MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-8432-TDI/774

MELHORAMENTOS NO ALGORITMO GENÉTICO CONSTRUTIVO E NOVAS APLICAÇÕES EM PROBLEMAS DE AGRUPAMENTO

Geraldo Ribeiro Filho

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Luiz Antonio Nogueira Lorena, aprovada em 06 de dezembro de 2000.

INPE
São José dos Campos
2001

681.3.019

RIBEIRO FILHO, G.

Melhoramentos no algoritmo genético construtivo e novas aplicações em problemas de agrupamento / G. Ribeiro Filho – São José dos Campos: INPE, 2000.

129p – (INPE-8432-TDI/774).

1.Algoritmo genético. 2.Otimização. 3.Teoria dos grafos. 4.Manufatura. 5.Tabela de horários. I.Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Doutor** em **Computação Aplicada.**

Dr. Solon Venâncio de Carvalho
Presidente/INPE-SJCampos/SP

Dr. Luiz Antônio Nogueira Lorena
Orientador/INPE-SJCampos/SP

Drª Sandra Aparecida Sandri
Membro da Banca/INPE-SJCampos/SP

Dr. Horácio Hideki Yanasse
Membro da Banca/INPE-SJCampos/SP

Dr. Pedro Paulo Balbi de Oliveira
Convidado/UNIVAP-SJCampos/SP

Drª Cintia Rigão Scrich
Convidada/UNIP-Campinas/SP

Candidato (a) : Geraldo Ribeiro Filho

São José dos Campos, 06 de dezembro de 2000.

“The relationship between the teaching and research is the same as between the confession and sin: If you have not sinned, than you have nothing to confess!”

Anônima

AGRADECIMENTOS

Ao Prof. Dr. Luiz Antonio Nogueira Lorena, que orientou este trabalho sempre de modo paciente, compreensivo e amigo.

Aos membros da banca examinadora pela predisposição em analisar este trabalho e pelas sugestões recebidas.

E, em especial a minha família pelo apoio, incentivo e pela compreensão durante todo o tempo do curso.

RESUMO

Os Algoritmos Evolutivos são tema de estudo há décadas e se baseiam na evolução através de gerações de populações cujos indivíduos são estruturas que representam possíveis soluções de um problema. Os chamados Algoritmos Genéticos estão nesse grupo de algoritmos e sua eficácia na aplicação a problemas de otimização combinatória está registrada em muitos trabalhos científicos. Recentemente tem sido objeto de estudo o Algoritmo Genético Construtivo (*AGC*), que trabalha com uma população de tamanho variável ao longo das gerações, não somente formada por estruturas, mas também por partes de estruturas. Este trabalho contribui com esse estudo apresentando uma adaptação do *AGC* para trabalhar com uma população formada apenas por partes de estruturas, criando estruturas não somente através da combinação dessas partes, mas também com complementação das partes selecionadas, e ainda utilizando um processo de busca local como mutação aplicada às estruturas. O processo mantém salva apenas a melhor estrutura eventualmente formada com a combinação ou complementação das partes na população. O estudo foi feito com a aplicação do *AGC* a três problemas de otimização muito estudados: Coloração de Grafos, Projeto de Células de Manufatura e Formação de Horários Escolares. O problema de Coloração de Grafos tem muitas aplicações práticas, essencialmente na formação de grupos de objetos sem incompatibilidades entre si. O problema de Projeto de Células de Manufatura está presente em ambientes de produção de peças utilizando máquinas, tratando de agrupar essas máquinas de modo a criar células de produção em que peças são completamente produzidas dentro da célula, evitando o transporte de produtos semi-acabados. O problema de Formação de Horários Escolares (*Timetabling*) tem importância evidente em instituições de ensino e sua automação se justifica por uma constante e cíclica demanda. Todos os problemas estudados foram considerados como problemas de formação de agrupamentos. O código do *AGC* foi escrito especificamente para cada problema a partir de uma estrutura básica e foi executado em microcomputadores e estações de trabalho, produzindo bons resultados para instâncias tomadas da literatura, criadas para testes ou mesmo instâncias reais.

CONSTRUCTIVE GENETIC ALGORITHM IMPROVEMENTS AND NEW CLUSTERING PROBLEMS APPLICATIONS

ABSTRACT

Evolutionary Algorithms has been a research subject for decades and are based on evolving populations of possible solutions for a problem along generations. Genetic Algorithms belong to this group and many scientific works have registered their efficiency applied to combinatorial optimization problems. Recently, the Constructive Genetic Algorithm (*CGA*) has been studied. This algorithm works with a variable size population over the generations, the population is formed not only by complete problem solutions but also problem solutions parts. This work contributes to the study of such algorithm by introducing a *CGA* adaptation that works with populations composed only by solution parts, creating complete solution not only by parts combination by also by parts complementation, and finally using a local search method as a mutation process over the complete solutions. This process keeps the best solution eventually found. The study was made using three very known optimization problems: Graph Coloring, Manufacturing Cell Design and School Timetabling. The Graph Coloring problem has many practical applications. It can be applied every time a set formed by objects with some incompatibility among its elements has to be partitioned into subsets with no incompatibilities inside. The Manufacturing Cell Design problem importance resides in planning environments to produce parts using machines, forming machine cells to completely produce parts, reducing the movement of non-completely produced parts. The School Timetabling problem has obvious importance for education institutions and its cyclic demand justify automation by using algorithms. All problems considered were seen as clustering problems. The *AGC* code was specifically written for each problem using a common base and was executed in microcomputers and workstations, producing good results for test instances taken from the literature, instances specially created for tests, and instances taken from the real word.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

A Otimização Combinatória se caracteriza pelo tratamento de problemas cujo conjunto de possíveis soluções é finito e enumerável. Problemas que, portanto, poderiam ter sua melhor solução obtida até por simples enumeração e avaliação de cada possibilidade. No entanto, dependendo das caraterísticas do problema, o conjunto de soluções possíveis pode ser tão vasto que se torna impossível avaliar cada um de seus elementos em tempo aceitável. A resolução de problemas desse tipo, em geral, requerem o uso de computadores e algoritmos especiais.

São dois os tipos básicos de algoritmos, os métodos exatos e as heurísticas. Os métodos exatos garantem a obtenção da melhor solução de um determinado problema, a chamada *solução ótima*, e as heurísticas fazem buscas no conjunto de soluções sem garantir a otimalidade da solução final obtida. Muitos problemas têm características que inviabilizam o uso, quando estes existem, dos métodos exatos. Nesse caso, a opção é o uso das heurísticas. Existem heurísticas mais gerais que por meio de adaptações podem ser usadas para vários problemas; são as chamadas meta-heurísticas (Michalewicz e Fogel, 2000), entre as quais podemos citar *Simulated Annealing* (Johnson *et al*., 1991), Busca Tabu (Glover e Laguna, 1997) e os Algoritmos Genéticos (Michalewicz , 1996).

Os Algoritmos Genéticos (*AGs*) fazem parte de um grupo conhecido como Algoritmos Evolutivos, e são baseados num processo coletivo de aprendizagem dentro de uma população de indivíduos, cada um dos quais representando um ponto no espaço de busca de soluções para um dado problema. A população é inicializada e evolui através de gerações com o uso dos operadores de seleção, reprodução (também neste trabalho chamada de recombinação) e mutação. Durante o procedimento, é feita uma avaliação da qualidade (adaptação) dos indivíduos e essa informação é usada para conduzir o processo evolutivo, favorecendo a seleção de indivíduos bem adaptados, para assim gerar mais indivíduos também bem adaptados. O mecanismo de recombinação permite

mesclar informações de indivíduos de uma geração e passá-las aos seus descendentes, e um processo de mutação introduz inovação na população.

A teoria tradicional dos *AGs* considera que seu funcionamento se deve à ênfase dada a bons *blocos construtivos* na geração das soluções. Essa idéia foi formalizada através da introdução da definição de *esquema* (Holland, 1975). Enquanto as soluções são representadas por *estruturas* formadas por uma seqüência de símbolos de um alfabeto, os esquemas equivalem a sequências desses símbolos, mas com posições sem símbolo definido, em que qualquer símbolo do alfabeto pode ser colocada. Sendo assim, podemos considerar as estruturas com instâncias dos esquemas.

Neste trabalho fazemos uso do Algoritmo Genético Construtivo (*AGC*), baseado nos trabalhos iniciais de Lorena e Lopes (1996) e depois aperfeiçoado em trabalhos subseqüentes de Ribeiro Filho (1997) e Furtado (1998). O *AGC*, na sua formulação inicial, começa com uma população de esquemas. Os esquemas modelam propriedades estruturais das soluções do problema e são avaliados através de funções que determinam quão promissor é cada esquema para formação de *boas* estruturas. Os melhores esquemas são recombinados com outros para produzir novos esquemas e estruturas através de sucessivas gerações. Um critério de poda elimina esquemas ou estruturas que não obtiverem boa avaliação. Enquanto nos *AGs* os esquemas são avaliados indiretamente pelas soluções que produzem, no *AGC* são diretamente avaliados através de duas funções que permitem a seleção de *bons* esquemas e/ou boas soluções.

Nas pesquisas que levaram a este trabalho, o *AGC* foi modificado para trabalhar com uma população formada exclusivamente por esquemas, sem estruturas. Durante as gerações do processo evolutivo, a recombinação dos esquemas na população produz mais esquemas, mas a partir de certo ponto na evolução também começa a produzir estruturas. Apenas a melhor estrutura produzida até cada momento é mantida salva e, ao final do processo, é apresentada como a solução final obtida para o problema. A razão para manter a população formada apenas por esquemas está no fato de que, ao longo das

gerações, as estruturas passam a ser dominantes no processo de seleção, impedindo que bons esquemas ainda não instanciados em estruturas tenham chance de se reproduzir.

A organização deste trabalho traz uma introdução aos Algoritmos Evolutivos no segundo Capítulo, em que são comentados os princípios da evolução natural, a forma original desses algoritmos, seus principais componentes, e algumas de suas variações.

No terceiro Capítulo é apresentado o *AGC*. São expostos os fundamentos do algoritmo e seus principais componentes, como: forma de representação de esquemas e estruturas, população inicial, método de avaliação de adaptação, operadores de seleção, recombinação e mutação, além do critério de eliminação de indivíduos da população.

Os três Capítulos subseqüentes apresentam aplicações do *AGC* a três problemas: Coloração de Grafos, Projeto de Células de Manufatura e Programação de Horário Escolar. Em cada um desses Capítulos é feita uma descrição do problema, da representação usada nos esquemas e do processo de criação da população inicial; são também descritos: o processo de recombinação, as funções para avaliação de adaptação dos esquemas e o processo de mutação, específico de cada problema; são ainda apresentados os resultados dos testes computacionais e conclusões dos experimentos.

Finalmente, são apresentadas as conclusões finais do trabalho, e as expectativas para futuros estudos.

CAPÍTULO 2

INTRODUÇÃO AOS ALGORITMOS EVOLUTIVOS

Neste Capítulo tem-se como objetivo mostrar como os princípios da evolução natural são utilizados na criação de algoritmos para obtenção de soluções de problemas de otimização combinatória. O Capítulo apresenta uma introdução aos Algoritmos Evolutivos, comentando seus principais componentes e algumas de suas variações. São utilizados, como exemplo, três problemas clássicos de otimização. São ainda relacionadas fontes de referência sobre o assunto.

2.1 – A EVOLUÇÃO NATURAL E ALGORITMOS EVOLUTIVOS

Os princípios da *sobrevivência dos mais adaptados* e da *evolução natural*, descritos por Charles Darwin em meados do século XIX, baseiam-se na idéia de que, na natureza, um processo evolutivo ocorre quando quatro condições são satisfeitas (Koza, 1992):

- um indivíduo tem habilidade de reproduzir a si mesmo;
- há uma população desses indivíduos;
- há variedade entre os indivíduos dessa população; e
- uma diferença na habilidade de sobrevivência no ambiente está associada com essa variedade.

Na natureza, a variedade se manifesta como diferenças na composição dos cromossomos dos indivíduos que formam uma população. Cromossomos são seqüências de símbolos de um alfabeto que, na natureza, têm quatro elementos associados às moléculas: adenina (A), citosina (C), guanina (G) e timina(T).

As diferenças nos cromossomos se traduzem em diferenças tanto na estrutura quanto no comportamento dos indivíduos, o que se reflete em diferenças na taxa de sobrevivência

e reprodução. Indivíduos que têm mais habilidade para executar tarefas em seu ambiente são considerados mais adaptados a esse ambiente, sobrevivendo e se reproduzindo em taxas maiores do que os menos adaptados.

Ao longo de gerações há um processo de renovação numa população. Indivíduos mais antigos e/ou menos adaptados são eliminados, e novos indivíduos são criados. Com o tempo, a população passa a ter indivíduos cujos cromossomos refletem maior habilidade para executar determinadas tarefas, sobreviver e se reproduzir. Essas são características de um processo de seleção de indivíduos ao longo de gerações.

Sendo assim, a estrutura e comportamento dos indivíduos numa população se alteram ao longo do tempo por causa de um processo de seleção natural. Quando essas alterações são visíveis e mensuráveis, diz-se que a população evoluiu.

Há mais de quatro décadas, pesquisadores começaram a estudar sistemas evolutivos, acreditando que o princípio da evolução natural poderia ser aproveitado na busca de soluções para determinados problemas.

A idéia básica dos algoritmos evolutivos é trabalhar com um conjunto de estruturas representando soluções e, utilizando operações inspiradas na evolução e seleção natural, iterativamente transformar esse conjunto gerando novas e melhores soluções a partir das anteriores.

A melhor solução encontrada na população no final das iterações é considerada como a solução final do problema dada pelo algoritmo.

Uma analogia pode ser feita imediatamente com as características de situações encontradas na natureza: o conjunto de estruturas é considerado uma população; as soluções são indivíduos dessa população e as estruturas representam seus cromossomos; a qualidade das soluções é considerada como adaptação dos indivíduos.

A história da aplicação desses algoritmos remonta ao início da segunda metade do século XX e nos mostra que as pesquisas eram feitas independentemente e produziam muitas vezes resultados semelhantes e até redundantes (Fogel, 1998).

Muitos autores criaram procedimentos ligeiramente diferentes uns dos outros e lhes deram nomes específicos como Estratégias Evolutivas (Rechenberg, 1965), Programação Evolutiva (Fogel *et al*., 1966; Fogel, 1991) e Algoritmos Genéticos (Holland, 1975; Goldberg, 1989). Em seguida, técnicas mais especializadas foram desenvolvidas em associação com determinadas estruturas de dados, como por exemplo Programação Genética (Koza, 1992).

Entretanto, mais recentemente, em trabalhos de Michalewicz (1996) e de Michalewicz e Fogel (2000), há evidências teóricas e empíricas de que nenhuma das abordagens "canônicas" originais desses procedimentos pode chegar a algum resultado que não possa ser obtido com outros procedimentos não-evolutivos.

Com o interesse crescente por esses algoritmos, idéias têm sido aproveitadas, trocadas e modificadas em todas essas abordagens. Não há base científica para discriminação entre procedimentos evolutivos, e o termo "Algoritmos Evolutivos" tem sido usado para descrever quaisquer procedimentos iterativos baseados em seleção e variação de populações.

Hoje, os Algoritmos Evolutivos formam uma classe de algoritmos muito estudada, com muitas variações e aplicações bem sucedidas na resolução de problemas em várias áreas da ciência (Michalewicz, 1996; Michalewicz e Fogel, 2000).

2.2 – ALGORITMOS GENÉTICOS

Resultado de seus primeiros estudos na década de 1960, e posteriormente aperfeiçoada por seus alunos, uma abstração da evolução biológica chamada Algoritmo Genético

(*AG*) foi apresentada por Holland (1975) em seu livro: *Adaptation in Natural and Artificial Systems.*

Como um algoritmo evolutivo, o *AG* processa estruturas que são representações de soluções de um problema. Essas estruturas são formadas por seqüências de símbolos de um determinado alfabeto. As soluções são consideradas indivíduos, e as estruturas são representações de seus cromossomos.

Uma função associa as estruturas a valores numéricos que medem a qualidade das soluções. A formulação dessa função reflete os objetivos de otimização do problema. A qualidade das soluções é considerada como adaptação dos indivíduos.

Iterativamente, o *AG* transforma um conjunto com um número fixo de estruturas, também chamado de população de indivíduos, em um novo conjunto, ou nova população, usando três operações genéticas definidas com base nos princípios fundamentais da evolução natural: seleção, reprodução e mutação.

O processo de seleção pode ser feito de várias maneiras, mas normalmente as estruturas são selecionadas para a formação de uma nova população de modo proporcional à sua adaptação. Estruturas mais adaptadas são selecionadas com mais freqüência do que as menos adaptadas.

As estruturas selecionadas podem ser então reproduzidas de várias maneiras: a reprodução pode ser feita com simples cópias individuais (clonagem), isto é, estruturas selecionadas são copiadas na nova população; ou com associações a outras estruturas (cruzamento). Nessas associações, duas estruturas selecionadas são combinadas para geração de novos indivíduos.

Um processo de mutação sobre os novos indivíduos altera a formação de suas estruturas. O objetivo da mutação é manter variedade na nova população.

Tanto a escolha da forma de reprodução quanto da aplicação ou não da mutação são feitas de acordo com probabilidades que são parâmetros do *AG*.

Os princípios básicos do *AG* podem ser aplicados de diferentes maneiras, produzindo algoritmos ligeiramente diferentes, formando uma classe de algoritmos que passou a ser conhecida como Algoritmos Genéticos (*AGs*).

O pseudocódigo da Figura 2.1 mostra uma dessas possibilidades, extraída de um trabalho de Michalewicz e Fogel (2000).

```
Criar população inicial com M indivíduos
Enquanto critério de parada não for satisfeito
    Avaliar cada indivíduo
    I ← 0
    Enquanto I < M
        Escolher reprodução, cruzamento ou mutação de acordo com probabilidades Pr, Pc, e Pm.
        Se clonagem
            Selecionar indivíduo com base em sua adaptação
            Copiar indivíduo para nova população
        Se cruzamento
            Selecionar dois indivíduos com base em sua adaptação
            Cruzar os indivíduos
            Inserir os novos indivíduos na nova população
            I ← I + 1
        Se mutação
            Selecionar indivíduo com base em sua adaptação
            Aplicar mutação
            Inserir indivíduo mutante na nova população
        I ← I + 1
    população ← nova população
Exibir melhor indivíduo
```

Fig. 2.1 – Exemplo de Algoritmo Genético.

Podemos fazer algumas considerações sobre o exemplo:

- A população inicial pode ser gerada tanto aleatoriamente quanto através de um outro método específico para tentar gerar indivíduos com alguma qualidade;

- O processo termina de acordo com um critério de parada que pode ser: atingir um limite de gerações previamente estipulado; encontrar uma solução de qualidade satisfatória; ou estagnação da população; ou ser atingido um nível de estagnação do processo evolutivo;

- O número *M* de indivíduos da população foi mantido fixo em todas as gerações;

- A escolha da maneira de criar novos indivíduos foi feita com base nas probabilidades: P_r para reprodução por cópia; P_c para cruzamento; e P_m para mutação. Pode-se, por exemplo: ter $P_r=0.30$, $P_c=0.60$ e $P_m=0.10$; gerar-se um número aleatório *p* entre *0* e *1*; se $p \leq 0.30$ escolher reprodução por cópia, se $0.30 < p \leq 0.90$ escolher cruzamento, e se $p > 0.90$ escolher mutação;

- A seleção de indivíduos na população atual deve ser feita com base em sua adaptação, isto é, indivíduos mais bem adaptados devem ter maior chance de serem selecionados;

- O cruzamento entre dois indivíduos selecionados deve produzir dois novos indivíduos, que têm duas partes, uma de cada indivíduo selecionado;

- A mutação deve inserir na nova população um novo indivíduo, produzido com alterações nos símbolos da seqüência de representação do indivíduo selecionado; e

- O indivíduo mais adaptado da última geração será considerado como a solução final encontrada pelo *AG* para o problema abordado.

Essa forma de *AG* é apenas uma possibilidade, e variações podem ser facilmente criadas. Pode-se, por exemplo, aplicar a mutação sobre os indivíduos obtidos após a reprodução, seja ela uma simples cópia ou resultado de cruzamento.

2.3 – EXEMPLOS DE APLICAÇÃO DE ALGORITMOS EVOLUTIVOS

A aplicação de qualquer algoritmo evolutivo tem como aspectos essenciais a escolha da forma de representação dos cromossomos, da forma de criação da população inicial, da formulação da função de avaliação da adaptação, e dos processos de seleção e geração de novos indivíduos (reprodução e mutação).

Nesta Seção, veremos como esses aspectos podem variar na abordagem de três problemas de otimização: satisfabilidade (*SAT*), caixeiro viajante (*CV*) e programação não-linear (*PNL*) (Michalewicz e Fogel, 2000).

No problema *SAT* é dada uma expressão lógica com variáveis x_i, $i=1,2,\ldots,n$, e seus respectivos complementos $\overline{x_i}$. Por exemplo:

$$F(X) = \left(x_{17} \vee \overline{x_{37}} \vee x_{35}\right) \wedge \left(\overline{x_{11}} \vee \overline{x_{56}}\right) \wedge \ldots \wedge \left(x_2 \vee x_{43} \vee \overline{x_{77}} \vee x_{89}\right) \qquad (2.1)$$

O problema é encontrar um vetor $X=(x_1, x_2, \ldots, x_n)$ de modo que $F(X)=$*verdadeiro*. Como as variáveis são lógicas, elas admitem apenas dois valores: *verdadeiro* ou *falso*.

No problema do caixeiro viajante (*CV*) é dado um conjunto de cidades de modo que sempre seja possível viajar de uma cidade à qualquer outra, e que as distâncias entre elas sejam fixas e conhecidas. O objetivo é encontrar uma seqüência de cidades de forma que o caixeiro viajante visite cada cidade uma única vez, fazendo o menor trajeto possível.

Um problema particular de programação não-linear (*PNL*) consiste em maximizar uma função não-linear *G2(X)*, definida abaixo, cujo máximo global é desconhecido, mas sabe-se que está próximo da origem.

O problema tem a seguinte formulação:

$$Max \quad G2(X) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{2.2}$$

$$Suj. \quad \prod_{i=1}^{n} x_i = 0.75$$

$$\sum_{i=1}^{n} x_i \leq 7.5n$$

$$0 \leq x_i \leq 10 \quad , \quad 1 \leq i \leq n$$

2.3.1 – REPRESENTAÇÃO

A maneira como as soluções são representadas é de fundamental importância para o sucesso dos algoritmos evolutivos. A forma de representação pode variar de muitas maneiras e basicamente depende da natureza do problema tratado.

De modo geral, as formas de representação que têm recebido mais atenção recentemente são aquelas que consideram características específicas do problema. Koza (1992), por exemplo, propôs uma forma de representação usando codificação em árvores para Programação Genética, e Fleurent e Ferland (1994), em sua aplicação para Coloração de Grafos, utilizaram seqüências de inteiros representando as cores associadas aos vértices do grafo.

Um elemento crucial nos estudos dos *AGs* é o conceito de *esquema*. Enquanto uma estrutura que representa um cromossomo é uma seqüência definida sobre um alfabeto *A*, um esquema é uma sequência de mesmo comprimento, mas definida sobre o alfabeto

AÈ{#}. O símbolo # é denotado neste trabalho como *do-not-care*, e quando presente numa posição de um esquema, significa que nessa posição qualquer símbolo do alfabeto *A* pode ser admitido.

O funcionamento do *AG* está intimamente ligado ao conceito de esquema e à forma com que esses esquemas se propagam através das gerações no processo evolutivo (Holland, 1975). Os esquemas são considerados blocos construtivos cuja propagação e combinação ao longo das gerações fundamentam hipóteses sobre o desempenho dos *AGs* (Goldberg, 1989).

Propagando e combinando esquemas pequenos e bem adaptados, estaríamos reduzindo a complexidade do problema. Ao invés de construirmos longas seqüências bem adaptadas tentando uma enorme quantidade de combinações possíveis, construiríamos novas seqüências cada vez mais bem adaptadas com o uso de partes de seqüências anteriores.

No caso do problema *SAT*, as variáveis envolvidas são lógicas e portanto podem assumir apenas um entre dois possíveis valores. Uma escolha natural para representar os vetores que são soluções do problema é simplesmente uma seqüência binária com *n* elementos, cada um deles correspondente a uma das *n* variáveis do problema.

O problema do *CV* tem caraterísticas bem diferentes. Uma solução é um trajeto em que o caixeiro passa por todas as cidades, uma após a outra, numa determinada ordem. A representação mais natural é uma seqüência de símbolos, cada um representando uma cidade na ordem em que elas são visitadas pelo caixeiro.

No caso do problema *PNL* a solução é um vetor de *n* valores reais e esta é exatamente a forma de representação mais imediata. As estruturas são seqüências de valores reais, cada um deles correspondendo a uma variável.

Neste trabalho, uma forma de representação geral para problemas de agrupamento utilizando fortemente a idéia de esquema foi empregada e será descrita nos próximos Capítulos.

2.3.2 – POPULAÇÃO INICIAL

Para iniciar um processo evolutivo, devemos gerar de alguma forma uma população inicial. O número de indivíduos dessa população, que em muitos algoritmos se mantém fixo durante todo o processo evolutivo, é considerado um parâmetro desses algoritmos e sua determinação é feita de modo empírico.

A geração dos indivíduos da população inicial pode ser feita de diferentes maneiras, que dependem fortemente da forma de representação adotada.

No caso do problema *SAT*, os indivíduos são seqüências binárias de *n* símbolos. Um processo simples poderia montar as seqüências gerando um símbolo de cada vez com 50% de chance de esse símbolo ser *1* ou *0* (*verdadeiro* ou *falso*).

No caso de problema do *CV*, um processo poderia gerar aleatoriamente permutações de todos os símbolos que representam as cidades. Cada permutação gerada seria uma seqüência de cidades na população inicial.

Para o problema *PNL* podemos gerar vetores de *n* valores reais, escolhendo aleatoriamente cada valor no intervalo [0,10]. Como há restrições que devem ser observadas, ao final da geração de um vetor, verificamos se alguma restrição é desrespeitada e, em caso afirmativo, o vetor é desconsiderado e um novo vetor é gerado. O processo continua até que todos os vetores da população inicial sejam gerados.

2.3.3 – AVALIAÇÃO DA ADAPTAÇÃO

A formulação de uma função que associe as estruturas da população a valores numéricos que são usados para avaliar sua adaptação é feita de acordo com as características de cada problema abordado.

No problema *SAT* a função *F(X)* tem apenas dois resultados numéricos possíveis, *1* ou *0*, o que a torna inadequada para avaliar a qualidade das seqüências binárias da população. Se a solução ótima fosse conhecida, a qualidade de uma seqüência da população poderia ser medida por uma função que avaliasse a diferença entre essa seqüência e a solução ótima. Como obviamente a solução ótima não é conhecida, poderíamos avaliar a qualidade de uma seqüência montando a expressão de *F(X)* com essa seqüência e contando quantas conjunções da forma $\left(x_i \wedge x_j \wedge ...\right)$ são verdadeiras (têm valor *1*).

No caso do *CV*, a escolha imediata para avaliar a adaptação de uma permutação de cidades seria utilizar uma medida com base na soma das distâncias percorridas com essa permutação. Quanto maior a distância total, pior a avaliação.

No problema *PNL*, a avaliação das seqüências de valores reais pode ser feita com o uso da própria função *G2(X)* que está sendo maximizada. Quanto às restrições, esse valor pode receber penalidades proporcionais ao desvio da seqüência em relação à região viável. Quanto mais distante a seqüência estiver das bordas da região viável, maior será a penalidade.

O algoritmo utilizado neste trabalho e detalhado no próximo Capítulo utiliza uma forma de avaliação comum tanto para esquemas quanto para estruturas. No entanto, do mesmo modo que nos exemplos citados, a formulação das funções é totalmente dependente do problema abordado.

2.3.4 – OPERAÇÃO DE SELEÇÃO

Nos trabalhos iniciais com algoritmos evolutivos a seleção dos indivíduos para geração da nova população era feita com base na evolução natural, isto é, indivíduos mais bem adaptados possuem maior probabilidade de serem selecionados para se reproduzir. Nessa técnica, denominada *seleção por roleta* (Holland, 1975), a probabilidade de um indivíduo ser selecionado é proporcional ao valor da sua adaptação dividido pela soma da adaptação de todos os indivíduos da população.

Com o passar do tempo, novos mecanismos foram propostos. Na chamada *seleção por torneio* (Goldberg e Deb, 1991), duas ou mais estruturas são escolhidas aleatoriamente na população e, de acordo com uma certa probabilidade, ou o indivíduo mais adaptado ou o menos adaptado, apenas um deles, será selecionado para reprodução, e ambos podem, eventualmente, ser novamente selecionados.

Outras formas de seleção incluem a *seleção com ordenação* (Baker, 1985), em que os indivíduos da população são ordenados de acordo com sua adaptação.

Há também a possibilidade da forma de seleção variar ao longo da busca, como por exemplo numa abordagem denominada *seleção de Boltzmann* (Goldberg, 1990). Mesmo em qualquer esquema proporcional de seleção a pressão seletiva varia ao longo da busca.

Nos exemplos de problemas citados neste Capítulo, *SAT*, *CV* e *PNL*, qualquer um desses métodos de seleção pode ser utilizado.

O mecanismo usado neste trabalho e descrito nos Capítulos seguintes é semelhante à seleção por torneio mesclada com um método de ordenação.

2.3.5 – OPERAÇÃO DE REPRODUÇÃO

A reprodução dos indivíduos selecionados pode ser feita com uma simples cópia inserida na nova população sem nenhuma modificação, ou então com modificações gerando novos indivíduos.

No problema *SAT* as estruturas são seqüências binárias, e um processo de cruzamento de dois indivíduos selecionados é usado para gerar dois novos indivíduos. O processo consiste basicamente em escolher um ponto de cruzamento em duas estruturas selecionadas e, usando partes dessas estruturas definidas pelo ponto de cruzamento, gerar dois novos indivíduos formados com partes alternadas.

No entanto, com a representação das estruturas sendo feita de formas variadas, isto é, formas que consideram as características do problema, os métodos de reprodução também podem ser variados, geralmente considerando características do problema e da representação, mantendo correta a codificação das estruturas geradas.

No problema do *CV* podemos gerar um novo indivíduo a partir de dois indivíduos selecionados, analisando cidade a cidade nas seqüências selecionadas. Para cada posição da permutação escolhemos aleatoriamente uma entre as duas cidades daquela posição nas duas seqüências. Obviamente devemos ter cuidado para respeitar as restrições do problema. Se uma das cidades daquela posição já tiver sido usada na seqüência sendo montada, a outra cidade é imediatamente escolhida, e se ambas as cidades já tiverem sido usadas, uma das cidades restantes pode ser escolhida aleatoriamente.

No caso de seqüências de valores reais, como no problema de *PNL*, uma maneira de gerar novos indivíduos é adicionar a cada elemento da seqüência selecionada um valor aleatório com distribuição Gaussiana. Escolhendo, por exemplo, a variância com valor um e média com valor zero, adicionamos a cada elemento da seqüência um valor real

diferente, independente dos demais, e obtemos um novo indivíduo que, na média, não será muito diferente da seqüência selecionada. Isso ocorre porque a probabilidade de que cada novo valor real seja muito diferente do seu correspondente na seqüência é considerável apenas quando os valores da variável não estão próximos da média.

2.3.6 – OPERAÇÃO DE MUTAÇÃO

O papel da mutação é introduzir variedade e evitar a homogeneidade da população ao longo das gerações. Os procedimentos de mutação podem variar bastante e dependem das formas de representação escolhidas.

No caso de seqüências binárias como no problema *SAT*, um processo bem simples consiste em escolher aleatoriamente uma posição da seqüência e inverter seu valor.

Formas mais sofisticadas podem ser utilizadas para que a mutação produza uma melhoria local da solução. Nesses casos, as características dos problemas podem ser exploradas.

No problema de *CV* podemos alterar uma permutação de cidades eliminando trechos do trajeto que cruzam sobre si mesmos. Certamente a soma das distâncias irá diminuir e, com a representação em forma de permutações, essa tarefa é feita com grande facilidade, bastando selecionar dois pontos da permutação com cruzamento entre eles e simplesmente inverter a ordem das cidades entre esses dois pontos.

Para o problema de *PNL* podemos, por exemplo, usar uma perturbação Gaussiana em apenas uma posição do vetor de números reais. Essa posição pode ser escolhida aleatoriamente.

Nas aplicações descritas nos próximos Capítulos a mutação é feita com técnicas que dependem fortemente do problema abordado e da representação utilizada.

Na literatura encontramos técnicas de reparação de inviabilidade de estruturas (Michalewicz , 1996; Michalewicz e Fogel, 2000). Se, por exemplo, a reprodução gera indivíduos que eventualmente desrespeitam restrições do problema, a mutação pode ser encarregada de corrigir esses desvios, evitando o descarte de indivíduos e a necessidade de repetir a reprodução.

2.3.7 – PARÂMETROS DOS ALGORITMOS

Pode-se observar que os Algoritmos Evolutivos têm seu funcionamento baseado em uma série de parâmetros. São parâmetros: o tamanho da população, as probabilidades de reprodução e mutação, e até mesmo o critério de parada.

A escolha de valores para os parâmetros depende do problema que está sendo abordado, e também da instância tratada. Estudos sobre o controle dos parâmetros de *AGs* podem ser vistos no trabalho de Grefenstette (1986) e Back (1992). Ajustes de probabilidades das operações dos *AGs* podem ser vistos nos trabalhos de Julstrom (1995) e de Smith e Fogarty (1996).

Michalewicz e Fogel (2000) mostram uma classificação das técnicas de determinação de parâmetros (Figura 2.2).

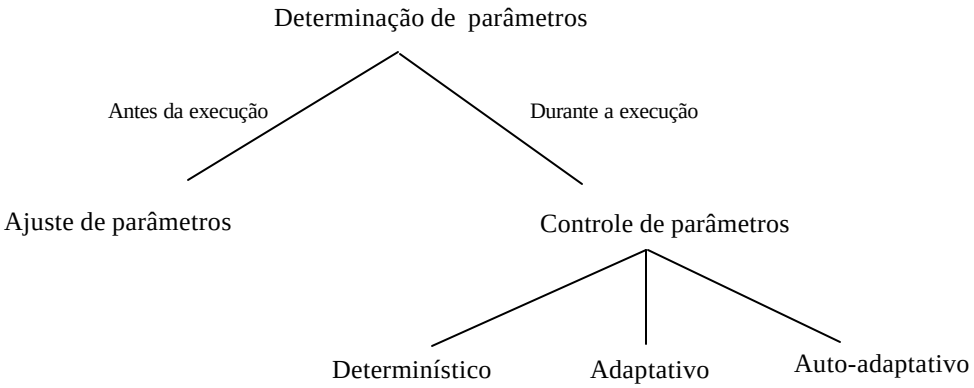

Fig. 2.2 – Classificação de técnicas de ajustes de parâmetros.

De acordo com essa classificação, antes da execução o ajuste é feito de modo empírico. Durante a execução o ajuste pode ser feito do modo determinístico, em que os valores dos parâmetros são alterados de acordo com alguma regra predeterminada, como por exemplo quando um certo número de iterações é atingido; pode ser feito também de modo adaptativo, isto é, de acordo com informações obtidas como retorno do próprio processo evolutivo; pode ainda ser feito de modo auto-adaptativo, em que informações sobre os parâmetros são codificadas dentro dos indivíduos e também sofrem reprodução e mutação. Os parâmetros evoluem juntamente com a população.

2.4 – CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste Capítulo foram apresentados os princípios básicos da evolução natural e como eles são utilizados numa classe de algoritmos conhecida como Algoritmos Evolutivos.

Foi especialmente abordada uma variação de algoritmos evolutivos conhecida como Algoritmos Genéticos, em que as bases dos princípios da evolução natural são facilmente identificadas.

Os principais aspectos da aplicação dos Algoritmos Evolutivos foram abordados com o auxílio de três problemas: satisfabilidade (*SAT*), caixeiro viajante (*CV*) e programação não-linear (*PNL*).

Finalmente, técnicas de controle dos parâmetros dos algoritmos evolutivos foram citadas.

Os próximos Capítulos descrevem e mostram aplicações de um algoritmo evolutivo conhecido como Algoritmo Genético Construtivo (*AGC*).

CAPÍTULO 3

ALGORITMO GENÉTICO CONSTRUTIVO

O Algoritmo Genético Construtivo (*AGC*), descrito neste Capítulo, teve origem no trabalho de Lorena e Lopes (1996) e foi depois aperfeiçoado nos estudos de Ribeiro Filho (1997) e Furtado (1998). A proposta básica do *AGC* é avaliar de modo comum tanto *estruturas* representantes de soluções de um problema, quanto partes dessas estruturas, chamadas *esquemas*. Enquanto outros algoritmos avaliam os indivíduos com base numa única função (chamada função de avaliação), o *AGC* utiliza duas funções, mapeando o espaço de estruturas e esquemas no R^+. O processo comum de dupla avaliação de adaptação, da consideração conjunta não só dos esquemas mas também das estruturas, e do tamanho variável da população constituem os pontos centrais no *AGC*. Outra característica do algoritmo é o uso de um parâmetro de controle evolutivo, usado para eliminação de indivíduos ao longo das gerações.

3.1 – FUNDAMENTOS

Diferentemente do Algoritmo Genético (*AG*) usualmente encontrado na literatura (Seção 2.2), o *AGC* trabalha com uma população de tamanho variável ao longo das gerações. A população é formada não apenas por *estruturas* que representam soluções do problema, mas também por partes de estruturas, os *esquemas*. A versão do *AGC* desenvolvida neste estudo trabalha com uma população formada apenas por esquemas. A teoria envolvendo os esquemas foi por longo tempo o ponto central dos estudos com o *AG*, mas tem sido menos explorada recentemente. Apesar de esquemas não serem representações das soluções dos problemas, sua recombinação pode produzir estruturas que de fato as representam.

O funcionamento do *AGC* está baseado na especificação *a priori* de valores para vários parâmetros. Normalmente os valores ideais para os parâmetros variam de acordo com o

problema sendo tratado, e seu valor é determinado de modo empírico, isto é, o algoritmo é executado algumas vezes até se encontrar valores para os parâmetros que proporcionem resultados satisfatórios.

O *AGC* considerado neste trabalho começa com a geração de uma população inicial P_0. O número de indivíduos dessa população é um dos parâmetros do algoritmo. Os indivíduos são esquemas S_i, $i=1,2,\ldots|P_0|$, formados de acordo com um mecanismo de representação, o qual deve ser determinado *a priori* e deve refletir características específicas do problema que está sendo tratado. Nesta pesquisa, a representação utilizada é geral para problemas de agrupamentos.

No processo de criação da população inicial, e também no processo evolutivo, cada esquema S_i criado tem a ele associado um *rank* $d(S_i)$ que será usado no processo de eliminação de indivíduos da população. Esse valor numérico é calculado com base na avaliação da adaptação do esquema.

O *AGC* inicia então um ciclo de gerações. Um parâmetro de controle evolutivo $\alpha \in R^+$ começa com valor zero na primeira geração, e é lentamente incrementado com um valor $\Delta\alpha$ a cada nova geração. O valor desse incremento pode variar ao longo do processo evolutivo e também é um dos parâmetros do *AGC*. A eliminação de indivíduos da população é feita com a comparação entre esse parâmetro α e o *rank* de cada indivíduo. Sendo assim, a variação de α tem grande importância no funcionamento do *AGC*. Um incremento acelerado demais pode rapidamente eliminar todos os indivíduos da população sem que esses tenham tempo de se reproduzir.

A cada geração do *AGC*, um certo número de esquemas da população é selecionado e recombinado. A quantidade de novos indivíduos gerados é outro parâmetro do *AGC*. Após a recombinação, tanto um novo esquema quanto uma estrutura podem ser produzidos. Se a recombinação produz um novo esquema, o esquema tem seu *rank* calculado e é inserido diretamente na população. Quando a recombinação gera uma estrutura, essa estrutura passa por um processo de mutação, e em seguida é comparada

com a melhor estrutura obtida até aquele momento. Apenas a melhor estrutura encontrada é mantida salva.

Cada esquema selecionado é temporariamente complementado para gerar uma estrutura que, após sofrer mutação, é comparada à melhor encontrada até o momento.

Após a geração dos novos indivíduos, são eliminados da população todos os esquemas S_i que satisfizerem um critério de eliminação, o qual considera o *rank* do indivíduo $d(S_i)$ e o valor atual do parâmetro α. O valor de α é atualizado com D_α e o processo recomeça com uma nova geração.

O critério de parada pode variar. O processo pode parar quando um certo limite de gerações (parâmetro do *AGC*) previamente estabelecido é atingido, quando eventualmente uma solução satisfatória para o problema abordado tenha sido obtida, ou mesmo quando a população tenha ficado vazia por causa do processo de eliminação de indivíduos. A escolha do critério de parada depende do problema que está sendo tratado.

3.2 – REPRESENTAÇÃO GERAL PARA PROBLEMAS DE AGRUPAMENTO

Este trabalho utiliza uma forma geral de representação para problemas de agrupamentos proposta por Furtado (1998). A representação dos esquemas usa uma seqüência de elementos que representam os objetos a serem agrupados, isto é, cada elemento representa um objeto. Cada elemento da seqüência pode assumir um entre três únicos símbolos: o símbolo "*do-not-care*" (*#*), indicando que o objeto correspondente ainda não está associado a nenhum agrupamento; o símbolo *1*, indicando que o objeto é uma *semente* para formar um agrupamento; e o símbolo *0*, indicando que o objeto já está associado a algum agrupamento. O número de objetos que são sementes é exatamente o número de agrupamentos que se deseja formar. Enquanto a seqüência que representa um esquema possui elementos com o símbolo *#*, uma seqüência representando uma estrutura tem todos os elementos com símbolos *1* ou *0* (Figura 3.1).

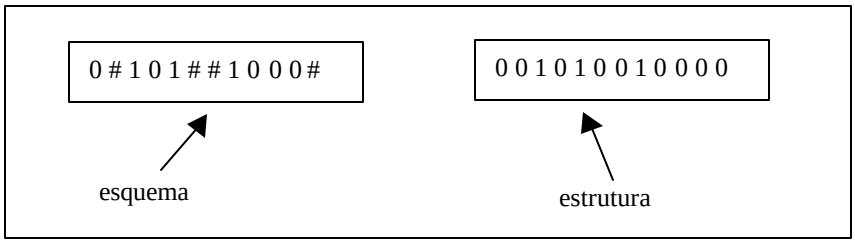

Fig. 3.1 – Representação de um esquema e de uma estrutura.

Com essa forma de representação, a associação de um vértice a um agrupamento deve ser feita por uma heurística auxiliar. Essa heurística depende do problema de agrupamento abordado e tem grande importância para o desempenho do algoritmo, uma vez que a formação dos agrupamentos é pré-requisito para a avaliação da adaptação do esquema ou estrutura.

Como exemplo, vamos considerar um problema de localização conhecido como Problema de Weber (Wesolowsky, 1993). O problema é localizar uma facilidade no plano Euclidiano de modo a minimizar a soma das distâncias dessa facilidade a um dado conjunto de usuários. As distâncias são avaliadas com pesos.

Considerando a localização simultânea de várias facilidades, uma variação do problema conhecida como Problema de Weber Multi-Origem (*PWMO*) corresponde a localizar várias facilidades que produzem um mesmo produto de modo a minimizar a soma das distâncias com pesos de todos os usuários, até sua facilidade mais próxima.

O *PWMO* pode ser formulado da seguinte forma:

$$\min_{x_k, y_k, z_{kj}} \quad \sum_{k=1}^{p} \sum_{j=1}^{n} z_{kj} \cdot w_j \cdot d_j(x_k, y_k) \tag{3.1}$$

$$Suj. \quad \sum_{k=1}^{p} z_{kj} = 1, \qquad j=1,2,\ldots,n.$$

$$z_{kj} \in [0,1] \qquad k=1,2,\ldots,p;\ j=1,2,\ldots,n.$$

onde p facilidades devem ser localizadas para satisfazer a demanda de n usuários; (x_k, y_k) denota as coordenadas da k-ésima facilidade; d_j é a distância Euclidiana entre (x_k, y_k) e o j-ésimo usuário; w_j é a demanda do j-ésimo usuário; e z_{kj} é a fração dessa demanda que é satisfeita pela k-ésima facilidade.

A versão discreta do *PWMO* é conhecida como Problema das p-medianas e consiste em localizar as facilidades dentro de um dado conjunto de possíveis locais. O trabalho de Hansen, Mladenovic e Taillard (Hansen *et al.*, 1998) mostra uma heurística para resolver o *PWMO*, utilizando sua analogia com o Problema das p-medianas.

Seja então uma instância do *PWMO* com $n=5$ usuários dados por: $a_1=(0,3)$, $a_2=(5,3)$, $a_3=(8,0)$, $a_4=(8,6)$ e $a_5=(11,3)$. As demandas são: $w_1=5$, $w_2=4$; $w_3=4$, $w_4=4$ e $w_5=4$. Desejamos localizar $p=2$ facilidades. A Figura 3.2 mostra a localização dos 5 usuários e de 2 facilidades com coordenadas $(0,3)$ e $(8,3)$. Considerando as parcelas de atendimento dos usuários pelas facilidades como sendo inversamente proporcionais à distância do usuário à facilidade, as 2 facilidades atendem principalmente aos grupos de usuários $C_1=\{a_1\}$ e $C_2=\{a_2, a_3, a_4, a_5\}$, respectivamente.

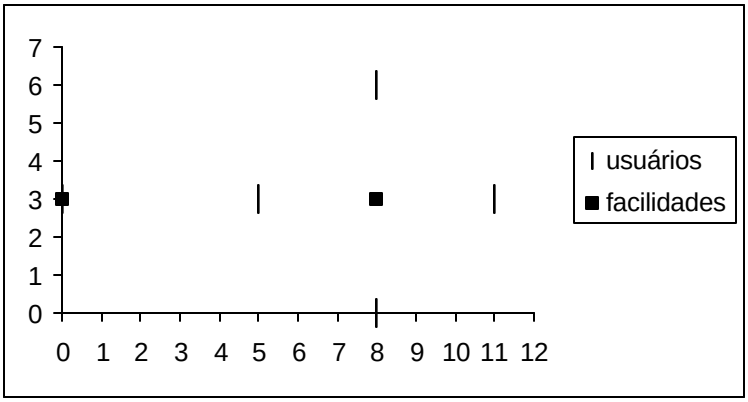

Fig. 3.2 – Instância exemplo de *PWMO*.

O problema pode ser visto então como sendo de formar agrupamentos de usuários C_k ($k=1,2,\ldots,p$), de modo que a média das coordenadas dos usuários de cada grupo nos dá a

localização da facilidade que mais atende às demandas desses usuários, e a parcela de atendimento de cada usuário pelas facilidades depende da distância entre esse usuário e as facilidades.

Neste caso, a representação usada no *AGC* para esquemas e estruturas é uma seqüência de *5* símbolos, cada um representando um usuário; o primeiro representando a_1, o segundo a_2, e assim por diante. Entre esses símbolos teríamos exatamente *2* símbolos indicando sementes (*1's*) e os demais podendo ser tanto *0's*, indicando usuários agrupados, quanto *#'s*, indicando usuários ainda não considerados no esquema.

Qualquer um dos usuários de um agrupamento pode ser a semente usada na sua formação. A Figura 3.3 mostra todas as possíveis estruturas representantes da solução do *PWMO* mostrada na Figura 3.2.

(1,1,0,0,0)
(1,0,1,0,0)
(1,0,0,1,0)
(1,0,0,0,1)

Fig. 3.3 – Representações de solução do exemplo de *PWMO*.

A heurística de associação de um usuário não-semente a um usuário semente para formação dos agrupamentos poderia ser a mesma utilizada por Furtado (1998) para o problema de *p*-medianas: simplesmente associar à semente que estiver mais próxima.

3.3 – AVALIAÇÃO DE ADAPTAÇÃO

Seja C o conjunto de todas as estruturas e esquemas que podem ser gerados pela representação com seqüências de símbolos do alfabeto *{0,1,#}*, e sejam consideradas duas funções *f* e *g*, $f: C \to R^+$ e $g: C \to R^+$, tais que $f(S_i) \leq g(S_i)$ para toda seqüência $S_i \in C$, $(i=1,2,\ldots,/X/)$.

As funções *f* e *g* devem ser definidas adequadamente para representar os objetivos específicos da otimização do problema que está sendo tratado. A função *g* deve representar características do esquema de modo que seu valor seja maior para esquemas mais próximos de estruturas, isto é, com menos símbolos #. A função *f*, não maior que *g*, deve ser formulada de modo que a diferença $\{g(S_i) - f(S_i)\}$ represente os objetivos de otimização do problema.

Logo, durante o processo evolutivo, o *AGC* procura gerar esquemas $S_i \in C$ de modo a atingir dois objetivos: minimização do intervalo $[g(S_i), f(S_i)]$, e maximização de $g(S_i)$.

Pode-se dizer então que o *AGC* resolve de forma indireta o problema abordado, seja ele de agrupamento ou não, como se esse fosse de otimização com duplo critério:

$$\begin{array}{ll} Min & \{g(S_i)-f(S_i)\} \\ Max & g(S_i) \\ & \\ Suj. & g(S_i) \ \mathbf{\hat{I}} \ f(S_i) \\ & S_i \ \mathbf{\hat{I}} \ X \end{array} \qquad (3.2)$$

Considerando o exemplo de *PWMO* citado na Seção 3.2, as formulações das funções *g* e *f* poderiam ser:

$$g(S_i) = \sum_{k=1}^{p} \sum_{j=1}^{n} z_{kj} \cdot w_j \cdot d_j(x_k, y_k) \qquad (3.3)$$

$$f(S_i) = \sum_{k=1}^{p} z_{kj} \cdot w_j \cdot |C_k| \cdot \min_{a_j \in C_k} \{d_j(x_k, y_k)\} \qquad (3.4)$$

onde: C_k é um conjunto de usuários atendidos principalmente pela *k*-ésima facilidade;
(x_k, y_k) é a localização da *k*-ésima facilidade, dada pela média das coordenadas dos usuários do agrupamento C_k;
$d_j(x_k, y_k)$ é a distância Euclidiana do *j*-ésimo usuário à *k*-ésima facilidade;
w_j é a demanda do *j*-ésimo usuário;
a_j é o *j*-ésimo usuário; e

$$z_{kj} = 1 - \frac{d_j(x_k, y_k)}{\sum_{m=1}^{p} d_j(x_m, y_m)}$$

A formulação de g representa simplesmente a soma de todas as distâncias de cada usuário a cada facilidade, multiplicadas por um peso equivalente à parcela da demanda do usuário sendo atendida pela facilidade.

A formulação da função f representa o mesmo que g, porém considerando as distâncias dos usuários de cada agrupamento até sua principal facilidade como sendo todas iguais à menor delas.

Diminuindo a diferença g-f, estaremos indiretamente formando agrupamentos com usuários cada vez mais próximos uns dos outros.

3.4 – O PROCESSO EVOLUTIVO

O processo evolutivo no *AGC* é conduzido de modo a atingir os dois objetivos vistos na formulação 3.2. No início do processo são fornecidos como parâmetros do algoritmo dois valores: um número real não-negativo g_{max}, tal que $g_{\max} > Max_{S_i \in X}\, g(S_i)$, ou seja, um limite superior para $g(S_i)$, para todo $S_i \in X$; e a dimensão de intervalo $d\, g_{\max}$, obtida a partir de $g_{\max}$, usando um número real $0 < d \leq 1$. O valor de g_{max} é de fato calculado no início do algoritmo, usando uma formulação dependente do problema tratado.

O processo considera então um limiar de rejeição adaptativo, que contempla ambos os objetivos da formulação 3.2. O limiar é definido com o uso de um parâmetro evolutivo $\alpha \geq 0$. Um esquema S_i será retirado da população quando:

$$g(S_i) - f(S_i) \geq d\, g_{\max} - \alpha \cdot d[g_{\max} - g(S_i)] \qquad (3.5)$$

O lado direito da expressão 3.5 é o limiar de rejeição, composto da dimensão de intervalo $d\,g_{\max}$, fornecida *a priori,* e a diferença $[g_{\max} - g(S_i)]$.

Para $\alpha = 0$, a expressão 3.5 equivale a comparar o intervalo entre $g(S_i)$ e $f(S_i)$ com a dimensão $d\,g_{\max}$. Porém, quando $\alpha > 0$, os esquemas com mais símbolos # são penalizados e têm maior possibilidade de serem removidos da população, uma vez que para eles, de acordo com o princípio de formulação de g descrito anteriormente, a diferença $[g_{\max} - g(S_i)]$ é maior.

O parâmetro α está relacionado ao tempo de evolução. Considerando que bons esquemas devem permanecer na população para serem recombinados, o parâmetro α começa com valor zero na primeira geração e é acrescido lentamente durante o processo. A população existente com um dado valor de α , denotada P_a , tem seu tamanho dinamicamente influenciado por α , e eventualmente pode ser esvaziada durante o processo.

Isolando α na expressão 3.5, obtemos a expressão

$$\alpha \geq \frac{dg_{\max} - [g(S_i) - f(S_i)]}{d[g_{\max} - g(S_i)]} = d(S_i) \qquad (3.6)$$

em que o lado direito da desigualdade equivale a uma medida de *rank* $d(S_i)$ associada ao esquema. Os esquemas S_i são removidos da população quando $\alpha \geq d(S_i)$.

Como a eliminação do esquema da população é determinada comparando seu *rank* com o valor de α, no momento da criação do esquema é possível determinar sua expectativa de vida. Quanto maior for o valor de $d(S_i)$, mais adaptado é considerado o esquema S_i e mais tempo de sobrevivência para recombinação ele tem.

3.5 – SELEÇÃO E RECOMBINAÇÃO

A seleção de indivíduos para recombinação é feita com características do processo conhecido como seleção por truncamento num esquema não-geracional (*steady-state*), considerando os indivíduos ordenados dentro da população, de acordo com sua adaptação e tamanho do esquema.

Durante a execução do *AGC*, os esquemas da população são mantidos ordenados de forma não-decrescente, de acordo com uma chave $\psi(S_i)$

$$\psi(S_i) = \frac{1 + d(S_i)}{h(S_i)} \tag{3.7}$$

onde: $d(S_i) = \dfrac{g(S_i) - f(S_i)}{g(S_i)}$; e

$h(S_i)$ é o número de símbolos diferentes de # em S_i.

que privilegia os esquemas mais próximos de estruturas e melhor adaptados, isto é, maior $h(Si)$ e menor $\{g(Si)-f(Si)\}$.

O método de seleção para recombinação é referenciado neste trabalho como tipo *base-guia*. Um primeiro esquema, chamado esquema *base*, é aleatoriamente selecionado entre aqueles que ocupam uma parte do início da população, considerada a ordenação com a chave da expressão 3.7. O tamanho dessa parte inicial da população é outro parâmetro do *AGC* e pode variar de acordo com a aplicação (problema e/ou instância), e até mesmo dinamicamente durante o processo evolutivo. Outro esquema, chamado esquema *guia,* é aleatoriamente selecionado entre todos na população.

Antes da recombinação, uma estrutura é formada com o instanciamento do esquema base, isto é, a simples substituição dos símbolos *#*'s por símbolos *0*'s. Essa estrutura

sofre então uma mutação e é comparada com a melhor estrutura obtida até o momento. O motivo para essa complementação é que o processo de recombinação usado no caso de agrupamentos, apesar de manter o número correto de sementes no novo esquema gerado, pode desfazer a seqüência de símbolos do esquema base. Com a complementação, as informações do esquema base, um bom esquema, não se perdem e passam diretamente a uma estrutura associada.

Usando os esquemas base e guia tomados da população, a recombinação é feita com um algoritmo de cruzamento (Figura 3.4), gerando um novo esquema S_{novo}, ou eventualmente uma nova estrutura, sempre mantendo o número de sementes para formação dos agrupamentos.

Se $S_{base}(j) = S_{guia}(j)$ *então* $S_{novo}(j) \neg S_{base}(j)$ *ou* $S_{guia}(j)$
$S_{base}(j) = 1$ *e* $S_{guia}(j) = \#$ *então* $S_{novo}(j) \neg 1$
$S_{base}(j) = 0$ *e* $S_{guia}(j) = \#$ *então* $S_{novo}(j) \neg 0$
$S_{base}(j) = \#$ *e* $S_{guia}(j) = 0$ *então* $S_{novo}(j) \neg 0$
$S_{base}(j) = \#$ *ou* 0 *e* $S_{guia}(j) = 1$ *então*
$S_{novo}(j) \neg 1$ *e* $S_{novo}(i) \neg 0$ *para algum* $S_{novo}(i) = 1$
$S_{base}(j) = 1$ *e* $Sguia(j) = 0$ *então*
$S_{novo}(j) \neg 0$ *e* $S_{novo}(i) \neg 1$ *para algum* $S_{novo}(i) = 0$

Fig. 3.4 – Algoritmo de recombinação.

Após a recombinação, se S_{novo} for um esquema, ele é simplesmente inserido na população na posição referente a seu valor da chave de ordenação, mas se for uma estrutura, sofre um processo de mutação e é comparado à melhor estrutura obtida até aquele momento.

A cada iteração do processo, um certo número de seleções e recombinações é feito. Esse número pode variar de acordo a aplicação do *AGC*. O número de novos indivíduos sendo criados a cada geração é outro parâmetro do *AGC*.

Esse processo de recombinação, mantendo o número de sementes, foi descrito no trabalho de Furtado (1998) para problemas de agrupamento em geral, e pode ser aplicado ao exemplo de *PWMO.*

3.6 – MUTAÇÃO

Como mostrado a seguir, os melhores resultados obtidos na aplicação de algoritmos evolutivos a problemas de otimização de maior dificuldade parecem ser produzidos com a utilização de heurísticas de busca local aplicada às soluções obtidas por esses algoritmos.

Os trabalhos de Michalewicz (1996) e Michalewicz e Fogel (2000) descrevem a utilização de um processo diferenciado de mutação em estruturas binárias, com o objetivo de melhorar a qualidade dos indivíduos da população através de um ajuste equivalente a uma busca local. O autor descreve também a utilização de heurísticas combinadas com os *AGs*, não apenas para melhoria de estruturas viáveis, mas também para a "reparação" de estruturas não-viáveis.

Os chamados *Algoritmos Meméticos*, surgidos inicialmente no trabalho de Moscato (1989), são heurísticas de busca para problemas de otimização baseadas em populações. O autor os descreve como a combinação de heurísticas de busca local e operadores de reprodução, sendo por esta razão, classificados por alguns pesquisadores como algoritmos genéticos híbridos. Segundo o autor, o método tem ganho aceitação por ter resolvido instâncias de conhecidos problemas de otimização em que outros métodos não tinham obtido sucesso. O método tem sido bastante explorado e uma vasta referência pode ser obtida através do endereço:
(http://www.densis.fee.unicamp.br/~moscato/memetic_home.html).

O *AGC* utiliza heurísticas dependentes do problema como processos de mutação. O objetivo da aplicação dessas heurísticas sobre as estruturas produzidas no processo

evolutivo é não apenas a melhoria da qualidade das estruturas, mas também a garantia de obtenção da viabilidade.

No caso do exemplo com *PWMO,* podemos usar como mutação um processo que busca uma melhoria local da solução representada por uma estrutura mudando a posição das sementes nos agrupamentos.

Proposto por Cooper (1963) e revisado recentemente nos trabalhos de Taillard (1996), Senne e Lorena (2000) e Narciso, Lorena e Furtado (2000), o processo é mostrado na Figura 3.5.

```
Selecionar aleatoriamente um agrupamento
Para cada usuário não semente desse agrupamento
        Mudar a semente para esse usuário
        Associar novamente todos os usuários formando novos agrupamentos
        Avaliar a nova solução e registrá-la se for melhor
Mover definitivamente a semente para o usuário com que se obteve a melhor avaliação
```

Fig. 3.5 – Processo de mutação para o exemplo com *PWMO.*

Um processo semelhante a esse foi utilizado nos experimentos descritos nos próximos Capítulos deste trabalho.

3.7 – PSEUDOCÓDIGO

O entendimento do funcionamento do AGC pode ser facilitado com sua representação em pseudocódigo (Figura 3.6).

Dados d , D_a , $M_{axi}t$, N_{ind}
$\alpha \leftarrow 0$
Calcular g_{max}
Inicializar P_α
Computar $\sigma(S_i)$ **para todo** $S_i \in P_a$

Ordenar P_α com base em $\sigma(S_i)$
It $\leftarrow 0$
Inicializar Melhor
Enquanto (It < M_{axit} e não achou solução satisfatória e população não está vazia)
Fazer N_{ind} vezes
Selecionar S_{base}
Instanciar S_{base} gerando estrutura E
Mutação em E
Se E é melhor que *Melhor* **então** *Melhor* $\leftarrow E$
Selecionar S_{guia}
Recombinar S_{base} com S_{guia} gerando S_{novo}
Se S_{novo} é esquema **então**
Calcular $\sigma(S_{novo})$ e inserir na lista
Senão
Mutação em S_{novo}
Se S_{novo} é melhor que *Melhor* **então** *Melhor* $\leftarrow S_{novo}$
Remover *todo* S_i *tal que* $\alpha \notin \sigma(S_i)$
$\alpha \leftarrow \alpha + D_a$
It $\leftarrow$ It + 1
Exibir Melhor

Fig. 3.6- Pseudocódigo do *AGC*.

Os próximos Capítulos mostram a aplicação do *AGC* a três problemas de otimização combinatória muito conhecidos: Coloração de Grafos, Projeto de Células de Manufatura e *Timetabling*.

Algumas figuras que ilustram conceitos do *AGC* descritos neste Capítulo podem ser vistas no apêndice A.

CAPÍTULO 4

APLICAÇÃO DO *AGC* AO PROBLEMA DE COLORAÇÃO DE GRAFOS

Coloração de Grafos é um problema clássico de otimização combinatória que tem sido muito estudado, e pode ser entendido como a tarefa de, usando o menor número de cores possível, escolher uma cor para cada vértice de um grafo, de modo que vértices adjacentes não recebam a mesma cor.

Claramente é um problema de formar agrupamentos de vértices não adjacentes. Situações práticas em que se pretende, a partir de um conjunto de objetos com incompatibilidades entre alguns de seus elementos, formar agrupamentos de objetos sem incompatibilidades entre si, podem ser modeladas como problemas de coloração. Neste caso, o grafo é formado de modo que cada vértice represente um objeto, e as arestas liguem os vértices referentes a objetos de alguma forma incompatíveis.

Neste Capítulo faz-se uma descrição do problema, características da primeira abordagem usando o *AGC*, características da abordagem atual tratando o problema como sendo de agrupamento, e resultados de testes computacionais. Os testes foram feitos com instâncias conhecidas na literatura. Ainda neste Capítulo, é apresentada uma abordagem combinada do *AGC* com um processo de geração de colunas.

4.1 – DESCRIÇÃO DO PROBLEMA

Seja $G = (V,E)$ um grafo não-direcionado. Uma *k-coloração* de G é um particionamento de V em k subconjuntos C_i $(i=1,2,\ldots,k)$, de modo que em cada subconjunto não haja vértices adjacentes. O problema conhecido como Coloração de Grafos é encontrar uma *k-coloração* de G usando o menor valor possível para k. O número mínimo de cores que se pode usar para colorir um grafo G, quando conhecido, é chamado *número cromático* de G, e denotado $\chi(G)$.

O problema é *NP-hard* (Garey e Johnson, 1979), e heurísticas são usadas para instâncias mais difíceis, grafos com muitos vértices ou muito densos. O uso de meta-heurísticas tem produzido os melhores resultados para instâncias com grafos de grande porte, com centenas ou milhares de vértices. Na literatura são encontradas aplicações de *Simulated Annealing* (Johnson *et al*., 1991; Chams *et al*., 1987), Busca Tabu (Hertz e de Werra, 1987), algoritmos evolutivos (Costa, *et al.*, 1995) e algoritmos genéticos híbridos (Fleurent e Ferland, 1994). Mehrotra e Trick (1993) determinaram limites inferiores para o número cromático usando um método de geração de colunas. Recentemente, Laguna e Martí (1999) aplicaram uma técnica conhecida como *"greedy randomized adaptive search procedure – GRASP"* (Feo e Resende, 1995) com bons resultados comparados com outras heurísticas.

Na página com endereço http://web.cs.ualberta.ca/~joe/Coloring/index.html pode ser encontrada uma grande lista de referências bibliográficas. Outra fonte de informações com acesso eletrônico é a página de Pesquisa Operacional mantida por Michael Trick (http://mat.gsia.cmu.edu/index.html), que também disponibiliza instâncias para teste.

Aplicações práticas aparecem em vários problemas: designação de freqüências (Hale, 1980; Gamst, 1986); alocação de registradores para variáveis em código executável (Briggs *et al*., 1989); e até mesmo em *scheduling* (Leighton, 1979) e em sistemas flexíveis de manufatura (Stecke, 1985).

4.2 – PRIMEIRA ABORDAGEM COM O *AGC*

Na primeira aplicação do *AGC* ao problema de Coloração de Grafos (Ribeiro Filho, 1997), o estudo com o algoritmo estava em sua fase inicial. Embora as bases do algoritmo fossem as mesmas, o método de representação dos esquemas e estruturas era diferente e, consequentemente, a geração da população inicial e recombinação também eram diferentes. A população era formada tanto por esquemas quanto por estruturas e não era utilizada nenhuma mutação.

Tanto na primeira abordagem quanto na forma atual, o uso do *AGC* começa com um número inicial *k* de cores admitido *a priori*. Enquanto o *AGC* encontrar uma *k*-coloração, o valor de *k* será decrescido e o algoritmo será executado novamente. A última *k*-coloração encontrada será então considerada como a melhor solução obtida.

Na primeira abordagem, a representação de esquemas e estruturas era feita com uma seqüência formada por elementos correspondentes aos vértices. Cada elemento da seqüência indicava a cor do vértice ou era um símbolo "*do-not-care*", indicando que o vértice correspondente não estava colorido no esquema. Por exemplo, supondo uma procura por uma *3*-coloração para um grafo com sete vértices, a seqüência *(1, #, 1, 3, #, 2, 2)* representava um esquema com uma coloração parcial, em que o primeiro e terceiro vértices tinham a cor 1, o quatro vértice tinha a cor 3, e o sexto e sétimo vértices tinham a cor 2, enquanto o segundo e o quinto vértices ainda não tinham cores a eles associadas. A seqüência *(1, 2, 1, 3, 1, 2, 2)*, por outro lado, equivale a uma estrutura que representa uma *3*-coloração.

A população inicial era gerada com uma quantidade de esquemas exatamente igual ao número de vértices do grafo, cada um dos esquemas tendo uma série de posições adjacentes (na seqüência) contendo todas *k* cores sendo utilizadas (Figura 4.1).

(1, 2, 3, #, #, #, #)
(#, 1, 2, 3, #, #, #)
(#, #, 1, 2, 3, #, #)
(#, #, #, 1, 2, 3, #)
(#, #, #, #, 1, 2, 3)
(3, #, #, #, #, 1, 2)
(2, 3, #, #, #, #, 1)

Fig. 4.1 – População inicial.

A seleção era feita com um esquema base e outro esquema guia (Seção 3.5), e a recombinação era feita mantendo o esquema base e complementando-o com partes do esquema guia (Figura 4.2).

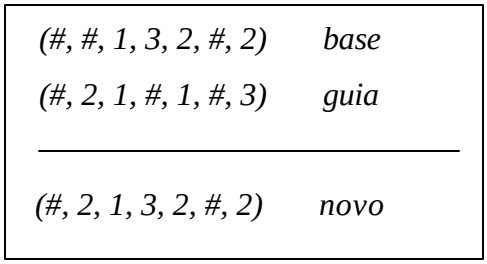

Fig. 4.2 – Recombinação.

A avaliação da adaptação dos esquemas e das estruturas era feita com as formulações 4.1 e 4.2 para as funções *f* e *g*, respectivamente. A formulação 4.3 era usada para o cálculo do limite superior g_{max} (Seção 3.4).

$$g(S_i) = \sum_{p=1}^{k} \left\lfloor \frac{\left(\left|C_p(S_i)\right| - 1\right)\left|C_p(S_i)\right|}{2} \right\rfloor \tag{4.1}$$

$$f(S_i) = g(S_i) - \sum_{p=1}^{k} \left| E\left(C_p(S_i)\right) \right| \tag{4.2}$$

$$g_{\max} = \phi \cdot k \cdot \left\lfloor \frac{\left\lfloor \frac{n}{k} - 1 \right\rfloor \cdot \left\lfloor \frac{n}{k} \right\rfloor}{2} \right\rfloor \tag{4.3}$$

onde:

n é o número de vértices do grafo;
k é o número de cores sendo usadas;
$C_p(S_i)$ é o conjunto de vértices com a cor *p* no esquema ou estrutura S_i;
$E(C_p(S_i))$ é o conjunto das arestas no conjunto $C_p(S_i)$; e
ϕ é um fator para garantir g_{max} como um limitante superior.

De acordo com as formulações acima, a função g corresponde à soma do número de arestas de k grafos completos, considerando os conjuntos de vértices referentes a uma determinada cor. A função f corresponde ao valor de g menos o número de arestas que de fato há em cada conjunto de vértices. O limitante superior para g, denotado g_{max}, é calculado de modo semelhante ao cálculo de g, mas considerando os vértices igualmente distribuídos entre os agrupamentos, multiplicado ainda por um fator empírico f que garante $g_{max} > g(S_i), \ \forall S_i \in X$.

Com a população formada tanto por esquemas quanto por estruturas, o processo evolutivo consistia da seleção e recombinação de um certo número de indivíduos, inserindo o indivíduo gerado na população, sendo ele um esquema ou uma estrutura. Não havia mutação e o processo de eliminação utilizava o mesmo parâmetro evolutivo e a mesma medida de *rank* descritos na Seção 3.4.

4.3 – ABORDAGEM COMO PROBLEMA DE AGRUPAMENTO

Na abordagem atual, o problema é considerado como sendo de formação de agrupamentos de vértices. Utiliza-se uma representação de esquemas e estruturas através de seqüências de símbolos de um alfabeto contendo apenas três elementos, descrita na Seção 3.3. Cada elemento da seqüência continua equivalendo a um vértice, e o número de vértices que são sementes é sempre exatamente igual ao número de cores usadas na execução do *AGC*.

Usando essa forma de representação, a associação de um vértice a um agrupamento é feita através de uma heurística auxiliar. Essa heurística usa uma adaptação de um algoritmo conhecido como *Recursive Large First (RLF)* (Leighton, 1979). O processo pode ser melhor entendido usando um exemplo: suponha que estejamos procurando uma *3*-coloração para um grafo com dez vértices e representado pela seguinte matriz simétrica de adjacências:

0111000000
1000001000
1001010000
1010101100
0001010110
0010100000
0101000001
0001100010
0000100100
0000001000

Sejam então os seguintes conjuntos:

C_i : conjunto dos vértices no *i*-ésimo agrupamento,

U_i : conjunto dos vértices do esquema que são adjacentes a algum outro em C_i,

V_{sch} : conjunto de todos os vértices do esquema, e

V_i : conjunto $V_{sch} - U_i$.

Vamos considerar agora o esquema:

(#,1,0,1,#,0,0,#,1,0)

onde: *1* = vértice semente;
0 = vértice a ser associado; e
= vértice a não ser associado.

Inicialmente temos então:

$C_1 = \{2\}$	$C_2 = \{4\}$	$C_3 = \{9\}$
$V_1 = \{3,6,10\}$	$V_2 = \{6,10\}$	$V_3 = \{3,6,7,10\}$
$U_1 = \{7\}$	$U_2 = \{3,7\}$	$U_3 = \{\ \}$

Agora, tomamos o vértice *v* em V_i *(i=1,2 ou 3)* com maior grau em U_i *(i=1,2 ou 3)* e associamos *v* a C_i. Em seguida, atualizamos os conjuntos C_i, V_i, e U_i, e assim obtemos:

$C_1 = \{2,10\}$	$C_2 = \{4\}$	$C_3 = \{9\}$
$V_1 = \{3,6\}$	$V_2 = \{6\}$	$V_3 = \{3,6,7\}$
$U_1 = \{7\}$	$U_2 = \{3,7\}$	$U_3 = \{\ \}$

Repetindo o processo, teremos:

$C_1 = \{2,10\}$ $C_2 = \{4,6\}$ $C_3 = \{9\}$
$V_1 = \{3\}$ $V_2 = \{\ \}$ $V_3 = \{3,7\}$
$U_1 = \{7\}$ $U_2 = \{3,7\}$ $U_3 = \{\ \}$

Continuando até que todos os conjuntos V_i estejam vazios, ao final teremos:

$C_1 = \{2,3,10\}$ $C_2 = \{4,6\}$ $C_3 = \{7,9\}$

Eventualmente, quando todos os conjuntos V_i estiverem vazios, pode haver ainda vértices sobrando nos conjuntos U_i. Nesse caso, esses vértices são colocados nos agrupamentos em que tiverem menos vértices adjacentes.

4.3.1 – POPULAÇÃO INICIAL

Considerando a representação dos esquemas e estruturas para problemas de agrupamento, o processo de criação dos indivíduos da população inicial foi bastante simples. Cada novo esquema criado inicialmente recebeu em posições aleatoriamente escolhidas da seqüência os *k* símbolos *1* para indicar os vértices sementes para geração dos agrupamentos. Em seguida, 20% das posições restantes, também escolhidas aleatoriamente, receberam o símbolo *0*, indicando que o vértice correspondente estaria pertencendo a algum agrupamento. O valor dessa porcentagem foi estipulado de modo empírico. As demais posições na seqüência receberam então o símbolo #, indicando vértices ainda não pertencentes a nenhum agrupamento, ou seja, ainda não coloridos.

A quantidade de indivíduos na população inicial foi sempre exatamente igual ao número de vértices do grafo, considerando que quanto maior o grafo sendo colorido, maior a necessidade de diversificação dos indivíduos da população inicial.

4.3.2 – SELEÇÃO E RECOMBINAÇÃO

A seleção e recombinação de indivíduos foram feitas de acordo com o mecanismo descrito na Seção 3.5. O esquema base foi aleatoriamente escolhido entre os melhores 20% dos esquemas na população a cada instante. Essa porcentagem foi também

empiricamente estabelecida, de acordo com a idéia de que como base deveríamos ter os esquemas de boa qualidade na população.

A cada geração, o número de seleções feitas foi exatamente igual ao número de vértices do grafo sendo colorido. Mais uma vez, esse número foi empiricamente estabelecido, de modo a ser proporcional ao tamanho do grafo.

4.3.3 – MUTAÇÃO

Ao contrário da primeira abordagem do problema usando o *AGC*, a abordagem atual utiliza uma heurística de mutação. Essa heurística é um processo iterativo baseado na idéia de, em cada agrupamento, tomar o vértice com maior grau e considerá-lo como sendo uma semente no lugar da atual, associar novamente todos os vértices aos agrupamentos, e repetir o processo até não obter mais melhoria da solução como um todo. A Figura 4.3 mostra esse processo na forma de pseudocódigo.

```
Repetir
        Para cada agrupamento
                Mover a semente ao vértice com maior grau no agrupamento
                Designar novamente os vértices usando RLF
                Contar conflitos e salvar a melhor no loop
Até que a melhor no loop acima não seja melhor que a solução original
Substituir a original pela melhor do Loop
Parar
```

Fig. 4.3 – Pseudocódigo do Processo de Mutação.

4.3.4 – FUNÇÕES DE AVALIAÇÃO

As formulações das funções g e f utilizadas, bem como a formulação para estimativa do limite superior g_{max} são as mesmas usadas na abordagem anterior, e podem ser vistas nas expressões 4.1, 4.2 e 4.3, respectivamente.

4.3.5 – TESTES COMPUTACIONAIS

Testes foram feitos em instâncias cujas melhores colorações já encontradas estão registradas, algumas delas com o próprio número cromático dos grafos. Essas instâncias são encontradas na página de Pesquisa Operacional mantida por Michael Trick com endereço: http://mat.gsia.cmu.edu/COLOR/instances.

As instâncias estão relacionadas em classes de acordo com sua forma de criação:

BOOK - (Anna, David, Huck e Jean) – Grafos baseados em obras literárias em que cada vértice representa uma personagem e uma aresta indica personagens que se encontram na história).

GAME - (Games120) – Um grafo em que cada vértice representa um time e uma aresta indica um jogo entre eles na temporada.

MILES - (Miles250, Miles500 e Miles750) – Grafos em que os vértices representam cidades e são adjacentes quando a distância entre elas é menor que um determinado valor.

REG - (Musol_1, Musol_2, Zeroin_1 e Zeroin_2) - Grafos baseados em problemas de alocação de registradores para variáveis em um programa.

MYC - (Myciel5, Myciel6 e Myciel7) - Grafos baseados na transformação de Mycielski;

QUEEN - (Queen55, 66, 77, 88 e 99) - Grafos com n^2 vértices, cada um correspondendo a uma posição no tabuleiro de xadrez de tamanho n, em que dois vértices são ligados por uma aresta se as posições correspondentes estão na mesma linha, coluna ou diagonal.

Na página de Michael Trick as instâncias são divididas em grupos de acordo com sua origem. As classes de instâncias *BOOK*, *GAME*, *MILES* e *QUEEN* fazem parte de um grupo chamado *SGB.* Tanto o grupo *SGB* quanto os grupos *REG* e *MYC* da página de Michael Trick contêm mais instâncias do que aquelas relacionadas nas classes citadas anteriormente e usadas nos primeiros testes, uma vez que seu objetivo é uma comparação de resultados obtidos com a abordagem atual e aqueles obtidos com a primeira abordagem, usando o *AGC* (Ribeiro Filho, 1997).

A Tabela 4.1 mostra os resultados das duas abordagens. De cada classe de grafos, a tabela mostra o número de instâncias, o número médio de vértices e de arestas, e o número médio de conflitos na melhor solução obtida. Como conflito entende-se a presença de um par de vértices adjacentes dentro de um agrupamento. A tabela mostra ainda os tempos médios de execução, em segundos. Os experimentos foram feitos com o programa rodando três vezes para cada instância, sempre procurando uma coloração com o número de cores igual ao melhor encontrado na página de Michael Trick.

TABELA 4.1: TESTES COM COLORAÇÃO

Grupos	*Qtde. Inst.*	*Vértices*	*Arestas*	*Abordagens*			
				Anterior		*Atual*	
				Conflitos	*Tempo*	*Conflitos*	*Tempo*
BOOK	4	94.8	363.5	0.0	46.0	0	1.9
GAME	1	120.0	638.0	0.0	91.3	0	3.0
MILES	3	128.0	1223.3	2.4	23595.1	0	17.4
REG	4	201.8	3862.8	1.2	22626.0	0	508.9
MYC	3	111.0	1117.0	1.0	1519.2	0	3.3
QUEEN	5	51.0	753.2	24.3	43.3	0.8	1021.5

A Tabela 4.1 evidencia a melhoria de resultados com a nova abordagem, usando *AGC* e tratando o problema como de formação de agrupamentos e utilizando mutação. Em ambas as abordagens o código de teste foi escrito em linguagem C. No entanto, como os testes foram feitos em épocas diferentes, consequentemente as plataformas são diferentes. Os testes com a primeira abordagem foram feitos em um DX4 de 100 MHz, e os novos testes num Pentium II de 266 MHz.

No trabalho de Laguna e Martí (1999) com aplicação de um método *GRASP* (Feo e Resende, 1995) para colorir grafos esparsos, os autores usaram os grupos de instâncias *MYC*, *REG* e *SGB*, este último com exceção dos grafos da classe *QUEEN*. Os autores comparam os resultados do método *GRASP* com resultados obtidos com outras heurísticas.

A Tabela 4.2 mostra uma comparação entre os resultados vistos no trabalho de Laguna e Martí e os resultados obtidos com a aplicação atual do *AGC*. Nesse caso, o número de instâncias dos grupos *SGB*, *REG* e *MYC* é maior do que os usados nos testes anteriores. A tabela mostra, de cada grupo, o número de instâncias, o número médio de vértices, o número médio de arestas, a média dos números cromáticos, e a média dos menores números de cores encontrados pelo *AGC* e as seguintes heurísticas: *GRASP* (Laguna e Martí, 1999); *TABUCOL* (Hertz e de Werra, 1987) e *Simulated Annealing (SA)* (Johnson *et al*., 1991). A tabela não traz os tempos porque as plataformas de testes são muito diferentes.

TABELA 4.2 – COMPARAÇÃO ENTRE HEURÍSTICAS PARA COLORAÇÃO

Gr.	*Inst.*	*Vért.*	*Arestas*	*Cores*	*TABUCOL*	*SA*	*GRASP*	*AGC*
MYC	5	73.4	688.4	6.0	6.0	6.0	6.0	6.0
REG	14	362.1	7608.1	37.4	57.1	40.1	37.4	37.4
SGB	10	113.9	2835.2	22.6	24.1	26.0	22.7	22.7

A Tabela 4.2 mostra que os resultados do *AGC* podem ser comparados com os melhores das outras heurísticas.

Nas Tabelas 4.1 e 4.2, os resultados do *AGC* foram obtidos com uma população inicial cujo número de indivíduos foi exatamente igual ao número de vértices do grafo, o mesmo ocorrendo com o número se seleções a cada iteração. Na população inicial, os indivíduos tiveram, além das posições dos vértices sementes, 20% das posições restantes associadas a alguma semente. O esquema base foi tomado entre os primeiros 33% da população. O valor adotado para *d* foi 0.1 e o incremento de α foi de 0.005.

4.4 – ABORDAGEM COMBINADA DO *AGC* E GERAÇÃO DE COLUNAS

Um processo para determinar limites inferiores para os números cromáticos usando geração de colunas foi proposto por Mehrotra e Trick (1993).

Dado um grafo não-direcionado $G=(V, E)$, um subconjunto de V formado apenas por vértices não adjacentes é chamado de conjunto independente de vértices. Dada uma k-coloração de G, todos os vértices com uma determinada cor formam um *conjunto independente de vértices*. Um *conjunto independente de vértices maximal* é um conjunto independente que não está contido em nenhum outro conjunto independente.

Seja Q o conjunto de todos os conjuntos independentes maximais de G, e seja uma matriz A (Figura 4.4) com $m = |V|$ linhas e $n = |Q|$ colunas, em que as colunas são seqüências binárias de modo que $a_{ij} = 1$ se o vértice i pertence ao conjunto independente maximal j, e $a_{ij} = 0$ em caso contrário.

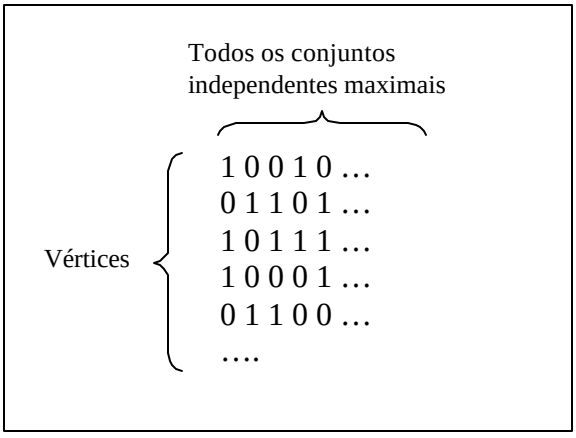

Fig. 4.4 – Matriz de conjuntos independentes maximais.

Considerando a matriz A descrita acima, o problema de determinar o número mínimo de cores com as quais podemos colorir o grafo G pode ser formulado da seguinte maneira para obter o número mínimo de conjuntos independentes maximais:

$$\begin{aligned} \textit{Min} \quad & \sum_{j=1}^{|Q|} x_j \\ \textit{Suj.} \quad & \sum_{j=1}^{|Q|} a_{ij} x_j \geq 1 \quad ; i = 1,...,|V| \\ & x_j \in \{0,1\} \end{aligned} \tag{4.4}$$

O problema assim formulado, denotado *CI*, é de cobertura de conjuntos, possui uma restrição para cada vértice de *G*, mas pode ter um número muito grande de variáveis (colunas).

Para resolver o problema *CI* de modo aproximado, é utilizado um conjunto inicial $\overline{Q}$ formado por conjuntos independentes de vértices, não necessariamente maximais. Faz-se também uma relaxação de linearização, admitindo $x_i \in [0,1]$.

O problema *CI* na forma relaxada é então resolvido, obtendo-se um conjunto de variáveis duais π_i, uma para cada restrição, isto é, uma para cada vértice do grafo *G*.

Para determinar se o conjunto $\overline{Q}$ deve ser expandido, de acordo com o método de geração de colunas, devemos resolver o problema de conjuntos independentes com pesos, denotado *CIP*, formulado da seguinte maneira:

$$\begin{aligned} \textit{Max} \quad & \sum_i \pi_i \cdot z_i \\ \textit{Suj.} \quad & z_i + z_j \leq 1 \qquad \forall (i,j) \in E \\ & z_i, z_j \in \{0,1\} \end{aligned} \tag{4.5}$$

Se a solução obtida for maior que um, os valores z_i formam um conjunto independente que é incluído em $\overline{Q}$, e o processo se repete até que não sejam encontrados mais conjuntos independentes para expandir $\overline{Q}$.

No final do processo, a solução do problema relaxado equivale a um limite inferior para o número cromático.

Neste estudo, o *AGC* foi utilizado para gerar o conjunto inicial $\overline{Q}$. A princípio, o *AGC* tenta encontrar uma coloração usando um número de cores que se acredita acima do número cromático. Se o *AGC* encontra essa coloração, o número de cores é reduzido em uma unidade e o *AGC* é executado novamente. Os conjuntos independentes equivalentes às cores usadas na última coloração encontrada com sucesso pelo *AGC* são usados para formar o conjunto inicial $\overline{Q}$.

O problema *CI* relaxado foi resolvido com o pacote conhecido como CPLEX 6.5 (ILOG, 1999), obtendo as variáveis duais $\boldsymbol{p}_i$.

Ao invés de resolver diretamente o problema *CIP*, o próprio *AGC* foi utilizado novamente para resolvê-lo de modo aproximado. O número de cores é reduzido em uma unidade e o *AGC* é executado. Todos os conjuntos independentes encontrados pelo *AGC* durante sua execução são salvos. Cada um desses conjuntos independentes admite uma representação por uma seqüência binária de elementos z_i, $i = 1,2,\ldots|V|$. Cada um desses elementos é então multiplicado por seu correspondente valor dual $\boldsymbol{p}_i$, e se a soma desses produtos for maior que um, o conjunto independente será incluído em $\overline{Q}$.

O problema *CI* na forma relaxada é então resolvido novamente com o CPLEX, e o processo todo se repete até que não sejam mais encontrados conjuntos independentes para expandir $\overline{Q}$.

A cada iteração do processo, um outro limite inferior pode ser estabelecido através do chamado "limite de Farley" (Farley, 1990), denotado f, e dado por:

$$f = \frac{v(CI)}{v(CIP)} \tag{4.6}$$

onde $v(.)$ é o valor ótimo para o problema.

4.4.1 – TESTES COMPUTACIONAIS

Para os testes foi escolhida a instância *Queen99*, uma vez que foi a instância de maior dificuldade de resolução com o AGC entre as citadas na Seção 4.3.5. Para esse grafo, a melhor solução encontrada pelo AGC foi com 11 cores.

A Tabela 4.3 mostra os resultados dos testes com geração de colunas indicando a iteração, o número de cores para o qual o *AGC* foi executado, o número de conflitos da melhor solução obtida pelo *AGC*, o resultado obtido com o CPLEX para o problema *CI* relaxado, a estimativa do limite de Farley, e o tempo em segundos correspondente à execução apenas do *AGC*. O CPLEX resolve o problema *CI* relaxado de forma muito rápida e seus tempos não são informados.

TABELA 4.3: GERAÇÃO DE COLUNAS PARA *QUEEN99*

Iter.	*Cores*	*Conflitos*	*CI (relax)*	*Farley (estimado)*	*Tempo(seg)*
0	10	2	9.226	8.359	295
1	9	10	9.059	8.155	283
2	8	25	9.007	8.542	246
3	7	47	9.000	-	177
4	6	75	-	-	121

Como a melhor solução encontrada pelo *AGC* foi com 11 cores, e os limites dados pelo problema *CI* e pela estimativa de Farley (arredondando) são iguais a 9, podemos afirmar que o número cromático do grafo é 9, 10 ou 11. De fato, conforme indicação na página de Michael Trick (http://mat.gsia.cmu.edu/index.html), o número cromático desse grafo é 10.

Com o decréscimo do número de cores, aumenta o número de conflitos na solução do *AGC*, mas os novos conjuntos independentes melhoram o limite dado pelo problema *CI* relaxado.

Acreditamos que a oscilação na estimativa do limite de Farley deve-se ao fato de que o problema *CIP* é resolvido de modo aproximado.

Os testes foram feitos numa estação de trabalho SUN-ULTRA30.

4.5 – CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste Capítulo foram mostrados resultados da aplicação do *AGC* ao problema de Coloração de Grafos, descrevendo tanto as características da primeira abordagem do problema com *AGC* quanto as características da nova abordagem, tratando o problema como sendo de formação de agrupamentos de vértices.

Foram utilizadas características do *AGC* próprias para problemas de agrupamento. A representação de esquemas e estruturas utiliza sementes e uma heurística de associação dos vértices às sementes baseada no algoritmo conhecido como *RLF – Recursive Large First*.

As formulações das funções para avaliação da adaptação dos esquemas e estruturas foram as mesmas da abordagem anterior. No entanto, outras características diferenciadas do *AGC* foram introduzidas: a população passou a ser formada apenas por

esquemas, sendo mantida somente a melhor estrutura obtida até o momento; e uma heurística de mutação foi utilizada para melhoria das estruturas obtidas, tanto pelo processo de recombinação quanto pelo instanciamento de bons esquemas da população.

Resultados comparativos entre as duas abordagens utilizando *AGC* mostram claramente uma grande melhoria dos resultados com as mesmas instâncias, tomadas da literatura e obtidas eletronicamente. Testes comparativos com a literatura recente mostram ainda que os resultados da nova abordagem com *AGC* se comparam, em qualidade de soluções, aos melhores obtidos com outras heurísticas muito conhecidas.

Foi ainda mostrada neste Capítulo a aplicação do *AGC* para determinação de um limite inferior para o número cromático de um grafo, utilizando um processo iterativo de geração de colunas para um problema formulado como sendo de cobertura de conjuntos. O limite obtido nos testes se mostrou consistente com outro limite conhecido na literatura como limite de Farley, que foi estimado com o uso do próprio *AGC* na resolução aproximada de um problema de maximização.

O código do *AGC* foi escrito em linguagem C e os testes foram feitos em microcomputadores e estações de trabalho. Nos testes com geração de colunas foi utilizado também o pacote de *software* chamado CPLEX, que se mostrou muito eficiente.

A impressão clara que os testes deixaram foi que a heurística de mutação tem grande importância na qualidade das soluções, resultado este já esperado pelo fato de se tratar de um problema de otimização resolvido com uma heurística evolutiva. No entanto, esse processo consome muito tempo de processamento, e o código utilizado deverá ser melhorado em futuras aplicações com grafos de maior porte, com milhares de vértices.

CAPÍTULO 5

APLICAÇÃO DO *AGC* AO PROJETO DE CÉLULAS DE MANUFATURA

A competitividade internacional e conseqüente necessidade de respostas rápidas à demanda do mercado têm levado muitas empresas a considerar abordagens não-tradicionais para o projeto e controle de sistemas de manufatura. Uma abordagem é a aplicação de tecnologia de grupo, caracterizada pela exploração de similaridades nas atividades ligadas à produção. Em essência, a tecnologia de grupo tenta decompor os sistemas de manufatura em vários subsistemas, ou grupos, controláveis.

Neste Capítulo é feita inicialmente uma descrição do problema, em seguida são mostradas as características da aplicação do *AGC* em duas fases do estudo. Finalmente, resultados de testes computacionais são mostrados e comentados.

5.1-DESCRIÇÃO DO PROBLEMA

Uma aplicação importante da tecnologia de grupo é o desenvolvimento de um sistema de manufatura celular em que peças similares são agrupadas em famílias e máquinas são agrupadas em células. A célula ideal é independente, isto é, famílias de peças são completamente produzidas dentro da célula. Tais sistemas proporcionam benefícios como simplificação de controle, de implementação e de automação, redução dos tempos de preparação, do tempo entre entrada de material e saída de produto, redução de manejo de material; além disso, esses sistemas contribuem para o aumento da qualidade do produto final.

Os métodos para formação de células de manufatura podem ser classificados como orientados pelo projeto ou pela produção. Enquanto os métodos orientados por projeto agrupam partes baseando-se em características de seu projeto, os métodos orientados

por produção o fazem baseando-se nos processos requeridos para sua produção. Este trabalho tem foco em técnicas de formação de células orientadas pela produção.

Os primeiros trabalhos nesta área foram feitos por Mitrofanov (Mitrofanov, 1966) e Burbidge (Burbidge, 1971, 1975). A análise do fluxo de produção de Burbidge (Burbidge, 1971) é uma das primeiras e mais reconhecidas metodologias associadas com tecnologia de grupo. Ainda sobre controle de fluxo, a literatura traz referências sobre o trabalho de El-Essawy e Torrance (El-Essawy e Torrance, 1972). O objetivo dessas técnicas é obter células de máquinas independentes, minimizando o transporte de peças entre as células (Logendran, 1990).

Há muitos métodos para tratar o problema, e uma boa parte deles opera sobre uma matriz peças/máquinas cujos elementos são zeros e uns, indicando quais máquinas são usadas na produção de cada peça (Figura 5.1). Sendo *A* a matriz *0-1* assim descrita, as linhas correspondem às peças e as colunas às máquinas (ou vice-versa) e se $a_{ij} = 1$ então a peça *i* necessita da máquina *j* para sua produção.

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

Fig. 5.1: Matriz original de peças e máquinas.

Os algoritmos manipulam linhas e colunas da matriz tentando produzir pequenos blocos de uns agrupados (Figura 5.2). Cada bloco corresponde a uma célula de máquinas que produz uma família de partes. O objetivo básico é formar blocos densos, isto é, sem zeros em seu interior, e também evitar uns fora dos blocos.

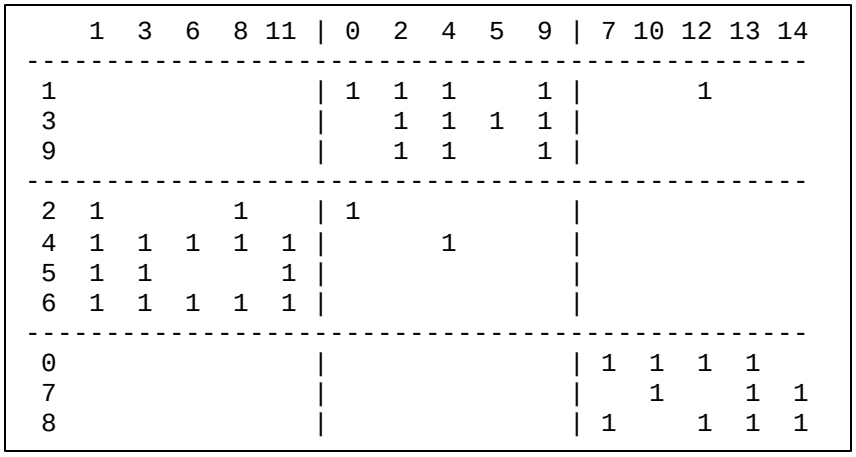

Fig. 5.2: Matriz de partes e máquinas processada.

Chandrasekharan e Rajagopalan (1989) e Venugopal e Narendran (1993) apresentaram análises na matriz *0-1* de modo a extrair propriedades para criação de algoritmos de formação de células. Algoritmos baseados em manipulação da matriz *0-1* podem ser vistos nos trabalhos de McCormick e seus colaboradores (McCormick *et al*., 1972), King (1980a, 1980b), King e Nakornchai (1982), e Chan e Milner (1982). Chu e Tsai (1990) apresentam um trabalho comparativo sobre técnicas desse tipo.

Uma grande variedade de abordagens tem sido experimentada no tratamento do problema. Técnicas de agrupamento hierárquico foram estudadas, sejam do tipo *divisivo* como em Stanfel (1985), ou *aglomerativo* como em McAuley (1972). Agrupamento não-hierárquico também foi estudado por Chandrasekharan e Rajagopalan (1986).

Técnicas baseadas em duplicação de máquinas podem ser vistas nos trabalhos de Seifoddini (1989), Logendran (1992) e Sofianopoulou (1999).

Técnicas de agrupamento baseadas em grafos foram estudadas e aplicadas desde os primeiros trabalhos de Rajagopalan e Batra (1975).

Técnicas de Inteligência Artificial como redes neurais artificiais foram aplicadas no trabalho de Malave e Ramchandran (1991), e lógica *fuzzy* no trabalho de Xu e Wang (1989).

Heurísticas mais genéricas como *Simulated Annealing* têm sido empregadas, como se vê nos trabalhos de Vakharia e Chang (1990) e Venugopal e Narendran (1992a). Algoritmos Genéticos foram aplicados nos trabalhos de Venugopal e Narendran (1992b) e Joines (1993).

De modo geral, muitos métodos têm sido utilizados na abordagem desse problema (Kumar e Vannelli, 1983; Seifoddini e Wolfe, 1986; Morris e Tersine, 1990; Wei e Gaither, 1990), e análises dos métodos podem ser encontradas nos trabalhos de Wemmerlov e Hyer (1989a, 1989b), de Miltenburg e Zhang (1991), e mais recentemente no trabalho de Sarker e Mondal (1999).

5.2-CARACTERÍSTICAS DA APLICAÇÃO DO *AGC*

Com base na generalização do *AGC* para problemas de agrupamento, foi feita uma adaptação para o problema de formação de células de manufatura. Com a especificação *a priori* do número de células desejado, a idéia é formar agrupamentos de máquinas e peças simultaneamente, alterando a posição das linhas e das colunas na matriz *0-1*, que representa o problema. Os primeiros resultados com essa adaptação foram apresentados por Ribeiro Filho e Lorena (1998); posteriormente alterações nas características do *AGC* foram feitas e novos resultados foram obtidos.

A analogia com os problemas de agrupamento ocorre pelo fato de que o algoritmo tenta encontrar *p* peças (ou linhas da matriz), de modo que cada uma das demais peças possa ser associada a uma delas de acordo com uma medida de "*distância*" entre as linhas que as representam, formando assim agrupamentos de peças com a menor soma de "distâncias" possível. O mesmo é aplicado às máquinas (ou colunas da matriz), simultaneamente.

5.2.1-REPRESENTAÇÃO

Para representar um *esquema* ou uma *estrutura* foi utilizada uma cadeia de $n+m$ símbolos, onde n é o número de peças (ou linhas da matriz) e m é o número de máquinas (ou colunas da matriz). Cada uma das duas porções do esquema evolui de forma independente da outra. São considerados apenas três símbolos possíveis: o símbolo *1*, que serve para indicar uma máquina ou peça *semente* para formação de um agrupamento; o símbolo *0*, para indicar uma peça ou máquina associada a uma semente; e o símbolo # (*do-not-care)*, para indicar peças ou máquinas ainda não associadas a sementes. Ambas as porções do esquema (peças e máquinas) possuem exatamente o mesmo número de *1*'s, equivalente ao número de agrupamentos que se deseja formar, e o restante das posições no esquema terá ou o símbolo *0*, ou o símbolo # . Uma estrutura terá apenas *1*'s e *0*'s.

Considerando o exemplo visto na Figura 5.1, com 10 peças (0-9) e 15 máquinas (0-14), os esquemas teriam então 25 posições. Se considerarmos as partes (linhas) 6, 8 e 9 como sementes, e as máquinas (colunas) 3, 9 e 13 como sementes, uma possível representação para um esquema S_i está ilustrada na Figura 5.3 .

(#,0,0,0,#,0,1,#,1,1 / 0,0,0,1,#,#,0,#,0,1,0,0,0,1,#)

Fig. 5.3 – Esquema com peças e máquinas.

No esquema da Figura 5.3, os 10 primeiros símbolos se referem a peças e os 15 símbolos restantes se referem a máquinas. Considerando $V_1^p(S_i)=\{6,8,9\}$, o conjunto de índices das peças *sementes*, e $V_2^p(S_i)=\{1,2,3,5\}$ o conjunto de índices das peças associadas às sementes, bem como os conjuntos $V_1^m(S_i)=\{3,9,13\}$ e $V_2^m(S_i)=\{0,1,2,6,8,10,11,12\}$ no caso das máquinas, os agrupamentos de peças e

máquinas serão então formados associando os elementos cujos índices estão nos conjuntos $V_2^p(S_i)$ e $V_2^m(S_i)$ aos elementos cujos índices estão nos conjuntos $V_1^p(S_i)$ e $V_1^m(S_i)$, respectivamente.

5.2.2-DISTÂNCIA

Para a associação de peças e máquinas com sua semente mais "próxima", é necessária uma medida de *dissimilaridade*, ou "distância", entre as linhas e entre as colunas da matriz. Nos testes foi utilizada uma medida baseada no *coeficiente de Jaccard*. O coeficiente de similaridade de Jaccard entre seqüências binárias é definido como o número de posições com valor 1 em ambas as seqüências, dividido pelo número de posições com valor 1 em ambas ou apenas em uma das seqüências. Esse coeficiente pode ser convertido em medida de distância subtraindo-o de um. Por exemplo, considerando a matriz mostrada na Figura 5.1, a distância entre as colunas (ou máquinas) 1 e 3 seria $m_{13}=1-\frac{3}{4}=0.25$, enquanto a distância entre as colunas 1 e 4 seria $m_{14}=1-\frac{1}{7}=0.86$.

Considerando o exemplo mostrado na Figura 5.2, a partir de uma estrutura e depois das associações, seriam identificados os agrupamentos de peças $C_1^p(S_i)=\{1,3,9\}$, $C_2^p(S_i)=\{2,4,5,6\}$ e $C_3^p(S_i)=\{0,7,8\}$. Os agrupamentos de máquinas seriam $C_1^m(S_i)=\{1,3,6,8,11\}$, $C_2^m(S_i)=\{0,2,4,5,9\}$ e $C_3^m(S_i)=\{7,10,12,13,14\}$.

5.2.3-POPULAÇÃO INICIAL

Do mesmo modo como foi feito para Coloração de Grafos (Seção 4.3.1), uma população inicial foi gerada aleatoriamente com apenas 20% das linhas e 20% das colunas compondo os esquemas com símbolos *0* e exatamente *k* (número de células) símbolos *1*

para as peças e *k* símbolos *1* para as máquinas. As posições que receberam os símbolos *0* e *1* foram determinadas aleatoriamente.

5.2.4-RECOMBINAÇÃO E MUTAÇÃO

Conforme princípios do *AGC*, a população foi mantida ordenada de acordo com um critério que privilegia esquemas mais próximos de uma estrutura e com melhor adaptação, isto é, menor "distância" entre os componentes de cada agrupamento de máquinas ou peças, e o método de seleção utilizado foi do tipo *base-guia*. A recombinação também obedeceu aos princípios já expostos na Seção 3.5.

Na primeira fase dos testes com o *AGC*, antes das alterações recentemente introduzidas, a população armazenava tanto esquemas quanto estruturas. Se o primeiro indivíduo selecionado fosse uma estrutura, aplicava-se apenas uma mutação e o resultado era comparado com o melhor obtido até o momento. Como mutação, foi usado um tipo de busca local em que escolhíamos aleatoriamente uma das sementes e a trocávamos de lugar na seqüência de representação, experimentando todas as posições possíveis e deixando-a na posição em que houvesse maior melhoria na solução. Caso o primeiro indivíduo selecionado fosse um esquema, um indivíduo guia era selecionado e a recombinação entre ambos era feita conforme descrito na Seção 3.5.

Na segunda fase dos estudos, a população passou a ser formada apenas por esquemas. Um novo processo de mutação foi aplicado tanto às estruturas obtidas com a complementação do esquema base, quanto às estruturas obtidas na recombinação. A mutação foi implementada como um processo iterativo de sucessivas alterações na posição da semente, tanto de peças quanto máquinas, para posições melhores dentro do agrupamento. A Figura 5.4 traz um algoritmo que ilustra o processo.

Repetir
S' ¬ S
Para *cada agrupamento de peças em S'*
udar a semente para onde der a menor soma de distâncias no agrupamento
Para *cada agrupamento de máquinas em S'*
muda a semente para onde der a menor soma de distância no agrupamento
S'' ¬ realocação dos vértices não sementes em S'
Se *S'' é melhor que S* **então**
S ¬ S''
Até *S é melhor que S''*
Retorna *S*

Fig. 5.4 - Processo de mutação iterativo.

Em ambas as fases de estudo, a cada iteração do algoritmo, $n+m$ novas soluções eram inseridas na população e a eliminação dos piores indivíduos era feita com base nos princípios básicos do *AGC* (Seção 3.4).

5.2.5-FUNÇÕES DE AVALIAÇÃO

A seguinte formulação para a função *g* foi mantida em ambas as fases do estudo:

$$g(S_i) = \sum_{a=1}^{k} \sum_{j \in C_a^p(S_i)} m_{z_a j} + \sum_{a=1}^{k} \sum_{j \in C_a^m(S_i)} m_{z_a j} \qquad (5.1)$$

onde: z_a é a semente do agrupamento a (peças ou máquinas); e

$m_{z_a j}$ é a distância entre a semente z_a e a peça ou máquina *j*.

Essa formulação da função *g* equivale à soma de todas as distâncias entra cada elemento e a semente de seu agrupamento, tanto de peças quanto de máquinas.

Na primeira fase dos estudos, a formulação utilizada para a função f foi a seguinte:

$$f(S_i) = \sum_{a=1}^{k} \min_{j \in C_a^p(S_i)} \{m_{z_a j}\} \cdot \left[\left| C_a^p(S_i) \right| - 1 \right] + \sum_{a=1}^{k} \min_{j \in C_a^p(S_i)} \{m_{z_a j}\} \cdot \left[\left| C_a^m(S_i) \right| - 1 \right] \quad (5.2)$$

onde: z_a *e* $m_{z_a j}$ são os mesmos da expressão 5.1; e

$\left| C_a^p(S_i) \right|$ e $\left| C_a^m(S_i) \right|$ representam a cardinalidade dos conjuntos.

A formulação da função f equivale a somar, para cada agrupamento, tanto de peças quanto de máquinas, a menor distância entre uma peça ou uma máquina e a semente daquele agrupamento, multiplicada pelo número de elementos do agrupamento, descontando a semente. Isto é, somam-se todas as distâncias entre os elementos de cada agrupamento e sua respectiva semente, considerando como se todas essas distâncias fossem iguais à menor delas encontrada no agrupamento.

Com a formulação acima, implicitamente, o objetivo do *AGC* de diminuir a diferença *g-f* implica a formação de agrupamentos com somas cada vez menores das distâncias entre cada elemento e sua semente. No entanto, o *AGC* não apresentou resultados satisfatórios porque em determinadas instâncias a menor distância dentro do agrupamento era muito menor do que as demais, iguais a zero em muitos casos, considerando a métrica utilizada. Por esse motivo, outra formulação para a função f foi utilizada numa segunda fase dos estudos.

Na segunda fase dos estudos, uma nova formulação para a função *f* foi utilizada:

$$f(S_i) = g(S_i) - \left[\sum_{a=1}^{k} \max_{j \in C_a^p(S_i)} \{m_{z_a j}\} + \sum_{a=1}^{k} \max_{j \in C_a^p(S_i)} \{m_{z_a j}\} \right] \quad (5.3)$$

onde z_a e $m_{z_a j}$ são os mesmos das expressões 5.1 e 5.2.

A formulação representa agora apenas a diferença entre $g(S_i)$ e a soma das maiores distâncias em cada agrupamento, tanto de partes quanto de máquinas.

Em ambas as fases do estudo, a estimativa do limite superior g_{max} foi feita aplicando a função g sobre uma estrutura gerada aleatoriamente, o que, em princípio, deve ser uma solução ruim.

5.3-TESTES COMPUTACIONAIS

Foi grande a dificuldade para encontrar disponíveis as instâncias citadas na literatura. Sendo assim, da literatura foram usadas nos testes apenas duas instâncias, uma de dimensões *20x35* (20 peças e 35 máquinas) (Burbidge, 1975), e outra de dimensões *40x100* (40 peças e 100 máquinas) (Chandrasekharan e Rajagopalan, 1986). Foram também geradas instâncias aleatórias de tamanhos variáveis.

Como *medida de desempenho* considerou-se um coeficiente que usa a quantidade de zeros dentro dos agrupamentos e a quantidade de uns fora deles, em relação ao total de uns da matriz que representa o problema. Quanto maior o valor do coeficiente, melhor a solução.

$$Coef = \frac{e - e_1}{e + e_0} \tag{5.4}$$

onde: e = número de *1's* na matriz;
e_0 = número de *0's* dentro das células; e
e_1 = número de *1's* fora das células.

Para as instâncias geradas aleatoriamente foram especificados o número de peças, número de máquinas, *densidade intracelular* (*WCD*) e *densidade intercelular* (*ICD*). Como *WCD* entende-se a razão entre o número de *1*'s dentro da célula e o tamanho da célula. Como *ICD* entende-se a razão entre o número de *1*'s fora de qualquer célula e o

número de posições fora de qualquer célula. Inicialmente a matriz é gerada com as células definidas da forma descrita, para em seguida ser perturbada com várias trocas de linhas e colunas de forma aleatória. A matriz perturbada é então usada nos testes.

Na Tabela 5.1 são vistos resultados de testes com duas instâncias tiradas da literatura. A tabela mostra as dimensões da instância, o número de células que se procurou formar (a partir de resultados da literatura), os coeficientes vistos da literatura e de ambas as fases de estudo com o *AGC*. Foram feitas três execuções para cada instância.

TABELA 5.1: TESTES COM INSTÂNCIAS DA LITERATURA

Instância	*Pec/maq*	*Cel.*	$Coef_{Lit}$	$Coef_{AGC}$ *(1a fase)*	*Gap (%)*	*Tempo (seg)*	$Coef_{AGC}$ *(2a fase)*	*Gap (%)*	*Tempo (seg)*
Burbidge	20/35	4	0.7571	0.7571	-	14	0.7571	-	28
				0.7571	-	12	0.7571	-	46
				0.7571	-	13	0.7571	-	49
Chandra	40/100	10	0.8403	0.7019	16.5	317	0.8403	-	1561
				0.7378	12.2	877	0.8403	-	535
				0.7642	9.1	739	0.8403	-	939

Os programas usados em ambas as fases foram escritos em linguagem C, e a plataforma utilizada foi um PENTIUM II de 266 MHz.

Nos primeiros testes com o *AGC* os resultados foram bons para o problema pequeno (*Burbidge*) e apenas razoáveis para o problema maior (*Chandra*). Porém, com as melhorias do algoritmo, principalmente pelo uso de uma nova heurística de mutação, os resultados da literatura (Joines, 1993) foram igualados em todas as rodadas de ambos os problemas.

Foram feitos em seguida alguns testes com as instâncias geradas aleatoriamente para tentar detectar alguma influência das densidades intercelular e intracelular sobre o desempenho do algoritmo. Também foram feitas três rodadas para cada problema.

As Tabelas 5.2 e 5.3 mostram que a densidade intracelular (*WCD*) parece não afetar o desempenho do algoritmo, enquanto a densidade intercelular (*ICD*) começa a afetar o desempenho do *AGC* apenas quando atinge os 10% na primeira fase de estudos. As tabelas mostram as características de cada instância, os valores de *ICD* e *WCD*, os coeficientes obtidos em ambas as fases de estudo e tempo de execução. Foram feitas três execuções para cada instância.

TABELA 5.2: TESTES DE SENSIBILIDADE AO *ICD* COM *WCD=0.8*

Instância	*Pec/ maq*	*Cel*	*ICD*	*Coef (Lit.)*	*Coef-AGC (1a fase)*	*Gap (%)*	*Tempo (seg)*	*Coef-AGC (2a fase)*	*Gap (%)*	*Tmp (seg)*
W80i02	20/35	4	0.02	0.7527	0.7527	-	19	0.7527	-	35
					0.7527	-	13	0.7527	-	76
					0.7527	-	21	0.7527	-	99
W80i03	20/35	4	0.03	0.7330	0. 7330	-	9	0. 7330	-	62
					0. 7330	-	12	0. 7330	-	62
					0. 7330	-	9	0. 7330	-	74
W80i05	20/35	4	0.05	0.6931	0. 6931	-	13	0. 6931	-	39
					0. 6931	-	21	0. 6931	-	61
					0. 6931	-	24	0. 6931	-	80
W80i10	20/35	4	0.10	0.6140	0.5462	11.0	12	0. 6140	-	27
					0.5527	10.0	30	0. 6140	-	57
					0.6000	2.3	19	0. 6140	-	57

TABELA 5.3: TESTES DE SENSIBILIDADE AO *WCD* COM *ICD=0.02*

Instância	*Pec/ maq*	*Cel*	*WCD*	*Coef (Lit.)*	*Coef-AGC (1a fase)*	*Gap (%)*	*Tmp (Seg)*	*Coef-AGC (2a fase)*	*Gap (%)*	*Tmp (seg)*
W70i02	20/35	4	0.7	0.6398	0. 6398	-	20	0. 6398	-	32
					0. 6398	-	28	0. 6398	-	67
					0. 6398	-	15	0. 6398	-	70
W80i02	20/35	4	0.8	0.7527	0. 7527	-	8	0. 7527	-	29
					0. 7527	-	11	0. 7527	-	40
					0. 7527	-	26	0. 7527	-	58
W90i02	20/35	4	0.9	0.8280	0. 8280	-	18	0. 8280	-	35
					0. 8280	-	9	0. 8280	-	51
					0. 8280	-	15	0. 8280	-	49

Em todos os testes, tanto com instâncias da literatura quanto com densidades, os parâmetros do *AGC* utilizados foram: limite de iterações = 150; D_a = 0.01 e d = 0.1.

Como critério de parada do processo evolutivo foram consideradas duas situações: atingir o limite máximo de iterações, ou eventualmente a população ficar vazia.

5.4 – CONSIDERAÇÕES FINAIS DO CAPÍTULO

O problema tratado neste Capítulo se caracteriza pela troca de posição de linhas e colunas de uma matriz binária que representa conjuntos de máquinas em que peças são produzidas.

Da mesma forma como no Capítulo anterior, o problema foi considerado como sendo de formação de agrupamentos e o estudo foi realizado em duas fases; e na segunda fase, as alterações nas características do *AGC* melhoraram os resultados.

A heurística de associação para formação dos agrupamentos fez uso de uma métrica para cálculo de dissimilaridade entre seqüências binárias baseada no coeficiente de Jaccard.

As formulações das funções de avaliação dos esquemas e estruturas foram mudadas de uma fase dos estudos para outra, uma vez que a medida de dissimilaridade utilizada fazia com que a menor distância dentro dos agrupamentos fosse muito menor do que as demais, e até zero em alguns casos.

Em ambas as fases, uma heurística de busca local foi utilizada sobre estruturas para melhoria da qualidade das soluções.

Como medida de qualidade dos agrupamentos produzidos, foi utilizado um coeficiente encontrado na literatura que considera tanto a densidade dos agrupamentos quanto as máquinas que eventualmente ficam fora das células.

Apesar da dificuldade na obtenção de instâncias de teste para comparação, em dois casos presentes na literatura o AGC produziu resultados de mesma qualidade que os obtidos com outras abordagens.

Ainda neste Capítulo foi feita uma análise da sensibilidade do AGC em relação à densidade intracelular e intercelular dos agrupamentos produzidos, e os resultados mostram que o AGC tem baixa dependência desses parâmetros. Nesse caso, foram usadas instâncias geradas aleatoriamente e com densidades especificadas.

O problema requer mais estudos, principalmente na análise dos efeitos dos parâmetros do algoritmo sobre os resultados. São necessários mais dados de teste presentes na literatura para efeito de comparação.

O apêndice B traz algumas figuras que ilustram o efeito do algoritmo aplicado sobre as matrizes partes/máquinas.

CAPÍTULO 6

APLICAÇÃO DO *AGC* AO PROBLEMA DE FORMAÇÃO DE HORÁRIOS ESCOLARES

O problema de formação de horários escolares, também conhecido pelo termo *Timetabling*, consiste em arranjar encontros entre professores e alunos em um período de tempo previamente fixado, tipicamente uma semana, de modo a satisfazer um conjunto de restrições que podem ser de vários tipos.

A solução manual do problema usualmente requer vários dias de trabalho. Ainda assim, a solução encontrada pode não ser satisfatória sob algum aspecto, por exemplo um aluno pode estar impedido de cursar duas determinadas disciplinas de seu interesse porque estão marcadas no mesmo horário e dia da semana.

Este Capítulo mostra como o problema foi modelado como sendo de formação de agrupamentos e como o *AGC* foi aplicado. São feitas considerações iniciais sobre o problema, suas variantes e abordagens; são descritas as características da aplicação do *AGC* e são mostrados resultados experimentais com instâncias reais.

6.1 – FORMULAÇÕES DO PROBLEMA

Muitas variantes do problema de *Timetabling* têm sido propostas na literatura, e diferem umas das outras pelo tipo de instituição de ensino envolvida, universidades ou escolas médias, e pelo tipo de restrições impostas ao problema. Schaerf (1999a) cita três classes de problemas:

School Timetabling: seqüenciamento semanal das aulas de uma escola, evitando que professores e alunos tenham mais de uma aula simultaneamente;

Course Timetabling: seqüenciamento semanal das aulas de um conjunto de cursos de uma universidade, evitando a simultaneidade de cursos com estudantes em comum; e

Examination Timetabling: seqüenciamento de exames de um conjunto de cursos em uma universidade, evitando exames simultâneos de cursos com estudantes em comum, e espalhando os exames o máximo possível.

Essa divisão não é tão restrita e pode haver situações com características particulares que se enquadram entre as classes citadas ou parcialmente em mais de uma delas.

6.2 – ABORDAGENS DO PROBLEMA

O problema de *Timetabling* é *NP-Hard* (Even, *et al*., 1976) e várias técnicas heurísticas têm sido experimentadas para automação de *Timetabling*.

Durante mais de trinta anos, desde os trabalhos iniciais de Gotlieb (1963), vários artigos têm aparecido em conferências e publicações, e vários sistemas têm sido desenvolvidos (Schaerf, 1999a).

A maioria das técnicas mais antigas (Schmidt e Strohlein, 1979) eram baseadas na simulação do comportamento humano na solução manual do problema. Nesses casos, uma solução parcial é ampliada passo a passo, até todas as aulas, todos os cursos ou todos os exames terem sido sequenciados. A estratégia gulosa utilizada era sequenciar primeiro os elementos com mais restrições.

Associações do problema de *Timetabling* com Coloração de Grafos são comuns e técnicas de abordagem que fazem uso dessa associação têm sido desenvolvidas (Neufel e Tartar, 1974).

Recentemente, técnicas baseadas em meta-heurísticas têm sido empregadas. São encontrados trabalhos com *Simulated Annealing* (Abramson, 1991), Busca Tabu (Costa, 1994; Schaerf, 1996 e 1999b) e Algoritmos Genéticos (Coloni, *et al*., 1998). Souza, Maculan e Ochi (2000) apresentam uma técnica tipo GRASP (Feo e Resende, 1995).

6.3 – PARTICULARIDADES E RESTRIÇÕES

O objetivo deste trabalho é produzir um método para tratar especificamente da variante citada como *School Timetabling*, porém com características com que ela se apresenta em escolas públicas brasileiras de ensino básico e médio.

Tipicamente, as escolas de ensino básico ou médio atendem a um determinado número de turmas que é limitado por sua capacidade física. Normalmente há mais turmas do que salas de aula, mas as escolas trabalham em mais de um período por dia, normalmente os períodos matutino, vespertino e noturno.

Cada turma possui uma relação de disciplinas que têm um certo número de aulas dependendo do currículo do curso e série aos quais a turma pertence. A quantidade de aulas de cada turma preenche completamente a semana, isto é, as turmas têm aulas em todos os horários de seu período, todos os dias da semana.

O número de horários do período e o número de dias da semana, multiplicados pelo número de turmas da escola nos dão o número total de aulas ministradas naquele período. Se considerarmos todos os períodos do dia teremos todas as aulas ministradas na escola.

Essas aulas são ministradas por um conjunto de professores que trabalham na escola. Cada professor tem seu próprio número de aulas. Muitas vezes os professores trabalham em mais de uma escola e em cada uma delas ministram um diferente número de aulas. Uma escola pode ter professores que trabalhem em um único período e professores que trabalhem em mais de um período.

Neste trabalho, cada período de aulas numa escola foi considerado como uma instância independente.

Podemos enumerar algumas características próprias desse contexto: todas as salas disponíveis na instituição são utilizadas; os alunos têm aula em todos os horários disponíveis do período; e conflitos de simultaneidade podem ocorrer tanto com professores quanto com turmas.

A Figura 6.1 mostra uma grade de aulas de uma turma em que todos os horários disponíveis estão ocupados.

	Seg	**Ter**	**Qua**	**Qui**	**Sex**
Aula 1	EdArt MLuc	Port Cris	Hist Cida	Hist Cida	Cienc Marg
Aula 2	Port Cris	Port Cris	Ingl Neja	Mat Jo	Port Cris
Aula 3	Geo Jane	Hist Cida	Cienc Marg	Mat Jo	Mat Jo
Aula 4	Mat Jo	Geo Jane	Port Cris	Mat Jo	Cienc Marg
Aula 5	Port Cris	Ingl Neja	EdArt MLuc	Geo Jane	Mat Jo

Fig. 6.1 – Grade de aulas de uma turma.

Podemos considerar uma intensidade para as restrições. As restrições mais “fortes” são aquelas que garantem a viabilidade de uma solução, isto é, aquelas referentes aos conflitos de simultaneidade de aulas tanto para professores quanto para turmas.

Além dessas restrições consideradas “fortes”, podem ser consideradas outras restrições mais “fracas” cuja não-satisfação não acarreta necessariamente inviabilidade da solução. Neste trabalho, foram consideradas restrições desse tipo referentes a situações que envolvem: preferência de determinados horários do dia ou semana por alguns professores; e horários vagos para os professores.

A Figura 6.2 mostra uma grade de aulas de um professor em que os horários livres foram colocados no início de cada dia, evitando janelas entre aulas.

	Seg	Ter	Qua	Qui	Sex
Aula 1					
Aula 2		Mat 5D	Mat 5C	Mat 5B	
Aula 3	Mat 5D	Mat 5C	Mat 5D	Mat 5B	Mat 5B
Aula 4	Mat 5B	Mat 5C	Mat 5D	Mat 5B	Mat 5D
Aula 5	Mat 5C	Mat 5C	Mat 5C	Mat 5D	Mat 5B

Fig. 6.2 – Grade de aulas de um professor.

Quantificar a intensidade das restrições é uma tarefa que depende da situação de cada instituição de ensino. A eliminação de janelas nas grades dos professores poderia ser uma necessidade imposta por lei. A preferência de determinados horários por professores pode estar relacionada a outras atividades por eles exercidas em situações em que a dedicação exclusiva a uma instituição não seja obrigatória. Podemos pensar ainda em situações em que determinados horários são proibitivos até por motivos religiosos.

Há muitas variantes que podem determinar se uma restrição é mais forte ou mais fraca, se implica (ou não) inviabilidade de uma solução. Por essa razão, o algoritmo mostrado neste trabalho trata as restrições com pesos que podem ser ajustados para cada contexto.

6.4 – CARACTERÍSTICAS DA APLICAÇÃO DO *AGC*

Neste estudo a aplicação do *AGC* foi feita sobre uma variante do problema, que considera um professor associado a uma aula de uma determinada disciplina dada para uma certa turma.

O problema foi considerado como sendo de formação de agrupamentos. Para cada aula foi criada uma seqüência binária que representa uma dupla formada pelo professor e pela turma referentes àquela aula.

Essas seqüências binárias têm duas partes: uma representando o professor e outra a turma. Na primeira parte, cada posição equivale a um professor e apenas o professor que forma aquela dupla tem na sua posição o valor 1, todos os demais têm valor 0. De modo análogo, a aula ministrada por esse professor é representada na segunda parte da seqüência.

A Figura 6.3 mostra uma seqüência binária que representa uma dupla de uma escola que teria 4 professores e 13 turmas.

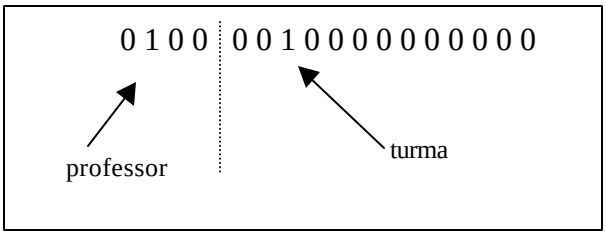

Fig. 6.3 – Exemplo de dupla professor/turma.

O objetivo foi então criar agrupamentos dessas duplas, um para cada horário de aula do período, evitando conflitos, isto é, colocando as duplas em agrupamentos que não contivessem outras duplas com o mesmo professor ou a mesma turma.

Como exemplo podemos ter uma escola que no período matutino atende a 11 turmas. Cada turma tem 6 aulas por dia, cinco dias da semana. Se todas as turmas tiverem aulas em todos os horários, teremos um total de 6x5x11=330 aulas na semana para serem alocadas em 6x5=30 agrupamentos de exatamente 11 aulas cada um. Essas aulas poderiam ser ministradas por 15 professores, alguns deles com mais aulas do que os outros.

6.4.1 – REPRESENTAÇÃO

A representação usada para os esquemas e estruturas do *AGC* foi a mesma dos Capítulos anteriores e válidas para problemas de agrupamento em geral.

Seja *m* o número de aulas ministradas numa escola durante a semana num determinado período. Seja ainda *k* o número de horários ou agrupamentos de aulas (duplas professor/turma) que se deseja formar.

Os esquemas e estruturas são então seqüências de *m* símbolos pertencentes ao alfabeto *{0,1,#}*. Cada esquema ou estrutura tem exatamente *k* posições com símbolo *1*, indicando duplas professor/turma que são sementes para formação de agrupamentos. As posições restantes têm os símbolos *0* ou *#*.

6.4.2 – ASSOCIAÇÃO

Como nas aplicações do *AGC* vistas nos Capítulos anteriores, a formação dos agrupamentos foi feita com associações entre as duplas presentes em cada esquema. No entanto, ao invés de associar cada dupla não semente a uma dupla que seja semente, nesta aplicação a associação foi feita entre a dupla e o agrupamento, representado por todas as duplas a ele já associadas.

Cada agrupamento é também representado por uma seqüência binária. Essa seqüência é obtida mesclando as seqüências das duplas já pertencentes ao agrupamento. Trata-se basicamente de uma operação lógica OU (disjunção inclusiva) aplicada sobre todas as seqüências das duplas associadas ao agrupamento. Inicialmente a única dupla de um agrupamento é a semente usada para sua formação.

A Figura 6.4 ilustra os resultados da mescla de duas seqüências de um mesmo agrupamento.

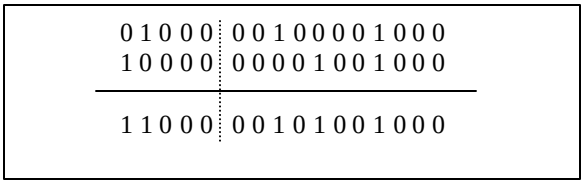

Fig. 6.4 – Mescla de seqüências binárias.

Para associação das duplas é necessária uma maneira de medir a "dissimilaridade" ou "distância" entre as seqüências binárias que as representam. A medida da dissimilaridade D_{pq} entre duas seqüências binárias p e q foi feita de acordo com a expressão:

$$D_{pq} = 1 - \frac{\sum_i \left| a_i^p - a_i^q \right|}{\sum_i \left(a_i^p + a_i^q \right)} \tag{6.1}$$

onde a_i^j corresponde à i-ésima posição da j-ésima seqüência .

A expressão 6.1 tem valor máximo apenas quando não houver nenhuma posição em que apenas uma das seqüências tenha valor 1.

As duplas do esquema ou da estrutura são associadas aos agrupamentos em uma determinada ordem, a qual considera uma precedência que alguns professores podem ter sobre os outros. Professores mais antigos ou com mais aulas na escola podem ter precedência sobre os demais na elaboração do horário e no atendimento de suas preferências. Se os professores têm a mesma precedência, são primeiro considerados aqueles com maior número de restrições de horários.

6.4.3 – POPULAÇÃO INICIAL, SELEÇÃO E RECOMBINAÇÃO

A população inicial foi gerada com 100 esquemas, cada um deles contendo exatamente k sementes em posições aleatórias e 20% das demais duplas associadas a algum agrupamento.

Conforme os princípios do *AGC*, a população foi mantida ordenada de acordo com um critério que privilegia esquemas mais próximos de estruturas e com melhor adaptação, isto é, menos conflitos dentro dos agrupamentos.

O método de seleção utilizado foi do tipo *base-guia*, da mesma forma que nas aplicações descritas nos Capítulos anteriores.

A recombinação também obedeceu aos princípios já expostos na Seção 3.5, sempre mantendo o número de sementes no esquema ou estrutura resultante.

6.4.4 – FUNÇÕES DE AVALIAÇÃO

De acordo com os princípios do *AGC*, para avaliar os esquemas ou estruturas S_i foram usadas as funções f e g. No entanto, diferentemente das aplicações descritas nos Capítulos anteriores, e porque esse problema envolve restrições de intensidades diferentes, foram usadas três formulações para as funções f e g.

A primeira formulação envolve apenas as restrições fortes de viabilidade, isto é, considera apenas conflitos de simultaneidade gerados por professores que estariam ministrando mais de uma aula ao mesmo tempo e turmas que teriam mais de uma aula ao mesmo tempo.

Nesse primeiro caso, as formulações utilizadas foram:

$$g_1(S_i) = \sum_{p=1}^{k} \left[\frac{\left(\left| C_p(S_i) \right| - 1 \right) \left| C_p(S_i) \right|}{2} \right] \qquad (6.2)$$

onde $C_p(S_i)$ é o p-ésimo agrupamento de duplas.

$$f_1(S_i) = g(S_i) - \sum_{p=1}^{k} \left| Conf\left(C_p(S_i) \right) \right| \quad (6.3)$$

onde *Conf(.)* representa o número de conflitos no agrupamento.

A segunda formulação foi utilizada para as restrições referentes às preferências dos professores por determinados horários. As formulações utilizadas foram:

$$g_2(S_i) = \textit{número total de duplas} \quad (6.4)$$

$$f_2(S_i) = g_2(S_i) - \textit{número de duplas com conflito de preferência} \quad (6.5)$$

A terceira formulação envolve conflitos referentes às restrições de janelas nas grades dos professores, isto é, situações em que os professores ficam sem aulas em horários intermediários entre sua primeira e última aula do período em um dia. As formulações utilizadas foram:

$$g_3(S_i) = \textit{número total de duplas} \quad (6.6)$$

$$f_3(S_i) = g_3(S_i) - \textit{número de janelas} \quad (6.7)$$

Um único limitante superior g_{max} foi utilizado para as três formulações de g. O limitante foi o mesmo da aplicação para Coloração de Grafos e sua formulação é:

$$g_{\max} = \boldsymbol{f} \cdot k \cdot \left\lfloor \frac{\left\lfloor \frac{m}{k} - 1 \right\rfloor \cdot \left\lfloor \frac{m}{k} \right\rfloor}{2} \right\rfloor \quad (6.8)$$

onde: m é o número total de duplas professor/turma;
k é o número de horários ou agrupamentos; e
f é um fator para garantir o limitante superior.

Para cada uma das três formulações de f e g foi considerada uma medida da diferença $\{g(S_i)-f(S_i)\}$ em relação a g, dada por:

$$d_{ji} = \frac{g_j(S_i) - f_j(S_i)}{g(S_i)}, \qquad j=1, 2 \text{ ou } 3. \tag{6.9}$$

Para unificar essa medida foi feita uma combinação envolvendo dois pesos, w_{pref} e w_{jan}, ambos com valor entre 0 e 1. A formulação usada nessa combinação foi a seguinte:

$$d_i = \frac{d_{1i} + w_{pref} \cdot d_{2i} + w_{jan} \cdot d_{3i}}{1 + w_{pref} + w_{jan}} \tag{6.10}$$

A expressão 6.10 foi utilizada tanto no cálculo da chave de ordenação dos indivíduos da população para o processo de seleção quanto para o cálculo do *rank* de cada indivíduo.

6.4.5 – MUTAÇÃO

Sem dúvida essa aplicação do *AGC* envolve várias características diferentes daquelas descritas nos Capítulos anteriores. O processo de mutação utilizado sobre estruturas geradas na recombinação ou geradas com a complementação dos esquemas base selecionados teve como principal objetivo a garantia da viabilidade da solução, além de procurar a melhoria da qualidade da solução. O processo de mutação foi feito em três etapas na seguinte ordem:

Viabilidade de turmas – esta fase procura remover conflitos em que uma turma tem mais de uma aula sendo ministrada ao mesmo tempo. O processo varre seqüencialmente todos os agrupamentos, procurando duplas com turmas

repetidas. Para cada uma dessas duplas que for encontrada, nos agrupamentos restantes é procurada uma dupla cuja turma não esteja no presente agrupamento, e as duplas são trocadas. O processo pode ser visto no algoritmo ilustrado na Figura 6.5.

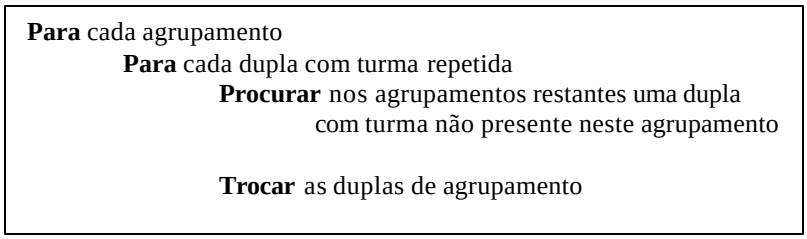

Fig. 6.5 – Mutação: viabilidade de turmas.

Viabilidade de professores – esta fase procura remover conflitos em que um professor tem mais de uma aula sendo ministrada ao mesmo tempo. Nesse caso, novamente os agrupamentos são varridos seqüencialmente à procura de duplas com professores repetidos. Para cada uma dessas duplas que for encontrada, todos os outros agrupamentos são sondados à procura de uma outra dupla, de mesma turma mas com outro professor. Se essa outra turma for encontrada e seu agrupamento não tiver ocorrência do professor repetido, as duplas são trocadas. O processo é visto no algoritmo ilustrado na Figura 6.6.

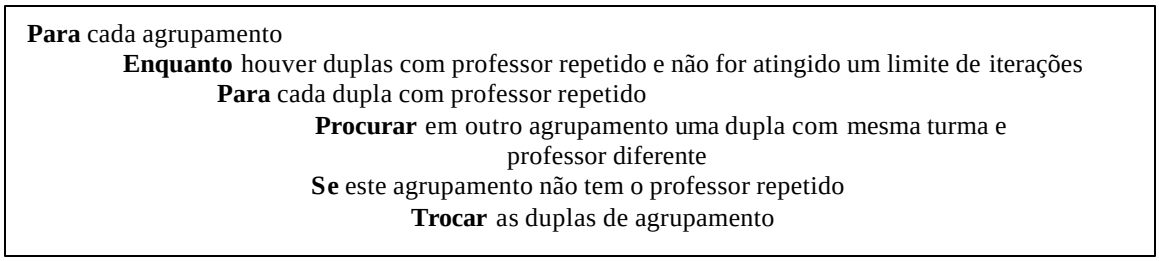

Fig. 6.6 - Mutação: viabilidade de professores.

Melhoria do atendimento de preferências – esta fase procura remover conflitos referentes a restrições mais fracas envolvendo preferências de professores por

determinados horários da semana. Nesse caso as duplas são consideradas ordenadas de acordo com o grau de precedência dos professores sobre seus colegas para formação do horário, no caso de professores com mesma precedência, de acordo com o número de restrições dos professores. Aqueles com maior precedência e mais restrições são considerados em primeiro lugar. Se a dupla estiver em um agrupamento referente a um horário em que o professor não pretende ministrar aulas, os outros agrupamentos sem restrições do professor são sondados à procura de uma dupla de mesma turma e outro professor. Se for encontrada essa dupla, elas são trocadas. O procedimento pode ser visto na Figura 6.7.

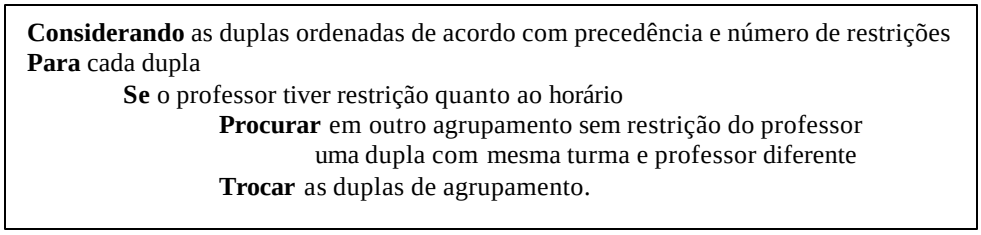

Fig. 6.7 - Mutação: melhoria de atendimento a preferências.

6.5 – TESTES COMPUTACIONAIS

Nos testes foram usadas duas escolas reais. Foram feitos testes com cada um dos três períodos de aulas da escola neste trabalho chamada *Gabriel* (da cidade de Lorena) e com um único período da escola aqui chamada *Massaro* (de Mogi das Cruzes), totalizando quatro instâncias reais.

Cada uma dessas instâncias tem seu próprio número de professores, turmas e restrições de preferências de horário.

Para efeito de testes, foram atribuídos níveis de precedência aos professores de acordo com o número de aulas que cada professor ministra na instância.

Os professores que ministram menos de 50% de todas as aulas do período receberam nível 3, o menor nível de precedência. Aqueles que ministram entre 50% e 75% das aulas receberam nível 2, e aqueles que ministram mais de 75% das aulas receberam o nível 1, o maior nível de precedência.

Com cada instância foram feitos testes variando os pesos w_{pref} e w_{jan} para tentar medir sua influência nos resultados. O peso w_{pref} foi testado com valores 0, 0.5 e 1.0, e para cada um desses valores, o peso w_{jan} foi testado também com valores 0, 0.5 e 1.0. Com cada uma dessas combinações de pesos foram feitos 3 testes. Portanto, com cada instância foram feitos 9x3=27 testes.

As Tabelas 6.1, 6.2, 6.3 e 6.4 mostram os resultados desses testes. Cada tabela traz o nome da instância e suas características; traz uma linha para cada combinação de pesos com a média dos 3 testes; traz colunas com os valores dos pesos, percentual de preferências de horários atendidas, percentual de preferências de horários de professores de nível 1 atendidas, número de janelas de professores na solução e número de janelas de professores de nível 1.

As tabelas trazem ainda a relação entre o número de professores e o número de turmas da escola para uma análise de uma eventual influência dessa razão sobre os resultados dos testes.

TABELA 6.1 – TESTES COM GABRIEL – MANHÃ

Gabriel *Manhã*	W_{pref}	W_{jan}	% prefer.	% prefer. (1)	Qtde Janelas	Qtde Janelas (1)	Tempo (Seg)
	0	0	89.39	83.33	55.00	12.33	718.67
	0	0.5	88.33	74.24	33.33	7.33	625.33
30 prof.	0	1	89.85	80.30	33.33	8.67	599.33
17 turmas	0.5	0	93.18	83.33	43.00	8.67	687.33
5x5 horários	0.5	0.5	91.52	81.82	36.00	7.33	632.00
220 restr.pref.	0.5	1	90.91	81.82	37.00	10.00	601.33
22 restr.pref. (1)	1	0	93.18	81.82	42.67	11.33	681.00
Prof/turma=1.77	1	0.5	92.12	87.88	35.67	7.33	628.00
	1	1	92.88	83.33	36.67	9.67	594.67

Na Tabela 6.1 pode ser vista a melhoria dos resultados com o aumento dos pesos; diminui o número de janelas de professores e aumenta o percentual de atendimento das preferências de horários.

Nas Tabelas 6.2 e 6.3 a melhoria no atendimento das restrições de preferência com o aumento dos pesos foi mais discreta.

TABELA 6.2 – TESTES COM GABRIEL – TARDE

Gabriel *tarde*	W_{pref}	W_{jan}	% prefer.	% prefer. (1)	Qtde Janelas	Qtde Janelas (1)	Tempo (Seg)
	0	0	92.75	75.76	48.67	6.00	840.33
	0	0.5	92.31	69.70	32.00	3.33	740.00
38 prof.	0	1	93.28	75.76	34.33	3.00	692.00
17 turmas	0.5	0	94.52	75.76	49.67	3.33	758.67
5x5 horários	0.5	0.5	93.72	83.33	38.67	4.00	687.67
377 restr.pref.	0.5	1	94.16	81.82	35.67	3.00	668.67
22 restr.pref. (1)	1	0	95.05	84.85	51.33	4.33	732.00
Prof/turma=2.24	1	0.5	94.16	80.30	41.67	4.33	679.00
	1	1	93.37	77.27	35.67	4.33	648.67

TABELA 6.3 – TESTES COM GABRIEL – NOITE

Gabriel *Noite*	W_{pref}	W_{jan}	% prefer.	% prefer. (1)	Qtde Janelas	Qtde Janelas (1)	Tempo (Seg)
	0	0	88.17	75.31	25.00	4.67	574.33
	0	0.5	88.17	76.54	12.33	2.67	518.67
38 prof.	0	1	88.69	79.01	13.00	1.67	503.33
17 turmas	0.5	0	90.59	77.78	22.67	3.33	486.33
5x4 horários	0.5	0.5	90.24	87.65	15.33	2.00	478.00
386 restr.pref.	0.5	1	89.55	82.72	13.33	2.67	480.33
27 restr.pref. (1)	1	0	90.85	76.54	26.67	2.33	451.00
prof/turma=2.24	1	0.5	90.59	77.78	16.33	3.00	444.67
	1	1	89.90	83.95	16.33	3.00	446.33

TABELA 6.4 – TESTES COM MASSARO

Massaro	W_{pref}	W_{jan}	% prefer.	% prefer. (1)	Qtde Janelas	Qtde Janelas (1)	Tempo (Seg)
	0	0	85.79	66.67	11.33	2.33	182.00
	0	0.5	88.80	86.67	4.67	0.33	169.67
18 prof.	0	1	89.89	76.67	4.00	1.67	163.00
11 turmas	0.5	0	93.44	86.67	7.00	1.33	163.67
5x4 horários	0.5	0.5	92.62	93.33	4.00	0.67	159.00
122 restr.pref.	0.5	1	93.17	96.67	6.33	1.00	160.00
10 restr.pref. (1)	1	0	93.72	90.00	7.67	1.67	157.33
prof/turma=1.64	1	0.5	93.44	86.67	5.33	1.00	158.33
	1	1	92.90	83.33	6.00	2.33	158.00

Na Tabela 6.4 a melhoria volta a ser acentuada com a variação dos pesos. Nas instâncias em que a melhoria foi mais acentuada com a variação dos pesos, a relação entre o número de professores e de turmas era menor.

Todos os resultados do *AGC* foram obtidos com uma população inicial com 100 indivíduos, indivíduos que tiveram, além das posições sementes, 20% das posições restantes associadas a alguma semente. O esquema base foi tomado entre os primeiros 33% de indivíduos da população. O valor adotado para *d* foi 0.1 e o incremento de $\boldsymbol{a}$ foi de 0.005. A cada iteração foram feitas 10 seleções.

Em todas as tabelas os tempos se referem a testes feitos com microcomputador Pentium II de 266 MHz.

6.6 – CONSIDERAÇÕES FINAIS DO CAPÍTULO

O problema de alocação de aulas em horários escolares mostrado neste Capítulo tem grande importância para instituições e ensino de médio ou grande porte, como as que encontramos em áreas urbanas das cidades brasileiras.

O problema foi considerado como sendo também de formação de agrupamentos. O objetivo foi criar agrupamentos de duplas professor/turma de modo a evitar conflitos de simultaneidade de aulas tanto de professores quanto de turmas.

Para se aproximar mais da situação real das escolas, restrições de preferência de horários pelos professores e de janelas em suas grades de aulas foram também consideradas.

Houve preocupação com a diferença de intensidade com que os tipos de restrições são considerados em situações reais. Foi considerada até mesmo a precedência que eventualmente alguns professores possam ter sobre os demais na elaboração dos horários.

A representação usada nos esquemas e estruturas foi a mesma dos Capítulos anteriores, usada para problemas de agrupamento em geral e que necessita de uma heurística para associação de elementos na formação dos agrupamentos.

A heurística de associação utilizada teve base numa medida de dissimilaridade entre seqüências binárias. No entanto, ao invés de associar dois elementos de uma estrutura ou de um esquema (um deles sendo uma semente de um agrupamento), a associação foi feita entre os elementos e os agrupamentos, estes representados por seqüências binárias obtidas com a mescla das seqüências dos elementos já pertencentes aos agrupamentos.

Foram usadas três formulações para as funções de avaliação f e g. Cada formulação levando em conta um tipo de restrição: viabilidade, preferência de horários e janelas nas grades dos professores. Foram usados pesos para ponderação do uso dessas funções.

Como mutação foi feito um processo dividido em três fases. A primeira visa garantir a viabilidade da solução, evitando os conflitos de simultaneidade tanto para professores quanto para turmas. A segunda e a terceira fase visam à melhoria no atendimento das

restrições de menor intensidade, restrições de preferências dos professores por determinados horários e de redução do número de janelas nas grades dos professores.

Os testes foram feitos com instâncias reais, e foram experimentadas combinações dos valores dos pesos de cada tipo de restrição. Os resultados se mostraram satisfatórios porque em todos os testes foram encontradas soluções viáveis, e o ajuste dos pesos pode ser utilizado para a melhoria do atendimento das restrições de menor intensidade.

Os resultados com as instâncias reais mão foram comparados com aqueles obtidos manualmente pela administração das escolar porque alguns dados, como disponibilidade de professores e seus níveis de precedência, não estavam disponíveis e foram apenas inferidos para realização dos testes.

Sem dúvida o problema requer mais estudos, principalmente na análise dos efeitos dos parâmetros do algoritmo sobre os resultados. São necessários mais dados reais para teste.

Uma continuação deste trabalho prevê a construção de um site na Internet através do qual as escolas poderiam fazer uso do algoritmo e devolver informações para seu aperfeiçoamento.

Algumas figuras que ilustram essa aplicação do *AGC* podem ser encontradas no apêndice C.

CAPÍTULO 7

CONSIDERAÇÕES FINAIS E CONCLUSÕES

O objetivo deste trabalho foi apresentar as melhorias introduzidas no Algoritmo Genético Construtivo (*AGC*) e os resultados de experimentos com aplicações sobre três problemas de otimização combinatória: Coloração de Grafos, Projeto de Células de Manufatura e *Timetabling*.

7.1 – DESENVOLVIMENTO DO TRABALHO

Na parte inicial deste trabalho foi apresentada uma introdução aos Algoritmos Evolutivos - uma classe de algoritmos que se caracteriza por processos iterativos de transformação em populações de soluções de um determinado problema.

Foram apresentados os fundamentos do processo de evolução natural nos quais se baseiam os princípios dos Algoritmos Evolutivos. Especificamente, as características de uma subclasse dos Algoritmos Evolutivos conhecida como Algoritmos Genéticos foram apresentadas. Podemos destacar entre essas caraterísticas a utilização de operações de seleção, reprodução e mutação, análogas às suas respectivas contrapartidas na natureza.

Os principais aspectos da aplicação de Algoritmos Evolutivos foram abordados. Foram analisadas formas de representação de estruturas, formas de geração da população inicial, de avaliação da adaptação dos indivíduos, de seleção, de reprodução e de mutação.

Para ilustrar a variedade de possibilidades encontradas nesses aspectos foram usados três problemas de otimização: satisfabilidade (*SAT*), caixeiro viajante (*CV*) e programação não linear (*PNL*). Em cada um desses casos foram mostradas possibilidades para cada um dos aspectos da aplicação dos Algoritmos Evolutivos.

Em seguida, este trabalho fez uma descrição do *AGC*, mostrando suas principais características, como a variação do tamanho da população, a utilização de esquemas como indivíduos da população, a utilização de um parâmetro de controle de penalização dos esquemas, e a avaliação de esquemas e estruturas, utilizando as funções $f(S_i)$ e $g(S_i)$ através de duplo objetivo: diminuir o intervalo entre as funções $g(S_i)$ e $f(S_i)$ e aumentar o valor de $g(S_i)$.

Nessa etapa do trabalho, o problema de localização de Weber foi utilizado para ilustrar as características da formulação dos problemas para aplicação do *AGC*.

Foram descritas também as melhorias introduzidas no *AGC*, como a utilização de populações formadas exclusivamente por esquemas, a complementação de esquemas bem adaptados, e a utilização de heurísticas auxiliares como mutação, com a tarefa de obter melhorias locais nas estruturas.

Este trabalho apresentou em seguida três Capítulos com a descrição de experimentos feitos com o *AGC* sobre três problemas de otimização. Em cada um desses Capítulos uma descrição do problema foi apresentada, juntamente com um levantamento de outras abordagens encontradas na literatura.

Os três problemas: Coloração de Grafos, Projeto de Células de manufatura e *Timetabling* foram formulados como sendo de formação de agrupamentos, e uma forma especial de representação para problemas desse tipo foi utilizada.

Com essa forma de representação, existe a necessidade de uma heurística de associação dos elementos de uma estrutura ou de um esquema para formação dos agrupamentos. Essa heurística é dependente do problema e em cada aplicação uma heurística diferente foi utilizada.

No caso do problema de Coloração de Grafos, foram apresentadas tanto as características da aplicação do *AGC* após as melhorias quanto as características de uma primeira aplicação do *AGC*, feita quando os estudos com esse algoritmo estavam em seus estágios iniciais.

Resultados de testes em ambas as fases de aplicação do *AGC* sobre as mesmas instâncias foram mostrados. Claramente, as melhorias introduzidas no algoritmo se refletiram de forma significativa no seu desempenho.

Ainda na aplicação ao problema de Coloração de Grafos, foi descrito um procedimento de geração de colunas para determinação de um limite inferior para o número cromático de um grafo. Nesse processo o *AGC* foi utilizado tanto para gerar o conjunto inicial de colunas quanto para gerar as novas colunas.

Também no caso de aplicação do *AGC* ao Projeto de Células de Manufatura foram descritas duas fases de estudo, antes e depois das melhorias. Foram descritas as diferenças existentes entre as duas fases de estudo em várias características de aplicação do algoritmo.

Foram apresentados testes comparativos sobre instâncias colhidas na literatura e sobre instâncias especialmente geradas. Com a dificuldade de obtenção de instâncias da literatura, apenas duas delas foram utilizadas. As outras instâncias foram geradas especialmente para tentar medir uma possível influência das densidades intracelular e intercelular das instâncias sobre o desempenho do algoritmo.

Na segunda fase da aplicação do *AGC*, com as melhorias incorporadas, os resultados sobre as instâncias da literatura se igualaram aos produzidos por outras abordagens. Os testes também mostraram que as densidades das instâncias produziram pouco ou nenhum efeito sobre o desempenho do *AGC*, especialmente após as melhorias.

No problema de formação de horários escolares (*Timetabling*), foram utilizadas instâncias reais, e o problema foi abordado numa variação que é comum nas escolas públicas brasileiras de ensino básico e médio.

Para nos aproximarmos mais da situação real das escolas, restrições de preferência de horários pelos professores e de janelas em suas grades de aulas foram consideradas.

A diferença de intensidade com que os vários tipos de restrições são vistos em situações reais também foi considerada. Até mesmo a precedência que eventualmente alguns professores pudessem ter sobre os demais na elaboração dos horários foi considerada.

Foram usadas três formulações para as funções de avaliação $f(S_i)$ e $g(S_i)$. Cada formulação levando em conta um tipo de restrição: viabilidade, preferência de horários e janelas nas grades dos professores. Foram usados pesos para ponderação do uso dessas funções.

Como mutação, foi usado um procedimento dividido em três fases: uma para garantir a viabilidade da solução, evitando os conflitos de simultaneidade tanto para professores quanto para turmas; e outras duas para melhoria no atendimento das restrições de menor intensidade, restrições de preferências dos professores por determinados horários e de redução do número de janelas nas grades dos professores.

Foram feitos testes com instâncias reais, e foram experimentadas combinações dos valores dos pesos de cada tipo de restrição. Os resultados se mostraram satisfatórios porque em todos os testes foram encontradas soluções viáveis, e foi constatado que o ajuste dos pesos pode ser utilizado para a melhoria do atendimento das restrições de menor intensidade.

7.2 – CONCLUSÕES

Com o volume e a variedade dos testes feitos com diferentes aplicações, podemos concluir que o *AGC* apresentou resultados muito bons.

A uniformidade de representação obtida com a formulação dos três problemas estudados como sendo de formação de agrupamentos possibilitou a utilização e uma base comum para o algoritmo, de forma que as alterações foram feitas apenas sobre alguns aspectos do *AGC*, como as heurísticas de associação, de mutação, nas formulações das funções de avaliação.

As melhorias com a formação da população apenas por esquemas e a complementação de bons esquemas selecionados produziram melhores resultados do que abordagens anteriores com o *AGC*. De fato, em todas as aplicações apresentadas foram obtidos resultados comparáveis aos encontrados na literatura.

A dificuldade na obtenção de instâncias para testes com células de manufatura e com *Timetabling* limitaram um pouco as possibilidades de análise do desempenho do *AGC* com esses problemas.

7.3 – FUTUROS ESTUDOS

Sem dúvida alguma há a necessidade de mais estudos com os parâmetros do AGC. A realização de testes com diferentes técnicas de controle dos parâmetros deve ser feita em breve. Uma análise do comportamento desses parâmetros com base em técnicas de Inteligência Artificial pode ser tentada.

Um estudo para melhoria na codificação do *AGC* deve ser feito. O uso de técnicas de orientação para objetos pode facilitar a adaptação do algoritmo para aplicações variadas, em que apenas alguns aspectos precisem ser especializados. Talvez até mesmo a criação de uma interface gráfica para geração dessas especializações possa ser implementada.

Quanto ao problema de coloração de grafos, a melhoria da implementação deverá possibilitar experimentos com instâncias cada vez maiores.

No caso de células de manufatura, devemos obter mais instâncias da literatura para poder comparar melhor o *AGC* com outras abordagens. Devemos ainda estudar variações para considerar outras restrições que os ambientes reais possuem, e não apenas gerar agrupamentos de máquinas sem maiores considerações sobre as peculiaridades de uma instalação industrial.

No caso de alocação de horários escolares, pretendemos criar uma página na Internet para que as escolas possam utilizar o algoritmo sobre seus próprios problemas, e nos retornem informações que serão então utilizadas para melhorar cada vez mais o algoritmo.

De modo geral, deveremos procurar aplicar o *AGC* a uma variedade cada vez maior de problemas de otimização.

REFERÊNCIAS BIBLIOGRÁFICAS

Abramson, D. Constructing schools timetables using simulated annealing: sequential and parallel algorithms. **Management Science**, n. 37, v. 1, p. 98-113, 1991.

Bäck, T. Self-Adaptation in Genetic Algorithms. In: European Conference on Artificial Life, 1., Paris, 1991. **Proceedings.** Cambridge: MIT Press, 1992, p. 263-271.

Baker, J.E. Adaptive selection methods for genetic algorithms. In: International Conference on Genetic Algorithms and Their Applications, 1., Pittsburgh, 1985. **Proceedings.** Hirllsdale: Erbaum, 1985, p. 101-111.

Briggs, P.; Cooper, K.; Kennedy, K.; Torczon, L. Coloring heuristics for register allocation. In: ASCM Conference on Program Language Design and Implementation, Portland, 1989. **Proceedings.** Portland: SIGPLAN Notices, 1989, v. 24, n.7, p. 275-284.

Burbidge, J.L. **The introduction of group technology.** London: Willian Weinemann, 1975. 267p.

Burbidge, J.L. Production Flow Analysis. **The Production Engineer**, v. 50, n. 4-5, p. 139-152, 1971.

Chams, M.; Hertz A.and Werra, D. Some experiments with simulated annealing for coloring graphs. **European Journal of Operations Research**, n. 32, p. 260-266, 1987.

Chan H.M.; Milner D.A. Direct clustering algorithm for group formation in cellular manufacture. **Journal of Management Science**, n. 1, v. 1, p.65-74, 1982.

Chandrasekharan M.P.; Rajagopalan R. Groupability: Analysis of the properties of binary data matrices for group technology. **International Journal of Production Research**, n. 27, v. 6, p. 1035-1052, 1989.

Chandrasekharan M.P.; Rajagopalan R. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. **International Journal of Production Research**, n. 24, v. 2, p. 451-464, 1986.

Chu C.H.; Tsai M. A comparison of three array-based clustering techniques for manufacturing cell formation. **International Journal of Production Research**, n. 28, v. 8, p. 1417-1433, 1990.

Coloni A.; Dorigo, M.; Maniezzo, V. Metaheuristics for high school timetabling. **Computational Optimization and Applications**, n.9, p. 275-298, 1998.

Cooper, L. Location-allocation problems. **Operations Research**, n.11, p. 331-343, 1963.

Costa, D. A tabu search algorithm for computing an operational timetable. **European Journal of Operational Research**, n. 76, p. 98-110, 1994.

Costa, D.; Hertz, A.; Bubuis, O. Embedding of a Sequential Procedure within an Evolutionary Algorithm for Coloring Problems in Graphs. **Journal of Heuristics**, v. 1, n. 1, p. 105-128, 1995.

El-Essay I.; Torrance J. Component flow analysis-an effective approach to productions systems design. **Production Engineer**, n. 51, v. 5, p. 165-170, 1972.

Even, S.; Itai, A.; Shamir, A. On the complexity of timetabling and multicommodity flow problems. **SIAM Journal of Computation**, n. 5, p. 691-703, 1976.

Farley, A A note on bounding a class of linear programming problems, including cutting stock problems. **Operations Research**, n. 38, v. 5, p. 922-923, 1990.

Feo, T.; Resende, M. G. C. Greedy Randomized Adaptive Search Procedures. **Journal of Global Optimization**, v. 2, p. 1-27, 1995.

Fleurent, C.; Ferland, J. A. **Genetic and hybrid algorithm for graph coloring**. Montréal : Université de Montréal, 1994. 30 p.

Fogel, D.B. **System identification through simulated evolution:** a machine learning approach to modeling. Needham Heights: Ginn Press , 1991.

Fogel, D. B. **Evolutionary computation:** the fossil record. Piscataway: IEEE Press, 1998.

Fogel, L.J.; Owens, A.J.; Walsh, M.J. **Artificial intelligence through simulated evolution**. New York: Wiley, 1966.

Furtado J. C. **Algoritmo genético construtivo na otimização de problemas combinatoriais de agrupamentos**. São José dos Campos. Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, 1998.

Gamst, A. Some lower bounds for a class of frequency assignment problems. **IEEE Transactions of Vehicular Technology**, v. 1, n. 35, p. 8-14, 1986.

Garey, M. R.; Johnson, D. S. **Computers and intractability:** a guide to the theory of NP-completeness. San Francisco: W. H. Freemann, 1979. 338p.

Glover F.; Laguna M. **Tabu search**. New York: Kluwer Academic, 1997. 408p.

Goldberg, D. E. **Genetic algorithms in search, optimization and machine learning**. Reading: Addison Wesley, 1989. 412p.

Goldberg, D.E. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. **Complex Systems**, v. 4, p. 445-460, 1990.

Goldberg, D.E.; Deb, K. A comparitive analysis of selection schemes used in genetic algorithms. In: G. Rawlins, ed., **Foundations of genetic algorithms**. Silicon Valley: Morgan Kaufmann, 1991.

Gotlieb, C. C. The construction of class-teacher timetables. In: IFIP Congress, Munich, 1962. **Proceedings**. North-Holland: Information Processing, 1963, p. 73-77.

Grefenstette, J.J. Optimization of control parameters for genetic algorithms. **IEEE Transactions on Systems, Man and Cybernetics,** v. 16, n. 1, p. 122-128, 1986.

Hale, W. K. Frequency assignment: theory and applications. **Proceedings of the IEEE**, v. 12, n. 68, p. 1497-1514, 1980.

Hansen, P; Mladenovic N.; Taillard E. Heuristic solution of the multisource Weber problem as a *p*-median problem. **Operations Research Letters**, n. 22, p. 55-62, 1998.

Hertz A.; de Werra, D. Using tabu search techniques for graph coloring. **Computing**, n. 39, p. 345-351, 1987.

Holland, J.H. **Adaptation in natural and artificial systems**. Michigan: MIT Press, 1975.

ILOG. **ILOG CPLEX 6.5:** getting start manual. Mountain View, California: 1999.

Johnson, D. S.; Aragon, C. R. ; McGeoch, L. A.; Schevon, C. Optimization by simulated annealing: an experimental evaluation - part II (graph coloring and number partitioning). **Operations Research**, v. 3, n. 39, p. 378-406, 1991.

Joines J.A. **Manufacturing cell design using genetic algorithms**. Raleigh, North Carolina. Master Thesis - North Carolina State University, 1993.

Julstrom, B.A. Adapting operator probabilities in a steady-state genetic algorithm. In: International Conference on Genetic Algorithms, 6., Pittsburgh , 1995. **Proceedings**. San Mateo, California: Morgan Kaufmann, 1995, p. 81-87.

King J.R. Machine-component grouping formation in group technology. **International Journal of Management Science**, n. 8, v. 2, p. 193-199, 1980a.

King J.R. Machine-component grouping in production flow analysis: an approach using rank order clustering algorithm. **International Journal of Production Research**, n. 18, v. 2, p. 213-232, 1980b.

King J.R.; Nakornchai V. Machine-component group formation in group technology: review and extension. **International Journal of Produtction Research**, n. 20, v. 2, p. 117-133, 1982.

Koza, J. R. **Genetic programming:** on the programming of computers by means of natural selection. Cambridge: MIT Press, 1992. 819p.

Kumar K.R.; Vannelli A. Strategic subcontracting for eficient disaggregated manufacturing. **International Journal of Production Research**, n. 25, p. 1715-1728, 1983.

Laguna, M.; Martí, R. **A GRASP for coloring sparse graphs**. Boulder: University of Colorado, 1999. 15p.

Leighton, F. T. A graph coloring algorithm for large scheduling problems. **Journal of Research of the National Bureau of Standards**, n. 84, p. 489-506, 1979.

Logendran R. A model for duplicating bottleneck machines in the presence of budgetary limitations in cellular manufacturing. **International Journal of Production Research**, n. 30, v. 3, p. 683-694, 1992.

Logendran R. A workload based model for minimizing total intercell and intracell moves in cellular manufacturing. **International Journal of Production Research**, n. 28, p. 913-925, 1990.

Lorena, L.A.N.; Lopes, F.B. A dynamic list heuristic for 2D-cutting. In: J. Dolezal; J. Fidler eds. **System modelling and optimization**. London: Chapman-Hall, p. 481-488, 1996.

Malave C. O.; Ramachandran S. A neural network based design of cellular manufacturing system. **Journal of Intelligent Manufacturing**, n. 2, p. 305-314, 1991.

McAuley J. Machine grouping for efficient production. **Production Engineer**, v. 2, n. 51, p. 53-57, 1972.

McCormick W.T. Jr.; Schweitzer P.J.; White T.W. Problem decomposition and data reorganization by a cluster technique. **Operations Research**, v. 5, n. 20, p. 993-1009, 1972.

Mehrotra, A.; Trick M. A. **A column generation approach to graph coloring**. Miami: University of Miami, 1993. 14p.

Michalewicz Z. **Genetic algorithms + data structures = evolution programs**. Berlin: Springer Verlag, 1996. 387p.

Michalewicz Z.; Fogel D. B. **How to solve it :** modern heuristics. Berlin: Springer Verlag, 2000. 467p.

Miltenburg J.; Zhang W. A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology. **Journal of Operations Management**, n. 10, p. 44-72, 1991.

Mitrofanov S. P. **Scientific principles of group technology**. London: National Lending Library, 1966.

Morris J.S.; Tersine R.J. A simulation analysis of factors in influencing the attractiveness of group technology cellular layouts. **Management Science**, n. 36, p. 1567-1578, 1990.

Moscato, P. **On evolution, search, optimization, genetic algorithms and martial arts:** towards memetic algorithms. Pasadena: California Institute of Technology, 1989. 68p.

Narciso, M. G. ; Lorena, L.A.N. and Furtado, J. C. Mutação de localização-alocação para problemas de p-medianas. [CD-ROM]. In: Simpósio Brasileiro de Pesquisa Operacional, 33., Viçosa, 2000. **Anais**. Rio de Janeiro: SBPO, 2000.

Neufeld, G. A.; Tartar, J. Graph coloring conditions for existence of the solution to the timetabling problem. **Communications of the ACM**, n. 17, v. 8 , p. 450-453, 1974.

Rajagopalan R.; Batra J.L. Design of cellular production systems: a graph theoretic approach. **International Journal of Production Research**, n. 13, v. 6, p. 567-579, 1975.

Rechenberg, I. **Cybernetic solution path of an experimental problem.** Farnborough Hants, UK: Royal Aircraft Establishment, 1965.

Ribeiro Filho, G. **Uma heurística construtiva para coloração de grafos**. São José dos Campos. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, 1997.

Ribeiro Filho, G.; Lorena, L. A. N. Algoritmo genético construtivo aplicado ao projeto de células de manufatura. In: Oficina de cortes e empacotamento, 3., Curitiba, 1998. **Anais**. p. 131-141, 1998.

Sarker B.R.; Mondal S. Grouping eficiency in cellular manufacturing: a survey and critical review. **International Journal of Production Research**, n. 37, v. 2, p. 285-314, 1999.

Schaerf, A. Tabu search techniques for large high school timetabling problems. In: National Conference on Artificial Intelligence, 13, Portland, 1996. **Proceedings**. Portland: MIT Press, 1996. v. 1, p. 363-368.

Schaerf, A. A survey of automated timetabling. **Artificial Intelligence Review**, n.13, p. 87-127, 1999a.

Schaerf, A. Local search techniques for large high school timetabling problems. **IEEE Transactions on Systems Man and Cybernetics Part A – Systems and Humans**, n. 29, p. 368-377, 1999b.

Schmidt, G.; Strohlein, T. Timetable construction: an annotated bibliography. **The Computer Journal**, v. 23, n. 7, p. 307-316, 1979.

Seifoddini H. Duplication process in machine cells formation in group technology. **IIE Transactions**, v. 21, p. 382-388, 1989.

Seifoddini H.; Wolfe P.M. Application of the similarity coeficient method in group technology. **IIE Transactions**, v. 18, p. 271-277, 1986.

Senne, E.L.F.; Lorena, L.A.N. Lagrangean/Surrogate Heuristics for p-Median Problems. In: Laguna M.; Gonzalez-Velarde J.L. eds. **Computing tools for modeling, optimization and simulation:** interfaces in computer science and operations research. Boston: Kluwer Academic Publishers, 2000, p. 115-130.

Smith, J.; Fogarty, T.C. Self adaptation of mutation rates in a steady state genetic algorithm. In: IEEE International Conference on Evolutionary Computation, Nagoya, 1996. **Proceedings**. New York: IEEE Press, 1996, p. 318-323.

Sofianopoulou S. Manufacturing cells design with alternative process plans and/or replicate machines. **International Journal of Production Research**, v. 37, n. 3, p. 707-720, 1999.

Souza, M.J.F.; Maculan, N.; Ochi, L.S. GTS-II: uma heurística para o problema de horário de escolas. In: Congresso Latino-Iberoamericano de Pesquisa Operacional, 10., Mexico, 2000. **Proceedings**. Mexico: Instituto Mexicano de Sistemas e Investigación de Operaciones, 2000.

Stanfel L. E. Machine clustering for economic production. **Engineering Costs and Production Economics**, n. 9, p.73-81, 1985.

Stecke, K. Design, planning, scheduling and control problems of flexible manufacturing. **Annals of Operations Research**, v. 3, p. 3-12, 1985.

Taillard, E.D. **Heuristic methods for large centroid clustering problems**. Lugano: IDSIA, 1996. (IDSIA96-1996).

Vakharia A.J.; Chang Y.L. A simulated annealing approach to scheduling a manufacturing cell. **Naval Research Logistics Quarterly**, n. 37, p. 559-577, 1990.

Venugopal V.; Narendran T. T. Cell formation in manufacturing systems through simulated annealing: an experimental evaluation. **European Journal of Operational Research**, n. 63, v. 3, p. 409-422, 1992a.

Venugopal V.; Narendran T.T. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. **Computers and Industrial Engineering**, n. 22, p. 469-480, 1992b.

Venugopal V.; Narendran T. T. Design of cellular manufacturing systems based on asymptotic forms of a boolean matrix. **European Journal of Operational Research**, n. 67, p. 405-417, 1993.

Wei J.C.; Gaither N. A capacity constrained multiobjective cell formation method. **Journal of Manufacturing Systems**, n. 9, p. 222-232, 1990.

Wemmerlov U.; Hyer N.L. Cellular manufacturing in the u.s. industry: a survey of users. **International Journal of Production Research**, n. 27, p. 1511-1530, 1989a.

Wemmerlov U.; Hyer N.L. Group technology in the u.s. manufacturing industry: a survey of current practices. **International Journal of Production Research**, n. 27, p. 1287-1304, 1989b.

Wesolowsky, G. The Weber problem: history and perspectives. **Location Science**, n. 1, p. 5-23, 1993.

Xu H.; Wang H. P. Part family formation for gt applications based on fuzzy mathematics. **International Journal of Produtction Research**, n. 27, v. 9, p. 1637-1651, 1989.

APÊNDICE A

A seguir são mostradas figuras que ilustram alguns aspectos do *AGC*. Uma figura mostra uma representação do duplo mapeamento das estruturas S_i da população no R^+ feito pela funções $f(S_i)$ e $g(S_i)$. São mostradas figuras com a variação do tamanho da população ao longo das gerações do processo evolutivo. São também mostradas figuras que ilustram a variação da diferença $\{g(S_i)- f(S_i)\}$.

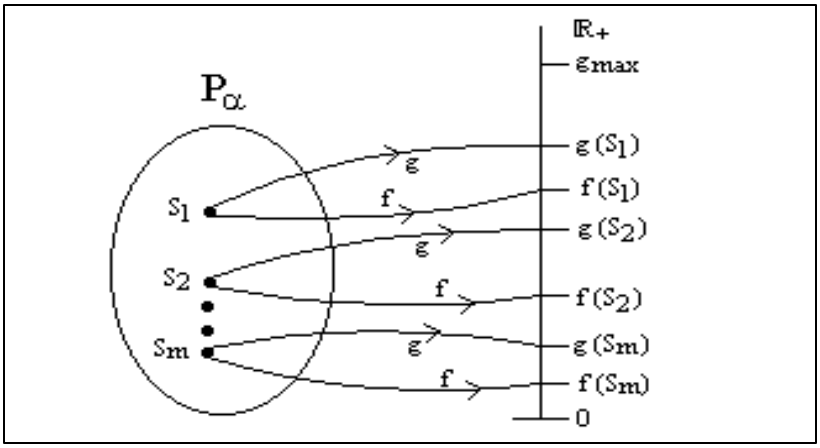

Fig. A1 – Mapeamento da população no R^+.

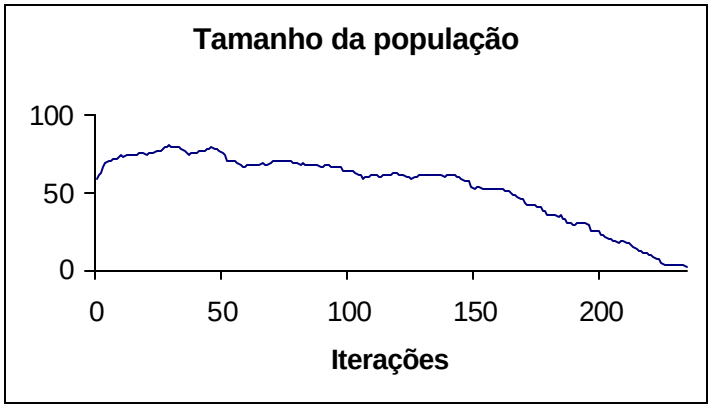

Fig. A2 – Variação do tamanho da população.

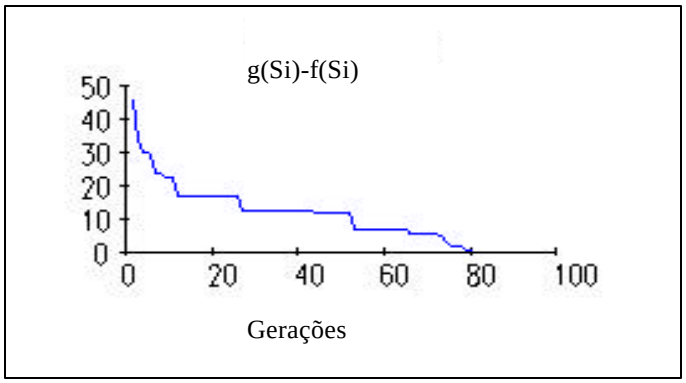

Fig. A3 – Variação da diferença *{g(Si)-f(Si)}*.

APÊNDICE B

A seguir são mostradas figuras que ilustram a composição das matrizes partes/máquinas encontradas na literatura, antes e depois da aplicação do *AGC*.

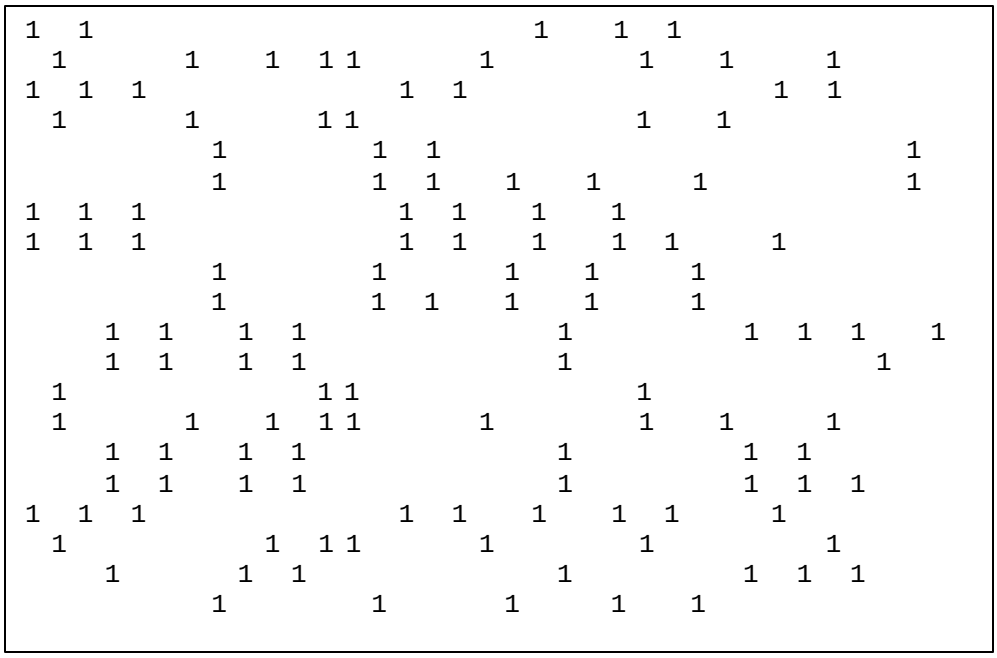

Fig. B1 – Matriz Burbidge antes do processamento.

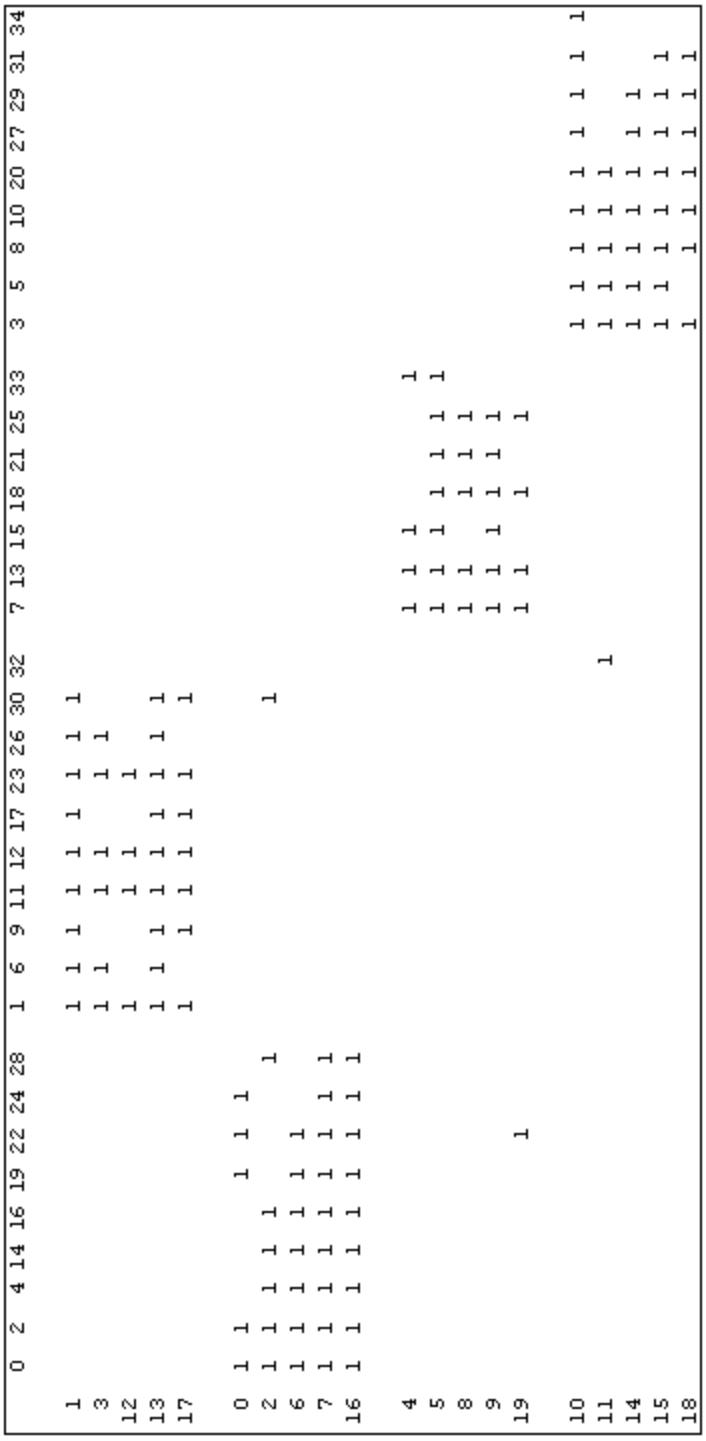

Fig. B2 – Matriz Burbidge após o processamento.

Fig. B3 – Matriz Chandra após o processamento.

APÊNDICE C

A seguir são mostradas figuras que ilustram parte da saída da aplicação do *AGC* sobre uma escola estadual de primeiro grau.

```
Horario das turmas...

5I
|----------|----------|----------|----------|----------|
| dia  1   | dia  2   | dia  3   | dia  4   | dia  5   |
|----------|----------|----------|----------|----------|
|      geo |      ing |    edart |     hist |      geo |
| MarleneG |   Simone |    Niria |  Vitoria | MarleneG |
|----------|----------|----------|----------|----------|
|    edart |     hist |      mat |      ing |      mat |
|    Niria |  Vitoria | Marilena |   Simone | Marilena |
|----------|----------|----------|----------|----------|
|    cienc |     port |      mat |     port |      mat |
| MarleneC |  Socorro | Marilena |  Socorro | Marilena |
|----------|----------|----------|----------|----------|
|    cienc |      mat |     port |     port |     port |
| MarleneC | Marilena |  Socorro |  Socorro |  Socorro |
|----------|----------|----------|----------|----------|

6D
|----------|----------|----------|----------|----------|
| dia  1   | dia  2   | dia  3   | dia  4   | dia  5   |
|----------|----------|----------|----------|----------|
|     hist |    cienc |      geo |    edart |      ing |
|  Vitoria |   Marcia | MarleneG |    Vania |   Simone |
|----------|----------|----------|----------|----------|
|      mat |      mat |     port |      mat |      mat |
|  Edvaldo |  Edvaldo |  Antonio |  Edvaldo |  Edvaldo |
|----------|----------|----------|----------|----------|
|     hist |    cienc |    edart |      mat |     port |
|  Vitoria |   Marcia |    Vania |  Edvaldo |  Antonio |
|----------|----------|----------|----------|----------|
|      geo |     port |      ing |     port |     port |
| MarleneG |  Antonio |   Simone |  Antonio |  Antonio |
|----------|----------|----------|----------|----------|
```

Fig. C1 – Horário de turmas.

```
Horario de professores...


Diolinda
|----------|----------|----------|----------|----------|
| dia  1   | dia  2   | dia  3   | dia  4   | dia  5   |
|----------|----------|----------|----------|----------|
|     hist |     hist |     hist |          |          |
|       6E |       8D |       8D |          |          |
|----------|----------|----------|----------|----------|
|     hist |     hist |     hist |          |          |
|       8E |       8E |       7C |          |          |
|----------|----------|----------|----------|----------|
|     hist |     hist |     hist |          |          |
|       7C |       8C |       7D |          |          |
|----------|----------|----------|----------|----------|
|     hist |     hist |     hist |          |          |
|       7D |       6E |       8C |          |          |
|----------|----------|----------|----------|----------|

MarleneG
|----------|----------|----------|----------|----------|
| dia  1   | dia  2   | dia  3   | dia  4   | dia  5   |
|----------|----------|----------|----------|----------|
|      geo |      geo |      geo |      geo |      geo |
|       5I |       5J |       6D |       5L |       5I |
|----------|----------|----------|----------|----------|
|      geo |      geo |      geo |      geo |      geo |
|       5L |       7D |       7D |       6E |       5K |
|----------|----------|----------|----------|----------|
|      geo |      geo |      geo |      geo |      geo |
|       8C |       8D |       6E |       5K |       8C |
|----------|----------|----------|----------|----------|
|      geo |      geo |      geo |      geo |      geo |
|       6D |       5J |       8D |       7C |       7C |
|----------|----------|----------|----------|----------|

Marilena
|----------|----------|----------|----------|----------|
| dia  1   | dia  2   | dia  3   | dia  4   | dia  5   |
|----------|----------|----------|----------|----------|
|          |      mat |      mat |      mat |      mat |
|          |       5K |       5K |       5K |       5K |
|----------|----------|----------|----------|----------|
|          |      mat |      mat |      mat |      mat |
|          |       5K |       5I |       5J |       5I |
|----------|----------|----------|----------|----------|
|          |      mat |      mat |      mat |      mat |
|          |       5J |       5I |       5J |       5I |
|----------|----------|----------|----------|----------|
|          |      mat |      mat |      mat |          |
|          |       5I |       5J |       5J |          |
|----------|----------|----------|----------|----------|
```

Fig. C2 – Horário de professores.

```
Funcoes de avaliacao...

g1=1100.000000  f1=1100.000000  g1-f1=0.000000  d1=0.000000
g2=220.000000  f2=212.000000  g2-f2=8.000000  d2=0.036364
g3=220.000000  f3=220.000000  g3-f3=0.000000  d3=0.000000
g4=220.000000  f4=208.000000  g4-f4=12.000000  d4=0.054545

Tempo total: 14

Total de restricoes: 122
Total de conflitos: 8
Restricoes atendidas: 93.442623

Total de restricoes nivel 1: 10
Total de conflitos nivel 1: 3
Restricoes atendidas nivel 1: 70.000000

Total de janelas: 12
Total de janelas nivel 1: 2
```

Fig. C3 – Parâmetros dos resultados.