Aprendizagem cooperativa e o problema de formação de grupos

Mark R. C. Lima [1], Sofiane Labidi[1], Othon C. Bastos Filho[2], Luis Carlos C. Fonseca[2]

[1] Universidade Federal do do Maranhão, Laboratório de Sistemas Inteligentes, LSI/UFMA, Campus do Bacanga, 65.000-000 - São Luís-MA, Brasil.

2 Universidade Federal Universidade Federal do Rio Grande do Sul (UFRGS)
Programa de Pós-Graduação em Informática na Educação - PPGIE
Av. Paulo Gama, 110 - prédio 12105 - 3º andar sala 332
90040-060 - Porto Alegre (RS) - Brasil

mark.renato@gmail.com, labidi@uol.com.br, othonb@hotmail.com, lccf_@uol.com.br.

Resumo. Este trabalho apresenta o problema da divisão de uma classe de alunos em grupos para aprendizagem cooperativa sujeita a fatores pedagógicos. Propomos uma abordagem baseada em algoritmos genéticos para a formação dos grupos no qual aplicamos os fatores de aceitação de um grupo, pelo professor, baseada no perfil dos alunos e pela coesão do grupo, baseada no uso de técnicas sócio-métricas. Esta abordagem é usada no ambiente NetClass de ensino-aprendizagem cooperativa.

Palavras chaves: Aprendizagem cooperativa, formação de grupos, algoritmos genéticos, sócio-gramas e netclass.

Abstract. This paper addresses the problem of group making in cooperative learning. This should be done based on pedagogical and dynamic group factors and not in a random way. We propose a genetic algorithm approach to tackle a special instance of this problem when the factors are the teacher acceptance of a group based on learner profiles and the group cohesion based on sociogram techniques. The approach is used by the NetClass Computer Supported Cooperative Learning Environment.

Keywords: Cooperative learning, genetic formation of groups, algorithms, partner-gram and netclass.

1. Introdução

A cooperação entre aprendizes em ambientes computadorizados de ensino-aprendizagem, como o *NetClass* (Labidi et al. 2003) (cf. figura 1), é um dos pontos chave para o sucesso da aplicação desses ambientes. Logo, muitos desses ambientes buscam promover a cooperação para um melhor aprendizado por parte dos aprendizes. Para tanto, uma das formas adotadas é a divisão das classes em grupos, onde os aprendizes de cada grupo interagem diretamente uns com os outros nas sessões de ensino-aprendizagem, pois é através da troca de informações, perguntas e comentários que este processo é complementado e, conseqüentemente, enriquecido (Wolz and Palm et al. 1997).

A distribuição dos aprendizes em grupos pode afetar diretamente nos seus desempenhos. Essa relação, distribuição x desempenho, pode se dá, entre outros fatores, em função do perfil de cada aprendiz e/ou dos laços de afinidade existentes entre eles.

Sendo assim, a tarefa de dividir a classe em grupos no *NetClass,* e em qualquer outro sistema de ensino-aprendizagem cooperativa, torna-se de extrema importância.

Do ponto de vista tecnológico, a cooperação entre aprendizes, em um ambiente computacional de ensino-aprendizagem, depende de duas tarefas: a definição dos grupos e o estabelecimento de interação entre os grupos e o ambiente, o qual se compõe de outros grupos, dos professores e dos componentes computacionais. Portanto, um ambiente de cooperação deve fornecer ferramentas para executar ambas as tarefas.

Neste trabalho apresentamos uma ferramenta que executa a tarefa de definição dos grupos. Baseando-se em fatores pedagógicos, definimos esta tarefa como um problema de otimização ao qual aplicamos um algoritmo genético para solucioná-lo.

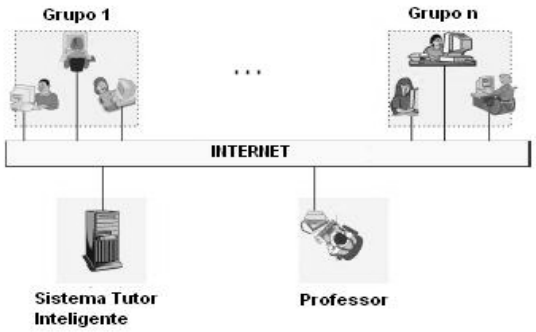

Figura 1. Ambiente netclass

O NetClass está sendo desenvolvido para suportar o processo de aprendizagem cooperativa através de um ambiente no qual os aprendizes serão distribuídos em grupos, orientados por um professor humano com a ajuda de um Sistema Tutor Inteligente.

2. Estrutura Básica dos AGs

Os algoritmos genéticos (AGs) são algoritmos de busca, baseados na teoria da evolução das espécies de Charles Darwin onde os indivíduos de uma determinada população evoluem de acordo com os princípios de seleção natural e sobrevivência dos mais aptos. Desenvolvidos por John Holland, juntamente com seus colegas e estudantes da Universidade de Michigan, estes algoritmos simulam os processos de evolução da natureza, onde cada indivíduo representa uma possível solução para um dado problema (Goldberg 1989). A idéia básica dos AGs consiste, em analogia direta às idéias darwinianas, na sobrevivência dos melhores indivíduos, ou seja, das soluções do problema dado (Viana 1998). Os indivíduos com maior capacidade de adaptação ao seu meio ambiente terão maiores possibilidades de sobreviverem e reproduzirem-se do que os indivíduos menos adaptados. Depois de muitas gerações, os indivíduos da população adquirem características que lhes conferem uma maior adaptabilidade ao ambiente que indivíduos das gerações anteriores. Quando se faz notória a diferença entre as populações, de forma que se possa mensurar tal diferença, diz-se que a população evoluiu (Koza 1992).

A execução de um algoritmo genético começa com a geração de uma população inicial formada por indivíduos selecionados aleatoriamente do espaço de busca. Durante o processo evolutivo, cada indivíduo da população é avaliado para determinar o seu valor de aptidão (*fitness*). Com base nestes valores, a população é transformada em uma nova população através da aplicação de operadores de seleção, cruzamento e mutação. Este processo é repetido até que um critério de parada seja atingido.

Para utilizarmos algoritmos genéticos na busca de soluções ótimas de um determinado problema, devemos considerar os seguintes aspectos: (1) temos que reconhecer o problema em questão como sendo um problema de otimização. Isto envolve a definição de um método para avaliar quão perto do ótimo está uma possível solução. (2) devemos poder codificar as soluções como uma *string* de modo a termos um modelo de cromossomos. Um terceiro ponto a se destacar é quanto à parametrização genética, onde se determinam taxas de cruzamento e mutação, tamanho das populações e os critérios de parada.

Na seqüência, mostraremos a solução proposta para formação de grupos no ensino cooperativo, aplicando os conceitos aqui apresentados.

3. O Problema da Formação de Grupos

O problema em questão é a formação de grupos para aprendizagem cooperativa. Declarando o problema formalmente temos: seja C uma classe de n x k (n: número de grupos, e k: tamanho dos grupos) aprendizes e $F_1, F_2, ..., F_m$ uma lista de m fatores pedagógicos. Suponha que o professor deseja uma "boa divisão" da classe C em n grupos de k aprendizes sujeitos aos fatores $F_1, F_2, ..., F_m$. Definimos um *grupo-k* em C como um subconjunto $\{l_1, l_2, ..., l_k\}$ de C com k elementos. Definimos uma *divisão-k x n* de C como um subconjunto $\{g_1, g_2, ..., g_n\}$ com n elementos, onde cada elemento é um *grupo-k* diferente em C. Definimos o *fitness* de uma *divisão-k x n D* de C levando-se em conta os fatores $F_1, F_2, ..., F_m$ como uma função definida em termos de $F_1, F_2, ..., F_m$ que mapeia D em um número φ que mede o *fitness*. Em símbolos:

$$\varphi = FITNESS(F_1, F_2, ..., F_m)(D).$$

Então definimos o problema da formação de grupos cooperativos como sendo encontrar uma *divisão-k x n D* de C que maximize a função $FITNESS(F_1, F_2, ..., F_m)(D)$. No caso específico do *NetClass* teremos dois fatores pedagógicos: a visão do professor sobre quão bom é um grupo, baseada no perfil dos aprendizes (F_1) e as preferências de agrupamento dos aprendizes, inferidas através da aplicação de teste sociométrico (F_2).

3.1. Perfis

O que chamamos de perfil do aprendiz é uma abstração das atitudes/características sociais e psicológicas de um determinado aprendiz. Definimos o perfil do aprendiz como uma *p-tupla* $\langle at_1^l, at_2^l, ..., at_p^l \rangle$, onde cada $at_{1 \le i \le p}^l$ é o valor de um atributo do aprendiz.

Inicialmente, adotamos um perfil baseado em três atributos: (1) A capacidade de cooperação do aprendiz (at^l_1), (2) A capacidade de liderança (at^l_2), e (3) O grau de conhecimento do aprendiz sobre determinado tópico (at^l_3). Todos atributos variam em uma escala que vai de muito baixo para muito alto, passando por baixo, médio e alto. Logo, todo estudante está associado a uma tripla tal como $l \xrightarrow{P} \langle medio, baixo, alto \rangle$. Estas informações são armazenadas e mantidas no modelo do aprendiz pelo agente de modelagem do aprendiz, parte do ambiente *NetClass*.

Estas definições são motivadas pelo foco sobre os aspectos pedagógicos e de dinâmica de grupos fundamentais na modelagem do aprendiz na perspectiva de um trabalho cooperativo. Maiores detalhes sobre a modelagem do aprendiz no projeto *NetClass* podem ser encontradas em (Labidi et al. 2003). Todavia, vale ressaltar que a técnica aqui proposta é genérica e pode, caso necessário, integrar outros fatores.

3.2. Perfis Aceitáveis para o Professor

A divisão de uma classe *C* em *grupos-k* é determinada, em parte, por critérios definidos pelo próprio professor. Esses critérios são embasados nos perfis dos aprendizes. O professor define um "grupo ideal" e um "grupo aceitável" levando-se em conta os perfis dos alunos. Com isso, para serem aceitáveis, os grupos de uma classe deverão estar enquadrados em um dos modelos de grupos definidos pelo professor. Para determinar a aceitação dos grupos definimos a função:

$$p[g]_A^I = \begin{cases} 3 \text{ if } \{P(a_1), P(a_2), ..., P(a_3)\} \in I - A \\ 2 \text{ if } \{P(a_1), P(a_2), ..., P(a_3)\} \in I \cap A \\ 1 \text{ if } \{P(a_1), P(a_2), ..., P(a_3)\} \in A - I \\ 0 \text{ caso contrário} \end{cases}$$

onde $P(a_n)$ é o perfil do aprendiz *n*, *I* é o conjunto de perfis dos aprendizes do grupo ideal e *A* o conjunto de perfis dos aprendizes do grupo aceitável. Esta função mapeia *grupos-k* em categorias de aceitação. A categoria 3 representa os grupos mais aceitáveis. A categoria 0 não é um grupo aceitável.

3.3. Coesão Sociométrica

Desenvolvido por Jacob Levy Moreno em seus trabalhos sobre sociometria (Moreno, 1954), o teste sociométrico é uma técnica que permite determinar o grau no qual os indivíduos são aceitos ou rejeitados em um grupo, descobrir as relações entre os indivíduos e revelar a estrutura do grupo. Consiste, basicamente, na aplicação de um questionário aos aprendizes e sua posterior análise. Devemos destacar alguns pontos primordiais na elaboração dos questionários que são: os tipos de perguntas (escolha, rejeição e percepção) e o número de escolhas/rejeições a serem feitos. No *NetClass* o aprendiz é solicitado a informar, em ordem de preferência, três outros aprendizes com os quais desejaria agrupar-se.

A partir das respostas dadas pelos aprendizes, podem ser feitas várias aplicações sobre esses resultados como detecção de problemas de adaptação social (existência de membros isolados, membros rejeitados pela maioria, grupos fechados, etc) e determinação de estratégias para formar os grupos.

Com base nessas teorias estabelecemos as seguintes regras para a formação de grupos:

- Regra 1. Nunca formar grupos quando não há nenhuma relação (escolha) entre seus membros *(R_1)*;
- Regra 2. Para um aluno isolado, sempre colocá-lo em um grupo em que sua primeira escolha se satisfaz *(R_2)*;
- Regra 3. Sempre formar os grupos satisfazendo pelo menos alguma escolha de todos os componentes *(R_3)*;
- Regra 4. Quando possível formar grupos que preservem as escolhas mútuas *(R_4)*.

Usando as regras *R*, definimos a coesão sociométrica de um *grupo-k* com sendo a função:

$$s[g]_R = \begin{cases} 0 \text{ if } \neg R_1(g) \vee \neg R_2(g) \\ 1 \text{ if } R_1(g) \wedge R_2(g) \wedge \neg R_3(g) \\ 2 \text{ if } R_1(g) \wedge R_2(g) \wedge R_3(g) \neg R_4(g) \\ 3 \text{ caso contrário} \end{cases}$$

Esta função mapeia *grupos-k g* em graus de coesão. O Grau 0 indica nenhuma coesão. A categoria 3 indica uma alta coesão.

3.4. O Problema

Juntando-se as peças, definimos quão boa é uma divisão *D* da classe levando-se em conta a definição dos perfis aceitáveis pelo professor $p[g]_A^I$ e pela coesão sociométrica $s[g]_R$ como sendo a função:

$$FITNESS(p[\cdot]_A^I, s[\cdot]_R)(D) = \sum_{g \in D} F(p[g]_A^I, s[g]_R)$$

onde *F(p,s)* mapeia os graus de aceitação e coesão sociométrica em números que refletem a qualidade de um grupo. Portanto, o nosso problema é de maximizar a função *FITNESS* aqui definida.

4. Aplicação dos AGs para Resolução do Problema

Tratando o problema como sendo de otimização, implementamos um algoritmo genético capaz de encontrar soluções ótimas. Em outras palavras, encontrar a melhor divisão da classe seguindo os critérios já apresentados. Nas seguintes subseções detalharemos a modelagem do algoritmo proposto, mostrando a codificação dos indivíduos, sua função de avaliação e todos os outros parâmetros genéticos necessários para a execução do algoritmo.

4.1. Codificação dos Indivíduos

Um dos pontos chave para uma execução satisfatória de um AG é a codificação adequada do problema (Banzahf et al. 1998). A codificação dos cromossomos de um AG deve ser feita por meio de uma cadeia finita em algum alfabeto definido (Goldberg 1989).

Em muitos problemas, o valor de aptidão não depende apenas do valor dos alelos na cadeia de genes do cromossomo. O valor de aptidão pode depender também de alguma combinação entre o valor do alelo e a ordem dos genes. Em outros casos, o valor de aptidão pode depender somente da ordem dos genes, a estes dá-se o nome de problemas de permutação (Goldberg 1989). Obviamente nestes problemas cada alelo aparece apenas uma vez (Déb. 2000). O espaço de busca para problemas representados por permutação possui *k*! possibilidades, sendo *k* o número de genes a serem ordenados (Goldberg 1989).

Para o nosso problema, cada cromossomo representará uma classe de aprendizes, onde cada gene do cromossomo caracteriza um aprendiz dentro da classe (cf. figura 2). Desta forma, o número de genes do cromossomo é definido pelo número *k* de aprendizes da classe. Seqüencialmente, cada conjunto de *n* genes representa um grupo dentro da classe, onde *n* é o número de aprendizes que compõem um grupo.

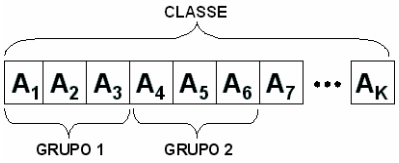

Figura 2. Codificação do cromossomo para *n* igual 3

Sendo assim, a diferença entre o código genético dos cromossomos se dará pela ordem dos genes no mesmo, o que caracteriza o nosso problema como sendo de permutação.

4.2. Função de Avaliação

A definição apropriada da função de avaliação é uma das tarefas principais na modelagem dos AGs, já que é através dela que será avaliada a qualidade de cada indivíduo na população (Knosala 2001). A qualidade de cada indivíduo é medida pelo seu valor de aptidão (*fitness)* que é a propensão de um indivíduo a sobreviver e se reproduzir em um determinado ambiente.

A função de avaliação de cada indivíduo da população será dada pelo cálculo das funções *p* e *s* anteriormente apresentadas. Onde *p* está relacionada aos perfis dos aprendizes e *s* está associada com a coesão sociométrica dos grupos. A função de avaliação será então definida por:

$$fitness = \sum_{g \in D} F(p[g]_A^I, s[g]_R) = \sum_{g \in D} (x.p[g] + y.s[g])$$

onde *x* e *y* são pesos podendo ser atribuídos pelo professor para favorecer uma ou outra abordagem.

4.3. Operador de Seleção

O objetivo do operador de seleção é direcionar o processo para as melhores regiões do espaço de busca. Para tanto esse operador busca os melhores indivíduos da população para reprodução, dando preferência para os indivíduos mais adaptados ao ambiente (Mitchell 1996). Utilizamos a seleção por torneio. Basicamente, duas soluções são escolhidas e aquela de melhor valor de aptidão é escolhida para ser processada pelo operador de cruzamento.

4.4 Operador de *Crossover*

O operador de *crossover* ou de recombinação permite a obtenção de indivíduos filhos mediante a combinação dos cromossomos dos pais.

O operador *position based crossover* é o de melhor desempenho entre os aplicados a problemas de permutação superando os operadores *order crossover #1*, *order crossover #2*, *partially mapped crossover* e *cycle crossover* (Starkweather et al. 1991).

Com base nisto, adotamos o operador *position based crossover*. Este operador visa preservar a posição das informações durante o processo de recombinação. Diversas posições são selecionadas aleatoriamente no cromossomo de um dos pais e os genes dessas posições são herdados desse pai para o filho. Os genes remanescentes são herdados pelo filho na ordem que aparecem no outro pai. O número de pontos de

crossover varia aleatoriamente entre 1 e *k-2*, onde *k* é o número de genes do cromossomo (Starkweather et al. 1991).

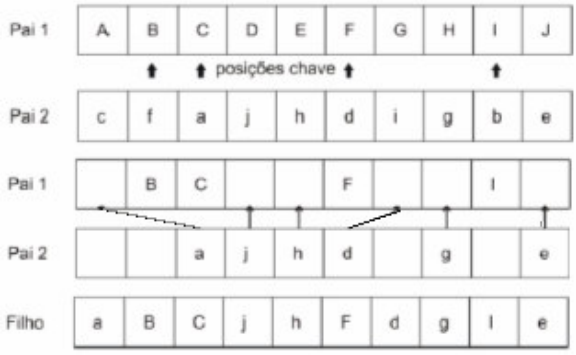

Figura 3. Exemplo de *Based Position Crossover*

Na figura 3, os genes *B, C, F* e *I* são herdados do primeiro pai nas mesmas posições. Os genes remanescentes são herdados do segundo pai na ordem em que aparecem, ou seja, os genes *a*, *j*, *h*, *d*, *g* e *e* são herdados nas posições 1, 4, 5, 7, 8 e 10, respectivamente.

4.5. Operador de Mutação

Nos AGs o operador de mutação executa um papel secundário, porém necessário, pois possibilita restaurar a diversidade genética eventualmente perdida durante o processo evolutivo (Goldberg 1989).

Ao utilizarmos o operador *position based crossover*, os operadores de mutação, tanto do tipo *swap* como *shift*, utilizados nas codificações por permutação, já estão implícitos. Ao aplicar-se a recombinação com apenas um ponto de *crossover*, obtém-se um efeito similar ao operador de mutação *swap*, e ao fazê-la com *K*-2 pontos de *crossover*, o efeito é similar ao operador de mutação *shift* (Souza 2004). Dessa forma suprimimos o uso do operador de mutação.

4.6. Critério de Parada

Utilizamos a mesma estratégia adotada por (Souza 2004) em seu trabalho. A execução do algoritmo é encerrada quando um número *G* de gerações apresentarem a mesma solução, ou seja, se o AG encontrar uma solução na *i-ésima* geração e ela permanecer como a melhor solução durante as *G* gerações seguintes, então se considera que o algoritmo convergiu e encerra-se a execução. O número *G* de gerações ideal também está sendo definido com a aplicação de testes.

4.7. O Algoritmo Genético

A figura 4 mostra o diagrama de classes (em UML) do algoritmo genético. A classe principal *Algorithm* implementa os operadores genéticos e tem como atributos principais os parâmetros genéticos para a execução do algoritmo. Na classe *GroupProblem* implementamos o método abstrato *evaluate()* da classe *Problem*, que vai calcular o *fitness* de determinado indivíduo. Os métodos *p()* e *s()* implementam as funções *p* e *s* já apresentadas e vão compor, como já mostrado anteriormente, o método *evaluate()*. Em *Population* encapsulamos o conjunto de indivíduos de uma população. Estes são implementados pela classe *Individual*. A classe *Chromosome* guarda a

informação genética dos indivíduos. Instâncias da classe *Learner* serão os alelos dos cromossomos de determinado indivíduo.

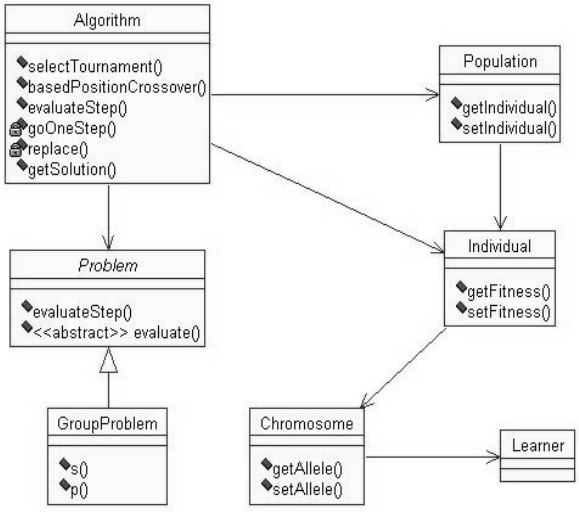

Figura 4. Diagrama de classes

As figuras 5 e 6 mostram a interface entre o professor e o algoritmo desenvolvido. Na primeira temos duas interfaces que demonstram a utilização da ferramenta utilizando a abordagem baseada nos perfis dos aprendizes. Em (a) o professor tem a lista dos aprendizes matriculados no seu módulo e tem a opção de escolher qual a abordagem a ser utilizada para formar os grupos. Escolhendo a opção baseada no perfil, a interface (b) é apresentada, mostrando os perfis dos aprendizes e possibilitando que o professor defina os grupos ideal e aceitável.

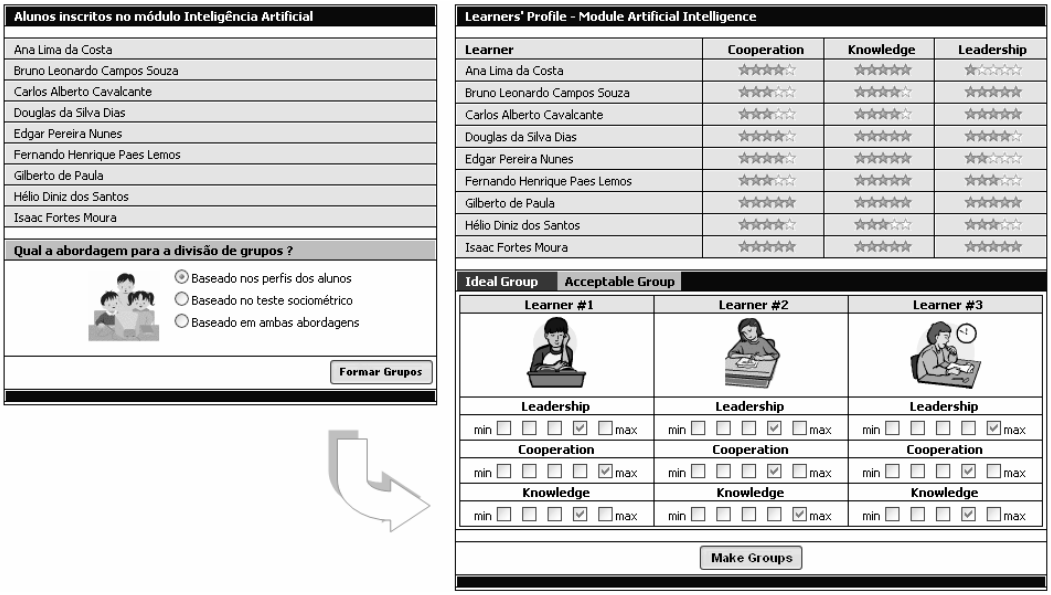

Figura 5. Formação de Grupos no *NetClass*

O professor então define os grupos ideal e aceitável em função do perfil de cada um dos aprendizes que irão compô-los. Pressionando *make groups* a interface passa os dados informados ao algoritmo genético que, ao fim de sua execução, mostra a interface

da figura 6, contendo a melhor solução encontrada para a divisão da classe em função dos grupos ideal e médio definidos pelo professor.

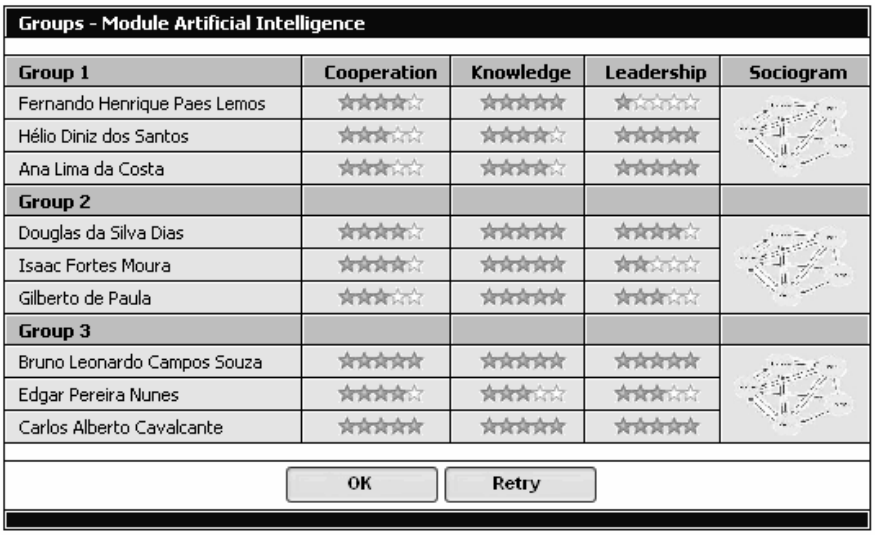

Figura 6. Grupos formados

5. Conclusão

O algoritmo genético proposto mostrou-se capaz de resolver o problema em questão. Em testes feitos em nosso laboratório com uma classe de 36 estudantes e grupos de três, as divisões propostas, quando possível, sempre respeitaram os critérios de aceitação do professor e da coesão sociométrica.

Atualmente estamos aplicando testes mais exaustivamente no intuito de melhorar a eficiência e a eficácia do algoritmo. Como produto dos testes, dados numéricos estão sendo coletados e, com base nestes dados, estamos desenvolvendo variações do algoritmo para chegarmos a sua versão ideal.

Encontramos uma série de trabalhos relacionados ao aqui apresentado, onde há uma preocupação com a formação de grupos para aprendizagem cooperativa. Apesar de utilizarem abordagens diferentes da nossa, podemos tirar muito proveito sobre aspectos referentes a formação de grupos. Em (Olguín et al. 2000) e (Cunha et al. 2003) temos a formação de grupos suportada por uma arquitetura de agentes inteligentes. Ambos também utilizam o modelo do aprendiz como base para formar os grupos. Já o Guardian Agent (Whatley et al. 2001) monitora as atividades realizadas pelos aprendizes e, a partir dos dados monitorados, atribui papéis específicos aos membros dos grupos. (Koriji et al. 2001) destacam a importância do suporte ao aprendizado em grupo em sistemas para aprendizagem cooperativa.

Utilizamos uma abordagem orientada a objetos na implementação do sistema com modelagem em UML. Todo o trabalho aqui apresentado está implementado e integrado no *NetClass* com tecnologia Java.

Referências Bibliográficas

Banzahf, W., Nordin, P., Keller, R. E. and Francone, F. D. (1998) “Genetic programming: an introduction on the automatic evolution of computer programs and its applications”. San Francisco, CA: Morgan Kaufmann.

Cunha, L. M., Fuks, H. and Lucena, C. J. P. de. (2003) "A Adaptação do Ambiente AulaNet para dar Suporte a Grupos de Aprendizagem e Sua Formação Utilizando os Conceitos de Agentes de Software". Revista Brasileira de Informática na Educação, ISSN: 1414-5685, Sociedade Brasileira de Computação, V.11, No. 1, pp. 26-46.

Deb, K. "Encoding and decoding functions". (2000) In: Evolutionary Computation 2: advanced algorithms and operators. Bristol, UK.

Goldberg, D. E. (1989) "Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, Reading.

Knosala, R. and Wal, T. (2001) "A production schedulling problem using genetic algorithm". Journal of Materials Processing Technology, v. 109.

Koriji, T., Ogawa, Y., and Watanabe, T. (2001) "Agent-oriented Supported Environment in Web-based Collaborative Learning". Journal of Universal Computer Science, v. 7, n. 3, p. 226-239.

Koza, J. R. (1992) "Genetic programming: on the programming of computers by means of natural selection". Cambridge, MA: MIT.

Labidi, S., Souza, C. M. and Nascimento, E. (2003) "NetClass: Cooperative Learner Modelling in a Web-Based Environment". In: 6th Int. Conf. on Computer Based Learning in Science (CBLIS). Nicosia, Cyprus: University of Cyprus.

Mitchell, M. "An Introduction to genetic algorithms". (1996) Cambridge, MA: MIT.

Moreno, J. L. (1954) "Fundamentos de la sociometría". Paidós, Buenos Aires.

Olguín, C. J. M., Delgado, A. L. N., Botero, S. W. and Ricarte, I. L. M. (2000) "O Uso de Agentes em Ambientes de Aprendizagem Colaborativos". In: Simpósio Brasileiro de Informática na Educação, Maceió, AL. p. 236-247.

Souza, D. O. de. (2004) "Algoritmos Genéticos Aplicados ao Planejamento do Transporte Principal de Madeira". Curitiba. 184f. Dissertação (Mestrado em Ciências Florestais) - Setor de Ciências Agrárias da Universidade Federal do Paraná.

Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., and Whitley, C. (1991) "A comparison of genetic sequencing operators". In: International Conference on Evolutionary Computation, 1996. Proceedings. Los Altos, CA: Morgan Kaufmann.

Viana, G. V. R. (1998) "Meta-heurísticas e programação paralela em otimização combinatória". Fortaleza: EUFC.

Wolz, U. and Palme, J. *et al.* (1997) "Computer-mediated communication in collaborative educational settings". ACM SIGCUE.

Whatley, J., Beer, M. and Stanford, G. *Facilitation of Online Student Group Project with a Support Agent*. In: Workshop of Agent-Supported Cooperative Work, Autonomous Agents 2001. Montreal, CA. p. 19-23