Elaboração de Grades Horárias Utilizando Algoritmos Genéticos

Lucas Bucior, Fabio Asturian Zanin, Marcos A. Lucas

Departamento de Engenharias e Ciência da Computação
Universidade Regional Integrada do Alto Uruguai e das Missões (URI)
Caixa Postal 743 – 99.709-910 – Erechim, RS – Brazil

051167@aluno.uricer.edu.br, {faz, mlucas}@uricer.edu.br

Abstract*. The task of producing hourly grids in an educational institution is not easy because of significant limitations on the adequacy of the responses produced. Artificial Intelligence techniques have been successfully applied, more specifically the methods of Evolutionary Computing. Evolutionary Computing defines a class algorithm that computes the concepts of Charles Darwin's theory of evolution. The main objective of this work is to present a solution for the construction of hourly grids based on Evolutionary Computing.*

Resumo. *A tarefa de produzir grades horárias em uma instituição educacional não é fácil por causa de limitações significativas na adequação das respostas produzidas. Técnicas de Inteligência Artificial têm sido aplicadas com sucesso, mais especificamente os métodos da Computação Evolutiva. A Computação Evolutiva define um algoritmo de classe que calcula os conceitos da teoria da evolução de Charles Darwin. O principal objetivo deste trabalho é apresentar uma solução para a construção de grades horárias baseados em Computação Evolutiva.*

1. Introdução

De acordo com [GOLDBERG 1989], Charles Darwin estudou as espécies e sua evolução durante anos, coletou uma grande quantidade de material que mostrou a existência de muitas variações genéticas em cada espécie. Outras pesquisas relacionadas deixaram claro que as espécies animais realmente mudam. Seleção natural, uma competição por recursos necessários para a sobrevivência dos mais aptos.

Baseado na teoria evolucionista de Darwin foi desenvolvida a Computação Evolutiva, que propõe modelar computacionalmente as regras básicas da teoria da evolução natural. Em essência, os métodos evolucionários trabalham com uma população de indivíduos. A população de indivíduos é processada em um ciclo evolutivo visando alcançar uma solução ideal entre os candidatos, melhorando gradualmente os cromossomos de cada indivíduo. Em função dos recursos oferecidos pela abordagem evolutiva, a principal aplicação é dada aos problemas de otimização numérica.

Os problemas classificados como otimização numérica consistem de uma função objetivo que precisa ser otimizada e restrições impostas no espaço de pesquisa. A solução do problema é, portanto, um ou vários pontos ótimos no espaço. O problema de otimização a ser analisado neste trabalho é conhecido com *timetabling*. Esse problema

apresenta várias restrições e um grande número de variáveis que devem ser atendidas na obtenção da solução ideal.

A principal motivação para usar a abordagem evolutiva é baseada no argumento de [CONCILIO 2000], fundamenta que, para problemas complexos de otimização, há uma explosão combinatória de candidatos à solução dentro de um espaço de busca. Desta forma, buscas exaustivas promovidas por métodos convencionais não conseguiriam alcançar uma solução ótima, caracterizando o problema como intratável computacionalmente.

O objetivo deste trabalho é apresentar uma solução para a construção de grades horárias, levando em consideração a preferência dos professores, turnos, número de horários por dia e dias da semana, com base na Computação Evolutiva, além de apresentar uma interface amigável para gerenciar as turmas, disciplinas, cursos e professores.

2. Definições de *Timetabling*

Segundo [ROSS et al. 1999], *timetabling* é essencialmente um problema de escalonamento de eventos e cada evento possui restrições sobre a aplicabilidade em um período de tempo determinado.

Timetabling pode ser aplicado a diversos contextos, e para isso basta modificar as variáveis e as restrições envolvidas no problema. Algumas variações de *timetabling* podem ser descritas como: escalonamento de funcionários em turnos, remanejamento de máquinas em fábrica, tabelas de horários ou recursos (exames, salas de aula, entre outros) para escolas ou para universidades. Assim, o problema é caracterizado por uma natureza fixa e um conjunto de restrições variáveis.

Segundo [ROSS et al. 2000], um *timetable* genérico pode ser definido por três conjuntos básicos. São eles:

- $E = \{ e_1 , e_2 , e\square \}$, ou seja, um conjunto finito de eventos que inclui atividades diversas como exames, seminários, projetos ou aulas.
- $T = \{ t_1 , t_2 , t\square \}$, que é um conjunto finito de horários, para realização dos eventos.
- $A = \{ a_1 , a_2 , a\square \}$, que implica em um conjunto finito de agentes (instrutores, monitores ou professores), que têm o papel de monitorar eventos particulares.

Conclui-se, portanto, que, com base nessas definições anteriores, um conjunto de elementos ou uma coleção de eventos está envolvido em um contexto de *timetabling* usando o conjunto $\{ E, T, A \}$, que representam respectivamente as atividades, horários e os agentes, interpretados como: o evento E inicia no tempo T e tem como agente A. As escalas produzidas não devem violar um conjunto de restrições pré-definidas pelo contexto.

3. Computação Evolutiva

Computação Evolutiva é um conjunto de técnicas de busca e otimização inspiradas na evolução natural das espécies. Segundo [NARDIN 1999], a natureza de cada organismo possui cromossomos, genes, éxons, íntrons e códons, constituindo o sistema genético. Um certo grupo de indivíduos vive junto, o que forma uma população. Nesta população,

há organismos melhores, mais propensos a gerar bons descendentes. Esses descendentes estão melhor adaptados, porque tiveram genes melhores. No final, vence a lógica de sobreviver aos mais adaptados ao nicho ecológico da população. Este sistema de escolha é transferido para gerações subsequentes, que melhoram cada vez mais as populações.

Segundo [CONCILIO 2000], o termo Computação Evolutiva surgiu em 1991. Representa uma tentativa de unir pesquisadores que trabalham com simulações de cálculos no campo da evolução.

Desde 1970, várias metodologias evolutivas têm sido propostas, tais como: Algoritmos Genéticos, Estratégias Evolutiva e Programação Evolutiva. Essas abordagens trabalham com um conjunto de soluções candidatas e alteram esses conjuntos aplicando dois princípios evolutivos básicos: seleção que representa a competitividade de recursos entre seres vivos onde é mais adequada à sobrevivência e à variação, que é a capacidade de propagação de novos seres através de recombinação e mutação.

Computacionalmente, estes processos naturais são apresentados na forma de operadores genéticos, instrumentos de avaliação, descrevendo a aptidão ou adequação e métodos seletivos para os indivíduos da população. Estes processos são utilizados para promover o ciclo evolutivo.

3.1. Algoritmos Genéticos

Os Algoritmos Genéticos foram desenvolvidos pela primeira vez pelo professor John Holland, da Universidade de Michigan, nos EUA, em suas explorações de processos adaptativos de sistemas naturais e sua possível aplicabilidade em projetos de software de sistemas artificiais. O método foi formalmente introduzido em seu livro *Adaptation in natural and artificial systems* [HOLLAND 1992].

Segundo [GOLDBERG 1989], a teoria da evolução prediz que o ambiente de cada geração seleciona os seres vivos mais adaptados para a sobrevivência. Durante a existência dos indivíduos, ocorrem fenômenos como mutação e cruzamento que atuam no material genético armazenado nos cromossomos.

Em Algoritmos Genéticos, os processos são expressos como uma metáfora para esses fenômenos, explicando assim a existência de muitos termos oriundos da biologia.

3.1.1. Inicialização da População

Segundo [CONCILIO 2000] o método mais utilizado para inicialização da população é a geração aleatória de indivíduos. No entanto, o conhecimento do problema pode ser usado para melhorar o desempenho dos indivíduos já na geração inicial. Para problemas com restrições, deve-se tomar cuidado para não gerar indivíduos inválidos. Este fato pode atrasar o processo evolutivo de uma população.

3.1.2. Métodos de Seleção

É necessário, após definir a população, escolher o método que promoverá a seleção, para gerar descendentes. A proposta deste processo é privilegiar os melhores indivíduos da população. [CONCILIO 2000]

3.1.3. Operadores Genéticos

Operadores genéticos ampliam a busca até que alcancem um resultado satisfatório. Esses operadores alteram a população ao longo de gerações e são vitais para a diversificação da população, mantendo as características adaptativas adquiridas pelas gerações anteriores. Os operadores genéticos são: mutação e cruzamento. [CONCILIO 2000]

3.1.4. Operador de Avaliação

O operador de avaliação atribui uma nota do nível de cooperação de uma espécie na área que está inserida. Para calcular este valor, representantes de diferentes espécies devem ser escolhidos e uma função de *fitness* é aplicada aos indivíduos. [CONCILIO 2000]

3.1.5. Operador de Cruzamento

O operador de cruzamento é um operador binário, usado em indivíduos escolhidos pelo operador de seleção, que promove a troca de genes ou informações entre eles para gerar novos indivíduos. Assim, cada par de cromossomos dá origem a novos indivíduos, que formarão a população da próxima geração. [CONCILIO 2000]

3.1.6. Operadores de Mutação

Os operadores de mutação são necessários para a introdução e manutenção da diversidade genética da população. A mutação oferece os meios para introduzir novos elementos, promovendo a diversidade e a variabilidade extra da população, sem interromper o progresso já alcançado pelo algoritmo genético. O operador de mutação substitui aleatoriamente um ou mais genes de um cromossomo. [CONCILIO 2000]

4. Resultados Obtidos

O esquema do algoritmo evolutivo descrito na Figura 1 é baseado nos conceitos descritos na seção 3.1 e a evolução natural é representada pelo processo computacional iterativo. Primeiro, uma população inicial é gerada, na sequência é aplicado um ajuste para garantir que todos os indivíduos da população sejam válidos. Depois, todos os indivíduos serão avaliados atribuindo um *fitness*. Até que a condição de parada seja satisfeita, o laço que consiste em etapas de seleção, recombinação, mutação e avaliação será executado várias vezes. As iterações são chamadas de gerações da população.

```
inicializar P(t)
ajustar P(t)
avaliar P(t)
enquanto não ( condição de parada ) faça
        t = t + 1
        selecionar P(t) a partir de P(t - 1)
        alterar P(t)
        avaliar P(t)
fim
```

Figura 9. Esquema representando o algoritmo evolutivo.

A Figura 2 representa a construção das informações necessárias para gerar a grade horária. A grade horária possui turmas, uma certa turma é de um curso, um curso possui semestres e cada semestre tem uma quantidade de disciplinas, divididas em turnos. Na Figura 2, a turma "CC2017" é do segundo semestre do curso "Computer Science" e possui vinte créditos divididos em cinco disciplinas e cada disciplina está

associada a um professor. Depois de ajustar as informações, na sequência, gerar a grade horária e visualizar uma possível solução para as informações contidas na instância.

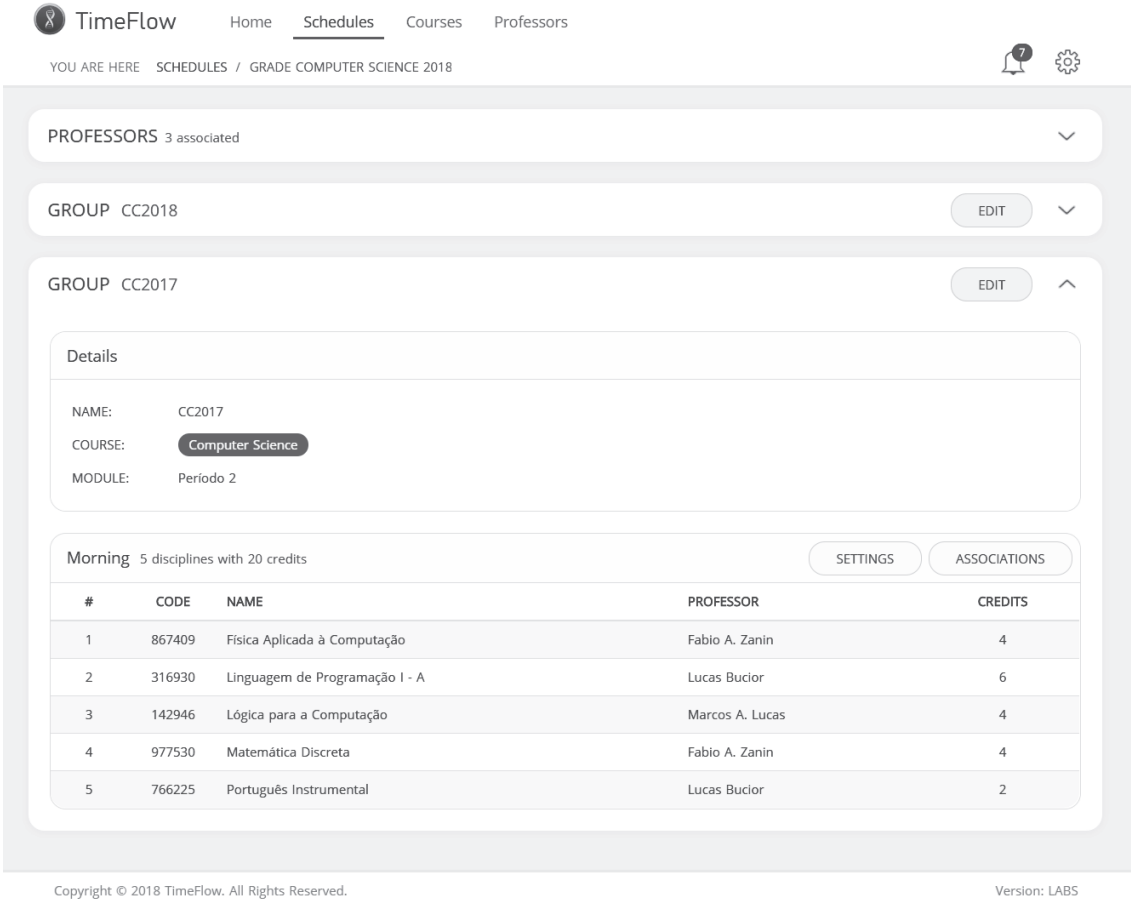

Figura 10. Interface para construção das informações da grade horária.

Na Figura 3, toda informação contida na construção da grade é organizada, possível solução. Os créditos das disciplinas são organizados de forma adjacente e um professor tem no máximo quatro créditos por turma em cada dia da semana. Lembrando que essa explicação é para este exemplo e essas configurações são realizadas na etapa de construção das informações Figura 2.

Em cada geração é possível visualizar a quantidade de colisões de professores, o *fitness* para essa geração em porcentagem (de 0 a 100, indicando respectivamente a pior e a melhor solução). Quando todos os critérios possíveis são atendidos, obtém-se 100% de *fitness*. Para obter um *fitness* 100%, com base nessa instância Figura 2, um professor não pode dar aula no mesmo horário para duas turmas, deve conter no máximo quatro créditos por turma em cada dia da semana, às preferências devem ser atendidas e os créditos devem estar adjacentes se possível.

Em toda construção das informações Figura 2, são feitas validações. Por exemplo, um professor não pode ter mais créditos que a turma possui. Alguns casos específicos ocorre a colisão de professores, por exemplo, um determinado professor possui quinze créditos na turma A e onze créditos na turma B, logicamente para esse professor não tem como organizar a grade, por possuir vinte e seis créditos e as duas

turmas possuírem vinte créditos cada. Para garantir que o algoritmo funcione corretamente as informações da instância são validadas.

A validação das informações presentes na construção da grade Figura 2, proporciona uma compilação inicial, informando o que precisa ser alterado, possíveis recomendações e redundâncias, antes da aplicação definitiva do algoritmo.

Em toda geração de horários é possível criar um PDF contendo às informações pertinentes a grade horária.

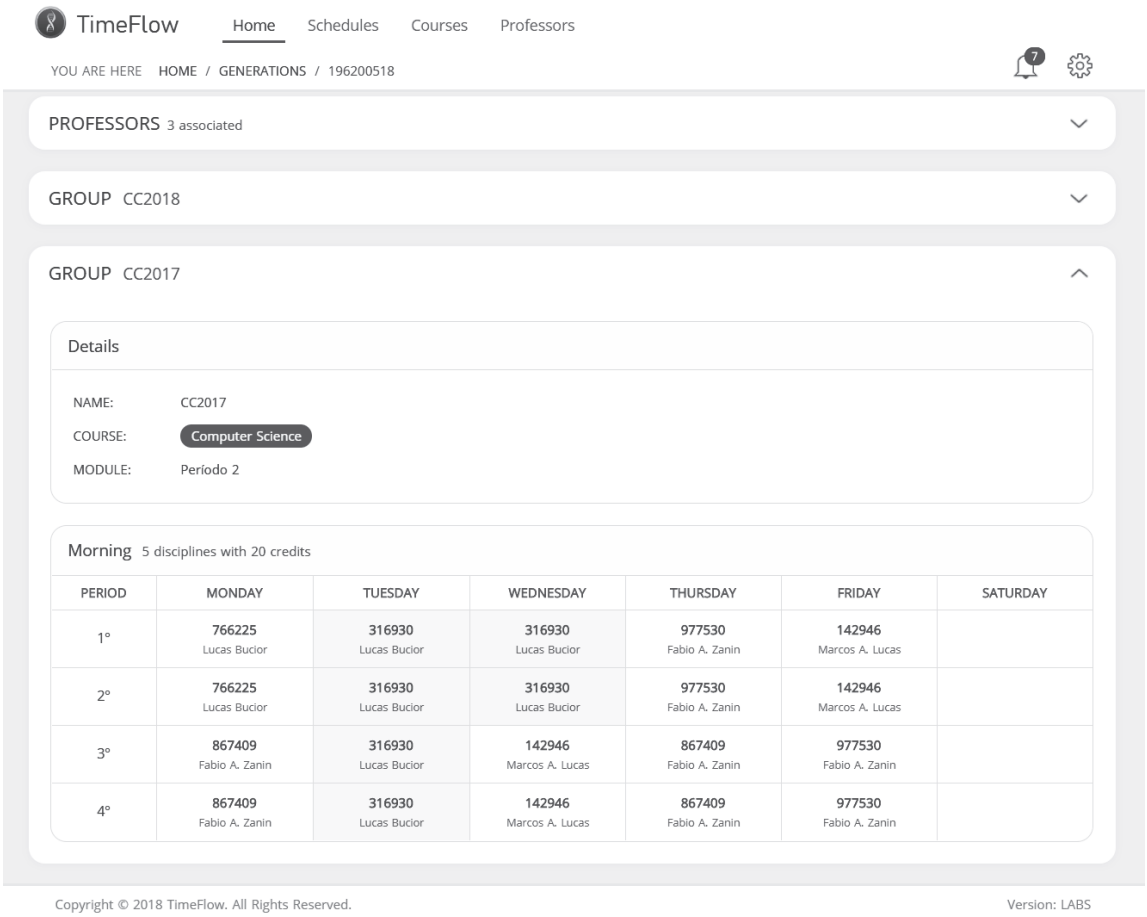

Figura 11. Interface para visualizar a grade horária.

Alguns professores possuem horários de disponibilidade, um critério que precisa ser avaliado na construção da grade horária. De acordo com a Figura 4, existe a possibilidade de ajustar para qualquer professor os horários que ele não pode, para cada turno. A taxa de preferência é organizada de forma que todos professores sejam atendidos dentro do limite, e são aplicadas em cada geração (iteração do algoritmo), para garantir que em novas gerações, todas sejam atendidas. Neste exemplo Figura 4, o professor não pode dar aula em vários períodos da semana para o turno da manhã.

É importante ressaltar que é possível organizar a grade horária de maneiras diferentes para atender às especificações. Neste exemplo de geração Figura 3, são duas turmas com vinte créditos cada, quatro períodos em cinco dias da semana.

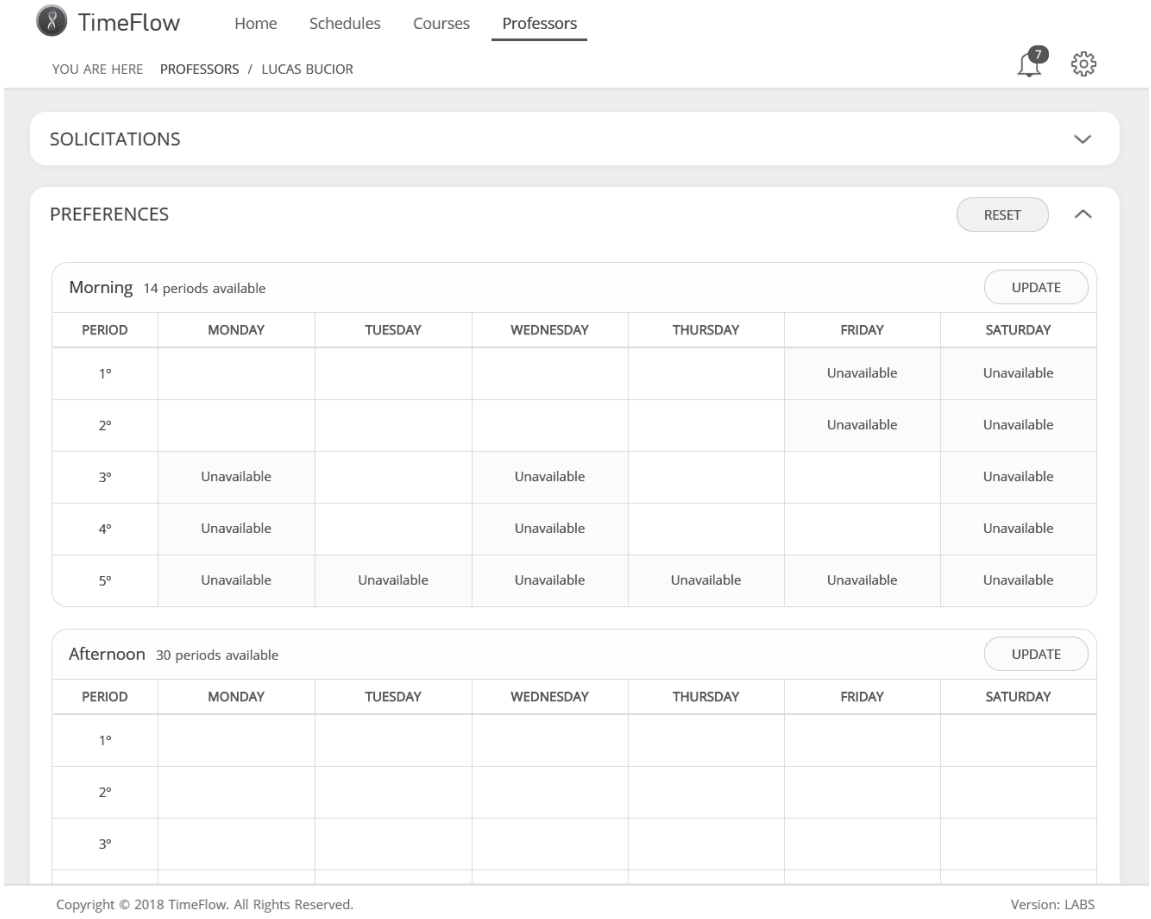

Figura 12. Interface para ajustar preferências do professor.

O foco da plataforma é facilitar a participação dos usuários, às características e funcionalidades estão adaptadas para gerar grades horárias em instituições de ensino.

A Computação Evolutiva descreve métodos e técnicas estudadas por vários pesquisadores. Com essa abordagem temos recursos reconhecidos como ferramentas poderosas para a resolução de vários tipos de problemas.

5. Considerações Finais

Para atingir uma população ideal ou próxima do resultado esperado, precisou de cem mil gerações (iterações do algoritmo) para todos os casos testados. A resolução do problema proposto utilizando técnicas de Inteligência Artificial, possibilitou o escalonamento correto dos créditos e aproximadamente cinco minutos processando.

A plataforma está na versão beta e alguns testes precisam ser realizados para atender um número maior de instituições. A eficiência do algoritmo propõe a possibilidade de gerar grades horárias para inúmeras turmas, adaptando-se a uma instituição de ensino.

Como sugestão para trabalhos futuros, adicionar solicitações aos professores para preencher os horários de disponibilidade e permitir mais de um professor por disciplina.

Referências

CONCILIO, R. (2000). *Contribuições à Solução de Problemas de Escalonamento pela Aplicação Conjunta de Computação Evolutiva e Otimização de Restrições.* Dissertação apresentada ao Departamento de Engenharia da Computação e Automação. Industrial (DCA) Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas UNICAMP-SP.

GOLDBERG, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

HOLLAND, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press.

NARDIN, L. D. (1999). *Estudo da Aplicação de Algoritmos Genéticos para Resolução do Problema de Geração de Horários*. Trabalho de Graduação em Inteligência Artificial apresentado ao curso de Informática na Universidade Estadual do Oeste do Paraná-UNIOESTE.

ROSS, P., CORNE, D., and FANG, H.-L. (1999). *Genetic Algorithms for Timetabling and scheduling*. Departamento de IA Universidade de Edinburgh, 80 South Brige, Edinburgh EH 11HN.

ROSS, P., CORNE, D., and FANG, H.-L. (2000). *Fast practical evolutionary timetabling.* Departamento de IA Universidade de Edinburgh, 80 South Brige, Edinburgh EH 11HN.