C G E F H J K
A B D C F E G H J K	A D B C E G F H J K	A D B C G F E H J K

Com base na representação literal das relações de precedência elaborou-se uma metodologia para criar automaticamente conjuntos de peças com diferentes relações de precedência e número de operações. Essa metodologia é apresentada no próximo item.

4. Metodologia para geração do conjunto de peças

No modelo que será apresentado o usuário define inicialmente o número de processos existentes no arranjo (NumProc). Em seguida deve-se determinar o número mínimo e máximo de operações que cada peça gerada poderá ter (OpMin e OpMax) e a faixa de tempo dentro da qual o tempo de cada operação será escolhido (Tmin e Tmax). Os últimos dados que devem ser configurados são o número de peças a gerar (NumPec) e o nome base dos arquivos que conterão as peças (Nome).

Ao final do processo tem-se um arquivo para cada peça contendo suas relações de precedência e o tempo que cada operação consome. Esses arquivos têm o mesmo nome, com exceção de uma numeração seqüencial que os diferencia.

Para que seja possível gerar peças com diversidade de relações de precedência o procedimento irá basear-se em relações pré-montadas e colocadas em arquivos denominados *Modelos*. Foi criado um arquivo modelo para cada número de operações (até o momento entre 4 e 15 operações) e cada arquivo contém algumas estruturas de precedência. Na Figura 3 tem-se o conteúdo do arquivo '06.txt' que contém cinco estruturas de precedência para peças com 6 operações (do lado direito foram colocados comentários para facilitar o entendimento).

5	=> Número de modelos do arquivo
A, B, C, D, E, F	=> Identificadores das operações
0, A, A, A, CD, BE	=> Modelo de precedência 1
0, A, A, B, C, DE	=> Modelo de precedência 2
0, A, A, A, A, BCDE	=> Modelo de precedência 3
0, A, A, A, BCD, E	=> Modelo de precedência 4
0, A, A, AB, C, DE	=> Modelo de precedência 5

Figura 3 – Conteúdo (comentado) do arquivo modelo para 6 operações.

Pela Figura 3 pode-se perceber que se trata de um arquivo de texto sem formatação, normalmente denominado de ASCII puro. Nele cada informação é colocada em uma linha ou, se estiverem em uma mesma linha, separadas por vírgula. A primeira informação do arquivo é o número de modelos que ele contém. Em seguida há os caracteres que representam as operações (poderiam ser letras minúscula, numerais ou combinações). Finalmente, na seqüência, tem-se em cada linha um modelo de precedência. O nó raiz (que não tem nenhum precedente) é sempre identificado nas relações de precedência pelo caractere zero (0).

A Figura 4 apresenta, de forma gráfica, os cinco modelos de precedência contidos no arquivo de modelo apresentado pela Figura 3, juntamente com o número de seqüências possíveis.

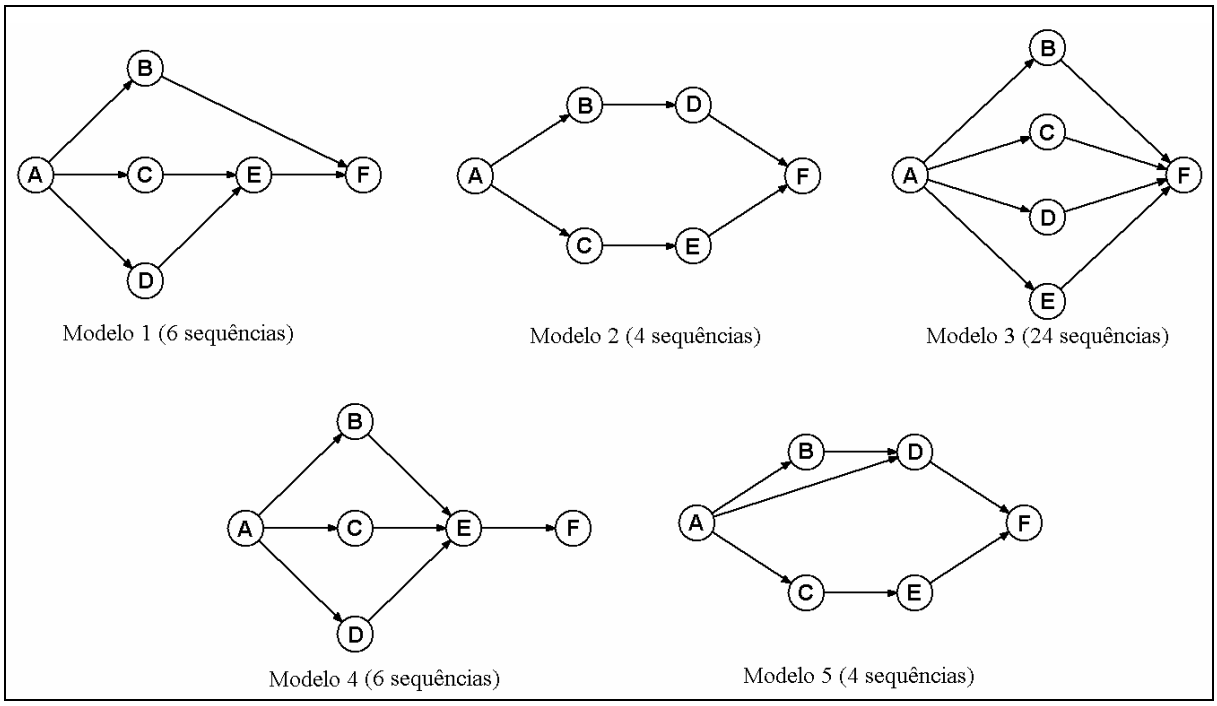


Figura 4 – Cinco modelos de precedência para peças com 6 operações.

A Figura 5 mostra o fluxograma geral de funcionamento do modelo proposto.

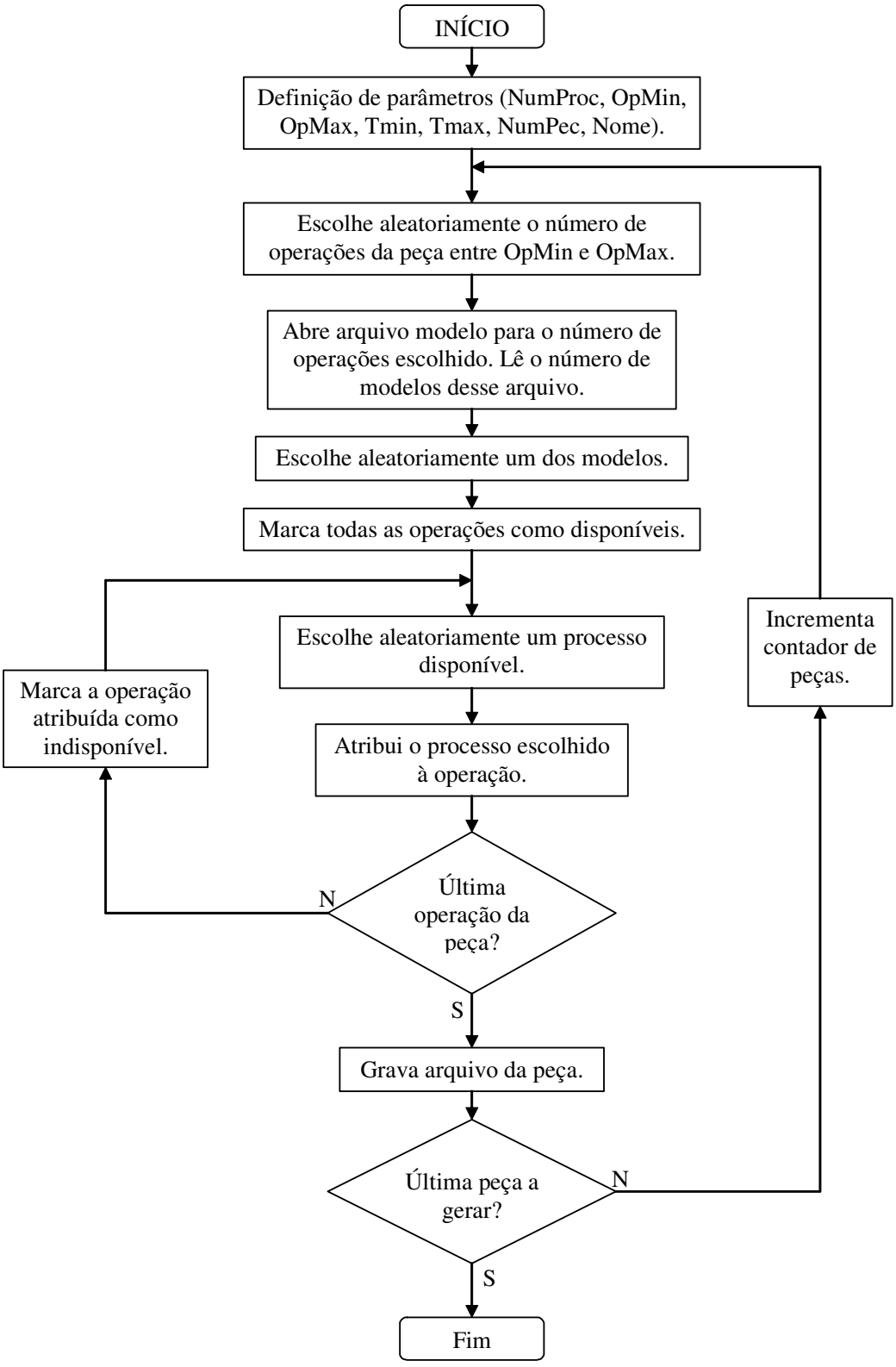


Figura 5 – Fluxograma de funcionamento do gerador automático de peças.

Analisando o fluxograma é possível notar que o uso de escolhas aleatórias leva a uma probabilidade muito pequena de gerar duas peças idênticas. Se tiverem o mesmo número de operações poderão ter estruturas de precedência distintas. Se a estrutura for a mesma podem conter operações diferentes. E mesmo na situação onde o número máximo e mínimo de operações da peça for igual ao número de processos do arranjo (OpMax=OpMin=NumProc) a ordem das operações e o tempo de cada uma irá diferir. Mas, se mesmo com toda essa

aleatoriedade, forem criadas peças idênticas em um mesmo grupo, isso não representa nenhum problema. Nada impede que em um ambiente volátil ocorra a repetição de uma peça em um curto período de tempo.

5. Implementação computacional

O modelo apresentado foi computacionalmente implementado em Visual Basic 6. Na Figura 6 tem-se a janela inicial do programa. Serão geradas 100 peças para um arranjo contendo 15 tipos de processos. Cada peça terá entre 5 e 10 operações e cada operação terá entre 10 e 100 minutos. Nota-se na parte inferior esquerda da tela uma configuração que será discutida adiante.

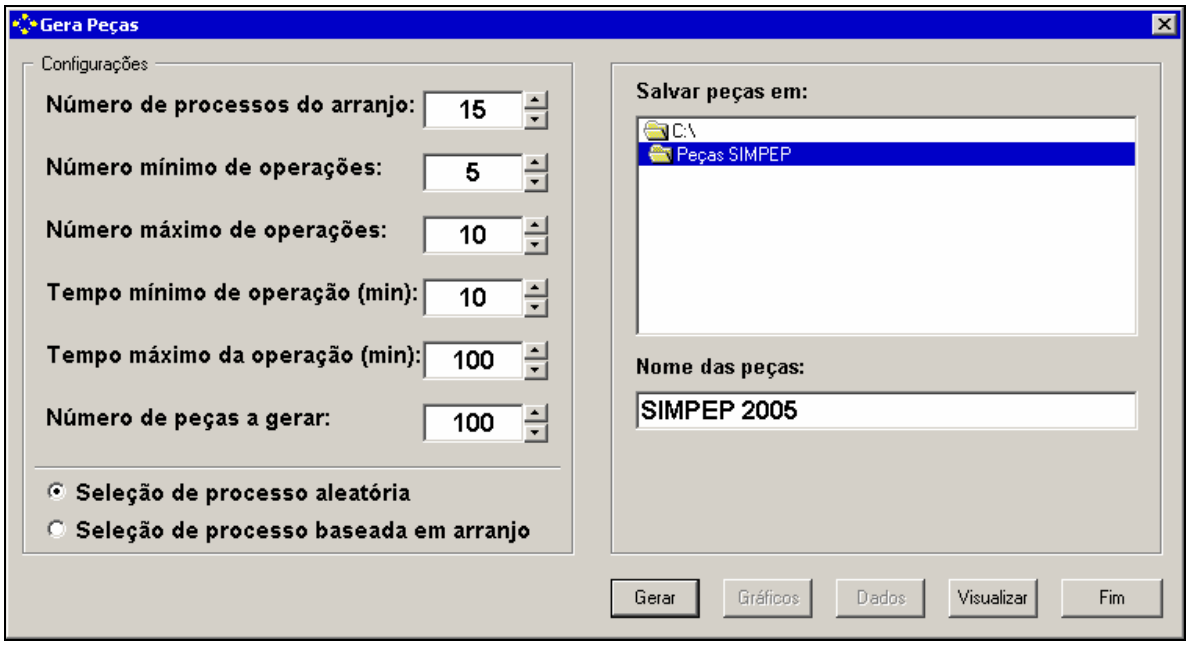


Figura 6 – Janela inicial do programa.

Usando o botão *Gerar* inicia-se o processo e ao final tem-se a janela de informação apresentada pela Figura 7. O exemplo apresentado gerou 100 peças e consumiu menos de 3 segundos de processamento em um Athlon XP 2600+ de 2.08 GHz e 1 Gb de RAM.

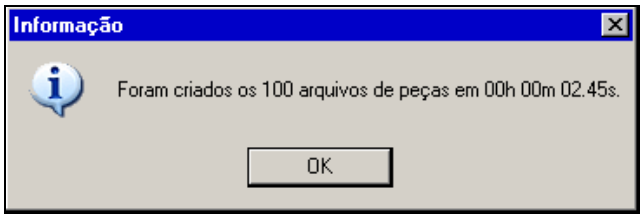


Figura 7 – Informação ao término do processo.

Acionar o botão OK da janela de informação faz com que ela seja removida e, na janela principal, os botões *Gráficos* e *Dados* ficam disponíveis enquanto o botão *Gerar* fica indisponível (até que uma das configurações seja alterada).

O botão *Gráficos* leva a três gráficos que auxiliam na avaliação do conjunto de peças gerado. O primeiro gráfico (Figura 8) apresenta o número de operações de cada peça. As linhas vermelhas indicam os números mínimo e máximo de operações que foram selecionadas pelo usuário. O segundo gráfico (Figura 9) permite avaliar a distribuição das peças com relação ao número de operações, ou seja, mostra quantas peças foram criadas para cada quantidade de operações entre o mínimo e o máximo. O terceiro gráfico (Figura 10) mostra quantas vezes cada processo disponível no arranjo foi usado para compor as peças.

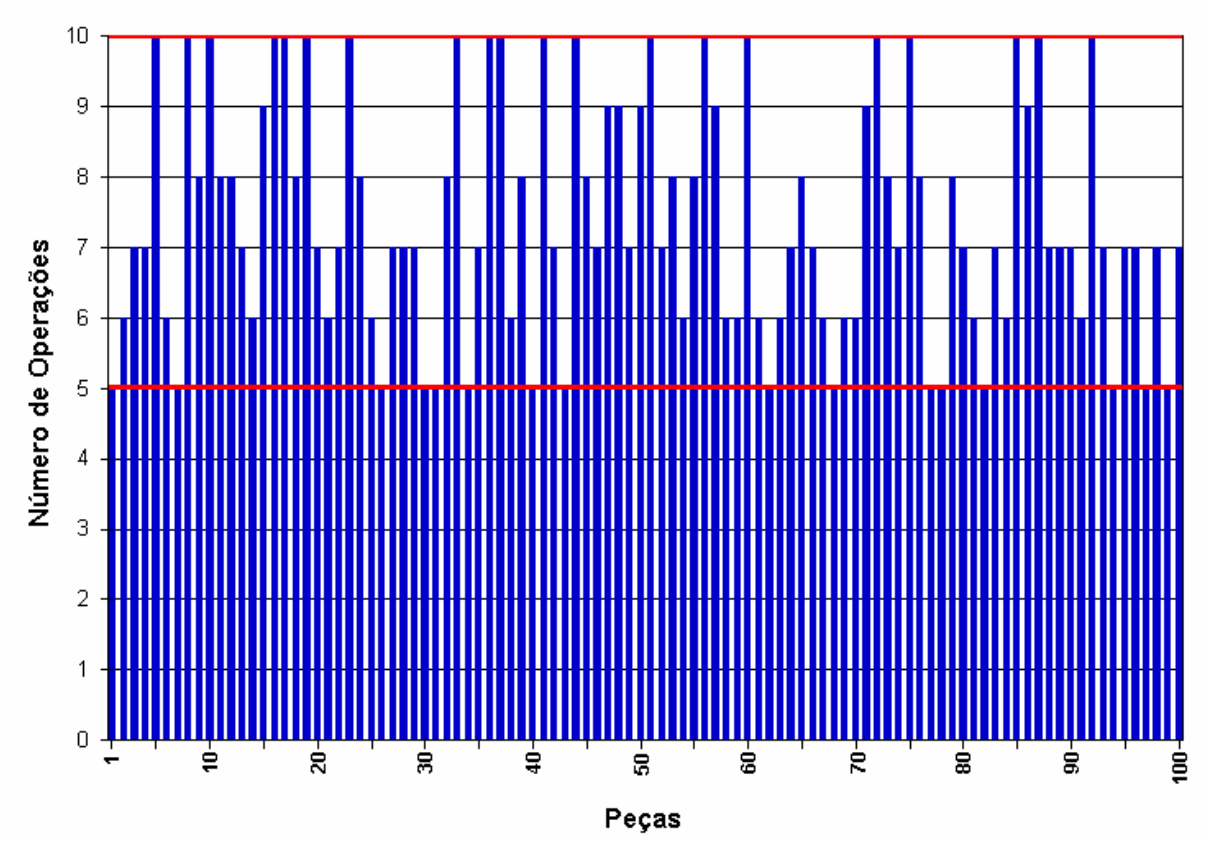


Figura 8 – Número de operações de cada peça.

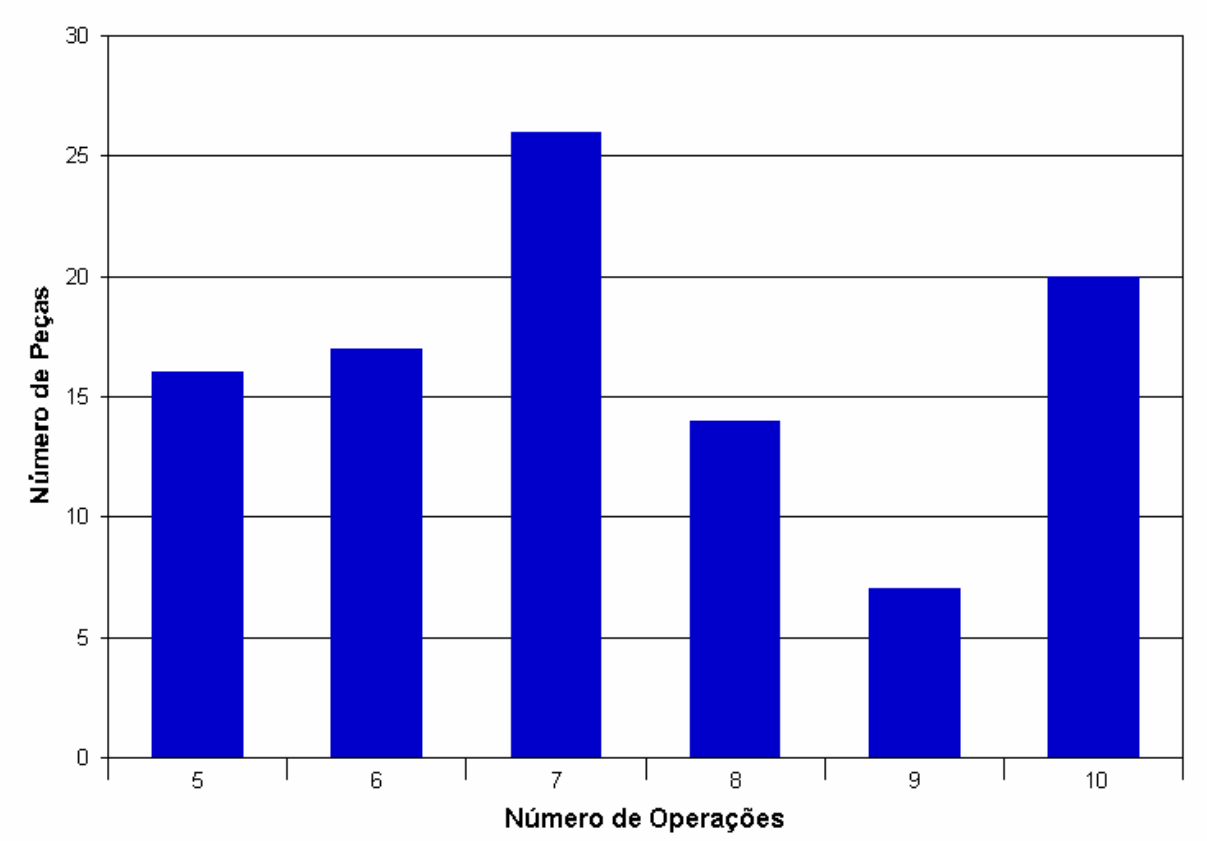


Figura 9 – Número de peças geradas para cada número de operações.

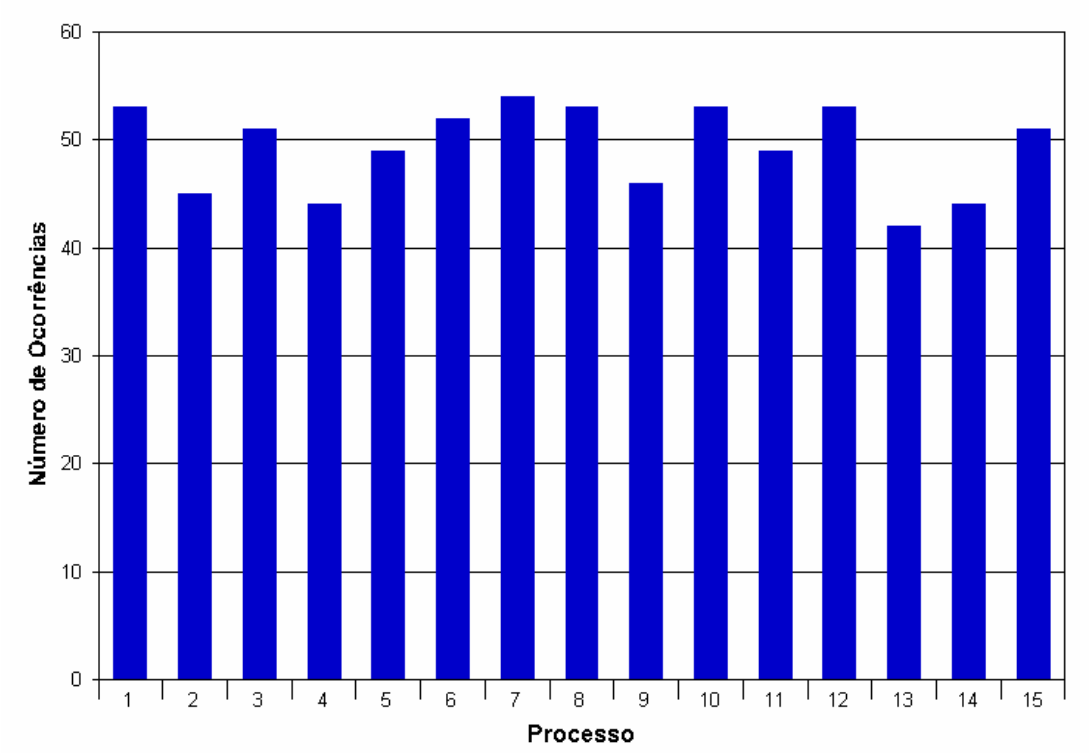
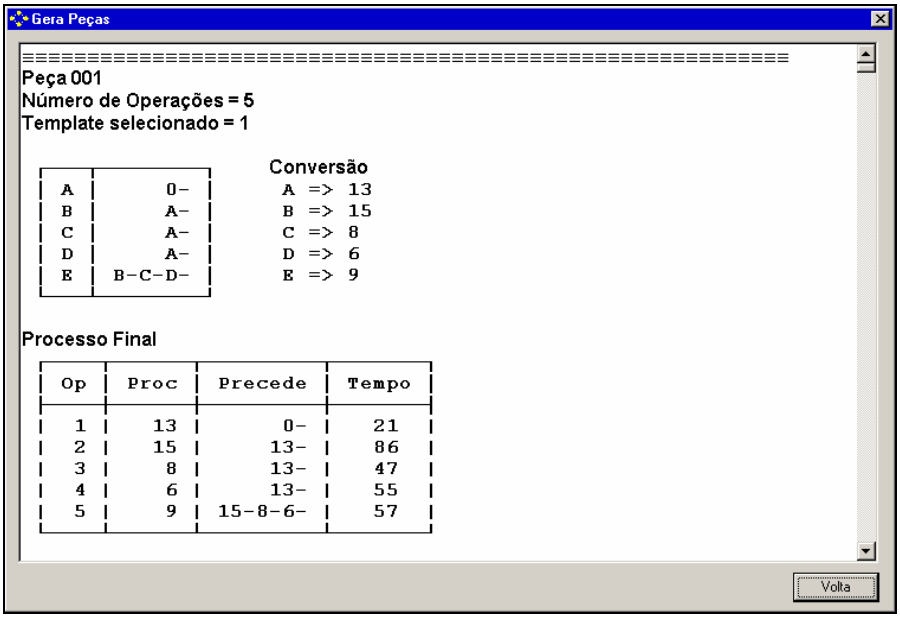


Figura 10 – Número de vezes que cada processo foi atribuído.

O botão *Dados* apresenta informações sobre as peças geradas (Figura 11). A primeira peça gerada tem 5 operações (como pode ser comprovado pelo gráfico da Figura 8). O modelo de estrutura (*template*) selecionado foi o primeiro. Em seguida é apresentada a tabela com as relações de precedência modelo e, ao seu lado, a conversão sorteada. Neste caso tem-se que para a operação A foi selecionado o processo 13, para a B o processo 15 e assim por diante. O processo final com o tempo de cada operação é mostrado em uma tabela na seqüência.



A	0-
B	A-
C	A-
D	A-
E	B-C-D-

Op	Proc	Precede	Tempo
1	13	0-	21
2	15	13-	86
3	8	13-	47
4	6	13-	55
5	9	15-8-6-	57

Figura 11 – Dados das peças do conjunto gerado.

Esses dados são automaticamente salvos em um arquivo RTF (Rich Text File), compatível com o editor de texto Word e similares, junto com os arquivos das peças (Figura 12). O uso de um arquivo RTF deve-se ao fato de serem necessárias duas fontes tipográficas para mostrar corretamente os dados (Arial e MS LineDraw).

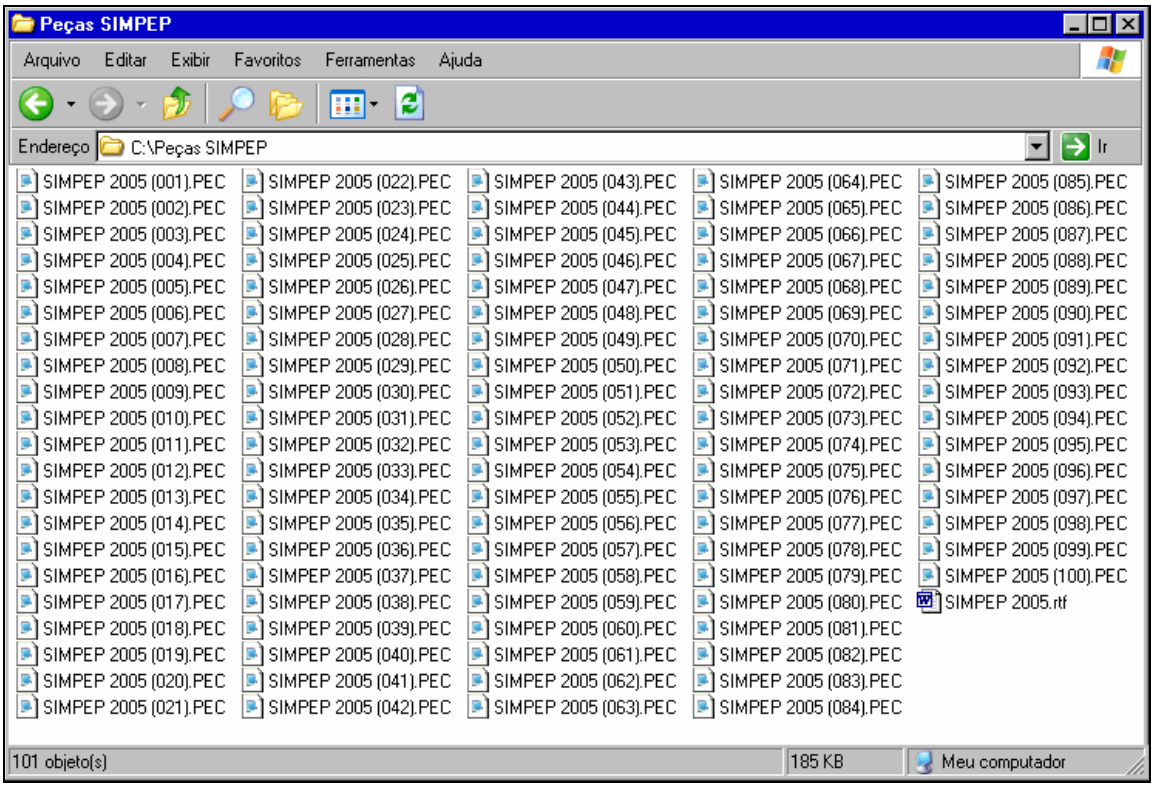


Figura 12 – Arquivos de peças e arquivo de dados.

O botão *Visualizar* leva até a janela mostrada pela Figura 13. Aqui é possível conferir graficamente o resultado obtido para cada peça, juntamente com o tempo de cada operação. Os controles da parte inferior direita permitem alterar a visualização (tamanho e posição).

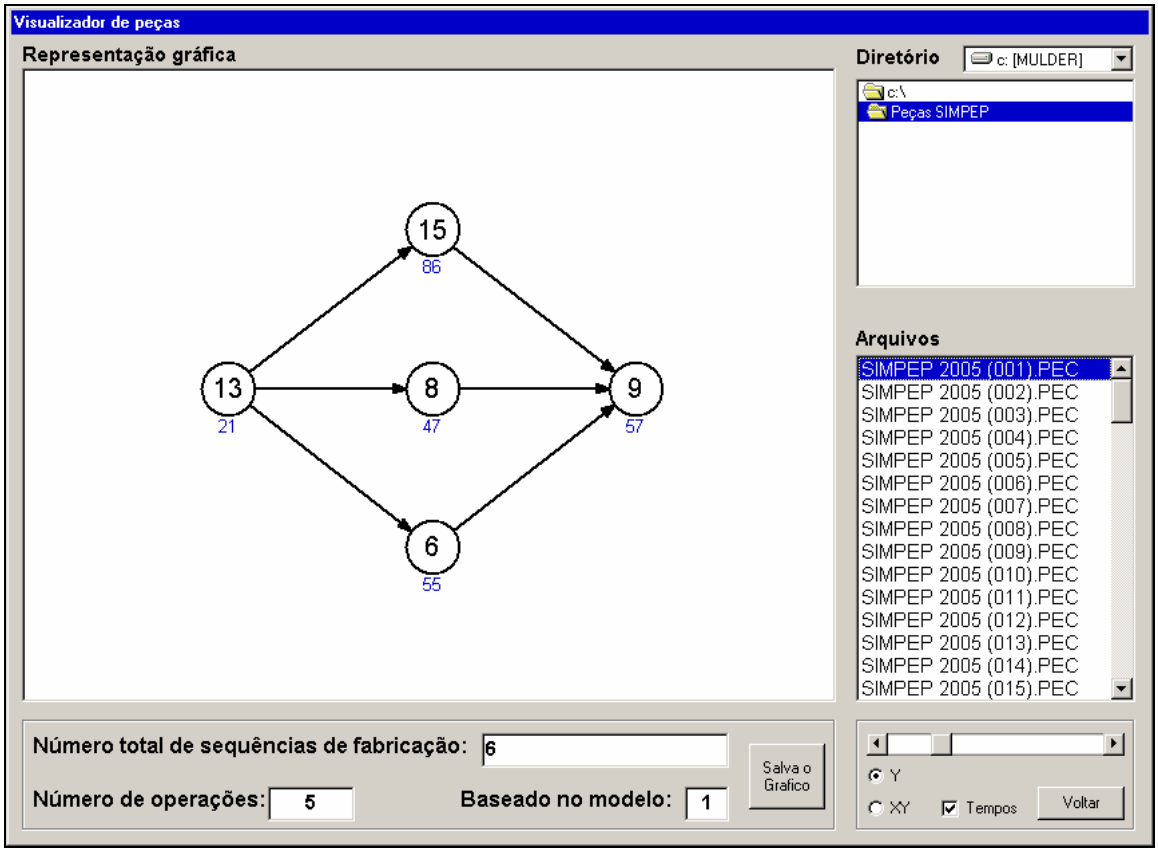


Figura 13 – Visualização gráfica da peça 001.

A Figura 14 mostra o conteúdo do arquivo 'SIMPEP 2005 (001).PEC' que está sendo visualizado graficamente pela Figura 13. A primeira linha, que será descartada no processo de simulação, traz informações sobre o processo de geração. A linha seguinte informa o número de operações e as demais linhas trazem as operações, seus predecessores e seus tempos.

```
Gerado automaticamente em 29/08/2005 às 14:33:12. Baseado no modelo  1
5
13, 0-, 21
15, 13-, 86
8, 13-, 47
6, 13-, 55
9, 15-8-6-, 57
```

Figura 14 – Exemplo de um arquivo de peça.

6. Resultados obtidos

Os resultados obtidos mostraram-se plenamente adequados às situações onde o número de máquinas de cada processo é igual. Pode-se afirmar isso pois a probabilidade de selecionar qualquer um dos processos é a mesma, e isso é comprovado pelo gráfico da Figura 10. Porém, em arranjos com números diferentes de máquinas para cada processo, espera-se que os processos com menor número de máquinas sejam menos utilizados pelas peças do que os processos com maior quantidade de máquinas. Isso levou a inclusão de uma segunda forma de seleção de processos, como se vê na parte inferior esquerda da Figura 6 (Seleção de processo baseada em arranjo).

Esta opção irá ler um arquivo que contém as informações de um arranjo, como mostra a Figura 15. Os dados desse arranjo foram obtidos do trabalho de Montreuil e Vankatadri (1991). Como mostra a figura há 15 tipos de processos sendo que alguns possuem apenas 1 máquina enquanto outros possuem até 7 máquinas.

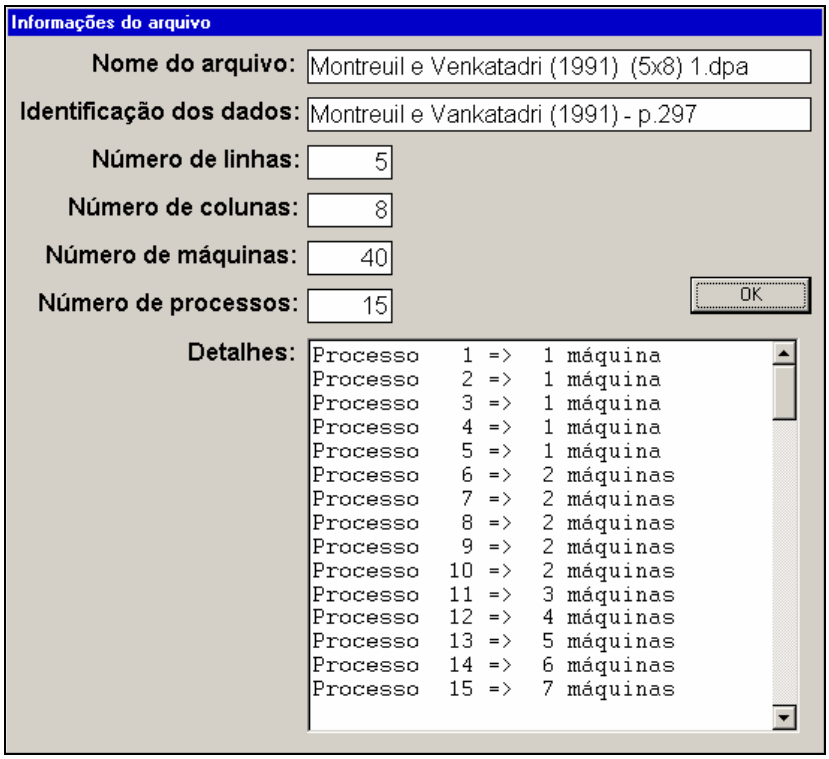


Figura 15 – Informações de um arranjo.

O procedimento usado para selecionar os processos com diferentes probabilidades foi baseado no *Método da Roleta*, usado em Algoritmos Genéticos (AG). Esse método faz com que os indivíduos mais adaptados tenham maior probabilidade de serem selecionados enquanto os menos adaptados tenham menor probabilidade. É como se cada indivíduo possuísse uma fatia de uma circunferência que é proporcional ao seu índice de aptidão. Na adaptação realizada os processos com maior número de máquinas tem maior probabilidade de serem selecionados do que os processos com menos máquinas.

A Figura 16 mostra o gráfico referente a um conjunto de 100 peças gerado com as mesmas configurações do primeiro exemplo, ou seja: 15 tipos de processo, peças entre 5 e 10 operações, tempo entre 10 e 100 minutos. Nota-se que, diferente do que ocorreu no primeiro grupo de peças (e que pode ser visto pela Figura 10), o número de operações que fazem uso de processos com menos máquinas é bem menor que os processos com maior número de máquinas.

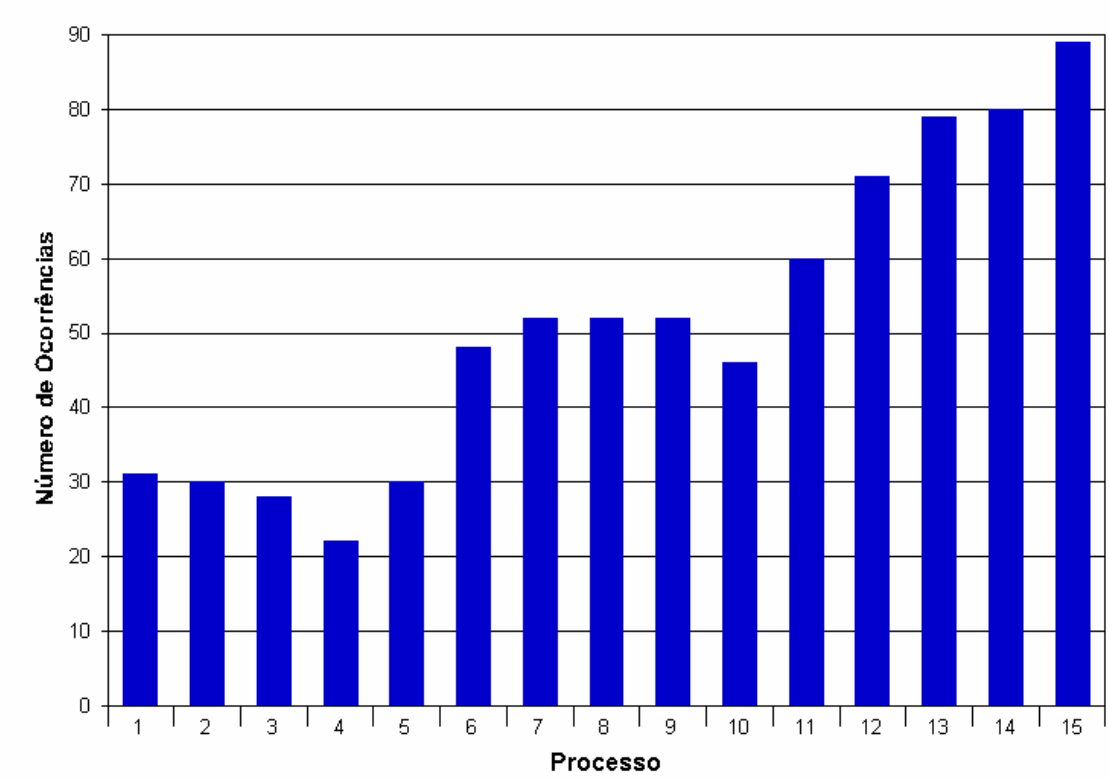


Figura 16 – Número de vezes que cada processo foi atribuído.

Essa nova forma de gerar os dados faz com que o conjunto de peças obtido seja mais compatível com a realidade de uma instalação de manufatura: processos pouco usuais participam menos das etapas de fabricação das peças.

7. Comentários finais

Este artigo mostrou uma forma de gerar automaticamente peças com flexibilidade de processo e número variado de operações.

A adoção de arquivos de texto para os modelos de estruturas de precedência permite alterar facilmente as relações já definidas. A remoção de estruturas existentes ou a inclusão de novas também é igualmente simples. Foram criadas estruturas para peças de 4 até 15 operações. Para aumentar essa faixa basta criar novos arquivos sendo que o importante é não deixar faltar nenhum arquivo entre o mínimo e máximo. Caso isso ocorra o programa informará que esse arquivo não foi encontrado.

A seleção de processo baseada na quantidade de máquinas disponíveis no arranjo, inspirado no método da roleta usado em algoritmos genéticos, permitiu obter uma solução mais próxima da realidade quando o arranjo é composto por processos com diferentes números de máquinas.

O próximo passo relacionado à pesquisa será implementar um sistema de programação (*scheduling*) que simulará a execução das peças geradas pelo procedimento aqui apresentado em vários arranjos (funcionais, parcialmente distribuídos, aleatoriamente distribuídos e maximamente distribuídos).

Referências

BAYKASOGLU, A. *Capability-Based Distributed Layout Approach for Virtual Manufacturing Cells*. International Journal of Production Research, v.41, n.11, 2003, p.2597-2618.

BENJAAFAR, S.; HERAGU, S. S.; IRANI, S. A. *Next Generation Factory Layouts: Research Challenges and Recent Progress*. Interfaces, v.32, n.6, November-December. 2002, p.58-76.

BENJAAFAR, S.; SHEIKHZADEH, M. *Design of Flexible Plant Layouts*. IIE Transactions, v.32, n.4, 2000, p.309-322.

BORENSTEIN, D. *A Direct Acyclic Graph Representation of Routing Manufacturing Flexibility*. European Journal of Operational Research, n.127, 2000, p.78-93.

GONDRAN, M; MINOUX, M; VAJDA, S. *Graphs and Algorithms*. Wiley-Interscience, Series in Discrete Mathematics, 1984, 670 p.

GROOVER, M. P. *Automation, Production Systems, and Computer-Integrated Manufactuirng*. Prentice-Hall, 1987, 856 p.

HUTCHINSON, G. K.; PFLUGHOEFT, K. A. *Flexible Process Plans: Their Value in Flexible Automation Systems*. International Journal of Production Research, v.32, n.3, 1994, p.707-719.

IRANI, S. A.; HUANG, H. *Custom Design of Facility Layouts for Multi-Product facilities Using Layout Modules*. IEEE Transactions. Robotics Automation, v.16, 2000, p.259-267.

LAHMAR, M.; BENJAAFAR, S. *Design of Distributed Layouts*. IIE Transactions, v.37, 2005, p.303-318.

MONTREUIL, B.; VENKATADRI, U. *Scattered Layout of Intelligent Job Shops Operating in a Volatile Environments*. Proceedings of the International Conference on Computer Integrated Manufacturing, Singapore, 1991, p.295-298.

MONTEUIL, B.; LEFRANÇOIS, P; MARCOTTE, S.; VENKATADRI, U. *Layout for Chaos – Holographic Layout of Manufacturing Systems Operating in Highly Volatile Environments*. Document de Travail 93-53, Faculté des Sciences de L'Administration, Université Laval, Québec, Canadá, 1993, 25 p.

RHEAULT, M; DROLET, J. R.; ABDULNOR, G. *Physically Reconfigurable Virtual Cells: A Dynamic Model for a Highly Dynamic Environment*. Computers and Industrial Engineering, v.29, n.1-4, 1995, p.221-225.

ROHDE, L. R.; BORENSTEIN, D. *Representação em Espaço de Estados para a Flexibilidade de Roteamento*. Gestão e Produção, v.11, n.2, 2004, p.251-261.

SARKER, B. R.; Li, Z. *Job Routing and Operations Scheduling: a Network-Based Virtual Cell Formation Approach*. Journal of Operational Research Society, n.52, 2001, p.673-681.

VENKATADRI, U; RARDIN, R. L.; MONTREUIL, B. *A Design Methodology for the Fractal Layout Organization*. IEE Transactions., v.29, n.10, 1997, p.911-924.