Um Algoritmo Evolutivo Eficiente para o Problema de Formação de Células de Manufatura

Áthila Rocha Trindade, Luiz Satoru Ochi (*)

Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156, Bloco E, Niterói – Rio de Janeiro – Brasil - 24210-240
E-mail: **athilarocha@bol.com.br**, (*) **satoru@ic.uff.br**

Abstract

This paper presents an efficient evolutionary heuristic to solve the problem of formation of machine cells and parts families in the design of production systems. The proposed metaheuristic includes in a standard genetic algorithm the following procedures: a randomized heuristic to generate an initial population of individuals; an efficient local search algorithm; a randomized crossover operator and a procedure to calibrate the number of cells for each problem. The effectiveness of the proposed algorithm is demonstrated on several instances from the literature where our algorithm is compared with an hybrid genetic algorithm that has presented the best results for this problem so far.

Keywords: *combinatorial optimization, metaheuristics, evolutionary algorithm, clustering Problems.*

1 – Introdução

Uma das aplicações mais populares de problemas de clusterização, ocorre em sistemas de manufatura onde surge o *Problema de Formação de Células de Manufatura (PFCM).* Dentro da abordagem de tecnologia de grupos, uma das técnicas mais usadas é o desenvolvimento de um sistema de manufatura clusterizado em que *partes* ou peças similares são agrupados em *famílias* e onde *máquinas* são agrupadas em *células*. Um agrupamento ideal é constituído de células independentes, que possuem as seguintes características: i) todas as partes pertencentes à célula são produzidas totalmente dentro da célula e; ao mesmo tempo, ii) nenhuma máquina da célula deve participar do processo de produção de qualquer parte que não pertença a esta célula. Porém, na prática é raro obter um sistema de manufatura clusterizado totalmente independente, mas o desejável é que se minimize o grau de dependência entre elas. Como o PFCM é classificado como NP-Completo, vários algoritmos heurísticos ou metaheurísticos tem sido propostos para a sua solução aproximada [Joines et al. (1996), Wang (1998), Aljaber et al. (1997), Gonçalves et al. (2002), Mak et al. (2000)]. Este trabalho apresenta um novo algoritmo evolutivo (AE) para resolver o PFCM incorporando num algoritmo genético básico (AGB) os seguintes procedimentos: uma heurística randomizada para fornecer uma população inicial semi otimizada; um módulo de busca local para refinar as soluções encontradas pelo AG; um procedimento de reprodução randomizada que substitui o crossover tradicional; e uma técnica para calibrar o número de clusters para uma dada instância. O desempenho do AE é comparado com metaheurísticas da literatura e os resultados computacionais mostram uma superioridade do nosso método em relação à qualidade das soluções geradas e utilizando tempos computacionais similares aos da literatura para um conjunto de instâncias disponíveis em bibliotecas públicas.

2 – Descrição do PFCM

O problema de formação de células de manufatura (PFCM) na sua versão original é representado como uma matriz *"parte x máquina"* onde as linhas representam as partes e as colunas às máquinas, ou vice-versa. Considerando aqui uma matriz A = (*parte x máquina)*, cada célula a_{ij} da matriz é igual a 1 se a *parte i* utiliza a máquina *j*, e 0 caso contrário; a formação dos grupos (clusters) "célula / família" é feita através da permutação das linhas e colunas desta matriz. Por exemplo na tabela 2.1, considere um fluxo de produção

composto por 6 *partes*, 4 máquinas e 14 atividades (posições da matriz com valor 1). As tabelas 2.1 e 2.2 representam respectivamente a matriz de entrada do problema e uma possível matriz solução com formação de duas células/*famílias*.

	M1	M2	M3	M4
P1		1	1	
P2	1	1		1
P3		1	1	
P4	1			1
P5	1	1		
P6		1	1	1

Tabela 2.1: Matriz de entrada do um PFCM

	M2	M3	M1	M4
P1	1	1		
P3	1	1		
P6	1	1		1
P2	1		1	1
P4			1	1
P5	1		1	

Tabela 2.2: exemplo de uma clusterização da matriz da tabela 2.1

Na matriz da tabela 2.2 existem duas *famílias*: [P1,P3,P6] e [P2,P4,P5]. Associadas a estas *famílias* temos 2 células: [M2,M3] e [M1,M4].

Este problema (PFCM), já é razoavelmente explorado pela literatura. O objetivo desta seção é citar alguns trabalhos existentes que acreditamos serem mais significativos para o enfoque deste trabalho. Um AG para o PFCM é apresentado por Joines et al. (1996). O algoritmo utiliza 4 tipos de mutação e 3 tipos de crossover, obtendo bons resultados segundos seus autores. Em Wang(1998), é apresentada uma heurística que, numa primeira etapa, determina a criação de *k famílias* (grupos de partes) ou *células* (grupos de máquinas), onde *k* é um parâmetro de entrada; através da determinação dos *k* pares de partes ou máquinas mais dissimilares, de acordo com um índice de dissimilaridade calculado entre elas. A partir daí, é feita uma atribuição das partes e máquinas restantes às células e famílias existentes com base num modelo de atribuição linear. Porém, as soluções se mostram fortemente dependentes do parâmetro inicial k e do número máximo de máquinas por células e de partes por família. Uma metaheurística Busca Tabu foi proposta Aljaber et al. (1997), sendo que ela se constitui basicamente de três fases: a primeira constitui-se na determinação do melhor sequenciamento de máquinas e partes, de maneira a se formar dois caminhos mínimos; um caminho para partes e outro para máquinas, onde as distâncias entre máquinas e entre partes estão diretamente relacionadas com o grau de similaridade entre elas. Estes dois caminhos são determinados através da aplicação do algoritmo de busca tabu propriamente dito. Numa segunda fase, é montada a matriz de clusterização referente à melhor solução encontrada na primeira fase. A terceira fase se constitui da determinação do número de clusters através da identificação das fronteiras de cada cluster. Porém necessita da informação de um limite superior de clusters a serem formados que deve ser informado inicialmente pelo usuário. Em Mak et al. (2000) é proposto um AG adaptativo onde as taxas de utilização dos operadores de mutação e crossover mudam de acordo com a eficiência dos mesmos ao longo da execução do algoritmo. Gonçalves e Resende (2002) apresentaram um algoritmo genético híbrido (HGA) que incorpora ao AG básico um procedimento de busca local, tendo este algoritmo igualado ou melhorado os resultados existentes na literatura para 35 instâncias do PFCM. O HGA constrói uma população inicial de indivíduos de forma aleatória; o critério de parada é pelo número de iterações; a função de aptidão é a mesma que a usada em Joines et al. (1996) e dada pela fórmula:

$$= \frac{e - e_0}{e + e_v}$$

onde: e = o número de operações (1's) na matriz de entrada; e_V = o número de lacunas nos blocos diagonais (células/famílias), que representam a esparcidade de cada cluster; e_0 = número de elementos excepcionais (elementos preenchidos com 1 que não pertencem às células/famílias).

Observa-se que a função de avaliação procura minimizar o número de elementos excepcionais (ou ruídos) minimizando o movimento entre clusters; e lacunas nos clusters (células/famílias), maximizando o uso das máquinas nos clusters. Quanto mais próximo de 1 é o seu valor, melhor será a solução. A busca local proposta é aplicada a todos os indivíduos e é iniciada sob uma solução parcial, ou seja, sob uma solução que

ainda não possui um conjunto de famílias de partes. Basicamente ela se constitui de dois passos, que são os seguintes: Inicialmente a associação de partes a células é feita com base no cálculo de um coeficiente, de maneira que cada parte é associada à célula para a qual possui o maior coeficiente. Caso esta nova associação de partes a células aumente a aptidão da solução, então se executa o passo seguinte sob a nova solução, senão a busca local é finalizada e as partes são associadas às suas células originais. Se for a primeira execução do passo 1, isto significa que foi formado o primeiro conjunto de famílias da solução e então o passo 2 deve ser obrigatoriamente executado. Num passo 2, após feita a associação das partes, será formado um conjunto de famílias que estarão associadas às células de forma similar ao passo anterior. Caso esta nova associação de máquinas aos clusters aumente a aptidão da solução então se executa novamente o passo 1 sob o novo conjunto de células formado, senão a busca local é finalizada.

3 – O algoritmo Evolutivo proposto (AE)

Esta sessão descreve resumidamente as etapas do AE aqui proposto.

Representação do indivíduo: Como proposto em Joines et al. (1996), cada indivíduo é composto por (m+n) genes, onde m representa o número de máquinas e n o número de partes do problema. Desta forma, um indivíduo é representado como um string da forma $C = (x_1,...x_m \mid y_1,...y_n)$. Cada gene x_i ou y_i assume valores inteiros, sendo feita à alocação de (máquinas a células) e (partes a famílias) simultaneamente. Considere o

indivíduo, para um problema de 12 máquinas, 15 partes e 4 células/famílias:
C_1 = (2 4 2 3 2 3 4 1 3 4 3 1 | 2 4 2 4 1 1 2 2 1 1 3 3 4 4 3). Cada gene x_i indica para qual célula cada máquina é alocada. Por exemplo, em C_1 o gene 1 (x_1=2) significa que a máquina M_1 é alocada para a célula 2 e cada gene y_i indica para qual *família* cada *parte* é alocada. O gene m+1 = y_1 = 2 significa que à *parte* P_1 é alocada para a *família* 2.

Função de aptidão: A função de avaliação usada neste trabalho é a mesma usada em Joines et al. (1996) e em Gonçalves et al. (2002) e já descrita anteriormente.

Geração da População inicial: A população inicial constitui-se parte de indivíduos gerados aleatoriamente e parte de indivíduos gerados através de um procedimento heurístico randomizado. Este procedimento é baseado na heurística apresentada por Wang (1998) para resolver o PFCM. A heuristica é adaptada incorporando escolhas aleatórias numa lista restrita composta pelos k melhores candidatos para cada gene de um indivíduo. A heurística de construção de indivíduos é dividida em três fases descritas a seguir:

P1) Na 1ª fase são escolhidos os 3 pares de máquinas menos similares (que possuem o menor número de partes atendidas em comum).

P2) Na 2ª fase é escolhido aleatoriamente um par dentre os 3 pares de máquinas determinados em P1) como sendo os dois clusters iniciais da solução. A partir daí, cada novo cluster inicial é determinado como sendo a máquina menos similar aos clusters já formados. Este processo se repete até que sejam gerados os k clusters, onde k é um parâmetro de entrada definido para cada inidividuo. Após o término desta fase, os k clusters iniciais estarão determinados.

P3) Na 3ª fase, para cada máquina são determinados o 1º e o 2º cluster com maior índice de similaridade (nº de partes atendidas em comum entre a máquina e pelo menos uma máquina do cluster). Então a máquina é atribuída aleatoriamente a um dos doi clusterss, desde que o número máximo de máquinas por cluster não seja excedido. Caso isso aconteça, a máquina é atribuída a um cluster qualquer que não tenha excedido o número máximo de máquinas. O número máximo de máquinas por cluster é calculado como

sendo igual a $\lceil (n^o \text{ de máquinas/k}) + 1 \rceil$ e o número máximo de partes por cluster igual a $\lceil (n^o \text{ de partes/k}) + 1 \rceil$.

O processo de atribuição das partes aos clusters é idêntico ao processo de atribuição das máquinas aos clusters. A única diferença é que a similaridade entre uma parte e um cluster é dada pelo número de máquinas do cluster em questão que servem à referida parte. No nosso AE, são gerados *5m* indivíduos na primeira população. Destes, permanecem os *m* mais aptos, onde *m* representa o tamanho da população.

Estratégia de seleção dos indivíduos reprodutores: Esta fase trata da escolha dos indivíduos que irão servir como pais para o processo de reprodução e também para decidir entre os pais e filhos gerados, quais irão para a próxima geração. Escolher simplesmente os melhores indivíduos pode levar o algoritmo genético a uma parada prematura em ótimos locais ainda distantes de ótimos globais. Em razão disto, neste trabalho foi dada uma probabilidade a cada indivíduo de ser selecionado, de acordo com o seu valor de aptidão; sendo a probabilidade de escolha maior para indivíduos de melhor aptidão e menor para indivíduos de pior aptidão.

Mecanismo de reprodução: Em nosso trabalho o operador crossover foi substituído por um mecanismo de reprodução da seguinte forma: A partir de p ($p \geq 2$) pais geramos p filhos. Escolhidos os *p* pais para a reprodução, cada gene do filho referente a uma máquina receberá aleatoriamente o valor correspondente a uma das *k'* células nas quais a presença desta máquina é mais freqüente, com base nas *p* soluções pai, onde *k'* é um parâmetro de entrada. Após a associação das máquinas, para cada parte é feita uma lista das *k''* máquinas que mais aparecem no mesmo cluster da referida parte, com base nas *p* soluções pai, sendo *k''* também um parâmetro de entrada. Então cada parte é associada à família correspondente à célula que a máquina escolhida está associada.

Procedimento de busca local: A busca local num AE tem contribuido significativamente no desempenho destas heurísticas [Vianna et al. (2001)]. Neste trabalho foi incorporado ao algoritmo evolutivo um procedimento de busca local que numa primeira fase procura associar as partes aos clusters de acordo com a associação das máquinas, gerando um novo conjunto de famílias. Em seguida, caso a aptidão da solução seja melhorada, as máquinas são reassociadas de acordo com as famílias formadas anteriormente, obtendo um novo conjunto de células. Estas duas etapas são executadas alternadamente enquanto houver melhoria na aptidão da solução resultante. Mais detalhadamente, os passos são os seguintes:

P1) Para cada parte *i*, é calculado o coeficiente abaixo, com relação a todo cluster *k*.

$P_{ik} = (N_{1intrak} - N_{ruidosk} - N_{0intrak})$ se $N_{1intrak} \neq 0$; senão (se $N_{1intrak} = 0$), faço $P_k = -\infty$; onde:

$N_{1intrak}$ = número de elementos do clusters *k* iguais a 1 se a parte for associada a ele
$N_{ruidosk}$ = número de elementos iguais a 1 fora do cluster *k* se a parte for associada a ele.
$N_{0intrak}$ = número de elementos iguais a 0 dentro do cluster *k* se a parte for associada a ele.

A penalidade no valor de P_{ik} quando $N_{1intrak} = 0$ faz com que uma parte *não seja* associada a um cluster o qual não possui máquinas que a servem. A seguir, cada parte é associada ao cluster *k* para o qual possui o maior coeficiente P_{ik}. Caso haja empate, a parte será associada para o cluster *k* cuja razão entre $N_{1intrak}$ e o número de máquinas pertencentes ao cluster *k* for a maior. Caso não haja melhoria na aptidão da solução, as partes são alocadas aos clusters originais e a busca local é finalizada. Caso haja melhoria, executa-se o passo P2.

P2) Usa-se procedimento similar ao passo P1, mas agora se faz a mudança das máquinas de cluster calculando-se M_{ik} de forma similar a P_{ik} para cada máquina *i* em relação a cada cluster *k*, de acordo com a

alocação das partes feitas pelo passo P1. Caso haja empate entre coeficientes, a máquina então será associada para o cluster *k* cuja razão entre $N_{1intrak}$ e o número de partes pertencentes ao cluster *k* for a maior. A seguir, cada máquina é associada ao cluster para o qual possui o maior coeficiente. Caso não haja melhoria na aptidão da solução, as máquinas são alocadas aos clusters originais e a busca local é terminada. Caso haja melhoria, executa-se o passo P1 novamente. Estes passos P1) e P2) buscam o mesmo objetivo requerido pela busca local proposta em Gonçalves et al. (2002), contudo a fórmula usada para atingir o objetivo em ambos são distintas.

Calibração do número de clusters: Um parâmetro do PFCM que influi diretamente no resultado da solução final é o número de clusters (células) a serem formados. Se o intervalo analisado para o número de clusters for demasiadamente grande, aumenta-se o espaço de busca e conseqüentemente o tempo de execução do algoritmo. Desta forma, é importante que tenhamos uma boa estimativa para os limites inferiores e superiores deste intervalo. Como são consideradas na literatura como soluções inválidas aquelas que contém uma máquina apenas ou uma parte apenas e dado que o número de máquinas é sempre menor que o número de partes, um limite superior do número de clusters proposto neste trabalho foi $\lceil (n^o \text{ de máquinas}/2) \rceil$. Este valor não evita que sejam formadas soluções com clusters unitários, mas restringe o espaço de busca, tendo em vista que soluções com um número de clusters maior que este limite dado possuem obrigatoriamente clusters unitários. O valor do limite mínimo de clusters é igual a 2. Antes da geração de cada indivíduo é sorteado um número dentro desta faixa de limites para ser o número de clusters para a solução associada. O algoritmo então só considera como soluções válidas aquelas sem clusters unitários, sem clusters com linhas ou colunas vazias ou clusters inteiramente vazios.

Parâmetros do AE: No AE são usados os seguintes parâmetros: A população de cada geração é sempre igual a 2,5 vezes o número de máquinas da instância em questão. A busca local é aplicada sobre 30% dos indivíduos, sendo estes escolhidos de acordo com o mesmo critério de escolha de indivíduos reprodutores. Para o mecanismo de reprodução, os valores de *p*, *k'* e *k''* foram respectivamente 15% do tamanho da população, 10% do número máximo de clusters e 10% do número de máquinas. Para *p*, caso este valor seja menor que 5, *p* assume o valor 5. Para *k' e k''*, caso os valores sejam menores que 3, *k' e k''* assumem o valor 3. Para instâncias em que o valor máximo de clusters é igual a 2, o valor de *k'* é igual a 2. O critério de parada usado foi um limite do número de gerações, igual a 150 gerações.

4 - Resultados computacionais

Para avaliar o algoritmo proposto (AE), tomamos como parâmetro de comparação o algoritmo HGA proposto por Gonçalves et al. (2002), que segundo seus autores detinha os melhores resultados aproximados da literatura até então. Em relação ao HGA, utilizamos as instâncias disponibilizadas e os resultados por ele obtidos. Os testes computacionais com o AE foram realizados em um computador AMD 1.410 Mhz e 250 Mb de memória. Foram utilizadas 35 instâncias da literatura. O AE proposto foi executado 10 vezes para cada instância. Além disso, os resultados foram comparados com os resultados obtidos em: AG-JCK, proposto por Joines et al. (1996), e implementado por nós; Outros algoritmos e resultados aqui relacionados estão disponíveis em Gonçalves et al. (2002) como o algoritmo Zodiac, proposto por Srinivasan e Narendan (1991); Grafics, proposto Srinivasan and Narendan (1991); algoritmo de clusterização proposto por Srinivasan em 1991 (Ca); algoritmo genético proposto por Cheng et. al em 1998 (GA). A tabela 4.1 ilustra os resultados para cada instância (colunas 3 a 9), a coluna 10 (Imp) mostra a diferença em percentual da solução do AE com a melhor solução da literatura. Para a execução do AG-JCK, o número de gerações variou de acordo com o tamanho das instâncias, como sugerido pelos autores. O número de gerações para as instâncias foi: instâncias 1 a 9, 150; 10 a 15, 500; 16 a 33, 5.000 e 34 e 35, 20.000 gerações.

Inst.	Dim	AG-JCK	Zodiac	Grafics	Ca	GA	HGA	AE	Imp
1	5 x 7	73.68	73.68	73.68	-	-	73.68	73.68	0%

2	5 x 7	62.50	56.52	60.87	-	**69.56**	62.50	62.50*	-10.14%
3	5 x 18	79.59	77.36	-	-	77.36	79.59	79.59	0%
4	6 x 8	76.92	76.92	-	-	76.92	76.92	76.92	0%
5	7 x 11	53.13	39.13	53.12	-	46.88	53.13	53.13	0%
6	7 x 11	70.37	70.37	-	-	70.37	70.37	70.37	0%
7	8 x 12	68.29	68.29	68.29	-	-	68.29	68.29	0%
8	8 x 20	57.26	58.33	58.13	58.72	58.33	58.72	58.72	0%
9	8 x 20	85.25	85.25	85.25	85.25	85.25	85.25	85.25	0%
10	10 x 10	70.59	70.59	70.59	70.59	70.59	70.59	70.59	0%
11	10 x 15	92.00	92.00	92.00	-	92.00	92.00	92.00	0%
12	14 x 23	66.67	64.36	64.36	64.36	-	69.86	69.86	0%
13	14 x 24	69.33	65.55	65.55	-	67.44	69.33	69.33	0%
14	16 x 24	44.92	32.09	45.52	48.7	-	51.96	51.96	0%
15	16 x 30	56.82	67.83	67.83	67.83	-	67.83	67.83	0%
16	16 x 43	54.86	53.76	54.39	54.44	53.89	54.86	54.86	0%
17	18 x 24	51.26	41.84	48.91	44.20	-	54.46	54.46	0%
18	20 x 20	40.91	21.63	38.26	-	37.12	42.96	42.96	0%
19	20 x 23	47.77	38.66	49.36	43.01	46.62	49.65	49.65	0%
20	20 x 35	76.22	75.14	75.14	75.14	75.28	76.14	76.14	-0,10%
21	20 x 35	56.52	51.13	-	-	55.14	58.06	58.06	0%
22	24 x 40	100.00	100.0	100.0	100.0	100.0	100.0	100.0	0%
23	24 x 40	85.11	85.11	85.11	85.11	85.11	85.11	85.11	0%
24	24 x 40	50.44	73.51	73.51	73.51	73.03	73.51	73.51	0%
25	24 x 40	51.50	20.42	43.27	51.81	49.37	51.85	51.97	+0,23%
26	24 x 40	42.39	18.23	44.51	44.72	44.67	46.5	47.06	+1,20%
27	24 x 40	42.94	17.61	41.67	44.17	42.50	44.85	44.87	+0,04%
28	27 x 27	54.23	52.14	41.37	51.00	-	54.27	54.27	0%
29	28 x 16	21.49	33.01	32.86	40.00	-	43.85	44.62	+1,76%
30	30 x 41	53.55	33.46	55.43	55.29	53.80	57.69	57.93	+0,42%
31	30 x 50	55.45	46.06	56.32	58.7	56.61	59.43	59.66	+0,39%
32	30 x 50	42.02	21.11	47.96	46.30	45.93	50.51	50.51	0%
33	30 x 90	15.78	32.73	39.41	40.05	-	41.71	43.66	+4,68%
34	37 x 53	56.83	52.21	52.21	-	-	56.14	57.54	+2,49%
35	40x100	84.03	83.66	83.92	83.92	84.03	84.03	84.03	0%

Tabela 4.1: resultados obtidos por algoritmos da literatura e o algoritmo proposto.

Pelos resultados da tabela 4.1, onde os melhores resultados estão em negrito, podemos notar que o AE conseguiu melhores resultados isoladamente para 8 das 35 instâncias, obteve a mesma melhor solução da literatura em outros 25 e para as instâncias (2 e 20) obteve uma solução com pior aptidão que a da literatura. Porém vale ressaltar que a solução encontrada para a instância 2 pelo algoritmo GA contém clusters unitários, o que não é permitido no nosso AE e no HGA. Para verificar o desempenho do AE sem esta restrição, executamos o AE e obtivemos na versão relaxada a mesma solução do GA. A análise dos tempos de cpu do AE será descrita resumidamente sem tabelas devido a limitação do tamanho deste paper. Os tempos de cpu do nosso AE foi em média de duas a 4 vezes menor que o tempo exigido pelo HGA. Na maior instância (35), o tempo do AE foi de 62,32 segundos e no HGA de 125, 33 segundos. Em relação ao AG-JCK, os tempos para instância menores o AG-JCK foi mais rápido que o AE, mas a medida que as dimensões cresciam o tempo do AG-JCK foi se tornando bem maior que o do AE. Por exemplo, na instância 34, o tempo do AG-JCK foi de 1765,91 segundos e do AE de 19, 49 segundos.

5 - Conclusões

Pelos resultados computacionais, podemos concluir que o algoritmo proposto obteve em relação a qualidade das soluções, resultados médios superiores do que os algoritmos propostos na literatura e em particular na média superou os resultados obtidos pelo HGA proposto em Gonçalves et al. (2002) que até então detinha os melhores resultados aproximados da literatura. Também em relação aos tempos computacionais, o nosso AE exige tempos similares ou menores que os de HGA. Desta forma, embora de forma empírica, mostramos o

potencial do algoritmo evolutivo aqui proposto, que nos testes realizados se mostrou superior tanto nas soluções por ele geradas como nos tempos computacionais exigidos.

8. Referências

[1] J. A. Joines, C. T. Culbreth and R. E. King (1996), Manufacturing Cell Design: An Integer Programming Model Employing Genetic Algorithms. Tec. Report, Department of Industrial Engineering, North Caroline State University, NC 27695-7906.

[2] J. Wang (1998), A Linear Assignment Algorithm for Formation of Machine Cells and Parts Families in Cellular Manufacturing, in Computers Ind. Engineering, vol 35(1-2), 81-84.

[3] N. Aljaber, W. Baek and C.-L. Chen (1997), A Tabu Search Approach to the Cell Formation Problem, in Computers and Industrial Engineering, Vol. 32(1), 169-185.

[4] J. F. Gonçalves and M.G.C. Resende (2002), A hybrid genetic algorithm for manufacturing cell formation, Tec. Report, Fac. de Engenharia do Porto, Portugal. (submitted to Comp. & Industrial Eng).

[5] K. L. Mak, Y. S. Wong and X. X. Wang (2000), An Adaptative Genetic Algorithm for Manufacturing Cell Formation, in Journal of Advanced Manufacturing Technology, Nº 16, 491-497.

[6] D. S. Vianna, L. Satoru Ochi and L. M. Drummond (2001), An asynchronous parallel metaheuristic for the period vehicle routing problem, in Future Generations on Computer Systems, vol. 17(4), 379-386, Elsevier.