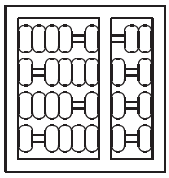


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Algoritmos Genéticos Aplicados ao Problema de Agendamento de Atividades Acadêmicas

Volnei dos Santos *Arnaldo Vieira Moura*

Technical Report - IC-03-027 - Relatório Técnico

December - 2003 - Dezembro

Algoritmos Genéticos Aplicados ao Problema de Agendamento de Atividades Acadêmicas

Volnei dos Santos [*]
volnei.santos@ic.unicamp.br

Arnaldo Vieira Moura [†]
arnaldo@ic.unicamp.br

17 de dezembro de 2003

Resumo

Este trabalho trata da utilização de heurísticas evolutivas na resolução do problema de agendamento de salas, professores e alunos, conhecido na literatura como *Timetabling*. O projeto é embasado no uso de *Algoritmos Genéticos*, uma classe de heurísticas que vêm demonstrando grande eficácia no tratamento de problemas NP-Difícies. A abordagem do trabalho é voltada para uma instância do problema adaptada à realidade brasileira e visa também, através da criação de uma interface amigável para usuários, prover uma ferramenta útil para instituições de ensino.

Palavras-chave: Algoritmos Genéticos, Algoritmos Evolutivos, Otimização, Timetabling.

[*]Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

[†]Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Pesquisa desenvolvida com suporte financeiro da FAPESP, processo 02/04164-3

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

Todas as instituições que lidam com a fabricação de algum produto ou com a prestação de algum serviço deparam-se com um problema crucial: a otimização de seus recursos (maquinário ou pessoal), de modo a conseguir uma melhor produtividade, qualidade nos serviços e redução de custos.

Dentre esse tipo de problema encontram-se os problemas de escalonamento (*Scheduling*) e agendamento (*Timetabling*). Existem diversos tipos de problemas de agendamento, sendo que uma classe que merece destaque é a voltada para agendamento de aulas, turmas e professores em instituições de ensino. Em geral, as escolas e faculdades não possuem softwares especializados que permitam a otimização de suas agendas, e despendem dias, ou mesmo semanas, de trabalho manual para encontrar uma solução razoável.

Podemos citar como motivo para a pouca disponibilidade de sistemas deste tipo a complexidade e as especificidades de cada instância do problema. Via de regra, esses são problemas classificados como NP-Difíceis [SCH95, CDM90a, CDM90b, CDM90c]. Além disso, a maioria dos trabalhos publicados sobre o assunto geralmente é voltada para instâncias adaptadas às escolas nos Estados Unidos e Europa [CL98, SCH96]. Existem poucos trabalhos bem sucedidos adaptados à realidade brasileira.

A inexistência de algoritmos polinomiais para a resolução do problema leva ao uso de técnicas heurísticas. Um conjunto de técnicas que têm apresentado resultados satisfatórios em diversas aplicações são coletivamente chamados de *Algoritmos Evolutivos* e, dentre esses, temos em especial os *Algoritmos Genéticos* (GA) [SCH95, CL98, LRF01, CDM90a, CDM90b, CDM90c, BEW95, RCF95, RCF94, FAN94]. Os algoritmos genéticos usam heurísticas inspiradas nas idéias de evolução e seleção naturais. A aplicação dessa técnica na busca de soluções para problemas de otimização combinatória, em especial problemas de timetabling, tem levado a algoritmos relativamente eficientes e que convergem para resultados satisfatórios, em tempo de processamento razoável [CL98, CDM90b, RCF95].

Este projeto visa estudar a utilização de algoritmos genéticos na resolução de problemas de agendamento acadêmico para escolas de nível fundamental e médio. O problema a ser tratado, conhecido na literatura como *School Timetabling* [SCH95, CL98], receberá uma abordagem voltada para uma instância adaptada à realidade brasileira, tratando das restrições típicas das instituições de ensino nacionais.

Outros projetos do GOA (Grupo de Otimização Aplicada do Instituto de Computação da Unicamp)[1] estão tratando o mesmo problema utilizando outras técnicas de resolução[2], o que permitirá futuramente uma comparação entre os métodos desenvolvidos. Também está sendo desenvolvida por esse grupo de alunos uma interface de uso comum para todos os projetos, o que permitirá a utilização das ferramentas pelos usuários finais.

Durante o processo de desenvolvimento, o grupo contou com o apoio do colégio Stella Maris, da cidade de Santos, que forneceu todos os dados necessários para os testes e sanou todas as dúvidas a respeito das necessidades da escola.

Este documento está disposto da seguinte forma: na seção 2 será feito o detalhamento do problema de agendamento, com as peculiaridades inerentes ao modelo brasileiro. Na

[1] Vide `http://goa.pos.ic.unicamp.br/otimo` para mais detalhes sobre o GOA

[2] Exemplos dessas técnicas são Programação por Restrições, Busca Tabu e Heurísticas GRASP

seção 3 serão descritas as características do algoritmo genético utilizado no projeto para a resolução do problema. A seção 4 contém os resultados obtidos nos trabalhos e uma comparação de desempenho e qualidade com relação a outros métodos utilizados por outros trabalhos semelhantes. Por fim, a seção 5 mostra a interface desenvolvida a fim de viabilizar o uso do sistema pelos usuários das escolas.

2 O Problema de Agendamento Acadêmico

O problema de *Agendamento Acadêmico* (*Timetabling*) consiste na alocação de um conjunto de aulas entre professores e alunos em um período de tempo pré-definido (tipicamente uma semana), satisfazendo restrições de vários tipos [SCH95]. Existem diversas variantes do problema, refletindo as várias formas com que diferentes tipos de instituições de ensino montam seus horários. Para escolas de nível médio e algumas faculdades temos na literatura o chamado *School Timetabling*, voltado para as características peculiares desse tipo de instituição [CL98], como por exemplo:

- agendamento feito por classes de alunos, e não de forma individual;
- grades de horários estruturadas e pré-definidas, muitas vezes com os locais de aulas já definidos;
- estudantes e professores com o máximo de ocupação possível durante o período em que se encontram na instituição;

A solução manual para o problema, que é a maneira como geralmente a maioria das escolas montam seus horários, pode consumir várias semanas de trabalho e nem sempre se obtem um resultado satisfatório. No caso das escolas no modelo brasileiro, existem poucas ferramentas computacionais disponíveis que realizam os agendamentos e que se adaptem às restrições específicas brasileiras [LRF01]. Este projeto visa construir uma ferramenta que possa ser utilizada pelas escolas de ensino médio e fundamental.

2.1 Definição Formal do Problema

Para enunciar uma descrição mais precisa do problema de agendamento acadêmico simplificado [SCH95], suponha um conjunto P com n professores $p_1, p_2, ..., p_n$, um conjunto C com m classes $c_1, c_2, ..., c_m$ e um conjunto H com t horários $h_1, h_2, ..., h_t$. A solução do problema é dada por uma matriz $R_{n \times m \times t}$ em que a entrada r_{ijk} vale 1 se o professor p_i dá aulas para a classe c_j no horário h_k, e r_{ijk} vale 0 caso contrário.

A disposição das aulas na solução R está sujeita a diversas restrições, distribuídas em duas categorias:

- **Restrições Fortes**: são aquelas que não podem ser violadas na resolução do problema, e sua presença na solução a torna inviável.
- **Restrições Fracas:** sua violação não é desejada na solução, mas é permitida. Um dos objetivos do algoritmo é minimizar a ocorrência das mesmas.

Como exemplos de restrições fortes, temos:

$$\sum_{j=1}^{m} r_{ijk} \leq 1, \text{ para todo } i = 1, ..., n, \; k = 1, \ldots, t \tag{1}$$

$$\sum_{i=1}^{n} r_{ijk} \leq 1, \text{ para todo } j = 1, ..., m, \; k = 1, \ldots, t \tag{2}$$

Significando, respectivamente, que a cada intervalo de tempo, um dado professor só pode ministrar aula para no máximo uma classe, e cada classe só pode ter aulas com no máximo um professor.

Para associar uma medida de qualidade à cada solução, definimos a função

$$f(p, c, h) : P \times C \times H \rightarrow \mathbf{R}$$

como sendo a penalidade incorrida caso o professor p dê aula para a classe c no horário h. Por exemplo, se o professor p_1 está indisponível em um horário h_1, então $f(p_1, c_j, h_1) > 0$, para todo $j = 1, 2, ...m$. O objetivo do algoritmo passa a ser então minimizar a função:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} f(p_i, c_j, h_k) r_{ijk} \tag{3}$$

Além dessas restrições, o problema real apresenta uma série de outras restrições que também devem ser tratadas de forma similar. A relação completa das mesmas será vista nas seções 2.2 e 2.3.

2.2 Restrições Fortes

Restrições Fortes são aquelas que não podem ser violadas para a geração de uma agenda usual. O conjunto de restrições fortes que foi abordado neste projeto é:

- **Preservar carga horária das classes**: cada série de ensino possui bem definida sua grade curricular, que não pode, em hipótese alguma, ser violada.
- **Preservar carga horária dos professores**: da mesma forma, a cada professor é atribuída, no início do período letivo, uma carga horária que deve ser mantida. Presume-se que a carga horária dos professores já vem alocada previamente, inclusive com as turmas. Essa é a situação real no caso do colégio tomado como modelo.
- **Preservar os períodos de início e término das aulas e intervalos**: a grade em que devem ser alocadas as aulas são pré-definidas, com os horários de início e fim de cada aula ou intervalo. Não podem existir aulas alocadas em horários não previstos nessa grade.
- **Não alocar aulas simultâneas**: não se pode violar o fato de que uma classe só pode assistir a uma aula e um professor só pode ministrar uma aula por intervalo de tempo.

- **Não deixar janelas para classes**: uma turma não pode ficar um horário ocioso aguardando por uma aula, a não ser no caso de intervalos pré-definidos.
- **Não deixar janelas para professores**: da mesma forma, um professor não deve ficar um período de tempo ocioso aguardando por uma aula. Essa restrição não é obrigatória em todas as instâncias do problema, pois existem escolas de maior porte em que não é possível realizar alocações sem causar janelas. Mas, em qualquer caso, é uma violação não desejável.
- **São proibidas aulas n-plas**: n aulas de uma matéria para uma mesma turma no mesmo dia, seguidas ou não. É proibido para qualquer matéria para $n > 2$. Aulas duplas ($n = 2$) só são permitidas para matérias com *mais* de 2 horas-aula por semana.
- **Preservar as indisponibilidades dos professores**: não devem ser alocadas aulas nos horários dos professores nos quais o mesmo indica estar incondicionalmente indisponível.
- **Manter aulas coordenadas num mesmo horário**: existem matérias que necessitam de coordenação para serem lecionadas, no sentido de que as aulas devem ocorrer no mesmo horário que as aulas de outra turma (da mesma matéria ou não). Exemplos de aulas coordenadas são as de educação física, que devem ser ministradas no mesmo horário mas com professores diferentes para masculino/feminino, e literatura/redação, que se alternam semana a semana.
- **Alocar apenas um professor por matéria nas turmas**: não pode ocorrer que mais de um professor lecione uma mesma matéria para uma mesma turma, mesmo que em horários distintos. Logo, a relação professor $\times$ (matéria,turma) é injetora.

2.3 Restrições Fracas

Restrições Fracas são aquelas cuja violação não compromete totalmente o resultado final da solução, sendo o objetivo do algoritmo minimizar a ocorrência das mesmas. Neste projeto, foram tratadas as seguintes restrições fracas:

- **Satisfazer os horários à preferência dos professores**: o professor pode indicar alguns horários onde *gostaria* de estar indisponível, sabendo que não necessariamente será atendido.
- **Distribuir aulas pela semana**: as aulas de uma mesma matéria devem ser eqüitativamente distribuídas através da semana.
- **Agrupar as aulas duplas:** as eventuais aulas duplas alocadas devem, preferencialmente, estar agrupadas.
- **Distribuir matérias de "baixo esforço mental"**: algumas matérias precisam estar distribuídas eqüitativamente através da semana. São consideradas pela escola matérias de "baixo esforço mental": Educação Artística, Computação, Educação Musical, Educação Física e Orientação Religiosa.

- **Satisfazer os dias livres de professores:** existem casos em que o professor manifesta o desejo de ter um dia - qualquer dia - livre na semana. O algoritmo deve tentar satisfazer essa restrição.

2.4 Modelagem do Problema

Uma vez definidas as características do problema, é necessário estabeler um modelo para o seu tratamento. Este modelo consiste em definir um esquema em que se encaixem todas as restrições e no qual seja possível refletir a situação real de agendamento.

Para a elaboração de um sistema de resolução do problema de Agendamento Acadêmico, estabeleceu-se uma ordem de prioridade nas restrições a serem tratadas. À medida que boas soluções foram sendo obtidas para determinado subconjunto de restrições do problema, novos parâmetros foram sendo acrescentados ao modelo até que as respostas fornecidas pelas soluções encontradas satisfizessem todas as necessidades do colégio tomado como base (colégio Stella Maris).

Nesta seção será apresentada uma modelagem simplificada do problema, assim como discutidas as modificações que devem ser feitas para que a mesma possa atender às necessidades das escolas de nível fundamental e médio brasileiras (em particular do colégio Stella Maris).

2.4.1 Modelagem Simplificada

Na primeira etapa do projeto, o objetivo foi estudar uma versão simplificada do problema de agendamento por meio de um algoritmo genético também simplificado para sua resolução. Após obtido sucesso com esse tratamento inicial, a modelagem do problema (e, por consequência, a implantação do algoritmo) foi adaptada para situações reais de uso.

Em qualquer uma das modelagens (simplificada ou final), um agendamento pode ser visto na forma de *tabelas*, onde são arranjados os professores, matérias e turmas nos horários disponíveis da semana letiva. O formato e os dados da tabela precisam ser dispostos de modo que todas as situações reais de alocação possam ser contempladas.

Analisando o problema de agendamento em sua forma simplificada, podemos modelá-lo em uma *grade única de alocação*, na qual todas as restrições seriam tratadas em apenas uma matriz de *turmas* por *horários*, na qual são dispostas as *matérias*[3]. Essa forma de trabalho acarreta as seguintes características:

Slots Uniformes : Os horários de aulas (doravante chamados de slots) correspondem a um período de tempo em que uma turma assiste a uma dada aula. No modelo simplificado, o problema é tratado como se todos os slots de aula ocorressem nos mesmos períodos de tempo, para todas as turmas. Ou seja, todas as primeiras aulas do dia começariam às 7:30, as segundas às 8:20 e assim por diante. No caso real, as turmas de diferentes séries podem vir a ter aulas em horários ligeiramente diferentes, ou mesmo possuir aulas com duração diferente.

[3]Não seria necessário especificar diretamente o professor, pois somente um professor pode lecionar uma mesma matéria para uma turma

Grupos de Séries : As séries do ensino fundamental e médio possuem algumas especificações diferentes em suas agendas. Essas especificações dizem respeito principalmente aos horários de início e fim dos slots de aula (vide item anterior) e à quantidade de aulas. À principio, o problema foi tratado apenas para o caso do ensino fundamental.

2.4.2 Modelo Final

Para que o problema real em toda sua complexidade pudesse ser tratado, foi necessário introduzir algumas mudanças estruturais na modelagem citada no ítem 2.4.1.

Para contornar os problemas de diferenças de horários nos slots das turmas e de quantidade de slots diferentes na semana letiva, foi criado o conceito de *multi-grade*. Nessa visão do problema não existiria apenas uma, mas n grades distintas em que são definidos os horários de aula de determinado conjunto de turmas com características semelhantes. Cada grade possui sua própria quantidade e horários de início de fim dos slots, de acordo com as turmas que a compõe.

Como exemplo, podemos citar o caso do ensino fundamental e médio: as turmas do primeiro grupo possuem 25 slots de aula na semana, 5 em cada dia útil; já as turmas do segundo grupo possuem 6 aulas diárias, totalizando 30 slots semanais. Mesmo os slots que seriam equivalentes no mesmo dia (como as primeiras aulas da segunda-feira, por exemplo) possuem horário de início e fim distintos, o que provoca conflitos entre alguns slots que seriam consecutivos se estivessem na mesma grade (como o primeiro slot da grade 1 e o segundo da grade 2, por exemplo). Isso impede que um mesmo professor lecione aulas nesses slots em conflito. Dessa forma, para tratar todo o conjunto de turmas, precisa-se de duas grades de horários, uma para as turmas do ensino fundamental e outra para as turmas do ensino médio. Além disso, precisa-se conhecer o conjunto de conflitos existentes entre os slots dessas grades, para impedir que em algum momento um professor precise lecionar para mais de uma turma ao mesmo tempo.

Esse conceito permite uma grande flexibilidade na definição do problema. Por exemplo, imaginemos que a escola deseja abrir turmas para cursos especializantes. Criando uma grade específica para essas novas turmas, podemos alocar aulas em quaisquer horários do dia, em quaisquer dias da semana, mesmo com quantidades de aulas diferentes para cada dia. Definindo um conjunto de conflitos com as demais grades, um professor não seria alocado para lecionar em slots sobrepostos.

2.5 Possíveis objetos de estudos futuros

Embora a nova visão do problema trate bem do caso das escolas de ensino fundamental e médio nos moldes brasileiros, algumas modificações podem ser feitas no modelo para que o mesmo se torne ainda mais abrangente. Alguns desses possíveis pontos de estudos, que não foram abordados no projeto, estão descritos a seguir.

2.5.1 Alocação de professores

O sistema poderia fazer também as alocações de horários dos professores para as aulas em cada turma, que a priori assume-se que vêm prontas. Dessa forma, seriam informadas apenas as competências de cada professor e o programa selecionaria quais séries e turmas seriam mais adequadas para o mesmo lecionar. É importante ressaltar que, no caso de mais de um professor poder ministrar uma mesma matéria para uma mesma série, cada turma deverá ter aulas daquela matéria com apenas um professor.

No caso de colégios nos moldes do Stella Maris[4] é premissa dos administradores que a alocação seja feita de forma manual. Isso porquê cada turma possui certas peculiaridades que podem exigir competências especiais dos professores - como experiência em lidar com turmas indisciplinadas. Os parâmetros para fazer esse tipo de alocação são de difícil modelagem e não foram abordados neste projeto.

2.5.2 Problema de Agendamento para Universidades

Uma variante do problema de Agendamento Acadêmico deste projeto lidaria com as restrições de escolas de nível superior. Nesse caso teríamos diversas modificações na definição do problema [SCH95, CL98], como por exemplo:

- alocações de aulas feitas por alunos e não por turmas;
- janelas nos horários, embora não desejadas, são permitidas;
- a capacidade das sala de aula e laboratórios deve ser levada em conta;

3 O Algoritmo Genético

Os *Algoritmos Genéticos* são uma classe de heurísticas evolutivas que, no decorrer das últimas décadas, têm sido bastante utilizadas na resolução de problemas de otimização e inteligência artifical.

Tais algoritmos utilizam idéias bastante peculiares, baseadas na teoria da evolução natural do *Neo-Darwinismo*. Deseja-se encontrar uma solução s para um problema de otimização P. A princípio, são criadas n soluções $s_1, s_2, \ldots, s_n$ para P. Cada solução s_i será chamada de *indivíduo* e todo o conjunto será chamado de *população* S. Partindo-se de uma população inicial S_0, cujos indivíduos podem ter sido gerados por métodos determinísticos ou aleatórios, o algoritmo simula um ciclo de evolução, em que uma nova população S_{i+1} é gerada a partir de uma população atual S_i através de processos de cruzamento de indivíduos. Também são realizados mutações e processos de seleção, para respectivamente garantir a diversidade da população e tentar privilegiar os indivíduos que são mais adaptados ao meio, ou seja, as melhores soluções para o problema.

É possível demonstrar, através de análises probabilísticas, que os indivíduos das população tendem a convergir para boas soluções do problema [WHI93]. Resta então definir

[4] Outros colégios entrevistados, como a "Escola Estadual de Ensino Fundamental e Médio João Franceschini", de Sumaré/SP, apresentaram esse problema da mesma forma

adequadamente todos os parâmetros do algoritmo, para que essa convergência ocorra de forma controlada e gere os resultados mais satisfatórios possíveis. O ciclo evolutivo termina quando um determinado critério de parada é atingido, como por exemplo um certo número de iterações sem obter melhores soluções para o problema ou um tempo máximo de processamento.

Para a construção de um algoritmo genético eficiente, vários pontos devem ser levados em consideração, como por exemplo: a estrutura de dados que representa os indivíduos (cromossomo), quais operadores genéticos serão definidos (como crossover e mutação), como os indivíduos serão avaliados, qual o processo de seleção a ser aplicado e qual o critério de parada do algoritmo.

Os *cromossomos* devem ser concebidos de forma a poder representar todas as soluções possíveis do problema com um mínimo de violações de restrições fortes. Os *operadores genéticos* definidos sobre os cromossomos são responsáveis pelos processos de recombinação de indivíduos e diversificação da população. A *função de avaliação* qualifica numericamente os indivíduos da população para fins de seleção dos indivíduos mais aptos de uma geração. Por fim, o *critério de parada* define quando o processamento do algoritmo deve ser encerrado e a melhor solução encontrada fornecida para o usuário. Todas essas características serão discutidas em detalhes em suas respectivas seções.

Vários pontos fazem dos GA's uma boa alternativa para se tratar problemas de otimização, tais como os de agendamento acadêmico. Dentre esses fatores estão a robustez do algoritmo e a flexibilidade para se definir restrições complexas do problema, fatores essenciais para permitir a implementação de uma ferramenta de qualidade para auxiliar na sua resolução.

3.1 Modelagem do Algoritmo no Projeto

Dificilmente encontramos na literatura dois Algoritmos Genéticos - ou, mais genericamente, Programas Evolutivos - com exatamente as mesmas características. Cada autor modela o seu sistema de acordo com as especificidades do problema que está tratando, buscando sempre os modelos que apresentam os melhores resultados em seu caso.

Vários parâmetros são importantes para se construir um algoritmo genético e há uma grande variedade de exemplos dos mesmos disponíveis na literatura. Baseando-se nesses exemplos e tendo sempre em mente as características do problema tratado, foi elaborado o Algoritmo Genético deste projeto.

Da mesma forma que na modelagem do problema de Agendamento Acadêmico (vide sessão 2.4), o algoritmo genético do projeto também foi desenvolvido partindo-se de um modelo simplificado, para depois sofrer as adaptações necessárias para abordar todas as características e restrições do problema. As modificações realizadas no GA e os motivos que levaram a determinadas implementações do algoritmo serão descritas nas sessões seguintes.

3.1.1 O Algoritmo Genético Canônico

O grande sucesso obtido pelos Algoritmos Geneticos na literatura teve inicio com o hoje chamado *Algoritmo Genético Canônico* [HOL75], que utilizava basicamente um cromossomo

no formato de string de bits e os operadores de crossover simples e mutação aleatória (vide sessões seguintes para mais detalhes sobre esses operadores).

Hoje, porém, muitas modificações têm sido propostas para o modelo inicial. Procura-se manter a idéia original de similaridades com os processos de evolução natural do neo-darwinismo, mas os detalhes de funcionamento do sistema são moldados para se obter mais eficiência na resolução dos problemas específicos.

As sessões seguintes descrevem os principais parâmetros de um Algoritmo Genético e fazem uma comparação do modelo adotado no projeto com o modelo original, evidenciando os porquês de tais escolhas.

3.2 Representação Interna do Cromossomo

Um algoritmo genético exige que seja utilizada uma estrutura de dados para representar internamente o cromossomo. Este, por sua vez, é a codificação de uma solução para o problema. Diversos modelos de representação podem ser encontrados na literatura para problemas de Agendamento Acadêmico utilizando Programas Evolutivos [SCH95, RCF95, CDM90c, BEW95]. Percebeu-se que as estruturas que apresentaram melhores resultados nos textos lidos seguem um mesmo padrão de modelagem, que serviu de base para a elaboração da estrutura usada neste projeto.

3.2.1 Modelo Simplificado

Para abordar o modelo simplificado do problema considerado na sessão 2.4.1, foi elaborada uma estrutura que satisfaz os requisitos para representar as soluções do problema de agendamento. Tomando um caso com n matérias, m classes e t horários de aula diferentes (somando os horários de todos os dias úteis da semana), o cromossomo possui as seguintes características:

- Representação dada por apenas uma matriz $M_{m \times t}$, na qual as linhas representam m turmas e as colunas t intervalos (slots) de aula.
- Cada entrada (i, j) da matriz corresponderia à matéria que seria lecionada para a turma i no horário j. Como vimos na sessão 2.4.1, no caso em questão, essa atribuição já estabelece diretamente quem é o professor.

As motivações para a escolha do modelo acima foram diversas [KER97]. Além do relativo sucesso em outros trabalhos semelhantes, ele é capaz de representar todas as soluções viáveis possíveis sem priorizar nenhuma delas. Das restrições fortes, são preservados o fato de uma classe nunca assistir a mais de uma aula simultaneamente, os horários fixos de início e fim das aulas e a ausência de horários vagos para classes. As demais restrições fortes podem ser tratadas pelo algoritmo, caso ocorra alguma violação.

Poucas das vantagens descritas acima seriam preservadas se fosse adotado o modelo de *string de bit's*, usado nos primeiros algoritmos genéticos desenvolvidos [HOL75]. Os cromossomos nesse modelo são representados por sequências de 0's e 1's, que codificam uma determinada solução. Os operadores genéticos atuam no cromossomo sem priorizar bits específicos, o que facilmente poderia gerar soluções corrompidas (inválidas).

3.2.2 Modelo Completo do Projeto

Para poder refletir o modelo de multi-grade descrito na sessão 2.4.2 foram necessárias algumas mudanças no cromossomo, como descrito a seguir:

- Como uma matriz apenas não seria mais suficiente para refresentar toda a estrutura, foi necessário definir que cada cromossomo conteria n matrizes, correspondentes a cada uma das n grades. Cada uma dessas matrizes teria as mesmas características da matriz descrita no modelo simplificado, mas conteria apenas informações das turmas que se encaixam em seu padrão de horário.

- Para facilitar a verificação de sobreposições de aulas de professores, foi necessário criar uma matriz extra, chamada de *matriz geral*. Dados n' slots e p professores para o problema geral, seria uma matriz $M'_{p \times n'}$, em que n' é um conjunto de slots tal que qualquer slot de qualquer grade coincide com um ou mais slots da matriz geral. Cada célula dessa matriz informa se o professor correspondente está lecionando no slot indicado.
  Por exemplo: se existir um caso trivial de uma grade com apenas um slot das 7:00 às 7:50 e outra grade com apenas um slot das 7:30 às 8:20, ambos na segunda-feira, a grade geral conterá três slots: um das 7:00 às 7:30, outro das 7:30 às 7:50 e outro das 7:50 às 8:20, todos na segunda. Com isso, cada slot das outras grades se encaixa na grade geral sobrepondo-se a dois slots.

Com as modificações acima, cada movimento realizado dentro de um cromossomo (como mutação ou *crossover*) em determinada grade deverá ser realizado também na grade geral, nas células que coincidem com aquelas onde ocorreu o movimento. Com isso, ao se fazer uma alteração, é possível verificar se determinado horário já estava sendo ocupado pelo professor ou não, independente da grade.

3.2.3 Exemplo de Representação

Supondo um caso simples em que existem três turmas, duas do ensino fundamental e uma do ensino médio, tendo as primeiras 5 aulas na semana e a segunda 6, somente na segunda-feira, a representação do cromossomo seria:

Grade 1

Classe	Horários				
	seg/1 7:00-7:30	seg/2 7:30-8:00	seg/3 8:00-8:30	seg/4 8:30-9:00	seg/5 9:00-9:30
Turma 1	Mario/Port	Mario/Port	João/Hist	Maria/Geog	José/Matem
Turma 2	João/Hist	José/Matem	Mario/Port	Mario/Port	Maria/Geog

Grade 2

Classe	Horários					
	seg/1 7:00-7:30	seg/2 7:30-8:00	seg/3 8:00-8:30	seg/4 8:30-9:00	seg/5 9:00-9:20	seg/6 9:20-9:40
Turma 3	José/Matem	João/Hist	Maria/Geog	José/Matem	Mario/Port	Mario/Port

Grade Geral

Professor	Horários						
	seg/1 7:00-7:30	seg/2 7:30-8:00	seg/3 8:00-8:30	seg/4 8:30-9:00	seg/5 9:00-9:20	seg/6 9:20-9:30	seg/7 9:30-9:40
Mario	Aula	Aula	Aula	Aula	Aula	Aula	Aula
João	Aula	Aula	Aula	Folga	Folga	Folga	Folga
Maria	Folga	Folga	Aula	Aula	Aula	Aula	Folga
José	Aula	Aula	Folga	Aula	Aula	Aula	Folga

Para representar esse horário internamente, será preciso transcrever para códigos numéricos as identificações das turmas, professores e matérias. Esse trabalho pode ser feito ainda na interface, que já transmitiria para o núcleo do algoritmo genético os registros codificados. Isso permite que o programa trabalhe de forma computacionalmente mais rápida.

Também serão necessários alguns vetores extras para armazenamento de informações secundárias (como penalidades por linha, por professor, etc). Esses vetores serão apresentados posteriormente quando forem descritos os operadores genéticos que os utilizam.

3.3 Função de Avaliação

Outro ingrediente essencial de um algoritmo genético é a *função de avaliação*, que qualifica numericamente os indivíduos da população. Uma função de avaliação de penalidades é capaz de, dado um cromossomo de entrada, retornar um valor numérico inteiro t que representa a soma das penalidades do indivíduo, sofridas devido à violação de restrições do problema. O valor de t será tão maior quanto pior for a solução. É através da função de avaliação que o Algoritmo Genético compara cromossomos diferentes, para escolher quais indivíduos serão selecionados e se multiplicarão em cada geração.

As principais características de uma função de avaliação eficiente são:

- **Incorporação de todas as restrições do problema:** devem ser incorporadas na função de avaliação todos os tipos de ocorrência que merecem penalidades em uma possível solução do problema. Um exemplo aplicado sobre o problema de Agendamento Acadêmico seria o fato de que não é permitido um professor dar aulas em horários em que o mesmo está indisponível. Caso a função de avaliação receba um cromossomo cuja solução representada viole esta restrição, a penalidade do indivíduo em questão sofrerá um incremento[5].

- **Penalidades diferenciadas de acordo com a restrição:** geralmente, o algoritmo genético trata de problemas com mais de uma restrição. Neste caso, a função de

[5] O valor deste incremento varia de acordo com a restrição violada, e será discutido posteriormente na sessão 4.2

avaliação deve ser balanceada de modo que aplique as penalidades coerentemente com a gravidade do caso. Isto é, violações graves sofrem altas penalizações e violações simples sofrem penalizações menores do que no primeiro caso, fazendo com que indivíduos mais viáveis tenham maiores condições de se propagar na população. Deve-se ter em mente também que, como o processo de seleção é probabilístico, penalidades muito semelhantes para violações diferentes possuem praticamente o mesmo efeito sobre o processo.

- **Desempenho:** é importante também que o código da função de avaliação possa ser executado rapidamente pelo computador onde o algoritmo está sendo utilizado. Como todos os indivíduos que surgem durante o processo de evolução precisam ser avaliados, a função é extremamente utilizada; e um mau desempenho na mesma pode acarretar em lentidão no processamento do algoritmo como um todo.

3.3.1 Função de Avaliação × Função de Fitness

Uma técnica também muito utilizada, tratando-se de algoritmos genéticos, é a utilização de uma *Função de Fitness* ao invés de uma Função de Avaliação de Penalidades. O valor retornado pela primeira é inversamente proporcional ao da segunda, ou seja, o indivíduo está mais apto para continuar na população quando seu assim chamado *fitness* for maior do que o dos demais.

$$fitness \propto \frac{1}{penalidade}$$

Uma desvantagem desse processo é a necessidade de se trabalhar geralmente com números de ponto flutuante - já que o cálculo do fitness envolve uma divisão - o que penalizaria o processamento computacional do algoritmo. Outro problema que surge é quando indivíduos com penalidades razoavelmente diferentes apresentaram valores de fitness muito parecidos, também devido à natureza do cálculo deste valor [BBM93]. Por esses motivos, optou-se por trabalhar diretamente com penalidades, de modo que o objetivo do algoritmo passe a ser *minimizar* o valor das penalidades, e não *maximizar* o valor do fitness. Os processos de seleção dos indivíduos também foram adaptados para tal comportamento.

3.3.2 Tratamento de Restrições pela Função de Avaliação

Para o bom desempenho do algoritmo genético do projeto, precisa-se estabelecer *quais* restrições serão tratadas pela função de avaliação e *como* as mesmas serão calculadas a partir dos cromossomos. É importante ressaltar que a função, neste caso específico, não precisará considerar todas as restrições do problema, já que a própria modelagem do cromossomo não permite que muitas das restrições sejam violadas (vide sessão 3.2).

Tratamento de Restrições Fortes

Apesar da violação de muitas restrições fortes do problema serem inviabilizadas pela própria estrutura do cromossomo, como no caso da restrição de carga horária das classes

e dos professores, algumas restrições ainda são passíveis de serem inflingidas. Um exemplo claro é o fato de um professor dar aula para duas classes distintas num mesmo intervalo de tempo. Neste caso, a função de avaliação precisará aplicar as penalidades devidas ao cromossomo, o que comprometerá sua propagação na população.

É citada na literatura a aplicação de altas penalidades para violações de restrições fortes. Entende-se por altas penalidades um valor que seja bem maior do que aqueles comumente aplicados a restrições fracas, dependendo do caso diferindo por várias ordens de grandeza (uma espécie de "valor infinito").

Este método possui vantagens e desvantagens. A principal vantagem é o fato de que um cromossomo que seja absurdamente penalizado terá estatisticamente inviabilizada sua reprodução e continuidade na população.

Uma desvantagem é que, numa população onde a grande parte dos indivíduos violam restrições fortes, será altamente beneficiado o primeiro cromossomo que se formar sem violar esse tipo de restrição. Como o mesmo possuirá um valor total de penalidade muito menor do que o dos demais, ele se multiplicará em grande número e a solução média final da população tenderá a convergir para algo semelhante a este indivíduo, que não necessariamente é uma boa solução. Esse processo é chamado de *convergência prematura*.

Outra desvantagem é a redução do grau de liberdade do algoritmo. Caso as populações evitem (mesmo que estatisticamente) a existência de indivíduos que violem restrições fortes, o espaço de busca reduz-se, diminuindo a diversidade dos indivíduos e, consequentemente, a probabilidade de eventuais mutações e crossover gerarem bons novos cromossomos, partindo-se de indivíduos que à princípio eram inviáveis.

Uma alternativa para se contornar os problemas acima seria penalizar mais brandamente as violações de restrições fortes e aplicar operadores genéticos especiais para tentar corrigir as mesmas. Podem ser mutações que, quando aplicadas sobre um cromossomo, promovem alterações justamente nos trechos em que existem violações graves. É importante deixar claro que esse processo é diferente de alguns métodos utilizados que *restringem* o espaço de busca do algoritmo [CDM90b, CDM90c]. Nesse caso, o algoritmo continuará a ter liberdade para vasculhar todos os tipos de soluções, mas tenderá evitar as violações de restrições fortes com o passar das gerações. Vide mais detalhes sobre mutações corretivas na sessão 3.4.2.

Tratamento de restrições fracas

O tratamento de restrições fracas é feito de forma mais natural na função de avaliação, devendo-se estabelecer para isso uma escala de penalidades para cada tipo de restrição. Todavia, um fato importante é a necessidade de se aplicar penalidades bem distintas para violações de gravidades diferentes, de modo que haja uma maior probabilidade de se priorizar a evolução dos indivíduos que possuem violações menos graves. Vide os valores de penalidades aplicados neste projeto na sessão 4.2.

Para penalizar a distribuição das chamadas aulas de "baixo esforço mental", foi utilizado um conceito de *níveis de dificuldade de uma matéria*, parâmetro esse informado pelo usuário no momento da entrada de dados. Uma determinada solução é penalizada de acordo com o acúmulo num mesmo dia de matérias de níveis de dificuldade altos.

3.4 Operadores Genéticos

Um algoritmo genético baseia-se no fato de que um cromossomo, que é a representação interna de uma solução para o problema tratado, sofre diversas modificações e mesclagens, de modo que o indivíduo que ele representa tenha uma probabilidade de "evoluir" para um solução melhor, de acordo com critérios de avaliação pré-definidos. Os indivíduos que formam uma população devem então ser selecionados de acordo com sua qualidade, para que a próxima geração tenha, ao longo do tempo, uma carga de violações de restrições menor do que a anterior.

Para que ocorra esse processo de evolução, é necessário aplicar alguns *operadores genéticos* sobre os cromossomos. Esses operadores tratam de modificar os indivíduos da população através de mutações e mesclagem de informações e são parte importante do algoritmo genético.

A seguir estão descritos os operadores que foram utilizados para a solução do problema de Agendamento Acadêmico neste projeto. Esses operadores foram baseados em vários modelos encontrados na literatura [SCH95, RCF95, CDM90c, BEW95, REE97], uma vez readaptados para as novas estruturas de dados do cromossomo.

3.4.1 Operadores de Crossover

Os operadores de crossover são aplicados sobre mais de um indivíduo da população (geralmente 2, que são chamados de progenitores), mesclando informações de cada um deles para formar um ou mais novos indivíduos. Durante um processo de crossover deve-se evitar mesclar dados de modo que o resultado viole as restrições fortes do problema, o que nem sempre é possível na prática. Existem diversas formas de se fazer essa mesclagem de informações, sendo algumas delas descritas a seguir.

Crossover de n pontos

Os operadores de crossover de n pontos tradicionais trabalham geralmente com dois cromossomos progenitores na forma de vetores de tamanho fixo e retornam outros dois cromossomos novos. Executam o seguinte processo:

- Escolhem-se aleatoriamente n pontos distintos $p_1, p_2, ..., p_n$, $n \geq 1$, de tal modo que o valor de um ponto qualquer seja menor do que o tamanho do vetor que representa os cromossomos. O valor de n pode ser fixo ou variável, sendo dois casos muito utilizados os crossovers de 1 e de 2 pontos.

- Separa-se os cromossomos progenitores em $n+1$ pedaços, usando-se os mesmos pontos de corte. Para facilitar a compreensão, pode-se numerar os pedaços de cada cromossomo seqüencialmente.

- Serão formados dois cromossomos neste processo; o primeiro deles conterá os pedaços pares de um dos progenitores e os pedaços ímpares do outro. O segundo conterá os pedaços ímpares do primeiro e pares do segundo.

Um exemplo de aplicação do crossover de 2 pontos aplicado sobre um vetor binário (composto de 0s e 1s) está abaixo:

```
Cromossomo progenitor 1   001101001011100
Cromossomo progenitor 2   010111011001111
Pontos escolhidos            |         |
Cromossomo resultante 1   001111011001100
Cromossomo resultante 2   010101001011111
```

Mas da forma como é concebido, esse operador aplicado ao problema de agendamento acadêmico do projeto apresenta vários problemas. Primeiramente, a representação de cromossomo escolhida não é um vetor, e sim uma matriz. Pode-se optar tanto por "linearizar a matriz" - colocando todas as linhas seqüencialmente na forma de um vetor - quanto por aplicar o operador em linhas específicas. Mas, de qualquer modo, não é garantido que o os cromossomos resultantes do processo não tenham violado algumas restrições fortes, como por exemplo manter a carga horária dos professores e das turmas. Dessa forma, esse operador não será aplicado dessa maneira ao problema.

Uma maneira de se resolver esses problema é fazer a separação dos progenitores de tal maneira que não seja violada a carga horária das turmas e a alocação das matérias. Isso mantém a característica principal do operador, que é aproveitar boas alocações de aulas e tentar novas combinações envolvendo essas alocações.

Algoritmicamente, podemos executar seguinte processo [BBM93]:

1. Escolhe-se n slots nos cromossomos progenitores.

2. Em um dos progenitores, seleciona-se as n matérias lecionadas nos slots escolhidos e transfere-se as mesmas em suas posições originais para o cromossomo resultante.

3. As demais matérias serão alocadas no descendente seguindo a mesma ordem do segundo cromossomo progenitor, respeitando as posições já alocadas.

Esse processo deve ser aplicado individualmente à cada linha de cada matriz, que representa os horários de uma turma. Escolhe-se dois cromossomos e aplica-se o operador para cada par de turmas equivalentes.

As vantagens desse processo são várias. Primeiramente, não são violadas as cargas horárias das turmas e professores. Isso porque a quantidade de aulas é mantida, assim como a alocação dos professores, variando-se apenas as disposição dos horários. Além disso, características positivas de cada progenitor são mantidas, ou seja, matérias bem alocadas em uma solução poderiam ser preservadas na geração seguinte. Apesar de parecer bem diferente de um crossover de n pontos tradicional, o novo processo preserva a essência do mesmo: juntar partes de cromossomos diferentes, tentando obter características de cada um deles para gerar um descendente.

Crossover uniforme

Outro operador de crossover muito utilizado é o crossover uniforme. É comumente aplicado a 2 cromossomos representados por vetores, resultando em um descendente. Apresenta a seguinte lógica:

> Para cada posição do cromossomo descendente a ser gerado, é associado um número inteiro aleatório dentre 0 e 1. Se determinado número for 0, a posição correspondente do vetor será ocupada pelo valor equivalente do primeiro progenitor; se for 1, pelo valor equivalente do segundo.

Veja abaixo um exemplo de crossover uniforme aplicado a dois cromossomos em forma de vetores de caracteres:

Cromossomo progenitor 1	`abcdefghijklmnop`
Cromossomo progenitor 2	`ABCDEFGHIJKLMNOP`
Números aleatórios escolhidos	`0011110110010101`
Cromossomo resultante	`abCDEFgHIjkLmNoP`

Muitas variações, utilizando vários progenitores ou gerando mais de um descendente ao mesmo tempo, podem ser aplicadas. Mas, para este projeto, o operador em sua forma tradicional inflingiria algumas restrições fortes, como a carga horária dos professores e das classes. Dessa maneira, foi necessário seguir outra lógica de funcionamento:

1. Para cada linha de cada grade do cromossomo descendente - ou seja, para cada turma - escolhe-se a turma equivalente menos penalizada entre os dois progenitores.

2. O descendente terá na linha correspondente a mesma alocação de horários de turma selecionada.

Para utilização deste operador, é necessário que o cromossomo tenha um vetor auxiliar indicando qual a penalidade de cada turma. Com isso, facilmente pode-se escolher a turma com menor penalidade local para o descendente, ou seja, com menor violação de indisponibilidades, aulas n-plas, preferências de indisponibilidades, etc. Outras restrições podem vir a ser violadas no descendente, como aulas simultâneas e janelas, mas verifica-se na prática que os descendentes apresentam, em geral, menor penalidade do que seus progenitores.

Ficou definido que, neste projeto, o crossover uniforme trabalhará com dois progenitores, gerando apenas um descendente. Dessa forma, mais recombinações, envolvendo mais indivíduos, serão feitas, buscando-se ampliar a diversidade da população.

3.4.2 Operadores de Mutação

Os operadores de mutação são aplicados sobre um indivíduo da população. Escolhido aleatoriamente, um cromossomo sofre pequenas alterações, geralmente também aleatórias, e o cromossomo resultante continua no processo de evolução.

O objetivo dos operadores de mutação é preservar a diversidade da população. Para um algoritmo genético apresentar um bom desempenho, depende-se muito das diferentes

propostas de solução que cada um dos indivíduos representa, que serão trabalhadas durante o processo evolutivo e permitirão que soluções melhores possam surgir. Quando todos os indivíduos forem muito parecidos, os operadores de crossover passam a não surtir mais efeito e o algoritmo vem a estagnar. Esse processo de uniformização da população, que ocasiona convergências prematuras, é natural em um algoritmo genético mas pode ser controlado pela aplicação das mutações.

Porém, se aplicado de forma descontrolada, o operador mutacional pode não surtir os efeitos práticos desejados. Se forem aplicadas mutações em quantidades muito pequenas de indivíduos, as modificações praticamente não serão sentidas na população como um todo e nenhum ganho é obtido. Pelo contrário, se for aplicado de forma abusiva, há uma grande probabilidade de se corromper as boas soluções que vão sendo encontradas, chegando-se ao extremo de transformar o algoritmo em uma busca aleatória.

Dessa forma, os operadores de mutação dependem do balanceamento de um parâmetro numérico real t, $0 < t < 1$, que diz qual a taxa de mutação que será usada em cada gene do cromossomos. Detalhes sobre os parâmetros usados neste projeto podem ser vistos na sessão 4.2.

A seguir serão listados os operadores de mutação que foram aplicados neste projeto para o problema de Agendamento Acadêmico.

Mutação de horários

Este operador parte do princípio que dois horários (slots) de aula de uma mesma classe, seguidos ou não, podem ser trocados. Nesse caso, serão mantidas as cargas horárias, correndo-se apenas o risco de violar as compatibilidades de horários dos professores.

O que esse operador faz é, dado um cromossomo, trocar duas células escolhidas aleatoriamente em uma mesma linha da matriz, que equivalem a dois slots de uma mesma turma.

Mutações corretivas

A natureza do problema de Agendamento Acadêmico deste projeto impõe o tratamento de muitas restrições fortes (vide sessão 2), o que restringe muito o espaço disponível de soluções viáveis. Os algoritmos genéticos em geral não são muito eficientes para trabalhar nesse tipo de ambiente, sendo então necessário buscar novas técnicas para que seu desempenho possa ser satisfatório.

Algumas maneiras de se lidar com essa situação são encontrada na literatura, como por exemplo restringir o espaço de busca apenas para o campo das soluções viáveis [CDM90b, CDM90c], ou aplicar penalidades muito altas para violações de restrições fortes. Mas ambas as formas apresentam problemas, envolvendo principalmente a diminuição do grau de liberdade do algoritmo genético. A forma encontrada nesse projeto para tratar das restrições fortes foi a aplicação de *mutações corretivas* nos cromossomos. Nessa técnica, o algoritmo e os operadores genéticos de crossover têm liberdade para trabalhar com todas as soluções possíveis, inclusive com soluções que violem restrições fortes, sendo essas violações penalizadas normalmente. Mas são aplicadas pequenas mutações especializadas nos locais em que foram violadas restrições fortes. Elas visam, com o passar das gerações, inibir

gradativamente a existência de tais violações, convergindo então a população para soluções viáveis.

Neste projeto foram criadas rotinas de mutação corretiva para as principais restrições fortes do problema, sendo que a violação das demais são evitadas pela própria estrutura do cromossomo. As restrições fortes que contaram com correções foram:

- aulas em horários simultâneos para um mesmo professor;
- violação das indisponibilidades dos professores;
- janelas para professores;
- aulas n-plas para turmas;
- aulas coordenadas para turmas.

Essas mutações corretivas consistem, basicamente, na troca de slots de aula de uma determinada turma, sempre em pontos de violação de restrições fortes. São aplicadas aleatoriamente nos indivíduos da população que violam alguma dessas restrições, com uma determinada taxa de incidência t. Ou seja, nem todos os indivíduos sofrem as correções de uma só vez, mas sim tendem a sofrê-las com o passar das gerações. Isso garante a liberdade do algoritmo para explorar todo o espaço de busca.

Podemos considerar duas formas de aplicar as mutações corretivas: uma delas seria a mais branda, consistindo meramenta na aplicação de uma correção qualquer ao invés de uma mutação aleatória. A outra, mais agressiva, seria utilizar as mutações mais fortemente nos indivíduos, aplicando correções para todos os tipos de violações no mesmo indivíduo de uma só vez para tentar eliminar o máximo possível de violações. As duas formas de aplicação foram testadas e os resultados estão na sessão 4.

Após a utilização das rotinas de mutação corretiva, o algoritmo apresentou notável melhora de desempenho se comparado com um algoritmo que apenas penalisa de forma elevada as violações de restrições fortes.

3.5 Processos de Seleção de Indivíduos

O desempenho de um algoritmo genético também depende muito da forma com que os indivíduos da população são selecionados para constituir a geração seguinte. Os processos de seleção devem balancear a *pressão de seleção* contra a *liberdade de exploração*, para que a evolução ocorra e a diversidade da população seja mantida.

Ou seja, não se pode impor apenas uma alta pressão evolutiva para selecionar os melhores indivíduos de uma população, pois a diversidade das gerações seguintes ficará comprometida e o algoritmo convergirá prematuramente. Também não se deve perder bons indivíduos encontrados até certo momento, para não se prejudicar a qualidade das novas gerações.

Estão indicados abaixo os processos de seleção utilizados no algoritmo, e os porquês de sua escolha para o projeto.

3.5.1 Seleção Elitista

Este operador recebe como parâmetro uma população com N indivíduos, uma função de avaliação e um número n, sendo $n < N$. Retorna os n indivíduos da população com menor valor de penalidade. Esses indivíduos farão necessariamente parte da próxima geração.

O sucesso deste processo depende da escolha do número n. Este número está intimamente relacionado com o valor de N, de modo que ambos devem ser bem balanceados. Se n for muito grande com relação a N, haverá pouca variação de indivíduos de uma geração para outra. Em contrapartida, se n for muito pequeno, pode-se perder bons resultados que se obtém durante o processo.

O balanceamento ideal de n foi obtido no projeto após vários testes. Vide detalhes sobre os valores utilizados na seção 4.2.

3.5.2 Seleção por torneio

Os indivíduos que não foram selecionados pelo processo elitista participarão de um procedimento que completará os cromossomos da geração seguinte. A assim chamada *seleção por torneio* [WHI93, BBM93] possui a seguinte lógica:

1. De uma população que será substituída, escolhem-se m indivíduos aleatoriamente com probabilidade uniforme.

2. Os indivíduos escolhidos participarão de um "torneio", no qual será vencedor aquele com menor valor de penalidade. O indivíduo vencedor do torneio fará parte da próxima geração.

3. Repete-se o processo tantas vezes quantas forem necessárias para completar os indivíduos da geração seguinte.

Existem diversas variantes deste processo, mas que seguem sempre a mesma idéia. Em alguns casos, permite-se até que os indivíduos perdedores componham a próxima geração, mas com uma probabilidade menor. Também é bastante diversificado o valor de m utilizado na escolha da quantidade de indivíduos de cada torneio. É importante salientar que esse número está relacionado com a pressão de seleção do processo, sendo que quanto maior for m, mais vezes os melhores indivíduos têm a probabilidade de participar de um torneio, serem vencedores e constituirem a próxima geração. As instâncias mais comuns de seleção por torneio utilizam um m de 2 [BBM93], o que foi também seguido neste projeto. Neste caso, testes com valores maiores causaram um aumento na quantidade de convergências prematuras.

Outro fato importante a ser considerado é a modificação na lógica do algoritimo genético utilizado no projeto (vide seção 3.6). Na ordem utilizada para aplicação dos operadores genéticos, os torneios são realizados em populações que incluem indivíduos progenitores e seus descendentes.

3.6 A lógica do Algoritmo Genético

Basicamente, um algoritmo genético tradicional funciona da seguinte forma: [BBM93]

```
/* Algoritmo Genetico */
INICIO
   /* Inicialização do Sistema */
   Gerar População Inicial
   Computar penalidades de cada indivíduo

   /* Ciclo Evolutivo */
   ENQUANTO não terminou
   INICIO

      REPETIR tamanho_da_população VEZES
      INICIO
         /* Priorizar indivíduos de menor penalidade */
         Selecionar dois individuos da população antiga
         Recombinar os individuos para gerar um descendente
         Inserir descendente na nova geração
      FIM

      Aplicar mutação na nova população

      /* Critério de parada */
      SE população convergiu ENTÃO
         terminou = sim
   FIM
FIM
```

Mas durante a fase de testes do projeto, foram feitas diversas experiências com outras formas de execução do algoritmo, procurando-se variações nos parâmetros, nos valores de penalidades e, inclusive, na própria ordem de execução do GA. Para o problema em questão, melhores resultados (com menor penalidade média final) foram obtidos utilizando-se a seguinte forma[6]:

```
/* Algoritmo Genetico */
INICIO
   /* Inicialização do Sistema */
   Gerar População Inicial
   Computar penalidades de cada indivíduo

   /* Ciclo Evolutivo */
```

[6] Vide seção 4 para mais detalhes sobre os resultados obtidos

```
   ENQUANTO não terminou
   INICIO

      /* Criação de um pool de descendentes */
      REPETIR tamanho_do_pool VEZES
      INICIO
         Selecionar aleatoriamente dois progenitores da geração atual
         Recombinar os individuos para gerar um descendente
      FIM

      /* Processo de seleção */
      REPETIR tamanho_da_população VEZES
      INICIO
         /* Priorizar indivíduos de menor penalidade */
         Selecionar um individuo da população antiga ou do pool
         Inserir individuo na nova geração
      FIM

      Aplicar mutação na nova população

      /* Critério de parada */
      SE população convergiu ENTÃO
         terminou = sim
   FIM
FIM
```

Esta forma de execução é basicamente a mesma do algoritmo genético tradicional [HOL75, MIC96, WHI93, BBM93, YS02], mas com a ordem de aplicação do processo de seleção e dos operadores genéticos de crossover invertida. Dessa forma, primeiramente são gerados os descendentes de uma população, que junto com seus progenitores constituem um *pool de indivíduos*, onde todos os indivíduos participarão dos processos seletivos.

Dois pontos podem sugerir a melhora do desempenho do algoritmo genético sob sua nova forma:

1. Na nova forma, o algoritmo impõe uma pressão de seleção maior sobre os indivíduos da população, já que os descendentes gerados que não são muito adaptados possuem poucas chances de continuarem na população.

2. Os operadores genéticos de crossover aplicados sobre o modelo de cromossomo do projeto têm elevada capacidade de produzir descendentes corrompidos. Ou seja, a probabilidade de um indivíduo gerado violar alguma restrição de aulas simultâneas é muito alta. A nova ordem de execução do algoritmo dificulta estatisticamente a seleção de indivíduos corrompidos, selecionando as melhores recombinações feitas na população.

3.7 Demais Parâmetros do GA

Além dos parâmetros citados nas sessões anteriores, um algoritmo genético possui outros fatores a serem considerados. Segue uma breve descrição dos mesmos:

Tamanho da população: esse valor está relacionado com diversos fatores, desde a capacidade do algoritmo genético de explorar nichos diferentes até o desempenho da execução do programa.

Critério de Parada: estabelece o critério para que o algoritmo cesse o processamento. Geralmente está associado com um período de estagnação da evolução. No caso deste projeto, o algoritmo encerra após processar um grande número de gerações sem encontrar melhores indivíduos.

Taxa de aplicação de operadores de mutação: é preciso estabelecer a quantidade de indivíduos da população em que serão aplicados os operadores genéticos de mutação. Neste projeto, optou-se por utilizar taxas variáveis. Ou seja, estabelece-se um valor mínimo v_{min}, um valor máximo v_{max} e um incremento i para cada operador de mutação. De acordo com o operador, o algoritmo começará aplicando o mesmo em v_{min} indivíduos, valor incrementado de i a cada nova geração até chegar a v_{max}, quando passa a ser constante; ou começará aplicando em v_{max} indivíduos, decrementado de i a cada nova geração até chegar a v_{min}, quando passa a ser constante.

Os resultados dos testes que levaram à escolha dos parâmetros acima estão na seção 4.

3.8 Possíveis objetos de estudos futuros

Embora o algoritmo do projeto, da forma com que foi implementado, atenda de maneira satisfatória ao problema de Agendamento Acadêmico tratado, novos recursos podem ser futuramente incorporados ao mesmo, de modo que a ferramenta final torne-se ainda mais eficiente na resolução do problema. Alguns desses possíveis pontos de estudos futuros estão descritos a seguir.

Atendimento dos requisitos da sessão 2.5

Na sessão 2.5 foram discutidos possíveis objetos de estudos futuros no que diz respeito à modelagem do problema. Para que o tratamento desses pontos - como alocação de professores e agendamento nos moldes das universidades - seja possível, é preciso que o algoritmo esteja adaptado ao novo modelo.

Evolução Parcial

Um recurso que poderia ser implementado seria um algoritmo que interagisse com o usuário durante o processo evolutivo. Por exemplo, o usuário solicita um agendamento, fornecendo uma série de dados para o algoritmo. Depois de alguns minutos de evolução, o processo seria interrompido, a solução atual seria apresentada aos usuários e o processamento continuaria, preservando eventuais seleções feitas pelos mesmos. Com isso, se o

usuário não ficar satisfeito com determinadas alocações de aulas, pode solicitar ao algoritmo um realocamento.

Esse recurso não necessita de mudanças de modelagem para ser implementado, mas necessitaria de uma série de modificações no código do programa e da interface de uso para poder lidar com as interações.

4 Resultados obtidos

Nesta seção serão apresentados detalhes qualitativos e quantitativos a respeito dos resultados obtidos com o algoritmo genético implantado no projeto. Serão mostrados exemplos dos parâmetros utilizados no algoritmo e de soluções encontradas, além de serem feitas comparações com os resultados de outras heurísticas que abordaram o mesmo problema em outros projetos do GOA.

4.1 Metodologia de Testes

Como um algoritmo genético é baseado em processos probabilísticos, para se poder levantar conclusões mais precisas sobre seu funcionamento é preciso realizar vários testes sobre uma mesma instância do problema usando os mesmos parâmetros de configuração. Pode-se, dessa forma, obter uma *média* do desempenho do programa para cada contexto de aplicação.

Nos testes realizados no âmbito deste projeto foi seguida a seguinte metodologia:

1. Cria-se um arquivo de parâmetros com as configurações do programa que devem ser testadas. Esse arquivo contém dados como tamanho da população, taxa de aplicação de operadores genéticos e valor das penalidades, dentre outros.

2. Executa-se o algoritmo genético baseado nestes dados pelo menos 10 vezes, gerando-se saídas com a penalidade do melhor indivíduo e a penalidade média da população, para cada geração.

3. Cria-se um arquivo de saída geral com as mesmas características citadas anteriormente, mas contendo agora a média dos 10 testes realizados. Estes serão os dados a serem analizados.

4. Calcula-se também a média de penalidade dos resultados finais gerados pelas execuções, assim como seu tempo de processamento.

Tal procedimento deve ser realizado para cada variante do arquivo de configuração. Dessa forma, se o objetivo é testar um determinado parâmetro do algoritmo (como tamanho da população, por exemplo), deve-se construir vários arquivos de configuração com as variações desejadas no parâmetro e executar o processo de testes para cada um deles. Com isso, pode-se analizar o processo evolutivo e chegar à conclusões sobre a eficiência e estabilidade do algoritmo sob as diversas condições.

É importante ressaltar que todos os testes foram realizados sobre dados reais fornecidos pelo colégio Stella Maris para o ensino fundamental e médio. O programa foi executado em

um PC Intel com processador Pentium Xeon de 2,5 GHz, um ambiente possível para uso do programa.

4.2 Parâmetros para Obtenção dos Melhores Resultados

Após inúmeros testes realizados com os mais diversos parâmetros, encontrou-se uma configuração que obteve relativamente os melhores resultados. Tratando todas as restrições do problema (vide seção 2), o algoritmo consegue encontrar soluções que em geral não violam as restrições fortes e eliminam grande parte das violações de restrições fracas.

Em nenhuma situação são geradas soluções com violações que o usuário não possa corrigir manualmente via interface, com as condições de contorno que achar mais adequadas. Exceto em casos de convergência prematura, quando o algoritmo retorna uma solução com excessivas violações de restrições fortes. Mas tais casos representaram menos de 20% do total de soluções para a instância do problema utilizada (fornecida pelo colégio Stella Maris). Nesses casos, o usuário deverá executar o algoritmo novamente para obter uma solução de melhor qualidade.

Os melhores resultados gerados foram obtidos utilizando-se as configurações da tabela 1.[7] A tabela 2 e o gráfico da figura 1 indicam a evolução média do algoritmo nos testes realizados. Estão computados os valores de penalidade médios do melhor indivíduo e da média da população a cada 200 gerações do ciclo do GA.

Parâmetro	**Valor***	**Observação**
Tamanho da população	70	
Aplicações de Crossover Uniforme	70	
Aplicações de Crossover de N-Pontos	00	
Mutação Simples	7 - 15**	Escala a partir da geração 1
Mutação Corretiva	0 - 250***	Escala a partir da geração 1300
Seleção Elitista	10	
Penalidade para Aulas Simultâneas	500	
Penalidade para Violação de Indisponibilidade	500	
Penalidade para Janela de Professores	500	
Penalidade para Violação de Aula Coordenada	250	Por slot não coordenado
Penalidade para Violação de Aula N-Pla	250	
Penalidade para Violação de Preferências	50	
Penalidade para Aula Dupla Separada	50	Restrição de baixa prioridade
Penalidade para Não Satisfação de Day-Off	50	
Utiliza algoritmo alternativo	Sim	Vide seção 3.6

* Os parâmetros de operadores genéticos e processos de seleção são indicados em quantidade de indivíduos

** O valor é acrescido de 1 a cada 100 gerações

*** O valor é acrescido de 8 a cada 100 gerações

Tabela 1: Parâmetros utilizados pelo Algoritmo Genético

Em uma análise dos resultados gerados, podemos ressaltar os seguintes pontos:

[7] Posteriormente, na seção 4.3, serão descritas as variações realizadas em tais parâmetros, para efeitos de teste

Geração	Melhor*	Média**	Gerac.	Melhor	Média	Gerac.	Melhor	Média
1	83830	94427	2200	5180	8236	4400	1745	3050
200	50005	50711	2400	4675	8073	4600	1605	3024
400	37300	37770	2600	4375	7302	4800	1530	2873
600	30920	31627	2800	4020	7318	5000	1470	2726
800	26875	27563	3000	3595	7073	5200	1455	2627
1000	24400	25283	3200	3365	6352	5400	1420	2831
1200	22825	23601	3400	2790	4792	5600	1405	2584
1400	16720	17914	3600	2380	3816	5800	1400	2657
1600	9745	12022	3800	2060	3546	6000	1390	2244
1800	7675	9994	4000	1955	3376			
2000	6225	8990	4200	1840	3149			

* Os valores de qualificação dos indivíduos são representados pela penalidade total calculada

** A média da população é apresentada apenas para efeito ilustrativo, pois somente o melhor indivíduo é considerado no retorno do algoritmo

Tabela 2: Evolução média do Algoritmo Genético

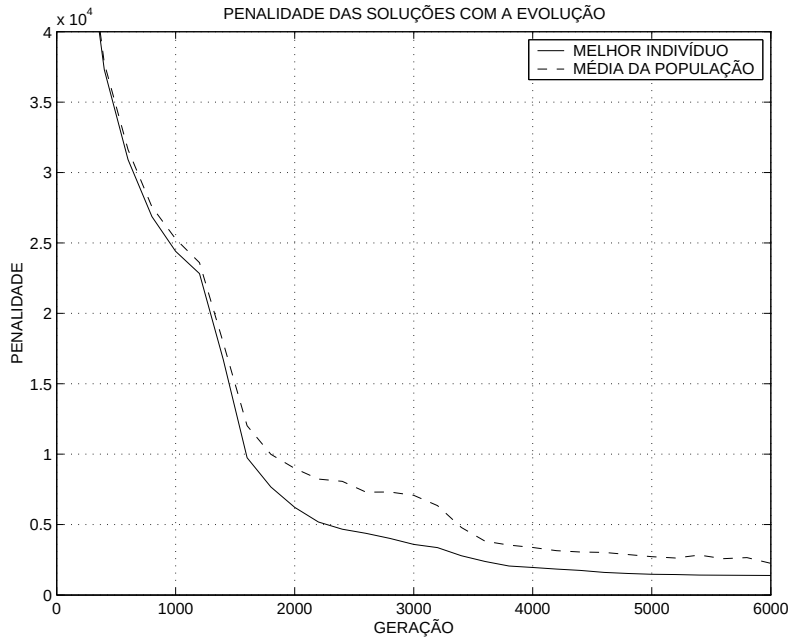


Figura 1: Gráfico da Penalidade x Geração durante a evolução do Algoritmo Genético

- Os indivíduos finais, ou seja, os melhores indivídusos encontrados no processo evolutivo, contém penalidade média de 1390, que é devida principalmente às violações de restrições fracas.

- O programa é executado em média durante 459 segundos (cerca de 7:40 minutos) antes de convergir, o que é um tempo razoável para um algoritmo genético e totalmente viável do ponto de vista prático.

- São processadas cerca de 6000 gerações, o que se mostrou suficiente para a obtenção de bons resultados. Depois desse valor, embora continue ocorrendo o processo evolutivo, o tempo de processamento começa a não compensar os ganhos de qualidade na solução.

- Os casos de convergência prematura são estimados em cerca de apenas 20% do total - calculado com 90% de confiança - o que é um valor tolerável do ponto de vista prático. No entando, o algoritmo teve dificuldade em tratar as aulas coordenadas, sendo que em média apenas 60% do total das mesmas são atendidas.

Pelo gráfico do ciclo evolutivo na figura 1 podemos levantar alguns pontos relevantes sobre a convergência das populações:

- Após o início do processo de estagnação do algoritmo, em torno da geração 1500, ocorre um ponto de queda mais acentuada da média de penalidades. Isso se deve ao início do uso das mutações corretivas, que se inicia após algum tempo de processamento. Vide seção 4.3.4 para mais detalhes sobre o ponto de início de aplicação das mutações corretivas.

- A média de penalidade da população mantém-se estável e relativamente próxima à do melhor indivíduo, o que indica a existência de um mesmo nicho sendo explorado pelo algoritmo. O caso contrário - média da população oscilante e diferente do melhor indivíduo - poderia indicar que o algoritmo não está convergindo adequadamente e mantém-se num processo mais próximo de uma busca aleatória[8].

Vide no anexo A um exemplo de solução encontrada com o algoritmo, utilizando-se a configuração descrita anteriormente.

4.3 Resultados dos Testes sobre os diversos Parâmetros

Para se chegar às conclusões da adequabilidade dos parâmetros citados na seção 4.2, uma grande quantidade de testes precisou ser realizada com o algoritmo, analisando-se as mais diversas variações possíveis e combinações de parâmetros. Para cada proposta de configuração, foram executados testes sobre instâncias reais para se poder chegar à conclusão dos efeitos de tais parâmetros na execução do algoritmo.

Não é garantido que a combinação encontrada na seção 4.2 seja a melhor possível. Encontrar tal combinação consistiria em solucionar um problema combinacional com quase 20 variáveis independentes (todos os parâmetros do algoritmo). Mas, comprovadamente, a mesma é uma *boa* combinação, que permite encontrar resultados adequados para o problema original de agendamento.

Serão apresentados, nas seções seguintes, os resultados obtidos para as principais alternativas de arquivos de configuração investigadas, evidenciando o porquê da escolha dos

[8] Tal situação ocorria nas primeiras versões do algoritmo, quando a taxa de mutação aplicada era consideravelmente mais alta do que a atual

parâmetros indicados na seção 4.2. Para realizar os testes, foi escolhida uma combinação-base encontrada empiricamente no desenvolvimento do algoritmo. A partir dessa combinação, foram usadas variações nos parâmetros a serem estudados, e executadas diversas intâncias do algoritmo para se estudar os efeitos de tal variação.

A combinação-base gera resultados cujas médias têm as seguintes características:

- solução com penalidade média de 1620;
- tempo de execução médio de 409 minutos (aproximadamente 6:50);
- parâmetros idênticos aos da tabela 1, exceto pelo valor de acréscimo da taxa de mutação corretiva (incremento de 7 indivíduos mutados a cada 100 gerações, ao invés de 8).

Dessa forma, todos os resultados serão comparados com tais valores, para sabermos se a variante dos parâmetros apresentou melhora ou piora no algoritmo. Obviamente o tempo de execução é extremamente dependente do computador onde o programa é executado, mas o que interessa em nossa análise é o ganho *relativo* entre as variantes, uma vez que todos os testes foram executados no mesmo computador.

4.3.1 Variações sobre tipo de Crossover

Os seguintes testes foram realizados explorando-se a geração de descendentes pelos dois tipos de crossover - Uniforme/n-pontos - a relação base é de 70/0 (70 gerações por crossover Uniforme e nenhuma geração por crossover de n-pontos), sobre uma população de 70 indivíduos:

	Mudança de Parâmero	**Média**	**Tempo**
Teste 1	Relação Crossover Unif./n-pontos = 60/10	1525	410
Teste 2	Relação Crossover Unif./n-pontos = 50/20	2185	414
Teste 3	Relação Crossover Unif./n-pontos = 40/30	1700	412
Teste 4	Relação Crossover Unif./n-pontos = 30/40	2700	376
Teste 5	Relação Crossover Unif./n-pontos = 20/50	3390	432

Pela tabela, que mostra a penalidade média das soluções encontradas e o tempo médio de execução em minutos, percebe-se uma melhora na solução média no Teste 1. Mas tal melhora não se manteve na exploração de parâmetro combinados no algoritmo, como será mostrado na seção 4.3.6. Dessa forma, a melhor configuração encontrada não utiliza o crossover de n-pontos descrito na seção 3.4.

Nem sempre tal fato se verificou durante o desenvolvimento do projeto. Nas primeiras etapas de testes, a aplicação de uma relação 60/10, embora não tendo ocasionado uma melhora substancial na qualidade das soluções encontradas, reduzia o número de convergências prematuras. Nesse estágio, tais convergências giravam em torno de 30% das execuções sem a utilização do crossover de n-pontos e 20% com o mesmo.

4.3.2 Variações sobre Tamanho da População

Para o tamanho da população, foram feitas as seguintes variações em torno do valor original de 70 indivíduos:

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	População de 90 indivíduos	1895	458
Teste 2	População de 110 indivíduos	2635	447
Teste 3	População de 130 indivíduos	2290	461
Teste 4	População de 50 indivíduos	2595	386
Teste 5	População de 30 indivíduos	2520	362

Nenhuma variação explorada resultou em melhores soluções. O que se obteve foi uma melhora no tempo de execução para populações menores, mas que não justifica a perda de qualidade obtida na solução final.

4.3.3 Variações sobre Seleção Elitista

As variações realizadas em torno da quantidade de indivíduos selecionados pelo processo elitista, originalmente de valor 10, foram:

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	Seleção elitista de 1 indivíduo	2425	417
Teste 2	Seleção elitista de 5 indivíduo	2611	391
Teste 3	Seleção elitista de 15 indivíduo	1545	410
Teste 4	Seleção elitista de 20 indivíduo	2150	412
Teste 5	Seleção elitista de 25 indivíduo	1945	408

Analogamente à seção 4.3.1, foi encontrada uma combinação que apresentou melhores resultados do que a padrão (Teste 3), mas tal melhora foi superada por outra variação a ser vista adiante e não surtiu resultados na combinação de ambas.

4.3.4 Variações sobre Aplicação da Mutação Corretiva

Pelo caráter específico do operador de Mutação Corretiva, já que tal recurso não é explorado em grande escala na literatura, mais testes foram realizados em torno de sua utilização.

Basicamente três aspectos foram analisados: a variação da taxa de aplicação do operador com o passar das gerações - uma vez que o mesmo foi adotado como parâmetro variável, o ponto de início na execução em que o operador começa a ser aplicado e o ponto em que o mesmo é aplicado de forma mais intensa, tentando corrigir mais de uma violação por cromossomo (vide seção 3.4.2 para mais detalhes).

No primeiro caso, os seguintes resultados foram obtidos, partindo-se do parâmetro base de acréscimo de 7 aplicações a cada 100 gerações (ou seja, a cada 100 gerações, 7 indivíduos são selecionados a mais para sofrerem uma mutação corretiva):

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	Acréscimo de 8 a cada 100 gerações	1390	460
Teste 2	Acréscimo de 6 a cada 100 gerações	2410	347
Teste 3	Acréscimo de 10 a cada 100 gerações	2280	559
Teste 4	Acréscimo de 4 a cada 100 gerações	2620	269

Podemos perceber que houve uma melhora na qualidade das soluções no Teste 1 (incremento de 8). Tal resultado não foi superado por nenhuma outra combinação e, como vimos na seção 4.2, foi o melhor encontrado em todos os testes realizados.

Na seqüência, foram feitos testes no ponto de início de aplicação do operador, ou seja, a partir de qual geração os indivíduos começam a sofrer mutações corretivas, e a taxa máxima de aplicação do mesmo. Esse parâmetro está estreitamente relacionado com o anterior, uma vez que os mesmos pontos de início de aplicação utilizando taxas de incremento diferentes implicam em taxas máximas em tempos diferentes.

Nesse caso, o parâmetro base é o início da aplicação na geração 1300, com uma taxa de incremento de 7 e limitado ao máximo de 250 aplicações.

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	Início na geração 1400, com máx. de 200 e incr. de 7	2635	378
Teste 2	Início na geração 700, com máx. de 200 e incr. de 7	1435	452
Teste 3	Início na geração 0, com máx. de 250 e incr. de 7	2760	502
Teste 4	Início na geração 0, com máx. de 250 e incr. de 5	1665	475
Teste 5	Início na geração 1000, com máx. de 250 e incr. de 15	2430	581

Nesse caso, vemos que os melhores resultados alcançados não superam em qualidade a melhor solução descrita na análise anterior do operador. Um fato interessante a ser notado é o alto valor de penalidade para a aplicação iniciada na geração 0 e incrementada de 7 a cada 100 gerações; esse valor deve-se ao alto número de convergências prematuras ocorridas (em torno de 50%), num algoritmo com alta pressão evolutiva. Tal característica não ocorre na evolução com incremento de 5, que não tenta corrigir os cromossomos com tanta intensidade nas primeiras gerações.

Por fim, foram feitos testes com a versão *forte* do operador de mutação corretiva. Nessa versão, cada vez que um indivíduo é selecionado para mutação, são feitas simultaneamente várias tentativas de correções - geralmente em número equivalente à quantidade de violações existentes no cromossomo. Nos parâmetros iniciais, tal aplicação é iniciada a partir da geração 5000.

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	Aplicação a partir da geração 4000	1405	413
Teste 2	Aplicação a partir da geração 3000	1800	411
Teste 3	Aplicação a partir da geração 2000	1890	429
Teste 4	Aplicação a partir da geração 6000	1850	403
Teste 5	Aplicação a partir da geração 4500	1650	434

Vemos que não ocorrem muitas variações nos resultados para os diversos pontos de aplicação, exceto em torno da geração 4000. Mas a melhor combinação obtida ainda é a descrita anteriormente.

4.3.5 Variações sobre Valores de Penalidades

Foram feitos testes sobre algumas variações de valores de penalidades. O objetivo, nesse caso, é tentar eliminar com mais eficiência determinadas restrições, mas procurando-se não afetar as demais. Para tanto, foram feitos os seguintes testes:

	Mudança de Parâmetro	**Eficaz***	**Tempo**
Teste 1	Dobra penalidade de Aula n-Pla	Sim	426
Teste 2	Dobra penalidade de Aula Junta	Não	420
Teste 3	Dobra penalidade de Day-Off	Não	414
Teste 4	Dobra penalidade de Aulas Coordenadas	Não	420
Teste 5	Dobra penalidade de todas as Restrições Fortes	Não	417

* Os valores de penalidade foram propositadamente omitidos, uma vez que não faz sentido compará-los com os valores obtidos com outros experimentos

Quase todas as variações se mostraram ineficazes para os objetivos aos quais foram destinadas. Isso é, a quantidade de violações das penalidades continuou praticamente a mesma ou mesmo aumentaram. Somente no caso de aumento da penalidade de aulas n-Plas foi conseguida uma menor carga de violação, mas a diferença foi pequena se comparada ao esquema padrão e não superou outras soluções de melhor qualidade.

4.3.6 Variações sobre Combinações de Parâmetros Distintos

Após feitos testes com os parâmetros do algoritmo, diversas combinações de novos parâmetros sugeririam melhoras na solução geral encontrada. Contudo, torna-se necessário testar tais combinações para se saber quais os efeitos de duas ou mais variações simultâneas.

As combinações escolhidas para testes foram aquelas que se mostraram mais promissoras quando testadas isoladamente, e que aparentemente não se relacionavam com outras modificações que também surtiram melhoras na solução média. Os resultados de alguns dos testes estão a seguir:

	Mudança de Parâmetro*	**Média**	**Tempo**
Teste 1	Combinação de Crossover mais Fator de Acréscimo das Mutações Corretivas	2820	417
Teste 2	Combinação de Acrésc. de Mutação Corretiva mais Seleção Elitista	1840	463
Teste 3	Combinação de Acrésc. de Mutação Corretiva mais penalidade de aula n-pla	2175	431
Teste 4	Combinação de Acrésc. de Mutação Corretiva, com o operador aplicado a partir da geração 700 até máx. de 200 indivíduos	2815	429
Teste 5	Combinação de Acrésc. de Mut. Corretiv, Relação entre Crossovers e Seleção Elitista	2385	432

* As mudanças citadas dizem respeito às modificações que obtiveram os melhores resultados nos testes independentes

Outras combinações também foram testadas, mas todas seguiram o mesmo padrão das apresentadas: não foram bem sucedidas quando aplicadas em conjunto.

4.3.7 Testes sobre a Lógica do Algoritmo

Por fim, foram feitos testes referentes ao Algoritmo Genético modificado citado na seção 3.6. Os testes consistem na re-inversão da ordem do algoritmo, para retomar a ordem comum dos GA's tradicionais: processo de seleção aplicado para escolha dos progenitores, dos quais serão feitas operações de crossover para originar os indivíduos da nova geração.[9]

O único parâmetro que é modificado nesse caso é o tamanho do *pool de indivíduos selecionados*. Este pool consiste num conjunto de indivíduos da população escolhidos através de processos de seleção e que serão os progenitores da próxima geração. Dessa forma, para se gerar a nova população são seguidos os procedimentos:

1. Pelos processos de seleção, escolhem-se os indivíduos que constituirão o pool;
2. São gerados os indivíduos da nova geração a partir de dois progenitores, escolhidos aleatoriamente do pool.

Os testes realizados apresentaram os seguintes resultados.

	Mudança de Parâmetro	**Média**	**Tempo**
Teste 1	Pool de 20 indivíduos	4770	442
Teste 2	Pool de 10 indivíduos	4485	448
Teste 3	Pool de 30 indivíduos	4740	474
Teste 4	Pool de 40 indivíduos	3525	439
Teste 5	Pool de 50 indivíduos	2900	545

[9] Na forma com que o algoritmo foi implementado neste projeto, primeiro são gerados descendentes a partir da população atual e todos os indivíduos, descendentes e progenitores, participam do processo de seleção para a geração posterior

Como podemos perceber, todos os testes apresentaram resultados bem inferiores aos do algoritmo genético modificado. Essa foi a razão da escolha de algoritmo utilizada no projeto.

4.4 Comparação com outras técnicas de resolução

Outros trabalhos do GOA exploraram a mesma instância do problema de Agendamento, mas utilizando outras técnicas de resolução. Foram finalizados um projeto que utilizou Programação por Restrições e outro que usou Busca Tabu. Novos projetos estão sendo preparados para futuramente aplicar outras técnicas, como Heurísticas GRASP, mas ainda não estão em um nível de desenvolvimento próprio para comparações com as demais técnicas.

Uma comparação qualitativa entre as soluções obtidas pelas heurísticas encontra-se a seguir.

Algoritmo Genético e Busca Tabu: De modo geral, ambas as heurísticas obtiveram soluções factíveis e um bom nível de satisfação das restrições do colégio Stella Maris. A Busca Tabu se mostrou mais eficiente no tempo de processamento do programa: executando no mesmo computador, o tempo de processamente gira em torno de um pouco menos de 6 minutos, enquanto o GA necessita de quase 8 minutos para convergir. Em contrapartida, o Algoritmo Genético apresentou maior facilidade para o tratamento das restrições fracas, eliminando um maior número dessas restrições. Porém, é importante frisar que todas violações apresentadas pelas heurísticas podem ser resolvidas facilmente pelo usuário, via interface, viabilizando a utilização das soluções encontradas.

Programação por Restrições: Pela própria natureza deteminística da técnica, foi encontrada grande dificuldade no tratamento de Restrições Fracas. Embora o programa encontrasse soluções sem violação de restrições fortes em frações de segundo, o processo de eliminação de restrições fracas demorava diversas horas para convergir, inviabilizando sua aplicação para o usuário final.

Futuramente, pretende-se fazer testes com a *hibridização* entre as técnicas, para melhorar a performance do programa e a qualidade das soluções. Mais informações sobre os trabalhos futuros podem ser vistos na seção 6.

5 Implementação da Interface

Um ponto importante do projeto é a implementação de uma interface de uso para os usuários do sistema de Agendamento Acadêmico. Essa interface deve permitir que o usuário obtenha de forma fácil e transparente as soluções que deseja, além de permitir pequenas alterações onde o usuário achar conveniente.

A tarefa de elaboração da interface foi realizada em conjunto com outros alunos do GOA[10], já que a princípio ela independe do método utilizado para solucionar o problema.

[10] Alunos que desenvolveram projetos semelhantes utilizando outras técnicas

A ferramenta utilizada no desenvolvimento foi o Delphi 5.0 com uma base de dados MS Access. Os motivos para tal escolha foram o conhecimento prévio sobre as ferramentas, o que agilizou razoavelmente o desenvolvimento, e a disponibilidade de licenças de uso dos softwares, que o Instituto de Computação da Unicamp dispõe para pesquisas acadêmicas.

5.1 Entrada e Saída de Dados no Sistema

O sistema a ser desenvolvido, incluindo-se a interface e os vários núcleos resolvedores (como o Algoritmo Genético), deverá estabelecer uma padronização para a passagem dos parâmetros a serem utilizados. Parâmetros esses tanto do problema (dados das grades, professores e turmas) quanto do algoritmo (penalidades aplicadas, taxas de operadores, tamanho da população, etc). Para isso, foi elaborada uma especificação detalhada de como os dados do problema devem ser passados para o sistema e como a interface fará a comunicação com os núcleos resolvedores.

5.1.1 Telas de entrada de dados pelo usuário

Primeiramente, é preciso que os usuário entrem com as informações do problema no sistema. Ou seja, é necessário que se informe quantas séries, turmas, professores e matérias existem e quais são os relacionamentos entre essas entidades. Essas informações devem alimentar o banco de dados MS Access para que o sistema possa realizar as consultas para utilização dos dados.

Para que o usuário não necessite dispor de licença de uso para o aplicativo MS Access, um software proprietário, foram desenvolvidas telas para que se possa digitar essas informações diretamente via interface. Exemplo destas telas estão nas figuras 2 e 3, que permitem a entrada dos dados referentes aos professores. Para cada entidade do sistema, como séries, grades, turmas e matérias, existem telas semelhantes para que as propriedades e relacionamentos possam ser informadas.

5.1.2 Comunicação entre Interface e núcleo resolvedor

É necessário que se estabeleça uma forma de comunicação entre a interface, desenvolvida em Delphi, e os núcleos resolvedores como o GA, desenvolvidos em C. Isso para que a interface possa transmitir as informações da base de dados para o núcleo e o mesmo possa retornar uma solução encontrada para a interface.

Um modelo de entrada interessante e de fácil manipulação seria um arquivo texto sem formatações. Cada seção de dados seria introduzida por um título entre colchetes (`[` e `]`) e nas linhas a seguir viriam as informações relacionadas. Cada campo de dados seria identificado por um nome, seguido de um sinal de igual (`=`) e terminando com seu valor. No caso de campos que podem conter vários valores, após o sinal de igual seria informada a quantidade destes e os dados viriam nas linhas seguintes, um por linha (ou um bloco de dados por linha, separados por espaços e devidamente padronizados)[11].

[11] A elaboração desse formato é baseada na estrutura dos antigos arquivos INI de configuração do Microsoft Windows, com algumas modificações

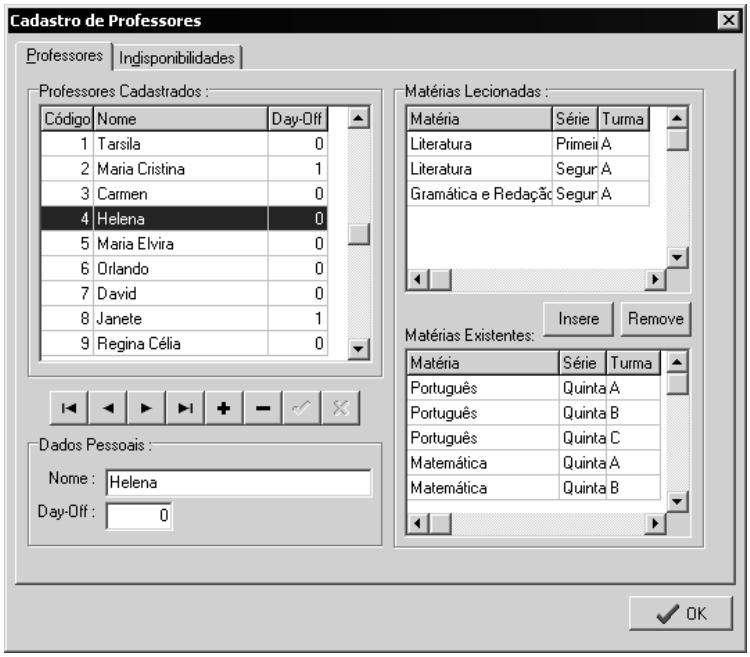


Figura 2: Tela de Cadastro de Professores na interface

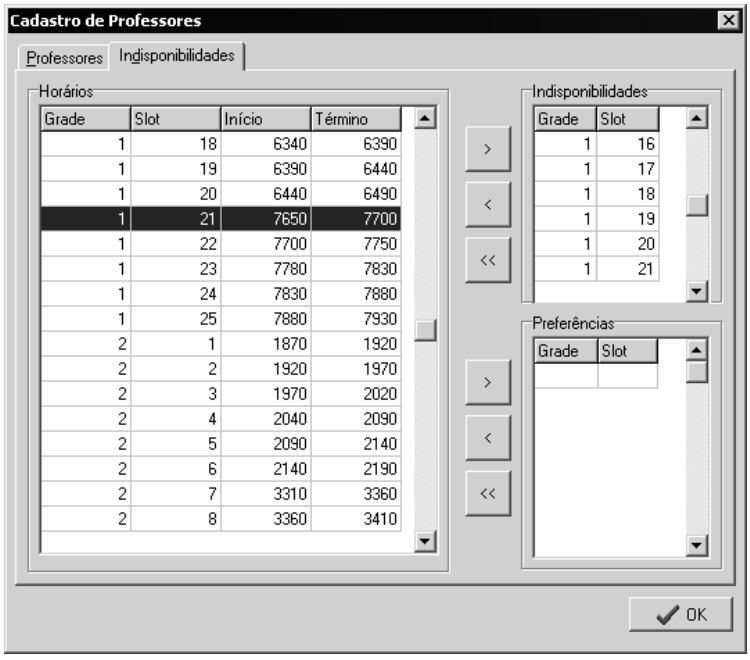


Figura 3: Tela de Cadastro de Professores na interface

Poderiam ser utilizados outros formatos, como o padrão XML (*Extensible Markup Lan-*

guage), mas o grupo chegou a conclusão de que não valeria a pena os esforços de implementação de um modelo mais complexo, visto que essa comunicação é feita apenas entre a interface e o núcleo, não sendo utilizada para outras finalidades - como transmissão de dados para sistemas de terceiros.

Baseado neste modelo, foram descritos os campos de dados a serem fornecidos pela interface, na ordem em que devem estar no arquivo de configurações. Tudo o que diz respeito ao problema deverá ser informado, ou poderá ser obtido facilmente pelo núcleo a partir das informações dadas. A classificação de tais campos encontra-se a seguir.

1. **Seção** `escola`: Dados gerais da escola. Contém os campos:
   (a) **Campo** `grades`: Tipo inteiro, informa qual a quantidade de grades da escola. Campos a serem repetidos na seqüência para cada grade:
      i. **Campo** `codigo`: Tipo inteiro, especifica o código de identificação da grade utilizado pela interface;
      ii. **Campo** `slots`: Tipo inteiro, informa qual a quantidade de horários de aula na semana. Nas linhas seguintes devem ser informados os horários de início de cada slot, um por linha, no formato "minuto da semana";
      iii. **Campo** `dias`: Tipo inteiro, informa qual a quantidade de dias úteis na semana. Para cada dia, deve ser informado o horário de início das atividades, no formato "dia da semana";
      iv. **Campo** `coordenadas`: Tipo inteiro, informa qual a quantidade de aulas coordenadas que a grade possui. Para cada coordenação, deve ser informado nas linhas seguintes 4 códigos inteiros por linha, respectivamente: código de uma turma que será coordenada, código da matéria dessa turma que será coordenada, código da segunda turma que será coordenada com a primeira, código da matéria da segunda turma que ocorrerá no mesmo horário que a matéria da primeira;

2. **Seção** `materia`: Dados gerais de cada matéria ministrada na escola. Contém os campos:
   (a) **Campo** `quantidade`: Tipo inteiro, informa qual a quantidade de matérias da escola. Campos a serem repetidos na seqüência para cada matéria:
      i. **Campo** `codigo`: Tipo inteiro, identifica a matéria;
      ii. **Campo** `cargahoraria`: Tipo inteiro, informa quantos slots de aula semanais devem ser ministradas para essa matéria;
      iii. **Campo** `serie`: Qual série terá essa matéria; [12]
      iv. **Campo** `nivel`: Tipo inteiro, especifica uma escala de nível de dificuldade para a matéria, de 1 (mais fácil) a 3 (mais difícil).

3. **Seção** `serie`: Dados referentes às séries. Contém os campos:

[12] Matérias de nomes semelhantes são vistas como diferentes se dadas para séries diferentes; é o caso de português para sextas e sétimas séries, por exemplo

(a) **Campo** `quantidade`: Tipo inteiro, informa a quantidade de séries existentes. Campos a serem repetidos na seqüência para cada série:

i. **Campo** `codigo`: Tipo inteiro, identifica a série em questão;

ii. **Campo** `turmas`: Tipo inteiro, informa a quantidade de turmas desta série. Para cada turma deve ser informado o código da grade na qual a mesma se encaixa (tipo inteiro), um por linha;

iii. **Campo** `materias`: Tipo inteiro, informa a quantidade de matérias na grade curricular. Devem ser informados nas linhas seguintes os códigos das matérias existentes (tipo inteiro) seguidos de sua carga horária (tipo inteiro), uma matéria por linha.

4. **Seção** `professor`: Dados referentes aos professores. Contém os campos:

(a) **Campo** `quantidade`: Tipo inteiro, informa a quantidade de professores existentes. Campos a serem repetidos na seqüência para cada professor:

i. **Campo** `codigo`: Tipo inteiro, identifica o professor em questão;

ii. **Campo** `materias`: Tipo inteiro, informa a quantidade de matérias que o professor leciona. Devem ser informados nas linhas seguintes os códigos das matérias em questão (tipo inteiro) seguidos da quantidade de turmas lecionadas a seus respectivos códigos (tipos inteiros), uma matéria por linha;

iii. **Campo** `indisponibilidades`: Tipo inteiro, informa em quantos horários de aula o professor estará incondicionalmente indisponível. Devem ser informados nas linhas seguintes os códigos desses horários (slots) em questão (tipo inteiro), um por linha;

iv. **Campo** `preferencias`: Tipo inteiro, informa em quantos horários de aula o professor gostaria de não ministrar aulas. Devem ser informados nas linhas seguintes os códigos desses horários (slots) em questão (tipo inteiro), um por linha;

v. **Campo** `dayoff`: Tipo inteiro, informa a quantidade dias inteiros que o professor gostaria de ter vagos.

Observação: o formato "minutos da semana" foi uma forma encontrada para que pudessem ser informados os horários num formato de número inteiro. Cada horário é calculado a partir do tempo corrido, em minutos, das 0:00 do domingo anterior. Isso permite identificar horas de início e fim das aulas e compará-las com horários de outros slots.

Da mesma forma que na entrada de dados para os núcleos, a saída também deve possuir uma padronização. Dessa forma, a interface saberá ler a solução obtida para o problema e apresentá-la para o usuário, permitindo manipulações posteriores por parte do mesmo.

Um padrão que poderia ser seguido seria semelhante ao da entrada de dados. Ou seja, um arquivo de texto sem formatações, com seções nomeadas entre colchetes e nomes de campos precedidos por um sinal de igual.

Os campos que seriam informados nesse formato seriam:

1. **Seção `timetable`**: Seção única, com o agendamento (timetable) resultante. Contém os campos:

   (a) **Campo `grades`**: Tipo inteiro, informa qual a quantidade de grades da escola. Serve para facilitar a leitura por parte da interface. Campos a serem repetidos para cada grade:

      i. **Campo `codigo`**: Tipo inteiro, informa qual o código da turma em questão;

      ii. **Campo `slots`**: Tipo inteiro, informa qual a quantidade de slots que a turma tem na semana. Novamente, este campo tem a função de facilitar a leitura;

      iii. **Campo `turmas`**: Tipo inteiro, informa qual a quantidade de turmas que estão alocadas nesta grade. Devem ser informados nas linhas seguintes os códigos das matérias que serão lecionadas (tipo inteiro), um por linha e em ordem de slot. A quantidade de linhas é ditada pela quantidade de slots de aula para a grade desta turma.

5.2 As grades de horários

Uma vez passados ao núcleo todos os dados que o mesmo necessita para encontrar um agendamento, e o mesmo tendo retornado a agenda encontrada, é necessário que se transmita essa informação para o usuário via interface. Para isso, a mesma contém telas de saída em que são dispostas as agendas de maneira que o usuário tenha facilidade em identificar as alocações e reconhecer os horários agendados.

A interface desenvolvida permite três visões da solução do problema: o ponto de vista do professor, da turma e geral. Exemplos de dada uma dessas visões estão, respectivamente, nas figuras 4, 5 e 6.

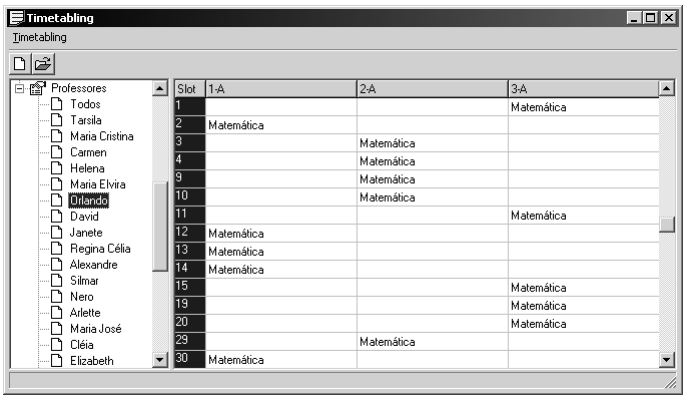


Figura 4: Solução do problema do ponto de vista de um professor

5.3 Funcionalidades do sistema

Uma funcionalidade imprescindível na interface de saída é a possibilidade do usuário trocar duas aulas de posição caso deseje. Essa funcionalidade foi implementada via o recurso "drag-

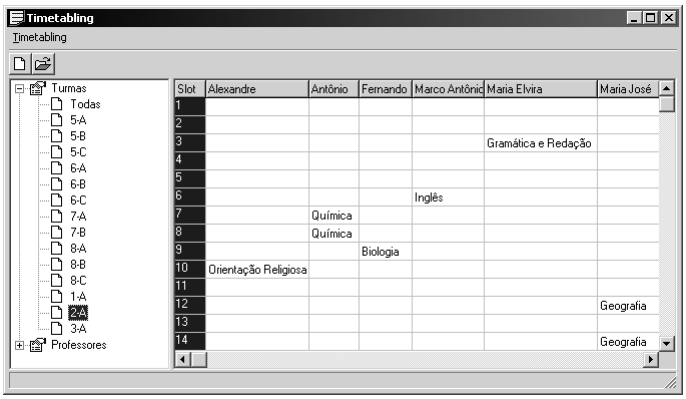


Figura 5: Solução do problema do ponto de vista de uma turma

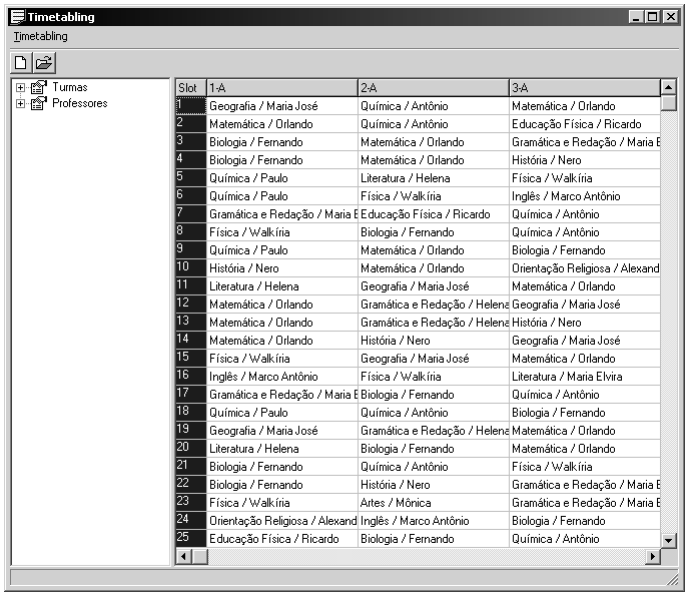
and-drop", padrão do sistema Windows, pelo qual se seleciona um objeto com o mouse, arrastando-o até a posição desejada e soltando-o. Com isso, se uma alocação de aula não for do gosto do usuário, o mesmo pode trocá-la na grade. Essa função foi implementada apenas para a grade de visão global, para que não se corra o risco do usuário alocar um professor num local em que outro já está lecionando.

Além dessa funcionalidade, outros recursos poderão ser incorporados posteriormente, já que o grupo de pesquisa do GOA continuará seus trabalhos através de outros projetos. Em reuniões com os usuários reais do sistema, no colégio Stella Maris e outros colégios que eventualmente serão consultados, deverão surgir novas necessidades que podem vir a melhorar as funcionalidades do programa.

6 Considerações Finais

O sistema desenvolvido no projeto para realizar o Agendamento Acadêmico nos moldes das escolas brasileiras de ensino fundamental e médio se mostrou adequado para atender às exigências do problema. Soluções de nível bastante satisfatório foram encontradas em um tempo de processamento relativamente curto. Além disso, a interface permite que o usuário faça as pequenas correções que julgar serem necessárias. Essas características motivam o uso do sistema em situações reais de agendamento. Utilizado de forma adequada, pode ser de grande valia para escolas públicas e privadas, reduzindo drasticamente o tempo necessário para a montagem dos horários.

O Algoritmo Genético desenvolvido, por ter apresentado dificuldades iniciais ao lidar com a grande quantidade de restrições fortes do problema (como janelas e aulas coordenadas), sofreu algumas modificações no sentido de se buscar maior eficiência no processo evolutivo. Alguns desses recursos não são comumente encontrados nos algoritmos genético da literatura, como é o caso das mutações corretivas e da antecipação do processo de recombinação frente ao processo de seleção de indivíduos. Esse é um ponto interessante, que pode inspirar projetos que enfrentem dificuldades semelhantes a explorar o pontencial desses recursos.



Slot	1-A	2-A	3-A
1	Geografia / Maria José	Química / Antônio	Matemática / Orlando
2	Matemática / Orlando	Química / Antônio	Educação Física / Ricardo
3	Biologia / Fernando	Matemática / Orlando	Gramática e Redação / Maria E
4	Biologia / Fernando	Matemática / Orlando	História / Nero
5	Química / Paulo	Literatura / Helena	Física / Walkíria
6	Química / Paulo	Física / Walkíria	Inglês / Marco Antônio
7	Gramática e Redação / Maria E	Educação Física / Ricardo	Química / Antônio
8	Física / Walkíria	Biologia / Fernando	Química / Antônio
9	Química / Paulo	Matemática / Orlando	Biologia / Fernando
10	História / Nero	Matemática / Orlando	Orientação Religiosa / Alexand
11	Literatura / Helena	Geografia / Maria José	Matemática / Orlando
12	Matemática / Orlando	Gramática e Redação / Helena	Geografia / Maria José
13	Matemática / Orlando	Gramática e Redação / Helena	História / Nero
14	Matemática / Orlando	História / Nero	Geografia / Maria José
15	Física / Walkíria	Geografia / Maria José	Matemática / Orlando
16	Inglês / Marco Antônio	Física / Walkíria	Literatura / Maria Elvira
17	Gramática e Redação / Maria E	Biologia / Fernando	Química / Antônio
18	Química / Paulo	Química / Antônio	Biologia / Fernando
19	Geografia / Maria José	Gramática e Redação / Helena	Matemática / Orlando
20	Literatura / Helena	Biologia / Fernando	Matemática / Orlando
21	Biologia / Fernando	Química / Antônio	Física / Walkíria
22	Biologia / Fernando	História / Nero	Gramática e Redação / Maria E
23	Física / Walkíria	Artes / Mônica	Gramática e Redação / Maria E
24	Orientação Religiosa / Alexand	Inglês / Marco Antônio	Biologia / Fernando
25	Educação Física / Ricardo	Biologia / Fernando	Química / Antônio

Figura 6: Solução do problema do ponto de vista global

Na sequência dos trabalhos, deverá se iniciar uma fase de divulgação dos resultados obtidos, com testes sobre mais dados reais de escolas e coletando-se sugestões de melhorias de outras instituições, além do colégio Stella Maris. Paralelamente, serão pesquisadas novas técnicas para aperfeiçoamento do Algoritmo Genético, além de formas de integração com outros algoritmos de resolução, visando encontrar soluções ainda melhores e em tempo de processamento mais curto. Alguns exemplos desses pontos de pesquisa a serem trabalhados são:

- *Paralelização do Algoritmo Genético*, através de *Modelos de Ilhas* ou *Modelos Celulares* [WHI93]. Com isso, procurar-se-á eliminar a ocorrência de convergências prematuras (que atualmente gira em torno de 20%) e concentrar o nível das soluções encontradas em patamares superiores aos atuais. Também poderão ser testados métodos paralelos para diminuir o tempo de processamento - por exemplo, distribuindo entre diversas CPUs a tarefa de avaliar as populações.

- *Integração com Busca Tabu e/ou Heurísticas GRASP*, no sentido de que tais algoritmos, executando em paralelo com o GA, podem participar do processo evolutivo através do intercâmbio de soluções encontradas. Por exemplo, a Busca Tabu poderia ser utilizada para se criar populações iniciais de qualidade para o GA; ou o GA poderia fornecer boas soluções iniciais para uma Heurística GRASP.

A existência de projetos que usem tais aperfeiçoamentos é ainda mais rara na literatura sobre *Timetabling* no Brasil, principalmente no caso da integração entre algoritmos diferentes. Dessa forma, deve-se aperfeiçoar ainda mais o sistema e enriquecer, no meio científico, a pesquisa de heurísticas aplicadas a soluções de problemas NP-Difícieis.

A Exemplo de solução

Para exemplificar a qualidade das soluções obtidas pelo algoritmo genético, será apresentado neste anexo uma solução encontrada com o conjunto de parâmetros mais eficiente (vide seção 4.2). As tabelas de 3 a 9 representam uma solução encontrada com valor de penalidade mediana. Alguns dados a respeito da agenda do colégio para o qual o exemplo é aplicado estão listados a seguir. A descrição completa das restrições do colégio podem ser vistas no anexo B.

- Quatro séries do ensino fundamental (5^a a 8^a séries), com 3 turmas cada uma (exceto a 7^a série, com 2 turmas), e três séries do ensino médio (1^a a 3^a séries), com uma turma cada. Ao todo são 14 turmas.
- 11 matérias diferentes para cada série, a serem ministradas em 5 slots de aula por dia para o ensino fundamental e 6 slots de aula por dia para o ensino médio, em 5 dias úteis na semana. Ao todo existem 77 matérias na escola.
- 30 professores disponíveis, cada qual com sua restrição de horário; muitos não são professores de período integral.

A solução encontrada para o problema possui penalidade total 1200 (parametrizada pela tabela 1) e foi encontrada em 7:36 minutos de processamento, em um computador Intel Xeon e 2,5 Ghz. Para analisar os dados a seguir é importante ter em mente que:

- Nas tabelas 3, 4, 5 e 6 encontram-se os horários alocados para as quintas, sextas, sétimas e oitavas séries do ensino fundamental, respectivamente. Os slots indicam os horários de aula e 5 dos mesmos correspondem a um dia letivo.
- Na tabela 7 encontra-se os horários para as 3 séries do ensino médio. Nesta grade, cada dia contém 6 slots.
- Nas tabelas 8 e 9 encontram-se as alocações sob o ponto de vista dos professores da Grade 1, divididas em duas partes. Cada entrada na tabela representa a série e turma para a qual será lecionada a aula.
- Nas tabelas 10 e 11 encontram-se as alocações sob o ponto de vista dos professores da Grade 2, divididas em duas partes. Cada entrada na tabela representa a série para a qual será lecionada a aula.

Nesta solução, nenhuma restrição forte é violada, exceto 3 slots que deveriam ser coordenados, mas para os quais não foi encontrada solução. Eventuais janelas de professores que a primeira vista existem nas tabelas 8, 9, 10 e 9, quando analisadas, indicam na verdade ou indisponibilidades dos professores ou horários em que os mesmos lecionam para outra grade.

Ensino Fundamental - Quintas Séries			
Slot	**Turma A**	**Turma B**	**Turma C**
1	Computação / (monitor)	Educação Artística / Regina	Educação Física / Cristina
2	Educação Musical / Regina	Educação Física / Cristina	Português / Tarsila
3	Geografia / Arlette	Português / Tarsila	Educação Artística / Regina
4	Ciências / Cléia	Educação Musical / Regina	Matemática / David
5	Português / Tarsila	Ciências / Cléia	Matemática / David
6	Matemática / David	Inglês / Maria Luísa	Geografia / Arlette
7	Geografia / Arlette	Orientação Religiosa / Nazareth	História / Silmar
8	Orientação Religiosa / Nazareth	Matemática / David	Português / Tarsila
9	Ciências / Cléia	Português / Tarsila	Matemática / David
10	Português / Tarsila	Geografia / Arlette	Ciências / Cléia
11	Filosofia / Nazareth	Geografia / Arlette	Português / Tarsila
12	Educação Artística / Regina	Filosofia / Nazareth	Português / Tarsila
13	Inglês / Maria Luísa	Matemática / David	Educação Musical / Regina
14	Matemática / David	Português / Tarsila	Filosofia / Nazareth
15	História / Silmar	Inglês / Maria Luísa	Computação / Linus Torvalds
16	Matemática / David	Português / Tarsila	Geografia / Arlette
17	História / Silmar	Educação Física / Cristina	Orientação Religiosa / Nazareth
18	Educação Física / Cristina	História / Silmar	Inglês / Maria Luísa
19	Português / Tarsila	Ciências / Cléia	Matemática / David
20	Português / Tarsila	Matemática / David	Ciências / Cléia
21	Matemática / David	História / Silmar	Português / Tarsila
22	Matemática / David	Português / Tarsila	Inglês / Maria Luísa
23	Inglês / Maria Luísa	Matemática / David	Educação Física / Cristina
24	Português / Tarsila	Matemática / David	História / Silmar
25	Educação Física / Cristina	Computação / (monitor)	Matemática / David

Tabela 3: Exemplo de Solução - Horário das Quintas Séries

Ensino Fundamental - Sextas Séries			
Slot	**Turma A**	**Turma B**	**Turma C**
1	Português / Tarsila	Matemática / Janete	Português / Maria Cristina
2	Geografia / Arlette	Matemática / Janete	Português / Maria Cristina
3	Matemática / Janete	Ciências / Elizabeth	História / Silmar
4	Inglês / Maria Luísa	Português / Tarsila	Ciências / Elizabeth
5	Ciências / Elizabeth	História / Silmar	Geografia / Arlette
6	Português / Tarsila	Filosofia / Nazareth	Português / Maria Cristina
7	Português / Tarsila	Matemática / Janete	Português / Maria Cristina
8	Matemática / Janete	Ciências / Elizabeth	Matemática / Alexandre
9	Ciências / Elizabeth	Geografia / Arlette	Orientação Religiosa / Alexandre
10	Ciências / Elizabeth	Inglês / Maria Luísa	Educação Musical / Regina
11	Educação Musical / Regina	Matemática / Janete	Ciências / Elizabeth
12	Matemática / Janete	História / Silmar	Inglês / Maria Luísa
13	Matemática / Janete	Português / Tarsila	Filosofia / Nazareth
14	Inglês / Maria Luísa	Educação Musical / Regina	História / Silmar
15	Português / Tarsila	Geografia / Arlette	Matemática / Alexandre
16	Filosofia / Nazareth	Educação Artística / Edi	Educação Física / Cristina
17	Matemática / Janete	Português / Tarsila	Educação Artística / Edi
18	Educação Artística / Edi	Português / Tarsila	Matemática / Alexandre
19	História / Silmar	Educação Física / Cristina	Matemática / Alexandre
20	Educação Física / Cristina	Orientação Religiosa / Alexandre	Inglês / Maria Luísa
21	Educação Física / Cristina	Inglês / Maria Luísa	Português / Maria Cristina
22	História / Silmar	Educação Física / Cristina	Matemática / Alexandre
23	Português / Tarsila	Matemática / Janete	Geografia / Arlette
24	Orientação Religiosa / Alexandre	Ciências / Elizabeth	Educação Física / Cristina
25	Geografia / Arlette	Português / Tarsila	Ciências / Elizabeth

Tabela 4: Exemplo de Solução - Horário das Sextas Séries

Ensino Fundamental - Sétimas Séries		
Slot	**Turma A**	**Turma B**
1	Desenho / David	Geografia / Arlette
2	Educação Física / Ricardo	História / Silmar
3	Português / Maria Cristina	Educação Física / Jussara
4	História / Silmar	Português / Maria Cristina
5	Matemática / Regina Célia	Português / Maria Cristina
6	Ciências / Fernando	Filosofia / Silmar
7	Ciências / Fernando	Matemática / Regina Célia
8	Inglês / Maria Luísa	Matemática / Regina Célia
9	Matemática / Regina Célia	Ciências / Fernando
10	Matemática / Regina Célia	História / Silmar
11	Filosofia / Silmar	Desenho / David
12	Desenho / David	Geografia / Arlette
13	Geografia / Arlette	Matemática / Regina Célia
14	Português / Maria Cristina	Orientação Religiosa / Alexandre
15	Ciências / Fernando	Português / Maria Cristina
16	Matemática / Regina Célia	Inglês / Maria Luísa
17	Inglês / Maria Luísa	Desenho / David
18	Matemática / Regina Célia	Português / Maria Cristina
19	Português / Maria Cristina	Ciências / Fernando
20	Português / Maria Cristina	Ciências / Fernando
21	Orientação Religiosa / Alexandre	Educação Física / Jussara
22	Educação Física / Ricardo	Português / Maria Cristina
23	Português / Maria Cristina	Matemática / Regina Célia
24	Geografia / Arlette	Matemática / Regina Célia
25	História / Silmar	Inglês / Maria Luísa

Tabela 5: Exemplo de Solução - Horário das Sétimas Séries

Ensino Fundamental - Oitavas Séries			
Slot	**Turma A**	**Turma B**	**Turma C**
1	Física / Walkíria	Matemática / Regina Célia	Educação Física / Jussara
2	Desenho / David	Matemática / Regina Célia	Física / Walkíria
3	Português / Carmen	Desenho / David	Matemática / Regina Célia
4	Matemática / Regina Célia	Geografia / Arlette	Português / Carmen
5	Inglês / Maria Luísa	Português / Carmen	Orientação Religiosa / Alexandre
6	Matemática / Regina Célia	Educação Física / Jussara	Física / Walkíria
7	Química / Paulo	Desenho / David	Inglês / Maria Luísa
8	História / Silmar	Português / Carmen	Geografia / Arlette
9	Português / Carmen	Inglês / Maria Luísa	História / Silmar
10	Desenho / David	Orientação Religiosa / Alexandre	Português / Carmen
11	Educação Física / Ricardo	Matemática / Regina Célia	Inglês / Maria Luísa
12	Matemática / Regina Célia	Física / Walkíria	Educação Física / Jussara
13	Física / Walkíria	História / Silmar	Português / Carmen
14	Geografia / Arlette	Português / Carmen	Matemática / Regina Célia
15	Português / Carmen	Matemática / Regina Célia	Desenho / David
16	História / Silmar	Física / Walkíria	Química / Paulo
17	Química / Paulo	Matemática / Regina Célia	Geografia / Arlette
18	Português / Carmen	Química / Paulo	Desenho / David
19	Inglês / Maria Luísa	Português / Carmen	Matemática / Regina Célia
20	Matemática / Regina Célia	História / Silmar	Português / Carmen
21	Geografia / Arlette	Química / Paulo	Matemática / Regina Célia
22	Matemática / Regina Célia	Geografia / Arlette	Química / Paulo
23	Português / Carmen	Educação Física / Jussara	História / Silmar
24	Educação Física / Ricardo	Inglês / Maria Luísa	Português / Carmen
25	Orientação Religiosa / Alexandre	Português / Carmen	Matemática / Regina Célia

Tabela 6: Exemplo de Solução - Horário das Oitavas Séries

Ensino Médio			
Slot	**Primeiro Ano**	**Segundo Ano**	**Terceiro Ano**
1	Matemática / Orlando	Literatura / Helena	Química / Antônio
2	Matemática / Orlando	Química / Antônio	Educação Física / Ricardo
3	Física / Walkíria	Geografia / Maria José	Matemática / Orlando
4	Física / Walkíria	Biologia / Fernando	Matemática / Orlando
5	Biologia / Fernando	Física / Walkíria	História / Nero
6	História / Nero	Artes / Mônica	Biologia / Fernando
7	Química / Paulo	Orientação Religiosa / Alexandre	Literatura / Maria Elvira
8	Gramática e Redação / Maria Elvira	Física / Walkíria	Orientação Religiosa / Alexandre
9	Gramática e Redação / Maria Elvira	Biologia / Fernando	Física / Walkíria
10	História / Nero	Matemática / Orlando	Física / Walkíria
11	Física / Walkíria	Química / Antônio	Matemática / Orlando
12	Geografia / Maria José	Educação Física / Cristina	Química / Antônio
13	Química / Paulo	Física / Walkíria	Química / Antônio
14	Química / Paulo	Química / Antônio	História / Nero
15	Biologia / Fernando	História / Nero	Matemática / Orlando
16	Literatura / Helena	Matemática / Orlando	Biologia / Fernando
17	Matemática / Orlando	Gramática e Redação / Helena	Gramática e Redação / Maria Elvira
18	Matemática / Orlando	Gramática e Redação / Helena	Gramática e Redação / Maria Elvira
19	Literatura / Helena	Biologia / Fernando	Geografia / Maria José
20	Literatura / Helena	Física / Walkíria	Biologia / Fernando
21	Física / Walkíria	Química / Antônio	Biologia / Fernando
22	Química / Paulo	Matemática / Orlando	Física / Walkíria
23	Inglês / Marco Antônio	Matemática / Orlando	História / Nero
24	Gramática e Redação / Maria Elvira	Inglês / Marco Antônio	Matemática / Orlando
25	Biologia / Fernando	Gramática e Redação / Helena	Gramática e Redação / Maria Elvira
26	Biologia / Fernando	Literatura / Helena	Literatura / Maria Elvira
27	Orientação Religiosa / Alexandre	Biologia / Fernando	Inglês / Marco Antônio
28	Geografia / Maria José	Matemática / Orlando	Física / Walkíria
29	Matemática / Orlando	Geografia / Maria José	Química / Antônio
30	Educação Física / Ricardo	História / Nero	Geografia / Maria José

Tabela 7: Exemplo de Solução - Horário do Ensino Médio

B Dados do Colégio "Stella Maris"

Este anexo contém as especificações gerais para a montagem do horário das séries e professores do colégio "Stella Maris", dadas cargas horárias, restrições gerais e específicas.

B.1 Carga Horária

A seguir é definida a carga horária de todas as séries. As séries do ensino fundamental tem entrada às 7:30 e saída às 12:10 de segunda à sexta, exceto para a sexta série, que tem como horário de saída 13:00 na segunda-feira e 12:10 nos demais dias. As séries do ensino médio tem entrada às 7:10 e saída às 12:30 de segunda à sexta.

A hora-aula é de 50 minutos e os intervalos são de 30 minutos para o ensino fundamental e de 20 minutos para o ensino médio, sendo os horários:

- Das 9:10 às 9:40 para quintas e sextas séries do ensino fundamental
- Das 9:40 às 10:00 para as séries do ensino médio
- Das 10:00 às 10:30 para sétimas e oitavas séries do ensino fundamental

Horário dos Professores - Ensino Fundamental															
Nome/Slot	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Alexandre					8 C			6 C	6 C	8 B				7 B	6 C
Arlette	7 B	6 A	5 A	8 B	6 C	5 C	5 A	8 C	6 B	5 B	5 B	7 B	7 A	8 A	6 B
Carmen			8 A	8 C	8 B			8 B	8 A	8 C			8 C	8 B	8 A
Cléia				5 A	5 B				5 A	5 C					
Cristina	5 C	5 B													
David	7 A	8 A	8 B	5 C	5 C	5 A	8 B	5 B	5 C	8 A	7 B	7 A	5 B	5 A	8 C
Edi															
Elizabeth			6 B	6 C	6 A			6 B	6 A	6 A	6 C				
Fernando						7 A	7 A		7 B						7 A
Janete	6 B	6 B	6 A				6 B	6 A			6 B	6 A	6 A		
Jussara	8 C		7 B			8 B						8 C			
Maria Cristina	6 C	6 C	7 A	7 B	7 B	6 C	6 C							7 A	7 B
Maria Luísa				6 A	8 A	5 B	8 C	7 A	8 B	6 B	8 C	6 C	5 A	6 A	5 B
Nazareth						6 B	5 B	5 A			5 A	5 B	6 C	5 C	
Paulo							8 A								
Regina	5 B	5 A	5 C	5 B						6 C	6 A	5 A	5 C	6 B	
Regina Célia	8 B	8 B	8 C	8 A	7 A	8 A	7 B	7 B	7 A	7 A	8 B	8 A	7 B	8 C	8 B
Ricardo		7 A									8 A				
Silmar		7 B	6 C	7 A	6 B	7 B	5 C	8 A	8 C	7 B	7 A	6 B	8 B	6 C	5 A
Tarsila	6 A	5 C	5 B	6 B	5 A	6 A	6 A	5 C	5 B	5 A	5 C	5 C	6 B	5 B	6 A
Walkíria	8 A	8 C				8 C						8 B	8 A		

Tabela 8: Exemplo de Solução - Horário dos Professores do Ensino Fundamental - Segunda a Quarta

Horário dos Professores - Ensino Fundamental										
Nome	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
Alexandre			6 C	6 C	6 B	7 A	6 C		6 A	8 A
Arlette	5 C	8 C				8 A	8 B	6 C	7 A	6 A
Carmen			8 A	8 B	8 C			8 A	8 C	8 B
Cléia				5 B	5 C					
Cristina	6 C	5 B	5 A	6 B	6 A	6 A	6 B	5 C	6 C	5 A
David	5 A	7 B	8 C	5 C	5 B	5 A	5 A	5 B	5 B	5 C
Edi	6 B	6 C	6 A							
Elizabeth									6 B	6 C
Fernando				7 B	7 B					
Janete		6 A						6 B		
Jussara						7 B		8 B		
Maria Cristina			7 B	7 A	7 A	6 C	7 B	7 A		
Maria Luísa	7 B	7 A	5 C	8 A	6 C	6 B	5 C	5 A	8 B	7 B
Nazareth	6 A	5 C								
Paulo	8 C	8 A	8 B			8 B	8 C			
Regina										
Regina Célia	7 A	8 B	7 A	8 C	8 A	8 C	8 A	7 B	7 B	8 C
Ricardo							7 A		8 A	
Silmar	8 A	5 A	5 B	6 A	8 B	5 B	6 A	8 C	5 C	7 A
Tarsila	5 B	6 B	6 B	5 A	5 A	5 C	5 B	6 A	5 A	6 B
Walkíria	8 B									

Tabela 9: Exemplo de Solução - Horário dos Professores do Ensino Fundamental - Quinta e Sexta

Para a montagem do horário é necessário ressaltar que aulas duplas são permitidas apenas para matérias cuja carga horária semanal exceda 2 horas aula semanais. Não é permitido mais do que 2 aulas de uma mesma matéria em um mesmo dia e se a carga

Horário dos Professores - Ensino Médio																	
Nome	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Alexandre							2	3										
Antônio	3	2									2	3	3	2				
Cristina												2						
Fernando				2	1	3			2						1	3		
Helena	2															1	2	2
Marco Antônio																		
Maria Elvira							3	1	1								3	3
Maria José			2									1						
Mônica						2												
Nero					3	1				1				3	2			
Orlando	1	1	3	3						2	3				3	2	1	1
Paulo							1						1	1				
Ricardo		3																
Walkíria			1	1	2			2	3	3	1		2					

Tabela 10: Exemplo de Solução - Horário dos Professores do Ensino Médio - Segunda a Quarta

Horário dos Professores - Ensino Médio												
Nome	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Alexandre									1			
Antônio			2								3	
Cristina												
Fernando	2	3	3				1	1	2			
Helena	1	1					2	2				
Marco Antônio					1	2			3			
Maria Elvira						1	3	3				
Maria José	3									1	2	3
Mônica												
Nero					3							2
Orlando				2	2	3				2	1	
Paulo				1								
Ricardo												1
Walkíria		2	1	3						3		

Tabela 11: Exemplo de Solução - Horário dos Professores do Ensino Médio - Quinta e Sexta

horária semanal for inferior a 3 horas aula semanais, este limite cai para 1 aula em um dia. As aulas de esforço mental mais baixo (Educação Artística, Computação, Educação Musical, Educação Física e Orientação Religiosa) devem, de preferência, serem distribuídas eqüitativamente na semana.

Cabe lembrar que a carga horária é, de certa forma, fixa, mudando muito pouco ou quase nada de ano para ano, ao contrário do que acontece com a disponibilidade dos professores.

Quinta série do ensino fundamental

A carga horária para as 3 turmas da quinta série do ensino fundamental é de 25 horas aula semanais, compostas de:

- **Português**: 5 horas aula semanais
- **Matemática**: 5 horas aula semanais

- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Ciências**: 2 horas aula semanais
- **Inglês**: 2 horas aula semanais
- **Educação Física**: 2 horas aula semanais
- **Filosofia**: 1 hora aula semanal
- **Educação Artística**: 1 hora aula semanal
- **Educação Musical**: 1 hora aula semanal
- **Computação**: 1 hora aula semanal
- **Orientação Religiosa**: 1 hora aula semanal

Sexta série do ensino fundamental

A carga horária para as 3 turmas da sexta série do ensino fundamental é de 25 horas aula semanais, compostas de:

- **Português**: 5 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Ciências**: 3 horas aula semanais
- **Inglês**: 2 horas aula semanais
- **Educação Física**: 2 horas aula semanais
- **Filosofia**: 1 hora aula semanal
- **Educação Artística**: 1 hora aula semanal
- **Educação Musical**: 1 hora aula semanal
- **Orientação Religiosa**: 1 hora aula semanal

Sétima série do ensino fundamental

A carga horária para as 2 turmas da sétima série do ensino fundamental é de 25 horas aula semanais, compostas de:

- **Português**: 5 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Ciências**: 3 horas aula semanais
- **Inglês**: 2 horas aula semanais
- **Educação Física**: 2 horas aula semanais
- **Filosofia**: 1 hora aula semanal
- **Desenho**: 2 horas aulas semanais
- **Orientação Religiosa**: 1 hora aula semanal

Oitava série do ensino fundamental

A carga horária para as 3 turmas da oitava série do ensino fundamental é de 25 horas aula semanais, compostas de:

- **Português**: 5 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Física**: 2 horas aula semanais
- **Química**: 2 horas aula semanais
- **Inglês**: 2 horas aula semanais
- **Educação Física**: 2 horas aula semanais
- **Desenho**: 2 horas aulas semanais
- **Orientação Religiosa**: 1 hora aula semanal

Primeira série do ensino médio

A carga horária para a turma da primeira série do ensino médio é de 30 horas aula semanais, compostas de:

- **Literatura**: 3 horas aula semanais
- **Gramática e Redação**: 3 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Física**: 4 horas aula semanais
- **Química**: 4 horas aula semanais
- **Biologia**: 4 horas aula semanais
- **Inglês**: 1 hora aula semanal
- **Educação Física**: 1 hora aula semanal
- **Orientação Religiosa**: 1 hora aula semanal

Segunda série do ensino médio

A carga horária para a turma da segunda série do ensino médio é de 30 horas aula semanais, compostas de:

- **Literatura**: 2 horas aula semanais
- **Gramática e Redação**: 3 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 2 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Física**: 4 horas aula semanais
- **Química**: 4 horas aula semanais
- **Biologia**: 4 horas aula semanais
- **Inglês**: 1 hora aula semanal
- **Educação Física**: 1 hora aula semanal
- **Orientação Religiosa**: 1 hora aula semanal
- **Artes**: 1 hora aula semanal

Terceira série do ensino médio

A carga horária para a turma da terceira série do ensino médio é de 30 horas aula semanais, compostas de:

- **Literatura**: 2 horas aula semanais
- **Gramática e Redação**: 3 horas aula semanais
- **Matemática**: 5 horas aula semanais
- **História**: 3 horas aula semanais
- **Geografia**: 2 horas aula semanais
- **Física**: 4 horas aula semanais
- **Química**: 4 horas aula semanais
- **Biologia**: 4 horas aula semanais
- **Inglês**: 1 hora aula semanal
- **Educação Física**: 1 hora aula semanal
- **Orientação Religiosa**: 1 hora aula semanal

B.2 Professores

O colégio "Stella Maris" possui diversos professores. Alguns ministram mais do que uma matéria e/ou para diversas séries diferentes. A maioria possui restrições de horário próprias. No caso da falta de um professor, a substituição é feita por um funcionário, orientador ou outro professor disponível. Não existe professor substituto imediato. As informações contidas nesta seção sofrem várias mudanças de ano para ano.

É proibido dar "janelas" para os professores, ou seja, um professor ficar desocupado durante uma ou mais horas aulas entre duas de suas aulas em um mesmo dia.

Tarsila

Ministra aulas de Português para as quintas séries, e para as sextas A e B do ensino fundamental. Não tem restrições de horário.

Maria Cristina

Ministra aulas de Português para as sétimas séries e para a sexta série C do ensino fundamental. Esta professora deseja ter um dia da semana sem aulas, de preferência terça-feira.

Carmen

Ministra aulas de Português para as oitavas séries do ensino fundamental. Somente pode dar aulas nos três últimos horários (a partir das 9:10).

Helena

Ministra aulas de Literatura para a primeira e segunda séries do ensino médio e de Gramática e Redação para a segunda série do ensino médio. Pode dar aulas somente nas segundas, terças e quartas. Existe a possibilidade de dar aulas na quinta-feira.

Maria Elvira

Ministra aulas de Literatura para a terceira série do ensino médio e de Gramática e Redação para a primeira e terceira séries do ensino médio.[13]. Não tem restrições de horário.

Orlando

Ministra aulas de Matemática para as três séries do ensino médio. Não tem restrições de horário.

David

Ministra aulas de Matemática para as quintas séries do ensino fundamental e de Desenho para as sétimas e oitavas séries do ensino fundamental. Não tem restrições de horário.

Janete

Ministra aulas de Matemática para as sextas séries A e B do ensino fundamental. Pode dar aulas qualquer dia, mas sempre nas três primeiras aulas. De preferência deve ter um dia livre.

Regina Célia

Ministra aulas de Matemática para as sétimas e oitavas séries do ensino fundamental. Não tem restrições de horário.

Alexandre

Ministra aulas de Matemática para a sexta série C do ensino fundamental e de Orientação Religiosa das sextas às oitavas séries do ensino fundamental e para as três séries do ensino médio. Sua única restrição de horário é que deve ter horário livre às sextas-feiras das 9:10 às 10:00 para acompanhar a missa realizada no próprio colégio. Isto não é considerado uma janela.

[13] As professoras Helena e Maria Elvira devem ter uma aula de Gramática e Redação de segunda e terceira série em um mesmo horário, pois revezam as turmas

Silmar

Ministra aulas de História para todas as séries do ensino fundamental e de Filosofia para as sétimas séries do ensino fundamental. Não tem restrições de horário.

Nero

Ministra aulas de História para as três séries do ensino médio. Somente pode dar aulas nos dois primeiros horários de segunda e sexta-feira, nos três últimos de terça feira e em qualquer horário de quinta-feira.

Arlette

Ministra aulas de Geografia para todas as séries do ensino fundamental. Em um dos dias da semana só pode ministrar as duas primeiras aulas.

Maria José

Ministra aulas de Geografia para as três séries do ensino médio. Somente pode dar aulas nos 3 primeiros horários de terça, nos três últimos de quinta e no primeiro de sexta.

Cléia

Ministra aulas de Ciências para as quintas séries do ensino fundamental. Pode ministrar aulas qualquer dia a partir de 9:40 exceto sexta-feira.

Elizabeth

Ministra aulas de Ciências para as sextas séries do ensino fundamental. Somente pode dar aulas dois dias da semana.

Fernando

Ministra aulas de Ciências para as sétimas séries do ensino fundamental e de Biologia para as três séries do ensino médio. Deve ter uma manhã livre, e, em outra manhã três horas aula livres.

Paulo

Ministra aulas de Química para as oitavas séries do ensino fundamental e para a primeira série do ensino médio. Não pode dar aulas segundas e quartas, nem no último horário de aula.

Antônio

Ministra aulas de Química para as segunda e terceira séries do ensino médio. Somente pode dar aulas nas quartas e sextas.

Walkíria

Ministra aulas de Física para as oitavas séries do ensino fundamental e para as três séries do ensino médio. Não pode dar aulas nas sextas-feiras e somente pode dar a primeira aula nas terças-feiras.

Maria Luísa

Ministra aulas de Inglês para todas as séries do ensino fundamental. Não tem restrições de horário.

Marco Antônio

Ministra aulas de Inglês para todas as séries do ensino médio. Somente pode dar aulas nas quintas.

Nazareth

Ministra aulas de Orientação Religiosa para as quintas séries do ensino fundamental e de Filosofia para as quintas e sextas séries do ensino fundamental. Não ministra aulas segunda nem sexta.

Regina

Ministra aulas de Educação Artística para as quintas séries do ensino fundamental e de Educação Musical para as quintas e sextas séries do ensino fundamental. Só pode dar aulas nas segundas, terças e quartas.

Edi

Ministra aulas de Educação Artística para as sextas séries do ensino fundamental. Somente pode dar aulas nos três primeiros horários de quinta-feira.

Mônica

Ministra aulas de Artes para a segunda série do ensino médio. Somente pode dar aulas nas quartas-feiras.

Cristina

Ministra aulas de Educação Física para as quintas e sextas séries do ensino fundamental e para o sexo feminino das três séries do ensino médio. Somente pode dar aulas segunda, quinta e sexta-feira.

Ricardo

Ministra aulas de Educação Física para o sexo masculino nas sétimas e oitavas séries do ensino fundamental e nas três séries do ensino médio. Não tem restrições de horário.

Jussara

Ministra aulas de Educação Física para o sexo feminino nas sétimas e oitavas séries do ensino fundamental. Somente pode dar aulas em dois dias da semana.

B.3 Educação Física

Educação Física é uma matéria com algumas exceções que devem ser levadas em consideração:

- Os sexos feminino e masculino de uma classe tem aula em um mesmo horário, embora com professores diferentes muitas vezes.
- As oitavas séries A e B do ensino fundamental compartilham a mesma aula, o mesmo acontece com as primeira e segunda séries do ensino médio.

Referências

[BBM93] David BEASLEY, David R. BULL, and Ralph R. MARTIN. An overview of genetic algorithms. *University Computing*, 1993.

[BEW95] Edmund K. BURKE, Dave G. ELLIMAN, and Rupert F. WEARE. A hybrid genetic algorithm for highly constrained timetabling problems. Technical report, University of Nottingham, UK - Department of Computer Science, 1995.

[CDM90a] Alberto COLORNI, Marco DORIGO, and Vittorio MANIEZZO. Genetic algorithms: a new approach to the time-table problem. *Lecture Notes in Computer Science - NATO ASI Series, Vol F 82*, 1990.

[CDM90b] Alberto COLORNI, Marco DORIGO, and Vittorio MANIEZZO. Genetic algorithms and highly constrained problems: the time-table case. *Proceedings of the First Int. Workshop on Parallel Problem Solving for Nature*, 1990.

[CDM90c] Alberto COLORNMI, Marco DORIGO, and Vittorio MANIEZZO. A genetic algorithm to solve the timetable problem. Technical report, Politecnico de Milano - Dipartimento di Elettronica, 1990.

[CL98] Michael W. CARTER and Gilbert LAPORTE. Recent development in practical course timetabling. Technical report, University of Toronto e École des Hautes Études Commerciales de Montréal, 1998.

[FAN94] Hsiao-Lan FANG. *Genetic Algorithms in Timetabling and Scheduling.* PhD thesis, University of Edinburgh - Department of Artificial Intelligence, 1994.

[HOL75] John HOLLAND. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[KER97] Aaron KERSHENBAUM. When genetic algorithms work best. *INFORMS Journal on Computing*, 1997.

[LRF01] Luiz A. N LORENA and Geraldo RIBEIRO FILHO. A constructive evolutionary approach to school timetabling. In *EvoWorkshops*, pages 130–139, 2001.

[MIC96] Zbigniew MICHALEWICZ. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 3 edition, 1996.

[RCF94] Peter ROSS, Dave CORNE, and Hsiao-Lan FANG. Fast practical evolutionary timetabling. *Proceedings of AISB Workshop on Evolutionary Computation*, 1994.

[RCF95] Peter ROSS, Dave CORNE, and Hsiao-Lan FANG. Successful lecture timetabling with evolutionary algorithms. In A. E. Eiben, B. Manderick, and Zs. Ruttkay, editors, *Applied Genetic and other Evolutionary Algorithms: Proceedings of the ECAI'94 Workshop.* Springer, Berlin, 1995.

[REE97] Colin R. REEVES. Genetic algorithm for the operations researcher. *INFORMS Journal on Computing*, 1997.

[SCH95] Andrea SCHAERF. A survey of automated timetabling. Technical report, Centrum voor Wiskunde en Informatica - Department os Software Technology, 1995.

[WHI93] Darrell WHITLEY. A genetic algorithm tutorial. Technical report, Colorado State University - Department of Computer Science, 1993.

[YS02] Enzhe YU and Ki-Seok SUNG. A genetic algorithm for a university weekly courses timetabling problem. *International Transactions in Operational Research*, 2002.

Índice Remissivo