Resolução de "Timetabling" utilizando Evolução Cooperativa

EDUARDO OLIVEIRA COSTA
MARLONN DELLA BRUNA
ORIENTADORA: PROFA. DRA. AURORA RAMIREZ POZO

UFPR – Universidade Federal do Paraná
DINF – Departamento de Informática
Caixa Postal 19081 - CEP 81531-990 - Curitiba (PR)
{eoc98, mdb98, aurora}@inf.ufpr.br

Resumo: O presente artigo apresenta um estudo sobre Algoritmos Genéticos e sua aplicação em um problema de *Timetabling*. Especificamente, analisa-se o problema de alocação de carga didática na definição de uma grade horária em instituições de ensino. Foi estudada a abordagem denominada Evolução Cooperativa [Potter, 2000], para a resolução deste problema. Uma ferramenta de testes, foi desenvolvida para a execução de experimentos e obtenção de resultados.

Palavras Chave: Inteligência Artificial, Pesquisa Operacional, Computação Evolucionária, Algoritmos Genéticos, Evolução Cooperativa, *Timetabling*, Alocação de Carga Didática.

1 Introdução

O presente artigo tem como propósito a realização de um estudo sobre Algoritmos Genéticos e sua aplicação na resolução de problemas de *Timetabling*. Em particular, é analisado o problema de alocação de carga didática na definição da grade horária em instituições de ensino.

O problema de *Timetabling* é citado como um dos problemas mais interessantes da Pesquisa Operacional [Michalewicz, 1996]. Ele vem sendo intensivamente estudado e reconhece-se como um problema *NP-hard*. Este problema incorpora muitas restrições de diferentes tipos. Desta forma, ele tem se mostrado de difícil solução sob o ponto de vista computacional, devido ao grande número de variáveis e restrições que devem ser atendidas para a obtenção de uma solução ótima. Técnicas convencionais têm se mostrado ineficientes e muitas vezes impraticáveis. Isto motiva este trabalho, apesar de não ter a intenção de neste estudo inicial propor um método que otimize problemas de desempenho e custo computacional, porém deseja-se, analisar a viabilidade de aplicar alguma técnica de Algoritmos Genéticos na solução de problemas de *Timetabling*.

Segundo [Wren, 1996], o problema de *Timetabling* pode ser definido como o arranjo dentro de padrões de tempo ou espaço, no qual algumas metas são atendidas ou praticamente atendidas e onde restrições devem ser satisfeitas ou praticamente satisfeitas.Ele também pode ser caracterizado como um problema de otimização [Michalewicz, 1996].

Nesse trabalho, estuda-se a criação de uma grade horária em uma instituição de ensino. Neste caso, podem ser citadas as seguintes restrições: preferência de horários dos professores, número de aulas de cada disciplina e principalmente as disciplinas ministradas por um mesmo professor não podem coincidir em horário.

Atualmente, várias técnicas têm sido propostas para resolução deste problema, o qual possui uma complexidade bastante alta. Entretanto, nenhuma técnica têm se mostrado muito melhor ou eficiente que outra, o que a denominaria como método ideal. Tendo isso em vista, o que se pretende analisar é mais uma possibilidade, pelo potencial que ela representa. Isto nos leva a um estudo mais aprofundado nos Algoritmos Genéticos e sua aplicação em problemas de otimização sujeito a restrições. A abordagem de Evolução Cooperativa foi implementada para solucionar este problema, utilizando-se uma ferramenta criada com este propósito.

A organização deste artigo é feita da seguinte forma: na seção dois são discutidos os conceitos relacionados a Algoritmos Genéticos e Problemas de Otimização. Na seção três é apresentada a abordagem estudada - Evolução Cooperativa. Na seção quatro é descrito o objeto de estudo, o qual propõe-se resolver e os experimentos realizados.

Na seção cinco são mostrados os resultados dos experimentos e por fim, na seção seis são apresentadas as conclusões e trabalhos futuros.

2 Algoritmos Genéticos e Problemas de Otimização

2.1 Algoritmos Genéticos

Algoritmos Genéticos são algoritmos de busca baseados nos mecanismos de seleção natural e da genética [Goldberg, 1989]. Eles exploram a idéia da sobrevivência dos indivíduos mais adaptados e do cruzamento de populações para criar novas e inovadoras estratégias de pesquisa.

Tais algoritmos pertencem à classe dos algoritmos probabilísticos. Entretanto, não são métodos de busca puramente aleatórios. Combinam elementos de busca direcionada e estocástica. Eles trabalham com uma população de soluções em vez de processar um único ponto no espaço de busca a cada instante.

O processo de busca é multi-direcional. Há a preservação das múltiplas soluções candidatas encorajando a troca de informação entre elas. A cada geração, soluções “boas” se reproduzem, enquanto que soluções “ruins” são eliminadas. Uma função-objetivo é empregada para simular o papel da pressão exercida pelo meio sobre cada indivíduo.

Os Algoritmos Genéticos utilizam o conhecimento adquirido de gerações anteriores de *strings* para a construção de uma nova geração que irá se aproximar da solução ótima.

A idéia básica de funcionamento dos Algoritmos Genéticos é a de tratar as possíveis soluções do problema como "indivíduos" de uma "população", que irá "evoluir" a cada iteração ou "geração". Para isso, é necessário construir um modelo de evolução onde os indivíduos sejam soluções de um problema.

2.1.1 Terminologia Biológica

Os Algoritmos Genéticos utilizam uma terminologia originada da Teoria da Evolução Natural e da Genética. Na Tabela 1, são descritos os principais termos biológicos utilizados no contexto dos Algoritmos Genéticos.

Termo	**Descrição biológica**	**Descrição no contexto dos Algoritmos Genéticos**
Cromossomo	Strings de DNA	String de bits. Indivíduo-candidato à solução do problema.
Alelo	Característica do gene	Valor do bit (0 ou 1, supondo codificação binária)
Lócus	Posição do alelo no cromossomo	Posição do bit no cromossomo.
Genótipo	Conjunto particular de genes contidos no conjunto de todos os cromossomos. Exemplo: TTA CGC CCA	Conjunto particular de genes contido no conjunto de todos os cromossomos. Exemplo: 0001 0010 0100
Fenótipo	Características físicas do indivíduo. Exemplo: Olhos azuis	Característica física do indivíduo. Exemplo: 0001 representa 1

Tabela 1: Terminologia Biológica no Contexto dos Algoritmos Genéticos

2.1.2 Componentes de um Algoritmo Genético

Basicamente, um Algoritmo Genético deve conter os seguintes componentes [Concilio, 2000]:

- uma representação genética para soluções candidatas ou potenciais (processo de codificação);
- uma população inicial, gerada aleatoriamente;
- função de avaliação que faz o papel da pressão ambiental, classificando as soluções de acordo com sua adaptação ao meio;
- operadores genéticos;
- valores para os parâmetros genéticos (tamanho da população, probabilidades de aplicação dos operadores, entre outros).

A seguir, são descritos os principais componentes de um Algoritmo Genético:

Processo de Codificação

Dado determinado problema, cuja solução será obtida através de Algoritmos Genéticos, a definição da codificação (representação) dos indivíduos é de extrema importância. A codificação é uma das etapas mais críticas na definição de um Algoritmo Genético e sendo inadequada pode levar a problemas de convergência prematura [Concilio, 2000].

No modelo clássico de Algoritmos Genéticos, proposto por [Holland, 1992], as soluções candidatas são codificadas em arranjos binários de tamanho fixo.

A motivação para tal codificação baseia-se principalmente no *Teorema dos Esquemas*, proposto por John Holland em 1975. Segundo [Holland, 1992], o Teorema dos Esquemas é benéfico para o desempenho do algoritmo, maximizando o paralelismo implícito do Algoritmo Genético.

Entretanto, em várias aplicações a utilização de codificação binária tem se mostrado ineficiente, por exemplo, em problemas de otimização com parâmetros reais. [Michalewicz, 1996] argumenta que a representação binária apresenta desempenho pobre quando aplicada a problemas numéricos com alta dimensionalidade e onde alta precisão é requerida.

Além da codificação binária, outros tipos podem ser citados [Goldberg, 1989]:

- Codificação por Permutação: utilizada em problemas de ordenação, por exemplo: no problema do Caixeiro Viajante, os cromossomos representam a ordem em que as cidades serão visitadas.
- Codificação por Valor: utilizada em problemas que lidam diretamente com números reais, onde o uso da codificação binária seria muito difícil. Exemplo: encontrar os pesos de uma Rede Neural Artificial.
- Codificação em Árvore: principalmente utilizada em Programação Genética, onde os problemas envolvem expressões.

Vale lembrar que a adoção de outra técnica de codificação pode implicar no desenvolvimento de novas operações de crossover e mutação, devido ao tipo de valor contido em cada cromossomo.

População Inicial

Normalmente a geração da população inicial é completamente aleatória, entretanto podem ser consideradas algumas exceções. Por exemplo, se na solução final sabe-se que serão apresentados mais 0s (zeros) do que 1s (uns), essa informação poderá ser utilizada na geração dos indivíduos. Em problemas com restrições, deve-se apenas gerar indivíduos válidos (ou factíveis).

O tamanho da população é um parâmetro genético de extrema importância para a execução do Algoritmo Genético. Com uma população pequena, não irá haver uma grande variabilidade genética, o que, dependendo da evolução poderá causar estagnação do processo evolutivo. Já uma população grande demais, poderá tornar o algoritmo extremamente lento e com baixo rendimento em termos de processamento computacional.

Função de Avaliação (*Fitness*)

Segundo [Goldberg, 1989], a função de *fitness* (*f)* pode ser pensada como uma medida de desempenho, lucratividade, utilidade e excelência que se queira maximizar. Usando conceitos biológicos, o *fitness* é associado ao seu grau de resistência e adaptabilidade ao meio onde o indivíduo vive. Assim, os indivíduos com maior *fitness* terão uma chance maior de sobreviver e serão responsáveis pela próxima geração. Nos Algoritmos Genéticos isso funciona da mesma forma. É associado um valor numérico de adaptação, o qual se supõe que é proporcional à sua "utilidade" ou "habilidade" em função do seu objetivo. A função de *fitness* deve ser rápida de ser calculada, já que será aplicada a cada indivíduo representado.

Seleção

Após o cálculo do *fitness* em toda a população, são escolhidos alguns indivíduos para, a seguir, serem aplicados os operadores genéticos. Essa seleção é considerada de acordo com seu valor de *fitness*, ou seja, o indivíduo com o

maior *fitness* tem a maior probabilidade de contribuir com um ou mais filhos na próxima geração [Goldberg, 1989]. Nas populações naturais o *fitness* é determinado pela habilidade da criatura sobreviver aos predadores, resistência a doenças, e outros obstáculos na vida adulta.

Um modo de realizar a seleção é o Método da Roleta [Goldberg, 1989], onde cada indivíduo recebe um valor, que analogamente seria sua porção na roleta. Esse valor é uma proporção entre o seu *fitness* e a soma dos *fitness* da população. Isso faz com que o indivíduo que tenha maior *fitness*, conseqüentemente tenha maior chance de ser escolhido.

Neste método, entretanto, existe a possibilidade de que o melhor indivíduo não seja escolhido para a próxima geração, pois sua probabilidade de ser escolhido não é 100%. Para que isso não ocorra, pode-se utilizar uma estratégia chamada de Estratégia Salvacionista [Michalewicz, 1996] onde uma porcentagem da população com os melhores *fitness* é preservada para a próxima geração automaticamente.

Além do Método da Roleta, existem outros métodos que podem ser implementados para se efetuar a seleção, dentre os quais: Integral (respeita rigidamente o *fitness* relativo), Torneio (indivíduos selecionados aleatoriamente disputam um torneio, no qual o melhor é selecionado para a reprodução) e Aleatória (são selecionados aleatoriamente *N* indivíduos da população).

Operadores Genéticos

- Cruzamento

Também chamado de recombinação, Esse operador cria novos cromossomos (os filhos) para a nova população que será criada, usando a combinação de dois cromossomos (pais) da população atual. Para isso, selecionam-se dois indivíduos e um ponto, onde será quebrado o cromossomo. Assim, na geração dos filhos, ocorrerá a troca de informações entre os cromossomos-pais a partir do ponto escolhido. Essa operação ocorre de acordo com uma certa probabilidade, que é calculada a partir da taxa de *crossover* ou taxa de cruzamento.

Além do *crossover* em um único ponto, existem outros tipos de *crossover*. Um exemplo é o *crossover* em dois pontos, onde, são escolhidos dois pontos para quebra dos cromossomos e a troca de informações entre os pais. Há também o *Crossover* Uniforme [Syswerda, 1989], onde para cada alelo é decidido qual dos pais contribuirá com o bit e qual filho receberá esse bit.

De acordo com os resultados obtidos a partir de experimentos realizados por [Eshelman, 1989], o pior desempenho ocorre com o crossover em um ponto, e nenhum dos operadores apresentou em todos os casos melhor desempenho que os demais. Com isso, conclui-se que cada operador de crossover é eficiente para um conjunto de casos e ineficiente para outros conjuntos.

- Mutação

O operador de mutação altera alguns bits do cromossomo randomicamente. A mutação é um processo muito útil, a idéia intuitiva por trás desse operador é a criação de diversidade e variabilidade extra na população, mas sem atrapalhar o progresso já alcançado no Algoritmo Genético. Considerando que a codificação utilizada é a binária, o operador de mutação clássico, segundo [Holland, 1992], troca o valor do alelo selecionado.

Parâmetros Genéticos

Os principais parâmetros que devem ser considerados são:

- Tamanho da População

Afeta diretamente o desempenho e a eficiência do Algoritmo Genético. Uma população muito pequena oferece uma pequena cobertura do espaço de busca, causando uma queda no desempenho pela demora para encontrar uma solução ou até mesmo podendo nunca chegar a uma solução ótima. Se a população for muito grande, tornam-se necessários recursos computacionais maiores, ou um tempo maior de processamento. Logo deve-se buscar um equilíbrio no que diz respeito ao tamanho escolhido para a população.

- Taxa de *Crossover* (P_c)

Quanto maior esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Isto pode gerar um efeito indesejado, pois a maior parte da população será substituída, ocorrendo assim a perda da variedade genética. Com uma taxa baixa, o algoritmo pode tornar-se muito lento para oferecer uma resposta adequada.

- Taxa de Mutação (P_m)

Previne que a busca fique estagnada, introduzindo características que podem ser essenciais a solução do problema. Geralmente essa probabilidade, onde são alterados os bits do cromossomo é muito baixa. Porém, uma taxa de mutação muito alta pode tornar a busca essencialmente aleatória.

- Número de Gerações

Este, juntamente com o tamanho da população, define diretamente o tamanho do espaço de busca a ser coberto.

É importante salientar que a influência de cada parâmetro no desempenho do algoritmo depende da classe de problemas que está se tratando.

2.1.3 Funcionamento de um Algoritmo Genético

De posse dos componentes necessários ao desenvolvimento de um Algoritmo Genético, pode-se visualizar através da figura 1 os passos de um Algoritmo Genético simples.

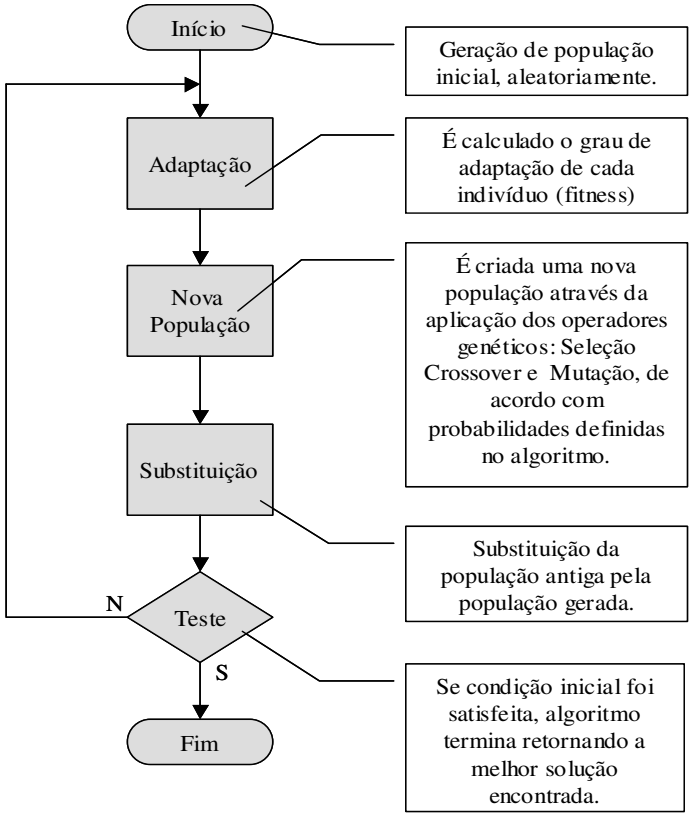

Figura 1: Funcionamento de um Algoritmo Genético Simples

2.2 Algoritmos Genéticos e Problemas de Otimização com Restrições

A aplicação de Algoritmos Genéticos na solução de problemas de otimização com restrições tem se mostrado interessante sob dois pontos de vista [Eiben, 1997]:

- Não se pode esperar que um algoritmo clássico de busca possa alcançar com facilidade a solução de um problema de satisfação de restrições. Várias heurísticas de busca se mostram eficientes em certos casos, mas falham em outros por restringir o escopo da busca.
- Algumas instâncias de problemas com restrições podem ser resolvidas pela diversificação da busca, pela manutenção de vários candidatos à solução em paralelo e pela aplicação de heurísticas que agreguem mecanismos de construção aleatória dos novos candidatos à solução. Tais princípios são essenciais dentro dos algoritmos evolucionários, portanto, sua aplicação na resolução de problemas com restrições apresenta-se bastante promissora.

Outro aspecto importante a ser considerado, é que algoritmos evolutivos são tradicionalmente utilizados em problemas de otimização que não apresentem nenhum tipo de restrição. Um motivo de extrema importância é que operadores genéticos convencionais não tratam diretamente restrições. Um exemplo desta deficiência é a impossibilidade de garantir que após uma operação de *crossover*, pais originalmente factíveis gerem filhos factíveis.

2.2.1 Caracterização dos Problemas de Otimização

A seguir tem-se uma breve descrição da divisão dos problemas de otimização, segundo [Eiben, 1997].

Convenção utilizada :

S : espaço de busca;

f : função-objetivo em S;

ϕ : função *booleana* em S.

FOP (*free optimization problem*) : Um problema de otimização sem restrições é formado por um par (S,f), onde f deve ser otimizada (minimizada ou maximizada). A solução de um FOP é um elemento $s \in S$, com um valor ótimo de f.

CSP (*constraint-satisfaction problem*): Um problema de satisfação de restrições é formado por um par (S, ϕ). A solução de um CSP é um elemento $s \in S$ *com* $\phi(s)$ = verdadeiro

COP (*constraint optimization problem*): Um problema de otimização com restrições é formado por uma tupla (S, f, ϕ). A solução de um COP é um elemento $s \in S$, com um valor ótimo de f e com $\phi(s)$ = verdadeiro.

Para ilustrar estes conceitos será utilizado o problema do Caixeiro Viajante. Este problema é de difícil solução sob o ponto de vista computacional. O objetivo do Caixeiro Viajante é percorrer x cidades prestando serviços ou entregando encomendas, por exemplo. Supondo A e B cidades que compõem a rota do Caixeiro Viajante:

- O problema é um FOP se o objetivo é obter o menor caminho de A até B (otimização)
- O problema é um CSP se o objetivo é passar por todas as cidades (satisfação de restrições)
- O problema é um COP se o objetivo é obter o menor caminho de A até B, passando por todas as cidades (otimização e satisfação de restrições)

Para *CSPs* e *COPs*, ϕ é chamada de condição de factibilidade, e o conjunto $\{ s \in S \mid \phi(s) = \text{verdadeiro} \}$ é o espaço de busca factível.

Em *CSPs* pode-se verificar que não existe a presença de uma função objetivo (função de *fitness*), essencial em algoritmos evolutivos. Em vista disso, é necessário que um *CSP* seja transformado num *FOP* ou num *COP*, embora esta transformação nem sempre seja possível ou viável.

Neste trabalho, o problema de alocação de carga didática é caracterizado como um problema de otimização com restrições (*COP*). Deseja-se não só garantir que todas as restrições sejam satisfeitas, mas também, que uma solução ideal (grade horária ideal) seja obtida baseada na preferência dos professores.

Existem várias técnicas utilizadas para se trabalhar com espaços de busca com restrições, dentre as quais vale ressaltar: Algoritmos de Reparação [Michalewicz, 1996], Decodificadores [Michalewicz, 1996], Algoritmos Meméticos [Concilio 2000], Funções com penalidades [Eiben, 1997]. Neste artigo, procurou-se implementar e verificar o desempenho de uma abordagem até agora pouco aplicada para o problema em questão, que será descrita a seguir.

3 Evolução Cooperativa

Neste trabalho foi implementada a abordagem denominada Evolução Cooperativa, proposta por [Potter, 2000], para se trabalhar com o problema em questão.

Esta abordagem pode ser considerada como uma extensão ao modelo tradicional de Algoritmos Genéticos, onde é possível representar e solucionar problemas complexos através da modelagem da coevolução de espécies [Potter, 2000]. Esta arquitetura modela um ecossistema consistindo de duas ou mais espécies. As espécies são geneticamente isoladas, ou seja, possuem suas próprias características e seus indivíduos somente cruzam com outros membros de sua espécie. Restrições de cruzamento são forçadas simplesmente por evoluir as espécies em populações separadas.

As espécies interagem entre si dentro de um modelo de domínio compartilhado e têm um relacionamento cooperativo. Assim, um indivíduo de uma espécie qualquer terá maiores chances de criar descendentes e participar de gerações futuras a medida que possua uma boa capacidade de cooperação com indivíduos de outras espécies.

3.1 Algoritmo de Evolução Cooperativa

Neste modelo, cada espécie é evoluída dentro de sua própria população. Para avaliação de um indivíduo, é formado um grupo com representantes de cada espécie. Esse grupo é denominado "modelo do domínio". O indivíduo a ser avaliado é combinado nesse modelo, onde é verificado o quanto o mesmo colabora e compatibiliza para a obtenção da solução do problema.

Há muitos métodos possíveis para escolher os representantes com os quais cooperar. Em alguns casos é apropriado permitir que o melhor indivíduo corrente de cada população seja o representante. Em outros casos, estratégias alternativas são preferidas [Cavalheiro, 2002].

O Algoritmo de Evolução Cooperativa inicia criando as populações de cada espécie e através do Algoritmo Genético, é calculado o valor de *fitness* de cada membro das espécies. Se uma solução satisfatória não for encontrada na criação da população inicial, todas as espécies são evoluídas.

Nesta etapa, são efetuadas as operações do Algoritmo Genético (seleção, *crossover* e mutação), já descritas anteriormente, em cada espécie separadamente. As populações antigas são substituídas pelas populações geradas após a aplicação dos operadores.

Após, são selecionados os representantes de cada espécie, formando assim o modelo do domínio. Com o modelo criado, cada indivíduo de cada espécie será avaliado, verificando o grau de colaboração em relação aos indivíduos do modelo. Este grau é calculado com base nas restrições que têm influência direta com a colaboração das espécies. Potter e De Jong se referem a isto como uma colaboração porque os indivíduos serão julgados de acordo a quão bem eles trabalham juntos para resolver o problema.

Outro problema que pode ocorrer no processo de evolução é a estagnação. Pode ser que existam poucos indivíduos no ecossistema com o qual construir uma boa solução. Caso isto ocorra, a população da espécie que estiver estagnada terá sua população novamente aleatoriamente inicializada. A estagnação da evolução pode ser descoberta monitorando-se a qualidade das cooperações, de acordo com a equação:

$$f(t) - f(t - K) < C \qquad \text{(i)}$$

Onde, $f(t)$ é o valor de *fitness* da melhor cooperação na geração t, C é uma constante especificando o aumento do valor de *fitness*, considerando-se ter uma melhoria significante (ou esperada) , e K é uma constante especificando o tamanho da janela evolutiva na qual uma melhoria significante deve existir.

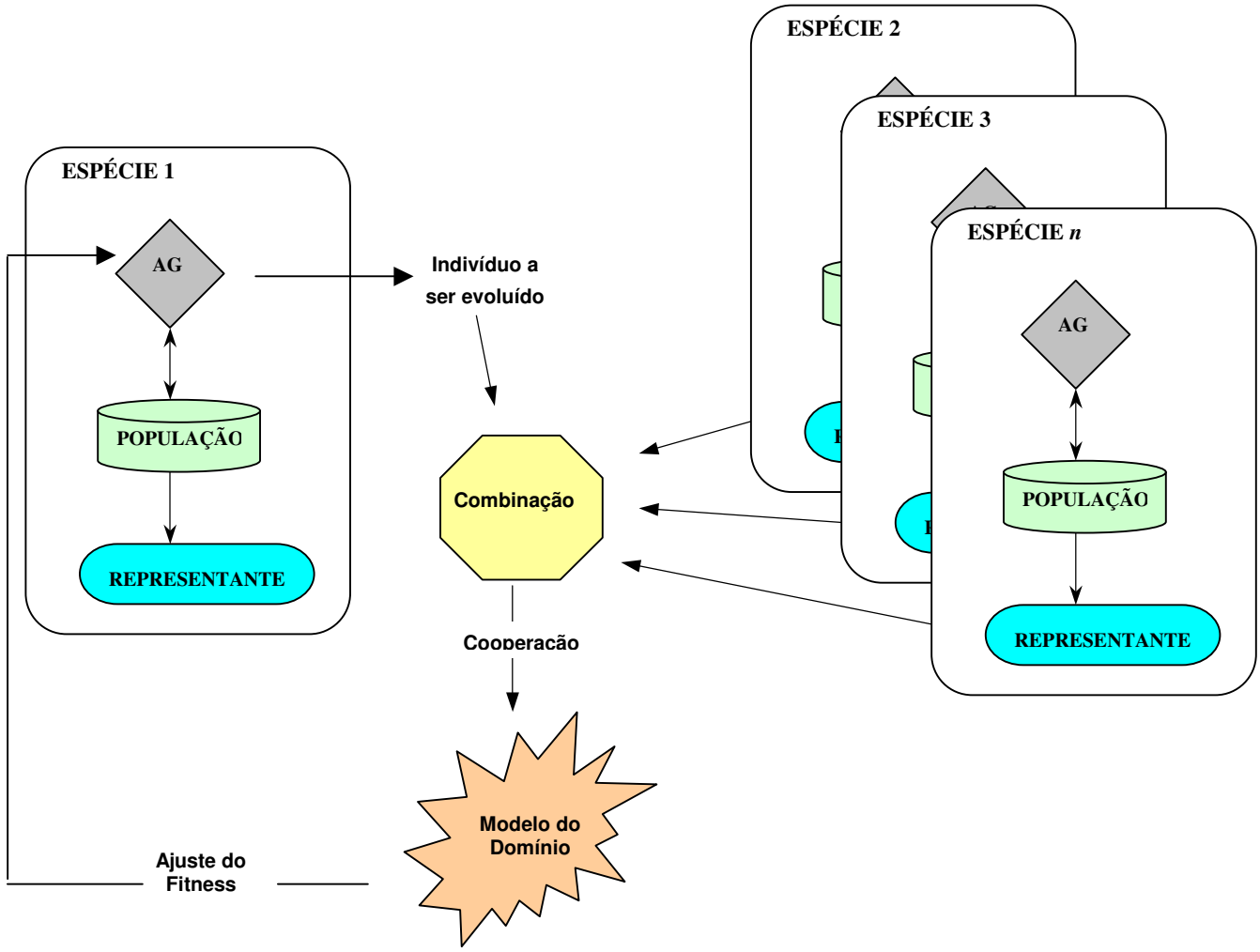

Figura 2: Modelo da Arquitetura de Evolução Cooperativa

4 Estudo de Caso

4.1 O problema da alocação de professores

A criação de uma grade horária em uma instituição de ensino é uma atividade muito importante, já que depois de elaborada, ela será utilizada durante todo o período letivo, fazendo com que todos tenham que se adaptar aos horários, influenciando assim todo o corpo docente e discente da instituição.

O que a maioria das pessoas não sabe é a complexidade da elaboração de toda a grade, devido à grande quantidade de regras e restrições que devem ser respeitadas, por exemplo [Concilio, 2000]:

- O número de aulas de cada disciplina;
- A preferência de horário de cada professor;
- Se duas ou mais matérias forem do mesmo professor, elas deverão estar em horários diferentes;

Na maioria das vezes, o que é feito pela direção da instituição de ensino é, uma vez concluída a grade do ano anterior, promover apenas uma atualização para o ano seguinte. Esta técnica é válida desde que seja suficiente realizar pequenas alterações naquilo que já era funcional [Concilio, 2000]. Entretanto, se houver profundas alterações nas demandas e disponibilidades, isto significará começar o trabalho do zero, o que certamente irá consumir um tempo elevado e sem garantia de produzir uma solução satisfatória.

Uma dificuldade encontrada neste tipo de situação é que, devido às inúmeras particularidades, o processo de solução adotado em um problema geralmente não poderá ser aplicado a outros fins. [Concilio, 2000]

Neste trabalho, está sendo considerado, a criação da grade horária em uma instituição de ensino universitária, em específico para um curso de graduação com quatro anos de duração. O curso é dividido em oito períodos, contém um elenco de 44 disciplinas e possui 30 professores disponíveis para ministrar as disciplinas. As aulas são ministradas no período da tarde e da noite e os horários de aula variam das 13h30 às 22h30. Os professores expressam a sua preferência com relação a cada horário indicando o grau de preferência para ministrarem disciplinas no horário.

O conjunto de restrições escolhido é o seguinte:

- Cada professor deverá ter o máximo de satisfação com relação aos horários alocados.
- Deve ser respeitado o número de aulas por semana de cada disciplina.
- Não pode haver aulas seguidas (aulas geminadas) da mesma disciplina.
- Não pode haver dias da semana sem aulas.
- Disciplinas diferentes, ministradas pelo mesmo professor, não podem coincidir.

O conjunto de restrições acima representa bem o problema em questão. Tais restrições são facilmente encontradas na elaboração de uma grade horária sendo, por este motivo, um bom exemplo de estudo de caso para o propósito deste trabalho.

4.2 Implementação da Ferramenta

Para a construção da ferramenta onde foi implementada a abordagem proposta neste trabalho, foi utilizada a linguagem C++. O objetivo principal foi o de criar um aplicativo simples e que através do qual fosse possível a extração de informações relevantes com relação ao experimento.

O núcleo da ferramenta é composto pelos componentes:

- **Algoritmo Genético** – responsável pela implementação das operações genéticas (seleção, mutação, crossover e cálculo de *fitness*) e também pelo processo como um todo (início/fim).
- **Park-Miller** – componente responsável pela geração de número aleatórios, utilizados nas operações de sorteio.
- **Evolução Cooperativa** – responsável pela avaliação dos indivíduos com relação a cooperação. Testa se um indivíduo coopera com os demais. É responsável também pela verificação da estagnação da população.

Outro componente importante no sistema são as **Bases de Dados**, locais onde é armazenado o conhecimento necessário para processamento do problema em questão. As bases de dados têm ligação direta com o programa principal, definido como Algoritmo Genético.

O funcionamento da ferramenta é bastante simples. De acordo com o tamanho da população são criadas as populações das oito espécies (que correspondem aos oito períodos do curso de graduação escolhido). Nesta etapa as Bases de Dados são acessadas para criação dos indivíduos (grades horárias). Após esta etapa inicial o Algoritmo Genético segue o fluxo de um Algoritmo Genético padrão. É realizada a seleção, cálculo de *fitness* e aplicação dos operadores genéticos. A seguir é efetuado o teste de cooperação entre as espécies e estagnação das populações. Por fim é testado se foi atingida a solução ótima. Espécies que já possuem uma grade horária ideal são retiradas do processo evolutivo, entretanto continuam participando da verificação da cooperação. Para as espécies que ainda não obtiveram uma solução ideal, o processo continua.

A seguir, serão detalhados os pontos mais relevantes da implementação da ferramenta, em termos de arquitetura:

- Codificação do Indivíduo

Um dos principais elementos da etapa de implementação, senão o principal, é a codificação do indivíduo. Neste trabalho, o indivíduo foi codificado conforme a Figura 3.

CI055 - 1	CI060 - 2	CI090 - 3	CI090 - 3	CI051 - 1
XXX - 0	CI055 - 1	CI220 - 4	XXX - 0	CI220 - 4
XXX - 0	CI221 - 3	XXX - 0	CI221 - 3	XXX - 0

Turno: N **Fitness: 50**

Figura 3 : Codificação do indivíduo

O indivíduo contém uma grade de horários, onde cada célula contém o código da disciplina e do professor responsável pela mesma. Cada indivíduo também contém o valor de *fitness* e o turno em que as disciplinas serão ministradas. Neste trabalho foi adotada a seguinte convenção para divisão de horários entre os turnos:

Turno T (tarde) : 13h30, 15h30 e 17h30
Turno N (noite): 17h30, 19h, 21h

Conforme descrito anteriormente, foi implementada a abordagem cooperativa, onde os indivíduos são divididos em espécies. Para tanto, todas as grades horárias do mesmo período foram agrupadas em espécies. Cada espécie contém portanto, uma lista de indivíduos que contém grades horárias do mesmo período. Cada espécie além da lista de indivíduos contém o valor de cooperação da espécie com relação às demais e ao cumprimento do objetivo do problema.

- Sorteio e Aleatoriedade

Para garantir resultados corretos e precisos em problemas onde é utilizada a Computação Evolucionária como solução, é imprescindível que seja utilizado um método de aleatoriedade confiável. Neste trabalho, foi utilizado o método de Park-Miller [Park, 1988] para a geração de números pseudo-aleatórios.

A geração de números se dá a partir de uma semente informada (geralmente é baseada no tempo corrente do sistema). Esta semente pode ser qualquer valor inteiro diferente de zero. A partir da semente inicial gerada, é possível obter números inteiros contidos no intervalo $\{ 1 \leq x \leq (2^{31} - 1) \}$.

- Base de Dados

A base de dados da ferramenta, consiste em duas tabelas:

- Dados das Disciplinas: Contém as informações referentes às disciplinas, tais como: nome, professor designado, carga horária, período, entre outras.

- Preferência de Horários de Professores: Contém a preferência de cada professor sobre determinado horário de aula na grade horária. Esta preferência pode assumir três valores:

0 - Insatisfação em ministrar aulas no horário

3 - Satisfação parcial em ministrar aulas no horário

5 - Plena satisfação em ministrar aulas no horário

Foram geradas várias bases de dados, de forma aleatória, para o preenchimento das preferências de horários e para a distribuição de disciplinas entre os professores.

- Criação da População Inicial

Conforme descrito nas seções anteriores, a criação da população inicial é de extrema importância para o desempenho do Algoritmo Genético. Este trabalho preocupou-se em gerar na criação da população inicial somente indivíduos factíveis, utilizando para tanto Algoritmos de Reparação.

A principal característica que é verificada através dos Algoritmos de Reparação é a de que o número de aulas por disciplina seja rigorosamente respeitada. Esta restrição é verificada através de Algoritmos de Reparação somente nesta etapa. Ao final da criação da população inicial, a restrição número de aulas por semana de cada disciplina está

satisfeita, entretanto, no decorrer do processamento, novamente esta verificação deverá ser feita, tendo seu resultado influência sobre o *fitness*, já que com a aplicação dos operadores genéticos as grades horárias serão modificadas.

A decisão da aplicação de Algoritmos de Reparação nesta etapa, baseou-se principalmente no aumento de performance que o algoritmo obtém nas etapas posteriores, já sendo iniciado com indivíduos factíveis. Apesar desta vantagem, foi observado também que com o aumento do número de indivíduos, o tempo de criação da população inicial aumenta bastante, tendo em vista a aplicação da utilização desta técnica.

- Cálculo de *Fitness*

O cálculo do *fitness* compõem-se da verificação das restrições definidas no problema. Cada restrição recebeu um peso, de acordo com a importância, sendo que o valor de *fitness* varia de 0 a 100, sendo 100 o valor ótimo.

Pela sua importância, a restrição de preferência de horário do professor recebeu peso 6, as demais (número de aulas por dia e impossibilidade de aulas geminadas) receberam peso 2.

O *fitness* relativo à preferência de horário é calculado utilizando-se a fórmula:

$$fitness(x) = (\sum_{i=1}^{m} preferencia_i * 100) / (h * 5) \qquad \text{(ii)}$$

Onde:

$preferencia_i$ é o valor da preferência [0, 3, 5], com relação ao horário *i*, lido da base de dados de preferências de horário;

h é a carga máxima de aulas do período.

A seguir, tem-se um exemplo de cálculo do *fitness*.

Exemplo: Seja a grade horária da Figura 4:

CI055 - 2	XXX - 0	CI090 - 2	CI090 - 2	CI221 - 1
XXX - 0	CI055 - 2	CI220 - 1	XXX - 0	CI220 - 1
XXX - 0	XXX - 0	XXX - 0	CI221 - 1	XXX - 0

Turno: T

Figura 4 : Grade horária com 2 professores e 4 disciplinas.

Supondo que as preferências de horários dos professores sejam as da Figura 5, onde , no exemplo **1 - 0**, representa: código do professor igual a 1, preferência para ministrar aula às 13h30 de segunda-feira igual a 0. Na primeira célula da Figura 5, temos que o professor de código 1 tem preferência 0 (insatisfação) em ministrar aula às 13h30, enquanto o professor de código 2 tem uma preferência 3 (satisfação parcial) em ministrar aula nesse mesmo horário.

	S	T	Q	Q	S
13:30	1 – 0 2 – 3	1 – 0 2 – 0	1 – 5 2 – 3	1 – 5 2 – 0	1 – 5 2 – 5
15:30	1 – 0 2 – 0	1 – 5 2 – 5	1 – 5 2 – 3	1 – 5 2 – 3	1 – 5 2 – 3
17:30	1 – 0 2 – 0	1 – 0 2 – 3	1 – 5 2 – 3	1 – 5 2 – 5	1 – 0 2 – 3

Figura 5 : Tabela com a preferência de horários dos professores

Verificando agora as restrições, com base na Figura 4:

1) Número de aulas por dia ≥ 1 (20 % do *fitness*): Não viola restrição ➔ 20

2) Aulas geminadas (20 % do *fitness*) : Não ocorrem. ➔ 20

3) Verificação da preferência de horário de professores (60% do *fitness*):

3 + 0 + 3 + 0 + 5 + 0 + 5 + 5 + 0 + 5 + 0 + 0 + 0 + 5 + 0 = 31

fitness = (31 * 100)/(8 * 5) = 77,5 * 0,6 ➔ 46,5

Totalizando os valores (20 + 20 + 46,5), tem-se o valor do *fitness* para a grade igual a: 86,5.

- Operadores Genéticos

- Seleção e Elitismo: Foi implementada a seleção por torneio, onde são escolhidos aleatoriamente dois indivíduos da espécie, e vence o torneio o que possuir o maior valor de *fitness*. Também foi implementada a seleção elitista, onde, de acordo com a Taxa de Elitismo, são preservados os melhores indivíduos para a próxima geração.

- Crossover: Para um melhor desempenho da operação neste trabalho, o crossover foi implementado no modo uniforme. São escolhidos aleatoriamente dois "pais" da espécie. São efetuados dois sorteios. O primeiro para determinar de qual pai o alelo será copiado para o filho e um segundo para determinar qual filho irá receber o alelo. O crossover somente é aplicado a indivíduos de mesma espécie.

- Mutação: A mutação tem como objetivo apenas trocar os alelos de posição no indivíduo. Para cada indivíduo de cada espécie são sorteadas duas posições de forma aleatória e o conteúdo de cada posição é trocado.

- Cooperação

Para realizar a cooperação, primeiramente é necessário obter um modelo de domínio na qual todos os indivíduos de todas as espécies serão avaliados. Neste trabalho, o modelo construído é um conjunto com um indivíduo de cada espécie com maior valor de *fitness*.

Na avaliação, cada indivíduo é inserido nesse modelo de domínio no lugar do indivíduo que pertence a sua espécie. A avaliação é realizada com um indivíduo de cada vez. Na avaliação, o indivíduo será penalizado caso existam conflitos de horário entre períodos diferentes. Essa penalização irá afetar diretamente o valor de *fitness*, fazendo assim, com que o processo evolutivo do indivíduo seja alterado. A verificação é realizada em todas as gerações criadas. A função que faz a cooperação deve ser uma função que avalie as relações do indivíduo com toda a população. Neste trabalho, essa função é uma restrição muito importante para o funcionamento do algoritmo: a alocação do mesmo professor que dará aula em mais de um período, alocado em horários coincidentes.

A seguir tem-se um exemplo de grades que serão penalizadas, já que contêm dois horários conflitantes (indicados na Figura 6). Neste exemplo, supõe-se que as grades horárias da Figura 6 pertencem ao grupo dos representantes das espécies.

CI055 - 2	CI210 - 1	CI210 - 1	CI055 - 2	CI221 - 3
CI063 - 4	CI020 - 6	CI020 - 6	CI088 - 5	CI063 - 4
CI088 - 5	XXX - 0	XXX - 0	CI221 - 3	XXX - 0

Fitness: 95 **Turno: T**

Carga Máxima: 12 **Grade 1**

CI237 - 6	CI237 - 6	CI095 - 8	CI011 - 2	XXX - 0
XXX - 0	CI011 - 2	CI220 - 1	XXX - 0	CI095 - 8
XXX - 0	XXX - 0	XXX - 0	CI220 - 1	XXX - 0

Fitness: 80 **Turno: T**

Carga Máxima: 8 **Grade 2**

Figura 6 : Grades de espécies diferentes com o mesmo professor alocado em horários coincidentes.

Na verificação da grade 1 com a grade 2, percebe-se que há um horário em que tem-se duas matérias diferentes (CI055 e CI011), ministradas pelo mesmo professor. Se ambas continuarem evoluindo normalmente, não será possível a obtenção de uma solução ideal, já que um professor pode ministrar matérias diferentes, mas nunca em um mesmo horário.

Após esta verificação, é aplicada às duas grades a seguinte equação, para redução do valor de *fitness*:

$$fitness = fitness - (fitness * ((\sum horarios\_incorretos)/h)) \quad \text{(iii)}$$

Onde: *h* é a carga máxima do período

Aplicando a fórmula (iii) para a grade 1, tem-se: *fitness* = 95 - (95 * (1/12)) ➔ 87,08

Depois de penalizados, os indivíduos são submetidos aos operadores de seleção, combinação e mutação, e em seguida são selecionados os melhores que participarão do modelo de domínio da próxima geração. Esses indivíduos, não necessariamente serão os mesmos da geração anterior. Por isso, um indivíduo que teve sua avaliação de cooperação ruim, pode em uma eventual avaliação futura com outro modelo de domínio obter um bom resultado.

- Estagnação

De 10 em 10 gerações é efetuada a verificação da estagnação da população. Esta verificação compõe-se da comparação do valor de *fitness* dos melhores indivíduos de cada período com uma constante *k* definida. Nesta implementação o valor de *k* é igual a 5, o que significa que deve haver pelo menos o acerto de uma preferência de horário a cada evolução de 10 gerações. Se houver variação menor que a constante uma nova população tem-se a estagnação da população, sendo então criada uma nova população daquela espécie.

5 Experimentos e Resultados

Vários experimentos foram efetuados com o intuito de precisar o desempenho da abordagem escolhida através da ferramenta implementada. Os experimentos executados procuraram medir os mais diversos aspectos, a partir dos quais foi possível obter várias conclusões com relação à abordagem implementada. Foi utilizada a linguagem C++ para desenvolvimento e os experimentos foram executados em uma máquina AMD K6 750MHz, com 128 Mbytes de RAM, rodando Sistema Operacional Linux. Os testes foram divididos em três categorias:

5.1 Tempo de Processamento do Algoritmo

A ferramenta desenvolvida foi implementada buscando-se simplicidade e eficiência. Procurou-se criar algo que não comprometesse ainda mais o custo do processamento do algoritmo. Foram feitos vários testes com o propósito de se medir o tempo gasto pelo processamento do Algoritmo Genético em si, da forma mais aproximada possível.

Neste teste, foram feitos sete experimentos variando-se a população e medindo o tempo de processamento até a obtenção da solução ideal ou do número máximo de gerações (caso a solução ideal não fosse encontrada). A configuração dos parâmetros do Algoritmo Genético para o teste foi a seguinte:

Número de Experimentos: 5
Número máximo de Gerações: 300
Taxa de Crossover: 20%
Taxa de Mutação: 80%
Taxa Elitista: 20% da população

A justificativa para a utilização da taxa de mutação igual a 80% tem embasamento após verificação em diversos trabalhos relacionados a este experimento [Michalewicz, 1996], também sendo observado neste trabalho, conforme será descrito na seção 5.2.

Para estes testes, foram calculadas as médias dos tempos de execução das rodadas para obtenção do tempo de processamento. Com a possibilidade de uma espécie ser reiniciada caso apresente estagnação, os números de gerações de cada espécie podem ser diferentes umas das outras. Portanto, para o cálculo do número de geração da execução foi considerada a maior quantidade de gerações processadas.

Na Tabela 3, tem-se os resultados:

População	Gerações Evoluídas	Tempo de Processamento (s)
10	300 (*)	3,08
50	291	5,89
100	269	13,05
200	121	8,55
400	42	12,02
800	25	29,43
1600	17	71,12

(*) – Número máximo de gerações. O Algoritmo evoluiu até o máximo possível e não encontrou uma solução ótima.

Tabela 3: Resultados do experimento para medição do tempo de processamento do algoritmo

Como pode ser observado pela Tabela 3, o tempo de processamento do algoritmo cresce de acordo com o tamanho da população. Entretanto, o número de gerações evoluídas para a obtenção de uma solução próxima da ideal decresce. Isto comprova que o aumento da população, promove o aumento da diversidade dos indivíduos, o que facilita a convergência para uma solução ideal.

Outro aspecto importante a ser salientado é a rapidez com que se alcança uma solução ótima (*fitness* = 100), o que indica que o espaço de busca é rico em soluções. Problemas de Timetabling envolvem muito mais restrições do que as consideradas aqui neste trabalho, entretanto para fins de verificação de comportamento de uma abordagem, tal simplicidade mostra-se relevante.

5.2 Parâmetros Genéticos

Para o bom funcionamento de um Algoritmo Genético - obtenção de resultados coerentes – é importante que os parâmetros genéticos estejam configurados de maneira adequada, conforme descrito nas seções anteriores.

Vários testes foram executados variando-se as taxas do Algoritmo Genético até a obtenção de um conjunto que melhor conduzisse o processo evolutivo até a solução do problema. Esta variação dos parâmetros genéticos está fortemente relacionada à abordagem implementada.

Parâmetros utilizados nos testes a seguir:

Número de Execuções: 5

Tamanho da População: 100

Número máximo de Gerações: 100

Nos testes que seguem, como medida de desempenho foi adotado o *fitness*, para se tentar verificar se ocorreram ou não alterações significativas com a variação das taxas de aplicação dos operadores genéticos. Nas Tabelas de número 4 a 10, os resultados são divididos em oito colunas que conforme o modelo adotado representam os períodos (1p..8p) do curso de graduação escolhido no Estudo de Caso, e no âmbito da Computação Evolucionária representam

as espécies que fazem parte do ecossistema em questão. Como cada período representa respectivamente uma espécie diferente, eles puderam ser avaliados individualmente.

% Crossover	*Fitness* (%)							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
0	100	100	100	100	100	100	97	100
20	100	100	100	100	97	97	97	100
30	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
40	100	98	100	100	100	100	97	100
50	100	100	100	100	100	100	97	100
60	100	98	100	100	97	100	94	100
70	100	100	100	98	97	100	97	100
80	100	100	100	98	97	100	100	100

Tabela 4: Resultados do experimento para medição da variação do *Fitness*, com variação da taxa de Cruzamento.

Como pode ser observado na Tabela 4, as cinco execuções executadas obtiveram um *fitness* máximo em todos os períodos com 30% de taxa de Crossover. Na Tabela 5, esse resultado ocorreu em duas porcentagens de taxa de Mutação, 50% e 80%. Para os experimentos deste trabalho, foi utilizada a taxa de mutação igual a 80%, já que tal comportamento pode ser verificados em trabalhos relacionados [Michalewicz, 1996] similares a este experimento.

Na Tabela 6 pode-se verificar também que nas cinco execuções efetuadas, obteve-se um *fitness* máximo com taxa Elitista igual a 20%, a qual também foi utilizada nos experimentos posteriores.

% Mutação	*Fitness* (%)							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
0	100	100	100	100	97	100	100	100
20	100	95	97	98	95	97	97	100
30	100	100	96	100	97	97	100	100
40	100	98	100	98	97	100	100	100
50	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
60	100	98	100	98	100	97	97	100
70	100	100	100	98	97	97	100	100
80	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Tabela 5: Resultados do Experimento para medição da variação do *Fitness*, com variação da taxa de mutação

% Elitismo	*Fitness* (%)							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
0	98	98	100	98	82	100	75	100
20	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
30	100	100	100	100	100	97	97	100
40	100	98	100	98	100	97	97	100
50	100	100	100	100	100	97	97	100
60	100	100	100	100	97	100	97	100

Tabela 6: Resultados do experimento para medição da variação do *Fitness*, com variação da taxa elitista.

5.3 Evolução das Espécies

Outro aspecto na validação da abordagem escolhida neste problema é a medição e a verificação se a evolução está ocorrendo de fato. Para se obter resultados mais significantes, foram executados quatro conjuntos de testes, com

cinco execuções cada um, a fim de acompanhar a evolução das espécies. Para os testes abaixo, a configuração do Algoritmo Genético foi a seguinte:

Número de Execuções: 5
Tamanho da População: 130
Número máximo de Gerações: 100
Taxa de Crossover: 30%
Taxa de Mutação: 80%
Taxa Elitista: 20% da população

- *Fitness*

Tradicionalmente, a variação do valor de *fitness* para próximo do valor máximo, é a prova de que o Algoritmo Genético está evoluindo. Neste trabalho, o *fitness* também tem um papel importante na evolução, entretanto de acordo com a abordagem implementada, sua melhoria individual não garante a obtenção de um resultado ideal.

Nesta implementação, o *fitness* é a medição da evolução de uma espécie de forma isolada, sem a preocupação de como esta evolução influi no desenvolvimento das outras espécies.

Na Tabela 7, encontram-se os resultados do experimento. A partir da análise da tabela pode-se verificar que o algoritmo iniciou de forma satisfatória, já que todos os períodos apresentaram *fitness* maior que 90. Na geração 5 é possível identificar uma evolução dos períodos 6 e 7. Na geração 10 percebe-se também que os períodos 3 e 8 atingem o valor de *fitness* igual a 100.

Pode-se também perceber que os demais períodos foram gradativamente evoluindo até chegar a 100, estando todos evoluídos na geração 40. Com exceção dos períodos 5 e 7, que demoraram 59 e 79 gerações, respectivamente, para chegarem a um valor de *fitness* igual a 100. Com o experimento pode-se concluir também que há evolução, mesmo que um pouco demorada (por exemplo: no período 1, foi preciso evoluir 40 gerações para o valor de *fitness* aumentar de 98 para 100).

- Cooperação entre as Espécies

Além do *fitness*, outra medição de extrema importância é a verificação da cooperação entre as espécies, já que é a partir dela que é feita a penalização do *fitness* dos indivíduos e também a verificação da convergência do sistema como um todo. Com base no teste anterior, foi analisado se a cada ponto de verificação do *fitness*, a espécie estava cooperando para a obtenção do objetivo. Neste teste, uma espécie coopera se e somente se todos os indivíduos nela contidos cooperarem também.

Geração	***Fitness***							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
1	98	96	98	98	95	92	91	97
5	98	96	98	98	95	95	94	97
10	98	96	100	98	95	95	94	100
20	98	98	100	98	95	97	94	100
30	98	100	100	98	95	100	97	100
40	100	100	100	100	97	100	97	100
50	100	100	100	100	97	100	97	100
60	100	100	100	100	97	100	97	100
70	100	100	100	100	100	100	97	100
80	100	100	100	100	100	100	100	100

Tabela 7: Resultados do Experimento para medição da variação do *fitness*

Pela Tabela 8, pode-se concluir que gradativamente as espécies iniciam o processo de cooperação. A partir da geração 30, todas as espécies estão cooperando entre si e permanecem neste estado até o final do processamento.

Como pode-se notar, a cooperação entre as espécies também não determina por si só a obtenção de uma solução ideal. Neste caso, após atingir o ponto em que todas as espécies estão cooperando, o algoritmo ainda precisou evoluir mais 50 gerações até obter uma solução ideal.

Geração	**Cooperação entre as Espécies**							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
1	S	S	S	**N**	**N**	**N**	**N**	S
5	**N**	**N**	S	**N**	**N**	**N**	**N**	S
10	S	S	S	**N**	**N**	**N**	**N**	S
20	S	S	S	S	**N**	S	**N**	S
30	S	S	S	S	S	S	S	S
40	S	S	S	S	S	S	S	S
50	S	S	S	S	S	S	S	S
60	S	S	S	S	S	S	S	S
70	S	S	S	S	S	S	S	S
80	S	S	S	S	S	S	S	S

Tabela 8: Resultados do Experimento para medição da Cooperação entre as Espécies

- Estagnação da População

Além do *Fitness* e da Cooperação, outra medida importante para verificação da ocorrência de evolução das espécies nesta implementação é a medida de estagnação da população.

A medida de Estagnação foi tomada de 10 em 10 gerações, com base nos experimentos anteriores. De forma resumida, havendo estagnação da população, o Algoritmo Genético reinicializa a população do período, com indivíduos gerados aleatoriamente, reiniciando assim o processo evolutivo da espécie.

Pela análise da tabela pode-se constatar que com o passar do tempo as gerações foram evoluindo e a estagnação das espécies diminuindo. Na geração 40, apenas dois períodos estavam estagnados (5 e 7), o que comparando-se com a avaliação do *fitness* mostra que a evolução do *fitness* tem relação com a estagnação das espécies.

Outro detalhe importante a ser considerado é que, neste caso, a verificação da estagnação da população representou um papel importante no processo evolutivo, já que foi necessário reiniciar as espécies várias vezes, para se chegar à solução esperada. Para a execução considerada neste experimento, como pode-se verificar pela tabela abaixo, a média de reinicializações da população por período é de ≈3 vezes.

Geração	**Estagnação da População**							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
10	**S**	**S**	N	**S**	**S**	**S**	**S**	N
20	**S**	**S**	N	**S**	**S**	**S**	**S**	N
30	**S**	N	N	**S**	**S**	N	**S**	N
40	N	N	N	N	**S**	N	**S**	N
50	N	N	N	N	**S**	N	**S**	N
60	N	N	N	N	**S**	N	**S**	N
70	N	N	N	N	N	N	**S**	N
80	N	N	N	N	N	N	N	N

Tabela 9: Resultados do Experimento para medição da variação da estagnação da população de 10 em 10 gerações

Outro dado importante obtido através da análise da tabela é a quantidade de indivíduos gerados e processados para a resolução deste problema. O número de reinicializações é igual a 23, multiplicando-se com a população (130)

é igual a 2990 indivíduos, ou seja, foi preciso além dos indivíduos gerados através das operações genéticas, injetar mais 2990 indivíduos novos para conseguir a obtenção da solução.

- ISP – Índice de Satisfação do Professor

Para a verificação do andamento da evolução, além das medidas citadas, foi criado o ISP (Índice de Satisfação do Professor), que representa o grau de satisfação dos professores alocados em dada grade horária. O ISP varia de 0 a 5, conforme grau de satisfação representada na base de dados de preferência de horários.

O ISP caminha proporcionalmente ao *fitness*, entretanto é uma medida isolada bastante útil para verificar o grau de evolução e melhoria com relação à restrição de preferência de horário do professor.

Geração	**ISP - Índice de Satisfação do Professor**							
	1p	**2p**	**3p**	**4p**	**5p**	**6p**	**7p**	**8p**
1	4.83	4.66	4.83	4.84	4.63	4.44	4.25	4.75
5	4.83	4.66	4.83	4.84	4.63	4.6	4.5	4.75
10	4.83	4.66	5	4.84	4.63	4.6	4.5	5
20	4.83	4.83	5	4.84	4.63	4.8	4.5	5
30	4.83	5	5	4.84	4.63	4.8	4.5	5
40	5	5	5	5	4.81	5	4.75	5
50	5	5	5	5	4.81	5	4.75	5
60	5	5	5	5	4.81	5	4.75	5
70	5	5	5	5	5	5	4.75	5
80	5	5	5	5	5	5	5	5

Tabela 10: Resultados do Experimento para medição da variação do ISP

5.4 Considerações Finais

Pode-se verificar que o processo evolutivo, nesta implementação, é bastante dependente do valor de *fitness*, da cooperação entre as espécies e da estagnação da população. Para um bom desempenho do processo evolutivo é necessário a conjunção desses três fatores.

O Algoritmo Genético apresentou resultados, com relação a tempo de processamento e execução satisfatórios. Outra característica bastante visível nos resultados obtidos é a importância da aleatoriedade no processo. Sem um método eficiente de geração de números aleatórios, o processo como um todo estaria comprometido, já que é a partir dele que são gerados os indivíduos de cada espécie e sorteadas as ocorrências para aplicação dos operadores genéticos.

6 Conclusão

Tendo em vista a complexidade do problema exposto e os resultados obtidos, pode-se concluir que a abordagem cooperativa também é uma alternativa para a solução do problema de Timetabling. O analises da qualidade da solução sempre é de difícil avaliação visto que o estudo de caso não foi resolvido por outras técnicas e também pela escolha de um caso real o que dá algumas características mais específicas ao problema.

Como principais contribuições deste trabalho, podem ser mencionadas:

- O estudo e aplicação de técnicas de computação evolutiva (Algoritmos Genéticos) em conjunto com o problema de restrições para a solução do problema de alocação de professores em horários de instituições de ensino;
- A construção de um Algoritmo Genético com Restrições, baseando-se na proposta de Evolução Cooperativa;

A partir da análise dos resultados obtidos e ilustrados no capítulo anterior, pode-se concluir que as técnicas implementadas em conjunto favorecem a obtenção de uma solução desejada para o problema. Em relação à aplicação baseada no estudo de caso, pode-se concluir que a abordagem estudada e construída tem um resultado satisfatório, podendo ser utilizada em instituições de ensino que possuam restrições e problemas com a satisfação do corpo docente em relação a grade horária a ser estabelecida.

As razões levantadas para o bom funcionamento da abordagem implementada baseada na técnica de Evolução Cooperativa são:

- O fato de todos os indivíduos das populações serem factíveis. Com isso, não há a perda de tempo desnecessária em primeiro achar um indivíduo factível e em seguida procurar otimizá-lo.
- A Evolução Cooperativa, por ser dividida em espécies, é processada de maneira paralela, isto é, os indivíduos de uma certa espécie que tiverem um valor de *fitness* baixo não prejudicarão indivíduos que pertençam a uma outra espécie. Entretanto, isso não significa que eles sejam independentes, já que os indivíduos são avaliados em conjunto com os demais e devem cooperar para a obtenção do cumprimento do objetivo.

Como perspectivas futuras para este trabalho, pode-se sugerir os seguintes tópicos:

- Verificação da necessidade de elaborar novos operadores genéticos que possam adaptar-se melhor à cada contexto de aplicação.
- Verificação do desempenho da abordagem com novas restrições.
- Testar o sistema em uma situação real, obtendo dados de professores e disciplinas de uma instituição de ensino existente.

Ao final do presente artigo conclui-se que Algoritmos Genéticos na sua forma tradicional encontram dificuldades no tratamento de problemas de otimização com restrições. Por outro lado, a aplicação de técnicas de otimização baseadas em restrições quando aplicadas isoladamente podem encontrar sérias dificuldades, em virtude da existência de ótimos locais de baixa qualidade [Concilio, 2000].

Deste modo, a aplicação de técnicas conjuntas, conforme proposto neste trabalho, mostra que resultados satisfatórios podem ser obtidos, desde que haja um bom equilíbrio entre as técnicas e metodologias a serem empregadas.

7 Referências

[Ahuja, 1993] Ahuja, R. K., Magnanti T. L., Orlin, J. B. **Network Flows: Theory, Algorithms and Applications**. Prentice Hall, Englenwood Cliffs, EUA, 1993.

[Cavalheiro, 2002] Cavalheiro, A. F. **GADBMS – Um Algoritmo Genético Minerador de Dados para Base de Dados Relacionais**. Dissertação de Mestrado em Informática. Departamento de Informática da Universidade Federal do Paraná – UFPR. Curitiba PR, 2002.

[Concilio, 2000] Concilio, R. **Contribuições à solução de problemas de Escalonamento pela Aplicação Conjunta de Computação Evolutiva e Otimização de Restrições**. Departamento de Engenharia da Computação e Automação Industrial. Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas – UNICAMP, São Paulo SP, 2000.

[Eiben, 1997] Eiben, A. E., Ruttkay, Z. **Constraint-Handling Techniques**, in Handbook of Evolutionary Computation, Oxfor University Press, 1997.

[Eiben, 2000] Eiben, A. E. **Solving CSPs using self-adaptative constraint weights: how to prevent EAs from cheating** at the 2000 Genetic and Evolutionary Computation Conference (GECCO-2000), Las Vegas, Nevada, USA, 2000.

[Eshelman, 1989] Eshelman, L. J., Caruana, R.A., Schaffer, J. D. **Biases in the Crossover Landscape**, in Schaffer, J. D. (ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1989.

[Goldberg, 1989] Goldberg, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley, 1989.

[Holland, 1992] Holland, J. H. **Adaptation in Natural and Artificial Systems**. MIT Press, 2ª. edição, 1992.

[Michalewicz, 1996] Michalewicz, Z., Schoemauer, M. **Evolutionary Algorithms for Constrained Parameter Optimization Problems.** Evolutionary Computation, 1996.

[Michalewicz, 1997] Michalewicz, Z. **Constraint-Handling Techinques**, in Handbook of Evolutionary Computation. Oxford University Press, 1997.

[Park, 1988] Park, S.K., Miller K.W. **Randon Number Generators:Good Ones Are Hard to Find**. Communications of the ACM, p. 1192-1201, 1988.

[Potter, 2000] Potter M. A., De Jong K. A. **Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents**. In Evolutionary Computation, MIT Press, 2000.

[Syswerda, 1989] Syswerda, G. **Uniform Crossover in Genetic Algorithms**, in Schaffer, J. D. (ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1989.

[Wren, 1996] Wren, A. **Scheduling, timetabling and rostering – a special relationship?** in The Practice and Theory of Automated Timetabling: Selected Papers from the 1st Internacional Conference, Lecture Notes in Computer Science. Berlin, 1996.