Novos métodos para a elaboração de layouts distribuídos visando minimizar o grau de distribuição a um tempo computacional satisfatório

Shih Yung Chin (USP-São Carlos, EESC, Depto. Engenharia Mecânica, SP, Brasil) – sychin@sc.usp.br
• USP – Av. Trabalhador São-Carlense, 400, Centro, CEP: 13566-590, São Carlos-SP
Eduardo Vila Gonçalves Filho (USP-São Carlos, SP, Brasil) – evila@sc.usp.br

Recebido em: 30/06/08 Aprovado em: 20/11/08

Resumo
O bom desempenho operacional do layout maximamente distribuído para o processamento de um conjunto de peças (com ou sem flexibilidade de processo) em relação aos demais layouts distribuídos se deve ao fato de que máquinas do mesmo tipo possuem bom espalhamento. Entretanto, a obtenção deste tipo de layout (e seu respectivo grau de distribuição) pelo algoritmo genético demanda elevado tempo computacional quando o layout é de porte grande (muitas máquinas). O layout aleatoriamente distribuído, apesar de demandar menos tempo para a geração, é o que mais aproxima dos resultados do maximamente distribuído em relação aos layouts parcialmente distribuídos e funcionais, porém ainda com desempenho operacional um pouco inferior. O objetivo deste trabalho é apresentar novas metodologias para a geração dos layouts distribuídos de grande porte que possam satisfazer tanto o desempenho operacional quanto o tempo na geração dos layouts. Para comprovar a eficácia dos métodos apresentados, resultados computacionais são coletados e comparados entre si variando-se os tamanhos de layouts e tipos de máquinas.
Palavras-chave: *Layouts distribuídos; Graus de distribuição; Tempos de CPU.*

Abstract
Maximally distributed layout has exceptional operational performance to process parts (considering processing flexibility or not) in comparison to other types of distributed layouts due to the fact of good machine distribution. Nevertheless, most size of it (and its distribution degree) demands huge computational time when done the traditional way (genetic algorithm). Randomly distributed layout requires less computational effort and it is more efficient than partially and processed layouts since its results are quite inferior to the maximally distributed layout. The main goal of this paper is to present new methodologies for elaborating large distributed layouts to satisfy operational performance in a low computational time. Computational results are presented and compared with each other by varying the size of layouts and types of machines to illustrate the effectiveness of presented methods.
Key-words: *Distributed layouts; Distribution degree; CPU time.*

GEPROS. Gestão da Produção, Operações e Sistemas – Ano 3, nº 3, Jul-Set/2008, p. 117-132

1. INTRODUÇÃO

Os layouts distribuídos são assim denominados por terem máquinas de diferentes tipos espalhadas no chão de fábrica. São concebidos para atuarem em um ambiente instável (produtos e quantidades variando constantemente), dispondo máquinas de tal forma que peças tenham o menor deslocamento possível durante o processamento.

Este artigo tem como objetivo apresentar três novas metodologias para a geração de layouts distribuídos de grande porte, cujo espalhamento é baseado apenas em quantidades de máquinas, que satisfaçam tanto o tempo CPU quanto desempenho operacional. A primeira busca melhorar a solução do aleatoriamente distribuído fazendo remover as máquinas que constantemente se repetem nas linhas (e colunas), principalmente aquelas que estejam encostadas uma das outras, dando assim maiores chances para os demais tipos de máquinas serem usadas na alocação. A segunda é bem similar à primeira, porém a escolha da máquina (do conjunto das máquinas removidas) é aquela que resulte no menor grau de distribuição. E por fim, a terceira, a escolha das máquinas para alocação é baseada sempre naquela que resulte no menor grau de distribuição. Ou seja, todas as máquinas são testadas em cada local por vez e aquela que fornecer o menor grau de distribuição é então mantida. Todos os métodos são implementados na linguagem de programação e os resultados (tempo CPU e grau de distribuição) dos métodos são comparados entre si.

2. METODOLOGIA DE PESQUISA

Este tópico apresenta brevemente a literatura sobre a metodologia científica e, a partir disso, define-se as mais adequadas para que os objetivos deste artigo possam então ser alcançados.

Cervo & Bervian (1983) comentam que, em geral, existem três tipos de pesquisa: bibliográfica (permite conhecer e analisar as contribuições existentes sobre o tema), descritiva (observa, analisa e correlaciona as variáveis sem manipulá-las) e experimental (o pesquisador controla as variáveis do objeto em estudo). Para Gil (1987), a pesquisa é desenvolvida com base nos conhecimentos disponíveis, com a utilização cuidadosa de métodos, técnicas e outros procedimentos científicos. Com relação a coleta de dados, Chizzotti (1991) afirma que existem basicamente dois tipos de dados: quantitativa (mensuração das variáveis pré-estabelecidas, procurando verificar e explicar sua influência sobre outras variáveis) e qualitativa (fundamenta-se em dados coligidos nas interações interpessoais, na co-participação das situações dos informantes, analisadas a partir da significação que estes dão aos seus atos).

Conforme a apresentação da metodologia científica, este trabalho adota o tipo de pesquisa de natureza bibliográfica, fazendo revisão da literatura referente ao layout distribuído sobre os problemas enfrentados para a geração de layouts de grande porte. O método para a geração dos layouts bem como obtenção dos resultados é simulação computacional e os dados coletados são de natureza quantitativa. Estes foram adotados porque julgou-se como suficientes para alcançar aos objetivos propostos.

3. REVISÃO BIBLIOGRÁFICA

Benjaafar & Sheikhzadeh (2000) fazem comparações de desempenho entre os layouts distribuídos (parcialmente distribuído e seus respectivos graus de desagregação) em relação ao funcional. Concluíram que, ao fazer a primeira desagregação a partir do funcional, é possível obter grandes reduções de fluxo. À

medida que o grau de desagregação aumenta, esta redução de fluxo passa a não ser tão significativa. Em seguida, Lahmar & Benjaafar (2002) apresentam um estudo comparativo e concluíram que o parcialmente distribuído, além de superar o funcional em termos de desempenho operacional, supera o aleatoriamente distribuído. Além disso, o desempenho do parcialmente se aproxima do maximamente.

Aparentemente, os estudos levam a concluir que o parcialmente distribuído é a solução inclusive até mesmo para substituir o próprio maximamente distribuído. Entretanto, Júnior & Filho (2007), analisando sobre a questão da flexibilidade de processamento de peças, concluíram que o aleatoriamente se sobressai em relação ao parcialmente distribuído e funcional, porém o maximamente se sobressai em relação à todos.

Os trabalhos citados levam em consideração layouts de pequeno porte (abaixo de 50 máquinas) e até 10 tipos diferentes de máquinas. Para layouts grandes, maiores são as opções de alocação dos grupos de máquinas de mesmo tipo no chão de fábrica e, portanto, maiores deslocamentos de peças. Isso quer dizer que o desempenho do parcialmente distribuído e funcional será prejudicado. Este problema já não é visto em aleatoriamente e nem em maximamente distribuídos.

Comumente, em projetos de layouts faz se do uso de algoritmo genético (AG), fundamentada em uma função-objetivo de minimização ou maximização. No caso do funcional, por exemplo, não é necessário aplicar AG porque o agrupamento do mesmo tipo de máquina acaba restringindo o número de possibilidades. O mesmo acontece para parcialmente distribuído. Quanto ao maximamente distribuído, é necessário fazer uso desta ferramenta porque existem várias combinações de máquinas em cada local do chão de fábrica. A função-objetivo usada também é chamada de grau de distribuição. Quanto menor o grau de distribuição, melhor o espalhamento das máquinas e, portanto, melhor desempenho operacional.

O grande problema para a obtenção do maximamente pelo AG é quando se trata de grandes quantidades de máquinas, o que resulta em elevado consumo de tempo computacional (CPU) para a obtenção de um arranjo físico e grau de distribuição satisfatórios. O aumento dos tipos de máquinas também contribui no aumento do tempo de CPU, mas não é significativo. Júnior (2006) comenta que para gerar o maximamente pelo AG do arranjo de Montreuil et al. (1993), contendo um total de 1296 máquinas (e 125 tipos de máquinas), o computador levaria cerca de 84 dias obtendo um grau de distribuição de valor 902,0626. Por outro lado, o consumo do tempo computacional para gerar o aleatoriamente distribuído é praticamente desprezível, porém mesmo com desempenho operacional pior apresenta bons resultados em comparação aos demais layouts distribuídos.

Por causa da ineficiência de AG, Montreuil e Venkatadri (1991) apresentam um procedimento heurístico para o desenvolvimento de um layout distribuído. A primeira etapa desta heurística encontra as posições ideais de cada tipo de máquina. Considera-se posições ideais como sendo aquelas que permitam o mesmo distanciamento entre os mesmos tipos de máquinas. Como podem existir sobreposições de máquinas distintas que ocupe a mesma posição ideal, deve-se ter um critério de desempate para definir qual máquina a ser priorizada na alocação (que é a segunda etapa). Mais tarde, Montreuil et al. (1993) nomearam esta heurística de TARGET.

Os critérios de desempate, adotados pelos autores para a escolha das máquinas, foram baseados na taxa esperada de utilização e lote esperado para processamento. Contudo, estas informações são difíceis de serem obtidas já que seria necessário analisar as curvas esperadas de chegada e de atendimento (processamento) de peças em cada máquina. Ou seja, por estar atuando em um ambiente instável, não se sabe quais peças serão solicitadas pelos clientes e, por causa disso, torna-se imprecisa a definição sobre quais tipos de máquinas serão usadas no processamento.

Devido aos problemas para a geração de layouts de grande porte, este artigo apresenta, então, três novas metodologias (detalhadas nos sub-tópicos 4.1, 4.2 e 4.3).

4. MÉTODOS PROPOSTOS

Júnior (2006) utiliza um critério de desempate baseado apenas nas quantidades de cada tipo de máquina (as máquinas de menor quantidade são priorizadas para serem alocadas nas posições ideais; livres e mais próximas do centro, caso houver disputa). Esta metodologia é denominada ALVO. Após a execução do método proposto, tanto para processamento rígido quanto para processamento flexível das peças, os autores concluíram que o tempo CPU é praticamente desprezível na geração dos layouts e os graus obtidos são levemente superiores (pior solução) ao maximamente do AG, mas que, se comparado aos demais distribuídos, tem bons resultados resultado. O problema é que ALVO só pode ser usado para réplicas distintas de máquinas. Os três métodos apresentados neste trabalho servem tanto para réplicas distintas quanto para iguais e sem lidar com nenhum tipo de critério de desempate oriunda de sobreposições.

Inicialmente, o usuário deve inserir o número de tipos de máquinas e réplicas. Ao ler as informações das réplicas de cada máquina, dependendo do método proposto, deve-se gerar layout aleatoriamente distribuído como comentados nos sub-tópicos seguintes.

4.1. Método 1 – Escolha aleatória de máquinas removidas

Este método surgiu ao se questionar sobre a possibilidade de aumentar o desempenho do aleatoriamente distribuído, melhorando o seu método de espalhamento de máquinas do mesmo tipo. Isso porque, ao gerar amostras deste tipo de layout, muitas máquinas do mesmo tipo ainda tinham chances de encostar uma nas outras. Caso isso fosse evitado, melhor seria o espalhamento. É claro que quanto maior a diferença na quantidade entre os tipos de máquinas, maior é a chance de encostamento. Um novo procedimento proposto consiste em balancear os tipos de máquinas nas linhas e colunas do chão de fábrica. A idéia é dar prioridade às máquinas que ainda não foram alocadas, evitando repetições do mesmo tipo de máquina. A proposta consiste de 4 passos:

- **Passo 1)** Aplicar procedimento de geração como se fosse de um layout aleatoriamente distribuído deixando as máquinas já na configuração do chão de fábrica;
- **Passo 2)** Contar o número de máquinas de cada tipo em cada linha. Se o valor absoluto da diferença do número de máquinas de um tipo (A, por exemplo) em relação ao outro (B) for maior ou igual a dois, então remova uma máquina A que repete e preencher o local da máquina removida por zero. Repita o passo 2 quantas vezes forem necessárias. Fazer o mesmo para todas as linhas. Armazenar as máquinas removidas em um vetor;
- **Passo 3)** Criar, em linguagem de programação, um procedimento de seleção aleatória das máquinas deste vetor e inserir estas no chão de fábrica (regiões armazenadas como zero).
- **Passo 4)** Executar os passos 2 e 3 também para as colunas;

No passo 2 se a diferença for maior ou igual a 2, isso quer dizer que algum tipo de máquina está repetindo enquanto outras ainda não foram alocadas ainda. O procedimento proposto evita que máquinas do mesmo tipo esteja na mesma linha (ou coluna), mas não impede que isso ocorra porque o critério adotado no passo 3 é de escolha aleatória.

Para entender melhor o procedimento, considere o seguinte exemplo (figura 1). Sabe-se que no chão de fábrica deve possuir nove tipos de máquinas, denominados de m1,... e m9. Já é conhecido o número de réplicas de cada tipo de máquina, dados por Qm1=4, Qm2=1, Qm3=1, Qm4=4, Qm5=3, Qm6=3, Qm7=3, Qm8=2, Qm9=4.

A análise sempre inicia da seguinte forma: – (sinal de menos).
{|máq1-máq2|, |máq1-máq3|... |máq1-máq9|}; {|máq2-máq3|, |máq2-máq4|... |máq2-máq9|}; {E assim por diante}.

Aplicando o passo 1, temos:

6	1	9	4	4
7	5	1	7	6
3	4	9	5	9
2	1	6	1	4
8	8	5	9	7

Figura 1 – Geração aleatória do layout

A figura 1 apresenta uma vista superior do chão de fábrica, onde os números representam os tipos de máquinas. Na linha 1, por exemplo, existe um tipo de máquina A, cuja diferença em relação ao outro tipo B é maior ou igual a dois. Isso ocorre, por exemplo, entre as máquinas 4 e 2. Assim, existem 2 réplicas de máquina 4 e 0 réplica de máquina 2. O mesmo acontece entre as máquinas 4 e 5 e assim por diante. Ou seja, a máquina 4 está impedindo que outros tipos estejam próximas à ela. Por causa disso, remove-se a máquina 4. Caso houvesse uma terceira máquina 4 nesta mesma linha, esta também deveria ser removida. O vetor obtido de máquinas removidas (passo 3) é: {4, 7, 9, 1, 8}

O layout alterado será o apresentado na figura 2.

6	1	9	0	4
0	5	1	7	6
3	4	0	5	9
2	0	6	1	4
0	8	5	9	7

Figura 2 – Layout alterado

Note que no momento, visualizando apenas as linhas, a diferença entre os tipos de máquinas nunca é superior ou igual a 2. Este método também permite que cada linha tenha as mesmas quantidades de cada tipo de máquina, salvo se existir diferença entre as réplicas, ou seja, no exemplo acima, nem todas as máquinas estão presentes em todas as linhas justamente por causa da restrição do número de réplicas.

Deve-se agora escolher aleatoriamente uma máquina do vetor formado e preenchê-la no layout (onde constam os valores nulos). Considere que a ordem de escolha das máquinas obtida pela linguagem de programação seja {9, 7, 1, 4, 8 }. Assim, o novo layout ficaria como mostrado na figura 3.

6	1	9	9	4
7	5	1	7	6
3	4	1	5	9
2	4	6	1	4
8	8	5	9	7

Figura 3 – Novo layout

Aplicar o mesmo procedimento para as colunas (que é o passo 4).

O procedimento parece demandar elevado tempo, mas em alguns centésimos de segundos, uma amostra bastante grande de layout é obtida. Para fins de ilustração, os layouts obtidos pela linguagem de programação (duas amostras) aplicados a um conjunto de nove máquinas de quantidades 4, 6, 5, 4, 5, 4, 4, 5 e 5 respectivamente estão apresentados na figura 4. Note que nenhuma máquina se repete tanto na mesma linha quanto na mesma coluna.

9	2	7	8	3	6	4
2	9	8	1	6	5	3
1	6	2	9	5	4	8
8	5	1	3	9	2	7
6	8	4	2	7	3	5
7	4	3	5	2	9	1

9	8	2	6	5	7	1
7	3	1	2	9	5	8
2	4	6	9	3	8	5
8	7	4	5	2	3	9
3	1	9	8	4	2	6
5	6	3	1	7	4	2

Figura 4 - Exemplo de layouts finais

4.2. Método 2 – Escolha direcionada de máquinas removidas

Como dito anteriormente, o método 1 procura não alocar os mesmos tipos de máquinas juntos, mas não evita que isso ocorra. Por coincidência, nos exemplos do método 1 apresentados não foram vistos máquinas do mesmo tipo juntas. Quanto maior a diferença entre as réplicas das máquinas, elas juntas tendem a serem vistas com mais freqüência. Portanto, para minimizar este impacto, considerou-se a possibilidade de melhorar o espalhamento do método 1. Uma possível solução escolhida neste trabalho para minimizar este problema é escolher a máquina (entre as removidas) que resulte no menor grau de distribuição. Estão descritos abaixo os passos para a obtenção do layout pelo método 2. O número de iterações depende do número total de máquinas que foram removidas.

- **Passo 1)** Aplicar os passos 1 e 2 do método 1;
- **Passo 2)** Removidas ← leia o número total de máquinas removidas; {Conta quantas máquinas foram removidas}
- **Passo 3)** i←1; Melhor ←10000; {Ou um outro valor suficientemente grande}

  Se i <= Removidas então
  máq[i] ← ler máquina removida; {Identifica o tipo de máquina removido}
  i←i+1;
  repetir passo 3 até i>Removidas;
- **Passo 4)** P← Encontrar um local ainda não alocado; {Procura no layout os nulos}
- **Passo 5)** i←1;
- **Passo 6)** Alocar "temporariamente" máq[i] na posição P; {Testar a máquina no local não alocado}
- **Passo 7)** Grau de distribuição [i] ← Calcular o grau de distribuição para toda fábrica;

  Se Grau de distribuição [i] < Melhor então
  Melhor_máquina:=máq[i]; {Identifica a melhor máquina}
  Melhor ← Grau de distribuição[i]; {Armazena o menor grau de distribuição}

- **Passo 8)** Se i<>Removidas então ißi+1 e repetir os passos 6 e 7; Caso contrário, ir para o Passo 9; {Escolhe outras máquinas removidas}
- **Passo 9)** P ← Melhor_máquina; Removidasß Removidas-1; {Alocando a melhor máquina}
- **Passo 10)** Se Removidas<>0 então reler o passo 2 e repetir os passos 3 até 9 {Checa se ainda se existem máquinas removidas}; Caso contrário Fim.

4.3. Método 3 – Direcionamento total

Foi a partir do método 2 que surgiu a idéia de construir layout escolhendo apenas máquinas que resultem sempre em menor grau de distribuição. Assim, quando existirem M tipos de máquinas, são realizados M cálculos do grau de distribuição para cada local. Escolhe-se a máquina que resulte no menor grau de distribuição, obtendo, assim o layout todo. Para obter o layout do método 3, inicia-se a construção do layout a partir do centro do chão de fábrica. A seguir, estão descritos os passos para a implementação do método 3.

- **Passo 1)** Entre com a formatação do chão de fábrica em x e em y, e entre com o total de tipos de máquinas M a serem alocadas; {Inserção das configurações e os tipos de máquinas}
- **Passo 2)** Calcular o centro de x e y: {Escolha do centro da fábrica}

  Centro_x ← (1+x)/2;
  Centro_y ← (1+y)/2;
- **Passo 3)** Preencher os locais do chão de fábrica por zero; Adotar i←0; Total_máquinas←0; {Representa que nenhuma máquina ainda está alocada}
- **Passo 4)** Encontrar o local P mais próximo do centro definido no passo 2 e que seja 0; {Escolha da posição não alocada mais próxima do centro}
- **Passo 5)** Enquanto i<>M faça {Alocando as máquinas nas posições vazias}

  I←i+1;
  Total_máquinas←Total_máquinas+1;
  Alocar a máquina i no local P;
  Vá ao Passo 4;

  Caso contrário, ir ao Passo 6;
- **Passo 6)** i←0; Menor←10000 (ou um valor suficientemente grande); {Dado inicial de comparação}
- **Passo 7)** Repetir o passo 4;
- **Passo 8)** Enquanto i <> M faça {Escolhendo a máquina que resulte no menor grau de distribuição}

  I←i+1;
  Alocar "temporariamente" i ao local P;
  Calcular o grau de distribuição[i] para toda fábrica;
  Se grau de distribuição[i] <Menor então menor ← grau de distribuição[i];
  Melhor_i:=i;
  Vá ao Passo 8;
- **Passo 9)** Alocar máquina "Melhor_i" ao local_P; Total_máquinas ← Total_máquinas+1; {Alocando a máquina escolhida}
- **Passo 10)** Se Total_máquinas<>x*y então repetir passos 6, 7, 8 e 9 e 10; Caso contrário, fim. {Checando se existem mais locais não alocados}

5. ANÁLISE DE DESEMPENHO PARA RÉPLICAS DISTINTAS DE MÁQUINAS

Este tópico apresenta os resultados dos métodos propostos. Os resultados são tempo de CPU e graus de distribuição. Para a realização da coleta de dados, as metodologias podem ser codificadas em software de programação (neste trabalho adota-se Pascal 7.0), instalado em um hardware Intel Celeron 2,26Ghz, 768mb RAM. Outros pacotes computacionais ou hardwares também podem ser usados.

A literatura vem trabalhando apenas com quantidades distintas de máquinas. Para que tenha um parâmetro em comum de comparação, foram simulados, inicialmente em código de programação, os métodos para quantidades distintas. Considerou-se, durante a simulação, um total de 1300 e 130 máquinas (limite que o software consegue trabalhar). Na literatura, existe apenas um trabalho e este lida com no máximo layout tamanho 1296, portanto este limite é razoável. Como o total de máquinas é de 1300, a quantidade mínima tende a 10, assim, a quantidade das máquinas tendem a ser similares (por terem 130 tipos de máquinas, sendo 10 unidades de cada máquina). Considere que a menor quantidade seja 6 e a porcentagem escolhida seja 40%. Significa que 40% dos 130 tipos de máquinas possuem 6 réplicas de máquinas.

As figuras 5, 6, 7 e 8 mostram os desempenhos variando-se as quantidades de máquinas dos seguintes layouts: aleatoriamente, métodos 1, 2 e 3 respectivamente. Comparando as figuras 5 e 6, nota-se que o método 1 é sensivelmente melhor que o aleatoriamente em termos de grau de distribuição, salvo quando dos 80% das máquinas forem ≈2 unidades. Os tempos CPU para geração de ambos são desprezíveis (máx. 0,18s para o aleatoriamente distribuído e máx. 12s para o método 1).

O desempenho do método 2 (figura 7) é melhor que o aleatoriamente distribuído (figura 5) e que o método 1 (figura 6) , porém o tempo CPU é maior (no pior caso leva cerca de 42 minutos). À medida que as réplicas das máquinas se tornam similares, o grau de distribuição ainda continua sendo menor que os demais. A partir da análise das figuras 6 e 7 (para 80%), mostra-se claramente que quanto maior a diferença entre as réplicas, o método 1 não evita que máquinas similares estejam próximas (comprovando a hipótese da criação do método 2).

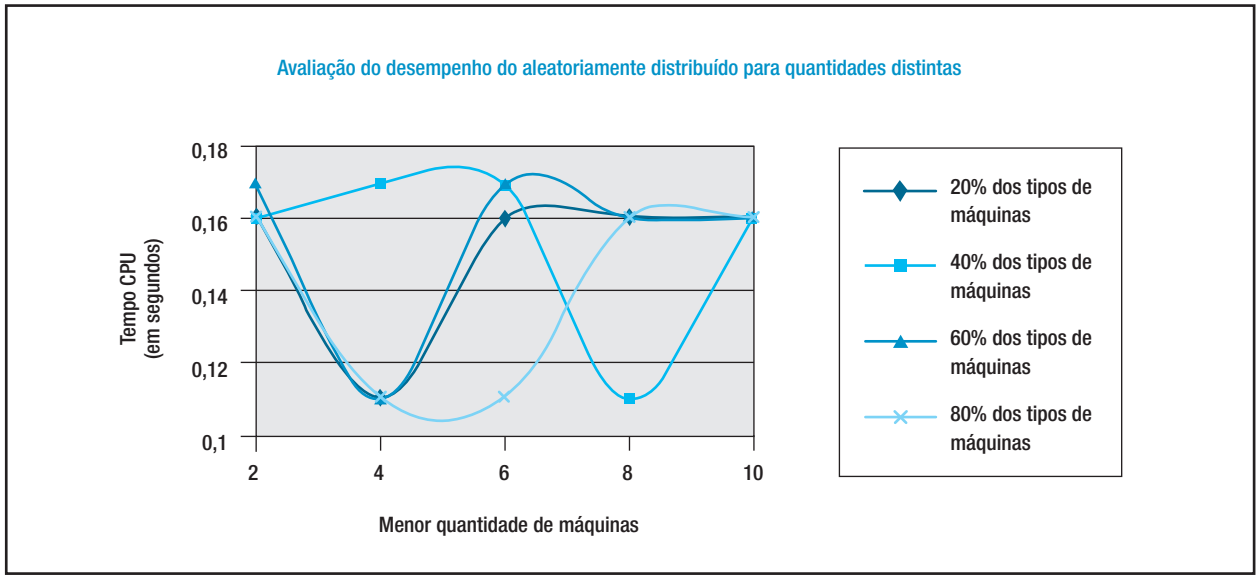
Em comparação ao método 2, o método 3 (figura 8) possui ainda menor grau de distribuição em todos os pontos, porém com tempo CPU ainda maior. É importante notar que, à medida que as réplicas se tornam similares, o tempo CPU acaba se tornando menor em comparação às réplicas distintas de máquinas do próprio método 3. Pode-se afirmar que o problema de elevado tempo de CPU não era por causa dos cálculos dos graus, mas porque no método tradicional (por AG) o computador passa muito tempo gerando dados (ou melhor, alocando máquinas) sem ter um critério de escolha de tipo de máquina, enquanto no proposto, a alocação das máquinas é de forma direcionada.

Note também, na figura 8, que devem ser evitadas réplicas distintas porque além de aumentar o tempo CPU, o grau de distribuição também aumenta. Outro ponto importante é que, fazendo as réplicas se tornarem cada vez mais similares, o grau de distribuição do método 2 pode ser tão bom quanto o método ALVO (conforme figura 7). É visível na figura 8 que, aumentando o número de réplicas de cada máquina, o grau de distribuição tende a se reduzir.

O resultado do método ALVO proposto por Júnior (2006) para o layout de Montreuil et al. (1993), o qual leva em consideração 1296 máquinas e 125 tipos de máquinas, resultou em 1m e 59s com um grau de distribuição de 919,1072. Se fosse pelo método tradicional (pelo AG), o grau de distribuição seria um pouco melhor (902,0626), porém levaria cerca de 84 dias. Ao executar o método 3 para as mesmas quantidades de máquinas de Montreuil et al. (1993), resultou em 908,34 e tempo de execução de 1h e 4min. Fazendo o mesmo para o método 2, resultou em grau de 1132,37 e tempo de 16,42s. Para o método 1, resultou em um grau de 1136,14 com um tempo de 1,55s e quanto ao aleatoriamente distribuído, grau de 1138,44 e

tempo de 1,2s. Infelizmente o método ALVO só pode ser usado para quantidades distintas (conforme as etapas indicadas em Júnior, 2006). Se fosse "forçada" a usar ALVO para quantidades de réplicas similares, tal metodologia tende a concentrar máquinas do mesmo tipo e na mesma região, reduzindo o desempenho operacional (aumento do grau de distribuição).

Em função do grau de distribuição ser menor nos três casos à medida que o número de réplicas se torna similar, deseja-se verificar se este fato acontece também ao variar o tamanho do layout e tipos de máquinas.



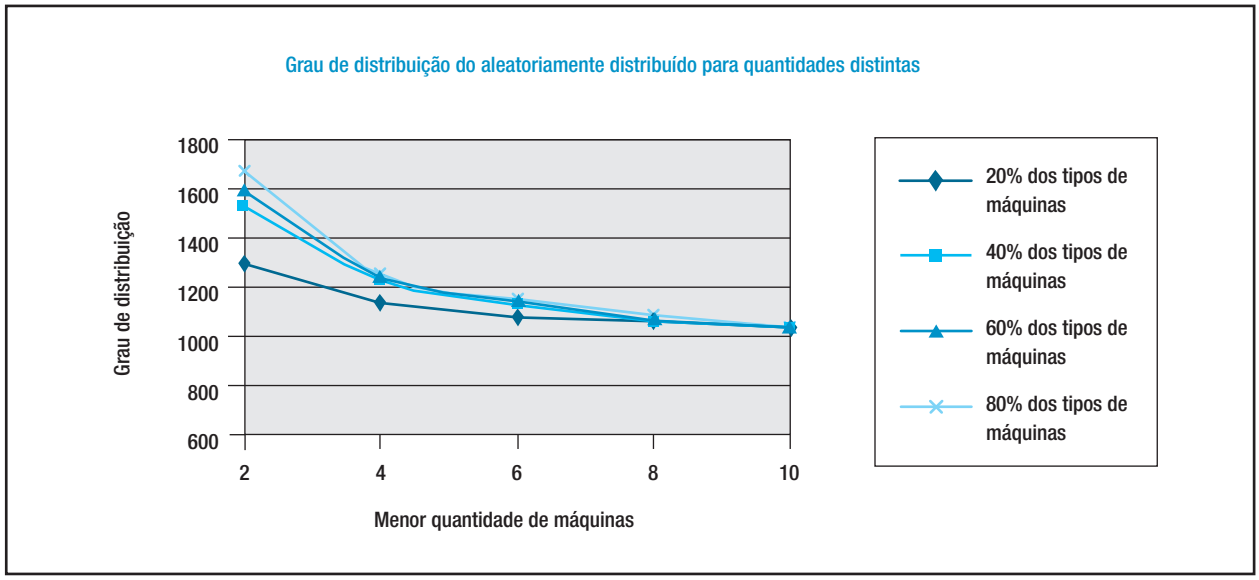


Figura 5 – Desempenho do aleatoriamente distribuído

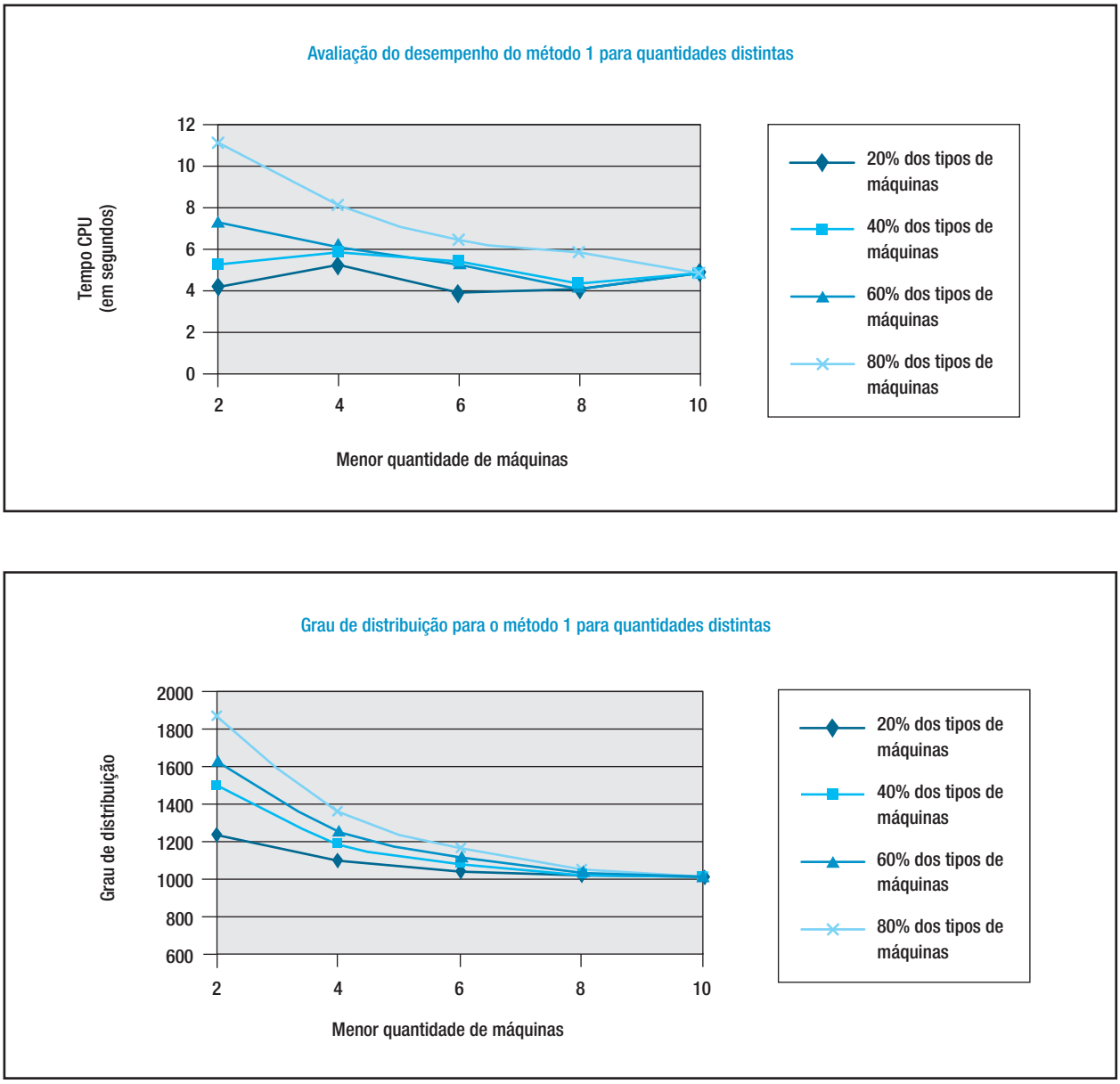
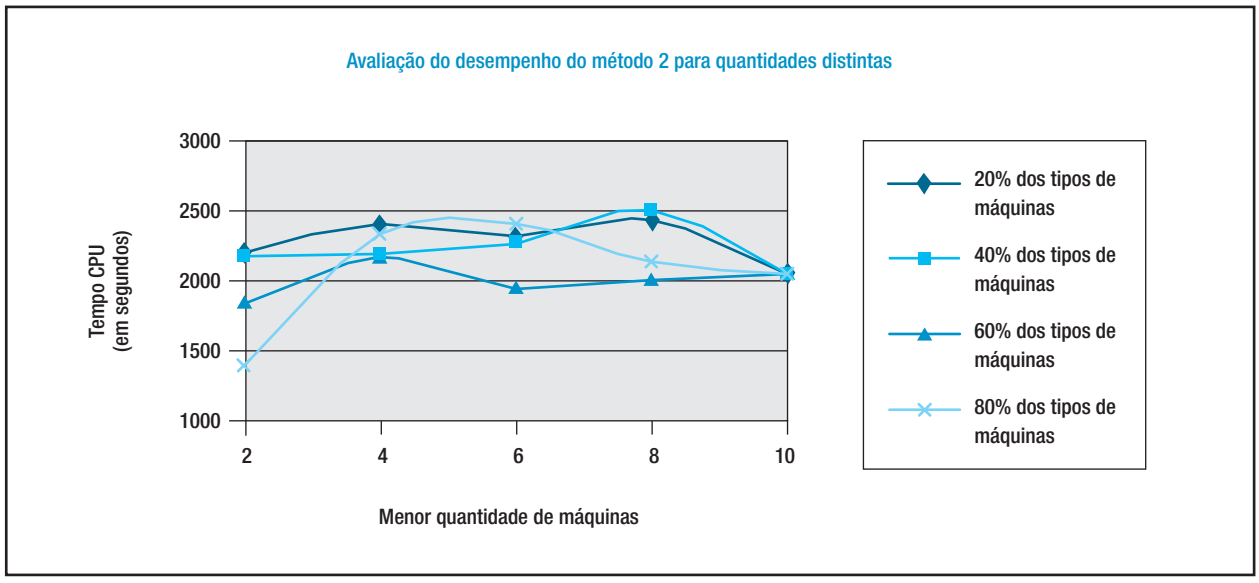


Figura 6 – Desempenho do método 1



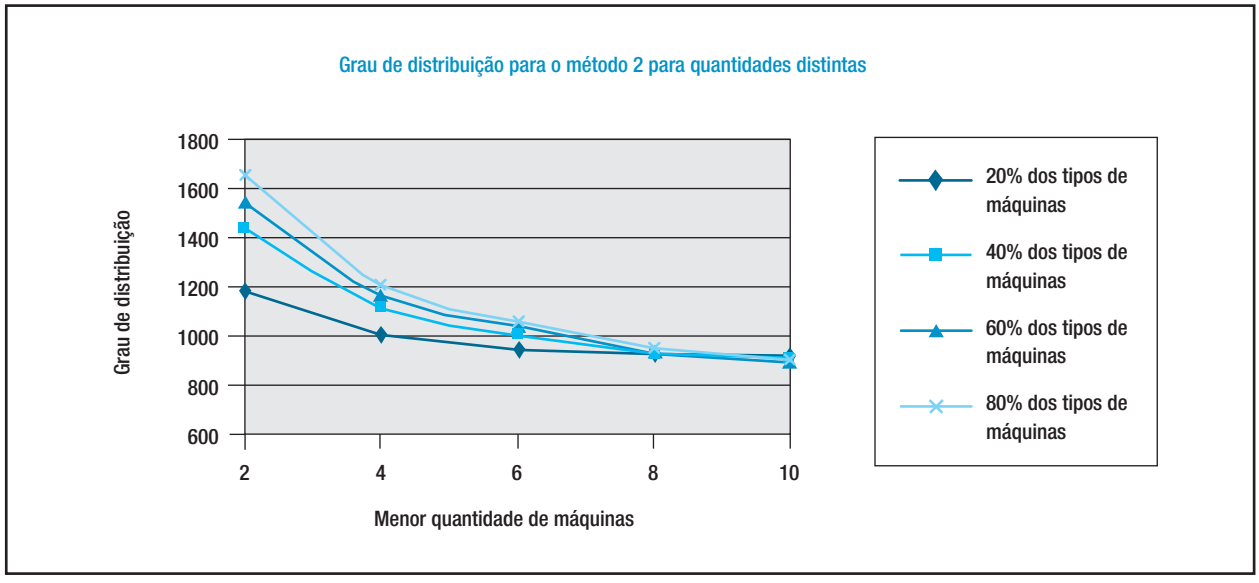


Figura 7 – Desempenho do método 2

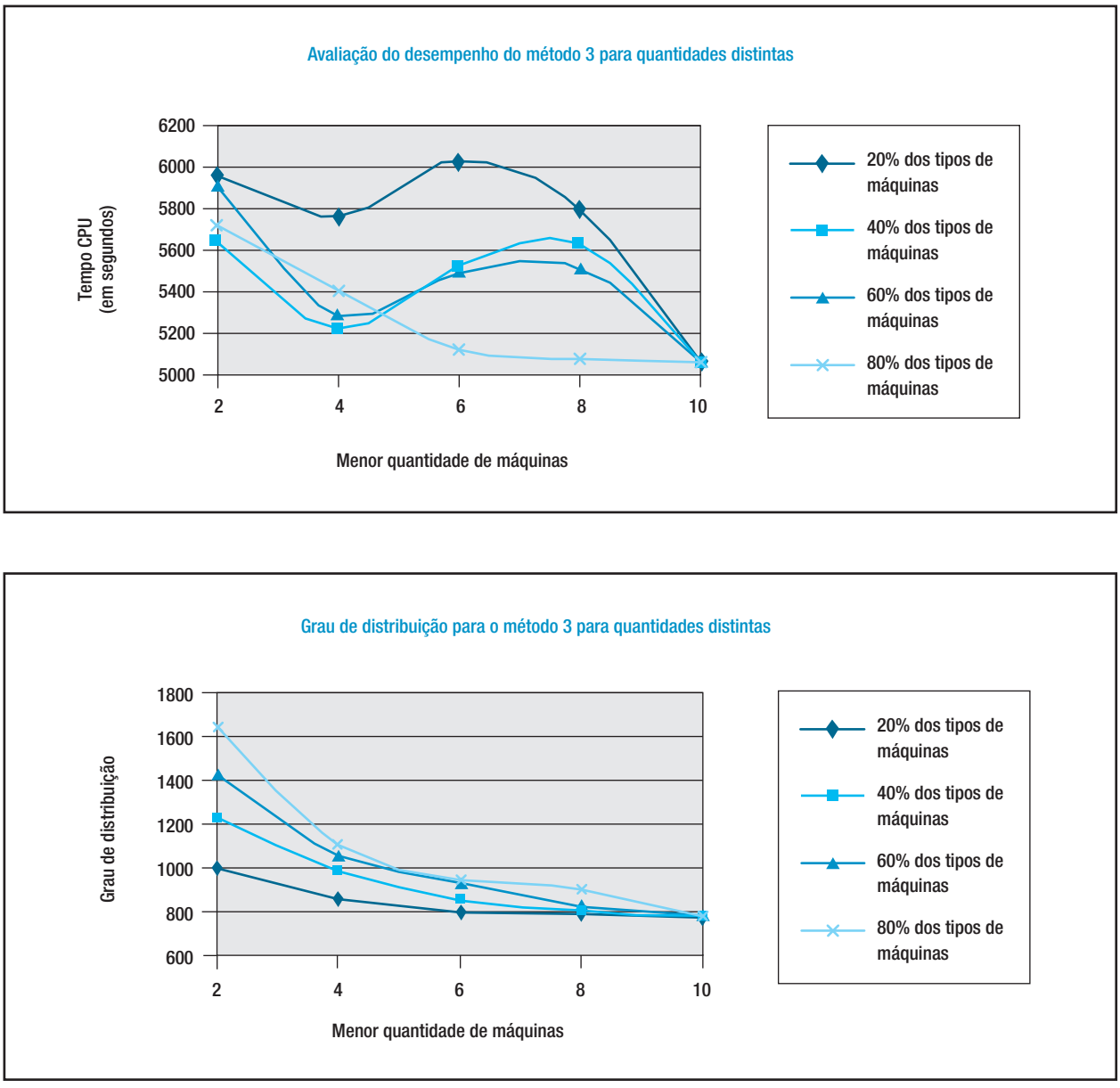


Figura 8 – Desempenho do método 3

6. ANÁLISE DE DESEMPENHO PARA RÉPLICAS SIMILARES DE MÁQUINAS

Executando os códigos de programação para layouts, o pior caso do aleatoriamente distribuído é justamente para layout de até 1300 máquinas com tempo de cerca de 0,11s e para o método 1 foi de apenas 3~5s. Para o método 2, no pior caso, o tempo CPU consumido, em média, foi de 33 minutos. Já para o método 3, o pior caso é justamente para 130 tipos de máquinas com um total de 1300 máquinas e leva 1h 24min 26s.

As figuras 9, 10, 11 e 12 mostram o desempenho dos layouts gerados, variando o tamanho do layout e os tipos de máquinas. Nota-se que as curvas se assemelham bastante a uma reta. À medida que aumenta o número de tipos de máquinas, o desempenho para cada tamanho de layout tende a se tornar distintos. Esta distorção é mais visível nas curvas do método 2.

Para layouts menores (e abaixo de 90 tipos de máquinas), o desempenho entre os métodos 1 e 2 são praticamente similares. No entanto, se aumentar o tamanho do layout e o número de tipos de máquinas, o método 2 acaba se sobressaindo em relação ao método 1. Já se aumentar os tipos de máquinas para layouts pequenos (abaixo de 500 máquinas), o método 1 tende a se sobressair em relação ao método 2 (ver figuras 10 e 11, mais especificamente em torno de 130 tipos de máquinas). Verifica-se que o método 1 ainda apresenta pequena vantagem em relação ao aleatoriamente distribuído em qualquer tamanho e em qualquer número de tipo de máquinas. Já o método 3 se sobressaiu em grau de distribuição em relação aos demais métodos.

7. CONCLUSÃO

Constata-se, ao executar os códigos de programação, que o aumento dos tipos de máquinas afeta diretamente no grau de distribuição enquanto o tamanho do layout, no tempo de CPU. As pequenas variações de curva (figuras 9, 10, 11, e 12) se devem ao fato dos formatos dos layouts, mas esta variação é de apenas algumas unidades. Se existe grande variação (≈100), com certeza é oriunda da variação do tamanho do layout e dos tipos de máquinas.

Conclui-se também que, caso for gerar um layout aleatoriamente distribuído para fins de estudo, sugere-se usar métodos 1 ou 2 (conforme as vantagens de cada um discutidas anteriormente). O método 2 também é uma alternativa para o método ALVO só quando o número de réplicas for bem similar.

Para quantidades grandes de máquinas, vários tipos de máquinas e réplicas distintas, o ALVO e o método 3 são considerados os melhores. Aumentando as réplicas, torna-se impossível usar ALVO e, então os métodos 2 e 3 acabam sendo as opções. Foi mostrado que, tanto para réplicas distintas quanto para similares, o método 3 se sobressai em todos os aspectos de graus de distribuição.

O método 1 é, em geral, mais eficiente em relação ao aleatoriamente distribuído. O método 2 é bom para trabalhar com layouts de grande porte, vários tipos de máquinas e independe se forem de réplicas similares ou distintas. Porém, este de forma alguma substitui o método 1 porque existe tendência de se sobressair para réplicas similares (com vários tipos de máquinas e layouts de pequeno porte abaixo de 500).

Aparentemente o método 3 substitui todos os métodos apresentados neste artigo (para todos os tamanhos de layout acima de 500 máquinas), independente se são de mesma réplica, ou não. Adotar apenas o método 3 gera uma série de controvérsias. Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2002) haviam previsto para layouts de pequeno porte (ou layouts grandes) que, quanto mais espalhadas as máquinas, menor é a chance de compartilhamento de recursos e, portanto, mais alto é o custo operacional. Sendo assim, para estes casos os métodos 1 e 2 são as únicas opções.

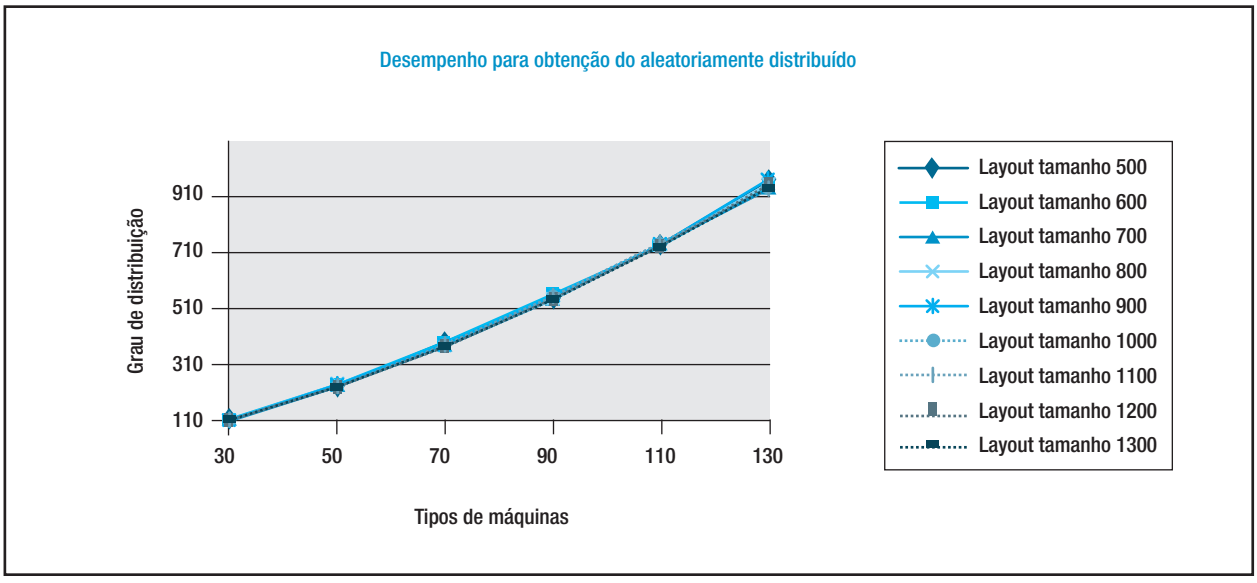


Figura 9 – Desempenho do aleatoriamente distribuído

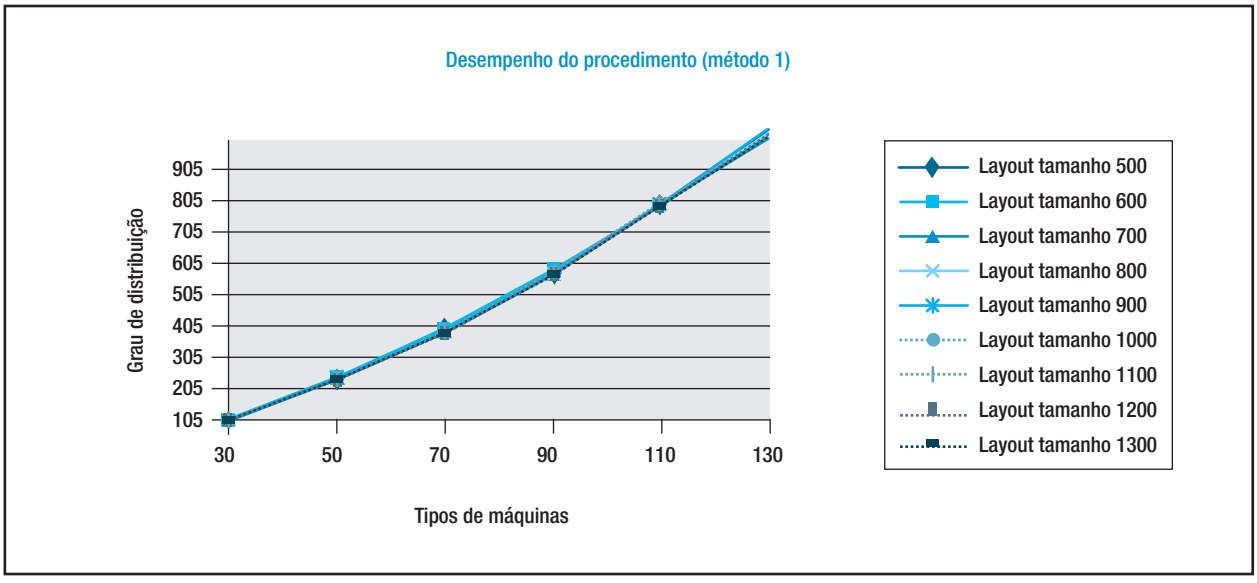


Figura 10 – Desempenho do método 1

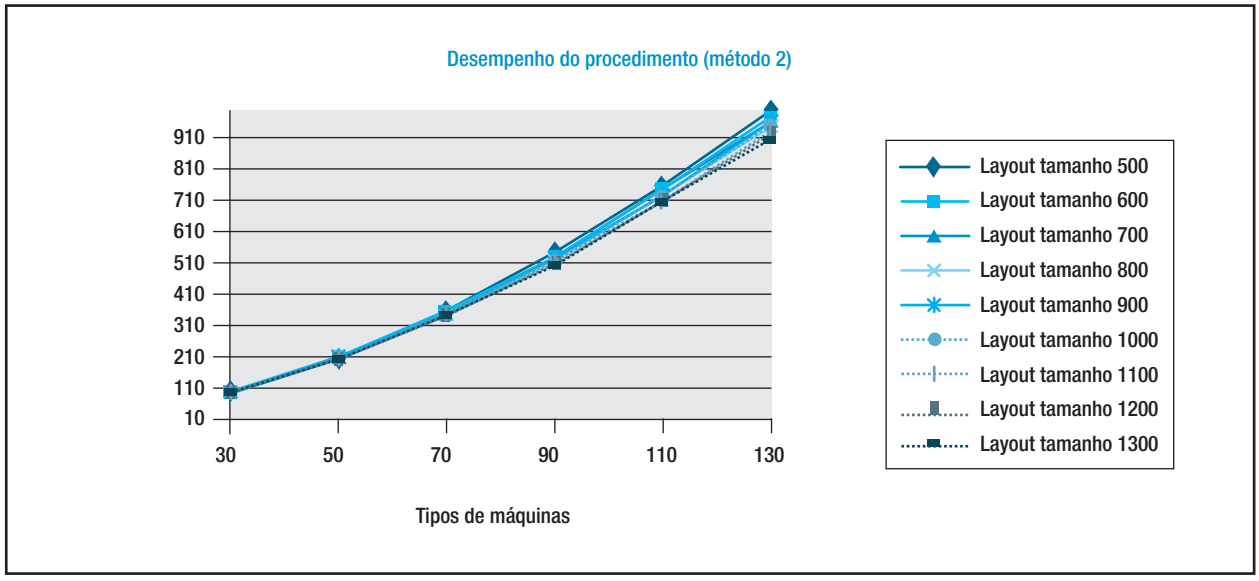


Figura 11 – Desempenho do método 2

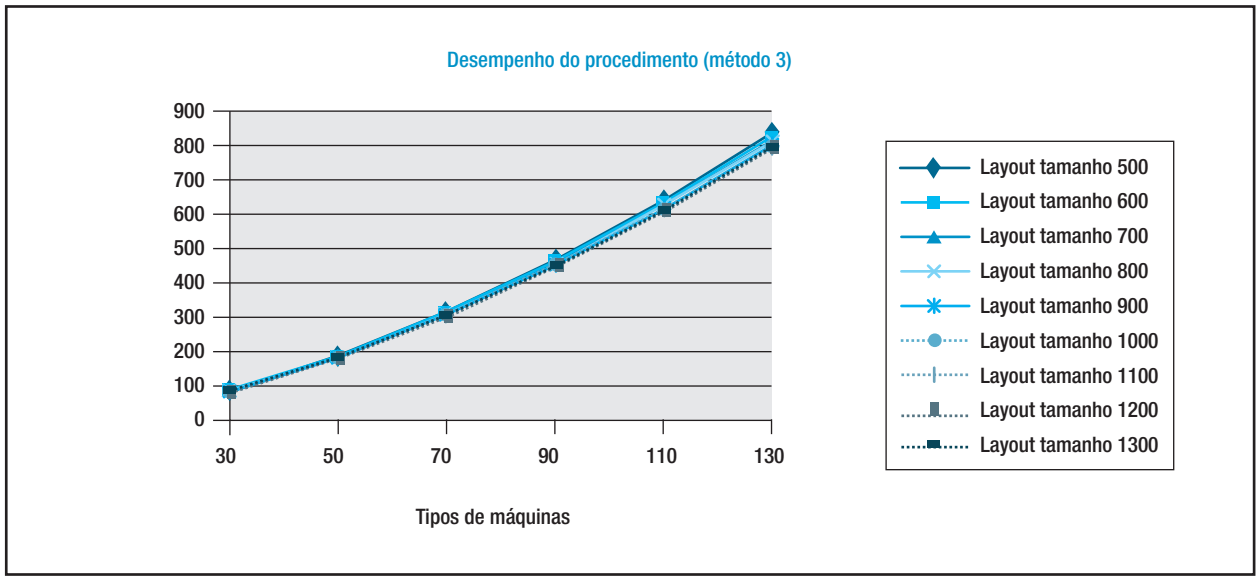


Figura 12 – Desempenho do método 3

8. AGRADECIMENTOS

Os autores agradecem aos avaliadores anônimos que contribuíram de forma significativa para a melhora deste trabalho. Agradecem também à Capes e ao CNPq pelas bolsas concedidas e à Editoria Gepros pela atenção dada.

9. REFERÊNCIAS BIBLIOGRÁFICAS

BENJAAFAR, S.; SHEIKHZADEH, M. Design of flexible plant layouts. **IEE-Transactions**, n. 32, p. 309-322, 2000.

CERVO, A. L.; BERVIAN, P. A. Metodologia científica. 3ed. **McGraw-Hill do Brasil**, São Paulo, 1983.

CHIZZOTTI, A. **Pesquisa em ciências humanas e sociais**. Cortez Editora (Biblioteca da educação. Série 1. Escola, v. 16), São Paulo, 1991.

GIL, A. C. **Como elaborar projetos de pesquisa**. Atlas S. A., São Paulo, 1987.

GORGULHO JÚNIOR, J. H. C. Análise do desempenho dos arranjos físicos distribuídos em ambiente de programação de tarefas com flexibilidade de seqüência de fabricação. **Tese apresentada à Escola de Engenharia de São Carlos**, USP, São Carlos, 2006.

GORGULHO JÚNIOR, J. H. C. ; GONÇALVES FILHO, E. V. Análise do desempenho dos arranjos físicos distribuídos operando sob roteamento de peças com flexibilidade de sequenciamento. **Revista Gestão Industrial**, v. 3, n. 1, p. 1-12, 2007.

LAHMAR, M.; BENJAAFAR, S. Design of dynamic distributed layouts. **Proceedings of the 11th Annual Industrial Engineering Research Conference (IERC)**, March 19-21, Orlando, Florida, 2002.

MONTREUIL, B.; VENKATADRI, U. Scattered layout of intelligent job shops operating in a volatile environment. Proceedings of the International Conference on Computer Integrated Manufacturing, **ICCIM**, Singapure,1991.

MONTREUIL, 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, Groupe de Recherche en Gestion de La Logistique, Faculte dês Sciences de L´Administration, Université Laval, Québec, Canadá, 1993.