Universidade Federal Fluminense

Metaheurísticas Aplicadas ao Problema de Formação de Células de Manufatura

Tese submetida para a obtenção do título de

Mestre em Computação

ao Programa de Pós-Graduação

em Computação da UFF

por

Áthila Rocha Trindade

Novembro 2004

Sonho que se sonha só é só um sonho que se sonha só, sonho que se sonha junto é realidade.

(Raul dos Santos Seixas, cantor, poeta e compositor)

Agradecimentos

Agradeço primeiramente a Deus e a meus pais, sem os quais não conseguiria realizar este trabalho. A meu irmão Athos e a minha irmã Júnia pela força e torcida, e ao restante de minha família. Ao meu orientador, o professor Dr. Luiz Satoru Ochi, pela orientação e amizade no decorrer deste trabalho. Aos amigos de Mouseville: Vitor, Sidney, Eduardo, Jonivan, André, Rone, Stênio, Eider e Sanderson pela amizade e companheirismo a mim dedicados; aos amigos do mestrado da UFF, que sempre me ajudaram e estiveram presentes às quintas feiras na "cantareira". Uma menção especial ao colega Jaques, sempre pronto a ajudar os colegas a resolver os "pepinos"no laboratório. Agradeço também a todos os demais professores, funcionários e colegas da UFF, pela atenção a mim oferecida.

Não poderia deixar de mencionar neste momento alguns professores que tive na graduação, os quais considero meus grandes incentivadores: Luiz Siqueira, Adilson Rodrigues e Marcone Jamilson.

A Ouro Preto, palco do começo de minhas pretensões à pós-graduado. A república Jardim Zoológico, onde aprendi a ser perseverante e superar as adversidades. A minha terra natal, a pacata cidadezinha de nome Belo Oriente, encravada nos confins das Minas Gerais; aos meus vários amigos os quais não mencionarei aqui por falta de espaço e ao ensino público, gratuito e de qualidade. Valeu!!!

Resumo da Tese apresentada à UFF como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

Metaheurísticas aplicadas ao Problema de formação de Células de Manufatura

Áthila Rocha Trindade

Novembro/2004

Orientador: Luiz Satoru Ochi
Programa de Pós-Graduação em Computação

Este trabalho tem como objetivo, apresentar um algoritmo evolutivo (AE) e uma metaheurística GRASP para a resolução do Problema de Formação de Células de Manufatura (PCM). O PCM tem se mostrado uma importante estratégia de auxílio em sistemas de produção diversos, contribuindo para um melhor gerenciamento e eficiência dos mesmos.

Os AEs e o GRASP tem resolvido com sucesso uma grande diversidade de problemas combinatoriais da área de otimização. Neste trabalho foram propostos vários procedimentos de construção e busca local de soluções aplicadas ao PCM e testadas várias combinações dos mesmos no AE e na heurística GRASP, a fim de realizar um estudo crítico sobre a importância individual de cada procedimento no desempenho dos algoritmos.

Os algoritmos propostos são comparados com procedimentos existentes na literatura e se mostram muito promissores na solução do PCM.

Abstract of Thesis presented to UFF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Metaheuristics Applied to the Manufacturing Cell Formation Problem

Áthila Rocha Trindade

November/2004

Advisor: Luiz Satoru Ochi
Department: Computer Science

This work aims to present an Evolutive Algorithm (EA) and a Greedy Randomized Adaptive Search Procedure (GRASP) algorithm to solve the Manufacturing Cell Formation Problem (MCP). The MCP has been an important tool for many kinds of production systems, helping managers to obtain a better control and efficiency of them.

The EA and GRASP algorithms have successfully solved several combinatorial problems related to the otimization. In this work we propose some procedures of local search and construction of solutions and many combinations of them were tested in the EA and GRASP algorithms in order to conclude about the relevance of each procedure in the performance of the algorithms.

The proprosed algorithms are compared with some available algorithms in the literature and their results are promissing, indicating that they can be used as a tool to solve the MCP.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O alto grau de complexidade computacional de uma grande variedade de problemas práticos tem levado os pesquisadores a propor métodos alternativos aos métodos exatos para resolvê-los aproximadamente.

Desta forma, esforços tem sido concentrados na busca de métodos que possam alcançar boas soluções (não necessariamente ótimas) num tempo computacional razoável, com mecanismos que permitam escapar de ótimos locais (muitas vezes distantes de um ótimo global). A reunião de conceitos das áreas de Otimização e Inteligência Artificial, viabilizou a construção das chamadas melhores estratégias ou dos métodos inteligentemente flexíveis, comumente conhecidos como metaheurísticos.

As metaheurísticas mais conhecidas na literatura são: Redes Neurais, Algoritmos Evolutivos e o seu representante mais popular, os Algoritmos Genéticos; Busca Tabu, *Greedy Randomized Adaptive Search Procedure* (GRASP), *Simulated Annealing* (SA) e *Variable Neighborhood Search* (VNS). Este trabalho tem sua atenção voltada para dois tipos de metaheurísticas: Os Algoritmos Genéticos (AGs) e a metaheurística GRASP.

Os AGs são metaheurísticas 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 à partir da população anterior.

Inicialmente num AG, existe a necessidade de codificar uma solução viável do problema analisado na estrutura de um indivíduo. Cada indivíduo é avaliado através de uma função de aptidão. Para gerar novos indivíduos, os individuos existentes são combinados de forma que os indivíduos filhos herdem as boas características genéticas dos indivíduos pais. Uma outra forma clássica de gerar novos indivíduos é a partir de processos de mutação, onde genes sorteados aleatoriamente de um determinado indivíduo tem seus valores modificados.

A filosofia de um AG, é iterativamente obter populações de indivíduos cada vez mais aptos justificando desta forma a sua inclusão na classe de algoritmos evolutivos [8].

A metaheurística GRASP é um algoritmo de reinício iterativo, no qual cada iteração consiste de duas fases: construção de uma solução e busca local [36, 38]. Uma característica deste algoritmo é que ao contrário dos AGs, as soluções geradas em cada iteração não tem qualquer relação com soluções de iterações passadas. A fase de construção do GRASP é iterativa, adaptativa, randômica e gulosa. Ela é iterativa pois uma solução é construída elemento a elemento e é adaptativa porque a escolha de um elemento leva em consideração escolhas de elementos selecionados anteriormente. Na seleção de um elemento, inicialmente lista-se todos os candidatos (numa lista denominada LC) e à partir desta LC, constrói-se uma lista de candidatos restrita (LCR) formada por um subconjunto dos melhores candidatos da LC (aspecto guloso). Então escolhe-se aleatoriamente (aspecto randômico) um elemento da LCR para ser incorporado à solução. Esta escolha aleatória permite que diferentes soluções sejam geradas à partir deste algoritmo construtivo.

A fase de busca local constitui-se por uma procura de ótimos locais na vizinhança da solução formada na fase de construção, de acordo com um critério de vizinhança que pode variar de acordo com o problema. Este processo de construção e busca local se repete a cada iteração até que um critério de parada seja alcançado (comumente se usa um número máximo de iterações). A melhor solução é então apresentada

como solução do GRASP.

Coloca-se como objetivo deste trabalho, o desenvolvimento e análise de desempenho de metaheurísticas GRASP e Algoritmos Genéticos híbridos (aqui denominados Algoritmos Evolutivos) aplicadas ao problema de otimização combinatória denominado Problema de Formação de Células de Manufatura (PCM). O restante deste trabalho está dividido da seguinte forma: O capítulo 2 apresenta uma explanação do PCM e uma formulação matemática. O capítulo 3 traz uma resenha da literatura sobre propostas de algoritmos para resolução do PCM. No capítulo 4 é proposto um novo Algoritmo Evolutivo aplicado ao PCM, bem como são discutidos seus aspectos computacionais. O 5 propõe um GRASP para o PCM. No 6 são apresentados e comentados os resultados computacionais obtidos pelos algoritmos acima citados. Finalmente, no capítulo 7 são feitas as conclusões e propostas de trabalhos futuros.

Capítulo 2

O Problema de formação de Células de Manufatura - PCM

2.1 Justificativas

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 de produção* 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 requerido. Para o caso de indústrias que possuem o foco na 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 é a formação de grupos (clusters) de máquinas com funcionalidades idênticas (grupos de tornos, fresadeiras, etc.), formando-se departamentos especializados em

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 favorece 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 [51]. Nas últimas duas décadas um novo modelo de organização destes sistemas de produção denominado *manufatura celular* vem sendo usado.

Em um sistema de produção baseado em *manufatura celular*, máquinas de diferentes funcionalidades são agrupadas em uma célula, a qual é dedicada à produção de uma família de partes composta de diferentes partes que possuem alto nível de similaridade entre si, no que diz respeito às máquinas necessárias para sua manufatura. O sistema de produção fica então dividido em vários grupos ou 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* [50], constituindo-se em 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.

Uma ilustração da formação de células de manufatura pode ser visto a seguir. Considere um sistema de produção formado por 10 máquinas e 15 partes como uma matriz de adjacência binária de máquinas-partes A(i,j), onde cada elemento a_{ij} é igual a 1 se a máquina i executa uma operação de manufatura sobre a parte j e 0 (ou valor em branco) caso contrário (figura 2.1).

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
M1					1				1	1			1		
M2	1			1				1			1		1		
M3					1				1	1			1		
M4	1			1				1			1				
M5		1	1			1	1					1		1	1
M6					1				1	1			1		
M7		1	1	1		1	1					1			
M8					1				1	1			1		
M9					1				1	1		1	1		1
M10		1	1			1	1					1		1	

Figura 2.1: Exemplo de um sistema de produção representado por uma matriz máquina-parte

A formação de células de manufatura se dá através do rearranjo das linhas e colunas da matriz de adjacência de maneira a se formar blocos de células/máquina e partes/família ao longo da diagonal da matriz.

	P5	P9	P10	P13	P1	P10	P4	P8	P11	P2	P3	P6	P7	P12	P14	P15
M1	1	1	1	1												
M3	1	1	1	1												
M6	1	1	1	1												
M8	1	1	1	1												
M9	1	1	1	1										1		1
M2					1	1	1	1	1							
M4					1	1	1	1	1							
M5										1	1	1	1	1	1	1
M7							1			1	1	1	1	1		
M10										1	1	1	1	1	1	

Figura 2.2: Matriz que representa as células de manufatura obtidas para o sistema de produção da figura 2.1

A figura 2.2 apresenta uma possível divisão do sistema de produção da figura 2.1 em 3 clusters de *células/máquinas-famílias/partes*. As células de máquina são

as seguintes: A célula 1 é formada pelas máquinas 1, 3, 6, 8 e 9; a 2 pelas máquinas 2 e 4 e a 3 pelas máquinas 5, 7 e 10. E as famílias são: a família 1 formada pelas partes 5, 9, 10 e 13; a 2 pelas partes 1, 10, 4, 8 e 11 e a 3 pelas partes 2, 3, 6, 7, 12, 14 e 15.

Os clusters de células de máquinas e famílias de partes devem ter um alto grau de independência entre si, caracterizado pela pequena presença de elementos inter-clusters (elementos da matriz iguais a 1 que não pertencem a nenhum cluster) na matriz de blocos diagonais, os quais representam a presença de partes que necessitam de mais de uma célula de máquina para serem completamente manufaturadas e portanto, movimento de material entre diferentes clusters de produção. No exemplo da figura 2.2, podemos identificar 3 elementos inter-clusters. Também é desejável que se tenha poucas lacunas (elementos da matriz iguais a 0) dentro dos clusters, o que representa sub-utilização de máquina. No exemplo da figura 2.2 podemos identificar 3 lacunas no terceiro cluster. A formação de clusters de células de máquinas e famílias de partes resulta 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 mesmo cluster;

2. redução do tempo de produção dos produtos e consequente 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 entre si e de acordo com [5] 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. Em face disto, muitos pesquisadores têm procurado resolver a interdependência entre clusters através da duplicação de máquinas responsáveis pelo aparecimento de elementos excepcionais [26, 31] (no exemplo da figura 2.2, a máquina 9 seria duplicada para o terceiro cluster e a máquina 7 para o segundo) ou a terceirização de partes (no exemplo da figura 2.2 as partes 4, 12 e 15 seriam terceirizadas). Em um caso real, a decisão de duplicação de máquinas e terceirização de partes envolvem 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 contudo não trata da duplicação de máquinas ou terceirização de partes.

2.2 O PCM como um problema de clusterização

De acordo com [49], o número de maneiras de se agrupar n dados em p clusters mutuamente exclusivos e não vazios é determinado pelo número de Stirling de segundo tipo, dado pela equação:

$$\sum_{i=1}^{p}(-1)^{p-i}i^n/[i!(p-i)!] \tag{2.1}$$

Considerando as partes como dados, o número de maneiras de se agrupar n partes em p clusters pode ser dado pela equação 2.1. Como no PCM 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 mutuamente exclusivos e não vazios; ou seja, o número de soluções possíveis do PCM para um número de clusters dado p é dado pela equação 2.2:

$$((\sum_{i=1}^{p}(-1)^{p-i}i^n/[i!(p-i)!]) * (\sum_{i=1}^{p}(-1)^{p-i}i^m/[i!(p-i)!])) \tag{2.2}$$

A tabela 2.1 ilustra a explosão do crescimento do número de soluções em função do número n de partes e m de máquinas, para um número de clusters $p = 3$ clusters.

Nro. de Partes	Nro. de Máquinas	Nro. de soluções
15	10	2,2159683 x 10^{10}
60	40	1,431603835 x 10^{46}
100	70	3,583372838 x 10^{79}

Tabela 2.1: Explosão do número de soluções em função do número de partes para formação de 3 clusters

Podemos concluir observando a tabela 2.1 que o uso de métodos de busca exaustiva torna-se impraticável para instâncias grandes do PCM.

2.3 Formulação Matemática

Qualquer uma das duas etapas do PCM (a alocação de partes para famílias ou máquinas para células) pode ser modelada matematicamente como se segue, de acordo com Wang [49].

Maximizar:

$$\sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j<i} s_{ij}x_{ik}x_{jk} \tag{2.3}$$

Sujeito a:

$$\sum_{k=1}^{p} x_{ik} = 1,\ i = 1,...n \tag{2.4}$$

$$\sum_{i=1}^{n} x_{ik} \leq u,\ k = 1, 2, ..., p \tag{2.5}$$

$$x_{ik} \in \{0, 1\},\ i = 1, 2, ...n,\ k = 1, 2, ...p \tag{2.6}$$

onde:

p = número de clusters a serem formados.

n = número de partes (ou número de máquinas).

u = número máximo de partes (ou máquinas) por cluster.

s_{ij} = índice de similaridade entre a parte (ou máquina) i e a parte (ou máquina) j.

x_{ik} = variável binária que assume o valor 1 se a parte (ou máquina) i está associada ao cluster k, 0 caso contrário.

x_{jk} = variável binária que possui assume o valor 1 se a parte (ou máquina) j está associada ao cluster k, 0 caso contrário.

As restrições (2.4) determinam que uma parte (ou máquina) pode estar associada à somente um cluster. As restrições (2.5) determinam que nenhum cluster pode conter mais partes (ou máquinas) do que o número máximo de partes (ou máquinas) determinado.

Salientamos que a formulação acima não trata da clusterização de máquinas e partes simultaneamente. Desta forma, tal formulação pode ser usada ou para a clusterização das máquinas ou para clusterização das partes.

2.4 Definição do número de clusters a serem formados

Um parâmetro do PCM que influi diretamente no resultado da solução final é o número de clusters a serem formados. Quanto maior este número, maior o espaço de busca a ser explorado, o que pode aumentar o tempo de execução de um algoritmo na busca por uma boa solução. Então é necessário que seja determinado um número razoável de clusters. Por outro lado, é muito difícil para o usuário através de uma

inspeção visual da matriz inicial conhecer ou estimar o número ideal de clusters a serem formados.

Como a resolução do PCM 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. clusters unitários seriam justificáveis em dois casos: **1)** Caso exista uma máquina apenas que atenda a um determinado conjunto 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 pois não possui partes atendidas em comum com nenhuma máquina sequer. **2)** Caso exista uma parte apenas que necessite de um determinado conjunto de máquinas; mas neste caso também não faz sentido esta parte e estas máquinas pertencerem à matriz de entrada do PCM, já que a parte não pode ser agrupada com nenhuma outra pois não existe nenhuma máquina sequer que atenda a esta parte e a qualquer outra.

Desta forma, são consideradas na literatura como soluções inválidas aquelas que contem clusters com uma máquina apenas ou uma parte apenas [22, 37]. Dado que o número de máquinas é sempre menor que o número de partes, um limite superior para o número de clusters usado neste trabalho foi $\lceil \frac{m}{2} \rceil$. Este valor não evita que sejam formadas soluções com clusters unitários, mas restringe o espaço de busca, pois soluções para o PCM com um número de clusters maior que este limite dado possuem obrigatoriamente clusters unitários. O valor limite mínimo de clusters é igual a 2. Antes da geração de cada solução pelas heurísticas aqui propostas, é sorteado um número dentro desta faixa de limites para ser o número de clusters para a referida solução. As heurísticas só consideram como soluções válidas aquelas sem clusters unitários, sem clusters com linhas ou colunas vazias (uma linha vazia ocorre quando uma máquina pertencente a um determinado cluster não realiza operação de manufatura sobre qualquer parte pertencente ao cluster em questão e uma coluna vazia ocorre quando uma parte pertencente a um determinado cluster não necessita de nenhuma máquina pertencente ao cluster em questão) ou clusters inteiramente vazios (clusters vazios ocorrem quando todas as partes pertencentes a

um determinado cluster não necessitam de nenhuma máquina pertencente ao cluster em questão).

Uma vez feita a explanação do PCM, no próximo capítulo é feita uma revisão da literatura existente sobre as diversas abordagens utilizadas para resolução do PCM bem como as diversas medidas de desempenho e similaridade usadas por tais abordagens.

Capítulo 3

Metodologias existentes para o PCM

Com o objetivo de se tornarem mais competitivas no mercado, muitas empresas tem adotado técnicas baseadas na Tecnologia de Grupos para a gestão de seus respectivos sistemas de produção. A Tecnologia de Grupos busca decompor os sistemas de produção em vários subsistemas menores, mais eficientes, mais fáceis de gerenciar e de preferência independentes entre si.

Um importante etapa do uso da Tecnologia de Grupos na gestão da produção é a formação de um sistema de manufatura celular, onde são formadas famílias de partes (peças de produtos) e células de máquinas, sendo que a cada célula de máquinas é associada uma família de partes. Num sistema de manufatura celular ideal, existe uma total independência entre os pares de partes-família e máquinas-célula. Isto quer dizer que todas as partes de uma família devem ser completamente manufaturadas pelas máquinas pertencentes à célula associada à esta família e; todas as máquinas de uma célula qualquer devem realizar operações de manufatura somente sobre as partes pertencentes à família associada à esta célula.

Tendo em vista a importância da formação de células de manufatura na indústria, várias abordagens têm sido propostas para agrupar máquinas e partes. Existe também na literatura, uma variedade de medidas desempenho e noções de similaridade que são usadas respectivamente para a avaliação das soluções obtidas e para a formação dos grupos num PCM.

Veremos a seguir uma classificação metodológica dos diversos tipos de algoritmos, medidas de desempenho e medidas de similaridade aplicados ao PCM.

3.1 Algoritmos baseados em representação matricial de agrupamento

Esta é uma das mais simples técnicas de representar o PCM. Os algoritmos que se utilizam desta representação tem como entrada uma matriz de incidência máquina-parte (ou parte-máquina), onde cada elemento a_{ij} é igual a 1 se a máquina i realiza uma operação de manufatura sobre a parte j e 0 caso contrário (ou cada elemento a_{ij} é igual a 1 se a parte i visita uma máquina j e 0 caso contrário) figura 3.1(a). Os algoritmos então buscam a formação de células de máquinas e famílias de partes através da permutação das linhas e colunas desta matriz, de maneira a formar blocos de 1's agrupados ao longo da diagonal da matriz, como mostrado abaixo na figura 3.1(b).

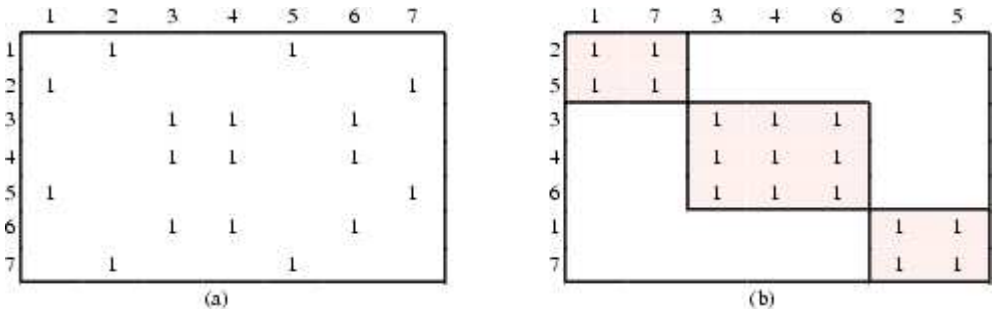


Figura 3.1: (a) Matriz parte/máquina e (b) Agrupamento ideal

Descrevemos nas seções seguintes uma síntese dos principais métodos existentes na literatura para a solução do PCM.

3.1.1 Algoritmo de Energia de Vinculação (*Bond Energy Algorithm* - BEA)

Este algoritmo, de acordo com Lorena et al. [11] busca a organização da matriz de incidência 0-1 do PCM num formato de blocos diagonais através da análise do *vínculo* de energia de cada elemento a_{ij} da matriz e seus quatro elementos vizinhos a_{ij-1}, a_{ij+1}, $a_{i-1,j}$ e $a_{i+1,j}$. Para cada permutação das m linhas e n colunas da matriz, é associado um valor total de vinculação de energia, que expressa o vínculo total de energia, para todos os elementos da matriz. O objetivo do algoritmo é examinar todas as permutações possíveis de linhas e colunas e encontrar a permutação que maximiza a função de vinculação de energia (*Total Bond Energy*) TBE(m,n), dada por:

$$TBE(m,n) = 1/2 \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}[a_{ij-1} + a_{ij+1} + a_{i-1j} + a_{i+1j}] \tag{3.1}$$

onde:

$a_{0,j}$=$a_{m+1,j}$=$a_{i,0}$=$a_{i,n+1}$ = 0 $\forall$ j=1,...,n, $\forall$ i=1,...,m

m = o número de máquinas

n = o número de partes

Pode-se concluir pela formulação acima que quanto mais próximos estiverem os elementos iguais a 1, maior será o valor da função de vinculação de energia TBE(m,n). Porém, conforme observou McCormick et al. segundo Lorena et al. [11], os agrupamentos produzidos pelo BEA parecem às vezes mais com um tabuleiro de xadrez do que com uma matriz de blocos diagonais. É necessária também uma análise visual da matriz por parte do projetista do sistema de manufatura para definir as alocações máquinas-células e partes-famílias. Além disso, o número soluções formadas pela permutação de linhas e colunas para uma matriz de m linhas e n colunas é $m!n!$, de forma que a aplicação do algoritmo pode se tornar inviável dependendo do tamanho da matriz de entrada do PCM.

3.1.2 *Rank Order Clustering* - ROC

O algoritmo ROC foi proposto por King em 1979 segundo Wysk [50] e utiliza uma representação binária das colunas e linhas da matriz de entrada. ROC associa um valor inteiro à essa representação para realizar uma ordenação descendente das linhas e colunas de acordo com tais valores inteiros, a fim de encontrar os blocos diagonais que representam os clusters (grupos) de células-máquinas e partes-famílias. Cada linha i e coluna j tem seus valores determinados através da conversão de sua representação binária em um valor decimal, sendo que os bits mais e menos significativos para as linhas e colunas são respectivamente os valores da primeira e última colunas e primeira e última linhas.

Baseando-se nestes valores associados às linhas e colunas da matriz de incidência, o algoritmo ROC pode ser descrito pelo seguinte pseudo-código:

1: **Procedimento** Algoritmo ROC
2: Calcule o valor numérico associado à cada coluna j da matriz
3: Se as colunas estão em ordem descendente, vá para a linha 4, senão ordene-as
4: Calcule o valor numérico associado à cada linha i da matriz
5: Se as linhas estão em ordem descendente, finalize o algoritmo. Senão ordene as linhas e volte à linha 2.
6: **Fim** ROC

Figura 3.2: Pseudocódigo do algoritmo ROC

Vejamos um exemplo da aplicação do algoritmo ROC. Considere a matriz da figura 3.3(a) como entrada para o algoritmo ROC. Os elementos da matriz em branco possuem o valor igual a "0". As sequências binárias para as colunas são as seguintes:

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

(a)

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

(b)

Figura 3.3: (a) Matriz de entrada para o algoritmo ROC e (b) Matriz após o ordenamento das colunas

P1 = 0 1 1 0 (pois a parte P1 necessita das máquinas M2 e M3 e não necessita da máquina M1 e M4). Analogamente, as demais sequências binárias para as colunas são: P2 = 0 1 1 0, P3 = 1 0 0 1, P4 = 1 0 0 0, P5 = 1 0 0 1 e P6 = 0 1 1 0; e os valores decimais das sequências binárias das colunas são: P1 = 6, P2 = 6, P3 = 9, P4 = 8, P5 = 9 e P6 = 6. Através da ordenação das colunas pelo algoritmo ROC (linha 3 do pseudocódigo da figura 3.2) obtém-se a matriz da figura 3.3(b). Observando a matriz da figura 3.3(b), temos as seguintes sequências binárias para as linhas: M1 = 1 1 1 0 0 0, M2 = 0 0 0 1 1 1, M3 = 0 0 0 1 1 1 e M4 = 1 1 0 0 0 0; e os valores decimais das mesmas são: 56, 7, 7 e 48. Ordenando descendentemente as linhas (linha 5 do pseudocódigo da figura 3.2), obtemos a seguinte matriz de clusterização:

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

Figura 3.4: Matriz obtida pelo algoritmo ROC após primeira ordenação de linhas e colunas

A segunda execução dos dois primeiros passos de ROC não alteram a matriz da figura 3.4 pois as colunas já estão ordenadas. A segunda execução dos passos 3 e 4 de ROC também não alteram a matriz da figura 3.4 pois as linhas também já estão ordenadas. Desta forma, o algoritmo termina e a matriz da figura 3.4 é a matriz solução. A definição dos clusters fica a cargo do projetista das células de manufatura. Neste exemplo, podemos perceber claramente dois clusters de *células/famílias* : o cluster 1, constituído das máquinas M1 e M4 e das partes P3, P5 e P4; e o cluster 2, constituído pelas máquinas M2 e M3 e das partes P1, P2 e P6. Entretanto salientamos que para matrizes grandes e *mal comportadas* (com um grande número de elementos inter-clusters) essa identificação visual dos clusters pode não ser tão fácil.

Para tratar o aparecimento de muitos elementos inter-clusters, para execução

do ROC primeiramente são identificados tais elementos através de procedimentos iterativos propostos por King, Chan e Milner e Chu e Tsai; de acordo com Lorena et al. [11]. Então ROC é executado sob a matriz resultante. Depois de sua execução, as máquinas associadas à estes elementos inter-clusters são duplicadas e associadas às células de máquinas para as quais existem os elementos inter-clusters e o algoritmo ROC novamente executado.

Chandrasekaran e Rajagopalan relataram em [6] importantes falhas no funcionamento de ROC: matrizes de entrada inicialmente *bem estruturadas* (com poucos elementos inter-clusters) podem vir a estar desorganizadas ao final do algoritmo; existe a tendência de aglomeração de elementos da matriz iguais a 1 no canto noroeste da matriz final permanecendo o restante da matriz desorganizada; a disposição inicial das linhas e colunas da matriz de entrada influi diretamente na forma da matriz de blocos diagonais final e como observado por [11, 24, 50] a representação binária para linhas e colunas restringe o tamanho da matriz de entrada ao tamanho da palavra interna do computador.

Para superar tais falhas, em [6] é proposto uma extensão do algoritmo ROC denominado MODROC (*Modificed Rank Order Clustering*). MODROC se inicia com a execução do algoritmo ROC sobre a matriz de entrada. A tendência de formação de um bloco de 1's no canto noroeste da matriz final por ROC, permite à MODROC identificar um bloco de elementos iguais a 1 no canto noroeste da matriz. Então as colunas deste bloco são retiradas da matriz e as partes associadas à estas colunas são identificadas como uma família. Então o algoritmo ROC é aplicado novamente sobre a matriz restante. Esta identificação e retirada de blocos no canto noroeste se repete até que todas as colunas sejam eliminadas da matriz. Este estágio pode resultar em máquinas associadas à mais de uma célula (células de máquinas não disjuntas) e um grande número de clusters com poucas máquinas e partes [11, 24, 6].

Para produzir uma formação de *células/famílias* adequada, o segundo estágio de MODROC se constitui da combinação de clusters através de um procedimento de clusterização hierárquico, baseado num coeficiente de similaridade entre máquinas. Porém, este é dependente da informação por parte do projetista do número máximo

e mínimo de máquinas por célula, um número máximo de células, o coeficiente de similaridade a ser usado e um valor limite do coeficiente entre máquinas para o qual a clusterização é finalizada.

Chandrasekaran e Rajagopalan constataram também em [6] que, ao contrário do que disse o autor do algoritmo ROC, tal algoritmo nem sempre é capaz de obter uma clusterização bem definida, mesmo que exista um rearranjo de linhas e colunas da matriz de incidência que a obtenha. Além disso, MODROC não possui as demais deficiências identificadas em ROC.

3.1.3 Algoritmo baseados em técnicas de clusterização direta (DCA)

Os algoritmos baseados em clusterização direta (*Direct Clustering Algorithm* - DCA) executam permutações das linhas e colunas da matriz do PCM, da seguinte maneira: as linhas contendo elementos iguais a 1 mais à esquerda são movimentadas para o topo da matriz e as colunas contendo elementos iguais a 1 mais ao topo para a esquerda da matriz. Estes algoritmos convergem relativamente em poucas iterações [11] mas como observou Wemmerlov segundo Lorena et al. [11], tendem a formar um bloco no canto noroeste da matriz, deixando o restante da mesma desorganizado.

3.2 Algoritmos baseados em clusterização hierárquica

Esta técnica realiza a formação de grupos de células/famílias (clusterização) com base em dados de entrada sob a forma de um coeficiente de similaridade ou função de distancia, e determina uma hierarquia de grupos ou partições, sendo que em cada nível de hierarquia pode haver um número diferente de grupos com diferentes números de membros. Contrariamente aos métodos baseados em matriz, a formação de células de máquina e famílias de partes não é feita simultaneamente. A formação

de grupos pode ser feita de maneira divisiva ou aglomerativa. Os algoritmos divisivos iniciam com todos os dados (no caso do PCM, máquinas e partes) pertencendo a um mesmo grupo e cria através de partições uma série de grupos até cada dado pertencer a um determinado grupo, enquanto que os algoritmos aglomerativos consideram inicialmente cada dado como um grupo e realiza a junção dos grupos até que um critério de parada esteja satisfeito.

Como estes algoritmos se baseiam em coeficientes e estes últimos podem incorporar informações outras senão as informações binárias da matriz de entrada do PCM, pode-se obter algoritmos para resolução do PCM que considerem outros aspectos importantes de um sistema de manufatura, como por exemplo: sequência de operações de manufatura das partes nas máquinas, volume médio de produção para cada parte, tempo de processamento para cada operação de manufatura das partes, volume de tráfego de material entre máquinas, etc. Em razão disto, uma série de medidas de similaridade tem surgido na literatura para aplicação no PCM. Um interessante estudo sobre medidas de similaridade para resolução do PCM foi feito em [48]. É imprescindível portanto uma escolha sensata do coeficiente de similaridade a ser usado de acordo com os aspectos do PCM a serem considerados num algoritmo de clusterização hierárquica.

Como exemplos de algoritmos de clusterização hierárquica aplicados ao PCM, citamos a segunda etapa do algoritmo MODROC [6] e o algoritmo divisivo proposto por Harhalakis et al. [18], sendo que este último leva em consideração para a clusterização a ordem das operações de manufatura para as partes.

3.3 Algoritmos baseados em clusterização não hierárquica

Técnicas de clusterização não-hierárquica obtém os clusters através do uso de uma função de similaridade ou distância entre os dados a serem clusterizados, mas sem a necessidade de uma computação prévia das similaridades ou das distâncias

entre todos os dados como na clusterização hierárquica. Entretanto é preciso que se informe a priori um número k de clusters a serem formados, o que em nossa opinião pode ser difícil de se determinar para instâncias grandes e *mal estruturadas* (matrizes com grande número de elementos inter-clusters) do PCM.

Basicamente, são escolhidos inicialmente os k dados que serão as sementes dos k clusters. Em seguida, os dados restantes são agrupados ao cluster mais próximo, com base numa função de similaridade ou distância entre os dados e os clusters. Um exemplo de algoritmo baseado em clusterização não hierárquica para o PCM será visto a seguir.

3.3.1 Algoritmo ZODIAC

O algoritmo ZODIAC (*zero-one data: ideal seed algorithm for clustering*), desenvolvido por Chandrasekaran e Rajagopalan [7] é um algoritmo de clusterização não-hierárquica onde a escolha da sementes dos clusters pode ser feita de forma arbitrária, artificial, representativa ou natural no primeiro estágio do algoritmo. Ao final do algoritmo é usado outro tipo de semente chamada semente ideal, para término da clusterização. ZODIAC considera as m linhas das máquinas e n colunas das partes independentemente como vetores em um espaço n-dimensional e m-dimensional, respectivamente; representando um conjunto de pontos. Com base nas escolhas das sementes e nas distâncias entre os demais pontos e as sementes, se dá a formação de células de máquinas e famílias de partes. As várias formas de escolha de sementes foram combinadas em sequências, de maneira que a clusterização se dá através de clusterizações sucessivas a cada nova forma de escolha de sementes. Com isso, os autores puderam relatar importantes conclusões sobre o uso de diferentes tipos de semente.

Uma escolha arbitrária de sementes iniciais pode levar o algoritmo à formação de um número pequeno de clusters, que podem não refletir uma clusterização satisfatória.

Uma clusterização baseada na escolha de sementes artificiais (um ponto no es-

paço m ou n-dimensional que não represente nenhum dos vetores de linhas ou colunas da matriz do PCM) seguida de uma clusterização baseada na geração de sementes representativas (pontos no espaço m ou n-dimensional que representam vetores de linhas ou colunas da matriz do PCM) e por fim sementes ideais (pontos no espaço m ou n-dimensional que representam vetores de linhas ou colunas da matriz clusterizada, excluídos os elementos inter-clusters e as lacunas no clusters) podem produzir bons resultados [7]. O uso de sementes ideais se faz necessário para escapar de ótimos locais ainda distantes de um ótimo global e melhorar a solução final, tendo em vista que os resultados alcançados pelo uso somente de sementes artificiais e representativas não foram satisfatórios, segundo os autores. Um limite superior para o número de clusters baseado em teoria dos grafos é usado como limiar para o número de clusters a serem formados.

Outra abordagem usada por ZODIAC é a clusterização através da geração de sementes naturais (pontos no espaço m ou n-dimensional que representam vetores de linhas ou colunas da matriz do PCM, cuja distância atende a um limiar determinado pelo projetista) e de sementes ideais. Chandrasekaran e Rajagopalan observaram ainda em [7] que a clusterização inicial feita usando as sementes naturais alcança bons resultados, sendo a fase de clusterização por sementes ideais uma tentativa de uma pequena melhora na solução. Além disso, uma vantagem da utilização de sementes naturais é que o número de clusters a serem formados surge como resultado da escolha destas sementes.

O uso de um coeficiente de similaridade ao invés de uma medida de distância no processo de escolha de sementes também é sugerido pelos autores.

3.4 Algoritmos baseados em teoria dos grafos

A teoria dos grafos também tem sido explorada para propor algoritmos de resolução do PCM. Geralmente os sistemas de manufatura são consideradas como se segue: os nós representam as máquinas (partes) e as arestas a relação (similaridade, fluxo de operações, etc.) entre elas, como visto a seguir.

3.4.1 Abordagem de Fluxo de Redes

Foulds and Neumann [13], apresentam um modelo de fluxo de redes para a resolução do PCM, baseando-se nas taxas de utilização das partes em relação às capacidades das máquinas. Uma vez constituída a rede, onde os nós da mesma são constituídos por partes, máquinas e células de manufatura a serem formadas, a resolução do PCM é feita através da minimização de uma função de fluxo desta rede que representa o custo de movimento entre células.

3.4.2 Abordagem de K-coloração de Grafos

Em [39] é apresentado um algoritmo que utiliza a K-coloração de grafos para resolver o PCM. Inicialmente, é calculado um coeficiente de dissimilaridade entre todas as partes, com base na matriz 0-1 de entrada. Após isto, um grafo G é construído onde os nós representam as partes e existirá uma aresta a_{ij} entre duas partes i e j se o coeficiente de dissimilaridade entre elas for maior ou igual a um limiar definido a priori. O passo seguinte se constitui na coloração do grafo em k cores empregando-se uma técnica baseado no algoritmo de Guénoche, segundo [39], onde k é o número de clusters a serem formados. Dado que dois nós adjacentes não recebem a mesma cor, a coloração de G não colocará em uma mesma família partes que possuam um coeficiente de dissimilaridade maior ou igual ao limite estabelecido. Uma vez determinada as famílias de partes, obtém-se as células de máquinas atribuindo-se cada máquina para a família com o maior número de partes que a máquina em questão atende.

3.5 Algoritmos baseados em técnicas de Inteligência Artificial

Uma grande variedade de algoritmos baseados em técnicas de Inteligência Artificial (IA) tem sido propostos para resolução do PCM. Geralmente estas técnicas com-

binam alguns dos métodos mostrados anteriormente (clusterização não-hierárquica, teoria dos grafos, etc.) com estratégias inerentes à IA. Em [10] encontra-se uma razoável referência à computação evolucionária aplicada à formação de células de manufatura. O grande número de trabalhos encontrados na literatura sugere estes métodos como promissores para a aplicação ao PCM.

3.5.1 Redes Neurais Artificiais

Em [3], Nguema et al. apresentam um modelo de redes neurais baseado no modelo de Hopfield para resolução do PCM. A principal vantagem desta abordagem é sua habilidade para resolver instâncias grandes do PCM. Shankar et al. [43] apresentam um modelo de rede neural transitoriamente caótica para o PCM. Os resultados computacionais desta abordagem são comparados com outras 4 abordagens de redes neurais, tendo superado estas últimas para 18 instâncias da literatura no que diz respeito à qualidade das soluções e ao tempo computacional exigido.

3.5.2 Lógica *Fuzzy*

O principal objetivo da resolução do PCM é agrupar máquinas nas células mais adequadas e partes nas famílias mais adequadas. Mas dependendo da matriz de entrada, nem sempre é muito claro qual a célula ou família mais adequada para a máquina ou parte. O algoritmo proposto em [35] utiliza um grau de relacionamento entre máquinas e partes representado por um coeficiente que varia entre 0 e 1, para realizar esta associação de máquinas à células e partes à famílias. A entrada do algoritmo é uma *matriz máquina-parte de relação fuzzy*, onde cada elemento a_{ij} representa o grau de relacionamento da máquina i para a parte j. Esta matriz é então convertida em uma matriz binária máquina-parte, sendo que é preciso que se informe o número de clusters a serem formados para tal conversão. Em seguida, as partes e máquinas da matriz binária que possuem respectivamente linhas e colunas iguais são unidas e consideradas como uma única máquina e parte. A etapa final se constitui do cálculo de um coeficiente de similaridade proposto entre máquinas e partes e no

agrupamento de máquinas e partes com base num modelo de programação linear. Smed et al. [44] utilizaram lógica *fuzzy* para a elaboração de um *software* para o PCM aplicado à indústria de placas de circuitos eletrônicos.

3.5.3 Algoritmos Genéticos

Os algoritmos genéticos (AGs) são uma metaheurística que tem sido usada para a resolução de uma grande variedade de problemas de otimização, e tem se mostrado uma ferramenta flexível e eficiente para tal.

Joines et al. [22] desenvolveram um AG para solucionar o PCM como problema de programação inteira, permitindo funções objetivo com múltiplos critérios e restrições. O algoritmo possui 4 operadores de mutação e 3 operadores de crossover. Uma extensão híbrida deste algoritmo foi proposta em [23], na qual foi inserido um procedimento de busca local. Ambos algoritmos necessitam da informação de um limite superior do número de clusters a serem formados. Baseado em [27], Lorena et al. [12] desenvolveram um AG construtivo que se baseia na modelagem do PCM como o problema das p-medianas, onde os clusters são formados com base na minimização da distância das partes e máquinas às partes e máquinas-medianas. Este algoritmo também necessita que seja informado apriori um número k de clusters a serem formados. Um aspecto interessante desta abordagem é que ela aceita que soluções parciais façam parte de uma população de soluções, tendo em vista que podem vir a se tornar boas soluções em gerações futuras. Um AG caracterizado pela dispensa do uso do operador de crossover é proposto em [46], igualando os resultados da literatura para 17 instâncias. Mak et al., [30] propuseram um AG adaptativo onde as taxas de utilização dos operadores de mutação e crossover mudam de acordo com a eficiência dos mesmos ao longo da execução do algoritmo. Resende et al. propuseram em 2002 [37], um AG híbrido que incorpora ao algoritmo genético básico um procedimento de busca local, tendo este algoritmo igualado ou melhorado os resultados da literatura para 35 instâncias do PCM.

3.5.4 Busca Tabu

A metaheurística Busca Tabu é uma técnica de otimização baseada na noção de vizinhança, originada dos trabalhos de Glover [15, 17, 16]. Sendo S o espaço de busca de um problema de otimização, f a função objetivo a minimizar ou maximizar e s uma solução viável do problema, denota-se *vizinhança de s* por $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de *vizinho de s* e é obtida por uma modificação na estrutura de s denominada *movimento*, modificação esta que depende do problema em questão.

A Busca Tabu se inicia com uma solução semente s (a qual pode ser obtida por alguma outra técnica ou gerada aleatoriamente) e investiga num subconjunto da vizinhança de s a melhor solução existente. Esta melhor solução vizinha de s é aceita como nova semente, mesmo que esta não seja melhor que a semente corrente s. Isto permite ao método escapar de ótimos locais ainda distantes do ótimo global. O processo de busca é então reiniciado, considerando-se a nova solução semente.

Para evitar que a Busca Tabu retorne à soluções já geradas anteriormente, existe uma **Lista Tabu**, que é uma lista contendo os movimentos proibidos. Geralmente a Lista Tabu contém os movimentos reversos aos últimos **T** movimentos realizados (onde T é um parâmetro do método), movimentos estes que poderiam fazer com que à partir da solução semente se chegasse a soluções já geradas.

Cada iteração do método é constituído da busca numa vizinhança de uma semente s e escolha de uma nova semente. As iterações se repetem até que um critério de parada seja alcançado (geralmente um número máximo de iterações sem melhoria da melhor solução) e a melhor solução encontrada é apresentada como solução para o problema em questão.

Uma metaheurística Busca Tabu para o PCM foi proposta por [2], sendo que ela se constitui basicamente de três fases: a primeira fase constitui-se na determinação do melhor sequenciamento de máquinas e partes, de maneira a se formar dois caminhos mínimos (um caminho para partes e outro para máquinas), onde as distancias entre máquinas e entre partes estão diretamente relacionadas com o grau de simi-

laridade entre elas. Estes dois caminhos são determinados através da aplicação do algoritmo de busca tabu propriamente dito. Numa segunda fase, é montada a matriz de clusterização referente à melhor solução encontrada na primeira fase. A terceira fase se constitui da determinação do número de clusters através da identificação das fronteiras de cada cluster e da informação por parte do projetista de um limite superior do número de clusters a serem formados. Sun et al. [47] apresentaram uma busca tabu que modela o PCM como um grafo onde os nós são máquinas e as arestas entre os nós representam os fluxos entre máquinas, obtidos através da sequência das operações para cada parte. O objetivo é particionar o grafo em subgrafos, tal que o custo das arestas entre os subgrafos seja minimizado, o que representa a minimização do volume do tráfego entre clusters. Deve ser informado pelo projetista um limite superior do número de clusters a serem formados. Em [9] é proposta uma busca tabu para resolver o PCM diante da possibilidade de rotas de produção alternativas para as partes. O objetivo final é minimizar o custo do tráfico entre clusters. Um aspecto interessante deste algoritmo é o uso de vários níveis de diversificação da busca. Longendran et al. [26] propõem uma busca tabu para o PCM com a possibilidade de duplicação de máquinas e terceirização de partes. Além disso, o algoritmo também procura pelo melhor leiaute das células de manufatura dentro do ambiente de produção. Em seu estudo, ele aplica o algoritmo a exemplos reais e obtém resultados satisfatórios.

3.5.5 *Simulated Annealing*

Simulated Annealing é uma metaheurística de busca local probabilística, proposta originalmente por Kirkpatrick et al. [25].

Inicialmente é gerada uma solução inicial s, chamada solução semente. Cada iteração do método constitui-se da geração de um vizinho s' de s e na escolha ou não deste vizinho como nova semente. Caso o vizinho não seja escolhido, uma nova iteração é iniciada com a antiga semente.

É justamente na escolha da nova semente que está a característica probabilística

do método: determinada a solução *s'* , esta passará a ser a nova semente caso o valor da função objetivo associado à ela seja melhor (maior em casos de maximização e menor em casos de minimização) que o da solução semente *s*. Caso contrário, *s'* também poderá ser aceita como nova semente, mas com uma probabilidade de $e^{\frac{-\Delta}{T}}$, onde Δ é a diferença entre os valores da função objetivo de *s'* e *s*, e T um parâmetro do método denominado *temperatura*. Inicialmente T assume um valor alto que é gradativamente diminuído iteração a iteração. Desta forma, nas iterações iniciais a probabilidade de escolha de soluções piores como semente é maior (mecanismo para fuga de ótimos locais) e diminui à medida que avançam as iterações (o método adquire caráter guloso).

O algoritmo é finalizado quando a temperatura atinge valores próximos a zero e soluções piores não são mais aceitas como semente.

Harhalakis et al. [19] apresentaram um *Simulated Annealing* cujo objetivo é minimizar o custo relativo ao tráfego entre clusters, com base na sequência de operações para cada parte, o custo de cada operação e o volume de produção de cada parte para um determinado horizonte de tempo. Este algoritmo foi executado para uma instância real e alcançou resultados satisfatórios. O *Simulated Annealing* proposto em [51] procura minimizar o volume do tráfego entre clusters com base no fluxo entre as operações das partes, a associação de operações à máquinas e a capacidade de produção de cada máquina. Uma questão interessante levada em conta é a existência de máquinas funcionalmente idênticas, permitindo uma maior flexibilidade de associação de operações à máquinas.

3.6 Outras abordagens

Além dos métodos mostrados até agora, existem muitos outros destinados à resolução do PCM. Ribeiro and Meguelati em [40] desenvolveram um algoritmo baseado em classificação cruzada de partes e máquinas que visa minimizar o número de movimento entre clusters e maximizar a taxa de utilização das máquinas. Baker et al. em [4] propuseram um algoritmo baseado no algoritmo ROC [50] como parte de

uma ferramenta computacional para gestão da produção. Ohta et al. [34], desenvolveram um algoritmo composto de um estágio de clusterização não hierárquica para agrupamento de máquinas e um estágio guloso para agrupamento das partes, a fim de resolver o PCM com o objetivo de reduzir o tempo de preparação das máquinas. Podemos citar ainda abordagens baseadas em *branch-and-bound* [14, 20], múltiplos critérios [31] e heurísticas [29, 28, 33, 49].

3.7 Medidas de desempenho

Juntamente com o grande número de trabalhos propostos para resolução do PCM, há um grande número de medidas de desempenho para avaliar as soluções obtidas pelos algoritmos. Tais medidas podem incorporar muitos aspectos do PCM, como tempo das operações de manufatura sobre as partes, capacidade de produção das máquinas, etc. Daremos atenção às medidas de desempenho destinadas à avaliação de soluções obtidas com base apenas nas informações da matriz 0-1 de entrada do PCM, tendo em vista que nosso trabalho visa a resolução do PCM considerando apenas as informações de tal matriz. No nosso entendimento, uma medida de desempenho eficaz para avaliação de soluções do PCM baseadas na matriz de incidência parte-máquina dele levar em consideração os seguintes fatores:

1. O número total de operações na matriz de entrada do PCM.
2. O número de operações executadas dentro dos clusters.
3. O número de lacunas nos clusters, que devem ser minimizados, pois representam sub-utilização de máquinas pelo cluster.
4. O número de elementos inter-clusters, que representam tráfego de material entre os clusters. Boas soluções geralmente possuem poucos elementos inter-clusters

3.7.1 Eficiência de Agrupamento

Proposta por Chandrasekharan e Rajagopalan em 1986 de acordo com Sarker e Mukattash et al. [42, 32], a *eficiência de agrupamento* (*Grouping Efficiency*) η é definida como:

$$\eta = q\eta_1 + (1 - q)\eta_2 \tag{3.2}$$

onde:

η_1 = razão entre o número de operações (1's) nos blocos diagonais (clusters) e o número total de elementos nos blocos diagonais.

η_2 = razão entre o número de lacunas (0's) fora dos blocos diagonais e o número total de elementos (0's ou 1's) fora dos blocos diagonais.

q = fator de peso ($0 \leq$ q ≤ 1)

$0 \leq \eta \leq 1$

Quando não houver um elemento sequer igual a 1 dentro dos blocos diagonais e todos os elementos iguais a 0 estiverem dentro dos blocos diagonais η_1 e η_2 serão iguais a 0, de acordo com suas definições. Desta forma, η será igual a 0 independente do valor de q. Por outro lado, quando todos elementos iguais a 1 estiverem dentro dos blocos diagonais e todos os elementos iguais a 0 fora dos blocos diagonais, então η_1 e η_2 serão iguais a 1, de acordo com suas definições. Desta forma, η será igual a 1 independente do valor de q.

Exemplos de uso desta medida podem ser vistos em [7, 35]. Falhas apontadas por [42, 32] são: a forte dependência da medida ao parâmetro q e a variação do valor da medida entre 0,75 e 1 para os testes de instâncias da literatura. Desta forma, mesmo soluções ruins com grande número de elementos inter-clusters apresentam uma eficiência de agrupamento em torno de 0,75.

3.7.2 Eficácia de Agrupamento

Proposto por Kumar e Chandrasekharan segundo [32, 4], a *eficácia de agrupamento* (*Grouping Efficacy*), tem a seguinte forma:

$$\tau = \frac{e - e_0}{e + e_v} \tag{3.3}$$

onde:

e = número de operações (1's) da matriz

e_0 = número de elementos inter-clusters

e_v = número de elementos iguais a 0 dentro dos clusters

$0 \leq \tau \leq 1$

Quando não houver elementos inter-clusters e não houver lacunas nos clusters, os valores de e_0 e e_v serão iguais a 0 e consequentemente o valor de τ igual a 1. Caso todos os elementos iguais a 1 estejam foram dos clusters, o numerador da equação 3.3 será igual a 0 e portanto, o valor de τ igual a 0.

A eficácia de agrupamento vem sendo usada largamente na literatura. Os algoritmos propostos em [22, 23, 37, 12] utilizam a eficácia de agrupamento. A eficácia de agrupamento não está sujeita às falhas mencionadas para a eficiência de agrupamento mas conforme observou Sarker [42], podem ser formadas soluções cujos clusters tenham colunas ou linhas vazias, o que não representam soluções viáveis. A eficácia de agrupamento não é portanto sensível à este tipo de falha nas soluções. Partindo desta conclusão, o uso desta medida se torna condicionado à não aceitação de tais soluções por parte do método de resolução do PCM usado. Além disso ainda segundo Sarker a eficácia de agrupamento é mais sensível à mudança no número de lacunas nos clusters que no número de elementos inter-clusters e não possui um fator de peso que permita ao projetista do sistema de manufatura dar mais importância ao número de elementos inter-clusters ou ao número de lacunas nos clusters, como na eficiência de agrupamento.

3.7.3 Índice de Capacidade de Agrupamento - GCI

O Índice de Capacidade de Agrupamento (GCI - *Grouping Capability Index*) proposto por Hsu segundo Mukattash et al. [32] tem a seguinte definição:

$$GCI = 1 - (\frac{e_0}{e}) \tag{3.4}$$

onde:

e_0 = número de elementos inter-clusters

e = número de operações da matriz

$0 \leq \text{GCI} \leq 1$

O GCI tem seu valor igual a 0 quando todos os elementos iguais a 1 (operações da matriz) estão fora dos clusters. Desta forma os valores de e_0 e e serão iguais. O valor máximo de GCI (igual a 1) acontece quando não existem elementos inter-clusters, ou seja, $e_0 = 0$.

O GCI não leva em considerações o número de lacunas nos clusters, o que pode fazer com que soluções contendo clusters grandes e com número considerável de lacunas (clusters esparsos) sejam consideradas boas soluções.

3.7.4 Eficiência Global - GLE

Proposta por Harhalakis de acordo com [32], a medida de desempenho denominada *Eficiência Global* (GLE - *Global Efficiency*) é dada por:

$$GLE = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=1}^{n}} \tag{3.5}$$

onde:

n = número de operações da matriz

$S_i = 1$ se a i-ésima operação é executada dentro de um cluster, 0 caso contrário

$0 \leq \text{GLE} \leq 1$

Desta forma, a medida GLE expressa a razão do número de operações executadas dentro dos clusters para o número total de operações do sistema de manufatura. GLE será igual a 0 quando o somatório dos S_i for igual a 0, o que ocorre quando não há uma operação sequer dentro dos clusters. GLE será igual a 1 quando o somatório dos S_i for igual ao somatório dos r_i, o que significa que todas as operações da matriz estarão dentro dos clusters. Como no caso da medida de desempenho anterior, GLE não leva em consideração o número de lacunas nos clusters.

3.7.5 Medida Primária

Segundo [21, 42], Miltenburg and Zhang propuseram a medida de desempenho denominada *Medida Primária* (M1), dada por:

$$\eta_g = \frac{e_1}{e_1 + e_v} - \frac{e_0}{e} \tag{3.6}$$

onde:

e_1 = número de operações executadas dentro dos clusters

e_v = número de lacunas nos clusters

e_0 = número de operações executadas fora dos clusters

e = número de operações da matriz

-1 $\leq \eta_g \leq$ 1

Quando não houver operações executadas dentro dos clusters, o valor de e_1 será igual a 0 e o número de operações executadas fora dos clusters será igual ao número de operações da matriz (e_0=e). Desta forma, o valor de η_g será igual a -1. Por outro lado, quando todas as operações da matriz forem executadas dentro dos clusters e não houver lacunas nos mesmos (e_0 e e_v iguais a 0), o valor de η_g será igual a 1.

Tal medida é utilizada por Mak et al. em [30]. A medida M1 significa a diferença da razão entre o número de operações dentro dos clusters e o número total de

elementos nos clusters e a razão entre os elementos inter-clusters e o número total de operações na matriz. Joglekar et al. [21] fez uma comparação experimental entre M1 e a *eficiência de agrupamento* e mostrou que M1 supera a falha da *eficiência de agrupamento* no que diz respeito à variação do valor da medida de desempenho entre 0,75 e 1.

3.7.6 Medida de Eficiência de Agrupamento duplamente ponderada

A *Medida de Eficiência de Agrupamento duplamente ponderada* (*Doubly Weighted Grouping efficiency measure*) foi proposta por Sarker [42] e é dada por:

$$\eta_Q = (\frac{q_1 e_1 + (1 - q_1)e_v}{e_1 + e_v})(\frac{q_2 e_1 + (1 - q_2)e_0}{e_1 + e_0}) \tag{3.7}$$

onde:

q_1 e q_2 são fatores de peso, tais que $0 \leq (q_1,q_2) \leq 1$

e_1 = número de operações dentro dos clusters

e_v = número de lacunas nos clusters

e_0 = número de elementos inter-clusters

$0 \leq \eta_Q \leq 1$

Quando todas as operações da matriz estiverem dentro dos clusters e não houver lacunas nos mesmos os valores de e_v e e_0 serão iguais a 0. Portanto, η_Q será igual a $(\frac{q_1 e_1}{e_1})(\frac{q_2 e_1}{e_1})$. Se os pesos q_1 e q_2 forem iguais e com valor igual a 1, então η_Q será igual a 1. O mesmo acontecerá se o número de operações dentro dos clusters for igual a 0 (e_1=0) e os valores dos pesos for igual a 0. Entretanto neste caso η_Q será igual a 1 para uma solução não viável, tendo em vista que todos clusters estarão vazios. η_Q será igual a 0 quando o número de operações dentro dos clusters for igual a 0 e os valores dos pesos iguais a 1; ou quando todas as operações da matriz estiverem dentro dos clusters, não houver lacunas nos mesmos e os valores dos pesos iguais a 0. Entretanto soluções com todas as operações dentro dos clusters e ausência de

lacunas nos mesmos são soluções ótimas. Portanto valores extremos para os pesos q_1 e q_2 devm ser evitados.

A justificativa de Sarker [42] para os dois fatores de peso (q_1 e q_2) é que q_1 obtém a eficiência intra-cluster ponderada (primeiro termo da equação) e q_2 obtém a eficiência relativa ponderada fora dos clusters. Desta forma, o projetista do sistema de manufatura celular pode atribuir pesos diferentes a estes dois fatores individualmente. Sarker também demonstrou que a medida é mais sensível ao número de elementos inter-clusters (que devem sempre ser evitados) do que às lacunas nos clusters. Porém, a medida é fortemente dependente dos fatores de peso, sendo que uma escolha precipitada dos valores de q_1 e q_2 pode tornar a medida ineficaz.

3.7.7 Índice de Célula - CI

Proposta por Mukattash et al. [32], o *Índice de Célula* (CI - *Cell Index*) é dado por:

$$CI = \frac{1}{n} \sum_{j=1}^{P} \frac{k_j n_j}{k_j + v_j + e_j} \tag{3.8}$$

onde:

n = número total de máquinas

n_j = número total de máquinas no j-ésimo cluster

v_j = número de lacunas no j-ésimo cluster

e_j = número de elementos inter-clusters no j-ésimo cluster (elementos da matriz iguais a 1 situados nas linhas do j-ésimo cluster, mas que não pertencem à ele)

k_j = número de operações no j-ésimo cluster

p = número de clusters

$0 \leq \text{CI} \leq 1$

CI será igual a 0 quando o valor de k_j for igual a 0 para todos os valores de j, ou seja, todos os clusters. Isto caracteriza a ausência de operações dentro dos clusters. Quando não houver elementos inter-clusters nem lacunas nos clusters, os valores de v_j e e_j serão iguais a 0, para todo j. Desta forma, a equação 3.8 será:

$\frac{1}{n}\sum_{j=1}^{P}\frac{k_j n_j}{k_j}$, ou seja: $\frac{1}{n}\sum_{j=1}^{P} n_j$. Consequentemente, o valor de CI será igual a 1. O aspecto interessante da medida CI é que ela permite a avaliação do "peso" de cada cluster no resultado final da clusterização, o que possibilita ao projetista do sistema de manufatura a identificação de clusters com "má formação".

3.8 Medidas de Similaridade

Muitas medidas (ou índices) de similaridade baseados nos dados da matriz binária do PCM tem sido propostos na literatura. Tais medidas tem relação direta com a matriz de clusterização final do PCM, tendo em vista que estas podem ser usadas para agrupar máquinas e partes. Daremos neste seção uma explanação daquelas que consideramos as mais relevantes.

3.8.1 Similaridade de Jaccard - JS

A Similaridade de Jaccard (JS - *Jaccard Similarity*) é dada por:

$$JS_{ij} = \frac{(Y_{ij})}{(Y_i + Y_j - Y_{ij})} \tag{3.9}$$

onde:
Y_{ij} = número de máquinas (partes) usadas (atendidas) em comum pelas partes (máquinas) i e j
Y_i = número de máquinas (partes) usadas (atendidas) pela parte (máquina) i
Y_j = número de máquinas (partes) usadas (atendidas) pela parte (máquina) j
$0 \leq JS_{ij} \leq 1$

O valor de JS_{ij} será máximo quando Y_{ij} for igual a $Y_i + Y_j$, ou seja, quando as máquinas (partes) usadas (atendidas) pela parte (máquina) i são exatamente as mesmas usadas (atendidas) pela parte (máquina) j. JS_{ij} será igual a 0 quando Y_{ij} for igual a 0, ou seja, não houver uma máquina (parte) sequer que é usada (atendida) por ambas as partes (máquinas). Um problema nesta avaliação apontado

por Vakharia e Wemmerlov em [48] é que ela pode minimizar a relação entre duas partes (máquinas), caso $Y_i \gg Y_j$ (Y_i seja muito maior que Y_j). Por exemplo, se $Y_i = 10$, $Y_j = 100$, $Y_{ij} = 10$, $Y_p = 10$ e $Y_{ip} = 5$, então $JS_{ij} = 0.1$ e $JS_{ip} = 0.33$. Então, de acordo com os cálculos, as partes i e p são mais similares do que as partes i e j. Entretanto, a parte j utiliza 100% das máquinas de i enquanto p utiliza apenas 50%.

Para contornar este problema, Vakharia e Wemmerlov propuseram a seguinte modificação nesta medida de similaridade:

$$S_{ij}(\alpha) = \alpha[Y_{ij}/Y_i] + (1-\alpha)[Y_{ij}/Y_j] \tag{3.10}$$

onde:
$Y_i \leq Y_j$ e $\alpha \in [0,1]$
$0 \leq S_{ij}(\alpha) \leq 1$

A ocorrência dos valores de mínimo e máximo (respectivamente, 0 e 1) de $S_{ij}(\alpha)$ se dão pelos mesmos motivos pelos quais os valores mínimo e máximo de JS_{ij} ocorrem, independente do valor de α. Observando a equação 3.10 e a condição de desigualdade entre Y_i e Y_j, constata-se que o primeiro termo da equação de S_{ij} leva em consideração a relação de proporcionalidade entre o número de máquinas (partes) usadas (atendidas) em comum pelas partes (máquinas) i e j e a parte (máquina) i ou j que usa (atende) o menor número de máquinas (partes). O segundo termo da equação leva em consideração a relação de proporcionalidade entre o número de máquinas (partes) usadas (atendidas) em comum pelas partes (máquinas) i e j e a parte (máquina) i ou j que usa (atende) o maior número de máquinas (partes). Desta forma, o projetista do sistema de manufatura tem liberdade para dar mais enfoque para o primeiro ou segundo termo de acordo com o valor do parâmetro α escolhido (um valor de α próximo de 1 prioriza o primeiro termo da equação 3.10 e um valor de α próximo de 0 prioriza o segundo termo da mesma).

Vakharia e Wemmerlov em [48] transformaram $S_{ij}(\alpha)$ em uma medida de dissi-

milaridade dada por:

$$DS_{ij}(\alpha) = 1 - S_{ij}(\alpha) \tag{3.11}$$

Utilizando variadas técnicas de clusterização e medidas de dissimilaridades, concluiram experimentalmente que a medida $DS(\alpha)$ obteve melhor eficiência entre as medidas de dissimilaridade usadas. Entretanto, o valor mais adequado para α varia de acordo com a matriz de entrada do PCM, a técnica de clusterização e a medida de desempenho. Além disso, concluiram que os valores extremos de $\alpha = 0$ e $\alpha = 1$ devem ser evitados.

3.8.2 Similaridade Euclidiana - EUC

A Similaridade Euclidiana (EUC) entre duas partes (máquinas) é definida como:

$$EUC_{ij} = \sqrt{\sum_{k=1}^{M} x_{ijk}} \tag{3.12}$$

onde:
M = número de máquinas (partes) $x_{ijk} = 1$ caso as partes (máquinas) i e j usem (atendam) a máquina (parte) k ou caso ambas as partes (máquinas) i e j não usem (atendam) a máquina (parte) k, 0 caso contrário

Uma extensão da Similaridade Euclidiana é a Similaridade Euclidiana Quadrada (SEUC), dada por:

$$SEUC_{ij} = (EUC_{ij})^2 \tag{3.13}$$

Vakharia e Wemmerlov concluiram em [48] que SEUC e EUC são inadequadas para o PCM, pois eventualmente estas "distâncias" entre duas partes ou máquinas podem não contribuir para uma coerente clusterização. Por exemplo, considere uma instância do PCM com 3 partes P1, P2 e P3, e 6 máquinas, rotuladas de M1 a M6. As máquinas necessárias para a manufatura de cada uma das partes são as seguinte: P1: M2 e M6; P2: M1, M3, M4 e M6; e P3: M5. Podemos então determinar os

vetores binários que representam as rotas de produção para cada uma das partes. São eles: P1 = [0 1 0 0 0 1], P2 = [1 0 1 1 0 1] e P3 = [0 0 0 0 1 0]. Os valores de EUC_{ij} e $SEUC_{ij}$ para os pares de partes são: $EUC_{12} = \sqrt{2}$ e $SEUC_{12} = 2$, $EUC_{13} = \sqrt{3}$ e $SEUC_{13} = 3$ e $EUC_{23} = 1$ e $SEUC_{23} = 1$. De acordo com $SEUC_{ij}$ e EUC_{ij} as partes mais "próximas" ou similares são as partes 2 e 3, mas entretanto tais partes não utilizam nenhuma máquina em comum.

3.8.3 Similaridade baseada em comparação de vetores

Ravichandran et al. [35] propuseram uma nova medida de similaridade entre duas partes (máquinas) i e j, com base numa comparação dos componentes dos vetores que representam as rotas de produção (utilização) das mesmas. Sejam A_i = [1 0 1 1 0 0 1 0] e A_j = [0 1 1 0 0 1 0 0] os vetores binários das rotas de produção de duas partes i e j, num sistema de manufatura composto por 8 máquinas. S_{ij} é dado por:

$$S_{ij} = \frac{a(a+d)}{(a\sqrt{(a+d)} + b + c)\sqrt{(a+d)}} \tag{3.14}$$

onde:

a = número de máquinas que efetuam operações nas partes i e j

b = número de máquinas que efetuam operações na parte i e não efetuam operações na parte j

c = número de máquinas que não efetuam operações na parte i, mas que efetuam operações na parte j

d = número de máquinas que não efetuam operações nem na parte i nem na parte j

$0 \leq S_{ij} \leq 1$

S_{ij} será igual a 0 se o parâmetro a for igual a 0, ou seja, se não existirem máquinas que efetuem operações em ambas as partes i e j. S_{ij} será igual a 1 se o parâmetro a for diferente de 0 os demais parâmetros iguais a 0.

No exemplo mostrado acima, os valores de a, b, c e d são 1, 3, 2 e 2 respectivamente e S_{ij} = 0,375.

3.9 Análise de uma proposta da literatura para o PCM usando AG

Nesta seção mostraremos com mais detalhes o algoritmo genético proposto por Joines et. al em 1996 [22] e implementado por nós, o qual foi usado para efeito de comparação com os algoritmos propostos.

A escolha deste AG para ser implementado neste trabalho se deve ao fato de que este produziu, segundo seus autores, resultados promissores e também porque na nossa opinião, o algoritmo apresenta algumas propostas interessantes que foram aproveitadas em nosso algoritmo evolutivo. Além disso, ao implementarmos este AG, o nosso objetivo foi de verificar o desempenho deste em novas instâncias do PCM da literatura.

3.9.1 Estrutura do indivíduo

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 C = (x_1,...x_m | y_1,...y_n). Cada gene x_i ou y_i assume valores inteiros, sendo feita a alocação de máquinas à células e partes à famílias simultaneamente. Para uma instância do PCM com 8 máquinas, 10 partes e 4 clusters a serem formados, podemos ter o seguinte indivíduo: C1 = (2 4 1 3 2 3 4 1 | 2 4 2 4 1 1 2 2 3 3). Cada gene x_i indica para qual célula cada máquina é alocada (x_1...x_m). Por exemplo, em C1 o gene ($x_1 = 2$) significa que a máquina m_1 é alocada para a célula 2. Cada gene y_i indica para qual família cada parte é alocada (y_1...y_n). Por exemplo, em C1, o gene $m_{+1} = y_1 = 2$ significa que a parte p_1 é alocada para a família 2.

3.9.2 Operadores crossover e mutação

Foram usados neste trabalho 4 tipos de operador mutação e 3 tipos de operador crossover. São eles:

1. **Mutação uniforme:** O operador seleciona um gene v_i do individuo e muda o seu valor para um número situado dentro de uma faixa definida por dois valores (mínimo a_i e máximo b_i).

2. **Mutação de fronteira:** O operador seleciona um gene v_i do indivíduo e muda seu valor para um valor igual ao mínimo de clusters (*célula/família*) ou o máximo de clusters a serem formados.

3. **Mutação não-uniforme:** O operador seleciona um gene v_i do indivíduo pai e incrementa ou decrementa seu valor de acordo com a função:

$$v_i = \begin{cases} v_i + (b_i - v_i)f(g), & se\ r_1 < 0,5; \\ v_i - (v_i - a_i)f(g), & caso\ contrário. \end{cases} \tag{3.15}$$

   onde:

   r_1 = um número gerado aleatoriamente entre 0 e 1

   g = geração corrente

   f(g) = r_2(1-g/g_{max})^2

   g_{max} = número máximo de gerações

   r_2 = um número gerado aleatoriamente entre 0 e 1

4. **Multi-mutação não-uniforme:** Idêntico ao operador anterior, só que todos os genes do indivíduo são selecionados e seus valores alterados.

5. **Crossover inteiro de troca:** Neste operador o ponto de corte é localizado na separação dos genes referentes às máquinas e partes. Por exemplo, sejam os indivíduos-pai P1 e P2, cujos genes de máquina são aqui representados por x_1,...,x_m e w_1,...,w_m respectivamente; e os genes de parte por y_1,...,y_p e z_1,...,z_p, respectivamente. Os indivíduos-filho F1 e F2 terão a seguinte configuração:

$$\text{P1} = (x_1,x_2,...,x_m \mid y_1,y_2,...,y_p)$$

$$\text{P2} = (w_1,w_2,...,w_m \mid z_1,z_2,...,z_p)$$

$$\text{F1} = (x_1,x_2,...,x_m \mid z_1,z_2,...,z_p)$$

$$F2 = (w_1,w_2,...,w_m \mid y_1,y_2,...,y_p)$$

6. **Crossover de dois pontos:** Este operador seleciona aleatoriamente um ponto de corte na faixa de genes de máquina e um ponto de corte na faixa de genes de partes; para cada indivíduo-pai. Por exemplo, sejam os indivíduos-pai P1 e P2 (considere o ponto de corte representado por um sinal de ";"). Os indivíduos filhos F1 e F2 terão a seguinte configuração:

$$P1 = (x_1,x_2 \ ; \ x_3,x_4,x_5 \mid y_1,y_2,y_3 \ ; \ y_4,y_5)$$

$$P2 = (w_1,w_2 \ ; \ w_3,w_4,w_5 \mid z_1,z_2,z_3 \ ; \ z_4,z_5)$$

$$F1 = (x_1,x_2,w_3,w_4,w_5 \mid y_1,y_2,y_3,z_4,z_5)$$

$$F2 = (w_1,w_2,x_3,x_4,x_5 \mid z_1,z_2,z_3,y_4,y_5)$$

7. **Crossover Aritmético:** Este operador produz uma combinação linear dos pais para gerar os indivíduos-filho. Sejam os cromossomos P1 e P2. Os cromossomos filhos F1 e F2 terão a seguinte configuração (considere os valores dos genes dos filhos representados entre os sinais de "<>"):

$$P1 = (x_1 \ x_2 \ x_3 \ x_4 \mid y_1 \ y_2 \ y_3 \ y_4 \ y_5 \ y_6)$$

$$P2 = (w_1 \ w_2 \ w_3 \ w_4 \mid z_1 \ z_2 \ z_3 \ z_4 \ z_5 \ z_6)$$

$F1 = (<x_1\text{a}+w_1\text{b}>,<x_2\text{a}+w_2\text{b}>,<x_3\text{a}+w_3\text{b}>,<x_4\text{a}+w_4\text{b}> \mid$
$<y_1\text{a}+z_1\text{b}>,<y_2\text{a}+z_2\text{b}>,<y_3\text{a}+z_3\text{b}>,<y_4\text{a}+z_4\text{b}>,<y_5\text{a}+z_5\text{b}>,<y_6\text{a}+z_6\text{b}>)$

$F2 = (<x_1\text{b}+w_1\text{a}>,<x_2\text{b}+w_2\text{a}>,<x_3\text{b}+w_3\text{a}>,<x_4\text{b}+w_4\text{a}> \mid$
$<y_1\text{b}+z_1\text{a}>,<y_2\text{b}+z_2\text{a}>,<y_3\text{b}+z_3\text{a}>,<y_4\text{b}+z_4\text{a}>,<y_5\text{b}+z_5\text{a}>,<y_6\text{b}+z_6\text{a}>)$

onde:

$0 < \text{a} < 1$ e

$\text{b} = 1 - \text{a}$

Como podem surgir valores reais para os genes de máquinas e partes de acordo com a combinação linear usada, a conversão de tais valores reais dos genes para valores inteiros se dá usando-se a seguinte função:

$$< v_1 a + v_2 b >= \begin{cases} \lceil < v_1 a + v_2 b > \rceil, & se\ v_1 \ > \ v_2; \\ \lfloor < v_1 a + v_2 b > \rfloor, & caso\ contrário. \end{cases} \tag{3.16}$$

onde:

v_1 = valores de gene multiplicados pelo fator a. No caso do filho F1 são os genes do pai P1 e no caso do filho F2 os genes do pai P2.

v_2 = valores de gene multiplicados pelo fator b. No caso do filho F1 são os genes do pai P2 e no caso do filho F2 os genes do pai P1.

3.9.3 Função de Aptidão

A função de aptidão nos algoritmos genéticos representa o mecanismo de obtenção de uma avaliação de cada indivíduo e indica se este é ou não uma boa solução para o problema. Assim, soluções mais aptas são aquelas que possuem melhor (maior em caso de problemas de maximização e menor em casos de problemas de minimização) aptidão. A função de aptidão usada no Algoritmo proposto em [22] é a medida de desempenho *Eficácia de Agrupamento*, mostrada no item 2.7.2 do capítulo 2 deste trabalho e tem a seguinte forma:

$$\tau = \frac{e - e_0}{e + e_v} \tag{3.17}$$

onde:

e = número de operações da matriz

e_0 = número de elementos inter-clusters

e_v = número de elementos iguais a 0 dentro dos clusters

$0 \leq \tau \leq 1$

Observa-se que a função de aptidão procura minimizar o número de elementos inter-clusters (ruídos) minimizando o movimento entre clusters; e lacunas nos clusters (células/famílias), maximizando o uso das máquinas nos clusters. Quanto mais próximo de 1 é o seu valor, melhor é a solução. Como observou Sarker em [42], a *Eficácia de Agrupamento* não é sensível à soluções que contenham clusters com linhas ou colunas vazias, permitindo que estas tenham valor de aptidão válido. Para superar esta falha, o algoritmo genético proposto por [22] e implementado neste trabalho não considera tais soluções como soluções viáveis.

3.9.4 Estratégia de seleção dos indivíduos reprodutores

Este passo do algoritmo genético trata da escolha dos indivíduos que irão servir como pais (reprodutores) para o processo de geração de novos indivíduos e também para decidir entre os pais e filhos gerados, quais irão para a próxima geração. Escolher simplesmente os melhores indivíduos pode levar o algoritmo a uma parada prematura em ótimos locais ainda distantes de um ótimo global. Em razão disto, em [22] foi dada uma probabilidade a cada indivíduo de ser selecionado, de acordo com o seu valor de aptidão; sendo a probabilidade de escolha maior para indivíduos de melhor aptidão e menor para indivíduos de pior aptidão, como em [22]. A função de probabilidade de escolha para cada indivíduo é:

$$\sum_{r=1}^{P} Probabilidade\ de\ escolha\ do\ r-ésimo\ indivíduo\ = \sum_{i=1}^{P} q'(1-q)^{r-1} \quad (3.18)$$

onde:

P = número de indivíduos na população

q = probabilidade de seleção do melhor indivíduo

q' = $\frac{q}{1-(1-q)^P}$

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

3.9.5 Critério de Parada

O critério de parada usado é o número de gerações, que variou de 500 (para a menor instância) a 20.000 (para a maior instância). Estes valores foram estabelecidos na tentativa de se obter um critério de parada similar ao relatado por Joines et al. em [22].

3.10 Conclusões

Como podemos concluir através da literatura, existe um grande número de técnicas para agrupamento e identificação de células de máquinas e famílias de partes e o mesmo pode-se dizer para as medidas de desempenho e de similaridade. Neste contexto o uso das metaheurísticas torna-se interessante devido ao fato destas permitirem o uso de funções objetivo multi-critério, várias medidas de desempenho, formação simultânea de famílias de partes e células de máquina sem a necessidade de uma inspeção visual da solução final por parte do projetista e a possibilidade de geração de soluções com diferentes quantidades de clusters para uma dada instância do PCM com a posterior determinação do número de clusters mais adequado para a referida instância.

Finalmente, para ainda justificar o uso de metaheurísticas para a solução do PCM, devemos observar que estas técnicas tem apresentado os melhores resultados aproximados da literatura para este problema, conforme pode ser verificado nos trabalhos de Joines et al. [22] e Resende et al. [37].

Apresentamos a seguir uma proposta de um algoritmo evolutivo (AE) para o PCM. A escolha do tipo de metaheurística foi tomada após um exaustivo estudo da literatura existente para o PCM com número variável de clusters. Este estudo nos mostrou que AGs ou AEs podem se tornar muito competitivos em relação à qualidade das soluções e tempo computacional exigidos.

Capítulo 4

O Algoritmo Evolutivo proposto para o PCM

4.1 O Algoritmo Evolutivo (AE) Proposto

Apesar de sua eficiência comprovada para diversos problemas de otimização, o desempenho dos algoritmos genéticos (AGs) na sua forma original não se mostra satisfatório para muitos problemas de otimização que já possuem algoritmos eficientes para a sua solução aproximada. Em razão disto, diversos pesquisadores tem proposto modificações nos AGs convencionais, incorporando à estes, técnicas de busca local, ou conceitos de outras metaheurísticas tais como: Simulated Annealing, Tabu Search, Scatter Search, entre outros. Com isso procura-se um algoritmo evolutivo que possua características básicas dos AGs e mecanismos adicionais, tornando-o mais especializado para determinada aplicação. Neste trabalho propomos um algoritmo evolutivo que possui as seguintes particularidades em relação à 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 possa gerar em baixo tempo computacional indivíduos di-

ferentes e com nível de aptidão média melhor do que se estes fossem gerados aleatoriamente.

2. Um novo mecanismo de cruzamento é usado para gerar novos indivíduos à partir de indivíduos pais ao invés do operador de cruzamento (*crossover*) tradicional. Além disso, os indivíduos pais disputam com seus filhos um lugar na população seguinte. O operador de mutação não é utilizado.

3. Um método de busca local é utilizado sobre parte da população à cada geração, realizando uma procura intensiva na vizinhança das melhores soluções encontradas pelo AE com o objetivo de atingir um número maior de soluções que representem ótimos locais num problema de otimização.

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

4.1.1 Estrutura do indivíduo

A estrutura (representação) do indivíduo é a mesma usada por Joines et. al [22] em seu algoritmo genético para o PCM, como já descrito no capítulo 3.

4.1.2 Função de Aptidão

A função de aptidão usada no AE proposto é a medida de desempenho *Eficácia de Agrupamento*, mostrada no item 3.9.3 do capítulo 3 deste trabalho e também usada em [22].

4.1.3 Geração da População Inicial

A população inicial constitui-se de indivíduos gerados parte aleatoriamente e parte gerados através de um procedimento de construção heurística. Este proce-

dimento tem sua idéia baseada na heurística apresentada em [49] para resolver o PCM, mas é aqui adaptado para incorporar passos randômicos para que se possa gerar indivíduos diferentes através de sua aplicação.

Heurística Randomizada para gerar População Inicial

Primeiramente, é necessário que seja 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 uma máquina i e uma máquina j, o número de partes atendidas em comum pelas mesmas. Sejam 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 a máquinas i e k e a variável $similaridade_{minima}$ 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 4.1.

De acordo com o algoritmo da figura 4.1, o cálculo das similaridades entre todas as máquinas e a determinação do 1º par de máquinas menos similar é feito em $O(m^2p)$ (as linhas 4 e 5 são $O(m)$, e a linha 7 é $O(p)$). As linhas 14 a 27 são feitas em $O(1)$. Na linha 30 é feita a atribuição das máquinas aos conjuntos de pares de máquinas menos similares Par_1, Par_2 e Par_3 em $O(1)$. Portanto, o algoritmo é computado em $O(m^2p)$. As linhas 14 e 21 garantem que não sejam formados pares de máquinas iguais. 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, 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.

```
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 4.1: Pseudocódigo do procedimento de determinação dos 3 pares de máquinas menos similares

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 é escolhido aleatoriamente um número de clusters k a serem formados dentro

da faixa de valores que varia de 2 a $\lceil \frac{m}{2} \rceil$ clusters. Após o término desta fase, os k clusters iniciais estarão determinados.

3. Na 3ª fase, primeiramente para a máquina i de menor rótulo ainda não pertencente a um cluster são determinados o 1º e o 2º cluster com maior índice de similaridade, sendo que 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. Então a máquina é atribuída aleatoriamente a um dos dois clusters, desde que o número máximo de máquinas por cluster não esteja alcançado. 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 seja associada a algum cluster e termina quando todas as máquinas tiverem sido clusterizadas.

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.

O número máximo de máquinas por cluster é calculado como sendo igual a $\lceil \frac{n^{o}\ de\ maquinas}{n_{clusters}} \rceil$ e o número máximo de partes por cluster igual a $\lceil \frac{n^{o}\ de\ partes}{n_{clusters}} \rceil$, onde $n_{clusters}$ representa o número de clusters a serem formados. Para apresentar o algoritmo da 1ª e 2ª fases do procedimento de construção, temos as seguintes variáveis a considerar:

$n_{clusters}$ = representa o número de clusters iniciais a serem formados

Par_1, Par_2 e Par_3 = os 3 pares de máquinas menos similares determinados pelo algoritmo da figura 4.1

$Par_{escolhido}$ = par de máquinas escolhido dentre Par_1, Par_2 e Par_3

C_k = k-ésimo cluster inicial formado

$Cromo$ = vetor de inteiros que representa o indivíduo

$similaridade_{minima}$ = menor similaridade dentre as similaridades entre as máquinas

e os clusters iniciais já formados

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

$maq_{escolhida}$ = máquina escolhida para ser um novo cluster inicial

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

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

p = número de partes

m = número de máquinas

$partes_{atendidas}$ = 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).

O pseudocódigo da figura 4.2 ilustra o procedimento de formação das máquinas semente dos clusters. A escolha das 2 primeiras máquinas semente é feita em $O(1)$ (linhas 2 a 4). A determinação do valor dos genes do indivíduo em construção referentes às duas primeiras máquinas semente também é $O(1)$ (linhas 5 e 6). As linha 7 e 8 a 12 são $O(p)$. A verificação que se faz na linha 16 se uma máquina i é semente é $O(1)$, checando num vetor de máquinas o valor da posição da máquina em questão que é falso se esta não é semente e verdadeiro se esta o é. O cálculo da similaridade entre uma máquina i e as sementes já formadas é feito nas linhas 18 a 22 em $O(p)$. Nas linhas 23 é feita a atualização da similaridade mínima entre uma máquina e as sementes já formadas, em O (1). Na linha 26 é escolhida a nova máquina semente, em $O(1)$. A atualização do vetor do estado das partes é feita em $O(p)$ nas linhas 28 a 32. O cálculo das similaridades será feito para todas as *m-(k-1)* máquinas que não são clusters iniciais (para todas as máquinas em que a linha 16 for verdadeira). Portanto, a complexidade das linhas 15 a 25 é dada por $O(mp)$. Como devem ser determinadas $n_{clusters}$ máquinas sementes e as 2 primeiras máquinas semente são definidas nas linhas 2 a 4, as linhas 13 a 34 são executadas $n_{clusters}$-2 vezes. Sendo o limite máximo de $n_{clusters}$ da ordem de m ($\lceil \frac{m}{2} \rceil$), a complexidade total do procedimento *Forma Clusters Iniciais* é $O(m^2p)$.

```
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_minima = 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_zmaq_escolhida ==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 4.2: Pseudocódigo do procedimento de determinação dos k clusters iniciais

Para a associação das máquinas restantes, considere C_1 e C_2 como sendo os dois melhores clusters (dois clusters com maior similaridade) para cada máquina em questão e a variável $tamanho_{maximo}$ como sendo o número máximo de máquinas por cluster. O pseudocódigo da figura 4.3 representa o algoritmo que associa cada máquina restante a um determinado cluster.

Sobre a complexidade do algoritmo da figura 4.3, podemos constatar o seguinte: A linha 9 é $O(tamanho_{maximo})$, tendo em vista que as máquinas de cada cluster estão em uma lista de acesso sequencial. Portanto, a complexidade das linhas 7 a 13 é $O(ptamanho_{maximo})$. A função *Atualizar*, da linha 14, responsável por determinar os clusters C_1 e C_2 é $O(1)$. A complexidade do trecho que vai da linha 4 a 16 é então $O(pn_{clusters}tamanho_{maximo})$. Como $tamanho_{maximo}$ é definido como $\lceil \frac{m}{n_{clusters}} \rceil$, a complexidade das linhas 4 a 16 é $O(mp)$. A escolha randomica de C_1 ou C_2 também é $O(1)$, o que nos dá uma complexidade $O(mp)$ para as linhas 3 a 18. A linha 19, responsável pela construção do indivíduo de acordo com a associação das máquinas aos clusters é $O(1)$. A complexidade da linha 3 é $O(1)$ e o resultado desta linha será verdadeiro para m-$n_{clusters}$ máquinas, o que significa que m-$n_{clusters}$ (ou seja, um número da ordem de m) máquinas devem ser clusterizadas. Portanto, a complexidade das linhas 2 a 20 é $O(m^2p)$. A complexidade do procedimento *Clusteriza Máquinas* é então $O(m^2p)$.

```
1: Procedimento Clusteriza Máquinas(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_mximo
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 randomicamente entre C_1 e C_2
18:    Fim-Se
19:    Cromo[i] = cluster_escolhido
20:  Fim-Para
21: Fim Procedimento
```

Figura 4.3: Pseudocódigo do procedimento de clusterização das máquinas as quais não são clusters inciais

Para a associação das partes aos clusters, considere as variáveis mostradas no pseudocódigo da figura 4.3 com as seguintes alterações:

$similaridade_{Ci}$ = similaridade entre o cluster C e a parte i

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

n = número de máquinas de um cluster C

$tamanho_{maximo}$ = número máximo de partes por cluster

O algoritmo para a associação de partes é o descrito na figura 4.4:

```
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 | máquina z ∈ cluster C
5:       Atualizar(C_1,C_2)
6:     Fim-Para
7:     cluster_escolhido = escolher randomicamente entre C_1 e C_2
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 4.4: Pseudocódigo do procedimento de clusterização das partes

Analogamente ao procedimento de clusterização das máquinas representado pela figura 4.3, a análise de complexidade do procedimento da figura 4.4 é o seguinte: A linha 4 é feita em $O(n_{max})$ onde n_{max} é o número máximo de máquinas permitido em um cluster e a 5 em *O(1)*. A complexidade das linhas 3 a 6 é portanto $O(n_{max}n_{clusters})$. Sendo o limite máximo do valor de n_{max} igual a $\lceil \frac{m}{n_{clusters}} \rceil$, a complexidade das linhas 3 a 6 é $O(m)$. As linhas 7 a 14 são feitas em $O(1)$. Deste modo, a complexidade das linhas 2 a 15 é $O(mp)$. Sendo assim, a complexidade do procedimento *Clusteriza Partes* é $O(mp)$.

Finalmente, o procedimento de construção de um indivíduo de acordo com o procedimento apresentado pode ser representado pelo pseudocódigo da figura 4.5:

1: **Procedimento** Constroi Indivíduos
2: **Procedimento** Forma Clusters Iniciais($n_{clusters}$)
3: **Procedimento** Clusteriza Maquinas($n_{clusters}$)
4: **Procedimento** Clusteriza Partes($n_{clusters}$)
5: **Fim** Procedimento

Figura 4.5: Pseudocódigo do procedimento de construção de indivíduos

A complexidade do procedimento é igual à complexidade dos dois primeiros procedimentos, ou seja, $O(m^2p)$. Vejamos um exemplo de construção de um indivíduo de uma instância do PCM representada pela figura a 4.6. Considere o número de

	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 4.6: Exemplo de matriz de entrada do PCM

clusters a serem formados igual a 3. O número máximo de máquinas por cluster será igual a 2 e o número máximo de partes por cluster igual a 3. Seja cada cluster k denotado por $C_k[x_1\text{-}x_2\text{-}...\text{-}x_m;\ y_1\text{-}y_2\text{-}...\text{-}y_n]$ onde a primeira e segunda seqüências indicam respectivamente as máquinas e partes integrantes do cluster. De acordo com as 4 fases do procedimento de construção de um indivíduo, uma possível solução pode ser determinada da seguinte maneira:

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 seja o 2º par. Então os dois clusters iniciais são o cluster 1 formado pela máquina 1 e o cluster 2 formado pela máquina 5, denotados por $C_1[1]$ e $C_2[5]$.

2. O terceiro cluster inicial será formado pela máquina com menor 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 2, que possui similaridade 0 com os 2 clusters iniciais. O cluster 3 então será: C_3[2]. Desta forma os k clusters iniciais já estão formados. São eles: C_1[1] e C_2[5] e C_3[2].

3. A fase 3 irá atribuir as máquinas restantes ao 1º ou 2º cluster com o qual elas tem maior similaridade, uma a uma. A máquina 3 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 = C_1[1-3]. A máquina 4 tem similaridade 2 com o cluster 3 e similaridade 1 com o cluster 1 (a parte P1 em comum com as máquinas 1 e 3). Suponha que M4 seja associada ao cluster 3. Agora o cluster 3 = C_3[2-4]. Por fim, a máquina 6 tem como 2 melhores clusters o cluster 2 (com o qual tem similaridade 2) e o cluster 1 (com o qual tem similaridade 1). Supondo que a máquina 6 seja associada ao cluster 2, então $C_2 = C_2$[5-6]. A configuração dos clusters após a associação das máquinas é a seguinte: C_1[1-3], C_2[5-6] e C_3[2-4].

4. As partes serão atribuídas ao 1º ou 2º cluster que possuir o maior número de máquinas que realizam uma operação de manufatura sobre as mesmas. Para a atribuição das partes, consideremos os 1º e 2º clusters candidatos e as seguintes escolhas randômicas para cada parte:
   P1: C_1[1-3] e C_3[2-4]. Cluster escolhido: C_1. Então $C_1 = C_1$[1-3;1]
   P2: C_3[2-4] e C_1[1-3;1]. Cluster escolhido: C_3. Então $C_3 = C_3$[2-4;2]
   P3: C_2[5-6] e C_1[1-3;1]. Cluster escolhido: C_2. Então $C_2 = C_2$[5-6;3]
   P4: C_1[1-3;1] e C_2[5-6;3]. Cluster escolhido: C_1. Então $C_1 = C_1$[1-3;1-4]
   P5: C_3[2-4;2] e C_1[1-3;1-4]. Cluster escolhido: C_3. Então $C_3 = C_3$[2-4;2-5]
   P6: C_2[5-6;3] e C_1[1-3;1-4]. Cluster escolhido: C_2. Então $C_2 = C_2$[5-6;3-6].
   P7: C_1[1-3;1-4] e C_2[5-6;3-6]. Cluster escolhido: C_1. Então $C_1 = C_1$[1-3;1-4-7].
   P8: C_3[2-4;2-5] e C_1[1-3;1-4-7]. Cluster escolhido: C_3. Então $C_3 = C_3$[2-4;2-5-8].

Após o término do procedimento de construção, a configuração dos clusters é a seguinte: C_1[1-3;1-4-7], C_2[5-6;3-6] e C_3[2-4;2-5-8]. 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 4.7 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 4.7: Possível matriz de clusterização obtida pelo procedimento de construção a partir da matriz da figura 4.6

Através desta heurística pode-se gerar indivíduos que representam soluções do PCM nas quais algumas 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 à clusters. Porém, é possível que se formem indivíduos inválidos. No exemplo mostrado anteriormente, por exemplo, se as partes P3 e P6 fossem associadas ao cluster 1 e as outras escolhas permanecessem as mesmas, o cluster 2 não teria partes associadas à ele, o que caracteriza uma solução inválida. Além disso, partes podem ser associadas à clusters que não possuem máquinas que realizam operações sobre elas, podendo o cluster desta forma não conter elementos iguais a 1, o que também caracteriza uma solução inválida. Pode também ocorrer a formação de clusters com colunas ou linhas vazias (quando uma determinada parte pertence a um cluster não possui máquinas que a atende ou uma máquina pertence a um cluster que não possui partes sobre as quais ela realiza uma operação de manufatura). Estas soluções não são descartadas, devido ao fato de a busca local ser capaz de, dada uma solução inválida obter uma

solução válida.

4.1.4 Estratégia de seleção dos indivíduos reprodutores

A escolha dos indivíduos reprodutores e dos indivíduos que irão estar na próxima geração é feita usando a mesma função de probabilidade usada por Joines et. al em [22] relatada no item 3.9.4 do capítulo 3, a qual dá uma probabilidade maior de escolha para indivíduos de melhor aptidão e uma menor probabilidade de escolha para indivíduos de menor aptidão.

4.1.5 Mecanismo de cruzamento

O operador crossover tradicional foi substituído no nosso AE por um mecanismo de cruzamento, que gera filhos para uma próxima geração com base em informação genética dos pais, de maneira que estes filhos tenham consigo as características destes últimos, onde o número de filhos e o número de pais pode variar. Escolhidos os N_pais pais para o cruzamento (através do processo de seleção referenciado no item anterior deste capítulo), onde $N\_pais \geq 2$ é um parâmetro de entrada do cruzamento, cada gene do filho referente à uma máquina receberá aleatoriamente o valor correspondente à uma das das c células de máquina nas quais a presença desta máquina é mais freqüente, com base nas N_pais soluções pai, onde c é um parâmetro de entrada. Para cada parte é feita uma lista das M máquinas que mais aparecem no mesmo cluster da referida parte, com base nas N_pais soluções pai, sendo M também um parâmetro de entrada. Após a clusterização das máquinas para o indivíduo filho, para cada parte uma máquina então é escolhida aleatoriamente nas respectivas listas de máquinas 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. Para a apresentação do pseudocódigo do mecanismo de cruzamento, considere as seguintes variáveis:

N_pais = número de pais

m = número de máquinas

p = número de partes

Pop = população de indivíduos corrente

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

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 4.8 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] 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 4.8: Pseudocódigo do procedimento de cruzamento

A respeito da complexidade do procedimento de cruzamento, temos o seguinte a esclarecer: a função *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 frequentemente associada, de acordo com os genes de máquina dos indivíduos pai. Esta lista é ordenada pela frequência

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 frequentemente associada, de acordo com os genes de máquina e de parte dos indivíduos pai. A função *Escolher_cluster_maquina* escolhe um cluster na lista de clusters de uma máquina k e a função *Escolher_maq_parte* escolhe uma máquina na lista de máquinas de uma determinada parte j. A função *Escolhe_pais* escolhe *N_pais* indivíduos de uma população (que é uma lista de indivíduos de acesso sequencial), tendo complexidade $O(N\_paisTam_{Pop})$, onde Tam_{Pop} é o tamanho da população, dado em função do número de máquinas m. Desta forma, a função escolhe pais é $O(N\_paism)$. As funções *Incluir_lista_cluster*, *Incluir_maq_lista_parte*, *Escolher_cluster_maquina* e *Escolher_maq_parte* tem complexidade $O(m)$, uma vez que a lista de máquinas e de clusters é da ordem de m. As linhas 3 a 7 tem complexidade $O(m^2N\_pais)$, as linhas 8 a 16 $O(m^2N\_paisp)$ e as linhas 17 a 24 $O(m^2N\_Filhos + mN\_Filhosp)$ que é $O(mN\_Filhosp)$, tendo em vista que o número de partes p é maior que o número de máquinas m. Considerando que o número de filhos *N_Filhos* e o número de pais *N_pais* são iguais e bem menores que m e p, a complexidade do procedimento é $O(m^2p)$. Vejamos um exemplo do procedimento de cruzamento. Considere uma instância do PCM com 6 máquinas e 8 partes, $N\_pais = 3$, $c = 2$ e $M = 3$. Os pais, aqui representados por C1, C2 e C3 são os seguintes:

C1 = [1 3 2 1 3 2 | 1 1 3 1 2 3 2 2]

C2 = [1 2 2 3 2 1 | 1 2 3 1 2 1 3 2]

C3 = [2 3 1 1 3 2 | 2 1 1 2 3 3 3 2]

As 2 células de máquina nas quais cada máquina mais aparece, de acordo com os pais são:

M1 : 1 e 2; M2 : 3 e 2; M3 : 2 e 1; M4 : 1 e 3; M5 : 3 e 2 e M6 : 2 e 1

As 3 máquinas mais freqüentes em um mesmo cluster para cada parte, são as seguintes, de acordo com as soluções pais: P1 = M1, M6 e M4; P2 = M3, M4 e M2; P3 = M4, M2 e M3; P4 = M1, M6 e M4; P5 = M3, M5 e M2; P6 = M2, M5 e M1; P7 = M2, M3 e M4 e P8 = M3, M6 e M2.

1 possível filho, aqui denotado por F1 seria:

F1 = [1 3 1 3 2 2 | 1 3 1 2 2 3 1 2]

Neste caso, para as máquinas M1 a M6 foram escolhidos respectivamente os seguintes clusters de suas listas de clusters: 1, 3, 1, 3, 2 e 2 e as partes P1 a P8 escolheram respectivamente as seguintes máquinas de suas listas de máquinas: 1, 2 ou 4, 3, 6, 5, 2, 3 e 6.

O procedimento de cruzamento pode ser aplicado também a pais com diferentes números de cluster. Por exemplo, considere uma instância do PCM com a mesma dimensão da relatada no exemplo anterior, os mesmos valores para os parâmetros c, M, o valor de N_pais igual a 5 e os seguintes indivíduos pai:
C1 = [1 1 2 2 2 1 | 2 2 1 1 2 2 1 2]
C2 = [2 2 1 1 2 1 | 2 1 2 2 1 1 2 2]
C3 = [2 2 1 2 1 1 | 1 2 2 2 1 1 1 2]
C4 = [3 3 2 1 2 2 | 2 2 3 3 2 2 1 1]
C5 = [3 3 2 1 1 2 | 1 1 3 3 2 2 2 3]
As 2 células de máquina nas quais cada máquina mais aparece, de acordo com os pais são:
M1 : 2 e 3; M2 : 2 e 3; M3 : 1 e 2; M4 : 1 e 2; M5 : 1 e 2 e M6 : 1 e 2
As 3 máquinas mais freqüentes em um mesmo cluster para cada parte, são as seguintes, de acordo com as soluções pais: P1 = M5, M3 e M4; P2 = M4, M3 e M5; P3 = M1, M2 e M4; P4 = M1, M2 e M4; P5 = M3, M5 e M6; P6 = M3, M5 e M6; P7 = M6, M1 e M3 e P8 = M1, M2 e M4.

Escolhendo-se os clusters 2, 2, 1, 1, 2 e 1 para as máquinas M1 a M6, respectivamente, e as máquinas M5, M3, M1, M2, M3, M6, M1 e M1 para as partes P1 a P8, respectivamente, o filho F2 teria a seguinte configuração: F2 = [2 2 1 1 2 1 | 2 1 2 2 1 1 2 2]

Nota-se que o filho F2 gerado possui 2 clusters, ou seja, um número de clusters diferente do número de clusters dos pais C4 e C5. Obviamente um filho gerado por pais que possuem um número diferente de clusters sempre terá um número de clusters diferente de pelo menos um de seus pais, mas quando se usa pais com um mesmo número de clusters o filho pode também não ter o mesmo número de cluster dos pais. Por exemplo, se a escolha das células de máquina para as máquinas M2 e

M4 para o filho F1 fosse respectivamente 2 e 1, todas as partes seriam necesariamente associadas ou ao cluster 1 ou 2 e então F1 teria apenas 2 clusters, ainda que seus pais possuam 3 clusters. Além disso, podem surgir filhos inválidos. Novamente considerando o filho F1, se os genes de máquinas permanecessem como no exemplo mostrado e as escolhas das máquinas para as partes P2 e P6 fossem respectivamente M3 e M5, F1 teria a seguinte configuração: F1 = [1 3 1 3 2 2 | 1 2 2 2 2 2 1 2], o que caracteriza uma solução inválida pois as máquinas M2 e M4 estariam num cluster o qual não possui nenhuma parte (cluster 3).

Evidentemente, quanto menor o número de pais N_pais e os valores dos parâmetros c e M maior é a possibilidade da geração de filhos idênticos. Por outro lado, o número de pais N_pais e os valores de c e M também não podem ser demasiadamente grandes, para não tornar o processo demasiadamente aleatório.

4.1.6 Procedimento de Busca Local

O melhoramento das soluções num AE se dá de geração a geração através da manutenção de bons indivíduos (boas soluções) e da criação de filhos com características dos pais através dos operadores de cruzamento. Os operadores clássicos de mutação por sua vez procuram modificar uma pequena parte de alguns indivíduos para que a busca se diversifique. Porém, este processo de convergência para bons indivíduos pode ser acelerado e melhorado, de modo que se chegue às melhores soluções dos problemas tratados em um número menor de gerações. Ou ainda, pode se chegar à soluções nunca antes alcançadas. Estes objetivos podem ser obtidos através da incorporação de um procedimento de busca local ao algoritmo evolutivo. A busca local se constitui de uma verificação de soluções vizinhas à boas soluções. Esta busca se dá através da mudança de valores nos genes dos indivíduos, com base em um certo critério que é definido de acordo com a natureza do problema em questão.

Neste trabalho foi incorporado ao AE um procedimento de busca local inspirado no modelo proposto por Resende e Gonçalves em [37]. 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 repetidamente 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 em (4.1), com relação à cada cluster k.

$$coef_{ik} = \begin{cases} N_{1intrak} - N_{ruidosk} - N_{0intrak}, & se \quad N_{1intrak} \neq 0; \\ -\infty, & caso \ \ contrário. \end{cases} \tag{4.1}$$

onde:

$N_{1intrak}$ = número de elementos do clusters k iguais a 1 se a parte i for associada à este cluster.

$N_{ruidosk}$ = número de elementos iguais a 1 fora do cluster k se a parte i for associada à este cluster.

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

Após isso, cada parte é associada ao cluster para o qual possui o maior coeficiente. Caso haja empate entre coeficientes, a parte então 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 terminada. Caso haja melhoria, executa-se o passo 2.

Passo 2: O passo 2 é idêntico ao passo 1, mas agora faz-se a mudança das máquinas de cluster calculando-se os coeficientes para cada máquina em relação à cada cluster, de acordo com a alocação das partes feitas pelo passo 1. O coeficiente de cada máquina i em relação à cada cluster k é calculo pela equação 2.6, onde:

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

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

associada à este cluster.

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

Igualmente às partes, 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 terminada. Caso haja melhoria, executa-se o passo 1 novamente. A penalidade aplicada ao valor de $coef_{ik}$ quando $N_{1intrak}$ é igual a 0 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 [37] 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 [37], o coeficiente é o seguinte:

$$coef_{iC} = \frac{N_1 - N_{1,C}^{Out}}{N_1 + N_{0,C}^{In}} \tag{4.2}$$

onde:

N_1 = número total de elementos da matriz iguais a 1.

$N_{1,C}^{Out}$ = número de elementos inter-clusters causados pela associação da máquina ou parte i ao cluster C.

$N_{0,C}^{In}$ = número de elementos iguais a 0 dentro dos clusters da matriz de clusterização causados pela associação da máquina ou parte i ao cluster C.

Comparando a equação 4.2 com a equação 3.3, concluimos que a busca local proposta por Resende e Gonçalves utiliza o conceito da medida de desempenho *Eficácia de Agrupamento* na procura por melhores soluções. Entretanto, a busca local pode encontrar soluções com clusters cujas colunas ou linhas estejam vazias e que ainda assim melhorem o coeficiente mostrado acima para máquinas ou partes. Isto se deve ao fato de que a medida de desempenho *Eficácia de Agrupamento* possui esta falha e como esta busca local utiliza a mesma formulação matemática de tal

medida, também possui tal falha. Em nossa proposta de busca local esta falha é evitada através do fator de penalidade usado na mesma. Entretanto, a nossa proposta não evita que uma solução que contenha inicialmente linhas ou colunas vazias se torne obrigatoriamente uma solução válida, já que o que determina a mudança de máquinas e partes é a melhoria da solução, avaliada pela medida de desempenho *Eficácia de Agrupamento*, que pode diminuir de valor caso a solução se transforme numa solução válida. O coeficiente usado em nossa busca local portanto decide sobre as mudanças de cluster de máquinas e partes, mas o que confirma esta mudança é a melhoria do valor da *Eficácia de Agrupamento*.

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 4.9 representa o procedimento de busca local. O cálculo de cada coeficiente $coef_{ik}$ de uma parte é feito em $O(m)$ (linha 6), pois para cada cluster k são verificadas as associações de todas as máquinas. Como devem ser calculados os coeficientes de cada parte em relação à cada cluster, o cálculo de todos os coeficientes das partes é feito em $O(mpn_{clusters})$. Como o número de clusters é da ordem do número m de máquinas, a complexidade das linhas 4 a 9 é $O(m^2p)$. A da linha 10 é $O(p)$.

O cálculo da aptidão do indivíduo, que é de acordo com medida de desempenho *Eficácia de Agrupamento* (equação 2.4) é feita em $O(mp)$ pois para cada máquina pertencente a um cluster k deve ser verificado à que clusters todas as p partes pertencem e se estas últimas necessitam das máquina em questão, de acordo com a matriz de entrada do PCM (linhas 11 e 15, pois para verificar se houve ou não melhoria na solução é preciso recalcular a aptidão do indivíduo). Assim, as linhas 11 a 14 tem complexidade $O(mp)$ (a 11 é $O(mp)$, a 12 é $O(p)$ e as demais $O(1)$). Portanto, as linhas 4 a 14 tem complexidade $O(m^2p)$. Analogamente ao cálculo dos coeficientes das partes, cada coeficiente das máquinas é feito em $O(p)$, pois para cada cluster k são verificadas as associações de todas as partes. Como devem ser calculados os coeficientes de cada máquina em relação à cada cluster, o cálculo de todos os coeficientes das máquinas é feito em $O(mpn_{clusters})$. Como o número de clusters é da ordem do número m de máquinas, a complexidade das linhas 16 a 21 é $O(m^2p)$. As linhas 22 a 26 tem complexidade $O(mp)$ (a 22 é $O(m)$, a 23 $O(mp)$,

```
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:      reassociar 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 4.9: Pseudocódigo do procedimento de Busca Local

a 24 $O(m)$ e as demais $O(1)$). Portanto, a complexidade do procedimento de busca local é $O(m^2p)$.

A seguir é mostrado um exemplo da busca local. Considere uma solução de uma instância do PCM constituída de 4 máquinas, 8 partes e 2 clusters, representada pela figura 4.10:

A aptidão da solução inicial é igual a $\frac{21-6}{21+1} = 0{,}6818$. O primeiro passo da busca local irá tentar realocar as partes entre os 2 clusters da solução. Seja $Coef_{ik}$ o coeficiente da i-ésima parte em relação ao cluster k. A tabela 4.1 mostra os coeficientes de cada parte para os dois clusters em questão:

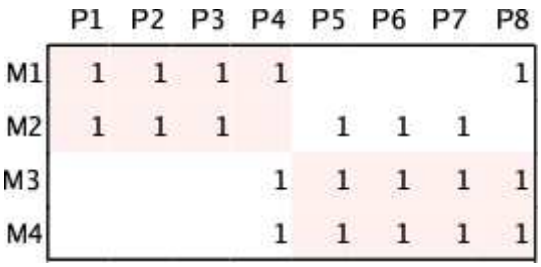


Figura 4.10: Matriz-solução do PCM inicial para a busca local

Coeficientes	
$Coef_{11}$ = **2-0-0 = 2**	$Coef_{12}$ = 0-2-2 = -4
$Coef_{21}$ = **2-0-0 = 2**	$Coef_{22}$ = 0-2-2 = -4
$Coef_{31}$ = **2-0-0 = 2**	$Coef_{32}$ = 0-1-2 = -3
$Coef_{41}$ = 1-2-1 = -2	$Coef_{42}$ = **2-1-0 = 1**
$Coef_{51}$ = 1-2-1 = -2	$Coef_{52}$ = **2-1-0 = 1**
$Coef_{61}$ = 1-2-1 = -2	$Coef_{62}$ = **2-1-0 = 1**
$Coef_{71}$ = 1-2-1 = -2	$Coef_{72}$ = **2-1-0 = 1**
$Coef_{81}$ = 1-2-1 = -2	$Coef_{82}$ = **2-1-0 = 1**

Tabela 4.1: Coeficiente das partes em relação aos clusters no primeiro passo da busca local

Os coeficientes em negrito indicam o maior coeficiente para cada parte. Observando a tabela 4.1, o novo conjunto de famílias de partes a serem formadas é o seguinte: A família 1 contém as partes P1, P2 e P3 e a família 2 as restantes. Como houve melhoria na aptidão, a nova solução do PCM é a seguinte (figura 4.11):

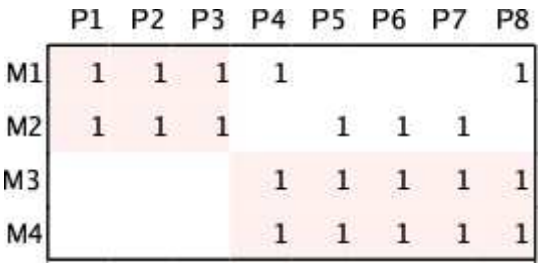


Figura 4.11: Matriz-solução do PCM depois da aplicação do primeiro passo da busca local

A aptidão da nova solução é igual a $\frac{21-5}{21+0} = 0{,}7619$. Executa-se então o segundo passo da busca local, que procura realocar as máquinas aos clusters, com base na nova formação de família de partes. A tabela 4.2 mostra os coeficientes de cada máquina em relação aos dois clusters da solução:

Coeficientes	
$Coef_{11}$ = **3-2-0 = 1**	$Coef_{12}$ = 2-3-3 = -4
$Coef_{21}$ = **3-3-0 = 0**	$Coef_{22}$ = 3-3-2 = -2
$Coef_{31}$ = 0-5-3 = -8	$Coef_{32}$ = **5-0-0 = 5**
$Coef_{41}$ = 0-5-3 = -8	$Coef_{42}$ = **5-0-0 = 5**

Tabela 4.2: Coeficiente das máquinas em relação aos clusters no segundo passo da busca local

Pela tabela 4.2, nota-se que as células de máquina não terão alteração em sua configuração, o que faz com que não haja melhoria na aptidão da solução. A busca local então é finalizada e a solução mostrada na figura 4.11 é a solução resultante da aplicação da busca local na solução inicial representada pela figura 4.10.

4.1.7 Pseudocódigo do AE proposto

A seguir é apresentado na figura 4.12 o pseudocódigo do AE proposto neste trabalho:

```
1: Procedimento Algoritmo Evolutivo
2:   Gera populacao inicial(Pop)
3:   nº de gerações = 1
4:   melhor indivíduo = melhor indivíduo(Pop)
5:   Enquanto nº de gerações < nº máximo de gerações faça
6:     Pop_{+1} = ∅
7:     cruzamento(Pop,Pop_{+1})
8:     Pop = Pop_{+1}
9:     Busca local(Pop)
10:    Atualizar melhor indivíduo
11:    nº de gerações++
12:  Fim-Enquanto
13:  Imprimir melhor indivíduo
14: Fim Algoritmo Evolutivo
```

Figura 4.12: Pseudocódigo do AE proposto

onde:

Pop = população corrente

Pop_{+1} = população seguinte à população corrente

No capítulo seguinte deste trabalho, propomos um algoritmo GRASP para a solução aproximada do PCM. O GRASP, embora seja recente dentre as metaheu-

rísticas mais populares da área de otimização, tem mostrado um bom desempenho em diferentes aplicações.

Capítulo 5

Uma Metaheurística GRASP para o PCM

5.1 A Metaheurística GRASP

Proposta por Resende e Feo em [38], a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) é um método iterativo onde cada iteração consiste em duas fases: fase de construção, na qual uma solução é gerada; e fase de busca local, na qual é feita uma pesquisa na vizinhança da solução gerada pela fase de construção com o objetivo de encontrar um ótimo local. Na fase de construção uma solução é construída iterativamente, elemento a elemento. Antes da inserção de um novo elemento na solução, é feita uma lista C ordenada de todos os elementos candidatos a serem o elemento inserido, associando-se um valor a cada elemento de C, de acordo com uma função gulosa $g : \mathrm{C} \mapsto \mathrm{R}$, que calcula o benefício de se inserir cada elemento na solução. Forma-se então um subconjunto de C contendo os melhores candidatos a serem inseridos, denominada lista de candidatos restrita (LCR). Por fim, a escolha do elemento a ser inserido é feita aleatoriamente na LCR (a componente probabilística do procedimento). A próxima iteração da fase de construção

reconstrói as listas C e LCR. As iterações da fase de construção são repetidas até que a solução esteja completamente formada (o aspecto adaptativo desta metaheurística é que a cada iteração da construção são recalculados os valores da função gulosa g associados aos elementos candidatos, de maneira a refletir as mudanças oriundas da inserção do elemento anterior [45]). Considere $\alpha \in [0,1]$ como um parâmetro. O pseudocódigo da figura 5.1 descreve a fase de construção GRASP.

```
1: Procedimento Construcao(g(.),α,s)
2:   s = ∅
3:   Inicialize o conjunto C de candidatos
4:   Enquanto s incompleta faça
5:     t_min = min{g(t) | t ∈ C}
6:     t_max = max{g(t) | t ∈ C}
7:     LCR = {t ∈ C | g(t) ≤ t_min + α(t_max-t_min)}
8:     Selecione, aleatoriamente, um elemento t ∈ LCR
9:     s = s ∪ {t}
10:    Atualize o conjunto C de candidatos
11:  Fim-Enquanto
12:  Retorne s
13: Fim Procedimento
```

Figura 5.1: Pseudocódigo do procedimento de construção

Pelo pseudocódigo da figura 5.1, podemos constatar que o parâmetro α controla o nível de aleatoriedade do procedimento *Construcao*. Um valor para α igual a 0 resulta na produção de soluções puramente gulosas e um valor de α igual a 1 na produção de soluções puramente aleatórias. Numa metaheurística GRASP tradicional, cada solução gerada na fase de construção é submetida à busca local, tendo em vista que provavelmente estas não representam ótimos locais. Seja V o conjunto de soluções vizinhas à uma solução dada s (entende-se como solução vizinha uma solução que se diferencie de s pela mudança de seus componentes) e *f()* a função que associa um valor a uma dada solução. Um procedimento de busca local na vizinhança da solução dada s pode ser descrito pelo pseudocódigo da figura 5.2.

O único parâmetro a ser ajustado numa metaheurística GRASP tradicional é o parâmetro α, que controla o tamanho da *LCR*. Em [36], M. G. C. Resende e C. C. Ribeiro resolveram um problema de maximização através da metaheurística GRASP. Como neste caso o objetivo foi maximizar o valor da função objetivo,

a linha 7 do pseudocódigo da figura 5.1 será: $LCR = \{t \in C \mid g(t) \geq t_{min} + \alpha(t_{max}\text{-}t_{min})\}$. Desta forma, para problemas de maximização a construção gulosa ocorrerá quando o valor de α for igual a 1 e a construção aleatória quando α for igual a 0. Os autores mostraram experimentalmente que valores de α próximos a 1 (construção gulosa) resultam em um número pouco diverso de soluções com uma média alta da função objetivo, e que valores de α próximos de 0 (construção aleatória) resultam em um número muito diverso de soluções, porém com média baixa de função objetivo. Obviamente nenhum dos dois valores de α citados (0 e 1) são desejáveis pois, provavelmente, a metaheurística GRASP não irá encontrar boas soluções tendo como ponto de partida um número diverso de soluções ruins ou irá ficar "presa" a um ótimo local tendo como ponto de partida um número muito restrito de boas soluções. O que mais provavelmente conduz a boas soluções é um conjunto relativamente diverso de soluções com um satisfatório valor médio da função objetivo associada à tais soluções. Em [36], o valor de α igual a 0,8 é recomendado.

```
1: Procedimento Busca Local(s)
2:   s* = s {Melhor solução recebe solução inicial s}
3:   Obter V de s
4:   Enquanto |V| > 0 faça
5:     Selecione s' ∈ V
6:     Se f(s') melhor que f(s*) então s* = s'
7:     Obter V de s'
8:   Fim-Enquanto
9: Fim Procedimento
```

Figura 5.2: Pseudocódigo do procedimento Busca Local

Além disso, M. G. C. Resende e C. C. Ribeiro mostram também que o desempenho da fase de busca local é afetado pelo valor de α. Para α igual a 0 (construção randômica) o valor médio da função objetivo obtido pela busca local é baixo, embora a melhor solução tenha sido encontrada. Quando α é igual a 1 (construção gulosa), o valor médio da função objetivo obtido pela busca local é alto, mas a melhor solução não é encontrada. Para valores de α iguais a 0,6 e 0,8, têm-se um valor médio alto da função objetivo obtido pela busca local e a melhor solução é alcançada. Isto posto, conclui-se que a calibração do valor de α é ponto crítico da

eficiência de uma metaheurística GRASP e valores intermediários para α são mais adequados que valores extremos, devido ao fato de permitirem uma maior diversidade de boas soluções semente para a busca local. Pode-se criar procedimentos GRASP mais sofisticados incluindo estratégias adaptativas para o parâmetro α [45], através do ajuste deste parâmetro com base num histórico dos resutados de iterações anteriores. Finalmente, uma metaheurística GRASP básica pode ser representada pelo pseudocódigo da figura 5.3:

```
1: Procedimento GRASP(g(.),α)
2:   Melhor Solução = ∅
3:   Para Iter = 1,2,...,N_MAX faça
4:     Construcao(g(.),α)
5:     Busca Local(s)
6:     Atualiza Melhor Solução
7:   Fim-Para
8:   Imprimir Melhor Solução
9: Fim Procedimento
```

Figura 5.3: Pseudocódigo do procedimento GRASP

5.2 A metaheurística GRASP proposta para o PCM

Os pontos que diferem a nossa metaheurística GRASP de uma metaheurística GRASP tradicional são os seguintes:

1. Devido à natureza do PCM, a fase de construção possui duas etapas. Como o objetivo é encontrar clusters eficientes com máquinas e partes, na primeira etapa da fase de construção é feito o agrupamento de máquinas e numa segunda etapa é feito o agrupamento de partes. A fase de construção possui portanto duas listas C de candidatos e duas listas de candidatos restrita LCR. Cada solução é gerada ao término das duas etapas da fase de construção. O parâmetro controlador do tamanho da primeira LCR é denominado β e o parâmetro controlador da segunda LCR é denominado α.

2. Implementamos versões usando uma estratégia adaptativa dinâmica (GRASP reativo) para escolha dos parâmetros β e α, além do algoritmo GRASP tradici-

onal. Esta estratégia avalia nas primeiras iterações GRASP, diferentes valores para o par (β, α). Num determinado momento da execução (nº de iterações), é feito uma avaliação destes parâmetros e a partir daí, o par (β, α) é fixado nas iterações restantes usando valores que obtiveram as melhores soluções até então.

3. Utilizamos três procedimentos de busca local alternadamente, sendo que um deles é o mesmo usado pelo algoritmo evolutivo proposto neste trabalho e os outros dois novos procedimentos propostos por nós.

A seguir trataremos da explicação detalhada de cada uma das etapas da metaheurística GRASP, como: estrutura da solução, função objetivo, algoritmo de construção, estratégia adaptativa e algoritmos de busca local.

5.2.1 Estrutura das soluções

A estrutura das soluções é idêntica à estrutura dos indivíduos do algoritmo evolutivo proposto no capítulo anterior deste trabalho.

5.2.2 Função Objetivo

A função objetivo das soluções é equivalente à função de aptidão usada pelo algoritmo evolutivo proposto no capítulo anterior deste trabalho.

5.2.3 Fase de Construção

Antes da fase de construção propriamente dita, devem ser selecionadas as máquinas que serão as sementes dos clusters. Para tal, inicialmente é feito o cálculo da similaridade de Jaccard (descrita no capítulo 3) entre todos os pares de máquinas e criada uma matriz triangular superior com $\frac{m^2-m}{2}$ elementos (onde m é o número de máquinas) nos quais são armazenados os rótulos das partes atendidas em comum por cada par de máquina i e j e a similaridade de Jaccard entre as mesmas. Esta

informação será usada pela fase de construção propriamente dita. Denotamos aqui esta matriz por *MatSim*. Além disso, As similaridades de Jaccard entre cada par de máquina serão armazenadas ordenadamente (ordem ascendente) em um vetor para posterior utilização. Este vetor é denominado *VSim*. Sejam as seguintes variáveis:

x_{iz} = variável binária que é igual a 1 se a máquina i efetua uma operação de manufatura sobre a parte z e 0 caso contrário

x_{ijz} = variável binária que é igual a 1 se as máquinas i e j realizam uma operação de manufatura sobre a parte z e 0 caso contrário

P = conjunto de todas as partes

p = número de partes

Jac_{ij} = similaridade de Jaccard entre as máquinas i e j

O pseudocódigo da figura 5.4 ilustra o procedimento utilizado para calcular e armazenar tais informações:

1: **Procedimento** Calcula Similaridades
2: k = 1
3: **Para** i=1,...,m-1 **faça**
4: **Para** j=i+1,...,m
5: Calcule $T_i = \sum_{z=1}^{p} x_{iz}$, $T_j = \sum_{z=1}^{p} x_{jz}$, e $C_{ij} = \sum_{z=1}^{p} x_{ijz}$
6: *MatSim*[i,j-i] = $z \in$ P | $x_{ijz} = 1$, $\forall$ z=1,...p
7: $Jac_{ij} = \frac{C_{ij}}{T_i + T_j - C_{ij}}$
8: VSim[k] = Jac_{ij}
9: **Fim-Para**
10: **Fim-Para**
11: Ordena(VSim)
12: **Fim** Procedimento

Figura 5.4: Pseudocódigo do procedimento de cálculo e armazenamento de similaridades entre pares de máquinas

Sobre a complexidade do procedimento da figura 5.4, temos o seguinte: As linhas 5 e 6 são feitas em $O(p)$. A linha 7 é $O(1)$. O trecho das linhas 3 a 10 tem portanto uma complexidade $O(m^2\mathrm{p})$. A função *Ordena* realiza uma ordenação dos $\frac{m^2-m}{2}$ elementos de *VSim*, portanto tem complexidade O ($\frac{m^2-m}{2}\mathrm{Log}(\frac{m^2-m}{2})$). Sendo o número de partes p maior que o número de máquinas m, a complexidade do procedimento *Calcula Similaridades* é O (m^2p). Com base na estrutura *VSim* são escolhidos os dois primeiros clusters, cada um formado por uma das máquinas do

par selecionado aleatoriamente dentre os k pares de *VSim* com menor índice de similaridade, onde k é um parâmetro de entrada. Cada novo cluster então é definido como sendo a máquina cujo maior índice de similaridade de Jaccard com os clusters já formados é o menor entre os maiores índices de similaridade das máquinas ainda não agrupadas e os clusters ja formados. Considerando este índice como sim_{min}, o *i-ésimo* cluster representado por C_i, Jac_{iC_i} a similaridade entre o *i-ésimo* cluster e a *i-ésima* máquina, o conjunto Par_r contendo os elementos m_1 e m_2 sendo o par de máquinas escolhido como clusters iniciais e o número de clusters a serem formados iguais a $n_{clusters}$,o pseudocódigo da figura 5.5 ilustra o procedimento de formação dos clusters iniciais:

```
1: Procedimento Forma Clusters Iniciais
2:   C_1 = m_1 ∈ Par_r
3:   C_2 = m_2 ∈ Par_r
4:   Para k=3,...,n_clusters faça
5:     sim_min = ∞
6:     Para i=1,...,m faça
7:       Se máquina i ∉ C_c, ∀ c=1,...,k-1
8:         similaridade = max{Jac_iC_c}, ∀ c=1,...,k-1
9:         Se sim_min é atualizada então máquina escolhida = i
10:    Fim-Para
11:    C_k = máquina escolhida
12:  Fim-Para
13: Fim Procedimento
```

Figura 5.5: Pseudocódigo do procedimento de formação dos clusters iniciais

As linhas 2 e 3 tem complexidade $O(1)$. A linha 7 é O (1), consultando um vetor de "estado" das máquinas para verificar se uma máquina é ou não semente. A linha 8 verifica a similaridade entre uma máquina e os k-1 clusters já formados, utilizando a estrutura *MatSim* para se ter acesso direto às similaridades de Jaccard entre os pares de máquina. Como o valor máximo de k-1 é igual a $n_{clusters}$-1, a complexidade da linha 8 é $O(n_{clusters})$, ou seja, O(m). Assim, as linhas 6 a 10 tem uma ordem de complexidade $O(m^2)$. A complexidade total do procedimento da figura 5.5 é portanto igual à complexidade das linhas 4 a 12, a qual é $O(n_{clusters}m^2)$, ou O (m^3). Após formados os clusters inciais, cada um contendo uma máquina, é usada a primeira etapa da fase de construção para que as máquinas restantes sejam agrupadas. Para o agrupamento de cada máquina restante à um determinado cluster

é gerada uma lista C contendo todos os clusters. Desta lista, são selecionados os melhores candidatos para que seja formada a lista de candidatos restrita LCR, de acordo com uma função associada à cada cluster candidato que representa o benefício de se agrupar a máquina i ao cluster candidato c. Tal função, denotada por *f(c)*, é definida pela equação 5.1:

$$f(c) = \frac{T_{iC_c}}{T_i + T_{C_c} - T_{iC_c}} \tag{5.1}$$

onde:

T_{iC_c} = número de partes atendidas em comum entre a máquina i e todas as máquinas pertencentes ao *c-ésimo* cluster

T_i = número de partes atendidas pela máquina i

T_{C_c} = número de partes atendidas pelas máquinas pertencentes ao *c-ésimo* cluster

$0 \leq f(c) \leq 1$

A LCR é construída obedecendo a seguinte regra: Sendo C o conjunto dos clusters, f_{min} e f_{max} respectivamente os valores mínimo e máximo da função *f(c)* $\forall$ c $\in$ C; estarão na LCR os seguintes candidatos t: $LCR = \{\text{t} \in C \mid \text{f(t)} \geq (f_{min} + (1\text{-}\beta)(f_{max}\text{-}f_{min}))\}$. A máquina é então associada à um dos candidatos escolhidos aleatoriamente na LCR. Para a próxima máquina, a LCR é reconstruida e a máquina associada à um cluster. Este processo se repete até que todas as máquinas estejam agrupadas. Terminado o agrupamento de máquinas, é feito o agrupamento das partes pela segunda etapa da fase de construção. Tal agrupamento é feito de maneira idêntica ao agrupamento de máquinas, salvo que no agrupamento de partes a função que representa o benefício de se agrupar uma parte i à um cluster c é a função *g(c)*, dada por:

$$g(c) = \frac{M_{iC_c}}{M_{C_c}} \tag{5.2}$$

onde:

M_{iC_c} = número de máquinas pertencentes ao cluster c que atendem a parte i

M_{C_c} = número de máquinas pertencentes ao cluster c

$0 \leq g(c) \leq 1$

Então, para o agrupamento de partes LCR = $\{\text{t} \in C \mid \text{g(t)} \geq (g_{min} + (1\text{-}\alpha)(g_{max}\text{-}$

g_{min}))}. Sendo m o número de máquinas e p o número de partes, o pseudocódigo da fase de construção é descrito pela figura 5.6.

1: **Procedimento** Construcao($n_{clusters}$)
2: contrua a lista de candidatos C contendo todos os clusters
3: **Para** i=1,...,m **faça**
4: **Se** $i \notin C_c \ \forall$ c = 1,...,$n_{clusters}$ **então**
5: calcule *f(c)* da máquina i em relação ao cluster c, $\forall$ c = 1,...,$n_{clusters}$
6: LCR = {c $\in C$ | f(c) $\geq$ (f_{min} + (1-β)(f_{max}-f_{min}))}
7: escolher aleatoriamente um candidato r da LCR
8: agrupar a máquina i ao cluster r
9: **Fim-Se**
10: **Fim-para**
11: calcule *g(c)* da parte i em relação ao cluster c, $\forall$ c=1,...,$n_{clusters}$
12: **Para** i=1,...,p **faça**
13: LCR = {c $\in C$ | $\geq$ (g_{min} + (1-α)(g_{max}-g_{min}))}
14: escolher aleatoriamente um candidato r da LCR
15: agrupar a parte i ao cluster r
16: **Fim-Para**
17: **Fim** Procedimento

Figura 5.6: Pseudocódigo do procedimento de construção do GRASP

A respeito da complexidade deste procedimento, têm-se: o cálculo da função *f(c)* (linha 5) é feito em $O(n_{clusters}mp)$, pois é feita a verificação das partes atendidas em comum entre uma máquina e todas as máquinas já agrupadas, para cada cluster. A construção da *LCR* e da lista C são feitas em $O(n_{clusters})$ (linhas 2 e 6) e as linhas 7 e 8 são feitas em $O(1)$. Como o cálculo de *f(c)* é feito para todas as máquinas que não são semente de clusters, a complexidade das linhas 3 a 10 é de $O(m^3p)$. O cálculo de *g(c)* é feito em $O(n_{clusters}m)$, ou O (m^2); pois para todos os clusters são verificadas quais máquinas pertencem aos mesmos. A linha 13 é $O(n_{clusters})$, e as linhas 14 e 15 $O(1)$. Portanto, a complexidade das linhas 12 a 16 é $O(m^2p)$. A complexidade do procedimento de construção é igual à complexidade das linhas 3 a 10, ou seja, $O(m^3p)$.

5.2.4 Busca Local

Busca Local 1: O primeiro modelo de busca local utilizado é idêntico à busca local utilizada no algoritmo evolutivo proposto neste trabalho e descrito no capítulo

4. Sobre os outros dois modelos, os veremos com detalhes.

Busca Local 2: O segundo modelo de busca realiza uma mudança nos componentes de uma solução inicial s. Primeiramente, para cada parte é calculado o índice P_{ik} em relação à cada cluster k, o qual é dado pela equação abaixo:

$$P_{ik} = \frac{m_{ik}}{M_k} - \frac{e_0}{M_i} \tag{5.3}$$

onde:

m_{ik} = número de máquinas pertencentes ao cluster k que atendem a parte i

M_k = número de máquinas pertencentes ao cluster k

e_0 = número de elementos inter-clusters (elementos iguais a 1 fora dos clusters) oriundos do agrupamento da parte i ao cluster k

M_i = número total de máquinas que atendem a parte i

$-1 \leq P_{ik} \leq 1$

Para cada parte é determinado o maior coeficiente. Desta forma, cada parte será agrupada ao cluster para o qual tem maior coeficiente P_{ik}. Caso este reagrupamento cause uma melhoria na função objetivo da solução, é executado o segundo passo da busca local. Caso não haja melhoria, as partes são agrupadas aos seus clusters originais e a busca é finalizada. O segundo passo deste segundo modelo de busca local obedece a mesma lógica do primeiro passo. Uma vez reagrupadas as partes, para cada máquina i é calculado o índice M_{ik} em relação à cada cluster k, dado pela equação:

$$M_{ik} = \frac{p_{ik}}{P_k} - \frac{e_0}{P_i} \tag{5.4}$$

onde:

p_{ik} = número de partes pertencentes ao cluster k que são atendidas pela máquina i

P_k = número de partes pertencentes ao cluster k

e_0 = número de elementos inter-clusters (elementos iguais a 1 fora dos clusters) oriundos do agrupamento da máquina i ao cluster k

P_i = número total de partes que são atendidas pela máquina i

$-1 \leq M_{ik} \leq 1$

Analogamente ao agrupamento das partes, é determinado o maior coeficiente para cada máquina. Então estas são reagrupadas aos clusters para os quais possuem o maior coeficiente. Caso isto não cause uma melhoria na função objetivo da solução, as máquinas são reagrupadas aos clusters originais e a busca é finalizada. Caso contrário, executa-se o passo 1 novamente. O pseudocódigo do segundo algoritmo de busca local é descrito na figura 5.7.

```
1: Procedimento Busca Local2
2:   melhora = 1
3:   Enquanto melhora==1 faça
4:       Para i=1,...,p faça
5:           Para k=1,...,n_clusters faça
6:               calcule P_ik
7:               atualizar max(P_ik)
8:           Fim-Para
9:       Fim-para
10:      reassociar as partes de acordo com max(P_ik), ∀ i = 1,...,p
11:      Se solução não melhorou então
12:          associar partes à clusters originais
13:          melhora = 0
14:      Fim-Se
15:      Se melhora==1 então
16:          Para i=1,...,m faça
17:              Para k=1,...,n_clusters faça
18:                  calcule M_ik
19:                  atualizar max(M_ik)
20:              Fim-Para
21:          Fim-Para
22:          reassociar as máquinas de acordo com max(M_ik), ∀ i = 1,...,m
23:          Se solução não melhorou então
24:              associar máquinas à clusters originais
25:              melhora = 0
26:          Fim-Se
27:      Fim-Se
28:  Fim-Enquanto
29: Fim Procedimento
```

Figura 5.7: Pseudocódigo do segundo procedimento de Busca Local

O cálculo de cada coeficiente P_{ik} é feito em $O(m)$, pois para cada cluster k são verificadas as associações de todas as máquinas. Como devem ser calculados os coeficientes de cada parte em relação à cada cluster, o cálculo de todos os coeficientes das partes é feito em $O(\mathrm{mp}n_{clusters})$. Como o número de clusters é da ordem do

número m de máquinas, a complexidade das linhas 4 a 9 é $O(m^2p)$. A linha 10 é $O(p)$ e o cálculo da aptidão do indivíduo é feita em $O(mp)$ (linhas 11 e 15). Assim, as linhas 11 a 14 tem complexidade $O(mp)$ (a 11 é $O(mp)$, a 12 é $O(p)$ e as demais $O(1)$). Analogamente ao cálculo dos coeficientes das partes, a complexidade das linhas 16 a 21 é $O(m^2p)$. A linha 22 é $O(m$, a 23 $O(mp)$, a 24 $O(m)$ e a 25 $O(1)$. Portanto, a complexidade das linhas 15 a 27 é $O(m^2p)$. Portanto, a complexidade do procedimento de busca local é $O(m^2p)$.

Para ilustrar esta busca local, considere uma solução de uma instância do PCM constituída de 4 máquinas, 8 partes e 2 clusters, representada pela figura 5.8.

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

Figura 5.8: Matriz-solução do PCM inicial para a busca local

A aptidão da solução incial é igual a $\frac{21-6}{21+1} = 0{,}6818$. O primeiro passo da busca local irá tentar realocar as partes entre os 2 clusters da solução. Os coeficientes P_{ik} das partes em relação aos clusters são mostrados na tabela 5.1.

P_{ik}	
P_{11} = **(2/2)-(0/2) = 1**	P_{12} = (0/2)-(2/2) = -1
P_{21} = **(2/2)-(0/2) = 1**	P_{22} = (0/2)-(2/2) = -1
P_{31} = **(2/2)-(0/2) = 1**	P_{32} = (0/2)-(2/2) = -1
P_{41} = (1/2)-(2/3) = -0,16	P_{42} = **(2/2)-(1/3) = 0,67**
P_{51} = (1/2)-(1/3) = 0,17	P_{52} = **(2/2)-(1/3) = 0,67**
P_{61} = (1/2)-(1/3) = 0,17	P_{62} = **(2/2)-(1/3) = 0,67**
P_{71} = (1/2)-(1/3) = 0,17	P_{72} = **(2/2)-(1/3) = 0,67**
P_{81} = (1/2)-(1/2) = 0,17	P_{82} = **(2/2)-(1/3) = 0,67**

Tabela 5.1: Coeficientes P_{ik} das partes após primeiro passo da busca local2

Os coeficientes em negrito indicam o maior coeficiente para cada parte. Observando a tabela 5.1, o novo conjunto de famílias de partes a serem formadas é o

seguinte: A família 1 contém as partes P1, P2 e P3 e a família 2 as restantes. Como houve melhoria na aptidão, a nova solução do PCM é a ilustrada na figura 5.9.

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

Figura 5.9: Matriz-solução do PCM depois da aplicação do primeiro passo da busca local2

A aptidão da nova solução é igual a $\frac{21-5}{21+0} = 0{,}7619$. Executa-se então o segundo passo da busca local, que procura realocar as máquinas aos clusters, com base na nova formação de família de partes. A tabela 5.2 mostra os coeficientes M_{ik} das partes em relação aos clusters:

M_{ik}	
M_{11} = **(3/3)-(2/5) = 0,6**	M_{12} = (2/5)-(3/5) = -0,2
M_{21} = **(3/3)-(3/6) = 0,5**	M_{22} = (3/5)-(3/6) = 0,10
M_{31} = (0/3)-(5/5) = -1	M_{32} = **(5/5)-(0/5) = 1**
M_{41} = (0/3)-(5/5) = -1	M_{42} = **(5/5)-(0/5) = 1**

Tabela 5.2: Coeficientes M_{ik} das máquinas em relação aos clusters no segundo passo da busca local2

Pela tabela 5.2, nota-se que as células de máquina não terão alteração em sua configuração, o que faz com que não haja melhoria na aptidão da solução. A busca local então é finalizada e a solução mostrada na figura 5.9 é a solução resultante da aplicação da busca local na solução inicial representada pela figura 5.8.

Busca Local 3: O terceiro modelo de busca local inicialmente faz uma avaliação da qualidade dos clusters de uma solução com base num coeficiente aqui chamado *qualidade do cluster* denotado por λ_k, cujo valor é dado pela seguinte equação:

$$\lambda_k = \frac{N1's_k}{A_k} - \frac{Nee_k}{N1's_k + Nee_k} \tag{5.5}$$

onde:

$N1's_k$ = número de elementos do cluster k preenchidos com o valor 1

A_k = área do cluster k

Nee_k = número de elementos inter-clusters ao longo das linhas e colunas do cluster k

-1 $\leq \lambda \leq$ 1

Após o cálculo do valor da *qualidade do cluster* para cada cluster k, são mantidos na solução apenas os clusters cujo valor de λ for maior ou igual a um valor dado como entrada para a busca local. As máquinas e partes não pertencentes aos clusters mantidos serão agrupadas a novos clusters a serem formados por um procedimento guloso. Tal procedimento constitui-se basicamente de três passos:

1. Primeiramente são formadas sementes de clusters até que o número de clusters seja igual a $n_{clusters}$, onde $n_{clusters}$ é o número de clusters da solução. As duas primeiras sementes são determinadas como sendo as duas máquinas com a menor similaridade de Jaccard. Cada nova semente será a máquina cujo maior valor de similaridade de Jaccard com as sementes já formadas seja o menor entre os maiores valores de similaridade de Jaccard entre as máquinas e as sementes já formadas.

2. Cada máquina restante será agrupada ao cluster para o qual possuir uma maior similaridade. Entende-se neste passo por similaridade entre uma máquina i e um cluster k como sendo o número de partes atendidas entre a máquina i e pelo menos uma máquina pertencente ao cluster k.

3. Cada parte será agrupada ao cluster que possuir o maior número de máquinas que a atende.

A seguir nas figuras 5.10, 5.11, 5.12 e 5.13 são mostrados os pseudocódigos dos algoritmos relacionados ao terceiro modelo de busca local. Primeiramente é apresentado o pseudocódigo que determina o valor de λ_k relacionado à cada cluster. Considere a variável binária x_{ij} que assume valor 1 se a máquina i atende a parte j e 0 caso contrário, o número de máquinas igual a m, o número de partes igual a p;

e a variável s como sendo o vetor que reprsenta a solução da qual os clusters serão classificados.

1: **Procedimento** Classifica clusters(s)
2: **Para** k=1,...,$n_{clusters}$ **faça**
3: calcule $nmaq_k = \sum_{i=1}^{m} \frac{s[i]}{k}$ | s[i]==k
4: calcule $npartes_k = \sum_{j=1}^{p} \frac{s[m+j]}{k}$ | s[m+j]==k
5: calcule $N1's_k = \sum_{i=1}^{m}\sum_{j=1}^{p} x_{ij}$ | s[i]==k, s[m+j]==k
6: calcule $Nee_k = \sum_{i=1}^{m}\sum_{j=1}^{p} x_{ij}$ | s[i]==k e s[m+j]≠k **ou** s[i]≠k e s[m+j]==k
7: $A_k = nmaq_k \quad * \quad npartes_k$
8: $\lambda_k = \frac{N1's_k}{A_k} - \frac{Nee_k}{N1's_k + Nee_k}$
9: **Fim-Para**
10: **Fim** Procedimento

Figura 5.10: Pseudocódigo do procedimento de construção do GRASP

A complexidade do procedimento *Classifica clusters* é $O(mp)$ (linhas 5 e 6). Após a classificação dos clusters de acordo com o valor de λ_k, são mantidos na solução aqueles que possuirem o valor de λ_k maior ou igual a um valor dado como entrada aqui denomina *valor de corte*, denotado por σ. Seja a variável *cópia* uma cópia da solução original, o pseudocódigo da figura 5.11 representa o algoritmo que mantém os melhores clusters numa solução s de acordo com o valores de λ_k e σ.

A linha 2 do procedimento da figura 5.11 realiza uma cópia da solução original em $O(m+p)$. As linhas 5 a 12 mantém na solução os clusters cujo valor de λ_k seja maior ou igual a σ, porém realizando uma nova rotulação dos clusters, começando de 1 e incrementando o rótulo em uma unidade a cada novo cluster a ser mantido. Por exemplo, se para uma dada solução apenas os clusters 5 e 7 forem mantidos, estes serão rotulados como os clusters 1 e 2, respectivamente. Desta forma pode-se determinar a quantidade de novos clusters a serem formados através da subtração $n_{clusters}$-rotulo. A complexidade das linhas 5 a 12 é $O(m+p)$. As linhas 13 a 19 desfazem os clusters cujo valor de λ_k é menor que σ, através da atribuição de um valor inválido para os componentes de máquinas e partes associados à estes clusters. Isto é feito em $O(m+p)$. A complexidade do procedimento *Mantem clusters* é portanto igual a complexidade das linhas 4 a 20, ou seja $O(n_{clusters}(m+p))$. Como o número de clusters é da ordem do número de máquinas m, a complexidade é $O(m^2+mp)$. Sendo o número de partes p sempre maior que m a complexidade é então $O(mp)$.

```
1: Procedimento Mantem clusters(σ,s,cópia)
2:   Copia solucao(s,cópia)
3:   rotulo = 1
4:   Para k=1,...,n_clusters faça
5:     Se λ_k ≥ σ
6:       Para i=1,...,m+p faça
7:         Se cópia[i]==k então
8:           s[i] = rotulo
9:         Fim-Se
10:      Fim-Para
11:      rotulo++
12:    Fim-Se
13:    Senão
14:      Para i=1,...,m+p faça
15:        Se copia[i]==k então
16:          s[i] = -1
17:        Fim-Se
18:      Fim-Para
19:    Fim-Senão
20:  Fim-Para
21: Fim Procedimento
```

Figura 5.11: Pseudocódigo do procedimento de manutenção de clusters

Com relação aos demais clusters a serem formados, primeiramente são escolhidas as máquinas que servirão como sementes para os mesmos. Após isto, as máquinas e partes restantes são agrupadas aos clusters mais apropriados. O pseudocódigo da figura 5.12 representa o algoritmo que determina as máquinas sementes dos demais clusters e associa cada máquina restante ao cluster mais apropriado. Considere as seguintes variáveis:

$Par_{escolhido}$ = par de máquinas que constituirão as duas primeiras sementes dos primeiros dois novos clusters

C_c = o *c-ésimo* cluster

$clusters_{formados}$ = número de clusters formados

Jac_{ik} = coeficiente de Jaccard entre duas máquinas i e k

Jac_{iC_c} = coeficiente de Jaccard entre uma máquina i e a máquina semente do *c-ésimo* cluster

sim_{min} = menor valor de $\max(Jac_{iC_c})$, $\forall$ i=1,...,m; c=1,...,$clusters_{formados}$, $\mid i \notin C_c$

$maquina_{escolhida}$ = máquina escolhida para formar a nova semente de um novo cluster

x_{zi} = variável binária que assume o valor 1 caso a máquina i atenda a parte z e 0 caso contrário

w_{zC_c} = variável binária que assume o valor 1 se existe alguma máquina $\in$ ao *c-ésimo* cluster que atende a parte z e 0 caso contrário

sim_{max} = similaridade máxima entre uma máquina i e um cluster c

1: **Procedimento** Clusteriza Maquinas(s,rotulo)
2: $Par_{escolhido}$ = min(Jac_{ik}) $\forall$ i,k=1,...m | $i \neq k$ e i,k $\notin C_c$, $\forall$ c=1,...rotulo-1
3: $C_{rotulo} = maquina_1 \in Par_{escolhido}$
4: $C_{rotulo+1} = maquina_2 \in Par_{escolhido}$
5: $clusters_{formados}$ = rotulo+1
6: **Enquanto** $clusters_{formados} < n_{clusters}$ **faça**
7: **Para** i=1,...,m **faça**
8: **Se** $i \notin C_c$ $\forall$ c=1,...,$clusters_{formados}$ **então**
9: calcule max(Jac_{iC_c}) $\forall$ c=1,...,$clusters_{formados}$
10: **Fim-Se**
11: **Se** sim_{min} é atualizada **então**
12: $maquina_{escolhida} = i$
13: **Fim-Se**
14: **Fim-Para**
15: $C_{clusters_{formados}+1} = maquina_{escolhida}$
16: s[$maquina_{escolhida}$] = $clusters_{formados} + 1$
17: $clusters_{formados}$++
18: **Fim-Enquanto**
19: **Para** i=1,...,m **faça**
20: **Se** $i \notin C_c$ $\forall$ c=1,...,$n_{clusters}$ **então**
21: sim_{max} = -∞
22: **Para** c=1,...,$n_{clusters}$ **faça**
23: calcule $Sim_{iC_c} = \sum_{z=1}^{p} x_{zi} w_{zC_c}$
24: **Se** sim_{max} é atualizada **então**
25: $cluster_{escolhido}$ = c
26: **Fim-Se**
27: **Fim-Para**
28: $C_{cluster_{escolhido}} = C_{cluster_{escolhido}} \cup i$
29: s[i] = $cluster_{escolhido}$
30: **Fim-Se**
31: **Fim-Para**
32: **Fim** Procedimento

Figura 5.12: Pseudocódigo do procedimento de criação dos demais clusters e agrupamento das máquinas

Sobre a complexidade do algoritmo representado pelo procedimento *Clusteriza Máquinas*, temos o seguinte a esclarecer: A linha 2 é $O(m^2)$, tendo em vista que

é preciso escolher dentre todos os pares de máquinas não agrupadas o de menor similaridade. As linhas 3 a 5 são $O(1)$. As linhas 8 a 10 são $O(n_{clusters})$, ou seja, $O(m)$ (utilizando-se da matriz triangular superior *MatSim*). Desta forma, as linhas 7 a 14 tem complexidade $O(m^2)$. As linhas 15 a 17 são $O(1)$. O trecho da linha 6 a 18 será executado ($n_{clusters}$-$clusters_{formados}$) vezes, que é $O(m)$. Portanto, as linhas 6 a 18 tem complexidade $O(m^3)$. Sobre o trecho da linha 19 a 31, temos o seguinte: a linha 23 é $O(pm)$, pois no pior caso deve ser verificado para cada máquina diferente da máquina i se esta pertence ao cluster em questão e se atende à parte em questão, para que seja calculado o valor da variável w_{zC_c}. Então, o trecho das linhas 22 a 27 é $O(m^2p)$. Desta forma, a complexidade do procedimento *Clusteriza Máquinas* é igual à complexidade do trecho que compreende as linhas 19 a 31, ou seja, $O(m^3p)$. O pseudocódigo da figura 5.13 se refere ao procedimento de clusterização das partes. A variável binária x_{zi} tem o mesmo significado relatado para o procedimento anterior e a variável sim_{max} representa a maior similaridade dentre as similaridades entre uma parte z e os clusters.

```
1: Procedimento Agrupa Partes
2:   Para z=1,...,p faça
3:     Para c=1,...,n_clusters faça
4:       calcule sim_zCc = Σ_{i=1}^{m} x_zi | máquina i ∈ C_c
5:       Se sim_max é atualizada então
6:         cluster_escolhido = c
7:       Fim-Se
8:     Fim-Para
9:     C_cluster_escolhido = C_cluster_escolhido ∪ z
10:  Fim-Para
11: Fim Procedimento
```

Figura 5.13: Pseudocódigo do procedimento de agrupamento de partes

Sobre sua sua complexidade, constata-se o seguinte: as linhas 3 a 8 são computadas em $O(n_{clusters}m)$ ou seja, $O(m^2)$. Como devem ser agrupadas as p partes, a complexidade do procedimento *Agrupa Partes* é igual à complexidade das linhas 2 a 10, ou seja, $O(m^2p)$. O pseudocódigo da figura 5.14 representa o algoritmo do terceiro modelo da Busca Local.

```
1: Procedimento Busca Local 3(s,λ,σ,cópia)
2:   Classifica clusters(s)
3:   Mantem clusters(σ,s,cópia)
4:     Se (n_clusters-rotulo) > 0
5:        Clusteriza Maquinas(s,rotulo)
6:        Agrupa Partes
7:     Fim-Se
8:     Senão
9:        Para i=1,...,m+p faça
10:          Se s[i]== -1 então
11:             s[i] = rotulo
12:          Fim-Se
13:       Fim-Para
14:    Fim-Senão
15: Fim Procedimento
```

Figura 5.14: Pseudocódigo do procedimento de agrupamento de partes

A complexidade do terceiro modelo de busca local é igual à complexidade do procedimento *Clusteriza Maquinas*, que é $O(m^3p)$. Vejamos um exemplo da aplicação do terceiro modelo de busca local. A figura 5.15 representa a clusterização de uma solução de entrada da busca local. O valor da função objetivo da solução

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

Figura 5.15: Matriz-solução do PCM inicial para o terceiro modelo de busca local

representada pela figura 5.15 é: $\frac{18-8}{18+5} = 0{,}4348$. O primeiro passo da busca local é calcular os valores de λ_k para cada cluster, que são: $\lambda_1 = \frac{3}{4} = 0{,}75$; $\lambda_2 = \frac{3}{6}\text{-}\frac{8}{11} = 0{,}23$ e $\lambda_3 = \frac{4}{6}\text{-}\frac{8}{12} = 0$. Assumindo um valor para $\sigma = 0{,}3$, os clusters 2 e 3 são destruídos, permanecendo apenas o cluster 1. Devem ser formados então dois novos clusters. O primeiro passo para definí-los é escolher duas máquinas semente. Os valores para a similaridade de Jaccard entre os pares de máquina candidatas a ser

uma máquina semente são mostrados na próxima tabela. De acordo com a tabela

Par de máquina i,k	Jac_{ik}
M2-M4	0,33
M2-M5	0,60
M2-M6	0,60
M4-M5	0,20
M4-M6	0,40
M5-M6	0,20

Tabela 5.3: Similaridades de Jaccard entre pares de máquina candidatas a máquinas semente

5.3, as máquinas semente escolhidas são as máquinas M4 e M5, pois é o primeiro par com menor similaridade de Jaccard (o par M5-M6 também tem similaridade de Jaccard igual a 0,20). O próximo passo é o agrupamento das máquinas restantes aos dois clusters em formação. A máquina M2 será agrupada ao cluster 3 com M5, pois M2 tem um maior número de partes atendidas em comum com as máquinas deste cluster (no caso, apenas a máquina M5). Analogamente, M6 será agrupada ao cluster 2 com M4. Finalmente cada parte será agrupada a um dos novos clusters que possuir o maior número de máquinas que a atende. A tabela 5.4 ilustra o número de máquinas dos clusters 2 e 3 que atendem as partes. As partes P3, P4 e P7

Parte	**Cluster 2**	**Cluster 3**
P3	M4 e M6	-
P4	M6	M2
P5	-	M2 e M5
P6	M6	M2 e M5
P7	M4 e M6	M2
P8	M4	M2 e M5

Tabela 5.4: Máquinas dos clusters 2 e 3 que atendem as partes

serão agrupadas ao cluster 2 (a parte P4 será agrupada ao cluster 2 porque ocorreu um empate entre os clusters 2 e 3 e o 2 é o cluster de menor rótulo). As partes P5, P6 e P8 serão agrupadas ao cluster 3. A figura 5.16 ilustra a solução obtida pela aplicação do terceiro modelo de busca local. O novo valor da função objetivo associada à solução é: $\frac{18-4}{18+2} = 0{,}70$.

Numa etapa seguinte do nosso trabalho, executamos experimentos com a metaheurística GRASP e com o Algoritmo Evolutivo aqui propostos para a resolução

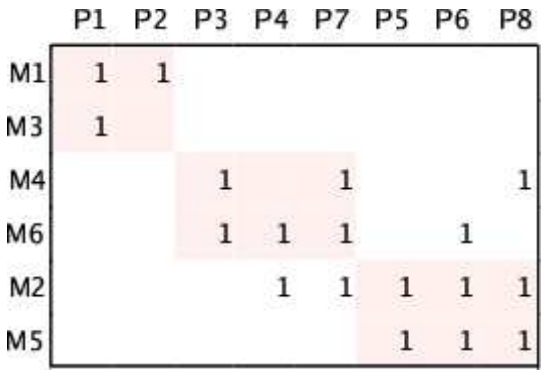


Figura 5.16: Matriz-solução após a aplicação da busca local 3

do PCM e comparamos seus resultados com os resultados da literatura como será visto no capítulo seguinte.

Capítulo 6

Implementações e resultados computacionais

Foram utilizadas 36 instâncias da literatura (tabela 6.1) disponíveis em [37] e [22] para a verificação da qualidade do AE. A tabela 6.1 lista as instâncias e seus tamanhos (nº de máquinas x nº de partes).

No que diz respeito ao número de clusters das soluções obtidas, as metaheurísticas aqui propostas não realizam uma análise desta característica das soluções do PCM, podendo surgir soluções com número de clusters diferentes mas com o mesmo valor de aptidão. Neste caso, as metaheurísticas aqui propostas armazenam a primeira melhor solução encontrada, não tendo mecanismos de análise para o melhor número de clusters caso surjam soluções com número de clusters diferentes e mesmo valor de aptidão. 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.

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

Tabela 6.1: Instâncias do PCM selecionadas da literatura

6.1 Resultados do AE

Para avaliar o algoritmo evolutivo proposto (AE), tomamos como parâmetro de comparação o algoritmo genético híbrido proposto recentemente (2002) por Resende e Gonçalves em [37] e aqui denotado por HGA, o qual segundo seus autores obteve os melhores resultados aproximados da literatura. Em relação ao HGA, utilizamos as instâncias disponibilizadas e os resultados por ele obtidos. 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. Além do HGA, os resultados foram comparados com os resultados obtidos em: Zodiac, proposto por Srinivasan e Narendan (1991); Grafics, proposto Srinivasan and Narendan (1991); algoritmo de clusterização proposto por Srinivasan em 1991 (Ca); algoritmo genético proposto por Cheng et. al em 1998

(Ga); algoritmo genético proposto por Joines et al. em [22], aqui denominado AG-JCK e implementado por nós neste trabalho. O número de gerações foi de 150 para o AE, o mesmo número utilizado em [37] para o AE. 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 em [22]. 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 algoritmos foi a *eficácia de agrupamento*. Os valores da *eficácia de agrupamento* estão multiplicados por 100, de maneira a apresentar os resultados em termos percentuais.

instância	Zodiac	Grafics	Ca	Ga	AG-JCK	HGA	AE	Melhora
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**	-8,08%
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	-0,10%
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**	1,20%
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%

Tabela 6.2: Resultados comparativos do AE com outros 6 algoritmos

Pelos resultados da tabela 6.2, onde os melhores resultados estão em negrito, podemos notar que o AE conseguiu melhores resultados isoladamente em 8 das 36 instâncias, obteve a mesma melhor solução da literatura em outras 26 e para as instâncias 21 e 2 obteve uma solução com pior aptidão que a da literatura. Porém

vale ressaltar que a solução encontrada para a instância 2 pelo algoritmo Ga, como relatado em [37], contêm clusters unitários. Desta forma, quando executamos nosso algoritmo para aceitar soluções com tais clusters para a instância 2 o AE alcança a mesma solução mostrada na tabela e tida como a melhor. O resultado da instância 21, no nosso entendimento, demonstra uma incoerência da medida de desempenho *Eficácia de Agrupamento*, pois nem sempre um maior valor desta significa necessariamente uma melhor clusterização. Observando a matriz de clusterização da referida instância no apêndice deste trabalho, entendemos que uma melhor clusterização poderia ser obtida se o segundo cluster fosse eliminado (o qual possui duas máquinas sem partes atendidas em comum) sendo a máquina 11 e a parte 32 agrupadas ao último cluster (pois a máquina 11 possui partes atendidas em comum com as demais máquinas deste cluster) e a máquina 4 e a parte 33 agrupadas ao segundo cluster (pois a máquina 11 possui partes atendidas em comum com as demais máquinas deste cluster). Mas desta forma o valor da *Eficácia de Agrupamento* seria igual a 76,14, ou seja, menor que o valor para a melhor solução. Isto porque a melhor solução possui 29 lacunas e 10 elementos intercluster, obtendo um valor igual a 76,22 para a *Eficácia de Agrupamento* ((135-10)/(135+29)). A solução sugerida aqui teria somente 1 elemento intercluster mas 41 lacunas, o que faria com que a *Eficácia de Agrupamento* tivesse valor igual a 76,14 ((135-1)/(135+41)). Entretanto a clusterização seria mais coesa de acordo com o nosso entendimento, pois as máquinas 11 e 4 seriam agrupadas com máquinas com as quais tem uma considerável similaridade. Tendo em vista esta particularidade da medida de desempenho *Eficácia de Agrupamento*, o seu uso implica que toda matriz-solução do PCM deve ser analisada pelo projetista do sistema de produção para que o mesmo possa decidir sobre mudanças na clusterização final.

Em termos de qualidade de solução o algoritmo proposto por nós foi ligeiramente melhor que os demais. A última coluna da tabela 6.2 mostra o desvio (diferença) percentual entre a solução obtida pelo nosso AE comparado com a melhor solução da literatura.

A tabela 6.3 apresenta os tempos de execução (em segundos) e a iteração que forneceu a melhor solução (msol) para o AG-JCK, HGA e AE. As colunas da ta-

bela representam, respectivamente: a identificação da instância, tempo do AG-JCK, HGA e AE em segundos e a iteração que forneceu a melhor solução (msol) para o AG-JCK, HGA e AE.

instância	Tempo AG-JCK	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	166,36	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**

Tabela 6.3: Tempos de execução (em segundos) e geração das melhores soluções para o AG-JCK, HGA e AE

Observando os resultados da tabela 6.3 podemos constatar que em termos do número de iterações, 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, onde o número de gerações foi igual ao HGA e AE, podemos notar que ele é o algoritmo que mais tardiamente alcança a melhor solução, em termos de geração. Entretanto, necessita de menos tempo devido ao fato de apenas possuir elementos básicos de um algoritmo genético (mutação, crossover e geração inicial aleatória). Porém, para as demais instâncias, que

são de maior dimensão, o AG-JCK necessita de um número maior de gerações e de mais tempo para alcançar suas melhores soluções em sua maioria inferiores às soluções do AE e HGA. A tabela 6.4 compara os resultados de execuções do AE utilizando o procedimento de construção na população inicial e sem o procedimento de construção. Para efeito de distinção das duas versões o AE com procedimento de construção será chamado *AE+C* e o sem o procedimento *AE*. A primeira coluna da tabela 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), media das melhores soluções, e geração da melhor solução para o AE e AE-C.

instância	AE+C	AE	Time AE+C	Time AE	Média AE+C	Média AE	Ger. AE+C	Ger. AE
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

Tabela 6.4: Resultados comparativos do AE com e sem procedimento de construção na geração inicial

A tabela 6.4 nos mostra que o uso do procedimento de construção de indivíduos na população inicial do algoritmo evolutivo conduz a uma melhor solução final em 6 das 36 instâncias, a uma mesma solução final em 28 e a uma solução ligeiramente in-

ferior em 2 instâncias; em relação ao algoritmo evolutivo sem o uso do procedimento de construção de indivíduos na população inicial. Além disso, para 17 instâncias a média da melhor solução do AE foi superior, foi igual em 18 instâncias e apenas em 2 instâncias a média foi ligeiramente menor. Constata-se portanto que o procedimento de construção auxilia a busca local na procura por melhores soluções.

Procurando analisar o impacto isolado do procedimento de construção, foram realizados experimentos computacionais com o AE com o procedimento de construção mas sem a busca local. A tabela 6.5 apresenta os resultados dos experimentos. As colunas representam respectivamente a identificação da instância, a melhor solução e a diferença percentual em relação à melhor solução encontrada pelo AE original. As colunas com um traço, significam que esta versão do AE 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 6.5 que a heurística de construção isoladamente não consegue resultados significativos e, além disso, gera muitos indivíduos inválidos. Entretanto, pela tabela 6.4 observa-se uma contribuição do procedimento de construção no resultado final do AE. Isto se deve ao fato de que, apesar de gerar muitos indivíduos inválidos, estes contêm clusterizações parciais baseadas em um critério de similaridade que auxilia a busca local pela busca das melhores soluções.

A próxima tabela mostra as melhores soluções obtidas pelo AE sem busca local e sem procedimento de construção e pelo AG-JCK com um número de gerações igual a 150. O objetivo foi analisar o desempenho do algoritmo proposto como um algoritmo genético tradicional. As colunas representam respectivamente a identificação da instância, melhor solução do AE, tempo de execução do AE, melhor solução do AG-JCK, tempo de execução do AG-JCK e percentual de melhoria em relação à melhor solução encontrada pelo AE original.

Claramente observa-se pelas tabelas 6.5 e 6.6 que o bom desempenho do AE é fortemente dependente do procedimento de busca local. Em nenhuma instância o AE conseguiu superar os resultados do AG-JCK. Além disso, o número de gerações (150) foi insuficiente para que os algoritmos alcançassem boas soluções, exceto para

instância	Solução	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	-	-

Tabela 6.5: Resultados do AE somente com heurística de construção e procedimento de cruzamento

algumas instâncias (1 a 7, 9 a 11 e 29).

Os resultados da tabela também apontam uma ineficiência do AE para gerar soluções válidas utilizando-se apenas do procedimento de cruzamento. Em 6 instâncias o AE não conseguiu gerar uma solução válida. Já os operadores do AG-JCK não conseguiram gerar indivíduos válidos para a instância 34. Isto sugere que algoritmos evolutivos rápidos e eficientes para o PCM necessitam de procedimentos especializados além de operadores tradicionais de um AG.

6.1.1 Análise Probabilística

Em outro conjunto de experimentos computacionais, foram verificadas as distribuições de probabilidade empíricas de alcance de um valor alvo em função do tempo

instância	Msol - AE	Tempo - AE	Msol - AG-JCK	Tempo - 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%

Tabela 6.6: Resultados comparativos do AE sem busca local e heurística de construção e AG-JCK

para 7 instâncias diferentes; para o AE e o AG-JCK. Foram determinados 3 níveis de alvo: alvos *fáceis*, *médios* e *difíceis*. Os alvos fáceis foram determinados como sendo as piores soluções entre as melhores soluções do AE e AG-JCK ou valores próximos à estes. Os alvos médios foram definidos como valores intermediários entre os valores dos alvos fáceis e difíceis e os alvos difíceis como sendo as melhores soluções entre as melhores soluções do AE e AG-JCK. Nestes experimentos, os dois algoritmos foram executados 100 vezes para cada instância e o tempo para alcance do alvo armazenado para todas execuções. Portanto, nesta bateria de testes o critério de parada dos algoritmos é: parar quando uma solução melhor ou igual à solução que fornece o valor alvo for encontrado. Os resultados do AE e AG-JCK para cada instância foram plotados associando-se ao i-ésimo menor tempo de execução t_i a probabilidade $p_i = \frac{i-0,5}{100}$, gerando pontos z_i no R^2 de coordenadas (t_i, p_i), para i de 1 até 100, analogamente à análise feita por Santos et al. em [41] e Aiex em [1]. Por

questões de tempo, foi estabelecido um limite máximo igual a 5 minutos para cada execução dos algoritmos. Para casos onde algum dos algoritmos não alcançou o alvo em todas as 100 execuções, a plotagem foi feita utilizando-se o número de execuções nas quais o algoritmo de pior desempenho alcançou o alvo.

Desta forma, cada ponto de uma curva indica a probabilidade de alcance do alvo em um determinado tempo de execução. As figuras 6.1 a 6.7 representam a distribuição de probabilidade empírica de alcance de alvos fáceis em função do tempo para o AE e o AG-JCK. Nota-se claramente que o AE é amplamente superior à performance do AG-JCK, sendo que as curvas que representam o AE são quase verticais, significando que os tempos para alcance dos alvos nas 100 execuções foram muito similares. Além disso os tempos do AE foram muito menores do que os tempos de alcance dos alvos para o AG-JCK, devido ao fato de que a curva do AE aparece muito mais à esquerda nos gráficos.

As figuras 6.8 a 6.14 representam a distribuição de probabilidade empírica de alcance de alvos médios em função do tempo para o AE e o AG-JCK e as figuras 6.15 a 6.18 a distribuição de probabilidade empírica de alcance de alvos difíceis. Novamente nota-se o desempenho superior do AE em relação ao AG-JCK. Vale salientar que a instância 36 também seria submetida à análise probabilística mas não foi possível fazê-lo porque em 98% das execuções do AG-JCK não se chegou nem ao menos ao alcance do alvo fácil. Isto faz sentido pois o AG-JCK alcança a melhor solução ou soluções regulares tardiamente, o que frequentemente fez com que o tempo das execuções superasse o tempo limite determinado de 5 minutos. Considerando os alvos difíceis, somente foi possível realizar a análise com 4 instâncias pois para as demais o AG-JCK não conseguiu alcançar os respectivos alvos dentro do tempo limite.

As razões para a diferença tão grande de performance entre o AE e o AG-JCK reside no fato de que a aplicação do procedimento de busca local no AE faz com que soluções de qualidade sejam alcançadas rapidamente nas primeiras iterações.

Por fim, salientamos que nesta análise não pudemos incluir o HGA proposto por Resende e Gonçalves [37] pelo fato de não termos acesso ao código do mesmo.

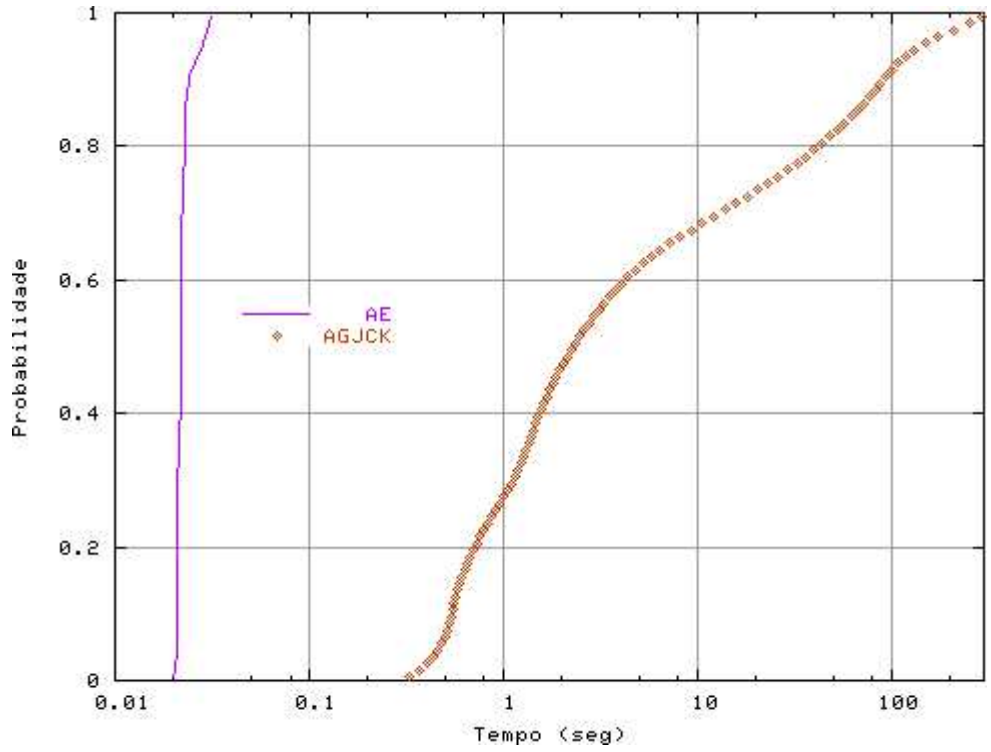


Figura 6.1: Instância: Askin and Subramanian - Alvo: 66,66

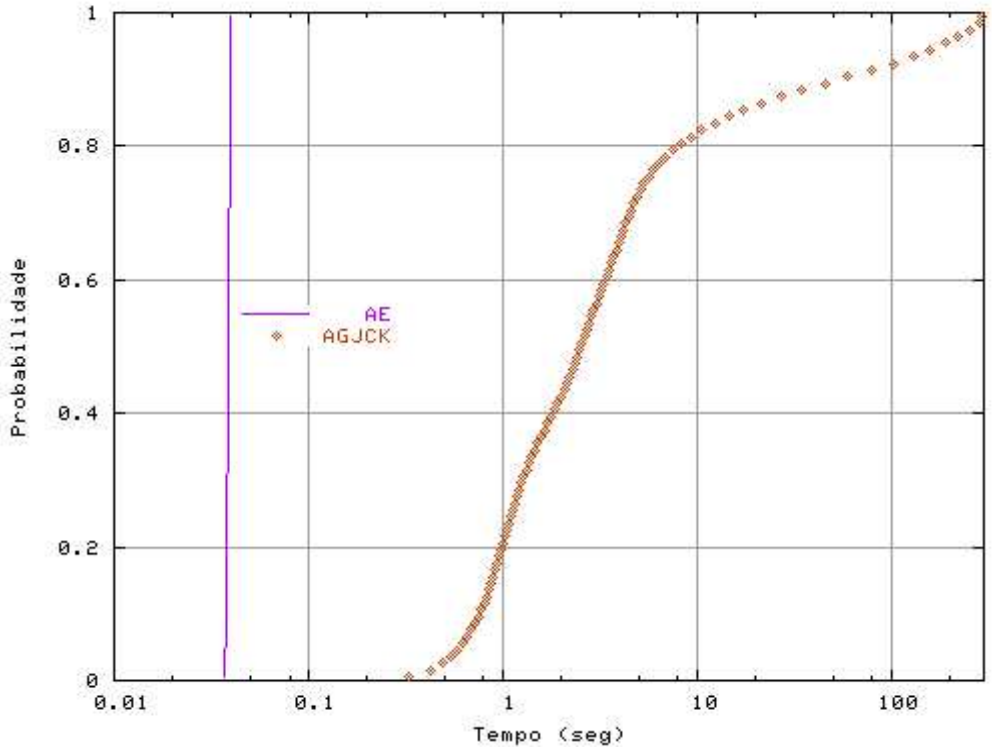


Figura 6.2: Instância: Srinivasan - Alvo: 56,81

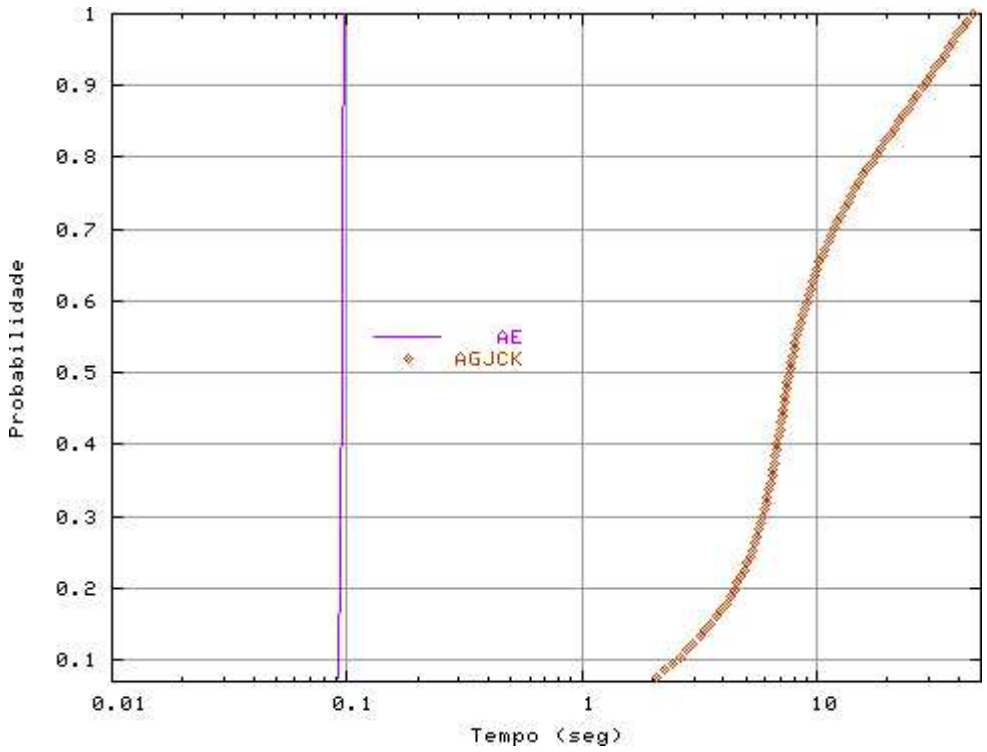


Figura 6.3: Instância: Chandrasekaran and Rajagopalan 24 x 40 (a) - Alvo: 85,00

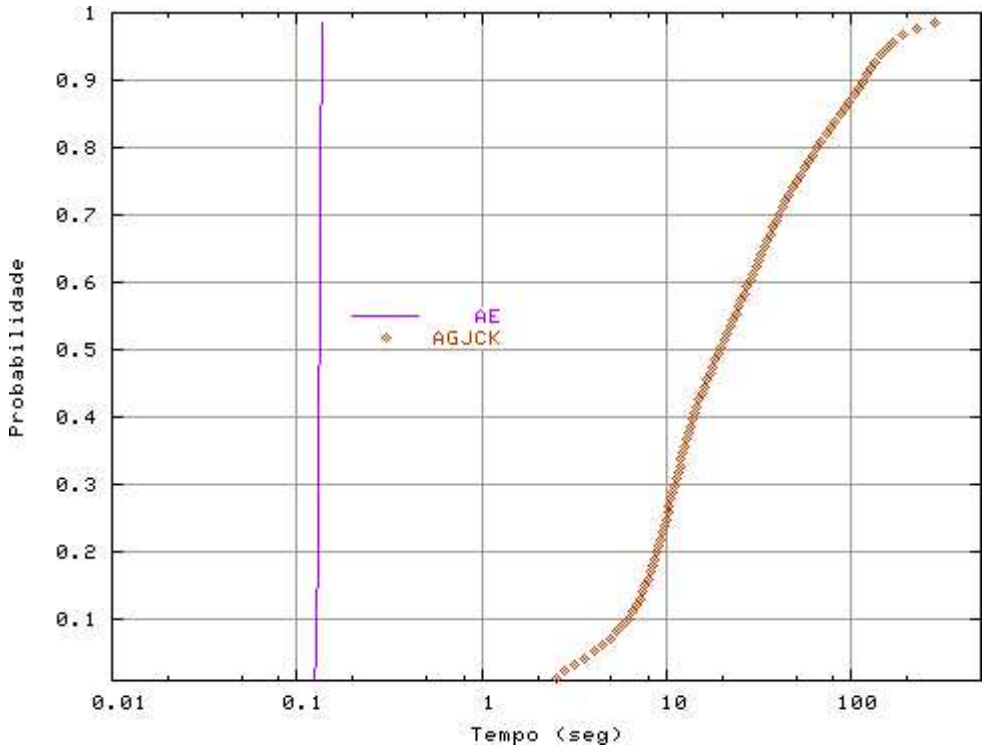


Figura 6.4: Instância: Chandrasekaran and Rajagopalan 24 x 40 (f) - Alvo: 39,00

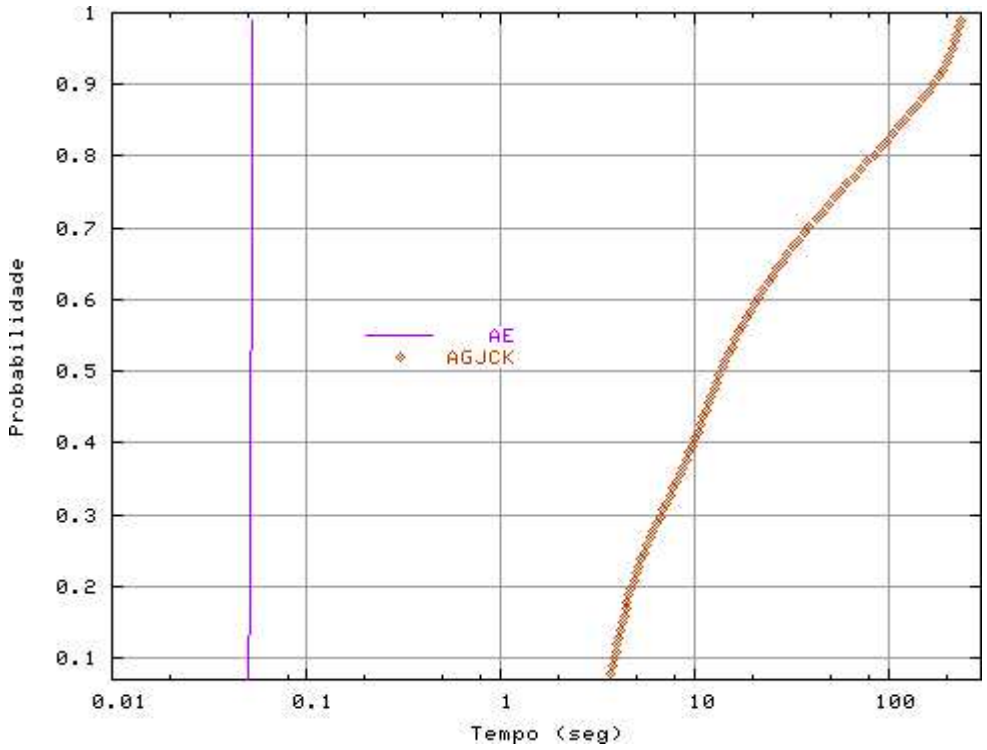


Figura 6.5: Instância: King - Alvo: 49,00

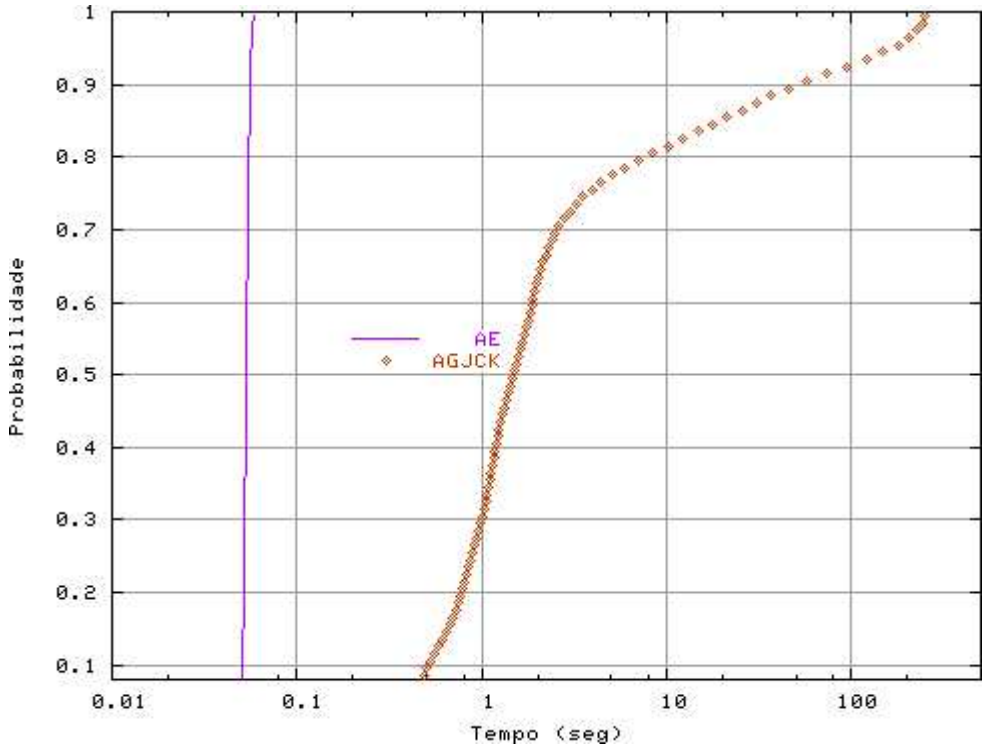


Figura 6.6: Instância: Kumar - Alvo: 40,00

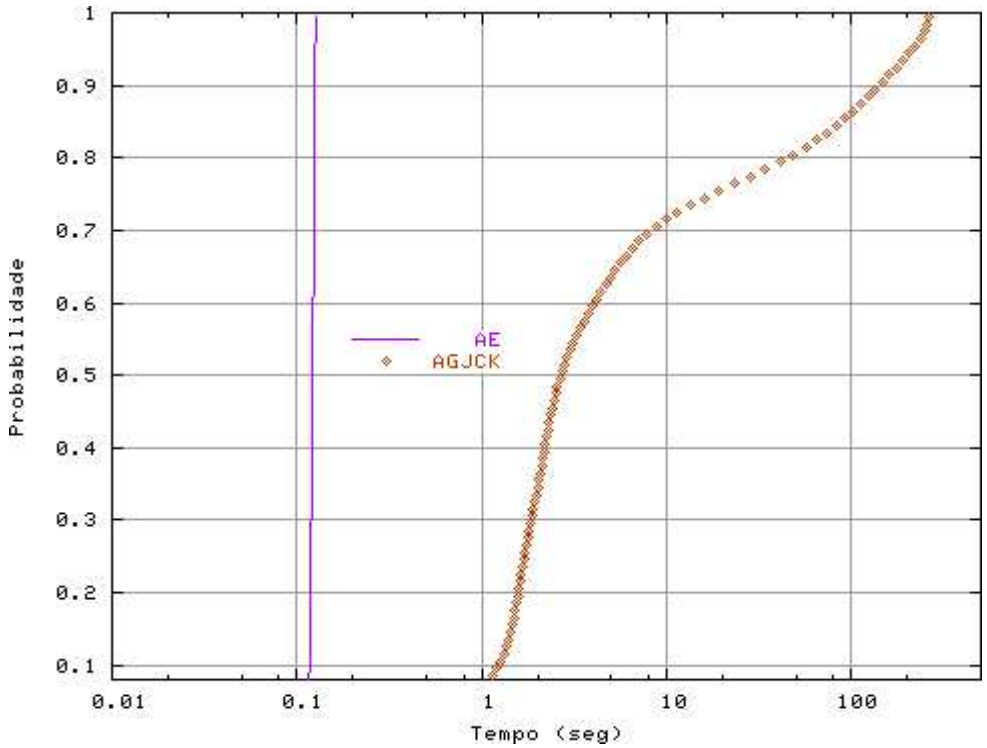


Figura 6.7: Instância: McComrick 27 x 27 - Alvo: 50,00

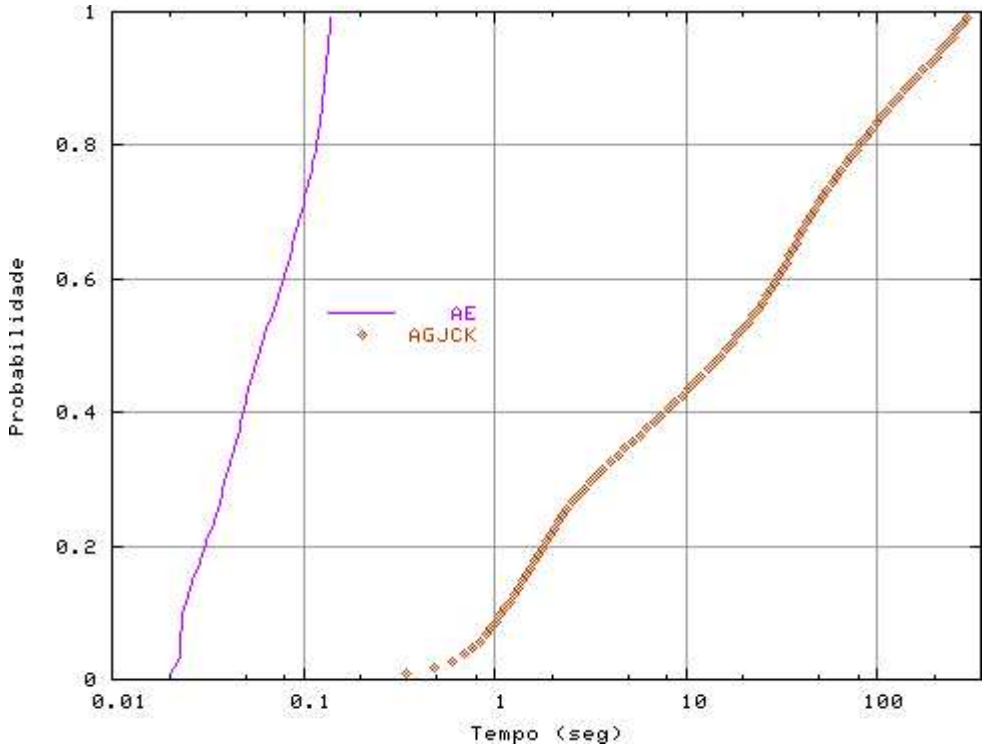


Figura 6.8: Instância: Askin and Subramanian - Alvo: 68,29

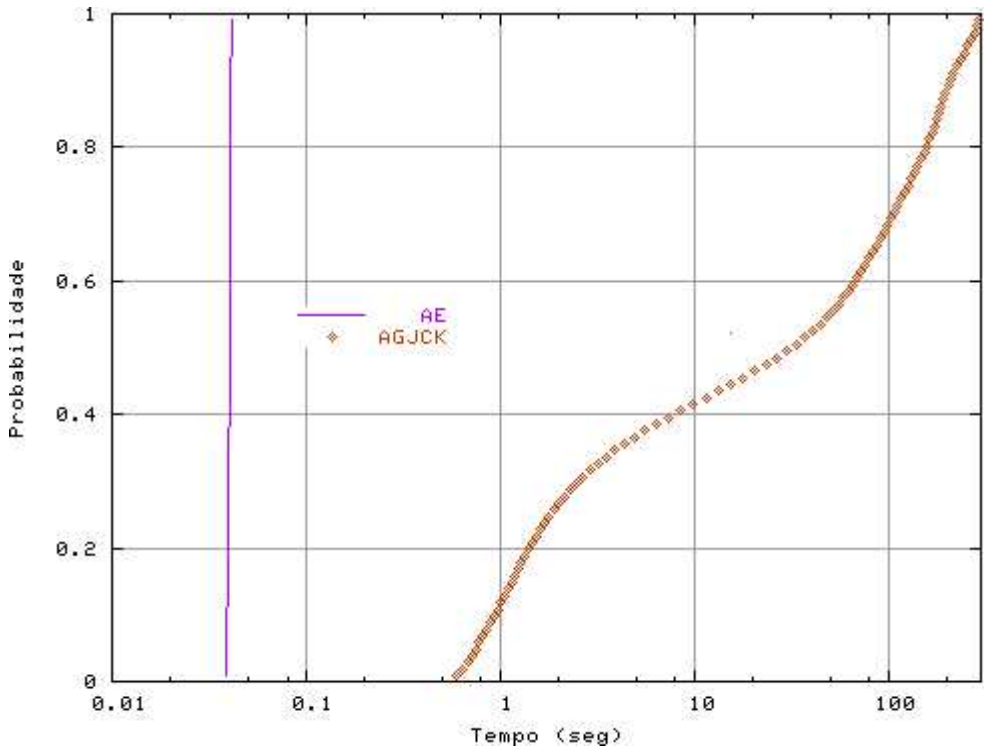


Figura 6.9: Instância: Srinivasan - Alvo: 62,32

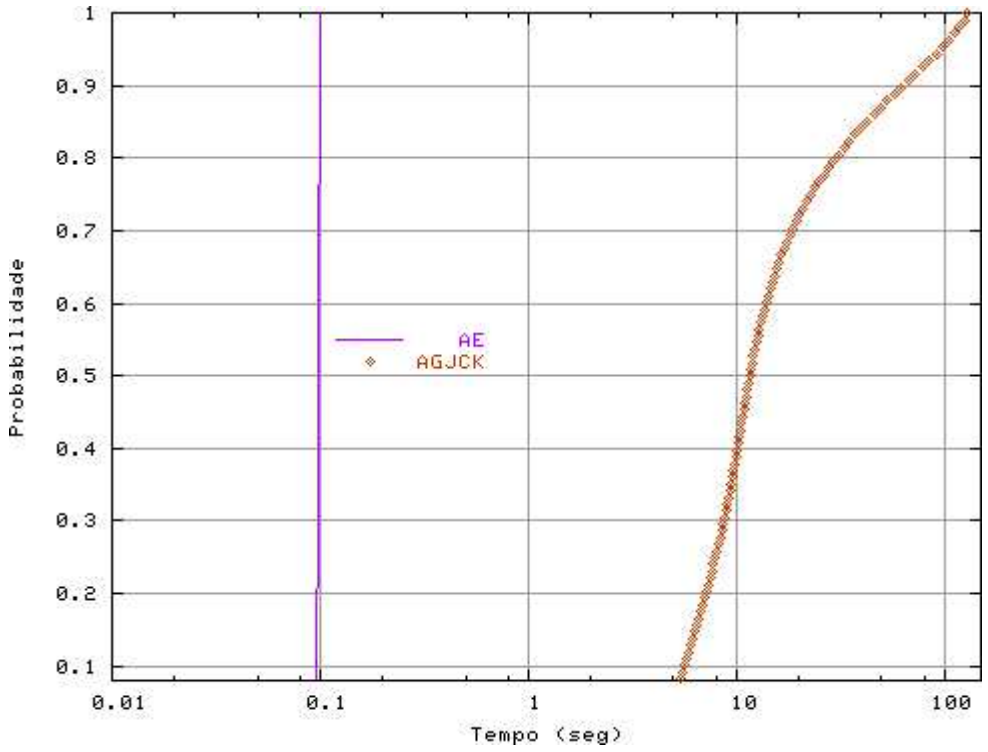


Figura 6.10: Instância: Chandrasekaran and Rajagopalan 24 x 40 (a) - Alvo: 92,50

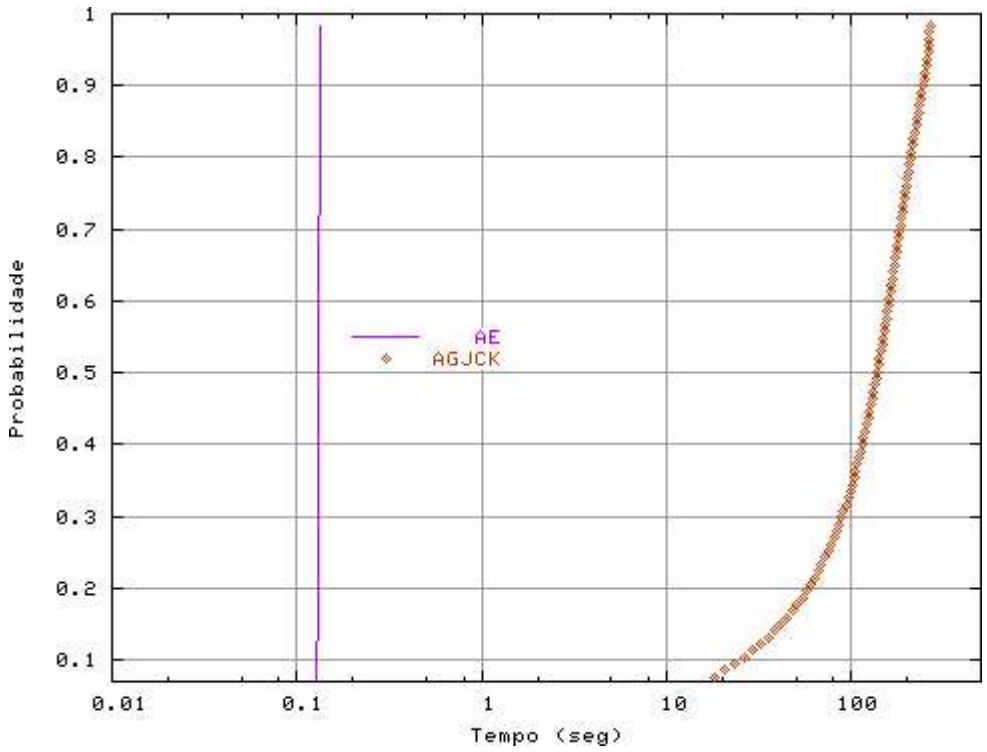


Figura 6.11: Instância: Chandrasekaran and Rajagopalan 24 x 40 (f) - Alvo: 41,90

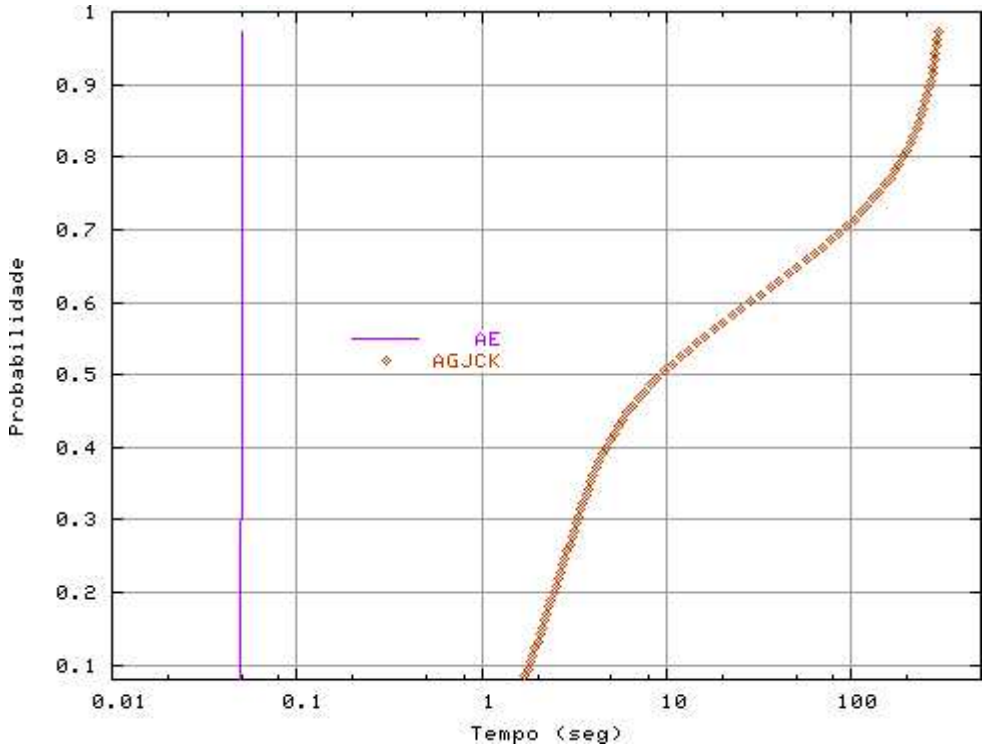


Figura 6.12: Instância: King - Alvo: 51,90

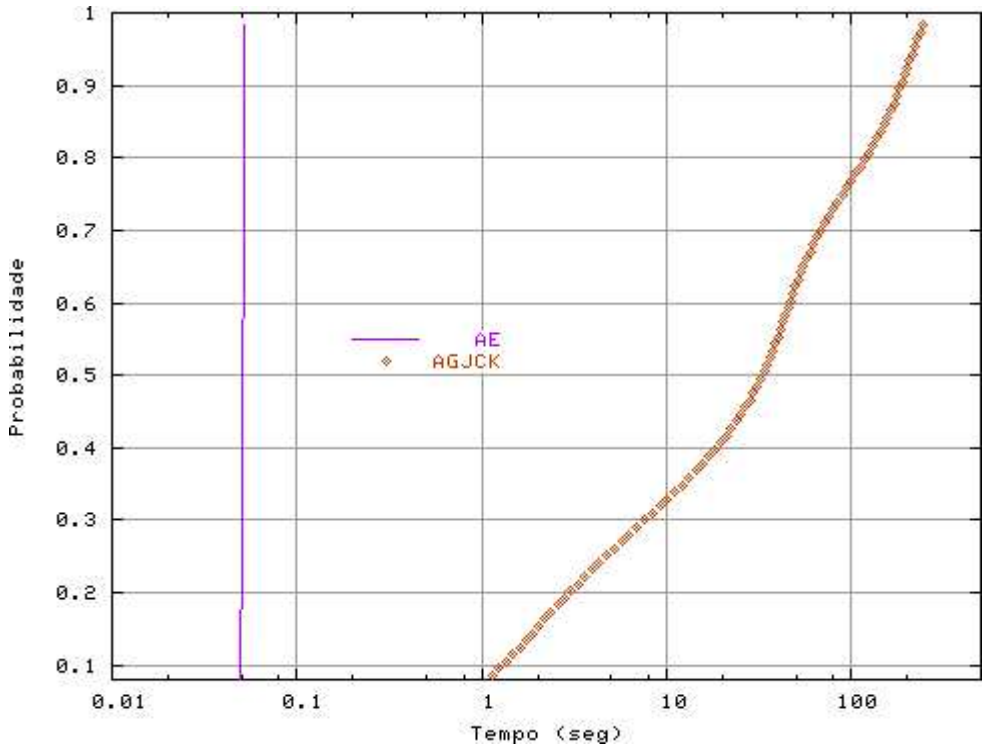


Figura 6.13: Instância: Kumar - Alvo: 44,82

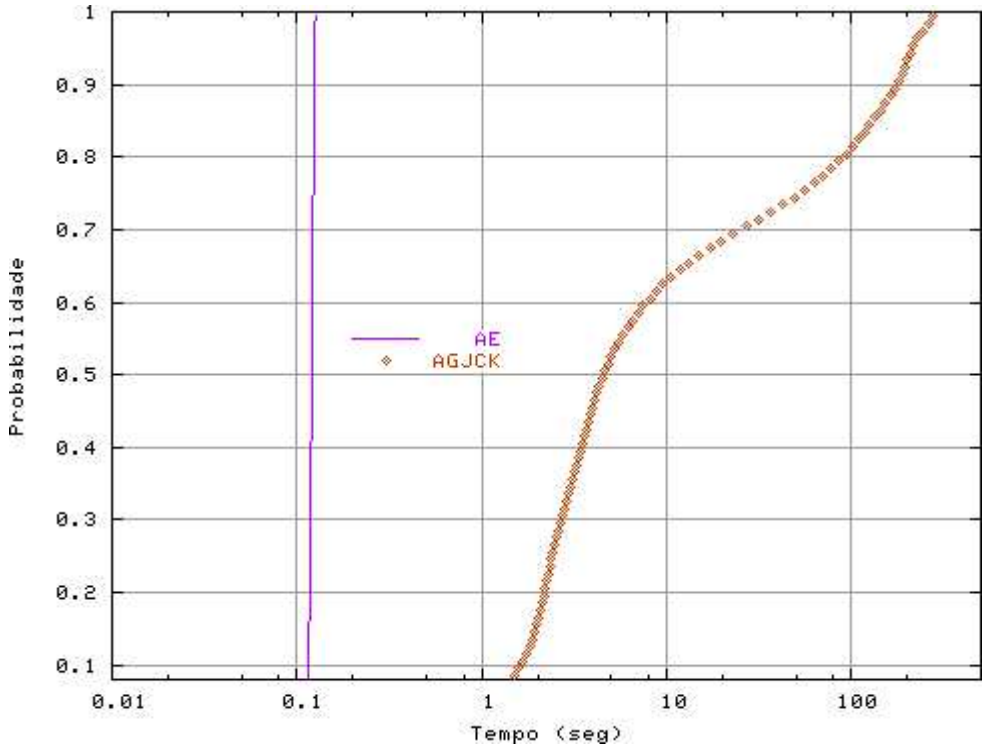


Figura 6.14: Instância: McCormick 27 x 27 - Alvo: 52,13

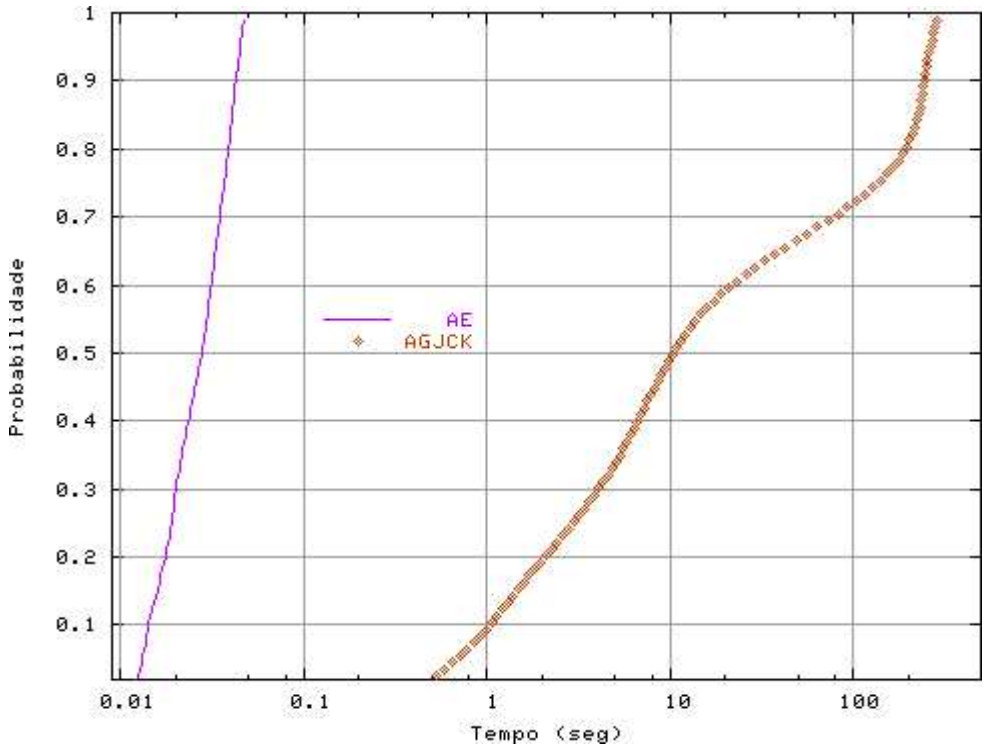


Figura 6.15: Instância: Askin and Subramanian - Alvo: 69,86

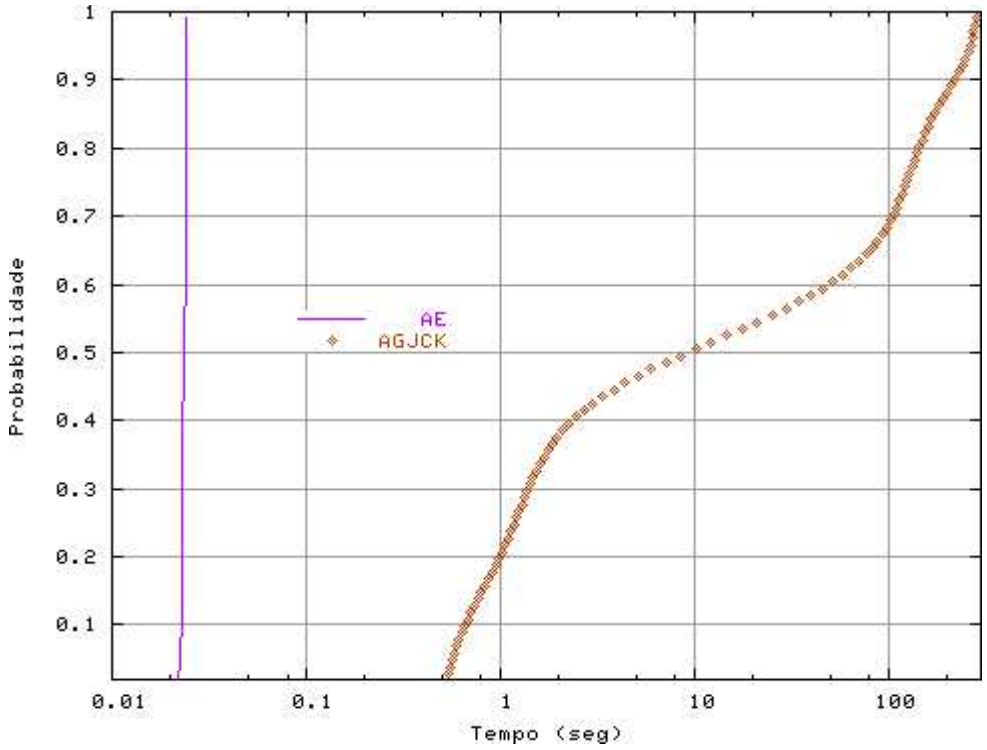


Figura 6.16: Instância: Srinivasan - Alvo: 67,83

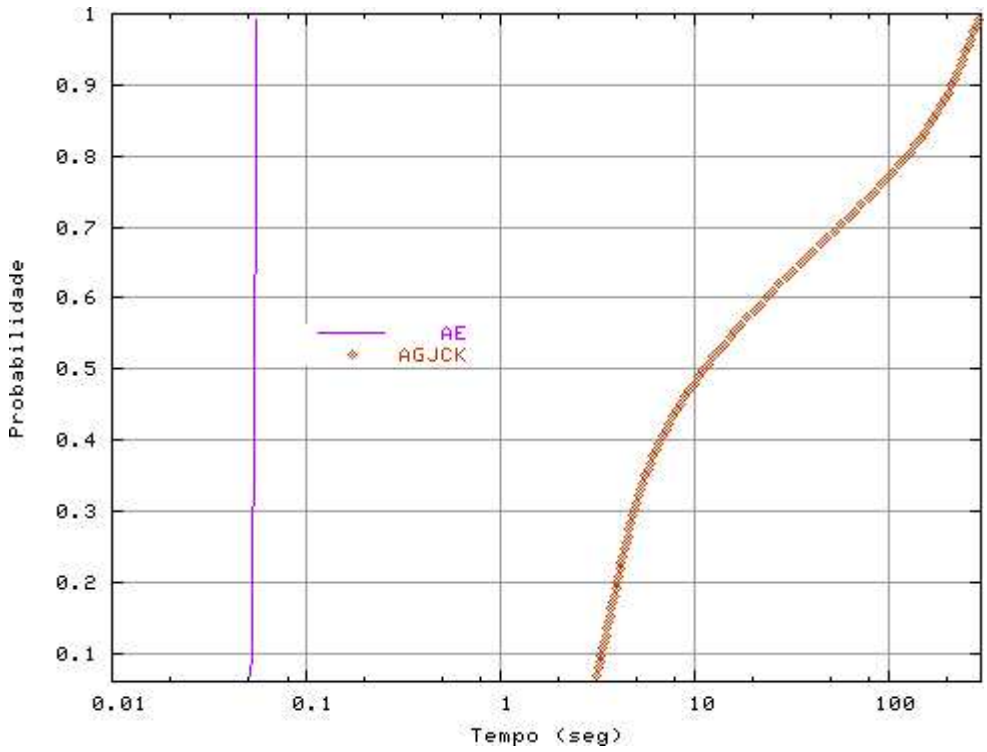


Figura 6.17: Instância: Chandrasekaran and Rajagopalan 24 x 40 (a) - Alvo: 100,00

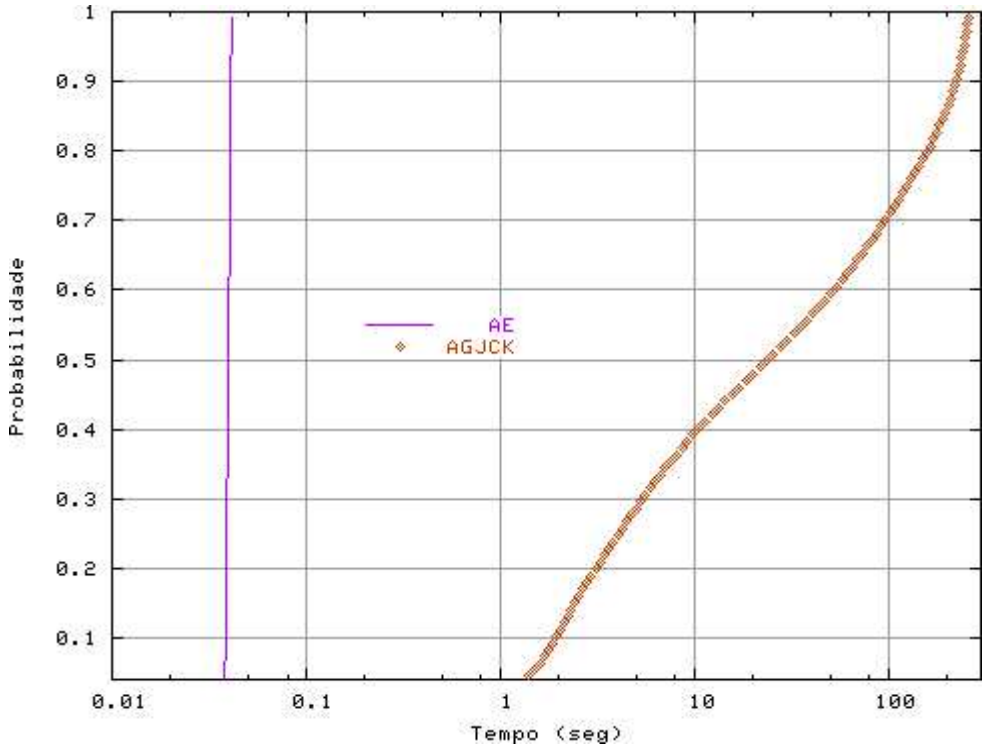


Figura 6.18: Instância: Kumar - Alvo: 54,86

6.2 Resultados da metaheurística GRASP

A metaheurística GRASP foi executada para o mesmo conjunto de instâncias utilizadas nos experimentos computacionais do AE e num sistema computacional com os mesmos recursos daquele utilizado pelo AE. Foram elaboradas 09 versões diferentes da metaheurística GRASP, sendo que cada versão foi executada 10 vezes para cada instância. Destas 09 versões, 5 versões utilizaram filtro, sobre o qual temos o seguinte a esclarecer: Inicialmente são geradas pela fase de construção um número determinado de soluções para todo par de valor (β, α) durante um certo intervalo ou *ciclo* de iterações GRASP. Então é formado um conjunto elite, contendo as melhores soluções obtidas e sobre este conjunto elite os procedimentos de busca local são aplicados. Após isto, o número de valores (β, α) que serão usados num novo *ciclo* de iterações é reduzido, sendo que os pares (β, α) a serem usados são definido através de uma verificação dos pares (β, α) associados às soluções elite.

Então o conjunto elite é eliminado e se inicia uma novo *ciclo* de iterações, agora com um conjunto reduzido de pares (β, α). Essa redução de acontece continuamente até que se chegue ao número máximo de iterações GRASP. A idéia é que nas primeiras iterações o algoritmo explore um espaço maior do universo de soluções através do uso de todos os pares de valores (β, α) e que faça uma seleção das regiões promissoras do espaço de busca ao longo de sua execução através do uso dos valores (β, α) associados às soluções elite de cada ciclo de iterações.

O critério de parada para todas as versões foi o número de iterações igual a 100. As versões são as seguintes:

1. **GRASP1**: Grasp com filtro. São geradas 20 soluções para cada par de valores (β,α) durante uma iteração. Os ciclos de iteração e os pares de valores (β,α) usados são os seguintes: **Ciclo 1**: iterações 1 a 10, usando todos pares formados pelas combinações dos valores (β, α), sendo que os valores máximo e mínimo para os mesmos são de 0.3 e 0.9, respectivamente. **Ciclo 2**: iterações 11 a 50, usando os 20 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α)

associados às soluções do conjunto elite caso sejam menos que 20. **Ciclo 3**: iterações 51 a 80, usando os 10 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 10. **Ciclo 4**: iterações 81 a 95, usando os 5 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 5. **Ciclo 5**: iterações 96 a 100, usando os 2 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 2. Ao final de todo ciclo de iterações os 3 procedimentos de busca local são aplicados alternadamente sobre as soluções elite.

2. **GRASP2**: Grasp com filtro. São geradas 20 soluções para cada par de valores (β,α) durante uma iteração. Os ciclos de iteração e os pares de valores (β,α) usados são os seguintes: **Ciclo 1**: iterações 1 a 10, usando todos pares formados pelas combinações dos valores (β, α), sendo que os valores máximo e mínimo para os mesmos são de 0.3 e 0.9, respectivamente. **Ciclo 2**: iterações 11 a 60, usando os 15 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 15. **Ciclo 3**: iterações 61 a 90, usando os 5 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 5. **Ciclo 4**: iterações 91 a 100, usando os 2 melhores pares de valores (β,α) associados ao conjunto de soluções elite obtido pelo ciclo anterior ou todos os pares de valores (β,α) associados às soluções do conjunto elite caso sejam menos que 2. Ao final de todo ciclo de iterações o processo de busca local é idêntico ao apresentado na versão GRASP1.

3. **GRASP3**: Grasp com filtro cujos parâmetros são os mesmos utilizados na versão GRASP2, mas somente o primeiro modelo de busca local é aplicado nas soluções do conjunto elite.

4. **GRASP4**: Grasp com filtro cujos parâmetros são os mesmos utilizados na versão GRASP2, mas somente o segundo modelo de busca local é aplicado nas soluções do conjunto elite.

5. **GRASP5**: Grasp com filtro cujos parâmetros são os mesmos utilizados na versão GRASP2, mas somente o terceiro modelo de busca local é aplicado nas soluções do conjunto elite.

6. **GRASP6**: Grasp sem filtro. A cada iteração são geradas somente 20 soluções utilizando valores (β,α) escolhidos randomicamente e toda a solução é submetida ao primeiro modelo de busca local. Não é gerado conjunto de soluções elite.

7. **GRASP7**: Grasp sem filtro. A cada iteração são geradas somente 20 soluções utilizando valores (β,α) escolhidos randomicamente e toda a solução é submetida ao segundo modelo de busca local. Não é gerado conjunto de soluções elite.

8. **GRASP8**: Grasp sem filtro. A cada iteração são geradas somente 20 soluções utilizando valores (β,α) escolhidos randomicamente e toda a solução é submetida ao terceiro modelo de busca local. Não é gerado conjunto de soluções elite.

9. **GRASP9**: Grasp sem filtro. A cada iteração são geradas somente 20 soluções utilizando valores (β,α) escolhidos randomicamente e não há aplicação de nenhum procedimento de busca local.

A tabela 6.7 apresenta em cada coluna as melhores soluções obtidas por cada versão da metaheurística GRASP, sendo que os titulos das colunas tem o seguinte significado: I - instâncias, G1 - GRASP1, G2 - GRASP2, G3 - GRASP3, G4 - GRASP4, G5 - GRASP5, G6 - GRASP6, G7 - GRASP7, G8 - GRASP8 e G9 - GRASP9. Os valores em negrito representam a melhor solução alcançada dentre todas as versões GRASP e a última coluna representa o percentual de melhoria em relação à melhor solução da literatura (já considerando os melhores resultados do AE aqui proposto).

I	G1	G2	G3	G4	G5	G6	G7	G8	G9	Melhora
1	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	0%
2	**60,87**	**60,87**	**60,87**	**60,87**	**60,87**	**60,87**	**60,87**	**60,87**	**60,87**	-10,48%
3	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	0%
4	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	0%
5	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	0%
6	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	0%
7	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	0%
8	**58,72**	58,33	58,33	54,74	54,74	**58,72**	58,33	54,74	54,74	0%
9	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	0%
10	**70,59**	**70,59**	**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**	**92,00**	**92,00**	**92,00**	0%
12	**67,65**	**67,65**	**67,65**	**67,65**	**67,65**	**67,65**	**67,65**	**67,65**	**67,65**	-3,16%
13	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	0%
14	48,72	49,54	49,60	48,72	48,72	**51,96**	50,00	48,18	48,21	0%
15	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	0%
16	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	0%
17	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	0%
18	51,79	51,35	52,21	51,64	51,35	**54,46**	51,33	51,18	50,87	0%
19	42,74	42,74	42,54	42,76	41,22	**42,96**	**42,96**	**41,35**	40,67	0%
20	**49,65**	**49,65**	**49,65**	**49,65**	47,89	**49,65**	**49,65**	46,81	46,53	0%
21	76,14	76,14	76,14	76,14	76,14	**76,22**	76,14	76,14	76,14	0%
22	**58,06**	**58,06**	**58,06**	57,47	57,71	**58,06**	57,84	56,68	56,90	0%
23	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	0%
24	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	0%
25	**73,51**	**73,51**	**73,51**	**73,51**	72,30	**73,51**	**73,51**	72,30	72,30	0%
26	51,66	51,50	51,66	51,50	47,59	**51,97**	51,59	46,34	49,03	0%
27	46,15	46,34	46,75	46,15	45,34	**47,06**	46,15	44,08	44,87	0%
28	**44,87**	**44,87**	**44,87**	44,59	44,59	**44,87**	44,58	44,00	44,00	0%
29	**54,27**	**54,27**	**54,27**	51,67	50,38	**54,27**	52,57	48,26	48,84	0%
30	40,00	40,56	40,66	39,84	40,91	**43,40**	43,20	38,72	38,57	-2,73%
31	57,89	57,89	57,89	57,89	57,89	**58,48**	57,89	57,31	57,89	0,58%
32	55,62	55,87	58,24	55,80	55,80	**59,66**	57,75	54,14	55,00	0%
33	**50,51**	**50,51**	**50,51**	**50,51**	50,25	**50,51**	**50,51**	48,74	48,74	0%
34	36,86	36,00	35,57	35,76	36,17	37,27	**37,76**	33,89	34,52	-11,59%
35	56,42	56,42	56,42	56,36	54,37	**56,42**	56,36	54,09	51,74	-1,95%
36	**84,03**	**84,03**	**84,03**	**84,03**	**84,03**	**84,03**	**84,03**	**84,03**	**84,03**	0%

Tabela 6.7: Resultados para as 9 versões da metaheurística GRASP implementadas

Analisando as tabelas 6.7 e 6.8 podemos constatar que as versões da metaheurística GRASP com filtro (versões G1 a G5) não trazem vantagens com relação às outras versões, tendo em vista que nenhuma destas versões foi a que chegou mais frequentemente às melhores soluções e além disso seu tempo de execução é bem maior que o das outras versões (salvo uma exceção, a instância 35), devido ao fato de serem geradas mais soluções por iteração.

Como nas versões com filtro a busca local é aplicada somente sobre o conjunto de soluções elite gerado pela fase de construção, observa-se que a aplicação da busca local sobre tais soluções é ineficaz pois a versão que usa somente a fase de construção (G9) por si só consegue obter os melhores resultados em 17 instâncias (instâncias 1 a 7, 9 a 13, 15 a 17, 23, 24 e 36) e para as instâncias em que G9 não conseguiu obter a melhor solução (19 instâncias), as versões com filtro conseguem obter a melhor solução em apenas 6 (instâncias 8, 22, 25, 28, 29 e 33). Portanto a busca local dever

I	G1	G2	G3	G4	G5	G6	G7	G8	G9
1	0,32	0,34	0,32	0,32	0,32	0,07	0,06	0,07	**0,04**
2	0,40	0,49	0,43	0,47	0,43	0,08	0,08	0,09	**0,04**
3	1,54	1,30	1,28	1,30	1,29	0,19	0,36	0,15	**0,09**
4	0,96	0,84	0,84	0,85	0,84	0,10	0,10	0,10	**0,06**
5	1,68	1,41	1,42	1,42	1,44	0,17	0,18	0,13	**0,10**
6	1,14	1,06	1,16	1,16	1,18	0,15	0,16	0,12	**0,08**
7	1,93	1,61	1,64	1,65	1,67	0,21	0,22	0,17	**0,11**
8	3,75	3,16	3,14	3,18	3,20	0,44	0,59	1,33	**0,22**
9	3,13	2,68	2,69	2,72	2,69	0,32	0,32	0,26	**0,19**
10	3,62	3,17	3,05	3,11	3,08	0,36	0,41	0,24	**0,22**
11	2,61	2,73	2,96	2,88	2,99	0,42	0,66	**0,25**	**0,25**
12	9,15	7,64	7,70	7,90	7,77	1,07	1,19	0,69	**0,55**
13	9,75	8,21	8,12	8,23	8,27	1,14	1,22	0,73	**0,58**
14	16,62	15,61	14,45	15,98	16,41	2,39	3,43	2,24	**1,44**
15	15,22	12,62	12,86	13,08	12,85	2,19	2,52	1,70	**0,94**
16	23,62	20,77	20,73	21,06	20,87	3,04	4,25	2,30	**1,53**
17	27,41	23,15	22,66	23,37	23,59	3,52	7,27	2,47	**1,59**
18	53,27	47,28	45,13	50,43	44,86	6,09	4,17	13,32	**4,68**
19	40,18	34,27	33,76	33,68	35,49	3,59	4,62	5,13	**2,98**
20	47,07	39,49	39,14	38,52	38,99	4,30	4,98	5,53	**3,20**
21	26,63	22,43	22,48	22,88	23,26	3,16	4,51	2,27	**1,59**
22	31,66	26,52	26,89	26,78	26,49	4,07	5,72	5,41	**2,09**
23	54,51	45,85	46,44	46,29	46,40	3,48	2,95	**2,56**	3,15
24	54,96	46,79	47,41	46,83	47,68	4,64	13,29	**3,03**	3,45
25	65,86	57,18	56,66	55,90	56,58	6,39	7,35	7,48	**4,32**
26	62,65	60,51	60,35	57,11	60,20	9,22	10,34	12,62	**5,53**
27	48,36	42,61	46,70	45,07	41,87	7,15	9,33	9,83	**4,92**
28	32,12	34,62	36,51	34,83	37,39	5,55	9,55	7,55	**3,75**
29	116,39	102,12	99,41	102,83	97,77	8,83	13,09	32,01	**8,60**
30	471,01	494,10	479,28	481,90	458,72	113,38	52,64	107,97	**65,22**
31	177,41	179,61	172,22	176,90	170,14	28,99	21,26	26,78	**16,82**
32	124,42	113,30	118,73	114,50	114,45	16,78	18,88	39,24	**10,36**
33	101,80	78,23	91,13	101,37	98,87	15,53	19,74	17,78	**11,10**
34	608,20	615,56	622,07	619,64	595,25	115,80	104,63	166,43	**91,06**
35	1093,31	879,76	904,50	889,23	795,84	71,37	64,85	4876,16	**58,51**
36	357,98	306,24	302,74	301,56	299,33	25,69	69,43	20,89	**20,89**

Tabela 6.8: Tempos de execução das melhores soluções para as 9 versões da metaheurística GRASP implementadas

ser aplicado à boa parte ou todas as soluções geradas pela fase de construção, mas nas versões com filtro isto torna-se inadequado devido ao grande número de soluções geradas.

Um outro fator que faz com que a construção de muitas soluções não tenham obtido uma melhoria no resultado final para as versões com filtro é que independentemente do parâmetro β usado, o processo de clusterização das máquinas tende a adquirir um caráter guloso a cada clusterização de uma nova máquina ou ainda pode não refletir claramente a similaridade entre uma máquina e um cluster. Vejamos um exemplo: Considere a figura 6.19 como a matriz de entrada de uma instância do PCM. Primeiramente a fase de construção determina as duas primeiras sementes dos dois primeiros clusters, aleatoriamente. Suponhamos que sejam as máquinas M1 e M2. Portanto, os clusters C1 e C2 contém respectivamente as máquinas M1 e

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

Figura 6.19: Matriz de entrada para a fase de construção

M2. Considerando o número de clusters igual a 3, a próxima máquina semente será a máquina menos similar aos clusters já formados, de acordo com a similaridade de *Jaccard*. Portanto, a máquina escolhida é a máquina M4. Desta forma, o cluster C3 contém a máquina M4. Para as máquinas restantes será construída uma lista restrita de clusters candidatos (LCR) de acordo com a similaridade entre a máquina e o cluster dada pela equação 5.1 mostrada no item 5.2.3 do capítulo 5. O tamanho da LCR será controlado pelo parâmetro de entrada β da fase de construção, que suponhamos que seja igual a 0,5. As máquinas restantes são clusterizadas uma a uma, por ordem crescente. Isto posto, a próxima máquina a ser clusterizada é a máquina M3. Os coeficientes de similaridade de M3 com os clusters são: M3-C1 = 0,67, M3-C2 = 0,2 e M3-C3 = 0,0. Os clusters da LCR para M3 serão aqueles que possuirem coeficiente maior ou igual a: (0 + (0,5 * 0,67)) = 0,335. Desta forma, somente o cluster 1 constitui a LCR. Então M3 será associada ao cluster C1.

Analogamente, M5 será associada à C1, que será o único cluster a constituir sua LCR. A associação da máquina M6 será feita ao cluster C3 pois os coeficientes de M6 com C1, C2 e C3 são respectivamente: 0,286, 0,125 e 0,5 . Sua LCR será constituída apenas por C3, pois somente C3 possui similaridade maior ou igual a (0,125 + (0,5 * 0,375)) = 0,313 com M6. Mas antes da associação de M3 a C1, M6 possuia similaridade igual a 0,33 com C1, ou seja, maior que 0,313 (o que caracterizaria sua inclusão na LCR de M6). Esta queda de similaridade entre máquinas e clusters acontece porque o numerador da equação que define a similaridade entre uma máquina e um cluster na equação 5.1 nunca aumenta no decorrer da clusteri-

zação das máquinas a serem clusterizadas antes da máquina em questão. Somente pode manter-se o mesmo ou diminuir. A consequência disto é que a clusterização das últimas máquinas tende a adquirir um caráter cada vez mais guloso uma vez que o número de candidatos na LCR tende a diminuir e isto pode contribuir para uma menor diversificação na busca por soluções promissoras. Além disso, pode haver casos de soluções com clusters nos quais algumas máquinas podem não possuir nenhuma parte atendida em comum com todas as outras máquinas do cluster mas partes atendidas em comum com um subcojunto destas. Nestes casos a similaridade entre a máquina e o cluster teria erroamente o valor igual a 0.

Ainda assim, o procedimento de construção mostra-se razoável (versão G9), tendo obtido os melhores resultados (não isoladamente) em 17 das 36 instâncias. Além disso, o tempo de execução da versão apenas com a fase de construção (G9) é bastante satisfatório.

Dentre as versões sem filtro e com busca local, a que alcançou um melhor desempenho foi a versão G6, que usa o mesmo modelo de busca local do algoritmo evolutivo proposto neste trabalho. A versão G6 só não alcançou o melhor resultado dentre todas as versões em 1 instância (34) e obteve o melhor resultado dentre todas as versões isoladamente em 8 instâncias (14, 18, 21, 26, 27, 30, 31 e 32). Para a instância 31, o resultado de G6 foi melhor que o melhor resultado alcançado até então (alcançado pelo AE proposto neste trabalho). Com relação à tempo de execução, G6 também obteve resultados satisfatórios. Verificando as instâncias para as quais a versão G9 não conseguiu a melhor solução, pode-se observar que a versão G7, que usa o segundo modelo de busca local conseguiu melhores soluções que G9 em 10 destas instâncias (8, 14, 18, 19, 20, 22, 25 a 30, 32 a 35) comprovando a eficiência do segundo modelo de busca. Entretanto, dentre os resultados destas 10 instâncias, somente para instância 31 é que G7 conseguiu a melhor solução dentre todas as versões (não isoladamente). O terceiro modelo de busca local mostrou-se o pior dentre os três modelos propostos, não alcançando resultados muito superiores aos da versão G9.

A versão G8 (que utiliza o terceiro modelo de busca local) apresenta tempos

de execução ora muito pequenos, ora muito grandes. Isto se deve ao fato de que a idéia do terceiro modelo de busca local é baseado na destruição de clusters ruins (de acordo com a métrica λ_k denominada *qualidade do cluster* mostrada na equação 5.5 do capítulo 5 deste trabalho) e sua reconstrução de acordo com um algoritmo guloso (que é de ordem $O(m^3\mathrm{p})$), sendo que um cluster é reconstruído se o valor de λ_k for menor ou igual à um valor limite, que foi fixado nestes experimentos em 0,3. Quando uma solução já possui clusters com boa qualidade, a busca local não reconstrói nenhum cluster e portanto não é executada. Portanto, instâncias para as quais a fase de construção já consegue bons resultados e cujo valor da função objetivo de tais soluções é alto tem pouca execução do terceiro modelo de busca local, o que faz com que o tempo de G8 seja pequeno (instâncias 1 a 7, 9 a 13, 15 a 17, 22 a 24 e 36). Já instâncias para as quais a fase de construção não é tão eficiente e que possuem valor baixo da função objetivo tendem a executar o terceiro modelo de busca mais frequentemente, o que faz com que o tempo de execução de G8 seja grande (instâncias 35 e 30).

Embora tenham sido obtidos bons resultados, para 5 instâncias nenhuma versão da metaheurística GRASP foi capaz de chegar à melhor solução. No caso da instância 2, experimentos realizados com todas as versões mostraram que quando as mesmas aceitam soluções com clusters unitários como válidas, todas elas alcançam a melhor solução da literatura, a qual possui clusters unitários.

Em todos os experimentos mostrados na tabela 6.7, o par de máquinas a serem as duas primeiras sementes dos dois primeiros clusters na fase de construção é escolhido aleatoriamente no subconjunto que contém os pares menos similares, sendo que tal subconjunto tem o tamanho igual a 30% do total de pares de máquinas. Para possibilitar uma maior exploração do espaço de busca as versões G2, G6, G7, G8 e G9 foram novamente executadas 10 vezes para cada instância, sendo que a escolha das máquinas semente foi feita aleatoriamente dentre todos os pares de máquina, embora isto possa causar a construção de soluções ruins, quando duas máquinas com alto índice de similaridade são escolhidas como semente de diferentes clusters. O número de soluções por iteração também foi alterado, sendo igual ao número de máquinas m de cada instância. O número de iterações permaneceu igual a 100.

A tabela 6.9 apresenta os resultados obtidos. Além disso, na versão G8 o valor limite do terceiro modelo de busca local, se define se um cluster deve ser ou não reconstruido, não foi fixado mas escolhido randomicamnente entre -1 e 1, que são os valores mínimo e máximo para a *qualidade do cluster* λ_k.

I	G2	G6	G7	G8	G9	Melhora
1	**73,68**	**73,68**	**73,68**	**73,68**	**73,68**	0%
2	**62,50**	**62,50**	**62,50**	**62,50**	**62,50**	0%
3	**79,59**	**79,59**	**79,59**	**79,59**	**79,59**	0%
4	**76,92**	**76,92**	**76,92**	**76,92**	**76,92**	0%
5	**53,13**	**53,13**	**53,13**	**53,13**	**53,13**	0%
6	**70,37**	**70,37**	**70,37**	**70,37**	**70,37**	0%
7	**68,29**	**68,29**	**68,29**	**68,29**	**68,29**	0%
8	58,41	**58,72**	58,41	58,33	58,33	0%
9	**85,25**	**85,25**	**85,25**	**85,25**	**85,25**	0%
10	**70,59**	**70,59**	**70,59**	**70,59**	**70,59**	0%
11	**92,00**	**92,00**	**92,00**	**92,00**	**92,00**	0%
12	67,65	**69,86**	67,65	67,65	67,65	0%
13	**69,33**	**69,33**	**69,33**	**69,33**	**69,33**	0%
14	51,48	**51,96**	**51,96**	49,04	49,51	0%
15	**67,83**	**67,83**	**67,83**	**67,83**	**67,83**	0%
16	**54,86**	**54,86**	**54,86**	**54,86**	**54,86**	0%
17	**75,71**	**75,71**	**75,71**	**75,71**	**75,71**	0%
18	**54,46**	**54,46**	54,05	51,79	50,81	0%
19	42,76	**42,96**	42,76	40,80	41,67	0%
20	**49,65**	**49,65**	**49,65**	47,52	46,85	0%
21	**76,14**	**76,14**	**76,14**	**76,14**	**76,14**	-0,10%
22	**58,06**	**58,06**	**58,06**	57,07	56,99	0%
23	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	0%
24	**85,11**	**85,11**	**85,11**	**85,11**	**85,11**	0%
25	**73,51**	**73,51**	**73,51**	72,30	72,30	0%
26	51,52	**51,97**	51,59	46,39	47,74	0%
27	46,53	**47,06**	46,95	44,67	44,00	0%
28	**44,87**	**44,87**	**44,87**	44,00	43,31	0%
29	**54,27**	**54,27**	52,61	47,86	47,29	0%
30	39,92	**42,75**	**42,75**	38,55	40,15	-4,39%
31	57,89	**58,48**	57,89	57,89	57,89	0,58%
32	58,76	**59,66**	57,29	56,45	56,76	0%
33	**50,51**	**50,51**	**50,51**	49,24	48,76	0%
34	37,53	**38,63**	38,28	35,46	35,24	-4,74%
35	**56,42**	**56,42**	56,36	53,12	53,73	-1,95%
36	**84,03**	**84,03**	**84,03**	**84,03**	84,03	0%

Tabela 6.9: Novos resultados para as versões G2, G6, G7, G8 e G9

Comparando os resultados das tabelas 6.9 e 6.7 obtemos as seguintes conclusões para cada versão: Os resultados de G2 melhoraram em 9 instâncias (2, 8, 14, 18, 19, 26, 27, 32 e 34) e pioraram em 1 (30); os resultados de G7 melhoraram em 9 instâncias (2, 8, 14, 18, 22, 27, 28, 29 e 34) e pioraram em 3 (19, 30, 32); os resultados de G8 melhoraram em 13 instâncias (2, 8, 14, 18, 20, 22, 26, 27, 30 a 34) e pioraram em 3 (19, 29 e 35); os resultados de G9 melhoraram em 12 instâncias (2, 8, 14, 19, 20, 22, 27, 28, 30, 32, 33 e 35) e pioraram em 4 (18, 26, 29, 34) e os resultados de G6 melhoraram em 3 instâncias (2, 12 e 34) e pioraram em 2 (30, 21), sendo que os resultados de 2 e 12 dizem respeito à melhor solução da literatura.

Desta forma, uma maior liberdade para a escolha das máquinas semente traz uma melhoria na performance do algoritmo GRASP. Na versão G8, além desta maior liberdade na escolha das máquinas semente, a variação do valor limite do terceiro modelo de busca local, o qual decide se um cluster será reconstruído ou não também foi fator determinante para um melhor desempenho.

Ainda, de acordo com as tabelas 6.7 e 6.9 a versão G6 foi a mais eficiente, chegando mais freqüentemente às melhores soluções num tempo computacional razoável. A tabela 6.10 mostra os tempos de execução e também as iterações das melhores soluções para os testes da tabela 6.9. Os rótulos das colunas tem os seguintes significados: I - instância, G2-T - tempo para a versão G2, G2-I - Iteração GRASP da melhor solução para a versão G2. Para as versões G6, G7, G8 e G9 os rótulos das colunas tem o mesmo significado.

I	G2-T	G2-I	G6-T	G6-I	G7-T	G7-I	G8-T	G8-I	G9-T	G9-I
1	0,15	1	0,02	1	0,02	1	0,02	1	0,01	1
2	0,14	1	0,02	1	0,02	3	0,02	4	0,01	1
3	0,33	1	0,05	1	0,09	1	0,03	1	0,02	1
4	0,26	1	0,03	1	0,03	1	0,03	1	0,02	1
5	0,49	1	0,06	1	0,06	1	0,05	1	0,04	1
6	0,44	1	0,06	1	0,06	1	0,04	1	0,03	1
7	0,70	1	0,09	1	0,10	1	0,07	3	0,05	1
8	1,25	11	0,17	11	0,25	1	0,15	1	0,09	1
9	1,03	1	0,14	1	0,14	1	0,10	1	0,08	1
10	1,59	1	0,18	1	0,20	1	0,13	1	0,11	1
11	1,75	1	0,21	1	0,33	1	0,14	1	0,13	1
12	5,19	1	0,76	1	0,82	11	0,45	2	0,39	8
13	5,57	1	0,84	1	0,91	1	0,50	1	0,45	1
14	19,67	77	2,13	7	2,47	97	1,64	9	1,29	26
15	10,71	1	1,70	1	1,99	1	1,14	1	0,75	12
16	19,61	2	2,46	1	3,43	9	1,68	18	1,24	36
17	23,49	1	3,43	1	6,18	1	2,10	1	1,65	1
18	83,36	26	6,31	4	4,50	30	9,32	7	4,66	52
19	51,05	11	3,48	9	4,66	5	4,81	99	3,21	50
20	49,42	11	4,19	1	5,17	1	4,58	63	3,14	1
21	23,07	1	3,24	1	4,19	1	1,94	1	1,55	1
22	28,88	11	4,09	1	5,67	23	3,52	4	2,00	59
23	56,80	1	4,09	1	3,52	1	3,37	1	3,89	1
24	57,15	1	5,41	1	14,84	1	3,92	1	3,97	1
25	68,49	4	7,78	1	8,77	1	7,01	3	4,89	9
26	86,89	61	10,55	8	12,12	39	9,61	10	6,15	95
27	72,56	11	8,22	25	11,22	37	8,24	31	5,64	66
28	56,98	11	6,71	10	10,54	1	6,44	51	4,41	12
29	177,25	11	11,79	2	17,21	27	25,25	87	11,72	84
30	1019,45	79	140,23	93	75,10	3	110,88	9	87,89	71
31	322,19	5	40,97	66	31,50	12	33,78	34	25,65	2
32	197,19	11	23,68	78	26,47	84	25,50	2	14,46	31
33	215,26	11	24,47	2	30,92	2	22,15	7	17,01	50
34	2257,78	12	179,21	78	165,70	72	204,97	4	161,60	69
35	1922,20	11	143,43	1	119,73	1	374,96	58	129,66	72
36	598,43	1	50,91	1	135,73	1	41,31	1	42,83	1

Tabela 6.10: Tempos de execução e iteração das melhores soluções da tabela 6.9

A tabela 6.10 nos mostra que os tempos de execução das versões sem filtro e da

versão G9 (versão somente com procedimento de construção) são bastante razoáveis e, além disso, a eficiência das versões com busca local, alcançando as melhores soluções nas primeiras gerações para a maioria das instâncias.

A tabela 6.11 compara os melhores resultados, tempos de execução e resultados médios das melhores soluções para o AE e a melhor versão da metaheurística GRASP (GRASP6). Os rótulos das colunas da referida tabela tem os seguintes significados: Instâncias - instâncias utilizadas nos experimentos; AE - melhor solução do algoritmo evolutivo (AE); G6 - melhor solução do GRASP6; AE-T - tempo de execução do AE para a melhor solução; G6-T - tempo de execução do GRASP6 para a melhor solução; AE-Media - média das melhores soluções do AE (os algoritmos foram executados 10 vezes para cada instância) para cada instância e G6-Media - média das melhores soluções do GRASP6 para cada instãncia.

Instância	AE	G6	AE-T	G6-T	AE-Media	G6-Media
1	**73,68**	**73,68**	0,06	**0,02**	**73,68**	**73,68**
2	**62,50**	**62,50**	0,06	**0,02**	62,17	**62,50**
3	**79,59**	**79,59**	0,11	**0,05**	**79,59**	**79,59**
4	**76,92**	**76,92**	0,09	**0,03**	**76,92**	**76,92**
5	**53,13**	**53,13**	0,16	**0,06**	**53,13**	**53,13**
6	**70,37**	**70,37**	0,15	**0,06**	**70,37**	**70,37**
7	**68,29**	**68,29**	0,20	**0,09**	**68,29**	**68,29**
8	**58,72**	**58,72**	0,29	**0,17**	**58,72**	58,59
9	**85,25**	**85,25**	0,30	**0,14**	**85,25**	**85,25**
10	**70,59**	**70,59**	0,28	**0,18**	**70,59**	**70,59**
11	**92,00**	**92,00**	0,36	**0,21**	**92,00**	**92,00**
12	**69,86**	**69,86**	1,17	**0,76**	**69,74**	68,53
13	**69,33**	**69,33**	1,25	**0,84**	**69,33**	**69,33**
14	**51,96**	**51,96**	**1,68**	2,13	**51,96**	51,23
15	**67,83**	**67,83**	2,09	**1,70**	**67,83**	**67,83**
16	**54,86**	**54,86**	3,26	**2,46**	**54,86**	**54,86**
17	**75,71**	**75,71**	4,02	**3,43**	**75,71**	**75,71**
18	**54,46**	**54,46**	**2,24**	6,31	54,12	**54,17**
19	**42,96**	**42,96**	**2,71**	3,48	**42,85**	42,84
20	**49,65**	**49,65**	**2,84**	4,19	**49,65**	**49,65**
21	76,14	**76,22**	3,76	**3,24**	**76,14**	**76,14**
22	**58,06**	**58,06**	4,15	**4,09**	**58,06**	**58,06**
23	**100,00**	**100,00**	5,61	**4,09**	**100,00**	**100,00**
24	**85,11**	**85,11**	6,77	**5,41**	**85,11**	**85,11**
25	**73,51**	**73,51**	8,76	**7,78**	**73,51**	**73,51**
26	**51,97**	**51,97**	**9,62**	10,55	**51,89**	51,69
27	**47,06**	**47,06**	9,96	**8,22**	**46,75**	46,68
28	**44,87**	**44,87**	9,69	**6,71**	44,38	**44,78**
29	**54,27**	**54,27**	**6,46**	11,79	**54,26**	**54,26**
30	**44,62**	42,75	**14,64**	140,23	**44,26**	42,03
31	58,14	**58,48**	**17,23**	40,97	57,73	**58,05**
32	**59,66**	**59,66**	**18,69**	23,68	**59,50**	59,23
33	**50,51**	**50,51**	**20,63**	24,47	50,33	**50,48**
34	**43,37**	38,63	**39,46**	179,21	**41,83**	37,12
35	**57,54**	56,42	**20,99**	143,43	**56,67**	56,42
36	**84,03**	**84,03**	79,45	**50,91**	**84,03**	**84,03**

Tabela 6.11: Melhores soluções, tempos de execução e soluções médias para o AE e a versão GRASP6

Analisando a tabela 6.11 nota-se que o AE mostrou-se mais eficiente na busca pelas melhores soluções do PCM. Embora com relação às melhores soluções o AE tenha uma pequena vantagem sobre o GRASP6 (superior nas instâncias 30, 34 e 35 e inferior nas instâncias 21 e 31), quando analisamos a média das melhores soluções notamos que o AE obteve um desempenho com uma superioridade considerável sobre o GRASP6 (foi melhor nas instâncias 8, 12, 14, 19, 26, 27, 30, 32, 34 e 35 e pior nas instâncias 2, 18, 28, 31, 33).

Por fim, com relação à tempo computacional, observamos que não existe uma diferença significativa entre os tempos de execução para o AE e o GRASP6, a não ser no caso das instâncias 30, 34 e 35. Observando a tabela 6.8 e a tabela 6.11 notamos que sempre nestas três instâncias a metaheurística GRASP tem um tempo de execução alto, seja com ou sem busca local, com ou sem filtro. Para tais instâncias houve uma maior dificuldade na geração de soluções válidas a cada iteração da metaheurística GRASP, a qual só avança de iteração para iteração após geradas um número de diferentes soluções válidas igual a um número determinado, de acordo com a instância. Desta forma, houve um maior tempo de execução. Foi observado que soluções com clusters com linhas ou colunas vazias são frequentemente geradas, devido à existência de várias máquinas nestas instâncias cuja clusterização não pode ser bem definida de acordo com a fase de construção, devido ao fato destas não possuirem similaridade com todas as outras máquinas do cluster mais adequado para as elas. Com isso, frequentemente estas máquinas possuem similaridade igual a 0 com todos os clusters, o que faz com que todos os clusters sejam incluídos na lista de clusters candidatos de tais máquinas, na fase de construção. Assim, a chance de se associar tais máquinas a um cluster inadequado é maior.

Capítulo 7

Conclusões e trabalhos futuros

Propomos neste trabalho duas metaheurísticas (um algoritmo evolutivo - AE, e um algoritmo GRASP) aplicadas ao problema de formação de células de manufatura (PCM). Além disso, implementamos um algoritmo genético tradicional disponível na literatura.

Nossos experimentos mostraram que embora para algumas instâncias um AG tradicional possa obter bons resultados, quando aplicado a instâncias maiores e com uma clusterização não muito bem definida tal algoritmo se mostra ineficaz e necessita de um tempo computacional alto para alcançar resultados regulares. Isto posto, salientamos que um algoritmo evolutivo para resolução do PCM deve conter mecanismos de aceleração da busca por soluções promissoras. Neste sentido, nossa contribuição foi a elaboração de um AE com um procedimento de busca local eficiente e um procedimento heurístico randomizado de construção de indivíduos iniciais que determinaram o bom desempenho do AE (embora a heurística de construção isoladamente tenha tido uma importância menor no desempenho do AE). Em comparação com outros algoritmos da literatura, nosso AE obteve os melhores resultados e em alguns casos (8 instâncias) conseguiu superá-los no que diz respeito à qualidade das soluções. Entretanto, os procedimentos de construção de indivíduos e de cruzamento possuem uma deficiência que reside no fato de que os mesmos

não possuem um controle sobre a consistência dos indivíduos gerados, podendo ser gerados indivíduos inválidos.

Através da análise probabilística pudemos notar que o AE se mostra muito mais robusto do que o algoritmo genético tradicional para o PCM da literatura implementado por Joines et al. [22] e aqui denominado como AG-JCK. O alcance dos alvos pelo AE é acelerado devido à eficiência da busca local, enquanto o AG-JCK utiliza apenas operadores de mutação e crossover.

Sobre a metaheurística GRASP, concluimos que a fase de construção, a qual é baseada na similaridade de *Jaccard* mostrou-se eficiente, obtendo os melhores resultados da literatura para 17 das 36 instâncias utilizadas em nossos experimentos computacionais. Entretanto, o processo de clusterização das máquinas pode adquirir um caráter guloso ou não conseguir refletir a similaridade entre uma máquina e um cluster, o que resulta na obtenção de resultados apenas regulares para algumas instâncias (por exemplo, instância 34). Dentre os procedimentos de busca local propostos para a metaheurística GRASP, o primeiro modelo de busca (o mesmo usado no AE aqui proposto) mostrou-se o mais eficiente, levando o algoritmo a conseguir mais frequentemente os melhores resultados dentre todas as versões GRASP usadas nos experimentos computacionais.

Com respeito à análise do algoritmo evolutivo e a metaheurística GRASP aqui propostas, entendemos que apesar do desempenho um pouco superior do AE, a metaheurística GRASP é mais bem elaborada, pois ambas as fases de construção e busca local possuem eficiência no que se refere à busca de boas soluções. Já no AE o que se vê é uma forte dependência do mesmo ao procedimento de busca local, sendo que os outros procedimentos (procedimento de construção e cruzamento) não conseguem obter bons resultados sem a presença da busca local.

Os experimentos do GRASP com filtro nos mostraram que é ineficaz a geração de um número grande de soluções a cada iteração e a aplicação da busca local sobre as melhores soluções obtidas ao invés da geração de um número menor de soluções por iteração e a aplicação da busca local sobre toda solução obtida, pois os procedimentos de busca local não conseguiram melhorar as melhores soluções geradas pela fase de

construção.

Pudemos ainda verificar, no nosso entendimento, uma incoerência da medida de performance *Eficácia de Agrupamento*, a qual pode não ser capaz de identificar numericamente a melhor clusterização dentre duas determinadas clusterizações.

Acreditamos que ainda possam surgir interessantes trabalhos à partir deste, tais como: a paralelização das metaheurísticas aqui propostas; um estudo experimental da influência do uso das várias medidas de performance para o PCM nos resultados finais das clusterizações, já que não existe um consenso na literatura sobre a melhor medida de performance; a incorporação por parte dos algoritmos de outras características do PCM como a capacidade de produção das máquinas e a possibilidade de duplicação de máquinas ou terceirização de partes "gargalo"; e o desenvolvimento de outras metaheurísticas aplicadas ao PCM como Busca Tabu, VNS e *Simulated Annealing* podendo aproveitar os conceitos aqui desenvolvidos de busca local e a construção de soluções baseadas na similaridade de *Jaccard* usada na metaheurística GRASP.

Além disso, seria interessante uma análise de probabilidade empírica do nosso algoritmo GRASP, assim como foi feito com o AE aqui proposto. Além disso, é possível a implementação de um GRASP cooperativo, ou seja, um algoritmo GRASP que inclua todas as características das 09 versões aqui implementadas e que possa dinamicamente "aprender" qual a melhor versão a ser usada de acordo com a instância do PCM.

Apêndice - Matrizes de entrada e melhores soluções

Instancia: King and Nakornchai

	0	1	2	3	4	5	6
0		1		1	1	1	
1	1		1				
2	1		1			1	1
3		1		1		1	
4	1				1		1

Matriz de clusterização:

	1	3	4	5	3	3	3
0	1	1	1	1			
3	1	1		1			
1					1	1	
2				1	1	1	1
4			1		1		1

73.68

Instancia: Waghodekar and Sahu

	0	1	2	3	4	5	6
0	1				1	1	1
1		1	1	1	1		
2			1	1	1	1	
3	1	1	1	1			
4		1		1	1	1	

Matriz de clusterização:

	1	2	3	4	5	0	6
1	1	1	1	1			
2		1	1	1	1		
4	1		1	1	1		
0				1	1	1	1
3	1	1	1			1	

62.50

Instancia: Seiffodini

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	1	1		1	1		1			1	1	1	1		1	1	
1	1		1	1		1	1	1		1	1	1	1		1			1
2				1			1			1					1			1
3	1	1	1		1	1		1			1	1	1	1		1	1	
4				1					1	1					1			1

Matriz de clusterização:

	0	1	2	4	5	7	10	11	12	13	15	16	3	6	8	9	14	17
0	1	1	1	1	1	1	1	1	1	1	1	1						
3	1	1	1	1	1	1	1	1	1	1	1	1						
1	1		1		1	1	1	1	1				1	1		1	1	1
2													1	1		1	1	1
4													1		1	1	1	1

79.59

Instancia: Kusiak

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

Matriz de clusterização:

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

76.92

Instancia: Kusiak and Chow

	0	1	2	3	4	5	6	7	8	9	10
0		1	1				1				
1	1				1						1
2										1	1
3	1		1			1					
4					1			1			
5	1			1				1	1	1	
6			1	1		1	1		1		

Matriz de clusterização:

	1	2	5	6	2	3	8	9	10	4	7
0	1	1		1							
3		1	1		1						
6		1	1	1		1	1				
2								1	1		
5					1	1	1	1			1
1					1				1	1	
4										1	1

53.13

Instancia: Boctor

	0	1	2	3	4	5	6	7	8	9	10
0	1	1				1					
1		1				1			1		
2	1		1				1				1
3			1				1				
4			1	1							1
5				1	1					1	
6					1			1		1	

Matriz de clusterização:

	0	1	5	8	2	6	10	3	4	7	9
0	1	1	1								
1		1	1	1							
2	1				1	1	1				
3					1	1					
4					1		1	1			
5								1	1		1
6									1	1	1

70.37

Instancia: Seiffodini and Wolfe

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	1	1	1								
1	1		1	1	1	1	1			1		
2			1	1	1	1	1	1	1			
3						1	1	1	1	1		
4							1	1	1	1		
5							1	1	1		1	
6											1	1
7											1	1

Matriz de clusterização:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	1	1	1								
1	1		1	1	1	1	1			1		
2			1	1	1	1	1	1	1			
3						1	1	1	1	1		
4							1	1	1	1		
5							1	1	1		1	
6											1	1
7											1	1

68.29

Instancia: Chandran and Rajago

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1		1	1					1	1				1	1	1		1	1	
1		1	1	1		1	1		1		1							1		1
2					1	1	1	1			1	1	1			1	1		1	1
3			1	1			1	1	1	1			1	1	1		1	1	1	1
4	1	1	1			1				1		1	1		1	1	1		1	1
5	1	1			1		1	1		1	1	1		1	1	1		1	1	
6					1	1	1	1			1	1	1			1	1		1	1
7	1	1	1	1	1			1	1	1			1	1		1	1			

Matriz de clusterização:

	0	1	2	3	6	7	8	9	13	14	17	4	5	10	11	12	15	16	18	19
0	1		1	1			1	1	1	1	1						1		1	
1		1	1	1	1		1				1		1	1						1
3			1	1	1	1	1	1	1	1	1					1		1	1	1
5	1	1			1	1		1	1	1	1	1		1	1		1		1	
7	1	1	1	1		1	1	1	1			1				1	1	1		
2					1	1						1	1	1	1	1	1	1	1	1
4	1	1	1					1		1			1		1	1	1	1	1	1
6					1	1						1	1	1	1	1	1	1	1	1

58.72

Instancia: Chandra and Rajagob

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		1	1					1	1		1		1	1		1	1		1	
1			1	1		1	1							1				1		1
2		1						1	1		1		1	1		1	1		1	
3			1	1		1	1			1								1		1
4	1				1	1				1		1			1		1			
5	1				1				1	1		1			1					1
6			1	1		1	1				1	1						1		1
7			1	1		1	1											1		1

Matriz de clusterização:

	1	7	8	10	12	13	15	16	18	2	3	5	6	17	19	0	4	9	11	14
0	1	1	1	1	1	1	1	1	1	1										
2	1	1	1	1	1	1	1	1	1											
1						1				1	1	1	1	1	1					
3										1	1	1	1	1	1			1		
6				1						1	1	1	1	1	1				1	
7										1	1	1	1	1	1					
4								1				1				1	1	1	1	1
5			1											1		1	1	1	1	1

8525

Instancia: Mosier and Taube

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

Matriz de clusterização:

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

70.59

Instancia: Chan and Milner

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		1								1	1	1			
1			1		1			1					1		1
2	1					1			1					1	
3	1			1					1					1	
4			1		1			1					1		1
5	1			1		1			1					1	
6		1					1			1	1	1			
7			1		1			1					1		1
8				1		1			1					1	
9		1					1			1	1	1			

Matriz de clusterização

	1	0	9	10	11	2	4	7	12	14	0	3	5	8	13
0	1		1	1	1										
6	1	1	1	1	1										
9	1	1	1	1	1										
1						1	1	1	1	1					
4						1	1	1	1	1					
7						1	1	1	1	1					
2											1		1	1	1
3											1	1		1	1
5											1	1	1	1	1
8												1	1	1	1

92.00

Instancia: Askin and Subramanian

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0							1	1															
1				1	1																		
2				1	1																1		
3		1	1														1		1	1		1	
4		1	1														1			1		1	
5	1									1	1	1		1	1	1							
6		1	1					1									1			1			
7	1					1				1		1		1	1	1							
8						1				1		1	1		1								
9				1																			1
10				1	1																1		
11								1	1														
12							1	1	1									1				1	
13										1	1		1		1								

Matriz de clusterização:

	1	2	10	18	19	21	10	12	22	0	5	9	11	13	14	15	3	4	20	6	7	8	17
3	1	1	1	1	1	1																	
4	1	1	1		1	1																	
6	1	1	1		1																1		
9									1								1						
13							1	1				1			1								
5							1			1		1	1	1	1	1							
7										1	1	1	1	1	1	1							
8								1			1	1	1		1								
1																	1	1					
2																	1	1	1				
10																	1	1	1				
0																				1	1		
11																					1	1	
12						1														1	1	1	1

69.86

Instancia: Stanfel

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0						1	1																	
1			1	1																				
2			1	1																	1			
3	1	1															1		1	1			1	
4	1	1															1			1			1	
5									1	1	1	1		1	1	1						1		
6	1	1					1										1			1				
7					1				1	1		1		1	1	1						1		
8					1				1			1	1		1									
9			1																					1
10			1	1																	1			1
11							1	1																
12						1	1	1										1					1	
13									1		1		1		1									

Matriz de clusterização:

	0	1	10	18	19	22	2	3	20	23	8	9	10	11	13	14	15	21	5	6	7	17	4	12
3	1	1	1	1	1	1																		
4	1	1	1		1	1																		
6	1	1	1		1															1				
1							1	1																
2							1	1	1															
9							1			1														
10							1	1	1	1														
5											1	1	1	1	1	1	1	1						
7											1	1		1	1	1	1	1					1	
0																			1	1				
11																				1	1			
12						1													1	1	1	1		
8											1			1		1							1	1
13											1		1			1								1

69.33

Instancia: McCormick

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1		1	1		1	1	1		1	1				1			1	1			1	1	
1										1							1					1	1	
2	1			1		1									1	1		1						
3															1			1						
4																				1				
5		1	1										1		1		1	1			1			
6	1																	1						1
7	1		1				1			1			1									1	1	
8							1																	1
9		1							1	1				1			1			1		1	1	1
10							1		1					1						1				1
11		1	1						1		1			1						1				1
12		1					1	1			1													
13	1			1		1						1		1										1
14					1	1			1											1				
15									1		1			1			1			1				1

Matriz de clusterização:

	0	2	9	21	22	3	4	5	11	1	8	10	13	16	19	23	12	14	17	20	6	7	15	18
0	1	1	1	1	1	1		1				1						1	1		1	1		1
1			1	1	1									1										
7	1	1	1	1	1										1						1			
13	1					1		1	1				1			1								
14							1	1			1				1									
9			1	1	1					1	1		1	1	1	1								
10											1		1		1	1					1			
11		1								1	1	1	1		1	1								
15											1	1	1	1	1	1								
4																				1				
5		1								1				1			1	1	1	1				
8																1					1			
12										1		1									1	1		
2	1					1		1						1									1	1
3																							1	1
6	1															1								1

5196

Instancia: Srinivasan

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0				1	1		1		1			1						1				1								1
1	1			1						1						1		1		1										
2	1												1										1		1		1	1	1	
3		1		1			1		1									1				1								1
4			1			1		1			1			1	1						1		1		1		1			
5																							1		1		1		1	
6		1		1			1				1							1				1			1					1
7		1		1			1		1		1							1				1				1				
8					1														1				1		1		1	1		
9						1		1			1			1	1			1			1			1		1				
10		1				1		1			1	1						1												1
11		1		1			1					1						1	1	1		1								
12			1							1			1							1									1	1
13						1		1			1			1	1		1				1						1			
14					1									1	1								1		1	1			1	1
15						1	1	1						1	1										1		1	1		

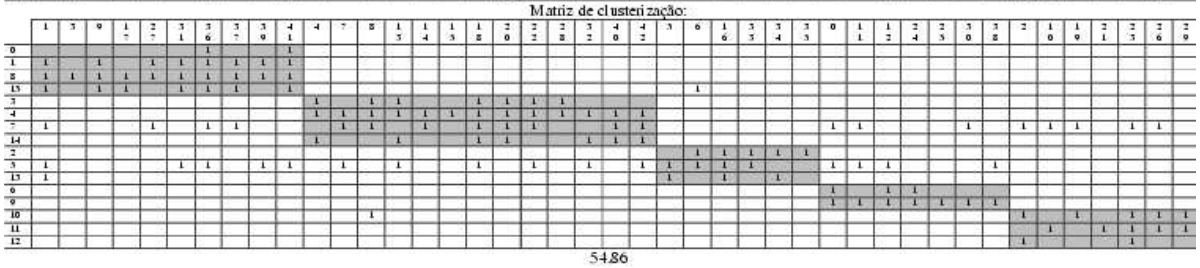
Matriz de clusterização:

	1	3	6	8	11	17	21	29	4	18	22	24	26	27	28	5	7	10	13	14	16	20	23	25	0	2	9	12	15	19
0		1	1	1	1	1	1	1	1																					
3	1	1	1	1		1	1	1																						
6	1	1	1		1	1	1	1				1																		
7	1	1	1	1	1	1	1							1																
10	1		1	1	1	1		1										1												
11	1	1	1		1	1	1			1																				1
2											1	1	1	1	1										1		1			
5											1	1	1		1															
8									1	1	1	1	1	1																
14								1	1		1	1	1		1										1					
4																1	1	1	1	1		1	1	1		1				
9						1										1	1	1	1	1		1	1	1						
13													1			1	1	1	1	1	1	1								
15			1										1			1	1		1	1			1	1						
1		1				1																			1		1		1	1
12								1							1											1	1	1		1

67.83

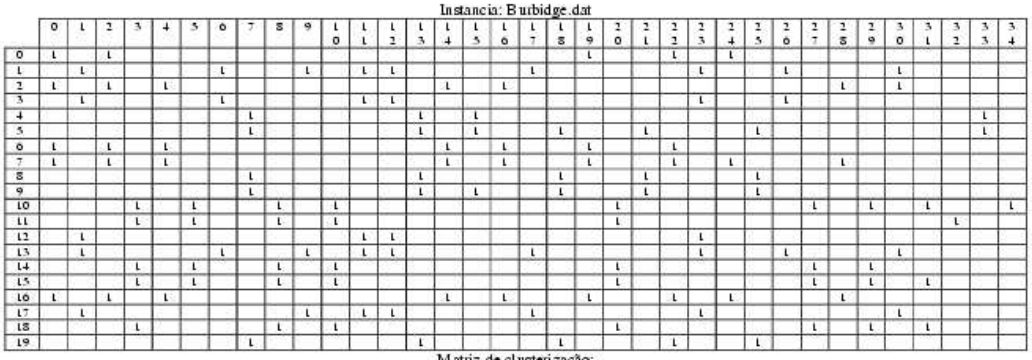
Instancia: King

Matriz de clusterização:

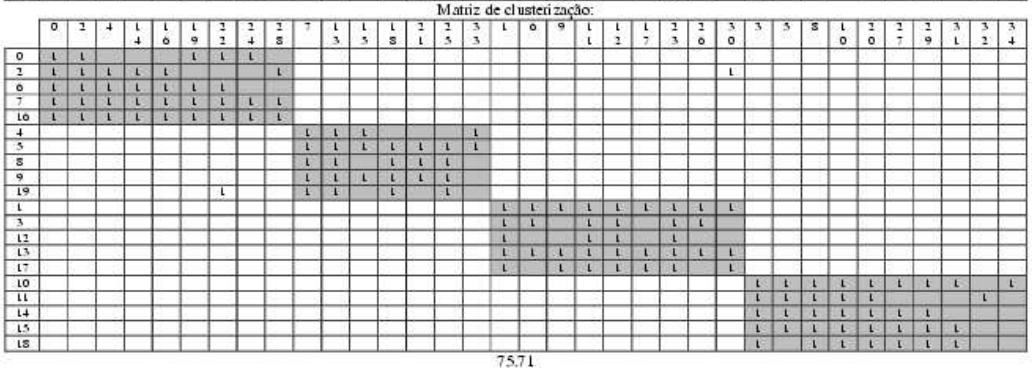


54.86

Instancia: Burbidge.dat



Matriz de clusterização:



75.71

Instancia: Carrie

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0										1			1										1	
1																							1	
2		1			1			1	1			1			1		1							
3		1			1	1		1	1			1			1		1					1		
4		1		1	1			1	1	1	1	1			1	1	1						1	1
5			1		1			1	1	1					1		1						1	1
6		1			1	1						1												
7	1		1	1						1										1			1	1
8			1																					1
9		1										1							1					
10							1							1					1		1			
11						1	1						1	1					1	1	1	1		
12								1	1					1						1		1		
13																			1		1	1		
14	1						1					1			1		1	1						
15				1																				
16				1																				
17											1					1								

Matriz de clusterização

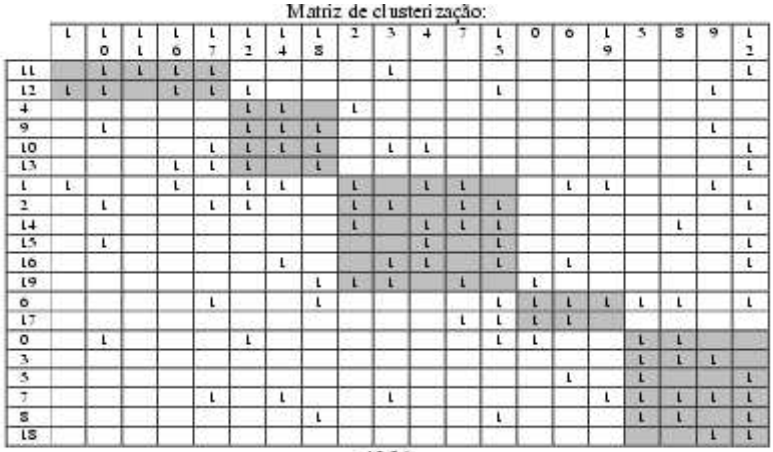
	1	4	5	7	8	11	14	16	9	22	18	21	3	10	15	0	2	19	23	6	12	13	17	20
2	1	1		1	1	1	1	1																
3	1	1	1	1	1	1	1	1				1												
4	1	1		1	1	1	1	1	1	1			1	1	1				1					
5		1		1	1		1	1	1	1							1		1					
6	1	1	1			1																		
0									1	1											1			
1										1														
9	1					1					1													
12				1	1						1	1										1		
15													1											
16													1											
17														1	1									
7									1	1			1			1	1	1	1					
8																1		1						
10																				1		1	1	1
11			1								1	1								1	1	1	1	1
13																		1					1	1
14							1	1								1				1	1		1	

54.46

Instancia: Mosier and Taubeb

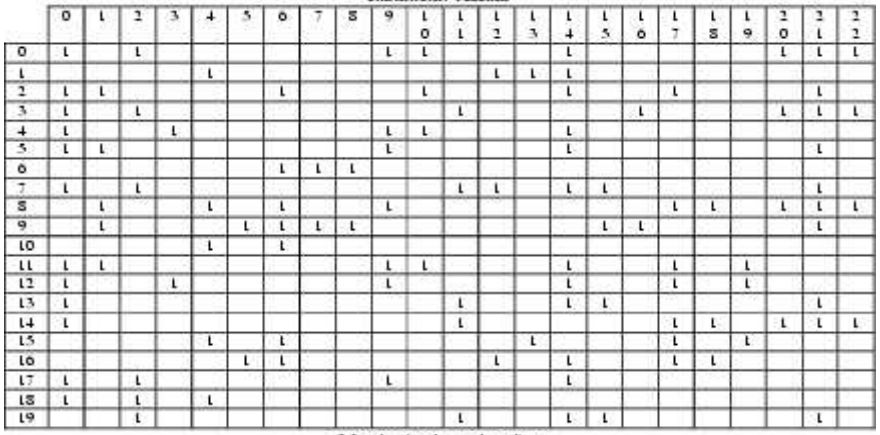
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1					1			1		1		1			1				
1		1	1		1		1	1		1			1		1		1			1
2			1	1				1			1		1	1		1		1		
3						1			1	1										
4			1										1		1					
5						1	1							1						
6	1					1	1		1					1		1		1	1	1
7				1		1			1	1				1	1			1		1
8						1			1					1		1			1	
9										1	1		1		1				1	
10				1	1								1	1	1			1	1	
11				1							1	1		1			1	1		
12		1								1	1		1			1	1	1		
13													1	1			1	1	1	
14			1		1				1	1						1				
15					1						1			1		1				
16				1	1				1					1	1	1				
17	1								1	1						1				
18										1						1				
19	1		1	1				1											1	

Matriz de clusterização:

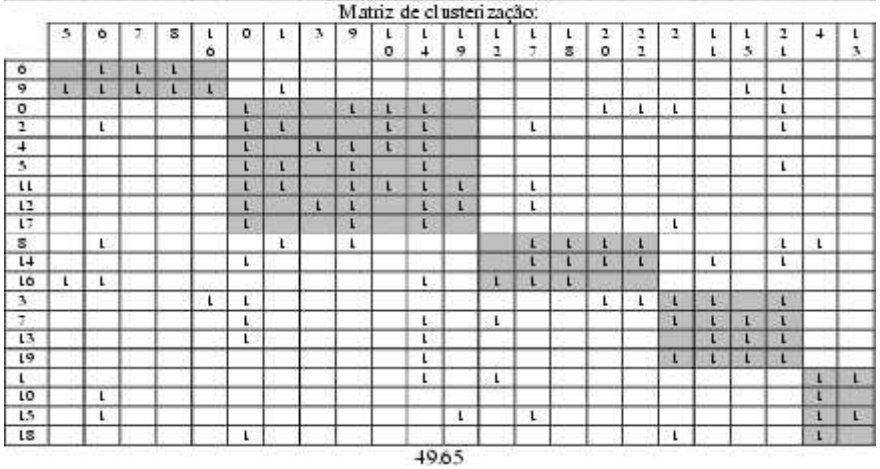


42.96

Instancia: Kumar

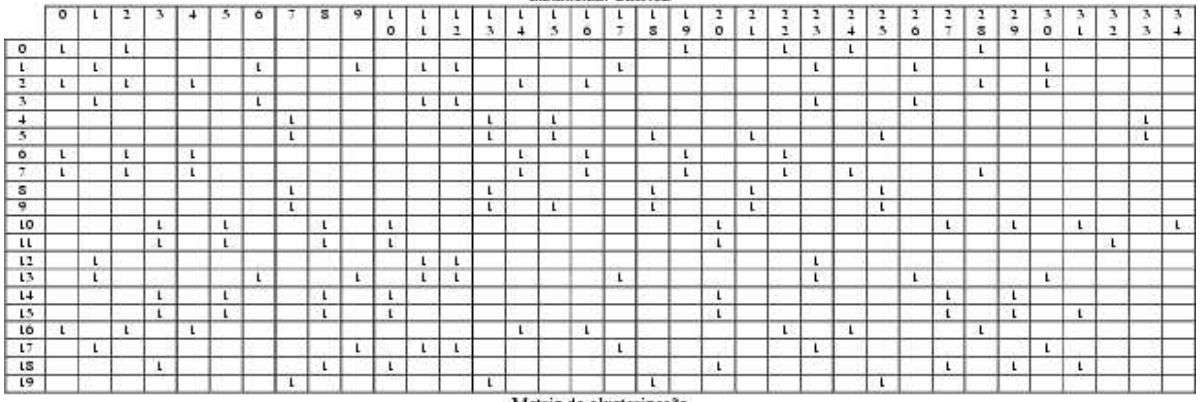


Matriz de clusterização:

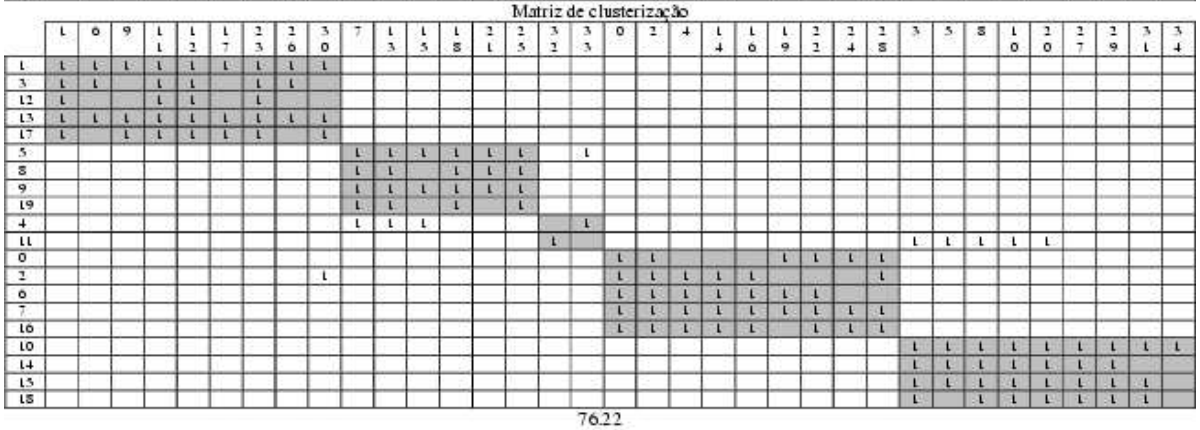


49.65

Instancias: CarrieB

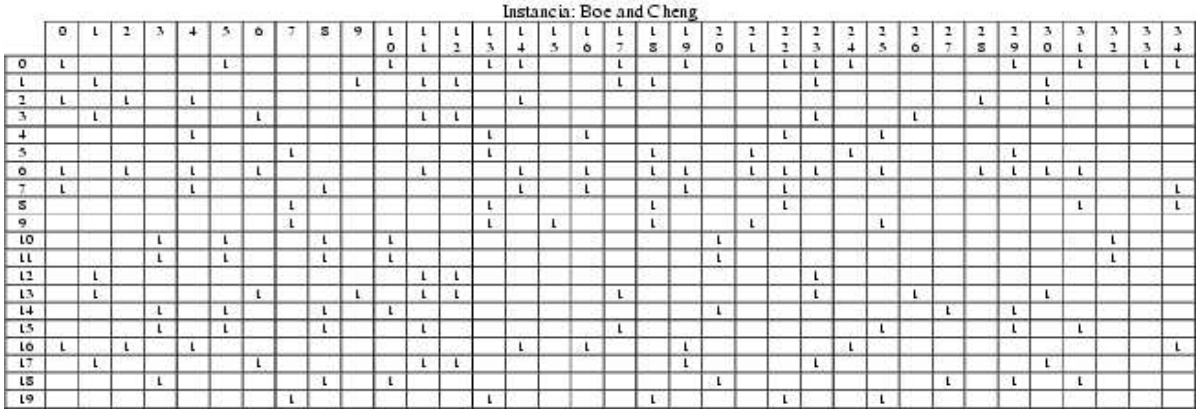


Matriz de clusterização

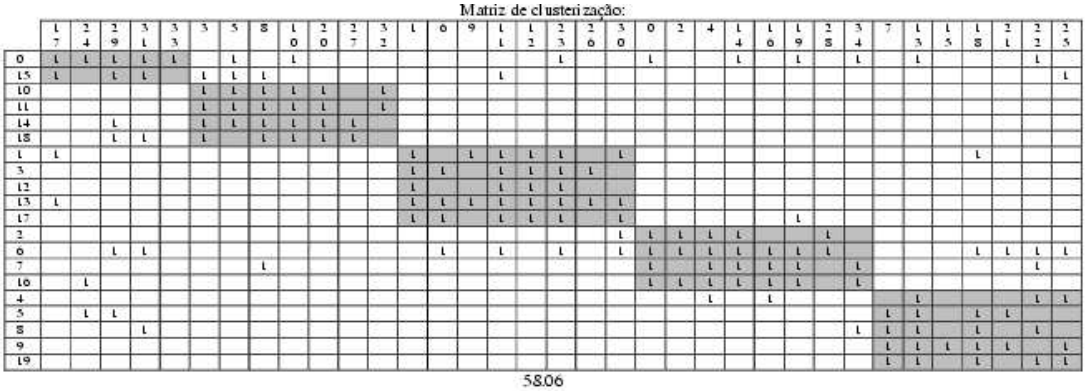


76.22

Instancia: Boe and Cheng

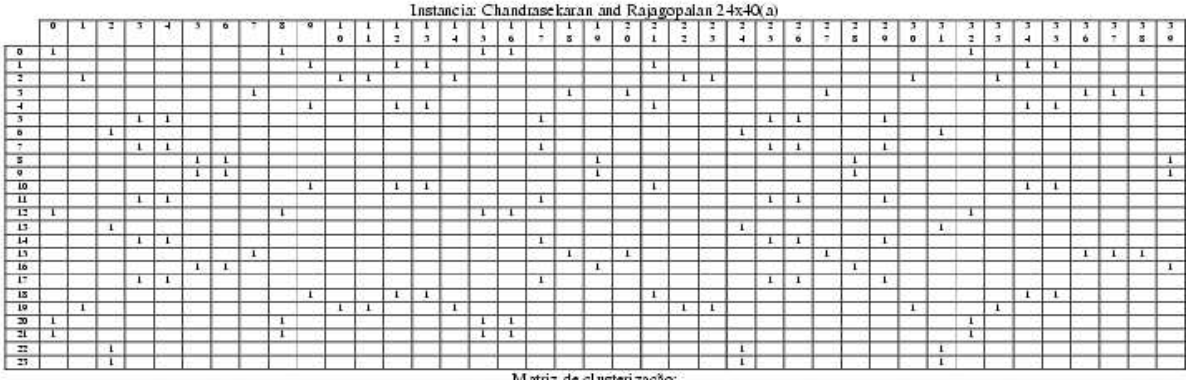


Matriz de clusterização:

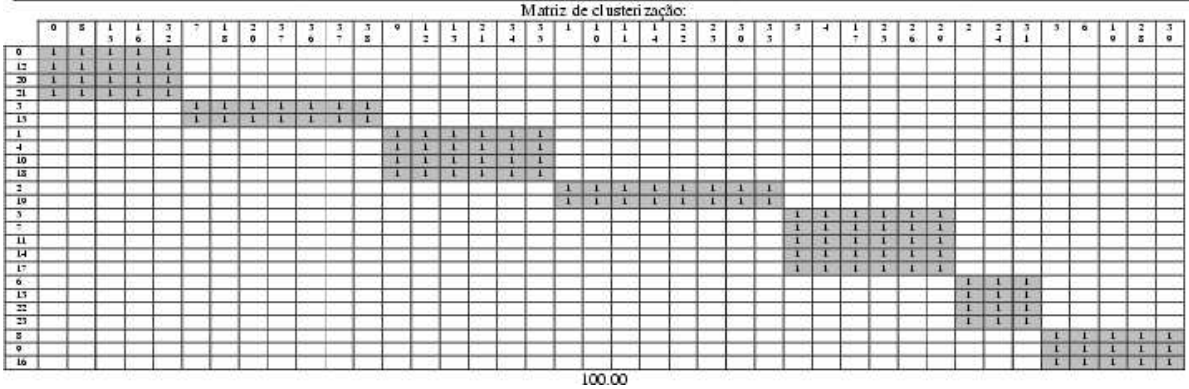


58.06

Instancia: Chandrasekaran and Rajagopalan 24x40(a)

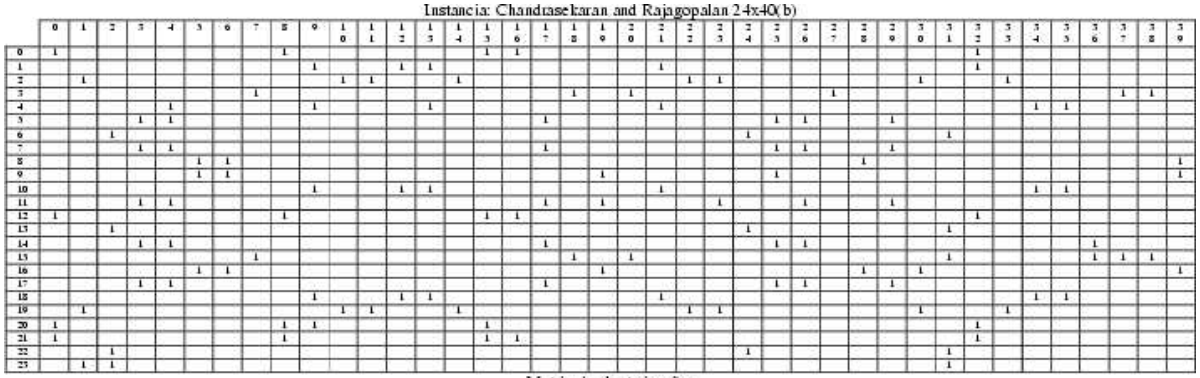


Matriz de clusterização:

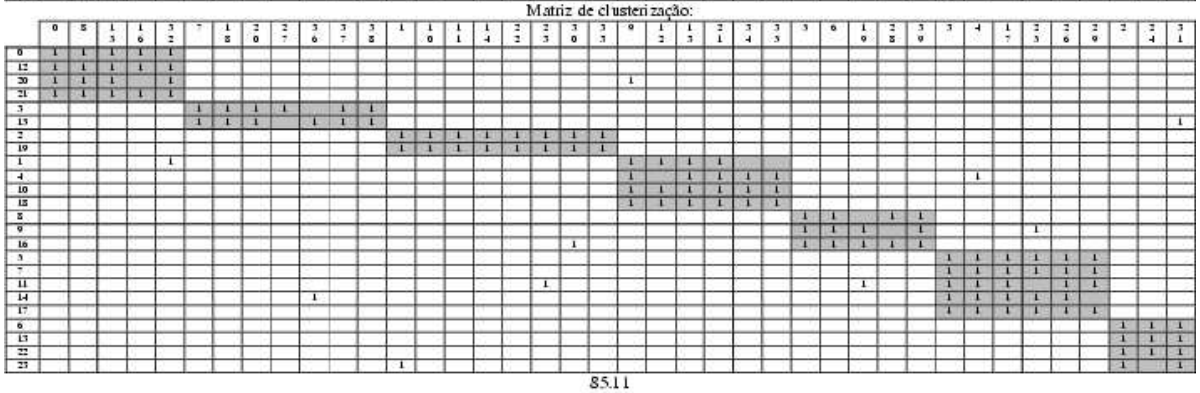


100.00

Instancia: Chandrasekaran and Rajagopalan 24x40(b)

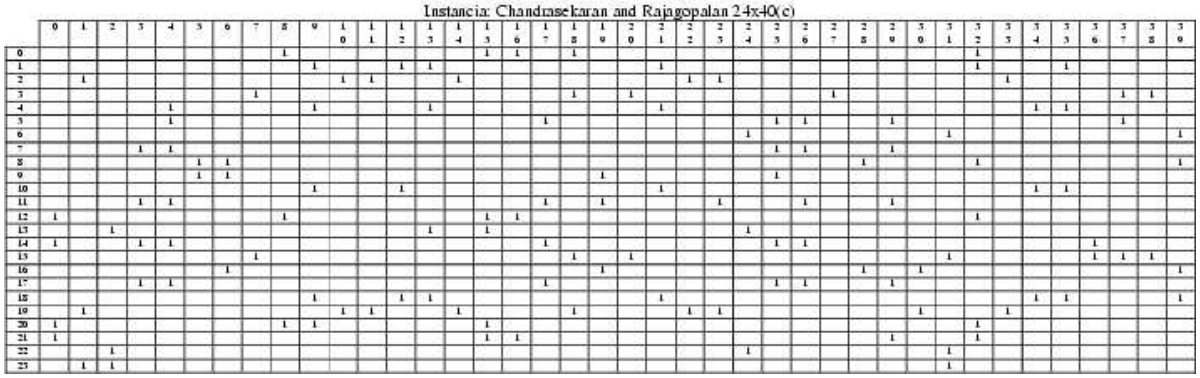


Matriz de clusterização:

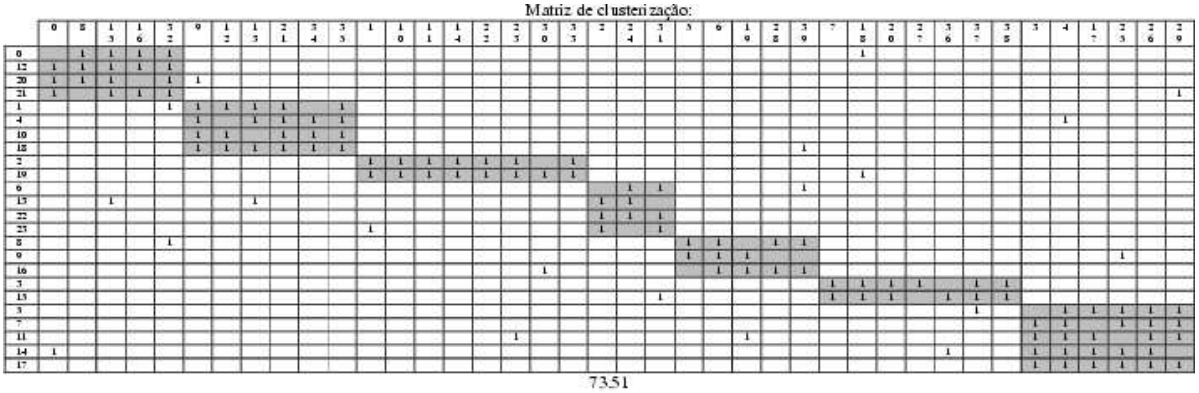


85.11

Instancia: Chandrasekaran and Rajagopalan 24x40(c)

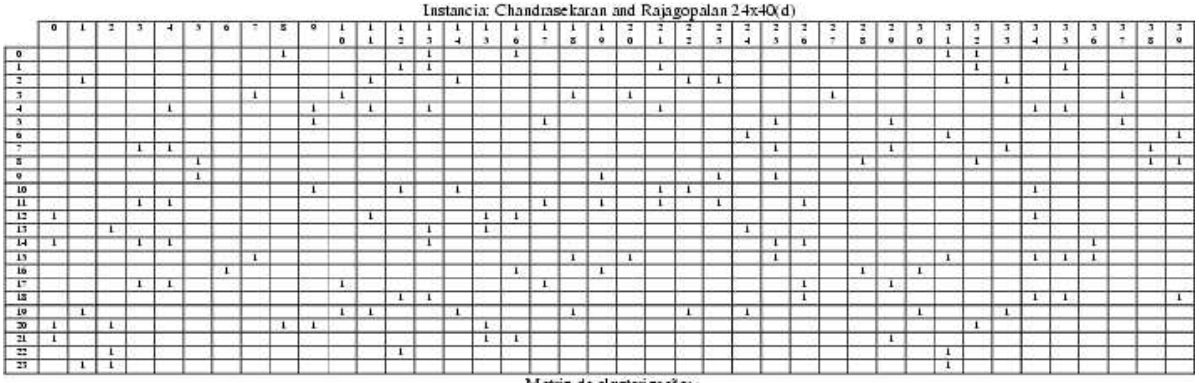


Matriz de clusterização:

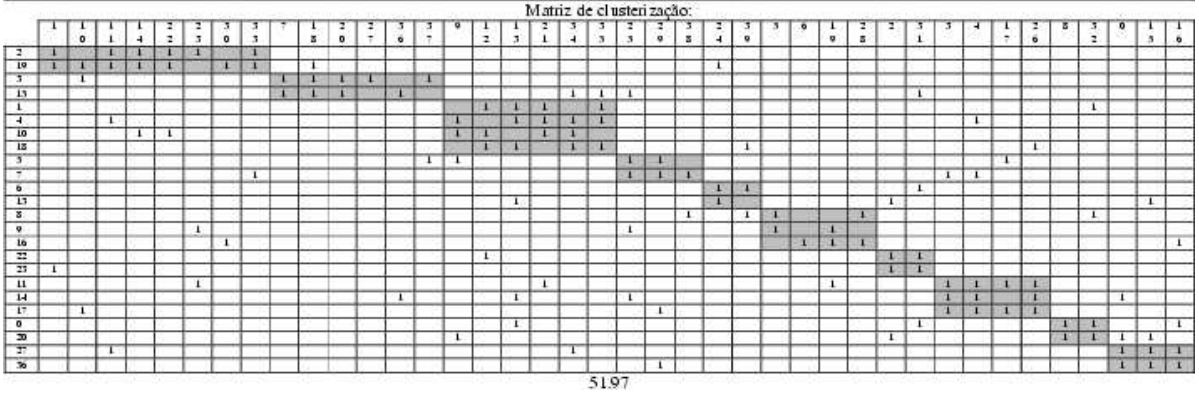


7351

Instancia: Chandrasekaran and Rajagopalan 24x40(d)

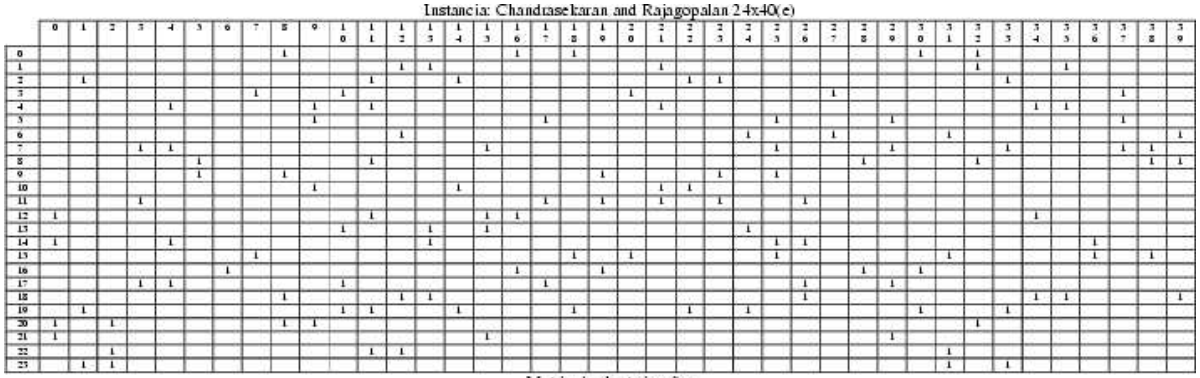


Matriz de clusterização:

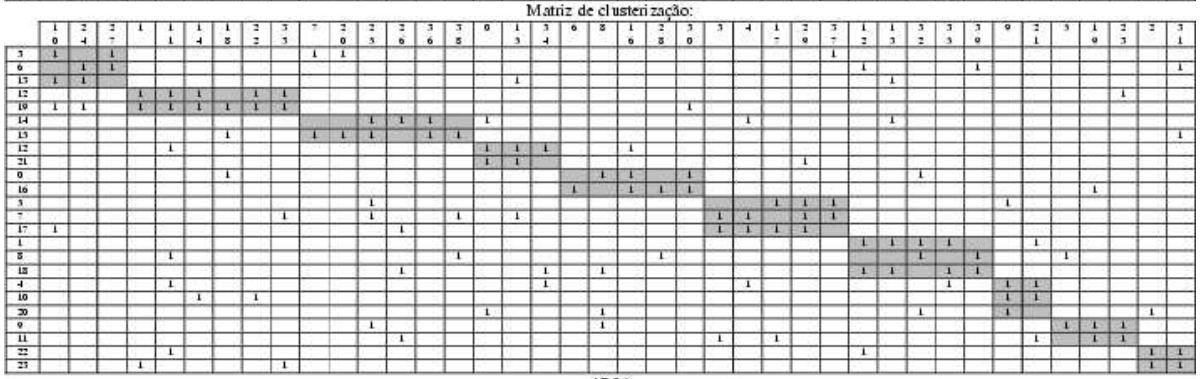


5197

Instancia: Chandrasekaran and Rajagopalan 24x40(e)

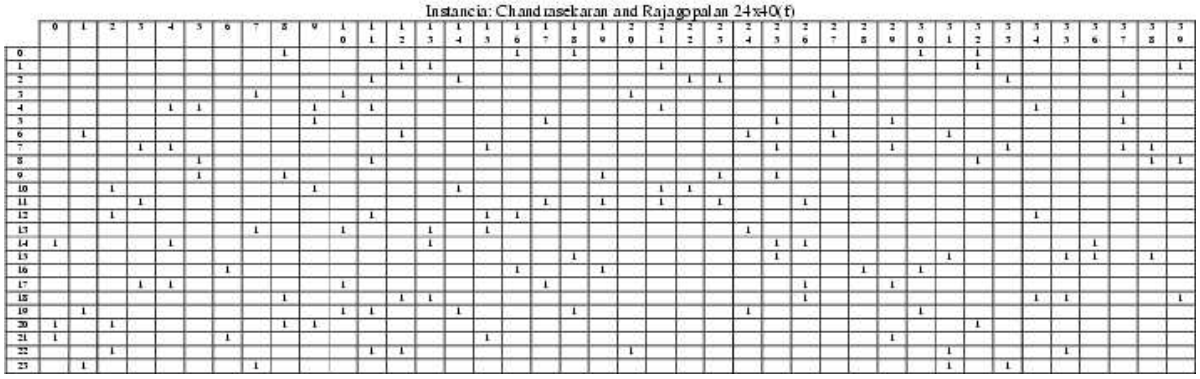


Matriz de clusterização:

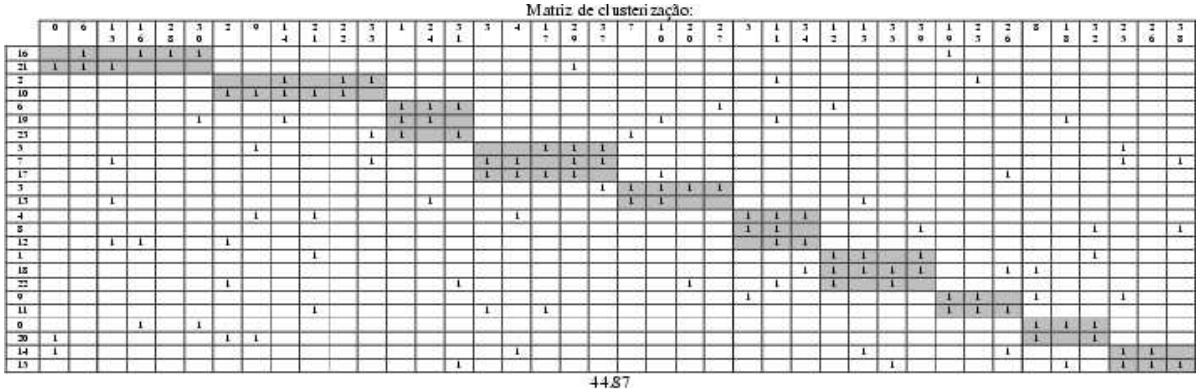


47.06

Instancia: Chandrasekaran and Rajagopalan 24x40(f)

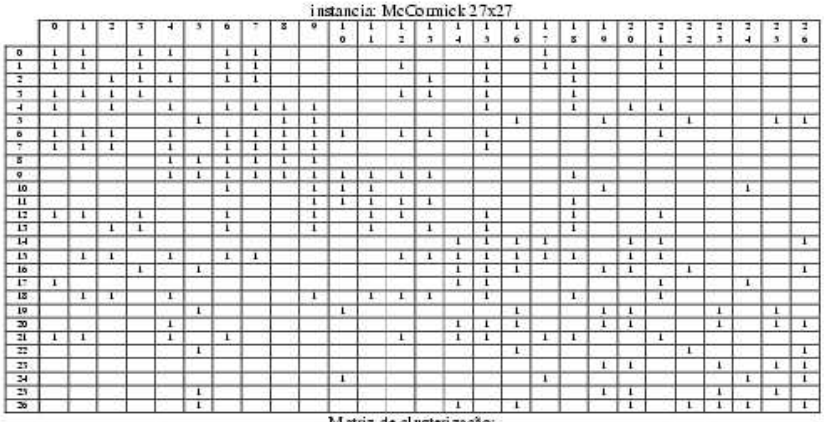


Matriz de clusterização:

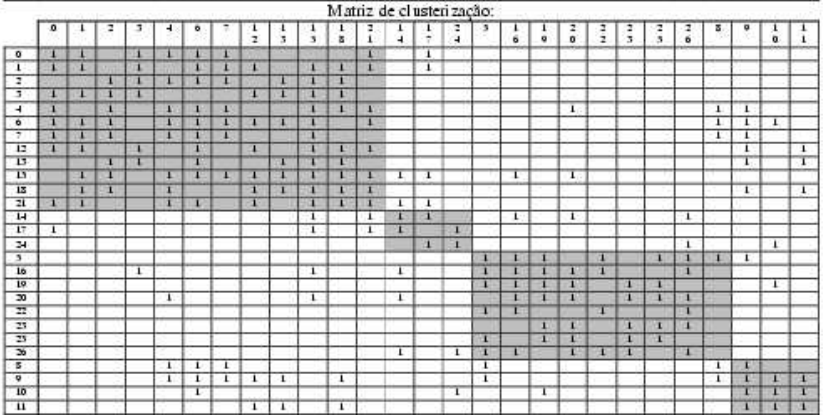


44.87

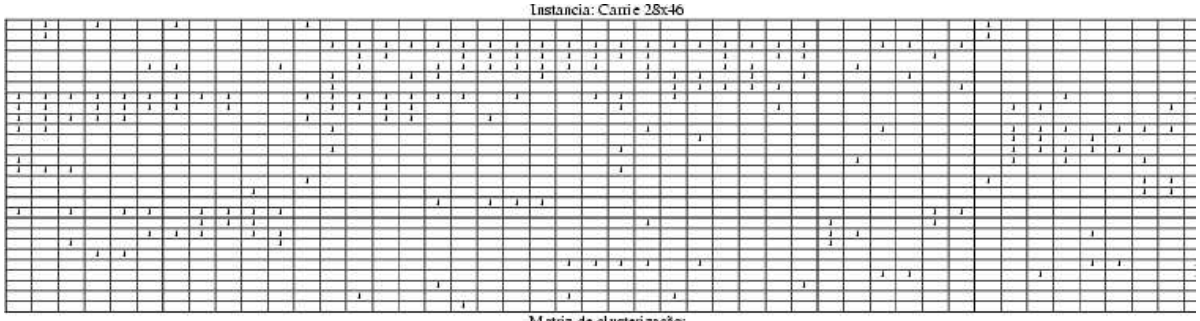
instancia: McCormick 27x27



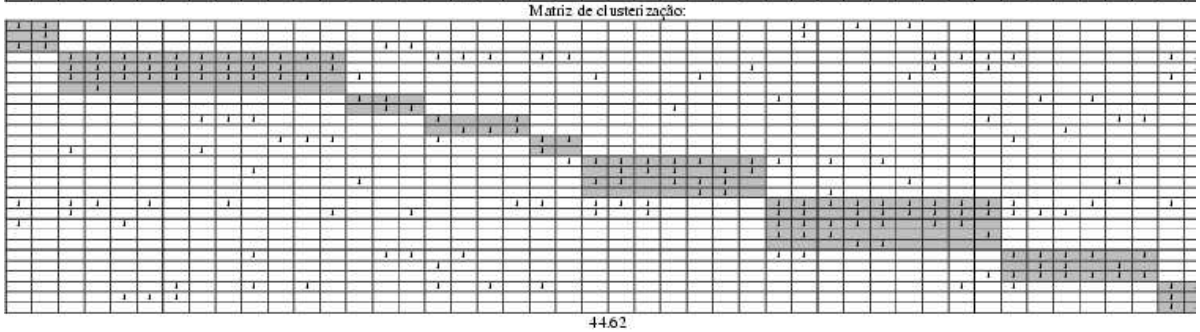
Matriz de clusterização:



Instancia: Carrie 28x46

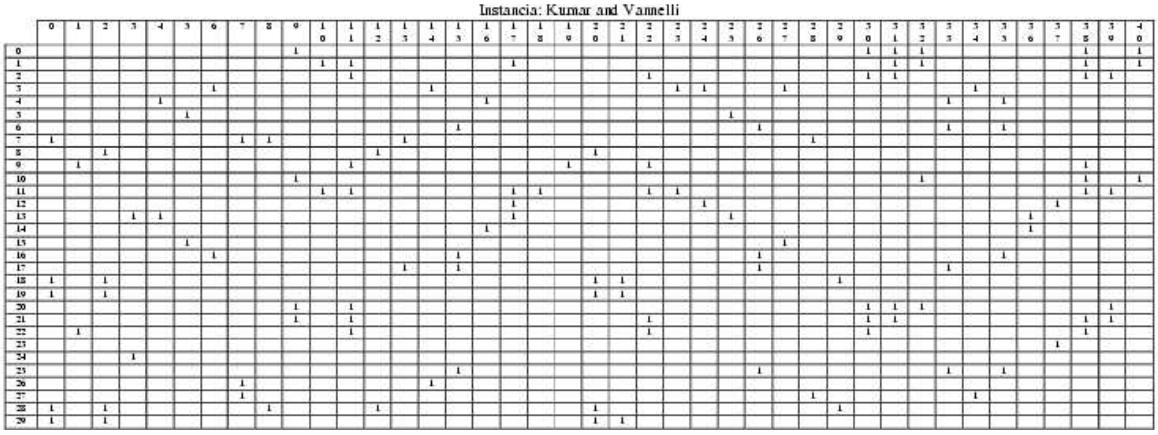


Matriz de clusterização:

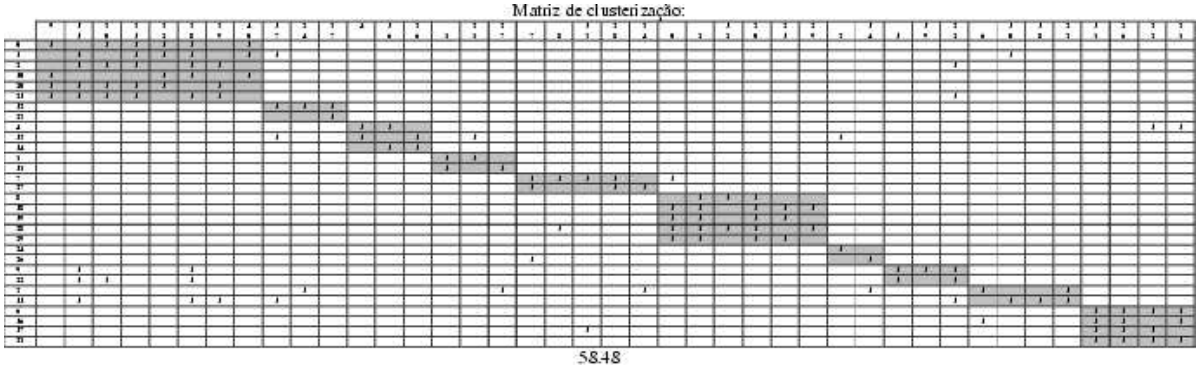


44.62

Instancia: Kumar and Vannelli

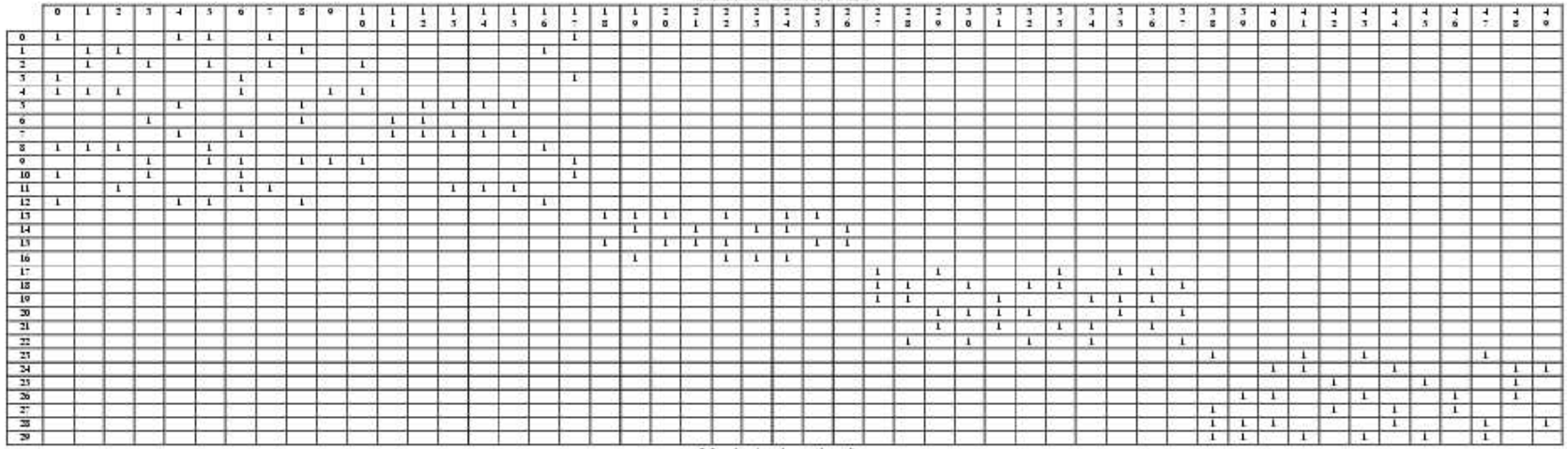


Matriz de clusterização:

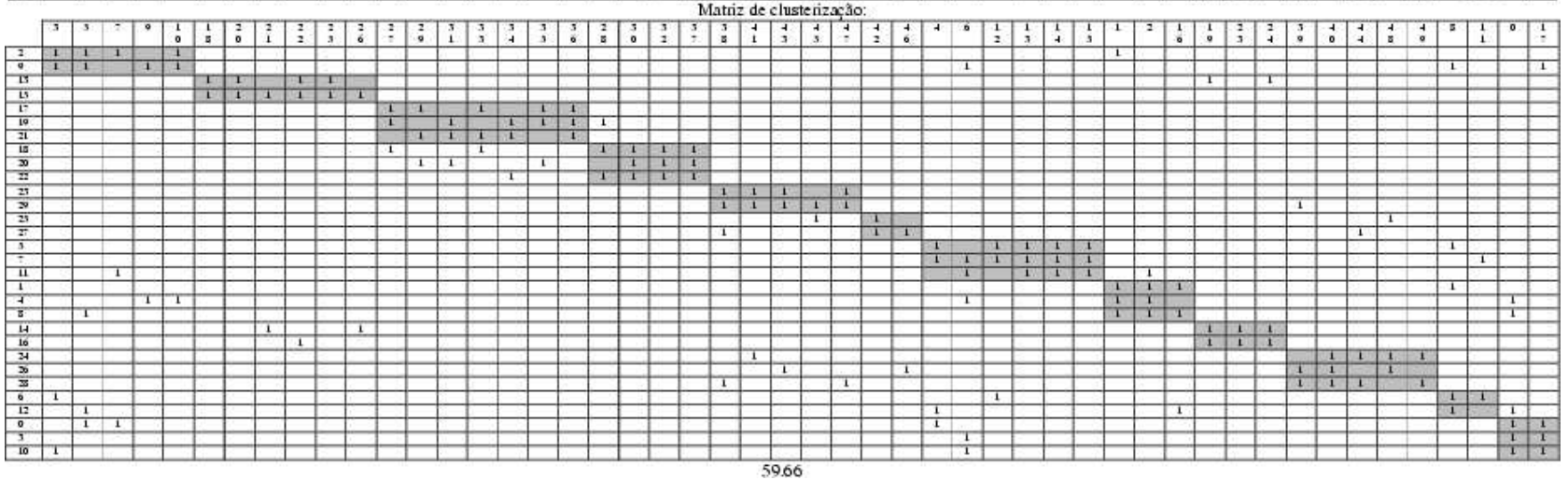


58.48

Instancia: Stanfel 30x50

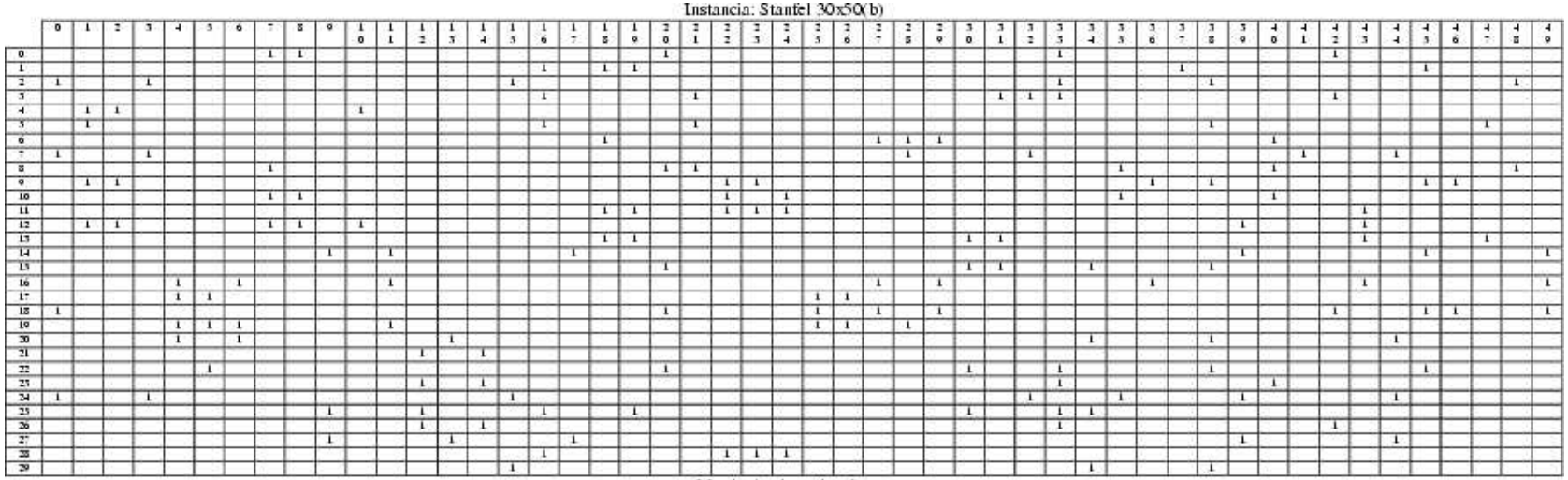


Matriz de clusterização:

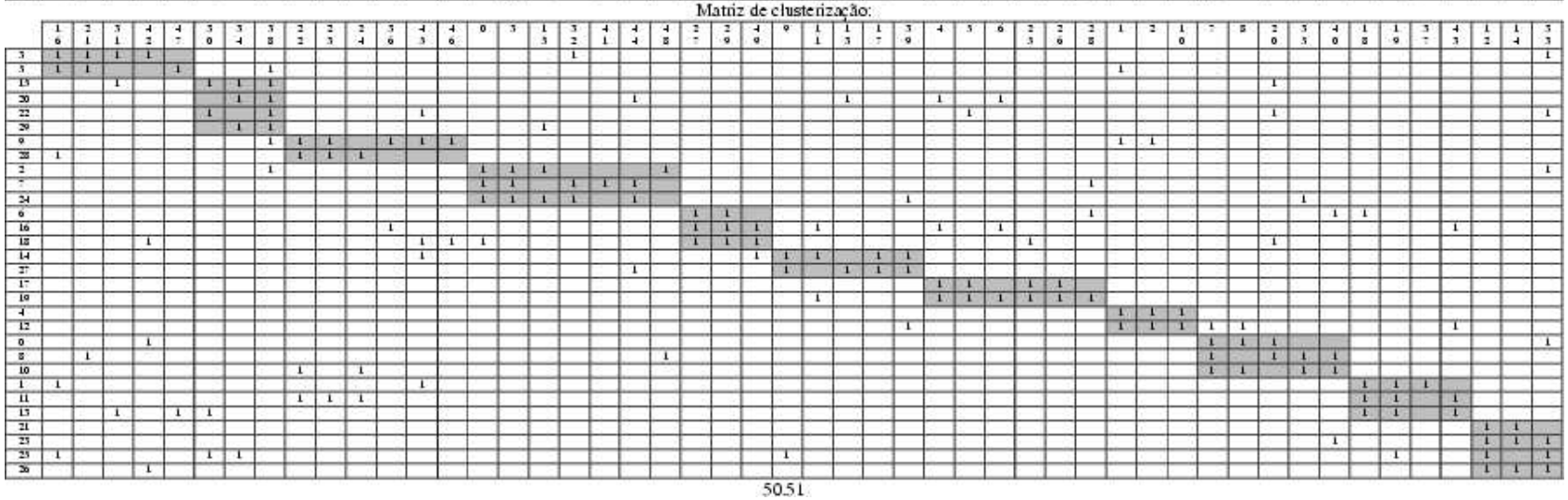


59.66

Instancia: Stanfel 30x50(b)



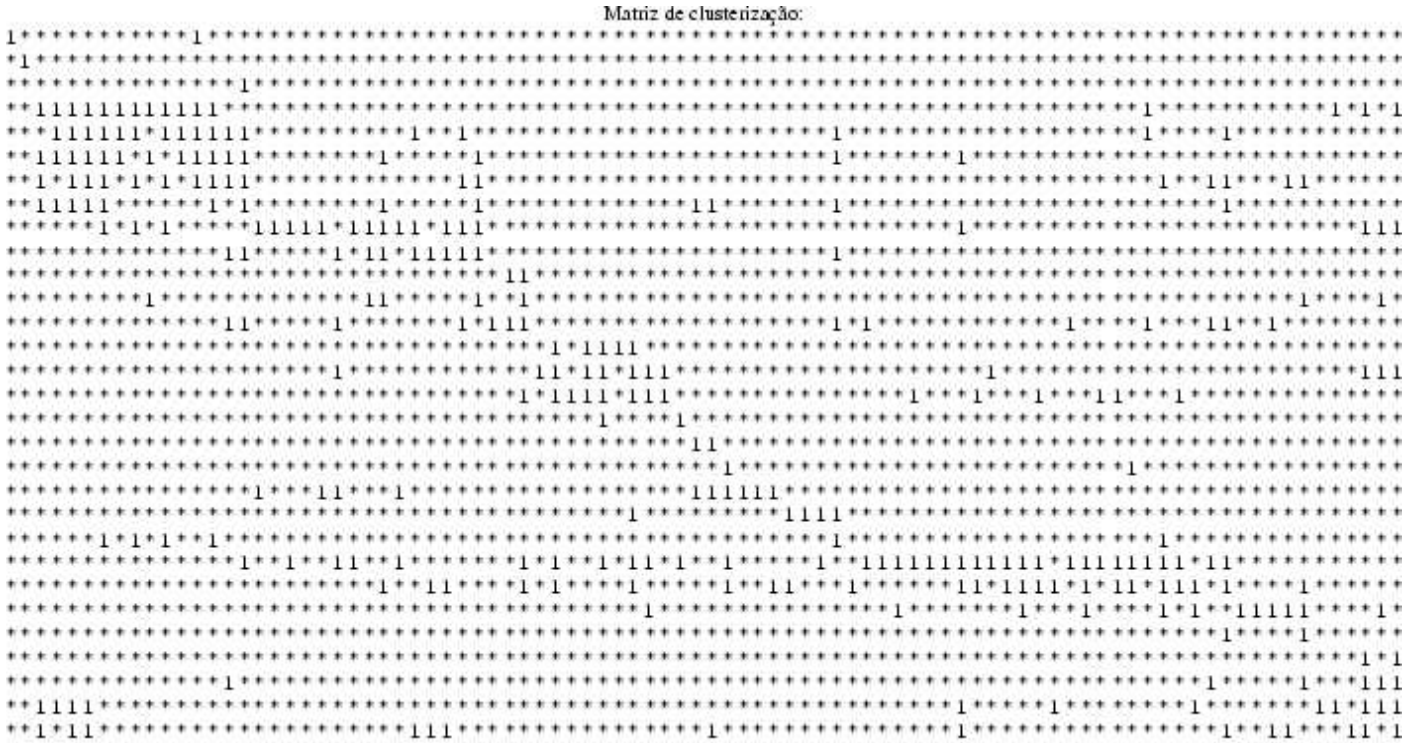
Matriz de clusterização:



50.51

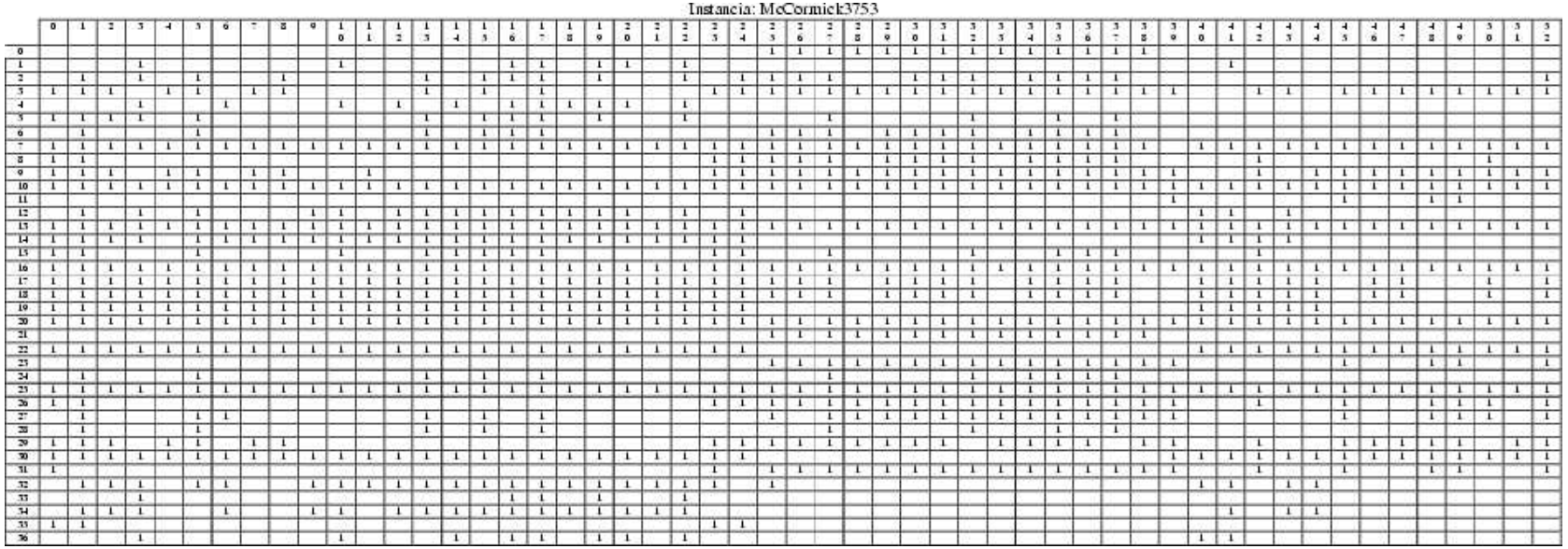
Instancia: King and Nakornchai 30x90

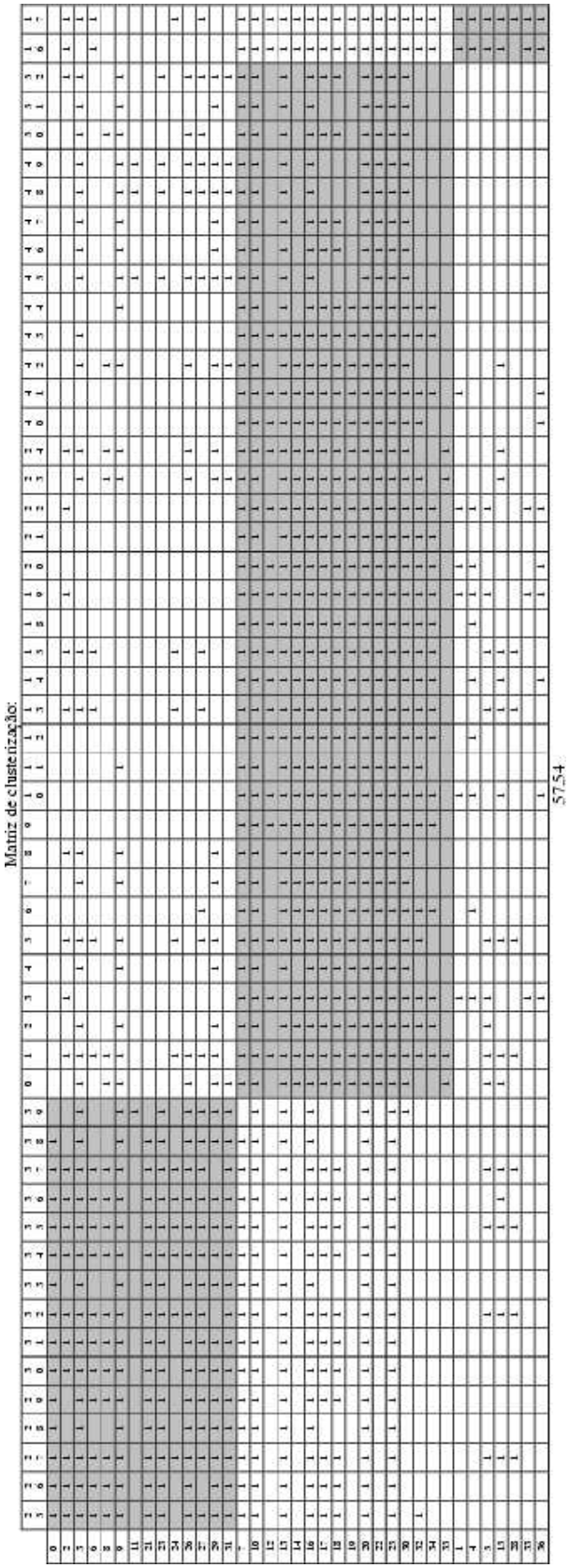
Matriz de clusterização:



4337

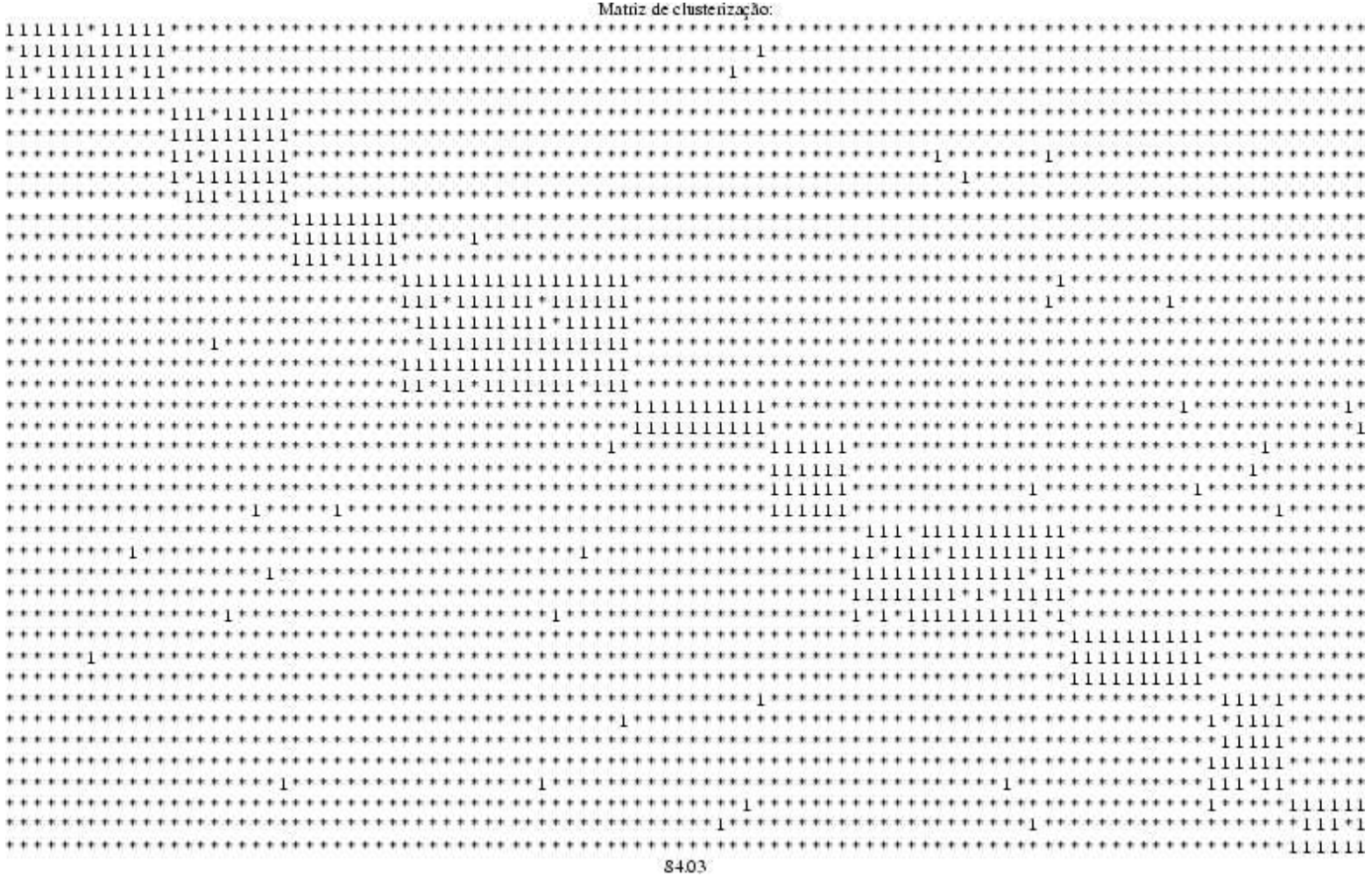
Instancia: McCormick3753




Matriz de clusterização:
57,54

Instancia: Chandrasekaran and Rajagopalan 40x100

Matriz de clusterização:



84.03

Referências Bibliográficas

[1] Aiex, R. Tese de doutorado. *Pontifícia Universidade Católica do Rio de Janeiro, capítulos 4 e 5:77-162*.

[2] Aljaber, N., Baek, W., and Chen, C.-L. A tabu search approach to the cell formation problem. *Computers Industrial Engineering 1*, 32 (1997), 169–185.

[3] Ateme-Nguema, H., and Dao, T.-M. Optimum design of cellular manufacturing systems using hopfield neural network approach: the first step. *Department of Mechanical Engineering. École de Technologie Supérieure, University of Quebec, Montreal, Canadá. Disponível em http://citeseer.ist.psu.edu/cache/papers/cs/26386/http://zzzzfie.engrng.pitt.eduz $ziie2073.pdf/optimum-design-of-cellular.pdf* (1997).

[4] Baker, R. P., and Maropoulos, P. G. An automatic clustering problem suitable for use by a computer-based tool for the design, management and continuous improvement of cellular manufacturing systems. *Computer Integrated Manufacturing Systems 3*, 10 (1997), 217–230.

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

[6] Chandrasekharan, M. P., and Rajagopalan, R. Modroc: an extension of rank order clustering for group technology. *International Journal of Production Research 5*, 24 (1986), 1221–1233.

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

[8] Davis, L. Handbook of genetic algorithms. *VNR computer Library* (1991).

[9] Diaz, B. A., Lozano, S., Racero, J., and Guerrero, F. Machine cell formation in generalized group technology. *Computers & Industrial Engineering*, 41 (2001), 227–240.

[10] Dimopoulos, C., and Zalzala, A. M. S. Recent development in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation 2*, 4 (July 2000), 93–113.

[11] Filho, G. R., and Lorena, L. A. N. Projeto de sistemas de células de manufatura. *Relatório Técnico, LAC/INPE - Institituto Nacional de Pesquisas Espaciais* (1998).

[12] Filho, G. R., and Lorena, L. A. N. A construtive evolutionary approach to the machine-part cell formation problem. *Buildings Competencies for International Manufacturing - Perspectives for developing countries* (2000).

[13] Foulds, L. R., and Neumann, K. A network flow model of group technology. *Mathematical and Computer Modelling*, 38 (2003), 623–635.

[14] Ghosh, S., Mahanti, A., Nagi, R., and Nau, D. S. Manufacturing cell formation by state-space. *Baltzer Journals. Disponível em http://techreports.isr.umd.edu/reports/1993/TR_93-75.pdf* (1993).

[15] Glover, F. Tabu search - part 1. *Orsa Journal on Computing 1*, 1 (1989), 190–206.

[16] Glover, F. Tabu search - a tutorial. *Special Issue on the Practice of Mathematical Programming, Interfaces 20*, 1 (1990), 74–94.

[17] Glover, F. Tabu search - part 2. *Orsa Journal on Computing 2*, 1 (1990), 04–32.

[18] Harhalakis, G., Hilger, J., Nagi, R., and Proth, J. M. Formation of manufacturing cells: An algorithm for minimizing the inter-cell traffic. *Technical Report, University of Maryland. Disponível em http://techreports.isr.umd.edu/reports/1989/TR_89-66.pdf* (1993).

[19] Harhalakis, G., Proth, J. M., and Xie, X. L. Manufacturing cell design using simulated annealing: an industrial application. *Technical Report, University of Maryland. Disponível em http://www.techreports.isr.umd.edu/reports/1990/TR_90-68.pdf* (1990).

[20] Jaumard, B., Labit, P., and Ribeiro, C. C. A column generation approach to cell formation problem in cellular manufacturing. *3rd International Industrial Engineering Conference. Montreal, Canadá* (1999).

[21] Joglekar, P., Chung, Q., and Tavana, M. 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 3*, 5 (2001), 253–268.

[22] Joines, J. A., Culbreth, C. T., and King, R. E. Manufacturing cell design: An integer programming model employing genetic algorithms. *Technical Report, Department of Industrial Engineering, North Carolina State University* (February 1996).

[23] Joines, J. A., Culbreth, C. T., and King, R. E. A hybrid-genetic algorithm for manufacturing cell design. *Technical Report, Department of Industrial Engineering, North Carolina State University* (February 1997).

[24] Joines, J. A., King, R. E., and Culbreth, C. T. A comprehensive review of production-oriented manufacturing cell formation techniques. *Technical Report, Department of Industrial Engineering, Furniture Manufacturing and Management Center, North Carolina State University* (August 1996).

[25] Kirkpatrick, S., Gellat, D. C., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220 (1983), 671–680.

[26] Logendran, R., and Puvanunt, V. Duplication of machines and subcontracting of parts in the presence of alternative cell locations. *Computers Industrial Engineering 1-2*, 35 (1997), 235–238.

[27] Lorena, L. A. N., and Furtado, J. C. Construtive genetic algorithm for clustering problems. *Apresentado no Optimization 98 - Coimbra, Portugal* (July 20-22 1998).

[28] Mahadevan, B., and Srinivasan, G. Software for manufacturing cell formation: Issues and experiences. *GT/CM World Simposium. Columbus, OH* (July 28-30 2003).

[29] Mahadevan, B., and Venkataramanaiah, S. Incorporating product information in the design of a cellular manufacturing system. *Technical Report. Disponível em http://202.41.106.14/ mahadev/pfc.pdf* (April 1989).

[30] Mak, K. L., Wong, Y. S., and Wang, X. X. An adaptative genetic algorithm for manufacturing cell formation. *The International Journal of Advanced Manufacturing Technology*, 16 (2000), 491–497.

[31] Malakooti, B., and Yang, Z. Multiple criteria approach and generation of efficient alternatives for machine-part family formation in group technology. *IIE Transactions*, 34 (2002), 837–846.

[32] Mukattash, A. M., Adil, M. B., and Tahboub, K. K. A modified cell formation grouping measure. *Disponível em http://www.umouncton.ca/cie/conferences/29thconf/29thicce/papers/paper099.pdf* (2001).

[33] Mukattash, A. M., Adil, M. B., and Tahboub, K. K. Heuristic approaches for part assignment in cell formation. *Computers & Industrial Engineering 329-341*, 43 (2002).

[34] Ohta, H., and Nakamura, M. Cell formation with reduction in setup times. *Computers & industrial engineering* (2002).

[35] Ravichandran, K. S., and Rao, K. C. S. A new approach to fuzzy part-family formation in cellular manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 18 (2001), 591–597.

[36] Resende, M. G. C., and Feo, T. A greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6 (1995), 109–133.

[37] Resende, M. G. C., and Gonçalves, J. F. A hybrid genetic algorithm for manufacturing cell formation. *AT&T Labs Research Technical Reports. Disponível em http://www.research.att.com/ mgcr/doc/gagt.pdf* (October 2002).

[38] Resende, M. G. C., and Ribeiro, C. C. Greedy randomized adaptive search procedures. *To appear in State of the Art Handbook in Metaheuristics, F. Glover and G. Kochenberger, eds., Kuwer* (2002).

[39] Ribeiro, J. F. F., and Azevedo, E. M. Tecnologia de grupo e coloração em grafos. *Disponível em http://www.icmc.usp.br/ std-cd/artigos/matematicacomputacional/ope/edsandramaraa.pdf* (2003).

[40] Ribeiro, J. F. F., and Meguelati, S. Organização de um sistema de produção em células de fabricação. *Revista Gestão & Produção 1*, 9 (April 2002), 62–77.

[41] Santos, H. G., Ochi, L. S., and Souza, M. J. F. An efficient tabu search heuristic for the school timetabling problem. *Experimental and Efficient Algorithms: Third International Workshop - WEA 2004, 468-481. Angra dos Reis, Brazil* (May 25-28 2004).

[42] Sarker, B. R. Measures of grouping efficiency in cellular manufacturing systems. *European Journal of Operational Research*, 130 (2000), 588–611.

[43] Shankar, R., Vrat, P., and Soleymanpour, M. A transiently chaotic neural network approach to the design of cellular manufacturing. *International Journal of Production Research 10*, 40 (2002), 2225–2244.

[44] Smed, J., Johnsson, M., Johtela, T., and Nevalainen, O. Group technology in electronic assembly. *Heinonen (ed.), Automation 1999, Helsink, Finland* (1999), 379–384.

[45] Souza, M. J. F. Inteligência computacional para otimização. *Notas de aula da disciplina Inteligência Computacional para Otimização. Departamento de Ciência da Computação - Universidade Federal de Ouro Preto* (2002).

[46] Stawowy, A. Evolutionary strategy for manufacturing cell design. *Proceedings of the 21th International Scientific School Managing Growth of Organisation Information and Technical Issues* (1999).

[47] Sun, D., Lin, L., and Batta, R. Cell formation using tabu search. *Computers Industrial Engineering 3*, 28 (1995), 485–494.

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

[49] Wang, J. 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 (January 1999), 100–104.

[50] Wysk, R. A. Chapter 18 lean - manufacturing. *Manufacturing Systems Course, Department of Industrial and Manufacturing Engineering, Penn State University. Disponível em http://www.engr.psu.edu/cim/ie550lean.pdf* (1997).

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