versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

UM ALGORITMO EVOLUTIVO HÍBRIDO PARA A FORMAÇÃO DE CÉLULAS DE MANUFATURA EM SISTEMAS DE PRODUÇÃO

Áthila Rocha Trindade
Luiz Satoru Ochi *
Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói – RJ
satoru@ic.uff.br

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 01/2005; aceito em 01/2006 após 1 revisão
Received January 2005; accepted January 2006 after one revision

Resumo

O Problema de Formação de Células de Manufatura (PFCM) é uma questão central para um projeto de geração de células de produção em sistemas de manufatura. Este problema é basicamente descrito por um conjunto de partes de produtos e máquinas. O objetivo é o de construir clusters ou células de manufatura associando produtos com agrupamentos de máquinas. Este trabalho apresenta um novo algoritmo evolutivo híbrido para a solução do PFCM. São mostrados resultados computacionais do algoritmo proposto para um conjunto de instâncias disponíveis na literatura. Das 36 instâncias analisadas, o algoritmo proposto encontrou uma solução superior aos existentes em 8 casos e em 26 instâncias encontrou a mesma melhor solução da literatura.

Palavras-chave: heurísticas; algoritmos evolutivos; algoritmos genéticos.

Abstract

The Manufacturing Cell Formation Problem (MCFP) is a crucial component of a cell production design in a manufacturing system. This problem is composed by a set of parts of products to be manufactured and machines. The objective is to construct manufacturing clusters by associating products with cell machines. This paper presents a new hybrid evolutionary algorithm to solve the MCFP. Computational results with the proposed algorithm on a set of instances available in the literature are also presented. For 8 out of 36 instances considered, the propose method overcame the previous results from the literature and for 26 instances, the same best solutions were found.

Keywords: heuristics; evolutionary algorithms; genetic algorithms.

1. Introdução

O alto grau de complexidade computacional encontrado em muitos problemas práticos de otimização tem levado pesquisadores a propor alternativas para resolvê-los aproximadamente. Neste sentido tem sido concentrados esforços na busca de métodos que possam alcançar boas soluções (não necessariamente uma ótima) num tempo computacional razoável, utilizando mecanismos que permitam escapar de ótimos locais muitas vezes ainda distantes de um ótimo global.

Um caminho promissor para alcançar esta meta, é conjugar conceitos das áreas de Otimização e Inteligência Artificial, viabilizando a construção das chamadas "*melhores estratégias*" ou dos "*métodos inteligentemente flexíveis*" comumente conhecidos como meta-heurísticas. A literatura apresenta atualmente um número bem razoável de metaheurísticas e dentre os mais populares podemos incluir: as Redes Neurais (RNs) [Mast95]; os Algoritmos Evolutivos (AEs) [Fog98] e o seu representante mais popular, os Algoritmos Genéticos (AGs) [Holl75]; Busca Tabu (BT) [Glo86], [Han86], *Greedy Randomized Adaptive Search Procedure* (GRASP) [FR95]; *Simulated Annealing* (SA) [KGV83] e *Variable Neithborhood Search* (VNS) [Mla95].

Este trabalho enfoca o uso de AEs e em particular dos AGs na solução de problemas de otimização de elevada complexidade computacional.

Os AGs são métodos que procuram realizar uma busca extensiva no universo de soluções, sendo seu funcionamento inspirado na evolução biológica. Cada solução (denominada indivíduo ou cromossomo) no AG é formada por componentes denominados genes. O AG é basicamente uma metaheurística iterativa onde a cada iteração uma população de indivíduos é gerada a partir da população anterior; utilizando mecanismos de reprodução conhecidos como *crossover* e mutação [Da91], [Mi94].

Este trabalho apresenta como proposta, o desenvolvimento de um algoritmo evolutivo híbrido (AE) aplicado ao Problema de Formação de Células de Manufatura (PFCM). O restante deste artigo está dividido da seguinte forma; a seção 2 apresenta o PFCM citando parte da literatura existente. Na seção 3 é descrito o algoritmo proposto e na seção 4, são apresentados os resultados computacionais acompanhados de sua análise. A seção 5 apresenta uma análise de convergência empírica do algoritmo proposto e em 6 são descritas as conclusões.

2. O Problema de Formação de Células de Manufatura

A gestão da produção é um fator muito importante para o sucesso de uma indústria. Através dela busca-se organizar o ambiente de produção de maneira a economizar custos e tempo de produção, sem perda de qualidade. Indústrias que possuem o foco na pequena variedade e alto volume de produção geralmente organizam o ambiente de produção em *linhas de produção*, sendo que cada *linha* é composta de vários tipos de máquinas (recursos de produção) dedicadas exclusivamente à produção de um único produto, tendo em vista o alto volume de produção normalmente requerido.

Para o caso de indústrias que possuem como foco a grande variedade e volume médio de produção, não há a necessidade de que as máquinas sejam dedicadas à produção de apenas um produto. Desta forma, é necessária a aplicação de uma nova abordagem de organização

destes sistemas de produção. Uma abordagem muito usada é a formação de grupos (*clusters*) de máquinas com funcionalidades idênticas (grupos de tornos, fresadeiras, etc.), formando-se departamentos especializados para uma função específica.

Assim, uma parte (peça) de um determinado produto que necessite de operações de manufatura de mais de um tipo de máquina, precisará percorrer todos os grupos que contenham os tipos de máquinas necessários para a sua completa manufatura; e isto tende a favorecer o aparecimento de um grande volume de material a ser manipulado ao longo dos diversos grupos de máquinas, o que contribui para o aumento no tempo gasto na produção e maior dificuldade de controle e manutenção do sistema de produção [XV03].

Nas últimas duas décadas um novo modelo de organização destes sistemas de produção denominado *manufatura celular* vem sendo usado.

Neste tipo de sistema de produção, máquinas de diferentes funcionalidades são agrupadas em uma célula, a qual é dedicada à produção de uma família de partes (partes que possuem alto nível de similaridade entre si no que dizem respeito às máquinas necessárias para sua manufatura).

O sistema de produção fica então dividido em vários clusters formados por *células* de máquinas e *famílias* de partes, clusters estes também chamados de *células de manufatura*.

Uma das primeiras técnicas de formação de um sistema de manufatura celular foi proposta por J. L. Burbidge em 1971; em seu trabalho intitulado *Análise do Fluxo de Produção* (veja [Wy97]), que se constitui como uma importante etapa da organização de um sistema de produção de volume médio e grande variedade de produtos.

A *Análise do Fluxo de Produção* é constituída das seguintes etapas: o levantamento dos recursos de produção (máquinas, ferramentas e profissionais) disponíveis e das rotas de produção de cada parte; a formação de células de manufatura; o escalonamento das operações das partes em cada célula de máquinas e o projeto do *leiaute* físico das células de máquinas no ambiente físico disponível da fábrica. A obtenção de células de manufatura é portanto uma das etapas da determinação de um sistema de produção baseado em manufatura celular.

Um exemplo da formação de células de manufatura pode ser visto nas Figuras 2.1 e 2.2. Considere um fluxo de produção composto por 6 partes, 4 máquinas e 14 atividades (posições da matriz com valor 1). As Figuras 2.1 e 2.2 representam respectivamente a matriz de entrada do problema e uma possível matriz solução com formação de duas células/famílias.

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

Figura 2.1 – Matriz de entrada do PFCM.

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

Figura 2.2 – Matriz de clusterização para a matriz de entrada do PFCM da Figura 2.1.

A Figura 2.2 apresenta a divisão do sistema de produção da Figura 2.1 em 2 clusters de (*células/máquinas; famílias/partes*). As células são ilustradas na Figura 2.2 e são descritas da seguinte forma: A célula 1 é formada pelas máquinas 2 e 3; e a 2 pelas máquinas 1 e 4. E as famílias são: a família 1 formada pelas partes 1, 3 e 6; a 2 pelas partes 2, 4 e 5.

Os clusters de células de máquinas e famílias de partes devem possuir um alto grau de independência entre si, caracterizado pela pequena presença de *elementos excepcionais* (elementos da matriz iguais a 1 que não pertencem a nenhum cluster) na matriz de clusters, os quais representam a presença de partes que necessitam de mais de uma célula de máquinas para serem completamente manufaturadas e, portanto gera a necessidade de movimento de material entre diferentes clusters de produção.

No exemplo da Figura 2.2, podemos identificar 3 elementos excepcionais ((M2, P2), (M2, P5), (M4, P6)). Também é desejável que se tenham poucas lacunas (elementos da matriz iguais a 0) dentro dos clusters, o que representa diferença na carga de trabalho de máquinas de um mesmo cluster de produção. Ou em outras palavras desejamos que cada cluster seja o mais denso possível. No exemplo da Figura 2.2 podemos identificar uma lacuna no segundo cluster dado pela componente (M1, P4).

A formação de clusters de: células de máquinas e famílias de partes resultam em diversas vantagens para a gestão da produção, dentro das quais destacamos:

1. Redução do transporte de material no ambiente de produção, já que as máquinas necessárias para manufatura de uma dada parte são agrupadas num menor número de clusters (de preferência num único cluster).
2. Redução do tempo de produção dos produtos e conseqüente aumento da capacidade de produção.
3. Simplificação do gerenciamento e controle do sistema de produção, agora dividido em vários subsistemas independentes.
4. Simplificação do escalonamento das atividades, que agora deve ser feito separadamente em cada cluster.
5. Aumento da segurança no trabalho, com a minimização de manipulação de material no ambiente de produção.

Um sistema de manufatura celular ideal é composto de clusters sem lacunas e sem qualquer interdependência (elementos excepcionais) e de acordo com Burbidge [Bu96] esta é a primeira meta a se alcançar para a implementação prática de um sistema de manufatura celular numa indústria.

No entanto, na prática é raro encontrar sistemas de produção que possam se tornar sistemas de manufatura com clusters totalmente independentes (excluindo obviamente o caso particular onde o sistema é composto por um único cluster). Em face disto, muitos pesquisadores tem procurado resolver a interdependência entre clusters através da duplicação de máquinas responsáveis pelo aparecimento de elementos excepcionais (no exemplo da Figura 2.2, a máquina 2 seria duplicada para o segundo cluster e a máquina 4 para o primeiro) ou a terceirização de partes (no exemplo da Figura 2.2 as partes P2, P5 e P6 seriam terceirizadas, pois estão associadas a elementos excepcionais).

Em um caso real, a decisão de duplicação de máquinas e terceirização de partes envolve questões monetárias que devem ser avaliadas e decididas pelo projetista do sistema de

manufatura. Além disso, existem outras questões inerentes ao sistema de produção a serem considerados como capacidade de produção das máquinas, tempo de duração de cada operação de manufatura, custo de cada operação de manufatura, etc.

A abordagem do Problema de Formação de Células de Manufatura dada neste trabalho não trata da duplicação de máquinas ou terceirização de partes, nem levam em consideração todos os aspectos relevantes ao projeto de um sistema de manufatura celular. O propósito deste trabalho é desenvolver um procedimento eficiente para clusterizar células de manufatura que possa ser adaptada para diferentes ambientes reais mais complexas.

2.1 O PFCM como um problema de clusterização

De acordo com Wang [Wa99], o número de alternativas para se agrupar **n** partes em **p** clusters mutuamente exclusivos e não vazios é determinado pelo número de Stirling de segundo tipo, dado pela equação 1.

$$\sum_{i=1}^{p}(-1)^{p-i} i^{n} \Big/ \left[i!(p-i)! \right] \qquad (1)$$

Como no PFCM uma solução é constituída do agrupamento de partes e máquinas, para cada maneira de se agrupar as partes pode se associar ***todas*** as maneiras de se agrupar as máquinas para se constituir uma solução.

Desta forma, o número de maneiras possíveis de se agrupar **n** partes e **m** máquinas em **p** clusters; ou seja, o número de soluções possíveis do PFCM para um número de clusters pré-fixado **p** é dado pela equação (2):

$$\left(\left(\sum_{i=1}^{p}(-1)^{p-i} i^{n} \Big/ \left[i!(p-i)! \right] \right) * \left(\sum_{i=1}^{p}(-1)^{p-i} i^{m} \Big/ \left[i!(p-i)! \right] \right) \right) \qquad (2)$$

A Tabela 2.1 ilustra a explosão no crescimento do número de soluções possíveis em função do número **n** de partes e **m** de máquinas, para um número de clusters fixado em **p** = 3 clusters.

Tabela 2.1 – Crescimento exponencial no número de soluções em função do crescimento do número de partes e máquinas para a formação de p = 3 clusters.

Nº de máquinas	**Nº de partes**	**Nº de soluções**
15	10	2,2159683 x $\mathbf{10^{10}}$
60	40	1,431603835 x $\mathbf{10^{46}}$
100	70	3,583372838 x $\mathbf{10^{79}}$

Observando os dados da Tabela 2.1, podemos concluir que o uso exclusivo de métodos exatos torna-se inacessível para instâncias de grande porte do PFCM; direcionando com isso a maior parte das pesquisas para o desenvolvimento de métodos heurísticos para a sua solução aproximada.

Na literatura o PFCM é classificado como um *Problema de Clusterização Automática* (PCA), ou seja, um problema de agrupar os elementos de uma base de dados onde o número de clusters (grupos) a serem formados é uma variável do problema.

No conceito de PCA existem diferentes aplicações além do PFCM incluindo, por exemplo: Problemas de Particionamento de Grafos [BGH99], [KK98], [MS98], [DO03]; Aplicações de clusterização em Redes de Telecomunicações [Mac03], [BOM05a], [BOM05b], [GLO03]; Problemas de Roteamento de Veículos [ODV98], [OVD01], [SOMD06]; Escalonamento de Tarefas [GY92], [HAR94], [GRS99] entre outras. Sobre PFCM, muitos métodos heurísticos tem sido propostos. Wemmerlov & Hyer [WH89], Selim, Askin & Vaklaria [SAV98] e Oduguwa, Tiwari & Roy [OTR04] apresentam revisões bibliográficas interessantes sobre o assunto. Em termos de técnicas mais usadas, destacam-se o uso de metaheuristicas e em particular: os Algoritmos Genéticos (AGs) ou Algoritmos Evolutivos (AEs) onde incluímos os trabalhos de: Joines, Culbreth & King [JCK96]; [CGLW98], [PP00], [ZW00], [FL00], [MWW00], DM01], [BS01], [OM01], [US02], [RG04]. A busca tabu também tem sido usada neste problema: [ABC97], [FSP99], [DLRG01], [SLB95], [Glo97]. Usando GRASP podemos citar os trabalhos de: [BOM05a], [BOM05b], [Ma03] e *Simulated Annealing* [HPX90], [CBP00], [XV03]. Na literatura encontram-se também várias heurísticas de clusterização propondo diferentes medidas de similaridade e de desempenho para a resolução do PFCM [CR86], [CR87], [JCT01], [Sa01], [VW95], [Wa99]. Entretanto, como experimentalmente concluiu Vakharia *et al*. [VW95], não existe uma combinação: *técnica de clusterização* - *medida de similaridade* - *medida de performance* que obtenha sempre os melhores resultados para qualquer instância do PFCM.

2.2 Determinação do número de clusters a serem formados

Um parâmetro do PFCM que influi diretamente no processo de resolução, é o cálculo do número ideal de clusters a serem formados. Quanto maior for o limite superior para este número, maior será o espaço de busca a ser explorado, o que pode aumentar o tempo de execução de um algoritmo na busca por uma boa solução. Desta forma, passa a ser fundamental a obtenção de uma boa estimativa do número ideal de clusters para uma dada instância.

Por outro lado, é reconhecidamente muito difícil para o usuário através de uma simples inspeção visual da matriz de entrada do PFCM, conhecer ou estimar este número ideal de clusters a serem formados.

Como a resolução do PFCM procura formar grupos nos quais as máquinas estejam agrupadas para atender a um conjunto de partes que possuem similaridade quanto às máquinas que as atendem, a formação de clusters com apenas 1 máquina ou apenas 1 parte é indesejável.

Estes clusters unitários seriam justificáveis em apenas dois casos: 1) Caso exista uma única máquina que atenda a um determinado subconjunto de partes; mas neste caso não faz sentido esta máquina e estas partes pertencerem à matriz de entrada do PCM, já que esta máquina não pode ser agrupada com nenhuma outra pelo fato desta não possuir partes atendidas em comum com nenhuma outra máquina. 2) Caso exista uma parte apenas que necessite de um determinado conjunto de máquinas e vice versa, ou seja que este subconjunto de máquinas seja utilizado apenas para esta parte; mas neste caso também não faz sentido esta parte e estas máquinas pertencerem à matriz de entrada do PFCM.

Desta forma, normalmente são consideradas na literatura como soluções inválidas aquelas que contém clusters com uma máquina apenas ou uma parte apenas [JCK96], [RG04].

Dado que o número de máquinas (m) é sempre menor que o número de partes, um limite superior para o número ideal de clusters usado neste trabalho é $\lceil m/2 \rceil$. Este valor não evita que sejam formadas soluções com clusters unitários, mas restringe o espaço de busca, pois soluções para o PFCM com um número de clusters maior que este limite possuem obrigatoriamente clusters unitários. O valor para o limite mínimo de clusters usado é igual a 2.

Assim num algoritmo heurístico onde é gerada mais de uma solução inicial, a nossa sugestão para definir em cada solução o número de clusters, é sortear um número dentro da faixa $[2, \lceil m/2 \rceil]$ para ser o número de clusters para a referida solução.

3. O Algoritmo Evolutivo Proposto para o PFCM

Apesar de sua eficiência comprovada para diversos problemas de otimização, o desempenho dos algoritmos genéticos (AGs) na sua forma original não tem se mostrado satisfatório para muitos problemas de otimização que já possuem algoritmos heurísticos eficientes para a sua solução aproximada.

Em razão disto, diversos pesquisadores têm proposto modificações nos AGs convencionais, incorporando a estes, técnicas de busca local [ERP02], [GMR04], [RG04], [ODV98], [OVD01], [SOMD06] ou incluindo nos AGs, conceitos de outras metaheurísticas tais como: *Simulated Annealing*, *Tabu Search (busca tabu)*, obtendo versões conhecidas como Algoritmos Meméticos [Mos89] ou simplesmente Algoritmos Evolutivos [LF01], [OVD01], [MB05].

Com isso procura-se um algoritmo evolutivo que possua as características básicas dos AGs mas incorporando mecanismos adicionais e com isso, tornando-o mais especializado para determinada aplicação. Neste trabalho propomos um algoritmo evolutivo (AE) que possui as seguintes particularidades em relação a um AG básico:

1. A população inicial não é gerada somente de maneira aleatória. Uma parte dos indivíduos é gerada através de um procedimento heurístico randomizado, de maneira que se consiga gerar em baixo tempo computacional indivíduos distintos e com nível de aptidão média melhor do que se estes fossem gerados aleatoriamente.
2. Um novo mecanismo de cruzamento é usado para gerar indivíduos a partir de indivíduos pais ao invés do operador de cruzamento (*crossover*) tradicional. O operador de mutação não é utilizado.
3. Um método eficiente de busca local é proposto para ser utilizado sobre parte da população a cada geração, realizando uma busca intensiva na vizinhança das melhores soluções encontradas pelo AE com o objetivo de atingir um número maior de soluções ótimos locais num problema de otimização.

A seguir é descrito cada uma das etapas do algoritmo evolutivo proposto. São elas: representação do indivíduo, função de aptidão, geração da população inicial, estratégia de seleção dos indivíduos reprodutores, mecanismo de cruzamento e procedimento de busca local.

3.1 Estrutura do indivíduo

Como proposto em Joines *et al.* [JCK96], 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 s = (x_1,...x_m | y_1,...y_n). Cada gene x_i ou y_i assume valores inteiros associados ao grupo (célula/família) em que pertence, sendo feita à alocação de máquinas a células e partes a famílias simultaneamente. Considere um indivíduo, para um problema de 12 máquinas, 15 partes e 4 células/famílias:

$$s_1 = (2\ 3\ 2\ 3\ 2\ 3\ 4\ 1\ 3\ 4\ 3\ 1 \mid 2\ 4\ 2\ 4\ 1\ 1\ 2\ 2\ 1\ 1\ 3\ 3\ 4\ 4\ 3).$$

Cada gene x_i indica para qual célula cada máquina é alocada. Por exemplo, em s_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 $m+2 = y_2 = 4$ significa que à *parte* P_2 é alocada para a *família* 4.

3.2 Função de aptidão

A função de aptidão ou função de avaliação que se deseja maximizar é a Eficácia de Agrupamento (*grouping efficacy*), a qual é também usada em Joines *et al.* [JCK96] e por Resende & Gonçalves [RG04]. A Eficácia de Agrupamento, representada por **Γ**, tem a seguinte definição:

$$\Gamma = \frac{e - e_0}{e + e_v} \tag{3}$$

onde:

e = o número de operações (1's) na matriz de entrada;

e_V = o número de lacunas nos blocos diagonais (células / famílias);

e_0 = número de elementos excepcionais (elementos preenchidos com 1 que não pertencem às células / famílias).

Observa-se que a função de aptidão definida em (3) procura maximizar a densidade dentro de cada cluster e minimizar o número de elementos excepcionais (ou ruídos) minimizando a dependência entre os clusters.

3.3 Geração da população inicial

A população inicial no algoritmo proposto constitui-se de indivíduos gerados parte aleatoriamente e parte gerados através de um procedimento de construção heurística. Este procedimento tem sua idéia baseada na heurística gulosa apresentada em Wang [Wa99] para resolver o PFCM, mas é aqui adaptado para que se possa gerar indivíduos diferentes a cada execução. Para isso, a heurística passa a ter uma estrutura de construção similar a etapa de construção do GRASP [FR95], [RR03].

3.3.1 Heurística construtiva randomizada para gerar uma população inicial

A heurística construtiva randomizada (HCR) é aqui proposta para obter soluções iniciais distintas para o PFCM. A idéia básica da HCR é descrita a seguir: Inicialmente é calculada e armazenada uma medida de similaridade entre as máquinas para determinar os 3 pares de máquinas menos similares. É tomada como medida de similaridade entre duas máquinas *i* e *j*, o número de partes em comum atendidas pelas mesmas. Considere os conjuntos Par_j, com j = 1, 2 e 3 como sendo respectivamente o 1º, 2º e 3º par de máquinas menos similares. Considere também: o número de máquinas igual a *m*, o número de partes igual a *p*, as variáveis binárias m_{zi} e m_{zk} iguais a 1 se as máquinas *i* e *k* realizam uma operação de manufatura sobre a parte *z* e 0 caso contrário, a variável $similaridade_{ik}$ representando a similaridade entre as máquinas *i* e *k* e a variável $similaridade_{\text{minima}}$ representando a menor similaridade entre todos os pares de máquinas; os 3 pares de máquinas menos similares podem ser determinados de acordo com o algoritmo descrito na Figura 3.1. Com base nestes três pares de máquinas aqui denominadas de *máquinas semente*, a construção de cada indivíduo é constituída de quatro fases:

1. Na 1ª fase é escolhido aleatoriamente um par dentre os 3 pares de máquinas semente como sendo os 2 clusters iniciais da solução parcial, cada um contendo uma máquina. No caso de empate entre pares será escolhido o que for determinado primeiro, de acordo com a ordem de determinação dos pares de máquinas.

2. Na 2ª fase, cada novo cluster inicial é determinado como sendo a máquina menos similar aos clusters já formados. Como similaridade entre uma máquina e os clusters já formados, entende-se como o número de partes atendidas em comum pela máquina em questão e por pelo menos um dos clusters já formados. Em caso de empate entre máquinas, a máquina com menor rótulo é escolhida. Este processo se repete até que sejam gerados os *k* clusters, onde *k* é um parâmetro de entrada da heurística, sendo que para cada solução a ser construída no AE, é escolhido aleatoriamente um número de clusters *k* a serem formados dentro da faixa de valores que varia de 2 a $\lceil m/2 \rceil$ clusters. Após o término desta fase, os *k* clusters iniciais estarão determinados.

3. Na 3ª fase, o objetivo é alocar as máquinas i ainda não alocado a nenhum cluster parcial (máquinas livres). Primeiramente são determinados o 1º e o 2º cluster com maior índice de similaridade para a máquina *i* de menor rótulo ainda não pertencente a um cluster. Onde o índice de similaridade entre uma máquina *i* e um cluster *k* é determinando como sendo o número de partes atendidas em comum pela máquina *i* e por pelo menos uma máquina pertencente ao cluster *k*.
   A seguir, a máquina i é atribuída aleatoriamente a um dos dois clusters selecionados, desde que a restrição do número máximo de máquinas por cluster não esteja violado. Caso isso aconteça, a máquina é atribuída a um cluster qualquer que não tenha excedido o número máximo de máquinas. Esta fase é então reiniciada para que a próxima máquina livre seja associada a algum cluster e termina quando todas as máquinas livres tiverem sido clusterizadas. A escolha aleatória das fases 1 e 3 permite a construção de soluções distintas a cada nova execução deste procedimento.

4. A 4ª fase cuida da atribuição das partes aos clusters, de maneira idêntica à atribuição das máquinas aos clusters feita na 3ª fase, exceto pelo cálculo da similaridade entre uma parte *i* e um cluster *k*, que é dado como sendo o número de máquinas do cluster *k* que executam uma operação de manufatura sobre a parte *i*.

```
1: Procedimento Pares Maquinas Semente
2:   Para j=1,...,3 faça
3:     similaridade_minima = p+1
4:     Para i=1,....,m-1 faça
5:        Para k=i+1,...,m faça
6:           Se j==1 então
7:           calcule similaridade_ik= Σ_{z=1}^{p} m_zi m_zk
8:              Se similaridade_ik < similaridade_minima então
9:                 similaridade_minima = similaridade_ik
10:                maquina_1 = i
11:                maquina_2 = k
12:             Fim-Se
13:          Fim-Se
14:          Se j==2 e Par_1 ∩ {i,k} ≤ 1 então
15:             Se similaridade_ik < similaridade_minima então
16:                similaridade_minima = similaridade_ik
17:                maquina_1 = i
18:                maquina_2 = k
19:             Fim-Se
20:          Fim-Se
21:          Se j==3 e Par_1 ∩ {i,k} ≤ 1 e Par_2 ∩ {i,k} ≤ 1 então
22:             Se similaridade_ik < similaridade_minima então
23:                similaridade_minima = similaridade_ik
24:                maquina_1 = i
25:                maquina_2 = k
26:             Fim-Se
27:          Fim-Se
28:       Fim-Para
29:    Fim-Para
30:    Par_j = maquina_1 ∪ maquina_2
31:  Fim-Para
32: Fim Procedimento
```

Figura 3.1 – Pseudocódigo do procedimento de determinação dos 3 pares de máquinas iniciais menos similares.

O número máximo de máquinas por cluster é calculado como sendo igual a: $\lceil n^o$ *máquinas/* n^o *clusters* $\rceil$ e o número máximo de partes por cluster igual a $\lceil$ nº partes/nº clusters $\rceil$.

Para apresentar o algoritmo da 1ª e 2ª fases do procedimento de construção (Figura 3.2), temos as seguintes variáveis a considerar:

$\boldsymbol{n_{clusters}}$ *= representa o número de clusters iniciais a serem formados;*

$\boldsymbol{Par_1, Par_2 e Par_3}$ *= os 3 pares de máquinas menos similares determinados pelo algoritmo da Figura 3.1;*

$\boldsymbol{Par_{escolhido}}$ *= par de máquinas escolhido aleatoriamente dentre Par1, Par2 e Par3;*

$\boldsymbol{C_k}$ *= k-ésimo cluster inicial formado;*

Cromo *= vetor de inteiros que representa o indivíduo;*

$\boldsymbol{similaridade_{minima}}$ *= menor similaridade dentre as similaridades entre as máquinas e os clusters iniciais já formados;*

$\boldsymbol{similaridade_{Ci}}$ *= similaridade entre os clusters já formados e uma máquina i;*

$\boldsymbol{maq_{escolhida}}$ = *máquina escolhida para ser um novo cluster inicial;*

$\boldsymbol{m_{zi}}$ = *variável binária que é igual a 1 se a máquina i realiza uma operação de manufatura na parte z e 0 caso contrário;*

$\boldsymbol{m_{zCc}}$= *variável binária que é igual a 1 se o cluster inicial c realiza uma operação de manufatura na parte z e 0 caso contrário;*

$\boldsymbol{p}$ = *número de partes;*

$\boldsymbol{m}$ = *número de máquinas;*

$\boldsymbol{partes_{atendidas}}$ = *vetor de valores inteiros de tamanho p que armazena o estado de uma parte em relação às máquinas semente (0 caso ela não seja atendida pelas máquinas semente e 1 caso contrário).*

```
1: Procedimento Forma Clusters Iniciais(n_clusters)
2:    Par_escolhido = escolha aleatória entre Par_1, Par_2 e Par_3
3:    C_1 = maquina_1 ∈ Par_escolhido
4:    C_2 = maquina_2 ∈ Par_escolhido
5:    Cromo[maquina_1] = 1
6:    Cromo[maquina_2] = 2
7:    Para z=1,...,p faça partes_atendidas[z] = -1
8:    Para z=1,...,p faça
9:       Se m_zC1==1 ou m_zC2==1 então
10:         partes_atendidas[z] = 1
11:      Fim-Se
12:   Fim-Para
13:   Para k=3,...,n_clusters faça
14:      similaridade_mínima = p+1
15:      Para i=1,...,m faça
16:         Se i ∉ C_c ∀ c = 1,...,k-1 então
17:            similaridade_Ci = 0
18:            Para z=1,...,p faça
19:               Se partes_atendidas[z]==1 e m_zi==1 então
20:                  similaridade_Ci++
21:               Fim-Se
22:            Fim-Para
23:            Atualizar(similaridade_minima)
24:         Fim-Se
25:      Fim-Para
26:      maq_escolhida = EscolherMaquina(similaridade_minima)
27:      C_k = maq_escolhida
28:      Para z=1,...,p faça
29:         Se partes_atendidas[z]==-1 e m_zmaqescolhida==1 então
30:            partes_atendidas[z] = 1
31:         Fim-Se
32:      Fim-Para
33:      Cromo[maq_escolhida]= k
34:   Fim-Para
35: Fim Procedimento
```

Figura 3.2 – Pseudocódigo do procedimento de determinação dos *k* clusters iniciais.

Para a associação das *i* máquinas restantes (máquinas livres), considere C_1 e C_2 como sendo os dois melhores clusters (dois clusters com maior similaridade) para a máquina livre *i* e a variável $tamanho_{maximo}$ como sendo o número máximo de máquinas por cluster.

O pseudocódigo da Figura 3.3 representa o algoritmo que associa cada máquina livre restante a um determinado cluster. Para a associação das partes aos clusters (Figura 3.4), considere as variáveis mostradas no pseudocódigo da Figura 3.3 com as seguintes alterações:

$\boldsymbol{similaridade_{Ci}}$ = *similaridade entre o cluster C e a parte i;*

$\boldsymbol{x_{zCi}}$ = *variável binária que é igual a 1 se a máquina z pertencente ao cluster C realiza uma operação de manufatura sobre a parte i e 0 caso contrário;*

$\boldsymbol{n}$ = *número de máquinas de um cluster C;*

$\boldsymbol{tamanho_{maximo}}$ = *número máximo de partes por cluster.*

O algoritmo para a associação de partes é o descrito na Figura 3.4:

```
1: Procedimento Clusteriza Maquinas(n_clusters)
2:    Para i=1,...,m faça
3:      Se i ∉ C_c ∀ c = 1,...,n_clusters então
4:         Para c=1,...,n_clusters faça
5:            Se Tamanho(C_c) < tamanho_maximo
6:               similaridade_Ci = 0
7:               Para z=1,...,p faça
8:                  Se m_zi==1 então
9:                     Se ∃ uma máquina k ∈ C_c | m_zk==1 então
10:                       similaridade_Ci++
11:                    Fim-Se
12:                 Fim-Se
13:              Fim-Para
14:              Atualizar(C_1,C_2)
15:           Fim-Se
16:        Fim-Para
17:        cluster_escolhido = escolher aleatoriamente entre C_1 e C_2
18:     Fim-Se
19:     Cromo[i] = cluster_escolhido
20:   Fim-Para
21: Fim Procedimento
```

Figura 3.3 – Pseudocódigo do procedimento de clusterização das máquinas livres.

```
1:  Procedimento Clusteriza Partes(n_clusters)
2:    Para i=1,...,p faça
3:       Para C=1,...,n_clusters faça
4:          similaridade_Ci = Σ_{z=1}^{n} x_zCi  onde máquina z ∈ cluster C
5:          Atualizar(C1,C2)
6:       Fim-Para
7:       cluster_escolhido = escolher randomicamente entre C1 e C2
8:       Se Tamanho(cluster_escolhido) < tamanho_Maximo então
9:          cluster_escolhido = cluster_escolhido ∪ i
10:      Fim-Se
11:      Senão
12:        cluster_escolhido = C_C qualquer | Tamanho(C_C) < tamanho_maximo
13:      Fim-Senão
14:      Cromo[m+i] = cluster_escolhido
15:   Fim-Para
16: Fim Procedimento
```

Figura 3.4 – Pseudocódigo do procedimento de clusterização das partes livres.

Finalmente, o procedimento de construção de um indivíduo de acordo com a heurística construtiva randomizada (HCR) pode ser representado pelo pseudocódigo da Figura 3.5.

```
1: Procedimento Constroi Individuos
2:    Procedimento Forma Clusters Iniciais(n_clusters)
3:    Procedimento Clusteriza Maquinas(n_clusters)
4:    Procedimento Clusteriza Partes(n_clusters)
5: Fim Procedimento
```

Figura 3.5 – Pseudocódigo do procedimento de construção de indivíduos.

No sentido de explicar os passos da heurística HCR, é ilustrado a seguir, o seu desenvolvimento de forma resumida através do seguinte exemplo: Considere a entrada do PFCM a matriz da Figura 3.6; suponha o número de clusters a serem formados seja igual a k = 3; o número máximo de máquinas por cluster igual a 2 e o número máximo de partes por cluster igual a 3. Cada cluster *i* é denotado por $C_i[x_1\text{-}x_2\text{-}...\text{-}x_m; y_1\text{-}y_2\text{-}...\text{-}y_n]$ onde a primeira e segunda seqüências indicam respectivamente as máquinas e partes que compõe o cluster.

	P1	P2	P3	P4	P5	P6	P7	P8
M1	1			1			1	
M2		1			1			1
M3	1		1	1			1	
M4	1	1			1			1
M5			1			1		
M6			1			1		

Figura 3.6 – Exemplo de uma matriz de entrada do PFCM.

Uma possível solução a ser determinada pela heurística construtiva randomizada pode ser a seguinte:

1. Como as similaridades entre os pares de máquinas são: [M1-M2] = 0; [M1-M3] = 3; [M1-M4] = 1; [M1-M5] = 0; [M1-M6] = 0; [M2-M3] = 0; [M2-M4] = 3; [M2-M5] = 0; [M2-M6] = 0; [M3-M4] = 1; [M3-M5] = 1; [M3-M6] = 1; [M4-M5] = 0; [M4-M6] = 0 e [M5-M6] = 2. Os três pares mais dissimilares pela ordem na qual aparecem são: [M1-M2] = 0; [M1-M5] = 0; e [M1-M6] = 0. Suponhamos que o par escolhido aleatoriamente seja o 2º par. Então os dois clusters iniciais são o cluster 1 formado pela máquina M1 e o cluster 2 formado pela máquina M5, denotados por $C_1[1]$ e $C_2[5]$.
2. O terceiro cluster será inicialmente formado pela máquina com menor nível de similaridade com $C_1[1]$ e $C_2[5]$. Em caso de empate é escolhida a de menor rótulo. Desta forma, o 3º cluster será formado pela máquina M2, que possui similaridade 0 com os 2 clusters iniciais. O cluster 3 inicialmente será composto pela máquina M2: $C_3[2]$. Desta forma os $k = 3$ clusters iniciais já estão formados. São eles: $C_1[1]$, $C_2[5]$ e $C_3[2]$.
3. A fase 3 irá atribuir uma a uma as máquinas restantes (máquinas livres) a um dos k clusters para o qual cada uma delas possuir maior similaridade. Assim seguindo o critério adotado pela heurística, a máquina livre M3 será atribuída ou ao cluster 1 (com o qual tem similaridade 3) ou ao cluster 2 (com o qual tem similaridade 1). Suponha que o cluster escolhido seja o cluster 1. Então agora o cluster 1 será formado pelas máquinas M1 e M3, isto é, $C_1[1\text{-}3]$. A máquina livre M4 tem similaridade 2 com o cluster 3 e similaridade 1 com o cluster 1 (a parte P1 em comum com as máquinas M1 e M3). Suponha que a máquina M4 seja associada ao cluster 3. Agora o cluster 3 será da forma: $C_3[2\text{-}4]$. Por fim, a máquina livre M6 tem como 2 melhores clusters candidatos, o cluster 2 (com o qual tem similaridade 2) e o cluster 1 (com o qual tem similaridade 1). Supondo que a máquina M6 seja associada ao cluster 2, então $C_2 = C_2[5\text{-}6]$. A configuração dos clusters após a associação das máquinas é a seguinte: $C_1[1\text{-}3]$, $C_2[5\text{-}6]$ e $C_3[2\text{-}4]$.
4. Similarmente as partes serão atribuídas ao 1º ou ao 2º cluster que possui o maior número de máquinas que realizam uma operação de manufatura sobre as mesmas. Para a atribuição das partes, consideremos o 1º e 2º clusters candidatos e as seguintes escolhas randômicas:

   P1: $C_1[1\text{-}3]$ e $C_3[2\text{-}4]$. Cluster escolhido: C_1. Então $C_1 = C_1[1\text{-}3;1]$.
   P2: $C_3[2\text{-}4]$ e $C_1[1\text{-}3;1]$. Cluster escolhido: C_3. Então $C_3 = C_3[2\text{-}4;2]$.
   P3: $C_2[5\text{-}6]$ e $C_1[1\text{-}3;1]$. Cluster escolhido: C_2. Então $C_2 = C_2[5\text{-}6;3]$.
   P4: $C_1[1\text{-}3;1]$ e $C_2[5\text{-}6;3]$. Cluster escolhido: C_1. Então $C_1 = C_1[1\text{-}3;1\text{-}4]$.
   P5: $C_3[2\text{-}4;2]$ e $C_1[1\text{-}3;1\text{-}4]$. Cluster escolhido: C_3. Então $C_3 = C_3[2\text{-}4;2\text{-}5]$.
   P6: $C_2[5\text{-}6;3]$ e $C_1[1\text{-}3;1\text{-}4]$. Cluster escolhido: C_2. Então $C_2 = C_2[5\text{-}6;3\text{-}6]$.
   P7: $C_1[1\text{-}3;1\text{-}4]$ e $C_2[5\text{-}6;3\text{-}6]$. Cluster escolhido: C_1. Então $C_1 = C_1[1\text{-}3;1\text{-}4\text{-}7]$.
   P8: $C_3[2\text{-}4;2\text{-}5]$ e $C_1[1\text{-}3;1\text{-}4\text{-}7]$. Cluster escolhido: C_3. Então $C_3 = C_3[2\text{-}4;2\text{-}5\text{-}8]$.

Após o término do procedimento de construção, a configuração dos clusters é a seguinte: $C_1[1\text{-}3;\ 1\text{-}4\text{-}7]$, $C_2[5\text{-}6;\ 3\text{-}6]$ e $C_3[2\text{-}4;\ 2\text{-}5\text{-}8]$.

Temos então a célula 1 formada pelas máquinas 1 e 3, a célula 2 formada pelas máquinas 5 e 6 e a célula 3 formada pelas máquinas 2 e 4.

A família 1 é formada pelas partes 1, 4 e 7, a família 2 pelas partes 3 e 6 e a família 3 pelas partes 2, 5 e 8. O indivíduo referente à esta clusterização é o seguinte: C = (1 3 1 3 2 2 | 1 3 2 1 3 2 1 3). A Figura 3.7 representa a matriz de clusterização para o referido indivíduo:

	P1	P4	P7	P3	P6	P2	P5	P8
M1	1	1	1					
M3	1	1	1	1				
M5				1	1			
M6				1	1			
M2						1	1	1
M4	1					1	1	1

Figura 3.7 – Uma solução inicial obtida pelo procedimento de construção HCR a partir da matriz de entrada da Figura 3.6.

Para justificar o uso da heurística HCR aqui proposta para gerar a população inicial do AE, podemos citar os seguintes argumentos: a de que através do HCR, se pode gerar indivíduos diferentes que representam soluções viáveis do PFCM nas quais máquinas e partes estão clusterizadas de acordo com um critério de similaridade, ao contrário de indivíduos gerados aleatoriamente onde não há nenhum critério para a associação de máquinas e partes a clusters e além disso na geração aleatória, o risco de obter indivíduos inviáveis é grande. Retornando ao AE, o próximo passo irá tratar das etapas de seleção e reprodução de novos indivíduos.

3.4 Estratégia de seleção dos indivíduos reprodutores

Esta seção trata da escolha dos indivíduos que irão servir como reprodutores (*indivíduos pais*) no processo de geração de novos indivíduos. Escolher sempre os melhores indivíduos para reprodutores, pode levar o algoritmo evolutivo a uma parada prematura em ótimos locais ainda distantes de um ótimo global em problemas de otimização. 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 menos aptos. Versões básicas desta técnica simplesmente somam o valor da aptidão de todos os P indivíduos de uma população (seja T este valor). Cria-se então o intervalo $[0, T]$ que é particionado em P sub-intervalos, onde cada sub-intervalo está associado a um indivíduo. Obviamente os indivíduos mais aptos irão possuir intervalos maiores. Definidos estes sub-intervalos, sorteia-se números dentro do intervalo $[0, T]$. Os indivíduos sorteados serão escolhidos como reprodutores. Uma versão mais eficiente desta estratégia foi usada em [JCK96] baseada na equação descrita em (4):

$$\sum_{população} \mathrm{P}[\text{seleção do r}-\text{ésimo indivíduo}] = \sum_{i=1}^{P} q'(1-q)^{r-1} \qquad (4)$$

onde:

P *= número de indivíduos na população;*

q *= probabilidade de seleção do melhor indivíduo;*

q' *= $q/(1-(1-q)^P)$;*

r *= rank do indivíduo (onde r = 1 é o melhor indivíduo).*

A equação (4) fornece o somatório das probabilidades de seleção de todos os P indivíduos de uma população.

Vejamos então um exemplo das probabilidades de escolha de indivíduos de uma população de *P* indivíduos. Os parâmetros de entrada escolhidos para a equação (4) serão por exemplo: *q = 0.10; P = 10* (tamanho da população). Para os *P* indivíduos ordenados do melhor para o pior (1º a 10º), utilizando a equação (4) temos os seguintes valores associados:

1 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{0}] = 0,15337$
2 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{1}] = 0,13803$
3 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{2}] = 0,12423$
4 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{3}] = 0,11180$
5 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{4}] = 0,10062$
6 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{5}] = 0,09056$
7 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{6}] = 0,08150$
8 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{7}] = 0,07335$
9 - $[(0,10/(1-(0,9)^{10}) * (0,9)^{8}] = 0,06602$
10- $[(0,10/(1-(0,9)^{10}) * (0,9)^{9}] = 0,05942$

O somatório dos valores acima é igual a 0,9989, ou seja, aproximadamente 1, que é a probabilidade total. Então para calcular a probabilidade de cada indivíduo; é sorteado um número aleatório entre 0 e 1 e de acordo com o valor deste número, um indivíduo é selecionado. Os indivíduos, do 1º ao 10º, para o exemplo acima, serão selecionados de acordo com os seguintes sub-intervalos:

1º - se número sorteado estiver entre 0 e 0,15337
2º - se número > 0,15337 e <= a 0,2914 (=0,15337 + 0,13803)
3º - se número > 0,2914 e <= a 0,41563 (= 0,2914 + 0,12423); e assim sucessivamente até o 10º indivíduo.

Através do valor do parâmetro de entrada *q* pode-se estabelecer uma equação que dá mais chances aos primeiros indivíduos ou que "*distribui melhor*" as chances entre os indivíduos. Por exemplo, se o parâmetro *q* for igual a *1*, somente o melhor indivíduo terá valor de probabilidade diferente de zero e, portanto, somente ele será escolhido. Nos testes preliminares utilizamos no AE proposto, diferentes valores para *q (0.6, 0.7, 0.8, 0.9 e 0.1)* e para *q = 0.7* foi obtido os melhores resultados, valor este que foi então usado para o conjunto das instâncias da literatura (em [JCK96] foi usado o valor *q = 0.10*).

3.5 Mecanismo de cruzamento

O operador *crossover* usado tradicionalmente nos algoritmos genéticos foi substituído no AE proposto por um mecanismo de cruzamento mais genérico que faz uso do conceito de freqüência.

A primeira diferença é que na etapa de cruzamento dois ou mais indivíduos pais podem ser utilizados, e não exatamente dois pais como nos operadores tradicionais do tipo *crossover*. Escolhidos os *N_pais* pais distintos para o cruzamento (através do processo de seleção referenciado no item anterior deste trabalho), onde $N\_pais \geq 2$ é um parâmetro de entrada do cruzamento. Cada gene do filho referente a uma máquina receberá aleatoriamente o valor correspondente a uma das *C* células de máquina nas quais a presença desta máquina é mais freqüente, com base nas *N_pais* soluções pais, onde *C* é um parâmetro de entrada. Para melhor entendimento deste procedimento considere a seguinte ilustração:

Por exemplo, suponha que temos *N_pais = 10* indivíduos pais e, queremos saber no indivíduo filho para qual célula de máquinas será alocado a máquina M1. Analisando os *10* pais suponha que em 4 deles, *M1* está associada à célula C2; em outros 3, na célula C1; em dois outros na célula C3, e finalmente em um indivíduo pai, *M1* está alocado na célula C4. Desta forma temos neste exemplo uma lista de 4 células candidatas que irão compor a lista de células candidatas (LC) para receber a máquina *M1*.

Como o tamanho desta lista pode ser muito grande, selecionamos uma lista de candidatos restrita (LCR) composta dos $C = |LCR|$ melhores candidatos de LC. Neste problema, essa classificação será dada pelo número de células comuns que recebe cada máquina. Assim no exemplo anterior, para C = 2, teremos LCR = {C2, C3}.

Obtida a LCR, uma célula de LCR será escolhida aleatoriamente para receber a máquina *M1*. Para a escolha da célula para *M2,* uma nova LCR será formada e assim sucessivamente. Esta lista de candidatos restrita (LCR) de células geradas para cada máquina e a escolha posterior de um elemento desta lista aleatoriamente, permite que diversos filhos distintos possam ser gerados através deste conjunto de pais. Esta lista LCR desempenha a mesma função das LCR da etapa de construção da metaheurística GRASP [FR95], [RR03].

Similarmente para cada parte é construída uma lista restrita das *M* máquinas que mais aparecem no mesmo cluster da referida parte, com base nos *N_pais* soluções pais, sendo *M* também um parâmetro de entrada. Neste caso, é também gerada para cada parte, uma lista de candidatos restritos (LCR) de máquinas. Após a clusterização das máquinas para o indivíduo filho, para cada parte uma máquina então é escolhida aleatoriamente na respectiva LCR de cada parte. Então a parte é associada à família correspondente à célula da máquina escolhida, com base nos genes de máquina do indivíduo filho em formação.

O uso das LCR neste caso tem primordialmente dois propósitos maiores: O primeiro, propiciar que a partir de um conjunto de indivíduos pais seja possível gerar diversos indivíduos filhos distintos. O segundo objetivo é que escolhendo um elemento da LCR, possibilite que candidatos de qualidade sejam selecionados.

Observe que nos casos particulares onde a LCR possui um único elemento, estaremos fazendo a opção pela escolha gulosa e no caso onde LCR contém todos os candidatos possíveis, a escolha será totalmente aleatória. A medida que a cardinalidade de LCR for reduzindo, o procedimento irá efetuar escolhas cada vez mais gulosas mas com mais risco de gerar filhos idênticos. Para a apresentação do pseudocódigo do mecanismo de reprodução, considere as seguintes variáveis:

N_pais = *número de pais;*

m = *número de máquinas;*

p = *número de partes;*

Pop = *população de indivíduos corrente;*

$\boldsymbol{Pai_j}$ = *j-ésimo indivíduo pai, formado por m genes de máquina e p genes de partes. Cada gene (máquina ou parte) possui o valor do cluster ao qual está associado;*

$\boldsymbol{F_i}$ = *i-ésimo indivíduo filho, formado por m genes de máquina e p genes de partes. Cada gene (máquina ou parte) possui o valor do cluster ao qual está associado;*

N_Filhos = *número de filhos a serem formados.*

O pseudocódigo da Figura 3.8 representa o algoritmo de cruzamento:

```
1: Procedimento Cruzamento(c,M)
2:   Escolhe_pais(N_pais,Pop)
3:   Para i=1,...,m faça
4:      Para j=1,...,N_pais faça
5:         Incluir_lista_cluster(C,Pai_j[i],i)
6:      Fim-Para
7:   Fim-Para
8:   Para i=1,...,p faça
9:      Para j=1,...N_pais faça
10:        Para k=1,...,m faça
11:           Se Pai_j[k]==Pai_j[i+m] então
12:              Incluir_maq_lista_parte(M,k,i)
13:           Fim-Se
14:        Fim-Para
15:     Fim-Para
16:  Fim-Para
17:  Para i=1,...,N_Filhos faça
18:     Para k=1,...,m faça
19:        F_i[k] = Escolher_cluster_maquina(k)
20:     Fim-Para
21:     Para j=1,...,p faça
22:        F_i[k+j] = F_i[Escolher_maq_parte(j)]
23:     Fim-Para
24:  Fim-Para
25: Fim Procedimento
```

Figura 3.8 – Pseudocódigo do procedimento de reprodução.

Sobre o pseudocódigo da Figura 3.8 observe que os procedimentos:

Escolhe pais(),
Incluir_lista_cluster(),
Incluir_maq_lista_parte(),
Escolher_cluster_maquina() e
Escolher_maq_parte()

têm as seguintes funcionalidades: *Escolhe_pais()* seleciona os *N_pais* indivíduos pai. A função *Incluir_lista_cluster()* inclui numa lista de clusters para a máquina *i*, o rótulo dos clusters nos quais a referida máquina *i* é mais freqüentemente associada, de acordo com os genes de máquina dos indivíduos pais.

Esta lista é ordenada pela freqüência de ocorrência dos clusters. A função *Incluir_maq_lista_parte()* inclui a máquina *k* numa lista de máquinas com as quais uma parte *i* é mais freqüentemente associada, de acordo com os genes de máquinas e de partes dos indivíduos pais. A função *Escolher_cluster_maquina()* escolhe um cluster na LCR de clusters de uma máquina *k* e a função *Escolher_maq_parte()* escolhe uma máquina na LCR de máquinas de uma determinada parte *j*.

3.6 Procedimento de busca local proposto

O melhoramento das soluções num AE e em particular nos algoritmos genéticos (AGs) pode ser obtido de geração a geração através da etapa de reprodução onde existe uma tendência à manutenção dos bons indivíduos (boas soluções) no processo de reprodução de indivíduos filhos onde estes herdam as boas características dos pais através dos operadores de cruzamento e mutação.

Contudo é reconhecido que estes procedimentos estão longe de serem considerados mecanismos eficientes de busca local. Isso tem motivado pesquisadores a incorporar nos AGs tradicionais, módulos de busca local mais efetiva gerando com isso variantes dos AGs conhecidos como Algoritmos Meméticos [Mos89] , AGs Híbridos [RG04], [FL00], [ERP02] ou simplesmente Algoritmos Evolutivos (AEs) [ODV98], [OVD01], [DO03], [SOMD06].

A busca local tem como meta, a partir de uma solução inicial, analisar uma vizinhança deste em busca de uma solução ótimo local em problemas de otimização.

Neste trabalho foi incorporado ao AE um procedimento de busca local inspirado no modelo proposto por Resende e Gonçalves em [RG04].

A busca local proposta, 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 associadas de acordo com as famílias formadas anteriormente, obtendo um novo conjunto de células de máquina. Estas duas etapas são executadas alternadamente enquanto houver melhoria na aptidão da solução. Mais detalhadamente, os passos são os seguintes:

Passo 1: Para cada parte *i*, é calculado o coeficiente descrito abaixo, com relação a cada cluster *k*.

$$coef_{ik} = \begin{cases} N_{1intrak} - N_{ruidosk} - N_{0intrak}, \text{ se } N_{1intrak} \neq 0; \\ -\infty, \text{ caso contrário} \end{cases} \quad (5)$$

onde:

$\boldsymbol{N_{1intrak}}$ = *número de elementos do cluster k iguais a 1 se a parte i for associada a este cluster;*

$\boldsymbol{N_{ruidosk}}$ = *número de elementos iguais a 1 fora do cluster k se a parte i for associada a este cluster;*

$\boldsymbol{N_{0intrak}}$ = *número de elementos iguais a 0 dentro do cluster k se a parte i for associada a este cluster.*

Após esta etapa, cada parte é associada ao cluster para o qual possui o maior coeficiente definido em (5). Caso haja empate entre coeficientes, 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.

Não havendo 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 2.

Passo 2: O passo 2 é idêntico ao passo 1, mas agora efetua-se a mudança das máquinas de cluster calculando-se os coeficientes para cada máquina em relação a cada cluster, de acordo com a alocação das partes feitas pelo passo 1. O coeficiente de cada máquina *i* em relação a cada cluster *k* é calculado pela equação (5), onde a notação agora tem a seguinte interpretação:

$N_{1intrak}$ = *número de elementos do cluster k iguais a 1 se a máquina i for associada a este cluster;*

$N_{ruidosk}$ = *número de elementos iguais a 1 fora do cluster k se a máquina i for associada a este cluster;*

$N_{0intrak}$ = *número de elementos iguais a 0 dentro do cluster k se a máquina i for associada a este cluster.*

De forma similar ao Passo 1, cada máquina é associada ao cluster para o qual possui o maior coeficiente. 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.

Não havendo melhoria na aptidão da solução, as máquinas são alocadas aos clusters originais e a busca local é finalizada. Caso haja melhoria, executa-se o passo 1 novamente.

A penalidade aplicada ao valor de $coef_{ik}$ quando $N_{1intrak}$ é igual a zero evita que a busca local associe uma parte (máquina) a um cluster que não possui máquinas (partes) que a servem (utilizam).

A busca local proposta por Resende e Gonçalves em [RG04] possui os mesmos passos de nossa busca local, com a diferença que o coeficiente usado para a clusterização das máquinas e partes é diferente. Em [RG04], o coeficiente usado é descrito em (6), que é a mesma função usada na parte evolutiva dada pela equação (3).

$$coef_{iC} = \frac{N_1 - N_{1,C}^{Out}}{N_1 + N_{0,C}^{In}} \qquad (6)$$

Ou seja, em [RG04], os autores usam na busca local, a mesma forma de avaliação usada na parte evolutiva do algoritmo, enquanto no AE, usamos na busca local uma outra forma de avaliação no sentido de buscar através de avaliações distintas, diversificar a população da próxima geração do algoritmo evolutivo. Contudo vale ressaltar que a solução final do AE, é sempre avaliada com a função de aptidão padrão da fase evolutiva dada em (3).

Considerando o número de máquinas igual a *m*, o número de partes igual a *p* e o número de clusters igual a $n_{clusters}$, o pseudocódigo da Figura 3.9 representa o procedimento de busca local aqui proposto.

```
1:  Procedimento Busca Local
2:    melhoria = 1
3:    Enquanto melhoria==1 faça
4:      Para i=1,...,p faça
5:        Para k=1,...,n_clusters faça
6:          calcule coef_ik
7:          atualizar Max(coef_ik)
8:        Fim-Para
9:      Fim-Para
10:     re-associar as partes de acordo com Max(coef_ik), ∀ i=1,...,p
11:     Se solução não melhorou então
12:       associar partes à clusters originais
13:       melhoria = 0
14:     Fim-Se
15:     Se melhoria==1 então
16:       Para i=1,...m faça
17:         Para k=1,...,n_clusters faça
18:           calcule coef_ik
19:           atualizar Max(coef_ik)
20:         Fim-Para
21:       Fim-Para
22:       reassociar as máquinas de acordo com Max(coef_ik), ∀ i=1,...,m
23:       Se solução não melhorou então
24:         associar máquinas à clusters originais
25:         melhoria = 0
26:       Fim-Se
27:     Fim-Se
28:   Fim-Enquanto
29: Fim Procedimento
```

Figura 3.9 – Pseudocódigo do procedimento de busca local proposto.

Após a descrição detalhada de cada etapa do AE proposto, uma síntese deste pode ser visto na Figura 3.10.

```
Algoritmo Evolutivo
      Gera populacao inicial(Pop)
      n° de gerações = 1
      melhor indivíduo = melhor indivíduo(Pop)
      Enquanto n° de gerações < n° máximo de gerações faça
            Pop+1 = Ø
            reproducao(Pop,Pop+1)
            Pop = Pop+1
            Busca local(Pop)
            Atualizar melhor indivíduo
            n° de gerações++
      Fim-Enquanto
      Imprimir melhor indivíduo
Fim Algoritmo Evolutivo
```

Figura 3.10 – Pseudocódigo do algoritmo proposto onde: Pop = população corrente
Pop_{+1} = população seguinte à população corrente.

4. Resultados Computacionais

Para avaliar o AE proposto, foram utilizadas as 36 instâncias da literatura disponíveis em Resende & Gonçalves [RG04] e Joines *et al.* [JCK96].

A Tabela 4.1 lista as instâncias, as fontes e suas dimensões (*nº de máquinas*) **x** (*nº de partes)*. No que diz respeito ao número de clusters das soluções obtidas, o AE aqui proposto não realiza uma análise desta característica das soluções do PFCM, podendo surgir soluções com número de clusters diferentes, mas com o mesmo valor de aptidão (Recordando que no AE foi usado a mesma função de aptidão usada em [JCK96] e em [RG04]).

Neste caso, o AE proposto armazena a primeira melhor solução encontrada, não tendo mecanismos de análise para avaliar *o melhor número de clusters*.

Entretanto, esta análise pode vir a ser interessante, pois isto possibilita ao projetista do sistema de produção uma maior liberdade de escolha na configuração do mesmo.

Para avaliar o algoritmo evolutivo proposto, tomamos como parâmetro de comparação o algoritmo genético híbrido proposto recentemente em 2004 por Mauricio Resende e Gonçalves em [RG04] e aqui denotado por HGA, o qual segundo seus autores detém os melhores resultados aproximados da literatura até então.

Em relação ao artigo descrito em [RG04], utilizamos para as nossas comparações, as mesmas instâncias disponibilizadas e os resultados por ele obtidos ou listados.

Vale esclarecer que as diferenças entre o HGA [RG04] e o AE aqui proposto são basicamente as seguintes:

i) No HGA [RG04] a população inicial de indivíduos é gerada aleatoriamente, enquanto no AE proposto, somente uma parte da população é gerada aleatoriamente e outra parte gerada através da heurística HCR descrita na secção 3.3.1;

ii) O operador de cruzamento no AE proposto, combina informações de $N \geq 2$ indivíduos pais e não exatamente $N = 2$ como no HGA. Além disso, no AE proposto, cada gene do indivíduo filho é obtido a partir da freqüência de cada máquina/parte em cada célula/família dos N pais, como descrito na seção 3.5;

iii) Finalmente a busca local do AE proposto utiliza o coeficiente definido em (5) e o HGA o definido em (6).

Os testes computacionais foram realizados em um microcomputador AMD Athlon 1.410 Ghz.

O AE proposto foi executado 10 vezes para cada instância, sendo reportados aqui os melhores resultados para cada uma. Os resultados médios não estão listados, mas o maior *gap* entre a pior solução (das 10 execuções) e a melhor nunca ultrapassou 3%.

Além do HGA, os resultados do AE proposto foram comparados com os resultados obtidos em: Zodiac, proposto por Srinivasan & Narendan (1991); Grafics, proposto por Srinivasan & Narendan (1991); algoritmo de clusterização proposto por Srinivasan em 1991 (Ca); algoritmo genético proposto por Cheng *et al.* em 1998 (Ga); estes resultados estão listados em [RG04].

Adicionalmente o AE foi comparado com o algoritmo genético proposto por Joines *et al*. em [JCK96], e aqui denominado AG-JCK (este algoritmo da literatura não disponibiliza nem instâncias nem códigos, desta forma este algoritmo foi implementado novamente neste trabalho).

Tabela 4.1 – Instâncias da literatura para o PFCM.

Instância	Fonte	Tamanho
1	King & Nakornchai (1982)	5 x 7
2	Waghodecar & Sahu (1984)	5 x 7
3	Seiffodini (1989)	5 x 18
4	Kusiak (1992)	6 x 8
5	Kusiak & Chow (1987)	7 x 11
6	Boctor (1991)	7 x 11
7	Seiffodini & Wolfe (1986)	8 x 12
8	Chandrasekaran & Rajagopalan (1986a)	8 x 20
9	Chandrasekaran & Rajagopalan (1986b)	8 x 20
10	Mosier & Taube (1985a)	10 x 10
11	Chan & Milner (1982)	10 x 15
12	Askin & Subramanian (1987)	14 x 24
13	Stanfel (1895)	14 x 24
14	McCormick (1972)	16 x 24
15	Srinivasan (1990)	16 x 30
16	King (1980)	16 x 43
17	Burbidge (1969)	20 x 35
18	Carrie (1973)	18 x 24
19	Mosier & Taube (1985b)	20 x 20
20	Kumar (1986)	20 x 23
21	Carrie (1973)	20 x 35
22	Boe & Cheng (1991)	20 x 35
23	Chandrasekaran & Rajagopalan (1989)-1	24 x 40
24	Chandrasekaran & Rajagopalan (1989)-2	24 x 40
25	Chandrasekaran & Rajagopalan (1989)-3	24 x 40
26	Chandrasekaran & Rajagopalan (1989)-4	24 x 40
27	Chandrasekaran & Rajagopalan (1989)-5	24 x 40
28	Chandrasekaran & Rajagopalan (1989)-6	24 x 40
29	McCormick (1972)	27 x 27
30	Carrie (1973)	28 x 46
31	Kumar & Vannelli (1978)	30 x 41
32	Stanfela (1985)	30 x 50
33	Stanfelb (1985)	30 x 50
34	King & Nakornchai (1982)	30 x 90
35	McCormick (1972)	37 x 53
36	Chandrasekaran & Rajagopalan (1987)	40 x 100

A razão para a escolha do AG-JCK como o algoritmo genético da literatura a ser aqui implementado, é pelo fato deste usar a mesma função de aptidão usada no AE proposto e no HGA. O número de gerações foi de 150 iterações para o AE, o mesmo número utilizado em [RG04] para o HGA.

Para a execução do AG-JCK, o número de gerações variou de acordo com o tamanho das instâncias, numa tentativa de executá-lo da mesma forma que Joines *et al*. o fizeram originalmente em [JCK96].

O número de gerações para as instâncias foi o seguinte: instâncias 1 a 9, 150; 10 a 14, 1000; 15 a 34, 5000 e 35 e 36, 20000. A medida de desempenho usada por todos os algoritmos analisados nesta seção, foi a *eficácia de agrupamento* definida anteriormente.

Na população inicial do AE, 40% dos indivíduos são gerados aleatoriamente e 60% usando a heurística HCR. Este percentual foi definido a partir de testes preliminares onde diferentes taxas foram usadas.

Percebeu-se que gerando 30, 40, ou 50% da população inicial pela heurística HCR, os resultados finais do AE são muito similares, havendo apenas uma pequena mudança nos tempos finais. Somente no caso onde a heurística HCR não é usada, ou seja, toda a população inicial gerada aleatoriamente, é que se percebeu uma pequena deterioração nos resultados finais do AE, como ilustrado na Tabela 4.4.

A busca local proposta no AE é sempre ativada para as 30% melhores soluções (e deste percentual descartando as duplicações) de cada iteração, enquanto no HGA a busca é aplicada para todas as soluções. A escolha do valor 30% também foi definida a partir de testes preliminares, onde diferentes percentuais foram testadas.

Percebeu-se que usando a busca local sobre 30, 40, ou 50% das melhores soluções, o impacto na qualidade da solução final do AE é relativamente pequeno, mas bastante significativo em termos de tempos computacionais exigidos quando o critério de parada foi o número máximo de iterações. Desta forma, foi fixado o valor 30% para não onerar os tempos finais do AE.

Os valores da *eficácia de agrupamento* estão multiplicados por 100, de maneira a apresentar os resultados em termos percentuais.

A Tabela 4.2 descreve os resultados obtidos pelos algoritmos da literatura e pelo AE proposto. Nesta tabela, a primeira coluna identifica as instâncias e as colunas (2 – 8) mostra os valores obtidos pelas melhores soluções de cada algoritmo.

Finalmente a coluna 9 descreve a melhora em percentual obtida pelo AE em relação a melhor solução da literatura. Valores em negrito indicam os melhores resultados.

Pelos resultados da Tabela 4.2, podemos notar que o AE conseguiu superar os *melhores resultados da literatura* em 8 das 36 instâncias; obteve a *mesma melhor solução da literatura* em outras 26 e para as instâncias 2 e 21 obteve uma solução com pior aptidão que a melhor da literatura.

Porém vale ressaltar que a solução encontrada para a instância 2 pelo algoritmo Ga, como relatado em Resende e Gonçalves [RG04], contém clusters unitários que são considerados inválidos (célula com uma única máquina e/ou família com uma única parte) no nosso AE e no HGA. Desta forma, quando executamos nosso algoritmo para aceitar soluções com tais clusters para a instância 2 observamos que o AE alcança também a "melhor solução" obtida por Ga, isso possivelmente ocorre também no HGA.

Assim usando os mesmos critérios de viabilidade, o AE só possui um desempenho pior que o melhor da literatura em um único caso (instância 21) onde a diferença é de um por cento.

Portanto em termos da qualidade da melhor solução obtida, o AE nos testes realizados, se mostra na média superior aos algoritmos da literatura incluindo o HGA [RG04].

Tabela 4.2 – Resultados comparativos de algoritmos da literatura e do AE proposto.

Instância	Zodiac	Grafics	Ca	Ga	AG-JCK	HGA	AE	Melhoria
1	**73,68**	**73,68**	-	-	**73,68**	**73,68**	**73,68**	0%
2	56,52	60,87	-	**68,00(*)**	62,50	62,50	62,50	-8, %(*)
3	77,36	-	-	77,36	**79,59**	**79,59**	**79,59**	0%
4	**76,92**	-	-	**76,92**	**76,92**	**76,92**	**76,92**	0%
5	39,13	53,12	-	46,88	**53,13**	**53,13**	**53,13**	0%
6	**70,37**	-	-	**70,37**	**70,37**	**70,37**	**70,37**	0%
7	**68,29**	**68,29**	-	-	65,12	**68,29**	**68,29**	0%
8	58,33	58,13	**58,72**	58,33	57,26	**58,72**	**58,72**	0%
9	85,24	85,24	85,24	85,24	**85,25**	**85,25**	**85,25**	0%
10	**70,59**	**70,59**	**70,59**	**70,59**	**70,59**	**70,59**	**70,59**	0%
11	**92,00**	**92,00**	-	**92,00**	**92,00**	**92,00**	**92,00**	0%
12	64,36	64,36	64,36	-	**69,86**	**69,86**	**69,86**	0%
13	65,55	65,55	-	67,44	67,05	**69,33**	**69,33**	0%
14	32,09	45,52	48,70	-	47,69	**51,96**	**51,96**	0%
15	**67,83**	**67,83**	**67,83**	-	**67,83**	**67,83**	**67,83**	0%
16	53,76	54,39	54,44	53,89	50,88	**54,86**	**54,86**	0%
17	-	-	-	-	**75,71**	-	**75,71**	0%
18	41,84	48,91	44,20	-	51,24	**54,46**	**54,46**	0%
19	21,63	38,26	-	37,21	41,14	**42,96**	**42,96**	0%
20	38,66	49,36	43,01	46,62	44,57	**49,65**	**49,65**	0%
21	75,14	75,14	75,14	75,28	**76,22**	76,14	76,14	-0,1%
22	51,13	-	-	55,14	56,52	**58,06**	**58,06**	0%
23	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	0%
24	**85,11**	**85,11**	**85,11**	**85,11**	69,59	**85,11**	**85,11**	0%
25	**73,51**	**73,51**	**73,51**	73,03	**73,51**	**73,51**	**73,51**	0%
26	20,42	43,27	51,81	49,37	50,30	51,85	**51,97**	0,23%
27	18,23	44,51	44,72	44,67	45,14	46,50	**47,06**	1,2%
28	17,61	41,67	44,17	42,50	43,64	44,85	**44,87**	0,04%
29	52,14	41,37	51,00	-	54,23	**54,27**	**54,27**	0%
30	33,01	32,86	40,00	-	27,71	43,85	**44,62**	1,76%
31	33,46	55,43	55,29	53,80	55,32	57,69	**58,14**	0,78%
32	46,06	56,32	58,70	56,61	52,54	59,43	**59,66**	0,39%
33	21,11	47,96	46,30	45,93	49,23	**50,51**	**50,51**	0%
34	32,73	39,41	40,05	-	15,78	41,71	**43,37**	3,98%
35	52,21	52,21	-	-	56,83	56,14	**57,54**	2,49%
36	83,66	83,92	83,92	**84,03**	**84,03**	**84,03**	**84,03**	0%

A Tabela 4.3 apresenta para o AE e o AG-JCK, os tempos médios de execução (em segundos) obtidos das 10 execuções em cada instância. Para o HGA são mostrados os tempos (em segundos) disponibilizados em [RG04].

Adicionalmente para AE, AG-JCK e HGA são mostrados a iteração que forneceu a melhor solução (msol). Devemos recordar aqui que o número de iterações máximas para AE e HGA foi 150 entretanto para o AG-JCK este número foi proporcional ao tamanho de cada

instância. As colunas da Tabela 4.3 representam, respectivamente: a identificação da instância, tempos médios do AG-JCK, HGA e AE em segundos e a iteração que forneceu a melhor solução (msol). No caso do AG-JCK e do AE, onde são feitas 10 execuções para cada instância o (msol) é determinado pela execução que forneceu a melhor solução.

Tabela 4.3 – Tempos médios de execução em segundos e gerações das melhores soluções para o AG-JCK, AE e HGA.

Instância	Tempo AGJCK	Tempo HGA	Tempo AE	msol AG-JCK	msol HGA	msol AE
1	0,04	0,28	0,06	16	**1**	2
2	0,03	0,35	0,06	8	**1**	2
3	0,05	0,47	0,11	49	**1**	2
4	0,05	0,41	0,09	9	**1**	2
5	0,07	0,73	0,16	132	6	**2**
6	0,07	0,73	0,15	134	**1**	2
7	0,09	0,96	0,20	32	**1**	2
8	0,14	1,27	0,29	55	**2**	**2**
9	0,13	1,28	0,30	124	**1**	2
10	0,97	1,36	0,28	84	**1**	2
11	1,10	1,46	0,36	142	**1**	2
12	2,97	4,60	1,17	581	10	**3**
13	3,01	4,67	1,25	350	**1**	2
14	3,92	6,53	1,68	296	21	**2**
15	24,10	7,51	2,09	3470	**1**	2
16	32,21	10,50	3,26	627	**1**	2
17	50,49	-	4,02	126	-	**2**
18	27,51	8,28	2,24	555	32	**9**
19	29,85	9,12	2,71	338	50	**28**
20	27,48	9,97	2,84	995	78	**2**
21	52,28	12,09	3,76	222	**1**	2
22	52,92	13,44	4,15	1723	**2**	**2**
23	166,91	14,65	5,61	317	**1**	2
24	89,63	20,04	6,77	1552	**1**	2
25	96,22	21,50	8,76	1074	**1**	2
26	102,63	22,81	9,62	4087	114	**25**
27	94,14	23,05	9,96	4128	117	**26**
28	99,81	22,56	9,69	4508	**75**	139
29	79,70	19,90	6,46	1275	8	**2**
30	124,90	34,12	14,64	923	117	**110**
31	161,02	34,19	17,23	4485	111	**57**
32	165,59	40,95	18,69	3719	113	**2**
33	224,97	41,68	20,63	4189	93	**21**
34	238,49	68,99	39,46	253	**45**	63
35	3387,51	58,35	20,99	16271	**1**	10
36	3684,26	125,33	79,45	14459	3	**2**

Observando os resultados da Tabela 4.3 podemos constatar que em termos do tempo computacional exigido, o AE requer na média um tempo menor que o HGA e AG-JCK. Comparando os tempos entre o AE e HGA, a diferença nos tempos vai se acentuando em favor do AE a medida que as dimensões das instâncias vai aumentando. Isso pode ser justificado pelo fato de no AE menos soluções serem analisadas pela busca local (recordando no AE no máximo 30% e no HGA 100% das soluções são analisadas pelas respectivas buscas locais).

Em termos do número de iterações exigidas para encontrar a melhor solução, o AE chegou à melhor solução antes do HGA em 15 instâncias, depois do HGA em 19 instâncias e para as demais chegou à melhor solução na mesma geração. Observamos também que a diferença em 16 das 19 instâncias em que o HGA chegou antes à melhor solução foi de apenas uma iteração. Com relação ao AG-JCK, para as instâncias 1 a 9, observamos que: o número de gerações foi sempre maior que o exigido pelo HGA e AE, mas o AG-JCK necessita de menos tempo computacional devido ao fato deste possuir apenas os módulos básicos de um algoritmo genético (mutação, *crossover* e geração inicial aleatória) sem incluir por exemplo, um módulo de busca local. Porém, para as demais instâncias, que são de maior dimensão, o AG-JCK necessita de um número maior de gerações e também de mais tempo computacional para alcançar suas melhores soluções que em sua maioria são inferiores às melhores soluções do AE e HGA. A Tabela 4.4 compara os resultados de novos testes computacionais com execuções do AE utilizando o procedimento de construção heurística na população inicial (AE) e sem o procedimento de construção (AE-C), neste último caso, a população inicial é gerada totalmente de forma aleatória. As mesmas instâncias da literatura (listadas anteriormente) foram utilizadas onde efetuamos para cada instância 10 execuções do AE e AE-C.

A primeira coluna da Tabela 4.4 representa a identificação da instância e cada par das demais colunas indicam respectivamente: a aptidão da melhor solução do AE e do AE-C, tempos de CPU (em segundos), média das melhores soluções em 10 execuções, e número da iteração que gerou a melhor solução (pela primeira vez) para o AE e AE-C. Valores em negrito significam que a melhor solução atualizada da literatura (incluindo os resultados anteriores do AE) foi obtido. Observe que nesta nova bateria alguns resultados do AE listados na Tabela 4.2 foram modificados, como por exemplo na instância 21 onde o AE-C conseguiu agora obter a melhor solução da literatura.

A Tabela 4.4 nos mostra que o uso do procedimento de construção de indivíduos (HCR) na população inicial do algoritmo evolutivo conduz a uma melhor solução final em 6 das 36 instâncias, a uma mesma solução final que no AE-C em 28 casos e a uma solução ligeiramente inferior que no AE-C em apenas 2 instâncias.

O fato do AE-C alcançar a melhor solução em dois casos não implica necessariamente no desempenho da geração aleatória de indivíduos e sim na capacidade dos demais módulos do AE de mesmo partindo de uma solução qualquer atingir após 150 iterações uma solução de boa qualidade. Um dos grandes responsáveis por este resultado possivelmente é o módulo de busca local utilizado. Além disso, para 17 instâncias a média da melhor solução do AE foi superior, foi igual em 17 instâncias e apenas em 2 instâncias a média foi ligeiramente menor que as médias do AE-C. Constata-se portanto, que o procedimento de construção HCR tende a influenciar positivamente na maioria dos casos a qualidade da solução final do AE.

Ainda procurando analisar o impacto do procedimento de construção, foram realizados experimentos computacionais adicionais com o AE com o procedimento de construção, mas sem módulo de busca local aqui denotado por AE-BL.

Tabela 4.4 – Resultados do algoritmo evolutivo com e sem o procedimento de construção.

I	AE	AE-C	Tempo AE	Tempo AE-C	Média AE	Média AE-C	Ger. AE	Ger. AE-C
1	**73,68**	**73,68**	0,06	0,06	**73,68**	**73,68**	2	2
2	**62,50**	**62,50**	0,06	0,06	62,17	**62,50**	2	3
3	**79,59**	**79,59**	0,11	0,11	**79,59**	**79,59**	2	2
4	**76,92**	**76,92**	0,09	0,09	**76,92**	**76,92**	2	2
5	**53,13**	**53,13**	0,16	0,15	**53,13**	52,81	2	2
6	**70,37**	**70,37**	0,15	0,15	**70,37**	**70,37**	2	2
7	**68,29**	**68,29**	0,20	0,20	**68,29**	**68,29**	2	2
8	**58,72**	**58,72**	0,29	0,29	**58,72**	**58,72**	2	2
9	**85,25**	**85,25**	0,30	0,29	**85,25**	83,01	2	2
10	**70,59**	**70,59**	0,28	0,29	**70,59**	**70,59**	2	2
11	**92,00**	**92,00**	0,36	0,35	**92,00**	**92,00**	2	2
12	**69,86**	**69,86**	1,17	1,12	**69,74**	69,54	3	3
13	**69,33**	**69,33**	1,25	1,08	**69,33**	69,10	2	2
14	**51,96**	**51,96**	1,68	1,46	**51,96**	51,33	3	3
15	**67,83**	**67,83**	2,09	1,76	**67,83**	**67,83**	2	2
16	**54,86**	**54,86**	3,26	2,49	**54,86**	**54,86**	2	2
17	**75,71**	**75,71**	4,02	3,02	**75,71**	**75,71**	2	2
18	**54,46**	**54,46**	2,24	1,92	**54,12**	53,20	9	7
19	**42,96**	42,86	2,71	1,94	**42,85**	42,78	28	92
20	**49,65**	**49,65**	2,84	2,41	**49,65**	**49,65**	2	2
21	76,14	**76,22**	3,76	3,37	**76,14**	**76,14**	2	62
22	**58,07**	**58,07**	4,15	3,39	**58,07**	**58,07**	2	2
23	**100,00**	**100,00**	5,61	3,95	**100,00**	**100,00**	2	2
24	**85,11**	**85,11**	6,77	5,12	**85,11**	**85,11**	2	2
25	**73,51**	**73,51**	8,76	6,28	**73,51**	70,27	2	2
26	**51,97**	**51,97**	9,62	8,13	**51,89**	51,52	25	27
27	**47,06**	46,84	9,96	7,32	**46,75**	45,98	26	83
28	**44,87**	44,85	9,69	7,08	**44,38**	42,96	139	45
29	**54,27**	**54,27**	6,46	5,01	**54,26**	**54,26**	2	2
30	**44,62**	44,28	14,64	12,00	**44,26**	44,02	110	4
31	**58,14**	57,86	17,23	13,94	**57,73**	57,24	57	63
32	**59,66**	**59,66**	18,69	15,82	**59,50**	57,71	2	54
33	**50,51**	**50,51**	20,63	16,87	**50,33**	46,99	21	11
34	**43,37**	42,16	39,46	28,29	**41,83**	39,83	63	40
35	57,54	**58,89**	20,99	18,74	56,67	**56,99**	10	12
36	**84,03**	**84,03**	79,45	61,29	**84,03**	77,72	2	4

A Tabela 4.5 apresenta os resultados dos experimentos. As colunas representam respectivamente a identificação da instância, a melhor solução obtida pelo AE-BL em 10 execuções para cada instância e a melhoria (valores maiores ou iguais a zero) percentual em relação à melhor solução encontrada pelo AE original (incluindo a busca local).

As colunas com um traço, significam que AE-BL não conseguiu obter soluções válidas (soluções sem clusters vazios, sem linhas ou colunas vazias e sem clusters unitários) para a

referida instância. Pode-se observar pela Tabela 4.5 que a heurística AE sem a busca local não consegue resultados significativos para a grande maioria das instâncias, além de em muitos casos, não conseguir sequer gerar indivíduos (soluções) viáveis.

Isso mostra a importância de uma busca local eficiente nos AEs, comprovando mais uma vez, a razão dos Algoritmos Genéticos na sua forma tradicional não conseguir resultados competitivos quando comparados com as melhores metaheurísticas da literatura.

Tabela 4.5 – Resultados do AE-BL.

Instância	Solução AE-BL	Melhoria
1	73,68	0%
2	62,50	0%
3	75,92	-4,61%
4	76,92	0%
5	46,51	-12,46%
6	65,52	-6,89%
7	65,00	-4,82%
8	56,88	-3,13%
9	49,52	-41,91%
10	70,59	0%
11	70,18	-23,72%
12	-	-
13	-	-
14	26,18	-49,62%
15	27,38	-59,63%
16	-	-
17	-	-
18	20,16	-62,98%
19	26,94	-37,29%
20	21,91	-55,87%
21	-	-
22	19,53	-66,36%
23	-	-
24	-	-
25	-	-
26	-	-
27	-	-
28	-	-
29	29,05	-46,47%
30	15,23	-65,86%
31	-	-
32	-	-
33	-	-
34	-	-
35	41,16	-28,47%
36	-	-

Um AG tradicional seria, por exemplo, o AE aqui proposto sem a heurística construtiva HCR e sem o módulo de busca local que aqui denotamos por AE-{C, BL}.

Para avaliar o desempenho do AE-{C, BL} comparamos com o AG-JCK da literatura.

Tabela 4.6 – Resultados comparativos do AE-{C, BL} e AG-JCK.

Instância	Msol AE-{C, BL}	Tempo(seg) AE-{C, BL}	Msol AG-JCK	Tempo(seg) AG-JCK	Melhoria
1	**73,68**	0,05	**73,68**	0,28	0%
2	**62,50**	0,05	**62,50**	0,28	0%
3	72,73	0,10	**79,59**	0,48	0%
4	**76,92**	0,08	**76,92**	0,43	0%
5	41,86	0,12	**53,13**	0,58	0%
6	56,67	0,12	**70,37**	0,57	0%
7	54,24	0,15	**68,29**	0,75	0%
8	56,88	0,23	**57,26**	1,22	-2,48%
9	51,89	0,22	**76,92**	1,08	0%
10	61,11	0,19	**70,59**	0,96	0%
11	54,12	0,25	**92,00**	1,20	0%
12	36,04	0,61	**59,04**	3,25	-15,48%
13	47,13	0,61	**63,33**	3,13	-8,65%
14	26,51	0,78	**44,85**	4,79	-13,68%
15	39,37	1,00	**64,00**	4,92	-5,64%
16	25,34	1,39	**37,41**	6,31	-17,45%
17	22,52	1,70	**50,75**	9,18	-32,96%
18	-	-	**47,01**	5,55	-13,67%
19	27,52	1,08	**39,23**	6,97	-8,68%
20	26,42	1,20	**45,03**	6,12	-9,30%
21	18,82	1,65	**60,80**	7,83	-20,23%
22	23,16	1,70	**52,83**	8,03	-9,00%
23	13,20	2,98	**58,74**	15,13	-41,26%
24	12,13	2,97	**39,29**	13,43	-53,83%
25	-	-	**46,09**	12,89	-37,30%
26	16,42	2,90	**39,62**	13,32	-23,76%
27	-	-	**31,95**	11,29	-32,10%
28	17,92	3,06	**34,36**	14,83	-23,42%
29	28,54	2,71	**53,75**	14,01	-0,96%
30	15,15	5,29	**20,63**	22,01	-53,76%
31	-	-	**32,21**	28,68	-44,60%
32	-	-	**35,87**	31,06	-39,87%
33	10,67	7,52	**31,80**	26,11	-37,04%
34	-	-	**-**	-	-
35	35,64	14,03	**54,83**	156,94	-4,71%
36	09,91	44,26	**13,41**	113,29	-84,04%

A Tabela 4.6 mostra uma nova bateria de testes executando as 36 instâncias 10 vezes mostrando as melhores soluções obtidas pelo AE-{C, BL} e pelo AG-JCK agora ambos os algoritmos usando o mesmo critério de parada de 150 iterações. O objetivo foi analisar o desempenho do algoritmo AE-{C, BL} com um algoritmo genético da literatura.

As colunas da Tabela 4.6 representam respectivamente: a identificação da instância, melhor solução do AE-{C, BL}, tempo de execução média do AE-{C, BL}, melhor solução do AG-JCK, tempo de execução média do AG-JCK e percentual de melhoria do AE-{C, BL} em relação à melhor solução encontrada pelo AE original.

Claramente observa-se pela Tabela 4.6 que o bom desempenho do AE proposto é fortemente dependente da heurística HCR e do procedimento de busca local.

De fato, em nenhuma instância o AE-{C, BL} conseguiu superar os resultados do AG-JCK (no máximo empatando em alguns poucos casos). Além disso, o número de gerações (150) foi insuficiente para que ambos os algoritmos alcançassem boas soluções.

Os resultados da Tabela 4.6 também apontam uma ineficiência do AE-{C, BL} para gerar soluções válidas utilizando-se apenas do procedimento de cruzamento. Em 6 instâncias o AE-{C, BL} não conseguiu sequer gerar uma solução válida. Já os operadores do AG-JCK não conseguiu gerar indivíduos válidos para a instância 34. Isto sugere que algoritmos evolutivos eficientes para o PFCM necessitam de procedimentos especializados além dos operadores tradicionais de um AG.

5. Análise Probabilística Empírica

Nesta seção o objetivo maior, é analisar o comportamento médio do algoritmo AE aqui proposto em termos da sua convergência empírica para soluções sub-ótimas. De fato, uma das limitações de muitas heurísticas e de algumas metaheurísticas da literatura, é a sua instabilidade.

Ou seja, algoritmos heurísticos que contém componentes probabilísticos (como por exemplo os AGs, GRASP e alguns AEs) podem a cada execução de uma mesma instância gerar soluções totalmente distintas. Ou em outros casos, heurísticas e metaheurísticas com ou sem componentes probabilísticos podem ter um desempenho muito diferenciado com pequenas alterações nos dados de uma instância.

Em síntese, ora podemos ter uma solução de boa qualidade ora de baixa qualidade colocando desta forma estes procedimentos como métodos pouco confiáveis na prática.

Uma forma de avaliar o comportamento de uma heurística é analisar o seu desempenho médio. Este caminho é aqui utilizado para avaliar a robustez e a confiabilidade do AE aqui proposto. Assim, em um outro conjunto de experimentos computacionais, são verificadas as distribuições de probabilidade empírica de alcance de um valor alvo em função do tempo gasto para parte das instâncias descritas anteriormente para o PFCM para o AE e o AG-JCK.

Neste experimento o HGA [RG04] não foi considerado pelo fato desta análise exigir a execução de um número elevado de vezes de cada instância; resultado este que não está disponível em [RG04]. A idéia aqui, é modificar o critério de parada dos algoritmos AE e AG-JCK da seguinte forma: o algoritmo é finalizado quando alcançar um resultado melhor ou igual a um valor alvo pré-definido. Este valor alvo normalmente é definido a partir de testes anteriores efetuados por estes algoritmos nas instâncias analisadas.

Foram determinados neste trabalho, 3 níveis de alvo: alvos *fáceis*, *médios* e *difíceis*. Os alvos fáceis foram determinados como sendo o valor da pior solução final entre as 10 soluções (em 10 execuções) obtidas pelo AE e/ou AG-JCK nos testes anteriores. Os alvos médios foram definidos como valores intermediários entre os valores dos alvos fáceis e difíceis e os alvos difíceis como sendo uma média entre os valores das melhores soluções do AE e AG-JCK nos testes anteriores. Nesta nova bateria de testes, cada um dos dois algoritmos foi executado 100 vezes para cada instância selecionada e armazenadas os tempos para alcance do alvo armazenado para todas as execuções. Os 100 tempos exigidos pelo AE e AG-JCK para cada instância são ordenadas em ordem crescente e plotados associando-se ao i-ésimo menor tempo de execução t_i a probabilidade empírica $p_i = (i - 0{,}5)/100$, gerando pontos z_i no R^2 de coordenadas (t_i, p_i), para *i* de 1 até 100, como proposto por Aiex *et al.* em [Ai02].

Para contornar os casos onde um determinado algoritmo nunca possa alcançar um alvo, foi estabelecido um limite máximo de tempo igual a 5 minutos para cada execução dos algoritmos.

Desta forma, cada ponto de uma curva das Figuras 5.1 a 5.11, indica a probabilidade empírica de alcance do alvo em um determinado tempo de execução. As figuras são apresentadas por instância, na seguinte ordem: primeiramente para alvos fáceis, seguidos de alvos médios e difíceis. Para algumas instâncias, o AG-JCK não conseguiu chegar aos alvos difíceis dentro do tempo limite de 5 para minutos, o que inviabilizou a construção do respectivo gráfico. Diante deste fato, mostramos somente os resultados de algumas das 36 instâncias analisadas, e para alguns casos, somente para alvos fáceis e médios (Instância McCormick).

Nota-se claramente pelas Figuras 5.1 à 5.11, que o AE sempre apresenta uma convergência mais rápida ao valor alvo que o AG-JCK (curva mais a direita). Na Figura 5.1 por exemplo, observamos que o AE necessita de menos de 0,1 segundos para atingir uma probabilidade de convergência de 100% para o valor alvo, ou seja, em 100 execuções do AE, existe uma probabilidade empírica deste atingir o valor alvo em 100% dos casos. Enquanto o AG-JCK necessita de mais de 100 segundos para obter esta taxa de convergência.

Esta situação se repete nos demais casos, onde nos testes realizados, sempre o AE apresenta uma convergência muito mais rápida que o AG-JCK. Além disso, na Figura 5.1 e em muitos casos podemos observar que a curva formada pelo AE é quase vertical, mostrando com isso, que a diferença de tempo exigido nas 100 execuções é muito pequeno, ao contrário do AG-JCK. Isso significa que o AE apresenta um comportamento muito similar ou muito regular em todas as execuções.

Ou seja, o AE sob este aspecto pode ser visto como uma técnica muito mais confiável e robusto que o AG-JCK.

Um outro aspecto interessante que esta análise probabilística nos mostra, é que nem sempre um método com uma iteração mais "*pesada*" necessita de mais tempo para atingir soluções sub-ótimas.

Isso é comprovado pelo AE, que mesmo usando busca local e uma heurística na construção da população inicial acaba exigindo um tempo computacional bem menor que o AG-JCK.

Este fato também nos alerta para a melhor escolha dos critérios de parada numa avaliação de algoritmos heurísticos, já que o critério número máximo de iterações nem sempre avalia corretamente o desempenho de uma heurística no fator tempo computacional exigido.

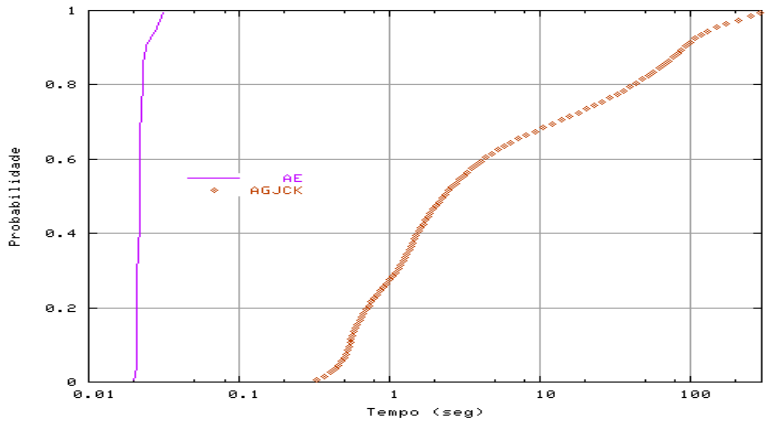

Figura 5.1 – Instância: Askin & Subramanian – Alvo fácil: 66.66.

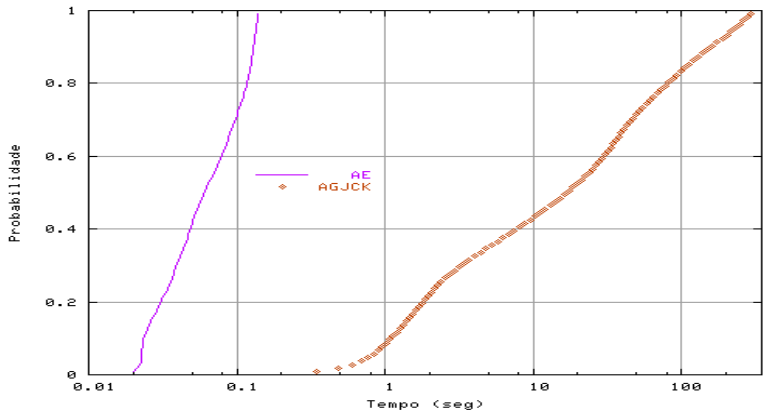

Figura 5.2 – Instância: Askin & Subramanian – Alvo médio: 68.29.

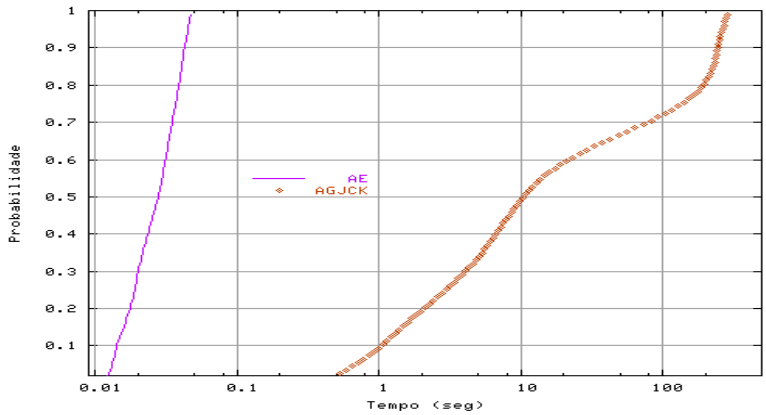

Figura 5.3 – Instância: Askin & Subramanian – Alvo difícil: 69,86.

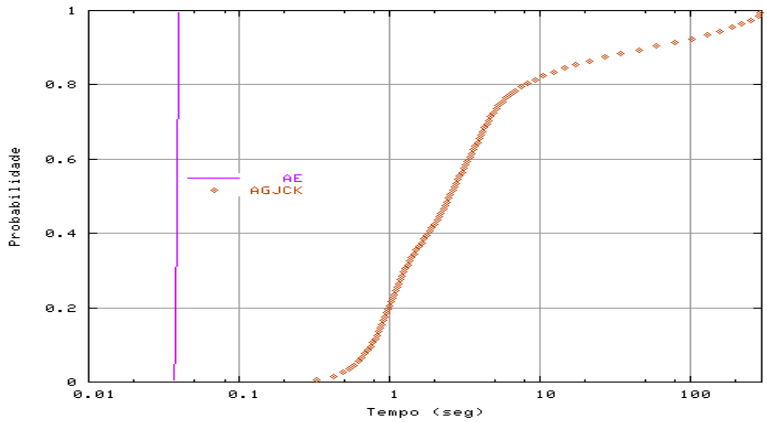

Figura 5.4 – Instância: Srinivasan – Alvo fácil: 56,81.

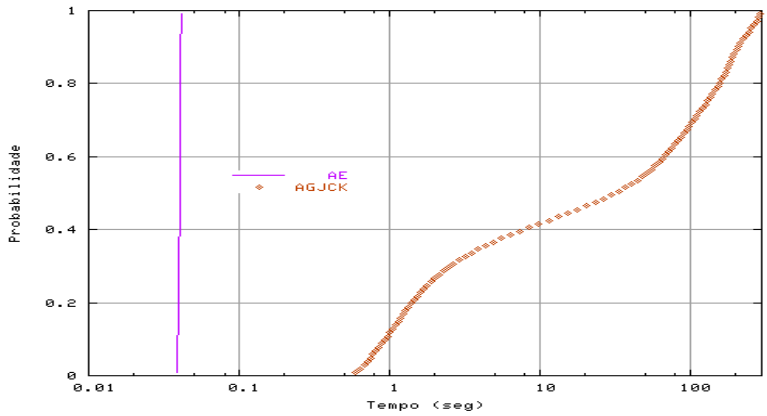

Figura 5.5 – Instância: Srinivasan – Alvo médio: 62,32.

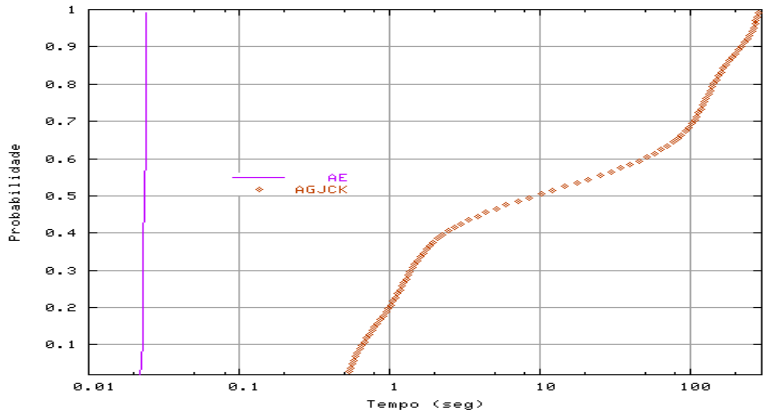

Figura 5.6 – Instância: Srinivasan – Alvo difícil: 67,83.

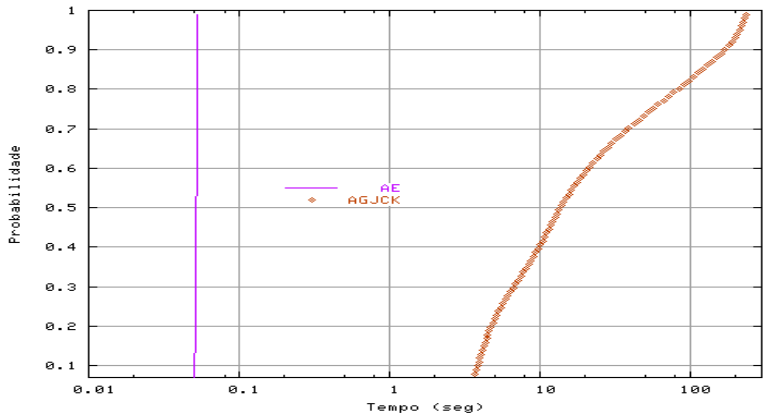

Figura 5.7 – Instância: King – Alvo fácil: 49,00.

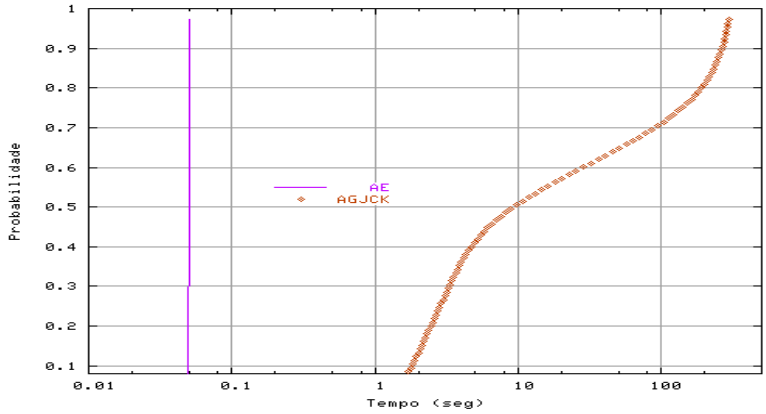

Figura 5.8 – Instância: King – Alvo médio: 51,90.

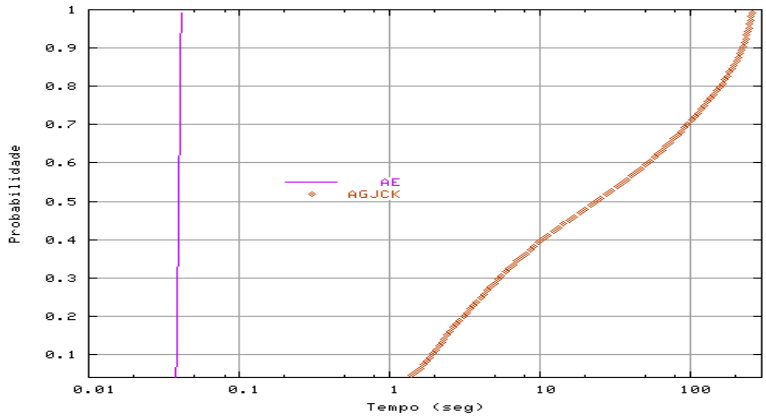

Figura 5.9 – Instância: King – Alvo difícil: 54,86.

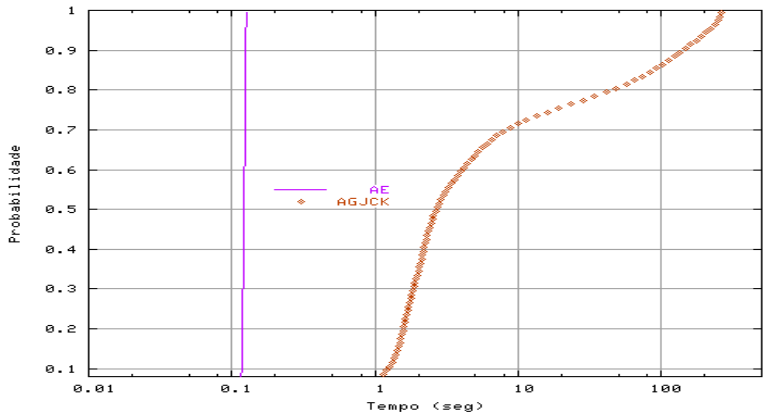

Figura 5.10 – Instância: McCormick 27 x 27 – Alvo fácil: 50,00.

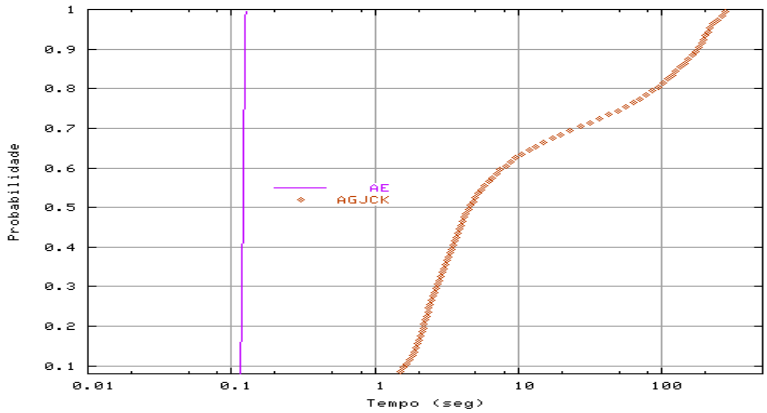

Figura 5.11 – Instância: McCormick 27 x 27 – Alvo médio: 52,13.

6. Conclusões

O objetivo principal deste trabalho foi a elaboração de um algoritmo evolutivo (AE) eficiente para a resolução do Problema de Formação de Células de Manufatura (PFCM). Para tal, são propostos uma heurística de construção randomizada (HCR) de indivíduos, um procedimento de cruzamento genérico e um procedimento eficiente de busca local que são incorporados no AE. Além disso, implementamos um algoritmo genético disponível na literatura (AG-JCK) para fins de comparação.

Nos experimentos computacionais realizados mostrou-se a importância do uso de uma heurística que gere soluções de boa qualidade para inicializar um AG e também a necessidade de incorporar num AG, um mecanismo eficiente de busca local. Para contrapor o trabalho computacional adicional exigido por estes módulos, mostramos que basta reduzirmos o número de iterações de um AE. Ou seja, a introdução de um conjunto já semi otimizado para inicializar um AG e uma busca local eficiente tende a reduzir drasticamente o número de iterações necessárias para um AE atingir soluções sub-ótimas.

Mostrou-se nos experimentos que o AE proposto supera na média os resultados da literatura e numa segunda fase, mostrou-se a robustez do algoritmo aqui proposto. Estes resultados empíricos mostram o potencial da técnica proposta e a possibilidade de ser usada para a solução de outros problemas de otimização.

Referências Bibliográficas

[ABC97] Aljaber, N.; Baek, W. & Chen, C.L. (1997). A Tabu Search approach to the Cell Formation Problem. *Computers Industrial Engineering*, **32**(1), 169-185.

[Ai02] Aiex, R.M.; Resende, M.G.C. & Ribeiro, C.C. (2002). Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, **8**, 343-373.

[BGH99] Boley, D.; Gini, M.; Gross, R.; Han, E.H.; Hastings, K.; Karypis, G.; Kumar, V.; Mobasher, B. & Moore, J. (1999). Partitioning based clustering for Web document categorization. *Decision Support Systems*, **27**, 329-341.

[BOM05a] Bastos, L.; Ochi, L.S. & Macambira, E.M. (2005). A relative neighborhood GRASP for the SONET Ring Assignment Problem. *Proc. of the International Network Optimization Conference – 2005 (INOC 2005)*, Book 3, 833-838. Co-sponsored by INFORMS. Lisbon - Portugal.

[BOM05b] Bastos, L.; Ochi, L.S. & Macambira, E.M. (2005). GRASP with Path Relinking for the SONET Ring Assignment Problem. *Proc. of the 5th International Conference on Hybrid Intelligence Systems (HIS2005)*, 239-244. Co-sponsored by IEEE Systems, Man, and Cybernetics Society.

[BS01] Brown, E.C. & Sumichrast, R.C. (2001). CF-GGA: a grouping genetic algorithm for the cell formation problem. *International Journal of Production Research*, **36(**16), 3651-3669.

[Bu96] Burbidge, J.L. (1996). The first step in planning group technology. *International Journal of Production Economics*, **43**, 261-266.

[CGLW98] Cheng, C.H.; Gupta, Y.P.; Lee, W.H. & Wong, K.F. (1998). A TSP based heuristic for forming machine groups and part families. *International Journal of Production Research*, **36**(5), 1325-1337.

[CR86] Chandrasekharan, M.P. & Rajagopalan, R. (1986). MODROC: an extension of rank order clustering for group technology. *International Journal of Production Research*, **24**(5), 1221-1233.

[CR87] Chandrasekharan, M.P. & Rajagopalan, R. (1987). ZODIAC: an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research*, **25**(6), 835-850.

[Da91] Davis, L. (1991). *Handbook of Genetic Algorithms*. VNR Computer Library.

[DLRG01] Díaz, B.A.; Lozano S.; Racero, J. & Guerrero, F. (2001) Machine cell formation in generalized group technology. *Computers & Industrial Engineering*, **41**, 227-240.

[DM01] Dimopoulos, C. & Mort, N. (2001). A hierarchical clustering methodology based on genetic programming for the solution of simple cell formation problems. *International Journal of Production Research*, **39**(1), 1-19.

[DO03] Dias, C.R.; & Ochi, L.S. (2003). Efficient Evolutionary Algorithms for the Clustering Problems in Directed Graphs. *Proc. of the IEEE Congress on Evolutionary Computation (IEEE-CEC)*, 983-988. Canberra, Australia (CD-ROM).

[ERP02] Ericsson, M.; Resende, M.G.C. & Pardalos, P.M. (2002). A genetic algorithm for the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization*, **6**, 299-333.

[FR95] Feo, T. & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**, 109-133.

[FL00] Filho, G.R. & Lorena, L.A.N. (2000). A Constructive Evolutionary Approach to the Machine-Part Cell Formation Problem. *In Buildings Competencies for International Manufacturing – perspectives for developing countries*, 340-348.

[FSP99] França, P.M.; Sosa, N.M., & Pureza, V. (1999). An adaptive tabu search algorithm for the capacitated clustering problem. *International Transactions in Operational Research*, **6**, 655-678.

[Fog98] Fogel, D.B. [editor] (1998). *Evolutionary Computation*. IEEE Press, Inc.

[GL97] Glover, F. & Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

[Glo86] Glover, F. (1986). Future paths for integer programming a links to artificial intelligence. *Computers & Operational Research*, **5**, 533-549.

[GLO03] Goldschmidt, O.; Laugier, A. & Olinick, E.V. (2003). SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics*, **129**, 99-128.

[GMR04] Gonçalves, J.F.; Mendes, J.J.M. & Resende, M.G.C. (2004). A hybrid genetic algorithm for the Job Shop Scheduling Problems. To appear in *European Journal of Operational Research.*

[GRS99] Gao, L.; Rosenberg, A.L. & Sitaraman, R.K. (1999). Optimal clustering of tree sweep computations for high-latency parallel environments. *IEEE Trans. On Parallel and Distributed Systems*, **10**(8), 813-824.

[GY92] Gerasoulis, A. & Yang, T. (1992). A comparison of Clustering heuristics for scheduling directed acyclic graphs on multiprocessors. *Journal of Parallel and Distributed Computing*, **16**, 276-291.

[Hans86] Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. *Proc. of the Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

[HAR94] Hou, E.S.H.; Ansari, N. & Ren, H. (1994). A genetic algorithm for multiprocessor scheduling. *IEEE Trans. On Parallel and Distributed Systems*, **5**(2), 113-120.

[Holl75] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[HPX90] Harhalakis, G.; Proth, J.M. & Xie, X.L. (1990). Manufacturing Cell Design Using Simulated Annealing: an Industrial Application. Technical Report, University of Maryland.

[JCK96] Joines, J.A.; Culbreth, C.T. & King, R.E. (1996). Manufacturing Cell Design: An Integer Programming Model Employing Genetic Algorithms. *IIE Transaction*, **28**, 69-85.

[JCT01] Joglekar, P.; Chung, Q.B. & Tavana, Madjid (2001). Note on a Comparative Evaluation of Nine Well-Known Algorithms for Solving the Cell Formation Problem in Group Technology. *Journal of Applied Mathematics & Decision Sciences*, **5**(3), 253-268.

[KGV83] Kirkpatrick, S.; Gellat, C.D. & Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, **220,** 671-680.

[KK98] Karypis, G. & Kumar, V. (1998). A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, **48**, 71-95.

[LF01] Lorena, L.A.N. & Furtado, J.C. (2001). Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, **9**(3), 309-327.

[Mac03] Macambira, E.M. (2003). Modelos e algoritmos de Programação Inteira no Projeto de Redes de Telecomunicações. Tese de Doutorado em Engenharia de Sistemas e Computação – COPPE/UFRJ. [Orientadores: Nelson Maculan e Cid C. Souza].

[Mast95] Masters, T. (1995). *Advanced Algorithms for Neural Networks*. John Wiley & Sons.

[MB05] Mester, D. & Braysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, **32**(6), 593-614.

[Mla95] Mladenovic, N. (1995). A Variable neighborhood algorithm – a new metaheuristic for combinatorial optimization. Abstract of papers presented at Optimization Days, Montréal, 112.

[Mi94] Michalewicz, Z. (1994). *Genetic Algorithm + Data Structures = Evolution Programs*. AI Series – Springer Verlag, NY.

[Mos89] Moscato, P. (1989). On evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Caltech Concurrent Computation Program*, C3P Report 8226.

[MS98] Maheshwari, P. & Shen, H. (1998). An efficient clustering algorithm for partitioning parallel programs. *Parallel Computing*, **24**, 893-909.

[MWW00] Mak, K.L.; Wong, Y.S. & Wang, X.X. (2000). An Adaptative Genetic Algorithm for Manufacturing Cell Formation. *The International Journal of Advanced Manufacturing Technology*, **16**, 491-497.

[ODV98] Ochi, L.S.; Drummond, L.M.A. & Vianna, D.S. (1998). A Parallel Genetic Algorithm for the Vehicle Routing Problems. *Future Generation on Computer Systems*. Elsevier, **14**(5-6), 285-292.

[OM01] Onwubolu, G.C. & Mutingi, M. (2001). A genetic algorithm approach to cellular manufacturing systems. *Computers and Industrial Engineering*, **39**(1-2), 125-144.

[OTR04] Oduguwa, V.; Tiwari, A. & Roy, R. (2004). Evolutionary computing in manufacturing industry: an overview of recent applications. To appear in *Applied Soft Computing*.

[OVD01] Ochi, L.S.; Vianna, D.S. & Drummond, L.M.A. (2001). An Asynchronous Parallel metaheuristic for the Period Vehicle Routing Problems. *Future Generation on Computer Systems*. Elsevier, **17**, 379-386.

[PP00] Plaquin, M. & Pierreval, H. (2000). Cell formation using evolutionary algorithm with certain constraints. *International Journal of Production Economics*, **64**(1-3), 267-278.

[RG04] Resende, M.G.C. & Gonçalves, J.F. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, **47**, 247-273.

[RR03] Resende, M.G.C. & Ribeiro, C.C. (2003). Greedy randomized adaptive search procedures. **In**: *Handbook of Metaheuristics* [edited by F. Glover and G. Kochenberger], Kluwer Academic Publishers, 219-249.

[Sa01] Sarker, B.R. (2001). Measures of grouping efficiency in cellular manufacturing systems. *European Journal of Operational Research*, **130**, 588-611.

[SAV98] Selim, H.M.; Askin, R.G. & Vakharia, A.J. (1998). Cell formation in group technology: Evaluation and direction for future research. *Computers and Industrial Engineering*, **34**(1), 3-20.

[SLB95] Sun, D.; Lin L. & Batta, R. (1995). Cell Formation using tabu search. *Computers Industrial Engineering*, **3**(28), 485-494.

[SOMD06] Santos, H.G.; Ochi, L.S.; Marinho, E.H. & Drummond, L.M.A. (2006). Combining an Evolutionary Algorithm with Data Mining to solve a Vehicle Routing Problem. To appear in *NEUROCOMPUTING* – Elsevier.

[US02] Uddin, M.K. & Shanker, K. (2002). Grouping of parts and machines in presence of alternative process routes by genetic algorithms. *International Journal of Production Economics*, **76**(3), 219-228.

[VW95] Vakharia, A.J. & Wemmerlov, U. (1995). A Comparative investigation of hierarchical clustering techniques and dissimilarity measures applied to the cell formation problem. *Journal of Operations Management*, **13**, 117-138.

[XV03] Xambre, A.R. & Vilarinho, P.M. (2003). A Simulated annealing approach for manufacturing cell formation with multiple identical machines. *European Journal of Operational Research*, **151**, 434-446.

[Wa99] Wang, J. (1999). A Linear assignment clustering algorithm based on the least similar cluster representatives. *IEEE Transactions on Systems, Man, and Cybernetics – part A: Systems and Humans*, **1**(29), 100-104.

[WH89] Wemmerlov, U. & Hyer, N.L. (1989). Cellular manufacturing in the US industry: a survey of users. *International Journal of Production Research*, **27**(9), 1511-1530.

[Wy97] Wysk, R.A. (1997). *Chapter 18 Lean – manufacturing*. Manufacturing Systems Course. Department of Industrial and Manufacturing Engineering, Penn State University. Disponível (01/2005) em: <http://www.engr.psu.edu/cim/ie550lean.pdf>.

[ZW00] Zhao, C. & Wu, Z. (2000). A genetic algorithm for manufacturing cell formation with multiple routes and multiple objectives. *International Journal of Production Research*, **38**(2), 385-395.