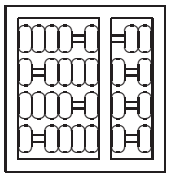

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Aplicação de Heurísticas de Busca Local ao Problema de Agendamento Escolar

Rafael Lamare Silveira *Arnaldo Vieira Moura*

Technical Report - IC-03-028 - Relatório Técnico

December - 2003 - Dezembro

Aplicação de Heurísticas de Busca Local ao Problema de Agendamento Escolar

Rafael Lamare Silveira* Arnaldo Vieira Moura†

Outubro de 2003

Resumo

Este trabalho lida com um problema de otimização muito conhecido no ramo da Pesquisa Operacional, o agendamento escolar (*timetabling*). A instância tratada deste problema reflete uma situação baseada na realidade brasileira. Para a resolução deste problema propomos o emprego de heurísticas de busca local, tais como: *hill climbing*, *simulated annealing* e, em especial, a Busca Tabu. Tais heurísticas vêm demonstrando grande potencial na solução de problemas NP-Difíceis. Paralelamente à construção do resolvedor heurístico, foi construída uma interface gráfica amigável que permite o uso do sistema por usuários leigos.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Pesquisa desenvolvida com suporte financeiro da FAPESP, processo 02/09208-9

†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

Atualmente, existe uma gama de problemas de otimização com alto grau de praticidade que vêm se tornando indispensáveis no processamento eficiente de informações e na tomada de decisões de toda espécie. Podemos citar como exemplos o escalonamento (*scheduling*), o roteamento de veículos e de dados, além dos problemas de agendamento (*timetabling*). A quase totalidade destes problemas, em suas versões mais gerais, não possui algoritmos eficientes para buscar uma solução ótima, tendo sido demonstrado que estão incluídos na classe de problemas NP-Difíceis [MS98].

Para problemas deste tipo, onde não existem algoritmos determinísticos eficientes para obter uma solução ótima, uma alternativa seria o uso de heurísticas, tais como algoritmos evolutivos, programação por restrições, *simulated annealing* e Busca Tabu.

Neste trabalho são propostas algumas técnicas de resolução desse problema utilizando-se heurísticas de busca local. Faremos uma análise geral do comportamento destas heurísticas e compararemos os resultados obtidos pelas mesmas, avaliando a qualidade das soluções geradas e o tempo de processamento necessário para encontrar tais soluções. Neste trabalho será dado um enfoque mais amplo à Busca Tabu já que esta heurística foi a que obteve os melhores resultados nesta abordagem [SIL03].

Este relatório está organizado da seguinte maneira: na próxima seção iremos apresentar uma visão detalhada do problema de agendamento tratado, baseando-se nas especificações do Colégio Stella Maris (de Santos-SP). Na seção 3 apresentamos a modelagem do problema de agendamento, exibindo algumas estruturas de dados úteis e a implementação da função objetivo. A seção 4 faz referência às heurísticas de busca local cobertas por este trabalho, indo desde as mais simples (*hill climbing*, *multi-start* e *simulated annealing* [NSA01]) até uma implementação mais avançada empregando a Busca Tabu. Na seção seguinte listamos os resultados das várias séries de sucessivos testes aplicados ao núcleo resolvedor do agendamento, que nos permitirão avaliar e comparar a eficácia das implementações adotadas. Por fim, a seção 6 mostra a interface gráfica construída para o sistema, tornando-o utilizável por usuários leigos em escolas de ensino fundamental e médio.

2 O Problema de Agendamento Escolar

Para a elaboração de um sistema que resolva o problema de Agendamento Acadêmico, deve-se seguir uma ordem de prioridade nas restrições a serem tratadas [MS98]. À medida que boas soluções forem sendo obtidas para determinado subconjunto do problema, novos parâmetros podem ser acrescentados ao programa até que as respostas fornecidas pelo mesmo satisfazem todas as necessidades do colégio tomado como base (Colégio Stella Maris).

O problema de Agendamento Acadêmico tratado no projeto possui, resumidamente, o conjunto de restrições a seguir, já na ordem de prioridade em que estas foram tratadas:

Restrições Fortes

São aquelas que não podem ser violadas na construção de uma agenda usual. Veja

abaixo essas restrições:

- **Prioridade 0**: Preservar carga horária das classes.
- **Prioridade 0**: Preservar carga horária dos professores.
- **Prioridade 0**: Preservar os períodos de início e término das aulas e intervalos.
- **Prioridade 0**: Uma classe só pode assistir a uma aula e um professor só pode ministrar uma aula por intervalo de tempo.
- **Prioridade 0**: Não deixar janelas entre classes.
- **Prioridade 1**: Preservar os horários dos professores nos quais os mesmos indicam estar incondicionalmente indisponíveis.
- **Prioridade 2**: São proibidas aulas n-uplas (n aulas no mesmo dia, da mesma matéria, seguidas ou não) para $n > 2$.
- **Prioridade 3**: Aulas duplas ($n = 2$) só são permitidas para matérias com *mais* de 2 horas-aula por semana.
- **Prioridade 4**: Não deixar janelas no horário dos professores.
- **Prioridade 5**: Manter aulas coordenadas num mesmo horário.

Observações:

1. 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.
2. O tratamento dado para as janelas de horários para os professores é uma peculiaridade de cada escola. No caso do colégio Stella Maris, esta restrição é tratada como forte, isto é, não pode ser violada. Mas é importante observar que este não é caso de outras escolas brasileiras de ensino fundamental e médio, onde as janelas de professores são tidas como restrições fracas de alta prioridade.
3. A forma como são tratadas as aulas n-uplas e as aulas coordenadas também é bastante variável, dependendo da escola tomada como problema.
4. Exemplos de aulas coordenadas são as aulas 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.
5. As restrições referentes às cargas horárias de professores e turmas, à obediência aos horários de início e fim das aulas e o fato de que em cada sala só pode ser ministrada uma aula em um dado período, são evitadas *a priori* pela própria modelagem adotada na programação do sistema.

O algoritmo desenvolvido age no sentido de minimizar o valor da função objetivo, que constitui-se da soma de todas as penalidades recebidas pela solução corrente. Dessa forma, o algoritmo prioriza a correção das restrições fortes, preocupando-se de forma secundária com as restrições fracas. Porém, isso é feito de forma balanceada sem dar prioridade excessiva às restrições fortes.

Restrições Fracas

São aquelas cuja violação não compromete totalmente o resultado final, sendo o objetivo do algoritmo minimizar a ocorrência das mesmas. Veja abaixo essas restrições, em ordem de prioridade:

- **Prioridade 1**: Satisfazer os horários à preferência dos professores.
- **Prioridade 2**: Distribuir aulas de uma mesma matéria eqüitativamente através da semana.
- **Prioridade 3**: Agrupar as aulas duplas existentes de modo que seus horários fiquem seguidos.
- **Prioridade 4**: Distribuir matérias de baixo esforço mental[1] eqüitativamente através da semana.
- **Prioridade 5**: Satisfazer as preferências abertas de professores como, por exemplo, ter um dia - qualquer dia - livre na semana.

Observações:

1. A modelagem que foi empregada permite não só a distribuição de aulas de "baixo esforço mental", mas também uma distribuição por vários níveis de dificuldade das matérias. Estes níveis são definidos de forma subjetiva e devem ser dados como parâmetros de entrada para o sistema.
2. O principal caso de preferências abertas dos professores é o *dia livre*, onde o professor pode solicitar ao sistema os n dias nos quais esteja livre dos encargos no colégio.

2.1 Modelagem

Agora iremos apresentar a modelagem do problema de agendamento empregada no núcleo resolvedor do sistema. Explicitaremos a evolução da modelagem durante o projeto, passando de um modelo simplificado até um modelo final que engloba todas as restrições apresentadas acima.

[1] As matérias enquadradas neste nível, de acordo com o Colégio Stella Maris, são: Educação Artística, Computação, Educação Musical, Educação Física e Orientação Religiosa.

2.1.1 Simplificações Iniciais

Da mesma forma que as restrições foram tratadas em uma determinada ordem, a forma com que o problema é modelado também foi simplificada, à princípio. À medida que boas soluções surgiram para os modelos iniciais outras restrições foram sendo embutidas no sistema.

Veja abaixo as simplificações no modelo que foram consideradas na primeira etapa do projeto:

Horários de aulas: Os horários de aulas (doravante chamados de slots) correspondem a um período de tempo em que uma turma assiste a uma mesma aula. Por exemplo, o primeiro slot do dia para as sextas séries começa às 7:30 e termina às 8:20.

- No problema real, os horários de início e fim dos slots não são necessariamente os mesmos para todas as séries. Deve-se evitar que um mesmo professor lecione em um slot para uma classe e no slot seguinte para outra, se esses slots possuem um intervalo de sobreposição.
- À principio, o problema foi tratado como se todos os slots de aula ocorressem nos mesmos períodos de tempo para todas as turmas. Ou seja, como se todas as primeiras aulas do dia começassem às 7:30, as segundas às 8:20 e assim por diante.
- No problema real, as séries do ensino fundamental possuem 25 aulas semanais e as do ensino médio possuem 30 aulas. Além disso, os horários de início e fim de cada aula são diferentes, assim como os intervalos. Apesar de raros, existem professores que lecionam para séries de ambos os grupos.
- Além da diferença no número de aulas semanais, existem ainda distinções entre os horários de inícios e fins das aulas. Por exemplo, o primeiro slot da grade do ensino médio inicia 20 minutos antes do primeiro da grade do ensino fundamental. Assim, a primeira aula das turmas de ensino fundamental sobrepõe-se à primeira e à segunda aula do ensino médio.

Na modelagem inicial, o problema foi tratado apenas para as turmas do ensino fundamental, evitando sobreposições de horários.

2.1.2 Modelo Final

As simplificações na modelagem permitiram uma visão inicial do problema, para o qual foi desenvolvido um resolvedor simplificado. Desta forma, foi possível conhecer também o funcionamento das heurísticas diante do problema.

Depois de ser implementado um algoritmo de busca local eficiente para resolver a modelagem inicial, ocorreu a migração deste modelo simplificado para o modelo final, batizado de *modelo multigrades*. Este modelo é bem mais flexível que o anterior já que permite que o agendamento seja tratado para turmas de alunos cujos horários das aulas não são os mesmos, podendo haver múltiplas grades de horário no colégio. O colégio Stella Maris

demonstra um caso claro de onde esta forma de agendamento deve ser adotado. Veja alguns exemplos disso:

- Levando em consideração o modelo de multigrades, identificamos no colégio Stella Maris 4 grades distintas: uma para as 3 quintas séries, outra para as 3 sextas séries, uma terceira para as sétimas e oitavas séries e, por último, uma grade para englobar as três turmas do ensino médio.

- Devemos ressaltar, também, outras vantagens que podem ser obtidas usando-se o modelo de multigrades, tais como agendamento de turmas em dias cujo número de aulas é diferente, tratamento de intervalos entre as aulas, aulas com duração variável, entre outras.

- Um outro importante acréscimo foi a inclusão do tratamento das aulas coordenadas no resolvedor, que era a última restrição faltante para completar o conjunto de restrições fortes do problema.

2.2 Tarefas Posteriores

Outras tarefas podem ser incorporadas dando continuidade ao projeto. O objetivo seria generalizar as restrições do problema para que o sistema seja válido para outros colégios. Basicamente, as restrições são as mesmas, mas existem algumas pequenas variações. Como por exemplo:

- O programa passaria a fazer também as alocações de horários dos professores para as turmas, que a priori assume-se que vêm prontas. Dessa forma, seriam informadas apenas as competências de cada professor e o programa checaria 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.

- Janelas para professores podem ser permitidas em outras escolas, deixando então essa restrição de ser forte para se tornar fraca.

- O tratamento dado às aulas n-uplas pode variar. Algumas escolas permitem, em última instância, até mesmo aulas triplas.

Outra vertente do problema de agendamento que pode ser tratada é o *agendamento interativo* [GOL01], onde o usuário pode participar ativamente do processo de geração das agendas para o colégio. Este método permite que o usuário pare o resolvedor em um dado ponto, avalie a melhor solução encontrada até o momento e faça as alterações que julgar necessárias. Assim, o usuário pode fixar certos professores em horário determinados, proibindo o resolvedor de mudá-los; pode também fazer movimentos de troca de horários dos professores e permitir que o algoritmo continue a partir daquele ponto para verificar qual será o comportamento das novas soluções geradas, entre muitas outras possibilidades. É

importante ressaltar que os algoritmos de busca local, tais como a busca tabu, são extremamente propícios para lidar com este tipo de extensão do problema de agendamento, dada a sua característica iterativa.

Problema de Agendamento para Universidades

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

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

3 Modelagem

Esta seção, ao contrário da seção 2, irá tratar da modelagem do sistema à nível de implementação. Assim, iremos definir as estruturas de dados básicas e mostrar como foi implementada a função objetivo para o problema.

3.1 Estruturas de Dados Básicas

Nesta seção são discutidos maiores detalhes das estruturas de dados usadas na construção do resolvedor. Basicamente, iremos analisar as duas principais estruturas da modelagem do problema: a estrutura que codifica uma solução para o problema, a qual chamaremos *agenda*; e a estrutura que armazena o movimento que é feito sobre a solução, o qual define a vizinhança do algoritmo de busca local empregado.

Vetores e Tabelas

Vetores e tabelas são duas estruturas de dados primárias, ambas utilizadas na implementação da estrutura da solução do problema. No sistema desenvolvido, estas duas estruturas são alocadas dinamicamente de acordo com as necessidades da instância do problema.

A estrutura `Tabela` está assim codificada em linguagem C Padrão:

```
typedef struct tabela {
  int **info;
  int nl, nc;
} Tabela;
```

Onde `info` é uma matriz de inteiros e, `nl` e `nc` informam, respectivamente, o número de linhas e colunas que a matriz possui.

O uso de tabelas isoladamente não é suficiente para a codificação de uma agenda para o problema. Por isso foi definida uma estrutura denominada `TabVet` que associa uma tabela a

um vetor cujo comprimento é igual ao número de linhas da tabela. Cada campo deste vetor servirá para guardar o total de penalidades encontrada na linha ao qual ele corresponde na matriz. Veja a codificação desta estrutura em linguagem C:

```
typedef struct tv {
  Tabela *t;
  int *v;
} TabVet;
```

Solução

A estrutura escolhida para modelar uma agenda do problema é constituída por três vetores tipo `TabVet`, um vetor de inteiros e um inteiro. Sua implementação no código do programa é mostrada abaixo:

```
typedef struct sol {
  TabVet **turmas;
  TabVet **professores;
  TabVet **materias;
  int *vetprof;
  int penalidades;
} Solucao;
```

Cada vetor de tabelas-vetor (estrutura `TabVet`) é alocado de acordo com o número de grades, isto é, o número de `TabVet`s de cada vetor é igual ao número de grades da instância do problema que está sendo resolvida. A existência de três vetores de `TabVet`s denominados turmas, professores e matérias é necessária para ter-se três diferentes visões da mesma solução pois, como veremos mais adiante, isso otimiza o cálculo da função objetivo.

Cada tabela-vetor do campo `turmas` indica qual professor está ministrando aula em uma dada turma num dado período. Assim, nas linhas da tabela temos as turmas de alunos e nas colunas os slots de aula. As células desta matriz são preenchidas com o código do professor que está ministrando a aula. O vetor associado a cada tabela indica o total de penalidades associadas à turma correspondente.

As tabelas-vetor do atributo `materias` indicam em suas tabelas uma informação similar a que é guardada em `turmas`, diferindo desta apenas pelo fato de que nas células são armazenados os valores que indicam a matéria que está sendo ministrada e não o professor que ministra a aula. Os vetores associados às tabelas guardam as penalidades obtidas em cada linha da matriz, estas penalidades referem-se principalmente às violações das distribuições das aulas das matérias nas turmas, tais como aulas n-uplas e distribuição das matérias por nível de dificuldade.

O atributo `professores` guarda nas linhas de suas matrizes os professores e nas colunas os slots de aula da semana. As células indicam o número de aulas que o professor está ministrando em um dado slot. Obviamente, estas células só poderiam receber os valores 0, quando o professor não estiver ministrando aula naquele período, ou 1, quando ele estiver

alocado para ministrar alguma aula. Porém a modelagem utilizada permite que um mesmo professor esteja ministrando aulas simultâneas em turmas diferentes, sendo tal solução fortemente penalizada de modo que, durante as iterações da busca tabu, soluções que violarem esta restrição serão consideradas muito desvantajosas pelo algoritmo e serão progressivamente eliminadas à medida que o resolvedor for se aproximando dos mínimos locais. Os vetores associados às matrizes de professores guardam as penalizações acumuladas principalmente pelas violações das restrições referentes aos horários dos professores, tais como indisponibilidades, preferências e aulas simultâneas.

O cálculo de violações de janelas e sobreposições nos horários dos professores envolvem interações entre grades distintas, pois as violações ocorrem simultaneamente em mais de uma grade. Deste fato surge uma dificuldade: se uma dada violação ocorre em mais de uma grade torna-se difícil saber a qual vetor de qual grade iremos atribuir a penalidade. Para resolver este conflito foi inserido na estrutura da solução um vetor denominado `vetprof` cujo comprimento é igual ao número de professores do colégio. Em cada célula desse vetor ficam acumuladas as penalidades recebidas pelo professor para restrições que envolvem interações entre grades distintas: janelas, sobreposições de horários e dias-livres.

Por fim, a estrutura `solução` contém também um inteiro que acumula o somatório de todas as penalidades da agenda correspondente.

Movimento Simples

Uma das características fundamentais dos algoritmos de busca local é a definição da vizinhança de uma solução. Entende-se por vizinhança de uma dada solução S, o conjunto de soluções S_0 que podem ser obtidas a partir da solução S realizando-se um movimento sobre esta. Neste caso, movimento é qualquer modificação estrutural sobre uma solução S que a transforma em um vizinho S_0.

No caso do problema de agendamento tratado na modelagem inicial, o movimento era reprensentado pela tripla <*turma, slot1, slot2*>. O modelo multigrades instituiu uma nova estrutura para o movimento, inserindo uma nova dimensão no movimento: a grade. Na nova modelagem, o movimento é descrito por uma quádrupla <*grade, turma_da_grade, slot1, slot2*>. O significado desta modelagem é o seguinte: quando um movimento representado pela quádrupla <*g, t, p1, p2*> é executado sobre uma dada solução, a matéria alocada no slot $p1$ da turma t na grade g é trocada pela matéria alocada no slot $p2$ desta mesma turma. Este tipo de movimento é denominado *swap*. Veja a implementação desta estrutura abaixo:

```
typedef struct ms {
  int grade, linha;
  int p1, p2;
} movimentoSimplesMG;
```

No código acima temos quatro inteiros: `grade` identifica em qual grade será executado o movimento, `linha`, representa a turma nesta grade, `p1` o slot 1 e `p2` o slot 2. Pode-se observar que esta estrutura não permite que um movimento envolva slots de duas turmas diferentes, evitando que as cargas horárias das turmas sejam violadas.

Uma observação importante é que a tripla que representa o movimento na modelagem antiga pode ser mapeada na quádrupla da nova representação, já que a dupla <*grade,turma_da_grade*> pode ser mapeada univocamente com o campo *turma* da tripla. Assim não seria necessária a substituição da antiga estrutura de movimento por esta nova implementação. Porém isto foi feito por motivos de eficiência, já que assim evita-se o *overhead* de fazer o mapeamento toda vez que um movimento é executado.

3.2 Função Objetivo e Penalidades

A função objetivo, na implementação usada, desempenha o papel de qualificador das soluções e dos vizinhos gerados pela busca tabu. O valor calculado pela função objetivo representa o somatório de todas as penalidades fracas e fortes recebidas pela solução, contando todas as violações de restrições ali verificadas. Este valor serve de norte para a exploração das vizinhanças analisadas pela busca local, guiando-a às regiões promissoras do espaço de busca.

Algumas características são importantes para uma função objetivo eficiente, tais como:

- **Restrições do problema bem definidas:** é importante que todo o conjunto de restrições esteja incorporado no código da função objetivo, de modo que o resolvedor possa lidar com todas as restrições de forma adequada.
- **Atribuir valores coerentes às penalidades:** a regra básica atribui um valor de penalidade proporcional à importância da restrição associada a esta penalidade. Assim, restrições fortes recebem penalidades mais rígidas que as fracas. Porém, deve existir também um balanceamento adequado que diferencie quantitativamente uma restrição fraca de uma forte, não permitindo que se favoreça demais estas últimas.
- **Algoritmo eficiente:** a eficiência do algoritmo implementado para a função objetivo é um dos pontos críticos para a convergência, em tempo aceitável, do resolvedor, já que o sistema passa cerca de 80% do seu tempo de processamento calculando o valor da função objetivo. Este valor é calculado algumas dezenas de vezes a cada iteração. Qualquer falha de projeto neste ponto comprometerá todo o desempenho do sistema.

A classe de algoritmos de busca local permite, por sua própria natureza, que se implemente uma função objetivo mais eficiente. Esta eficiência vem do fato de que não é necessário, para o cálculo da função objetivo, percorrer completamente todas as matrizes da estrutura `Solucao` a cada vez que é realizado um movimento sobre esta. Isto ocorre porque o movimento de troca que define a vizinhança da busca local produz apenas uma pequena modificação na estrutura da solução, ou seja, o vizinho de uma solução é estruturalmente similar a ela e, portanto, os valores de suas penalidades também o são.

Deste modo, tendo-se a soma x das penalidades (e os valores de penalidades acumulados nos vetores associados às matrizes da solução) de uma dada solução S é muito simples e rápido encontrar a soma de um vizinho S_0 que foi gerado realizando-se sobre S uma troca representada pela quádrupla <*g, t, p1, p2*>. Para se fazer isto, basta calcular a diferença d do somatório de todas as penalidades na solução S_0 em relação à S. E esta diferença

pode ser calculada por um algoritmo que considera apenas a linha t da grade g da tabela `materias` e as linhas $p1$ e $p2$ da tabela `professores`, já que são apenas estas linhas da estrutura da solução que são afetadas pelo movimento descrito. De modo que o valor da função objetivo calculada para a solução S_0 é $x + d$, onde d pode ser maior, menor ou igual a 0.

O uso desta particularidade dos algoritmos de busca local permitiu que se implementasse um resolvedor que usa um tempo de processamento 70% menor que o algoritmo que sempre calculava a função objetivo completa, ou seja, uma função que varria toda a estrutura da solução. Agora, a função objetivo completa só é calcula uma única vez: quando a solução inicial é instanciada.

Este detalhe da construção do resolvedor demonstra a importância do campo inteiro `penalidades` e dos vetores que acumulam as penalidades de cada linha das matrizes da estrutura `Solucao`, já que eles são responsáveis por armazenar os valores atuais das penalidades atribuidas à solução. Assim, para se encontrar o novo valor da função objetivo depois de realizado um dado movimento basta apenas calcular o valor da diferença entre as penalidades das duas soluções e atualizar estes campos da estrutura.

Restrições Fortes

Como já foi dito anteriormente, a própria modelagem adotada impede que algumas restrições fortes sejam violadas, são elas: cargas horárias de turmas e professores, horário de início e fim das aulas e aulas simultâneas numa mesma turma de alunos. Sendo assim, restam apenas 6 restrições fortes para serem tratadas pelo sistema: (1) aulas simultâneas dos professores (`AULAS_SIMULTANEAS`); (2) janelas no horário dos professores (`JANELAS_DE_PROFESSOR`); (3) indisponibilidades dos professores (`INDISPONIBILIDADES`); (4) tratar adequadamente as aulas n-uplas de uma mesma matéria (`AULAS_N_UPLAS`); e (5) dar um tratamento especial às aulas coordenadas `AULAS_COORDENADAS`. A sexta restrição forte surgiu devido à implementação do tratamento de multigrades: é a restrição de `SOBREPOSICAO` de aulas nos horários dos professores em diferentes grades. Ela é equivalente a uma aula simultânea, porém a simultaneidade ocorre em slots de grades distintas.

Além da inclusão da restrição de sobreposição de slots em grades diferentes, o modelo multigrades modificou o tratamento da restrição de janelas nos horários dos professores e das preferências por dias livres, uma vez que estas são as restrições que envolvem interações entre as grades.

O valor das penalidades atribuídas às restrições fortes está na ordem de algumas centenas para cada restrição violada, enquanto que para as restrições fracas os valores são da ordem de algumas unidades, no máximo, umas poucas dezenas. Estes valores mostraram-se adequados nos testes realizados com o resolvedor. Maiores detalhes sobre os valores das penalidades e sobre os testes são mostrados na seção 5.

Restrições Fracas

Os valores penalizadores atribuídos às restrições fracas são os mais críticos para definir a qualidade das soluções encontradas pelo resolvedor, pois nas soluções finais todas as restrições fortes devem ser obedecidas e o valor atribuído pela função objetivo será um

somatório resultante apenas das violações de restrições fracas.

No caso do agendamento proposto neste projeto o conjunto de restrições fracas está assim definido: (1) atender as preferências de horários dos professores (`PREFERENCIAS` e `DIA_LIVRE`); (2) procurar distribuir de maneira equilibrada as matérias durante a semana.

4 Heurísticas de Busca Local

Neste ponto, iremos apresentar uma visão geral das heurísticas de busca local testadas na implementação do núcleo resolvedor do sistema de agendamento. Começamos mostrando as heurísticas mais simples, tal como a busca local descendente. A seguir, refinamos esta heurística implementando um mecanismo de inicialização em múltiplos pontos do espaço de soluções (*multi-start*) e apresentando um algoritmo de recozimento simulado para o agendamento. Por fim, descreveremos de maneira mais detalhada a heurística de Busca Tabu, a qual foi o foco principal de desenvolvimento neste projeto.

4.1 Busca Local Descendente

Esta heurística, também conhecida como *hill climbing* aplica uma metodologia de busca local bastante simples, que tenta, a cada iteração, encontrar um vizinho que seja melhor ou de qualidade igual à solução atual. O algoritmo pára depois de um número fixo de iterações sem melhorar a solução atual.

Veja o pseudocódigo do algoritmo resolvedor do timetabling aplicando esta heurística:

```
/* BUSCA LOCAL DESCENDENTE */

INICIALIZAÇÃO:

   ATUAL = geraSolucaoInicial();          //solução inicial aleatoriamente gerada

BUSCA LOCAL:

   FAÇA ENQUANTO ISM < MAX_ISM
   {
      MOVIMENTO = movimentoAleatório();

      SE MOVIMENTO reduz o valor da funcaoObjetivo ENTÃO
        ATUAL = executaMovimento(MOVIMENTO);
      SENÃO
        ISM++;
   }

   RETORNA ATUAL;
```

4.2 Multi-Start

Consiste de uma heurística similar ao *hill climbing*, porém desenvolve um mecanismo de diversificação bastante simples: depois que a busca local estacionar em um dado mínimo local durante um dado número fixo de iterações, o algoritmo é reinicializado com uma nova solução gerada aleatoriamente (em outra vizinhança do espaço de busca). O efeito deste algoritmo é igual a uma busca local descendente executado várias vezes. Na prática, ele obteve soluções de melhor qualidade que a busca local descendente, dado um tempo pré-fixado para o algoritmo executar.

4.3 *Simulated Annealing*

É uma heurística baseada em uma analogia com processos estocásticos e cujo parâmetro principal é denominado *temperatura*. A temperatura é um parâmetro que mede o grau de diversificação que o algoritmo pode ter. É ela quem indica a probabilidade de que uma solução de pior qualidade seja escolhida para substituir a atual. A temperatura geralmente cai com o transcorrer do tempo de busca. Existem várias formas de controlar esta queda de temperatura: queda linear, exponencial, etc. No algoritmo que foi implementado foram feitas algumas modificações estruturais em relação à implementação clássica, porém mantêm-se a mesma idéia de escolha probabilística dos movimentos. Um algoritmo genérico para o *simulated annealing* para a resolução do timetabling é dado abaixo:

```
/* BUSCA LOCAL DESCENDENTE */

INICIALIZAÇÃO:

   ATUAL = geraSolucaoInicial();          //solução inicial gerada por 10.000
                                          //iterações de busca local descendente

RECOZIMENTO SIMULADO:

   FAÇA ENQUANTO ISM < MAX_ISM
   {
      MOVIMENTO = movimentoAleatório();

      SE ( FATOR_PROBABILÍSTICO <= numeroAleatorio(0, VALOR_LIMITE) ) ENTÃO
        ATUAL = executaMovimento(MOVIMENTO);
      SENÃO
        ISM++;
   }

   RETORNA ATUAL;
```

Veja que o algoritmo mostrado é bastante similar ao já referido pelo *hill climbing*. A única alteração ocorre na linha que apresenta a seguinte comparação:

```
SE ( FATOR_PROBABILÍSTICO <= numeroAleatorio(0,VALOR_LIMITE) ) ENTÃO
```

Esta linha diz que: se o valor atual do FATOR PROBABILÍSTICO (que é dependente da temperatura, como veremos) for menor ou igual a um dado número aleatório escolhido entre 0 e um VALOR LIMITE então o movimento referido poderá ser executado pela solução, embora possa piorar o valor atual da função objetivo. Tal modificação possui um grande potencial para diversificar a busca, tornando esta heurística bem mais efetiva na resolução de problemas combinatórios que uma simples busca local exaustiva.

No caso da nossa implementação deste método de busca local estocástico, o fator probabilístico é calculado de acordo com a relação dada a seguir:

$$\texttt{FATOR\_PROBABILÍSTICO} = \frac{-\Delta}{T}$$

onde T é a TEMPERATURA e Δ é o ganho obtido pela solução ao ser executado o movimento avaliado, isto é, o valor da função objetivo depois de o movimento ser realizado subtraído do seu valor atual.

Como vimos, a implementação do *simulated annealing* possui a vantagem de ser tão fácil de implementar quanto o *hill climbing*. O que torna esta heurística potencialmente superior é uma boa escolha para os dois principais parâmetros especificados: FATOR PROBABILÍSTICO e VALOR LIMITE. É importante lembrar que o VALOR LIMITE está no intervalo de valores reais de 0 a 1. No caso em que FATOR PROBABILÍSTICO < 0 temos que o movimento implica em uma solução melhor que a atual (já que $\Delta > 0$), deste modo este movimento é automaticamente executado.

Os dois parâmetros referidos podem ser escolhidos empiricamente, baseando-se em exaustivos testes sobre diversos espaços de busca. Em geral, o valor da TEMPERATURA é atrelado ao valor do ganho proporcionado pelo movimento e o VALOR LIMITE deve ser passado como parâmetro para o algoritmo a fim de serem testados diversos valores para encontrar um que proporcione uma convergência equilibrada, já que se VALOR LIMITE for muito pequeno (próximo de zero) iremos restringir demais a busca pois dificilmente permitiremos que algum movimento piore a solução atual (veja que se VALOR LIMITE = 0 temos o mesmo efeito de um hill climbing), enquanto que se VALOR LIMITE for muito grande (próximo de um) então a busca ficará muito dispersa, dificilmente convergindo para ótimo locais (com VALOR LIMITE = 1 temos um busca exaustiva aleatória).

No nosso caso foram testadas três diferentes implementações modificando-se apenas a forma como é calculado o FATOR PROBABILÍSTICO. Estas implementações são as seguintes:

- **Temperatura fixa**: O valor do parâmetro FATOR PROBALÍSTICO é fixado e passado como parâmetro para o algoritmo. Neste caso, a probabilidade de uma solução pior tornar-se a atual é igual ao próprio valor passado como parâmetro e indenpende da temperatura e do Δ.

- **Temperatura Variável 1**: a temperatura é igual ao valor da função objetivo da solução inicialmente gerada, assim temos um valor constante para T. Assim, FATOR PROBABILÍSTICO $= \frac{-\Delta}{ObjIni}$.

- **Temperatura Variável 2**: a temperatura é igual ao valor da função objetivo da solução atual, deste modo temos um valor dinâmico de T. Agora: `FATOR PROBABILÍSTICO` $= \frac{-\Delta}{ObjAtual}$.

Veja que em todas as implementações acima o valor do fator probabilístico é um função linear (um quociente ou uma constante). Não foi implementada uma heurística cuja probabilidade de escolha da nova solução fosse uma função exponencial entre a temperatura e o ganho, já que isto foge do escopo deste trabalho, pois a abordagem mais explorada foi a Busca Tabu.

4.4 Busca Tabu

Nesta seção iremos focar nossa atenção especificamente na heurística de Busca Tabu, já que a mesma foi a que obteve melhores resultados na implementação do resolvedor do problema de agendamento. Veremos na seção 5 os testes comparatórios entre as diversas metodologias de busca local utilizadas, confirmando a superioridade da implementação da busca tabu que será descrita em detalhes a seguir.

4.4.1 Uma Visão Geral da Busca Tabu

A busca tabu é uma heurística, ou melhor, uma *metaheurística*, proposta por Fred Glover para resolver problemas de otimização combinatória [GL97, GLO90, HTW95]. Trata-se de um método de busca inteligente que procura evitar que a busca estacione em um ótimo local. Para conseguir esse efeito, a heurística faz uso de estruturas de memória bastante flexíveis, que são estrategicamente usadas para guiar um agressivo processo de busca em todo o espaço de soluções.

O algoritmo realiza buscas no espaço das soluções com o intuito de, a cada iteração, encontrar o melhor vizinho da solução corrente e que não esteja na *lista tabu*. Dada a flexibilidade desta metaheurística, pode-se relaxar esta condição de *tabu-ativo* usando-se um *critério de aspiração*, o qual age no sentido de ignorar a restrição tabu de um dado vizinho. Um fato significativo da metodologia da busca tabu é que a mesma não avalia a qualidade das soluções apenas pelo valor (quantitativo ou qualitativo) da sua função objetivo. Além disso, leva ainda em consideração quatro fatores que medem a atratividade de uma dada solução: qualidade, influência, frequência e"recência" (do inglês, *recency*). E, ainda, o processo de busca não se dá de forma estática pois ocorrem, durante o processo de busca, alternâncias entre estratégias de *diversificação* e *intensificação*. A primeira tenta dispersar a busca por todo o espaço de soluções e a segunda tenta focar uma vizinhança ou, mais ainda, uma região promissora de uma dada vizinhança.

Em concordância com o que é exposto pelo próprio Glover [GL97], esta metaheurística emprega duas estratégias distintas e complementares: *memória de curto prazo* e *memória de longo prazo*. A primeira estratégia é encontrada em praticamente todas as aplicações que usam busca tabu, dado que a mesma é de fácil implementação e obtém ótimos resultados. O algoritmo básico de uma estratégia de memória de curto prazo envolve os aspectos já citados no início do parágrafo anterior: a característica tabu da solução, um ou mais critérios de

aspiração, além de uma (ou mais) estratégia de lista de candidatos. O algoritmo básico pode ser resumido assim:

1. Gere uma solução inicial, x. A melhor solução, x^*, e a solução corrente, x', são inicializadas como x. A lista de movimentos tabu-ativos, T, é inicializada como vazia.
2. Crie, a partir da solução x', uma lista L de movimentos candidatos.
3. Escolha, entre os movimentos de L, algum movimento M onde a solução decorrente, $M(x') = y$, tem um valor baixo para a função objetivo, $F(y)$.
4. Se M não está na lista T de movimentos tabu-ativos, então execute o passo 6, caso contrário execute o passo 5.
5. Se M satisfaz o critério de aspiração, execute o passo 6, caso contrário volte para o passo 3.
6. Tome y como a nova solução corrente x'. Se $F(y) < F(x^*)$ então estabeleça y como a nova melhor solução x^*.
7. Inclua M na lista T de movimentos tabu-ativos e atualize T, removendo os movimentos que deixaram de ser tabu-ativos.
8. Repita a partir do passo 2, até que um critério de parada seja alcançado.
9. A solução encontrada está em x^*.

Os critérios para criar a lista L (passo 2), escolher o movimento M (passo 3), além do critério de aspiração (passo 5) e da estratégia para gerenciar a lista T (passo 7) são parâmetros determinantes para o sucesso da heurística.

O uso de estratégias de memória de longo prazo envolve métodos mais sofisticados para auxiliar os métodos de memória de curto prazo na busca rápida por uma boa solução. Estas estratégias exploram, de forma mais consistente, os conceitos de diversificação e intensificação, tornando o uso da memória ainda mais flexível. Dentre as diversas formas de se usar este tipo de memória podemos destacar a *oscilação estratégica*, que faz com que a busca oscile entre níveis de viabilidade estipulados de acordo com a necessidade de diversificação ou intensificação. Nesse caso a busca tabu permite que se sacrifique a viabilidade das soluções para que se possa encontrar regiões com soluções promissoras. Outra forma de uso desta memória é a estratégia de *ligar caminhos* (*path relinking*), que consiste de escolher algumas soluções críticas, chamadas de soluções de referência, e partir de uma solução de referência inicial para alcançar uma outra. O que se espera com a aplicação deste último método é que no caminho entre duas soluções de referência (que provavelmente são de alta qualidade) possamos encontrar soluções que também são de alta qualidade, talvez até melhores que as soluções de referência originais. Existem ainda outros mecanismos para explorar a memória de longo prazo [GL97].

4.5 O Algoritmo de Busca Tabu do Resolvedor

Esta seção descreve todo o histórico de evolução dos algoritmos implementados na realização deste projeto, desde uma descrição resumida das versões programadas na fase inicial do projeto até as versões que tratam da modelagem completa. No final da seção será apresentado o algoritmo final para o resolvedor e um pseudocódigo para o mesmo.

4.5.1 As Versões Iniciais do Algoritmo

Durante o desenvolvimento do projeto foram construídos, em sequência, quatro algoritmos de busca local para o resolvedor. Cada algoritmo que era implementado visava suprir as deficiências dos anteriores e corrigir as possíveis falhas dos mesmos. Esta seção fará um resumo destes quatro algoritmos visando compará-los com a versão final empregada pelo sistema.

- **Algoritmo Protótipo**: esta versão do algoritmo serviu apenas para testar as estruturas de dados que estavam sendo implementadas. Ele implementava uma heurística não-ascendente simples (*hill climbing*).
- **Busca Tabu Não-Ascendente**: o algoritmo implementado nesta versão é similar ao anterior porém empregava uma forte política de seleção de movimentos candidatos, além de empregar uma lista tabu aliada a um critério de aspiração.
- **Busca Tabu Alternada**: nesta versão foi empregado um mecanismo de diversificação na heurística de busca local de modo a compensar a forte intensificação que foi usada nas versões anteriores. Este algoritmo operava em ciclos, cada um englobando duas fases: uma de diversificação e outra de intensificação.
- **Busca Local Híbrida**: este algoritmo surgiu da necessidade de uma diversificação mais forte, pois em problemas com espaço de busca muito reduzido (soluções com poucas turmas de alunos) o algoritmo anterior estagnava em ótimo locais. Assim, nesta nova versão o que se fez foi substituir a fase de diversificação da busca tabu alternada por um *simulated annealing* com temperatura constante.

Um fato importante a ser ressaltado é o emprego da lista tabu e do critério de aspiração nas três últimas implementações acima. A lista tabu usada era uma lista de movimentos proibidos de tamanho fixo. O critério de aspiração empregado é o conhecido *improved best* que viola a condição de tabu-ativo de um dado movimento caso ele produza um ganho quantitativo no valor da função objetivo.

4.5.2 O Algoritmo Final

A quarta versão do algoritmo consistiu simplesmente do mesmo algoritmo de Busca Local Híbrida (descrito na seção anterior) porém com as estruturas de dados e a função objetivo adaptadas ao modelo multigrades. Todavia esta heurística não obteve os mesmos resultados que foram obtidos na modelagem anterior. Percebeu-se que a fase de diversificação do

algoritmo usando o *simulated annealing* acabava por dispersar excessivamente a solução dos ótimo locais, tornando a busca muito custosa. Por isso o uso do *simulated annealing* foi descartado na versão final da heurística.

A quinta versão do resolvedor é bastante similar à implementação da Busca Tabu Alternada descrita anteriormente. As principais diferenças encontram-se na codificação das fases de intensificação e diversificação do algoritmo. Estas fases são codificadas da seguinte maneira:

- **DIVERSIFICAÇÃO**: aplica um método de busca tabu simples aliado a uma forte estratégia para a criação de uma lista de movimentos candidatos (descrita no final desta seção). Esta fase opera da seguinte forma: gera-se uma lista de movimentos candidatos e escolhe-se um deles para ser executado através de um método guloso (escolhendo o melhor movimento), verifica-se se o movimento é tabu ou obedece ao critério de aspiração e, caso não possa ser executado, escolhe-se o próximo movimento da lista gerada. Esta fase da busca possui dois parâmetros importantes de diversificação: um especifica o comprimento da lista de movimentos candidatos, o outro indica o número de iterações sem melhora que esta fase irá executar em cada ciclo. Esta fase é considerada de diversificação porque permite que a qualidade da solução, medida pela função objetivo, seja degradada deixando que a busca se desloque para outras regiões do espaço de soluções.

- **INTENSIFICAÇÃO**: aplica um mecanismo não-ascendente aliado aos efeitos da lista de movimentos tabu-ativos, porém sem o emprego do critério de aspiração, uma vez que sua aplicação eliminaria os efeitos da lista tabu já que a própria heurística não-ascendente proibe movimentos que degradem o valor da função objetivo. Poderiam ser empregados outros critérios de aspiração, diferentes do *improved best* (descrito no fim da seção), porém não se julgou necessária esta implementação.

A aplicação da alternância entre estas duas estratégias de busca operando em ciclos obteve ótimos resultados. Todavia isto só foi possível depois da realização de vários testes para se obter um nível razoável de balanceamento entre estas duas fases em cada ciclo [SCH96, SG01]. Este nível é controlado por três parâmetro distintos. Estes parâmetros e os testes que permitiram que se encontrasse este nível de equilíbrio serão discutidos com maiores detalhes na seção 5.1.3.

Lista de Movimentos Candidatos

Para as últimas versões da heurística de busca tabu foi empregado um mecanismo simples para gerar uma lista de movimentos candidatos. O método usado é basicamente um algoritmo que escolhe aleatoriamente m movimentos e, dentre estes, escolhe o melhor, ou seja, aquele que representará um maior ganho (ou menor perda) na qualidade da solução.

Um aspecto que deve ser salientado é o fato de que todos os movimentos aleatoriamente escolhidos pela função, passam pelo filtro da lista tabu e do critério de aspiração. Assim, esta função não retorna movimentos que sejam tabu-ativos.

Lista Tabu

A implementação da lista tabu sofreu modificações substanciais nas novas versões do resolvedor. Estas modificações implicaram numa maior eficiência e flexibilidade desta estrutura, importantíssima para a heurística em questão.

A estrutura anterior da lista tabu era constituída por uma lista circular que armazenava os movimentos tabu e uma tabela de espalhamento onde era possível verificar em tempo quase constante o estado tabu de um dado movimento. Era uma estrutura relativamente eficiente, porém podia ser melhorada.

Na versão final do resolvedor a lista tabu passou a ser implementada apenas por uma tabela de espalhamento, descartando a lista circular e diminuindo, assim o *overhead* desnecessário. Agora, ao invés de guardarmos apenas 0 ou 1 na tabela de espalhamento para indicar se o movimento é ou não tabu-ativo, guardamos o número da última iteração do algoritmo em que foi atribuído o estado de tabu-ativo ao movimento. Assim, tendo-se esta informação sobre todos os movimentos, o número da iteração atual e o comprimento da lista podemos facilmente verificar o estado tabu do movimento. Deste modo:

```
SE ( hash[ Movimento ] + listaTabu.Comprimento > iteracaoAtual ) ENTÃO
  Movimento é TABU-ATIVO.
SENÃO
  Movimento não é TABU-ATIVO.
```

Esta implementação é mais eficiente porque evita o manuseio da lista circular que antes controlava os movimentos tabu. E a maior flexibilidade dela refere-se ao fato de ser muito fácil modificar o comprimento da lista tabu. Antes era necessário realocar toda a lista circular com um novo comprimento. Agora basta setar o valor do campo inteiro `comprimento` da estrutura para o valor desejado.

Outro aspecto importante da nova implementação da lista tabu refere-se à definição "do que é tabu" na nova modelagem. Agora existem duas definições para o estado de tabu-ativo de um dado movimento. Elas são:

- **Tabu Simples**: esta é definição que foi usada durante quase todo o desenvolvimento da heurística do resolvedor. Ela considera como tabu-ativo, durante n iterações, o próprio movimento que foi executado, isto é, a quádrupla: *<g, t, p1, p2>*. Esta implementação não garante que o movimento tabu não seja executado, ou melhor, ela permite que o efeito do movimento possa ser feito de forma indireta. Ou seja, uma dada sequência de movimentos pode implicar no mesmo efeito de um movimento que é tabu-ativo.

- **Tabu Estrito**: esta implementação é mais restritiva que a anterior e não permite a ocorrência do problema relatado acima. Nesta definição de tabu-ativo não é o movimento que é posto na lista tabu e sim a quádrupla: <grade g, turma da grade t, slot s, matéria m >. O significado desta definição é o seguinte: a matéria m da turma t da grade g não pode ser inserida no slot s desta grade durante n iterações. Veja que o uso desta a lista tabu não impede o movimento, mas sim os seus efeitos.

E, desta forma, não é possível que o estado de tabu seja violado por uma sequência de movimento que não é tabu.

O algoritmo empregado no resolvedor está apto a trabalhar com ambas as definições da lista tabu. Na próxima seção serão apresentados os resultados dos testes que avaliam a eficácia da primeira implementação (tabu simples).

Critério de Aspiração

O critério de aspiração atrelado à condição de tabu-ativo dos movimentos continua, nesta nova modelagem, sendo o *improved best*, o qual relaxa o tabu-ativo caso o movimento faça com que a solução obtenha ganhos quantitativos em sua função objetivo.

Pseudocódigo da Heurística Implementada

Esta seção descreve o algoritmo heurístico implementado na versão 5 do resolvedor através de um pseudo-código. Como já foi comentado, o algoritmo possui um laço maior cujo critério de parada é um dado número fixo de ciclos, determinado pelo parâmetro `CICLOS_MAXIMO`. Internamente a este laço existem dois laços menores: o primeiro executa um processo diversificado e o segundo realiza um processo intensificado. Cada um destes laços internos possui o seu próprio critério de parada, baseados na idéia de um dado número de iterações sem melhorar a solução atual. Na fase diversificativa o parâmetro que indica o número de iterações sem melhora é dado por `ISM1`. Na fase intensificada este parâmetro é dado por `ISM2`. O algoritmo mantém duas soluções simultaneamente: a solução atual, denominada `ATUAL`, e uma solução de elite, dita `ELITE`. Além disso existe o parâmetro `DIV2` que indica o comprimento da lista de movimentos candidatos da fase de diversificação.

```
/* BUSCA TABU ALTERNADA II */

INICIALIZAÇÃO:

   ATUAL = geraSolucaoInicial();          //solução inicial aleatoriamente gerada
   ELITE = ATUAL;

BUSCA LOCAL:

   FAÇA ENQUANTO CICLOS < CICLOS_MÁXIMO
   {
      nISM1 = 0;
      nISM2 = 0;

      /* FASE DE DIVERSIFICAÇÃO */
      FAÇA ENQUANTO nISM1 <= ISM1
      {
         Lista = listaDeMovimentosCandidatos(DIV2);
         i = 0;
```

```
         ENQUANTO (Lista.Movimento[i] for TABU e não acatar o CRITÉRIO DE ASPIRAÇÃO)
           i++;

         MOVIMENTO = Lista.Movimento[i];
         ATUAL = executaMovimento(MOVIMENTO);
         listaTabuInsere(MOVIMENTO);

         SE MOVIMENTO reduz o valor da funcaoObjetivo ENTÃO
            ELITE = ATUAL;
            nISM1 = 0;
         SENÃO
            nISM1++;
      }

      /* FASE DE INTENSIFICAÇÃO */
      FAÇA ENQUANTO nISM2 <= ISM2
      {
         MOVIMENTO = movimentoAleatório();

         SE MOVIMENTO reduz o valor da funcaoObjetivo ENTÃO
            ATUAL = executaMovimento(MOVIMENTO);
            listaTabuInsere(MOVIMENTO);
            ELITE = ATUAL;
         SENÃO
            nISM2++;
      }

      CICLOS++;
   }

RETORNA ELITE;
```

5 Testes

O objetivo desta seção é apresentar os resultados de uma gama de testes realizados sobre as diversas heurístcas implementadas e comparar os resultados obtidos. Para este propósito dividiremos a seguinte seção em três partes: primeiramente iremos detalhar todos os testes realizados sobre a implementação de Busca Tabu enfocando diversos aspectos da mesma através da análise de vários parâmetros. Em seguida apresentaremos os resultados obtidos pelas três heurísticas mais simples (*hill climbing*, *multi-start* e *simulated annealing*). Por fim faremos uma comparação entre todos os resultados o que permitirá obter conclusões pertinentes sobre as diversas implementações.

A metodologia utilizada nestes testes buscou avaliar cada um dos parâmetros isoladamente, isto é, no estudo de um determinado parâmetro do algoritmo todos os outros parâmetros eram mantidos fixos. Em alguns casos alteramos dois parâmetros simultaneamente de modo a avaliar a possível relação entre os mesmos. Os testes também foram

executados sucessivas vezes, para obtermos o melhor refinamento possível.

Antes de explicitar a metodolodia usada nos testes será apresentada uma proposta adotada para a classificação das soluções geradas pelo resolvedor de acordo com a sua qualidade.

1. **Solução Infactível:** solução que viola alguma restrição fisicamente forte[2].
2. **Solução Factível:** solução que não viola nenhuma restrição fisicamente forte.
3. **Solução Viável:** solução factível que não viola nenhuma aula n-upla e que contém, no máximo, 150 minutos de janelas nos horários dos professores.
4. **Solução Boa:** solução viável que não apresenta nenhuma janela de horário.

Para cada valor de um parâmetro (ou conjunto de parâmetros) avaliado foi feita uma série de 30 execuções no algoritmo, nestas execuções eram coletados os seguintes dados:

- o tempo necessário para o algoritmo executar n iterações no algoritmo (no qual n varia de acordo com a heurística de busca em questão).
- fixado este número de ciclos para o algoritmo executar, verifica-se a qualidade (através do valor da função objetivo e da classificação dada acima) da melhor solução encontrada em cada execução.
- calculam-se as médias e os desvios médios dos valores obtidos nas 30 execuções.

Com os dados acima é possível avaliar, para um determinado conjunto de parâmetros, o tempo de processamento consumido pelo resolvedor, a qualidade das soluções em um dado tempo fixo de execução e a estabilidade do algoritmo baseando-se no desvio médio dos valores obtidos nos testes.

Foram realizados testes baseados nas especificações do colégio Stella Maris. Os valores de tempo tabelados nas próximas seções mostram o desempenho do algoritmo executando em uma plataforma Intel Pentium IV de 2,4 GHz presente nos laboratórios de ensino do Instituto de Computação da Unicamp.

5.1 Resultados Para a Busca Tabu

Nas subseções a seguir iremos detalhar o estudo de cada uma das classes de parâmetros do algoritmo de busca tabu mostrado na seção 4.5.2.

5.1.1 Valores de Penalidades Aplicados

Os valores atribuídos às violações das restrições fortes são os mais críticos para a velocidade de convergência do algoritmo no problema de agendamento. Nos testes realizados foram avaliadas dezenas de combinações para os valores de penalidades fortes. As penalidade

[2]São consideradas restrições fisicamente fortes as seguintes: cumprimento da carga horária, satisfação das indisponibilidades, eliminação de aulas simultâneas e sobreposições.

fracas foram desconsideradas nestes testes, já que o valor atribuído a estas é baseado no seu grau de prioridade e não na eficiência que a mesma trará para o algoritmo.

As penalidades referentes às restrições fortes analisadas são as seguintes: aulas simultâneas (**SIM**), indisponibilidades (**IND**), sobreposições (**SBR**), janelas[3] (**JAN**), aulas n-uplas (**ANU**) e aulas coordenadas (**CRD**).

A tabela 1 mostra algumas das combinações de valores de penalidades testados apresentando a qualidade média das soluções geradas e o percentual de soluções viáveis e boas encontradas pelas respectivas configurações.

SIM	**IND**	**JAN**	**SBR**	**ANU**	**CRD**	**Qualidade**[1]	**Viáveis**	**Boas**
100	400	1	100	200	100	47,5	95%	40%
100	400	1	100	150	100	30,5	95%	65%
100	500	1	100	200	100	26,4	97%	63%
250	800	2	250	400	100	142,5	85%	25%
150	400	2	150	250	100	82,5	60%	50%
100	200	1	100	150	100	44,0	75%	55%
200	**700**	**2**	**200**	**250**	**100**	34,0	97%	74%

1. Média do valor da função objetivo da melhor solução obtida em 300 ciclos de processamento do resolvedor.

2. Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris. Parâmetros fixados: `LISTA_TABU` = 60, `DIVERSIFICACAO1` = 100 isms[3], `DIVERSIFICACAO2` = 35, `INTENSIFICACAO` = 3000 isms, `CICLOS = 300`.

3. isms = Iterações Sem Melhora. Critério de Parada adotado para esta fase da busca.

Tabela 1: Testes para valores das penalidades fortes

A tabela 1 mostra boas configurações encontradas durante a fase de testes. Pode-se perceber que o valor atribuído à violação de INDISPONIBILIDADES é, dentre todas as penalizações, o maior. A convergência do resolvedor para soluções factíveis é muito sensível a este parâmetro. O valor atribuído à violação de aulas n-uplas recebe o segundo maior valor. Isso dá prioridade à correção deste tipo de violação, caso contrário o resolvedor geraria agendas factíveis porém inviáveis na prática, já que conteriam grandes quantidades de aulas triplas. As restrições de AULAS SIMULTÂNEAS e SOBREPOSIÇÕES penalizam as soluções com um valor idêntico, já que seus efeitos são análogos. As violações de janelas de horários são penalizadas proporcionalmente ao tamanho da janela, tornando o algoritmo mais equilibrado na correção de janelas. O valor atribuído para a não satisfação de aulas coordenadas foi mantido fixo durante os testes já que esta restrição é muito dependente da instância do problema tratado. No caso do Stella Maris, por haverem apenas 3 aulas coordenadas, não houve dificuldade na satisfação desta restrição.

O conjunto de valores escolhidos para penalizar a violação de restrições fortes é o que se encontra na última linha da tabela (em negrito). Os valores atribuídos às restrições fracas serão apresentados na seção 5.1.4.

[3]O valor atribuído à penalidade por janelas no horário do professor é proporcional ao tempo da janela, ou seja, atribui-se como penalidade um valor fixo vezes o número de minutos da janela

5.1.2 Lista Tabu

Os testes que avaliam a lista tabu buscam medir sua eficácia baseando-se no seu comprimento. A tabela 2 mostra os resultados obtidos por esta implementação variando-se o comprimento da lista tabu simples no intervalo de 0 a 200 movimentos tabu.

COMPRIMENTO	**Qualidade**[1]	**Viáveis**
0	324,8	50%
5	327,4	57%
10	327,8	67%
20	365,8	60%
30	293,2	63%
40	325,5	70%
50	306,7	77%
60	**261,7**	**87%**
70	270,6	77%
80	273,4	77%
100	321,9	60%
200	360,6	50%

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris. Parâmetros fixados: `DIVERSIFICACAO1` = 100 isms, `DIVERSIFICACAO2` = 30, `INTENSIFICACAO` = 2000 isms, `CICLOS = 300`.

1. Média do valor da função objetivo da melhor solução obtida em 300 ciclos de processamento do resolvedor.

Tabela 2: Testes para comprimento da Lista Tabu Simples

A lista tabu tem um efeito considerável sobre a qualidade das soluções retornadas pelo sistema. Porém o tempo de processamento praticamente independe desta estrutura, já que o mesmo é fortemente determinado pelo número de iterações das fases de intensificação e diversificação. Assim, só faz sentido analisar os efeitos da lista em relação à qualidade das soluções e o percentual de soluções viáveis (e boas) encontradas. Vemos na tabela 2 que o melhor valor encontrado corresponde a um comprimento de lista tabu igual a 60 movimentos. Valores menores que 50 praticamente não têm o efeito desejado. Os maiores que 100 são muito restritivos e acabam dificultando a busca, impedindo-a de propagar-se pelas diferentes vizinhanças.

A figura 1 mostra o gráfico correspondente à tabela 2, permitindo-nos uma melhor vizualização dos pontos críticos e dos melhores valores para o comprimento da lista tabu.

O algoritmo de busca tabu desenvolvido possibilita que variemos o comprimento da lista tabu durante a busca, permitindo que a busca se adapte ao espeaço que está sendo avaliado. Assim, podemos relaxar a busca reduzindo o comprimento da lista tabu em regiões promissoras. Em contrapartida, podemos restringir ainda mais a busca em regiões de baixa qualidade fazendo a lista tabu ficar maior.

Uma idéia bastante simples de emprego da lista tabu com comprimento variável é aplicar um comprimento assimétrico desta lista nas duas fases do algoritmo: diversificação e

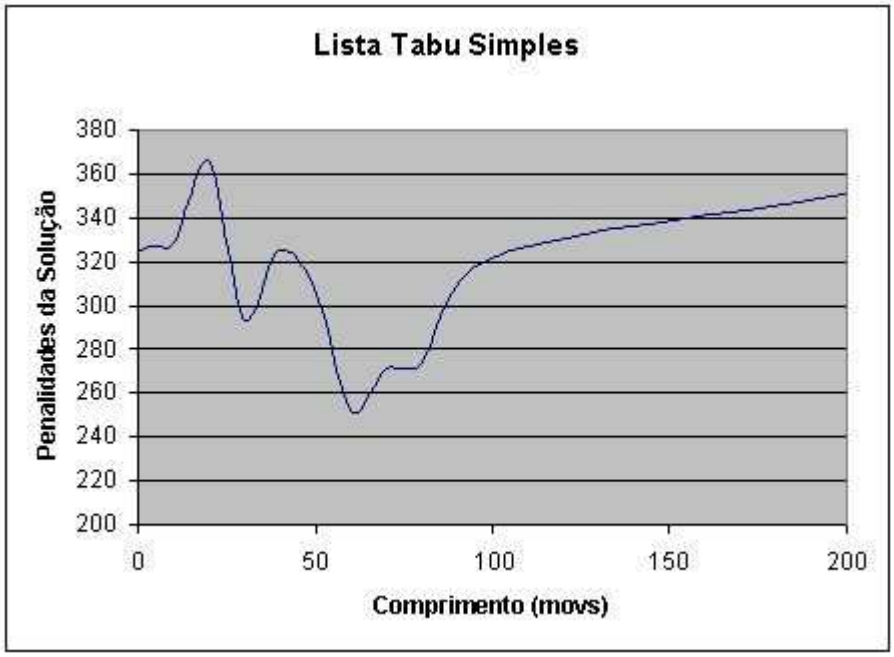

Figura 1: Efeitos do Comprimento da Lista Tabu sobre a Qualidade das Soluções Geradas

intensificação. Podemos avalair como a lista interfere em cada uma destas fases. Nesse sentido, nos testes realizados, obtivemos os resultados dados na tabela 3.

Na tabela 3 COMPRIMENTO_INT e COMPRIMENTO_DIV referem-se, respectivamente, ao comprimento da lista tabu nas fases de intensificação e diversificação. Os resultados obtidos não justificam o emprego de uma lista tabu com tamanho assimétrico em relação às diferentes fases, visto que o melhor valor obtido foi o comprimento igual a 60 para as duas fases (intensificação e diversificação).

5.1.3 Diversificação e Intensificação

O balanceamento adequado entre as fases de intensificação e diversificação do algoritmo descrito na seção 5 resulta numa convergência rápida e segura do resolvedor. Tal balanceamento se dá pelo critério de parada adotado por cada uma dessas duas fases. Estes critérios de parada irão determinar por quantas iterações o algoritmo irá permanecer nos respectivos laços.

Para a avaliação deste balanceamento é necessário verificar o comportamento de três parâmetros distintos do algoritmo de busca tabu:

- **INTENSIFICACAO**: refere-se ao critério de parada da fase de intensificação da busca. É medido em iterações sem melhorar (sigla, ism) a solução atual.
- **DIVERSIFICACAO1**: de modo similar ao parâmetro anterior, este define o critério de parada da fase de diversificação em iterações sem melhora.

COMPRIMENTO_INT	COMPRIMENTO_DIV	Qualidade[1]	Viáveis
30	0	179,7	90%
60	0	265,9	73%
120	0	158,3	97%
0	30	171,0	90%
0	60	187,6	77%
0	120	238,1	80%
120	30	176,1	93%
120	60	161,8	93%
120	120	155,3	97%
60	60	145,8	97%

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris. Parâmetros fixados: `DIVERSIFICACAO1` = 100 isms, `DIVERSIFICACAO2` = 30, `INTENSIFICACAO` = 4000 isms, `CICLOS` = 300.

1. Média do valor da função objetivo da melhor solução obtida em 300 ciclos de processamento do resolvedor.

Tabela 3: Testes para comprimento da Lista Tabu Assimétrica

- **DIVERSIFICACAO2**: como já foi explicado anteriormente (seção 4), cada iteração da fase de diversificação do algoritmo avalia uma lista de movimentos candidatos escolhendo o melhor deles. Este parâmetro indica exatamente o comprimento desta lista de movimentos.

Estes três parâmetros são fundamentais para determinar o tempo de convergência e a qualidade das soluções encontradas na busca. A tabela 4 nos dá uma visão geral de como estes parâmetros afetam a qualidade das soluções e o tempo de processamento do resolvedor.

As figuras 2 e 3 apresentam os efeitos do parâmetro DIV1, que determina o tempo da fase de diversificação da busca. Veja que o valor ótimo mostrado pelo gráfico 2 é DIV1 = 150. Nesse ponto conseguimos obter as soluções com melhor qualidade. Valores menores que 100 fazem com que a busca apresente baixa diversidade de soluções, enquanto que para valores superiores a 200 temos uma forte diversificação que acaba desviando a busca dos mínimos locais. O gráfico 3 mostra a proporcionalidade entre o valor de DIV1 e o tempo total de processamento.

As figuras 4 e 5 nos dão um panorama similar ao anterior para o parâmetro de intensificação (INT). Nestes gráficos vemos que, quanto maior a fase de intensificação maior é a qualidade das soluções geradas pelo resolvedor. O tempo de processamento também é diretamente proporcional ao valor de INT. Assim, temos que propor para INT um valor que permita que o resolvedor encontre boas soluções em um tempo aceitável. Este valor encontra-se no intervalo de 4000 a 10000 isms (iterações sem melhora). Com INT = 4000 a qualidade das soluções não é tão boa. Já INT = 10000 é muito dispendioso em relação ao tempo. Assim, para as especificações do Colégio Stella Maris, concluímos que o melhor compromisso entre qualidade das soluções e tempo de processamento se dá em torno de INT = 6000.

DIV1	DIV2	INT	Qualidade[1]	Tempo Médio
1000	100	30	688,6	124
3000	100	30	154,3	249,2
5000	100	30	96,7	356,9
6000	100	30	63,9	405,6
10000	100	30	56,1	602,5
3000	30	30	192,2	150,5
3000	50	30	186,4	182,8
3000	100	30	187,2	284,2
3000	150	30	167,7	303,3
3000	200	30	182,8	349,4
3000	300	30	230,1	435,2
3000	70	15	352,4	140,2
3000	70	20	270,0	160,7
3000	70	30	106,0	199,1
3000	70	35	58,2	210,6
3000	70	40	157,8	260,8

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris. Parâmetros fixados: `LISTA_TABU` = 60, `CICLOS = 300`.

1. Média do valor da função objetivo da melhor solução obtida em 300 ciclos de processamento do resolvedor.

Tabela 4: Relação temporal entre as fases de diversificação e intensificação – Colégio Stella Maris

Não iremos entrar em maiores detalhes sobre os testes realizados com o parâmetro DIV2 (que indica o comprimento da lista de movimentos candidatos da fase de diversificação). Os testes que foram feitos permitiram verificar que DIV2 = 35 é um valor bastante razóável, balanceando de forma coerente um bom nível qualitativo das soluções geradas com um tempo de processamento viável.

Por fim façamos uma análise sobre o balanceamento da busca local comparando os valores de DIV1, DIV2 e INT. É notável que INT assume valores bem maiores que DIV1, porém isso não é suficiente para dizer que a fase de intensificação possui um tempo de execução maior que a fase de diversificação. É bom também lembrar que em cada iteração da fase intensificada é analisado um único movimento, enquanto que na fase diversificada são analisados DIV2 movimentos (número de movimentos candidatos). Assim, uma análise correta do balanceamento da busca local deve comparar INT com DIV1×DIV2. E, de acordo com os parâmetros escolhidos, temos: INT = 6000 e DIV1×DIV2 = 5250. Repare que as duas fases são comparáveis, isto é, gastam um tempo de processamento aproximadamente igual.

5.1.4 Resumo - Melhores Parâmetros da Busca Tabu

Depois de toda a série de testes, podemos apresentar uma tabela com os melhores valores obtidos para os parâmetros usados no resolvedor. Os resultados aqui mostrados já foram

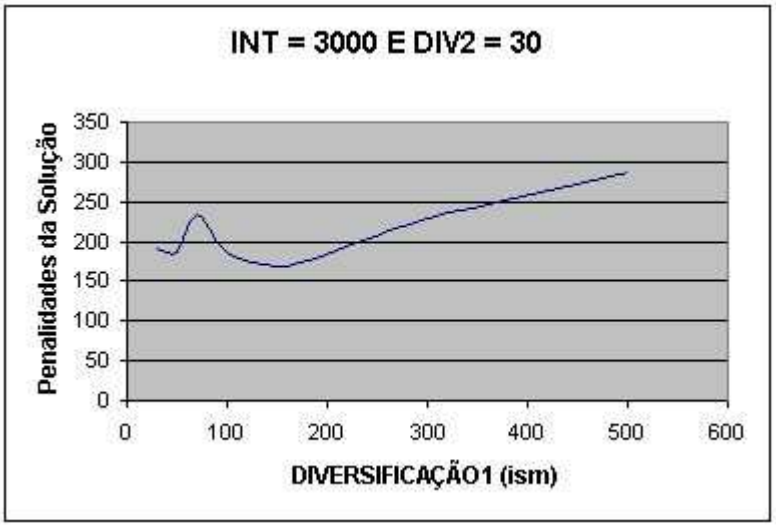

Figura 2: Efeitos do Parâmetro DIV1 sobre a Qualidade das Soluções Geradas

discutidos nas seções anteriores, e a decisão por estes valores já foi comentada. A tabela 5 sintetiza todo o conjunto de parâmetros adotados até o presente momento na versão final do resolvedor.

5.2 Resultados Para as Outras Heurísticas de Busca Local

Agora, mostraremos de forma concisa os resultados obtidos pelas outras 3 heurísticas de busca local. É bom frisar que temos 3 diferentes implementações para o *simulated annealing* (veja seção 4.3), assim temos um total de 5 implementações de busca local para o resolvedor do agendamento. Estas cinco implementações são: *hill climbing* (**HC**), *multi-start* (**MS**), *simulated anneling* com temperatura fixa (**SATF**), *simulated annealing* com temperatura variável 1 e 2 (respectivamente, **SATV1** e **SATV2**).

Em todos os testes iremos empregar os valores de penalidades obtidos nos testes realizados com a busca tabu e que estão listados na tabela 5.1.4.

Hill Climbing

Utilizamos três diferentes configurações para a execução dos testes nesta heurística. Estas três configurações estão descritas a seguir em ordem crescente de dificuldade:

- **CONFIG1:** na primeira configuração só levamos em consideração as restrições fisicamente fortes (incluindo aulas n-uplas), deixando de fora janelas de horários e todas as restrições fracas.
- **CONFIG2:** aqui incluimos a restrição para o tratamento de janelas de horários, porém as restrições fracas continuam sendo negligencidas.

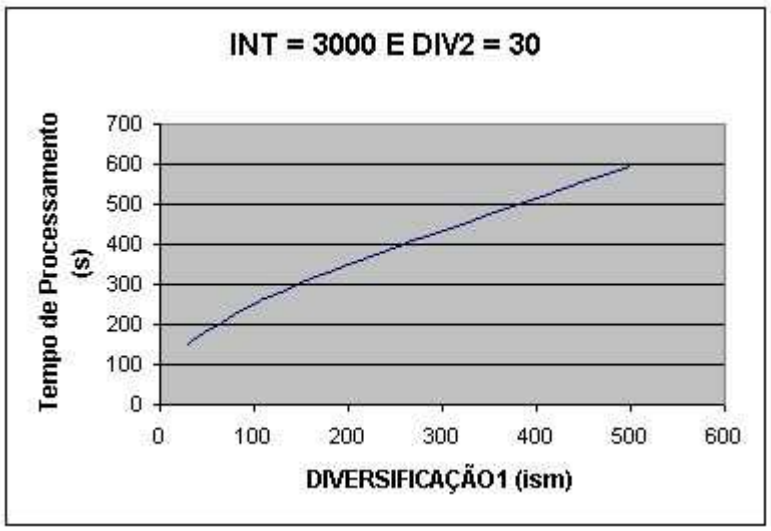

Figura 3: Efeitos do Parâmetro DIV1 sobre o Tempo de Processamento do Resolvedor

- **CONFIG3:** nesta configuração todas as restrições (fracas e fortes) são consideradas, utilizando os mesmos valores de penalidades obtidos como melhores pela busca tabu.

A tabela 6 resume os resultados dos testes aplicados, informando a convergência, o tempo utilizado e a qualidade média da melhor solução gerada. Veja que este método de busca local só obteve convergência nas duas configurações mais simples do problema. Quando ocorre a inclusão das restrições fracas, o nível de convergência é nulo. Outra observação interessante é que a simples inclusão da restrição de janelas em CONFIG2 fez com que o tempo de convergência pasasse de cerca de 4 segundos para mais de 10 minutos. Este é um grave problema da busca local descendente, onde a adição de novas restrições faz com que o tempo de busca cresça muito rapidamente.

Multi-Start

Testes similares ao anterior foram executados para o multi-start. A tabela 7 resume os resultados alcançados por esta heurística nas três configurações do problema de agendamento já descritas na seção anterior. Esta heurística obteve um desempenho um pouco superior à anterior, já que conseguiu elevar o percentual de convergência para a configuração CONFIG2 e reduziu o tempo de convergência em cerca de 20%. Porém ainda não foi capaz de obter uma taxa de convergência significativa para a configuração 3, ou seja, não conseguiu resolver satisfatoriamente o problema na presença das restrições fracas.

Simulated Annealing

Para testar esta heurítica utilizaremos apenas a configuração CONFIG3, já que nas outras duas conseguiu-se convergência de 100% usando o *simulated annealing*. Como dis-

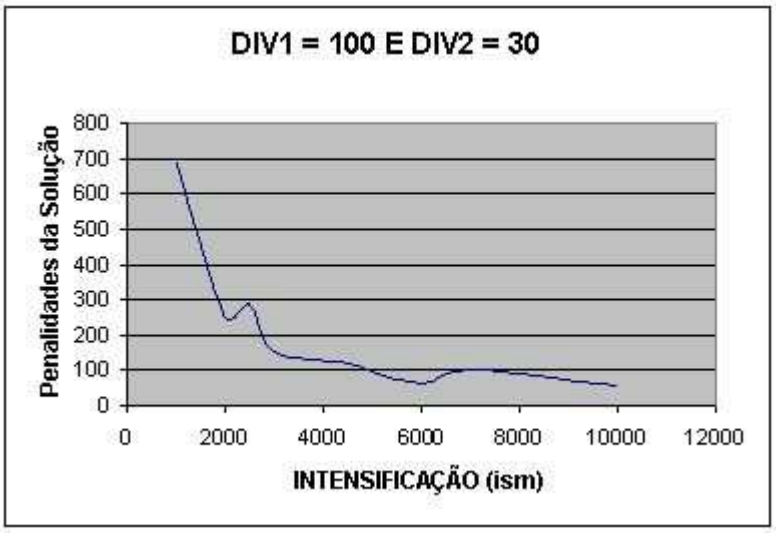

Figura 4: Efeitos do Parâmetro INT sobre a Qualidade das Soluções Geradas

semos no início da seção, existem 3 versões distintas (SAF, SATV1 e SATV2) para esta heurística, aqui compararemos os resultados obtidos pelas três variando os seus parâmetros fundamentais (TEMPERATURA e VALOR_LIMITE).

A tabela 8 resume todos as bateladas de testes executadas. Iremos, primeiro, entender os resultados obtidos por cada uma das implementações isoladamente, em seguida faremos uma comparação entre as três.

- **SAF**: esta implementação adota uma idéia bem simples de diversificação: permitir movimentos que degeneram a função objetivo com base em uma probabilidade fixa. Vemos que, nos testes realizados, o melhor valor para tal probabilidade foi o de 0,01%, isto é, para cada 10.000 movimentos, em média, um deles irá degenerar a solução atual. Este simples mecanismo permite que a busca escape dos ótimo locais e mude de direção.

- **SATV1**: neste caso o valor probabilístico para a execução de movimento considera um relação entre a perda ocasionada pela tomada do movimento e o valor da função objetivo da solução inicial. Aqui o valor de destaque foi VALOR_LIMITE $= 3,5\%$.

- **SATV2**: agora o valor probabilístico relaciona a perda ocasionada pelo movimento e o valor da função objetivo atual. Assim, temos uma implementação mais dinâmica que a anterior, já que o valor depende da solução atual e não somente da solução inicial (que possui um valor de função objetivo constante). Esperava-se que com tal dinamicidade tivéssemos uma melhor adaptação do valor probabilístico à busca local, porém os resultados obtidos não foram satisfatórios, já que em muitas execuções esta metodologia sequer encontrava soluções factíveis.

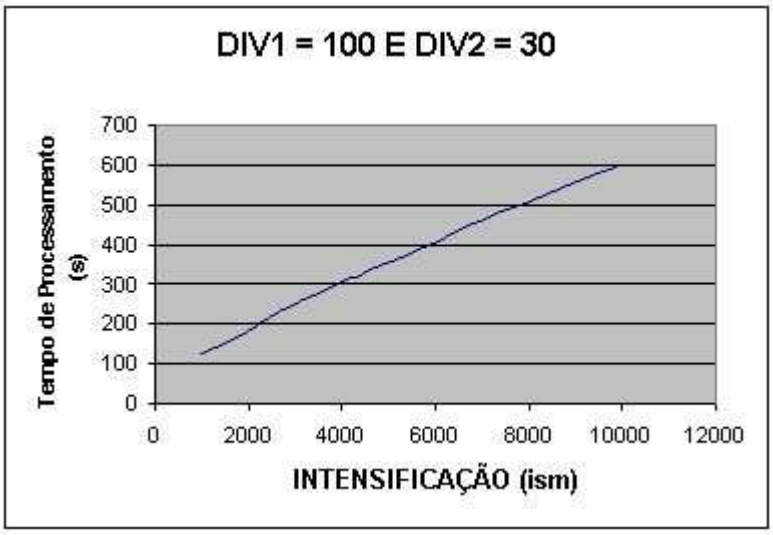

Figura 5: Efeitos do Parâmetro INT sobre o Tempo de Processamento do Resolvedor

Vimos que o emprego de valores probabilísticos de seleção de movimento feitos de forma mais estática (uma constante em SAF e a perda dividida por uma constante em SATV1) obtiveram os melhores resultados para este modelo de resolução estocástica do problema de agendamento. A terceira modelagem, SATV2, não obteve êxito em seus propósitos.

5.3 Comparando os Resultados Obtidos

Depois de listarmos todos os resultados podemos delinear um estudo comparativo entre as heurísticas testadas. A tabela 9 nos dá uma visão comparativa global entre os melhores resultados obtidos por cada uma das heurísticas.

As duas implementações mais simples, busca local descendente e multi-start, tiveram um desempenho ruim na resolução do agendamento com o conjunto completo de restrições. Isto ocorre em decorrência do probre mecanismo de diversificação destas heurísticas que faz com que busca realizada fique presa em mínimos locais de baixa qualidade.

As três versões do método de busca local estocástica (*simulated annealing*) já conseguiram obter resultados satisfatórios, embora inferiores à Busca Tabu. Um estudo comparativo sobre os resultados obtidos pelas três diferentes implementações usadas já foi feito na seção 5.2.

Pelos resultados percebe-se claramente que a Busca Tabu obteve soluções de melhor qualidade em um tempo inferior às outras heurísticas. Isto é decorrência de vários fatores associados para tal sucesso, tais como: o balanceamento entre a busca intensificada e diversificada, o emprego da lista de movimentos candidatos e o emprego da lista tabu aliada a um critério de aspiração.

Parâmetro	**Valor**	**Observação**
Fase de Diversificação	150 ism	Critério de parada adotado
Fase de Intensificação	6000 ism	Critério de parada adotado
Comprimento da Lista de Movimentos Candidatos	35 movs.	
Comprimento da Lista Tabu	60 movs.	
Penalidade para Aulas Simultâneas	200	Restrição Forte
Penalidade para Sobreposição de Horários	200	Restrição Forte
Penalidade para Violação de Indisponibilidade	700	Restrição Forte
Penalidade para Janela de Professores	2	Restrição Forte
Penalidade para Violação de Aula N-upla	250	Restrição Forte
Penalidade para Violação de Aula Coordenada	100	Restrição Forte
Penalidade para Violação de Preferências	4	Restrição Fraca
Penalidade para Não Satisfação de Dayoff	8	Restrição Fraca
Penalidade para Violação de Aula Dupla Separada	10	Restrição Fraca
Penalidade para Má Distribuição das Aulas	**X**	Conj. de Regras

Tabela 5: Melhores parâmetros

Configuração	**Convergência**[1]	**Tempo**	**Qualidade**[2]
CONFIG1	100%	4,4s	0
CONFIG2	70%	644,0s	242
CONFIG3	0%	–	–

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris.

1. Indica o percentual de execuções do algoritmo que convergiram para uma solução factível.
2. Média do valor da função objetivo da melhor solução obtida em 15 minutos de processamento do resolvedor.

Tabela 6: Testes de Convergência do Hill Climbing

5.4 Comparação Com Outras Técnicas de Resolução

Outros trabalhos do GOA[4] 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 utiliza Algoritmos Genéticos. 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.

Busca Tabu e Algoritmos Genéticos: De modo geral, ambas as heurísticas obtiveram

[4]Grupo de Otimização Aplicada do Instituto de Computação da Unicamp. Home-page: http://goa.pos.ic.unicamp.br/otimo.

Configuração	**Convergência**[1]	**Tempo**	**Qualidade**[2]
CONFIG1	100%	4,7s	0
CONFIG2	90%	492,0s	197
CONFIG3	10%	–	–

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris.

1. Indica o percentual de execuções do algoritmo que convergiram para uma solução factível.
2. Média do valor da função objetivo da melhor solução obtida em 15 minutos de processamento do resolvedor.

Tabela 7: Testes de Convergência do Multi-Start

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 7.

5.5 Resultados Obtidos Pelo Resolvedor Usando Busca Tabu

Depois da análise dos dados das seções anteriores é interessante também relatar o comportamento do resolvedor sobre um determinado problema, usando os parâmetros encontrados na fase de testes. Aqui, iremos descrever graficamente o funcionamento do algoritmo de busca tabu e daremos um exemplo de solução encontrada por ele.

O gráfico representativo do funcionamento do algoritmo é bastante simples, mostrando apenas como evolui o valor da função objetivo durante o tempo de busca, isto é, a cada ciclo. O gráfico 6 tem no eixo horizontal o número de ciclos da busca e no eixo vertical o valor da função objetivo naquela iteração. Neste gráfico, a linha mais clara representa a evolução da solução atual, enquanto que na linha mais escura podemos ver a qualidade da melhor solução gerada até um dado instante.

Versão	VALOR_LIMITE	Qualidade[2]
SAF	**0,0001**	**198,7**
SAF	0,0002	214,4
SAF	0,0003	260,4
SAF	0,0005	425,9
SATV1	0,002	393,2
SATV1	0,004	329,2
SATV1	0,008	271,1
SATV1	0,010	307,0
SATV1	0,020	317,0
SATV1	0,025	352,8
SATV1	**0,035**	**232,4**
SATV1	0,040	286,4
SATV1	0,050	381,2
SATV2	0,005	365,0
SATV2	0,010	426,0
SATV2	0,020	398,8
SATV2	0,030	379,8
SATV2	**0,040**	**301,6**
SATV2	0,050	398,2

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris.

1. Indica o percentual de execuções do algoritmo que convergiram para uma solução factível.
2. Média do valor da função objetivo da melhor solução obtida em 15 minutos de processamento do resolvedor. Os melhores valores obtidos estão em **negrito**.

Tabela 8: Testes nas 3 Versões do Simulated Annealing

Um Exemplo de Solução Gerada

As figuras 7 e 8 mostram, respectivamente, a alocação dos professores paras as turmas de ensino fundamental (grade com 25 slots) e ensino médio (grade com 30 slots). Estas figuras apresentam a solução da seguinte maneira: nas linhas temos as turmas de alunos e nas colunas temos os horários de aula da semana. As células contém o código correspondente ao professor que está ministrando aula naquele período para a turma especificada. A agenda gerada recebeu uma penalidade 92, referente aos valores dados na tabela 5. Esta é uma solução classificada como **boa**, isto é, não apresenta janelas nos horários dos professores. A solução apresenta um bom espalhamento das aulas por níveis de dificuldade e possui apenas duas aulas duplas não-geminadas (porém é muito frequente que o resolvedor construa agendas isentas deste tipo de violação). Esta solução conseguiu também atender a cerca de 50% das preferências de horários requeridas pelos professores, tanto preferências por horários livres quanto dias-livres.

Heurística	**Convergência**	**Qualidade**[1]	**Tempo**
Hill Climbing[2]	Converge apenas na ausência de restrições fracas	–	–
Multi-Start[2]	Converge apenas na ausência de restrições fracas	–	–
SAF	Converge em 90% dos casos	198,7	600 s
SATV1	Converge em 90% dos casos	232,4	600 s
SATV2	Converge em 70% dos casos	301,6	600 s
Busca Tabu	Converge em 99% dos casos	63,9	405,6 s

Testes realizados sobre as especificações das turmas de ensino fundamental do Stella Maris.

1. Média do valor da função objetivo da melhor solução obtida em 15 minutos de processamento do resolvedor. Os melhores valores obtidos estão em **negrito**.
2. Não julgou-se necessário especificar nem a qualidade final das soluções geradas nem o tempo de convergência para estas duas heurísticas já que as mesmas não convergiram satisfatoriamente nas configurações reais do agendamento, ou seja, aquelas que incorporam todo o conjunto de restrições.

Tabela 9: Comparação Entre as Diversas Heurísticas de Busca Local

6 Implementação da Interface

Além do programa resolvedor faz-se necessário, para tornar o sistema utilizável, a construção de uma interface gráfica amigável para usuários leigos. Por esse motivo, a equipe de iniciação científica do GOA trabalhou em conjunto para implementar uma interface única e independente do resolvedor de agendamento usado.

6.1 Entrada e Saída de Dados no Sistema

O sistema desenvolvido no projeto de Iniciação Científica, incluindo a interface e os vários núcleos, estabelece uma padronização para a passagem dos parâmetros - do problema e do algoritmo - a serem utilizados. 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.

6.1.1 Comunicação entre Interface e núcleo resolvedor

É necessário que se estabeleça uma forma de comunicação entre a interface, desenvolvida em Delphi [CAN98], e os núcleos resolvedores como o de Busca Tabu, desenvolvido 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 sessão 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

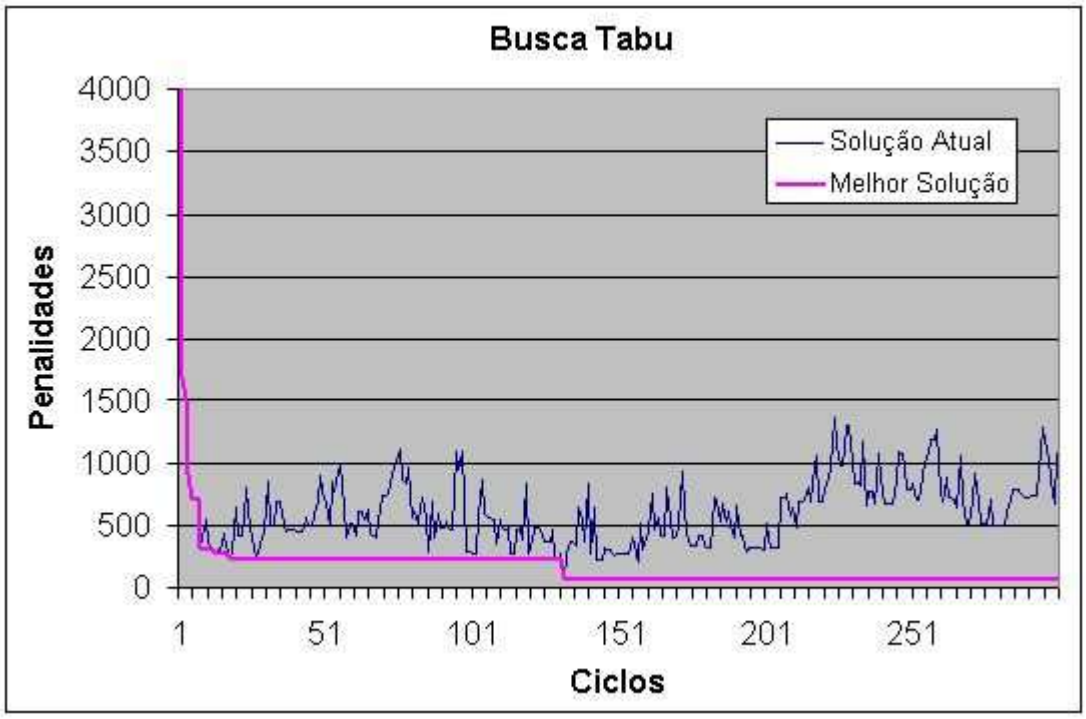

Figura 6: Evolução Temporal da Busca Tabu

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)[5].

Poderiam ser utilizados outros formatos, como o XML (*Extensible Markup Language*), mas o grupo de alunos chegou a conclusão de que não valeria a pena o esforço 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.

1. **Sessão** `escola`: Dados gerais da escola.

   (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.

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

```
=================================================================================
Trm| 01;02;03;04;05| 06;07;08;09;10| 11;12;13;14;15| 16;17;18;19;20| 21;22;23;24;25|
=================================================================================
  0| 23;00;14;06;12| 20;00;06;23;29| 10;00;06;14;22| 20;00;26;10;06| 00;26;06;12;22|
  1| 26;23;06;00;23| 10;22;00;22;06| 12;14;00;00;20| 26;20;06;06;10| 29;06;12;00;14|
  2| 00;06;23;23;06| 06;10;22;20;00| 29;12;14;06;00| 00;22;20;14;26| 10;12;00;26;06|
  3| 15;07;00;15;22| 07;15;10;00;20| 00;07;20;12;24| 07;26;12;09;00| 07;00;26;10;23|
  4| 07;15;09;10;00| 00;07;15;12;22| 07;20;12;23;26| 24;07;00;00;20| 26;07;10;15;00|
  5| 20;26;15;09;09| 15;09;09;01;23| 20;10;01;09;12| 22;12;10;26;01| 01;01;15;09;24|
  6| 08;28;16;20;01| 12;06;01;10;08| 16;28;10;10;01| 16;08;08;12;09| 06;20;01;01;08|
  7| 10;08;10;01;16| 09;20;12;06;01| 08;08;28;01;06| 08;10;01;01;16| 08;16;20;28;12|
  8| 06;20;02;12;17| 28;02;08;08;19| 06;19;08;20;09| 10;02;02;08;28| 12;10;17;08;02|
  9| 09;10;20;02;08| 17;08;02;02;12| 28;06;19;02;10| 12;06;19;20;08| 17;08;08;02;28|
 10| 27;02;08;08;10| 08;12;20;09;10| 19;02;02;08;08| 06;27;17;02;12| 20;17;02;06;19|
=================================================================================
```

Figura 7: Exemplo de solução codificada para uma instância do problema. Nesta tabela temos a alocação dos professores para as turmas de Ensino Fundamental

```
=============================================================================================================
Trm| 01;02;03;04;05;06| 07;08;09;10;11;12| 13;14;15;16;17;18| 19;20;21;22;23;24| 25;26;27;28;29;30|
=============================================================================================================
 11| 19;19;03;16;05;04| 16;13;17;03;03;09| 11;05;04;19;19;17| 17;17;13;16;11;05| 16;27;04;05;05;21|
 12| 11;16;09;03;19;19| 03;03;03;05;05;18| 05;25;16;16;03;21| 27;19;16;13;19;18| 11;05;05;18;18;13|
 13| 16;11;19;19;04;05| 13;16;16;27;11;05| 18;18;05;05;16;19| 04;04;21;19;13;11| 05;18;18;04;04;09|
=============================================================================================================
```

Figura 8: Exemplo de solução codificada para uma instância do problema. Nesta tabela temos a alocação dos professores para as turmas de Ensino Médio

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".

2. **Sessão** `materia`: Dados gerais de cada matéria ministrada na escola;

   (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. [6]

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

iv. **Campo** `nivel`: Tipo inteiro, especifica uma escala de nível de dificuldade para a matéria, de 1 (mais fácil) a 3 (mais difícil).

3. **Sessão** `serie`: Dados referentes às séries;

   (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. **Sessão** `professor`: Dados referentes aos professores;

   (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 sessões nomeadas entre colchetes e nomes de campos precedidos por um sinal de igual.

Os campos que seriam informados nesse formato seriam:

1. **Sessão** `timetable`: Sessão única, com o agendamento (timetable) resultante.
   (a) **Campo** `quantidade`: Tipo inteiro, informa qual a quantidade de turmas a serem informadas. Campos a serem repetidos na seqüência para cada turma:
      i. **Campo** `turma`: Tipo inteiro, informa qual a turma em questão.
      ii. **Campo** `slots`: Tipo inteiro, informa qual a quantidade de horários (slots) de aula agendados para a turma. Devem ser informados nas linhas seguintes os códigos das matérias que serão lecionadas (tipo inteiro), um por linha.

6.2 Interface Gráfica

Neste seção, vamos comentar sobre as interfaces de entrada e saída implementadas nesse projeto.

6.2.1 Interface de Entrada

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 9 e 10, que permitem a entrada dos dados referentes aos professores e matérias, respectivamente. Para cada entidade do sistema, como séries, grades, turmas e matérias, existem telas semelhantes para que as propriedades e relacionamentos possam ser informados.

6.2.2 Interface de Saída – Multigrades

Uma vez passados ao núcleo todos os dados que o mesmo necessita para encontrar um agendamento, e o mesmo tendo construido a agenda, 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 a visão geral de todas as turmas da escola. Exemplos de dada uma dessas visões estão respectivamente nas figuras 11, 12 e 13.

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

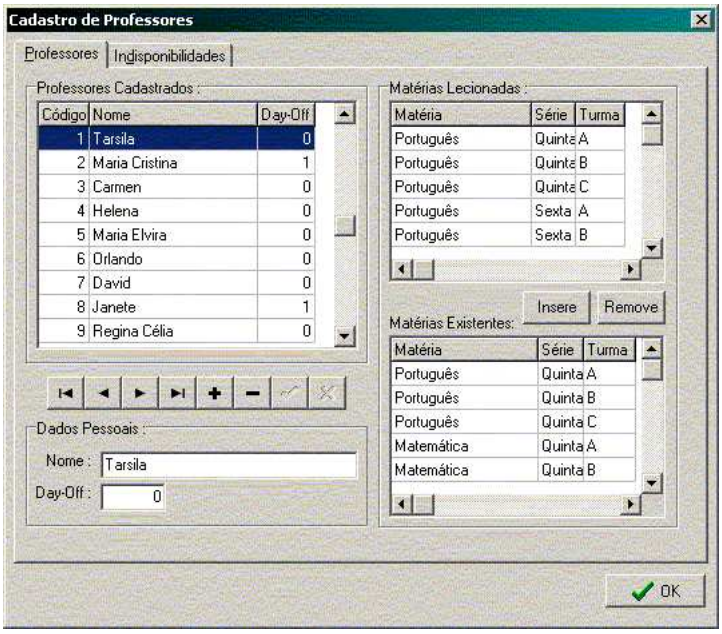

Figura 9: Entrada de dados na interface - Professores

recurso "drag-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 corra o risco do usuário alocar um professor num local em que outro já está lecionando.

Além dessa funcionalidade, outros recursos serão incorporados posteriormente, já que o grupo de estudos do GOA continuará seus trabalhos através de outros projetos de iniciação científica com outros alunos. Uma das funcionalidades que já está sendo trabalhada é a que visa adaptar melhor as telas de saída de resultados com o modelo multigrades permitindo, por exemplo, que o professor veja os horários em que irá ministrar aula de forma independente da grade para o qual está alocado. Em reuniões com os usuários reais do sistema, no colégio Stella Maris, deverão surgir novas necessidades que podem vir a melhorar as funcionalidades do programa.

7 Considerações Finais

Este projeto obteve êxito em suas expectativas ao criar um sistema, com interface gráfica amigável, e provido de um resolvedor heurístico que permite que sejam encontradas boas agendas em tempo viável de processamento, mesmo em computadores pessoais simples.

O problema proposto foi baseado em uma instância real cedida pelo Colégio Stella Maris. Ao final dos trabalhos de implementação, o resolvedor passou por uma fase de exaustivos testes que incluiram uma nova instância: o agendamento das turmas do período noturno da Escola Estadual Barão Geraldo de Rezende, localizada na cidade de Campinas. Algumas

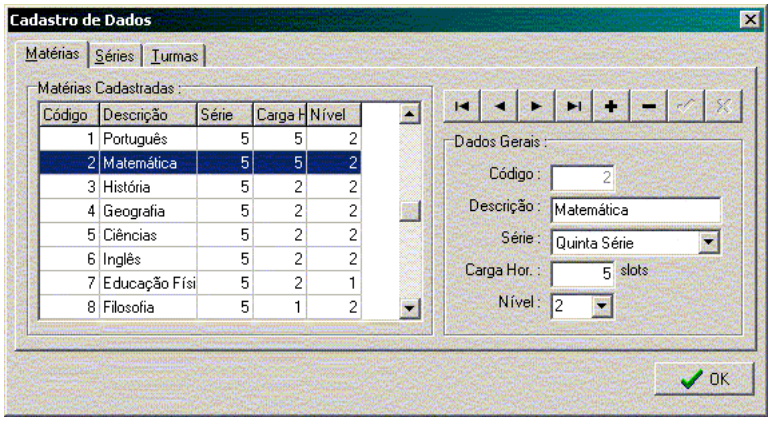

Figura 10: Entrada de dados na interface - Matérias

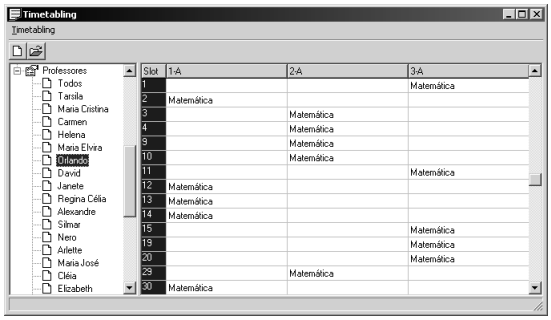

Figura 11: Solução do problema do ponto de vista de um professor

agendas geradas para esta instância foram apresentadas à diretoria desta escola que está analisando a sua aplicação ainda este ano.

Um ponto muito positivo deste trabalho é o enfoque realista que foi dado ao problema, de tal modo que a versão do algoritmo resolvedor apresentada neste documento poderá, incluindo-se algumas melhorias, ser implantada em escolas públicas ou particulares.

Analisando sob o ponto de vista das heurísticas empregadas no resolvedor pôde-se fazer um amplo estudo de diversas meotodologias possíveis, comparando-se os resultados obtidos. Além disso, foi importante focar no estudo aprofundado de uma heurística de busca local escolhida: a Busca Tabu. Deste modo foi possível aplicar métodos avançados [BT93] de busca local como os que foram apresentados na seção 4. Estas técnicas obtiveram resultados satisfatórios no sentido de encontrar boas soluções para o agendamento em um tempo de execução relativamente curto [SIL03].

Trabalhos Futuros

Posteriormente, planeja-se incrementar o sistema de agendamento focando outros aspectos:

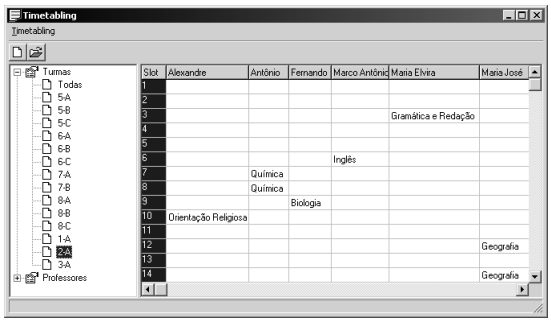

Figura 12: Solução do problema do ponto de vista de uma turma

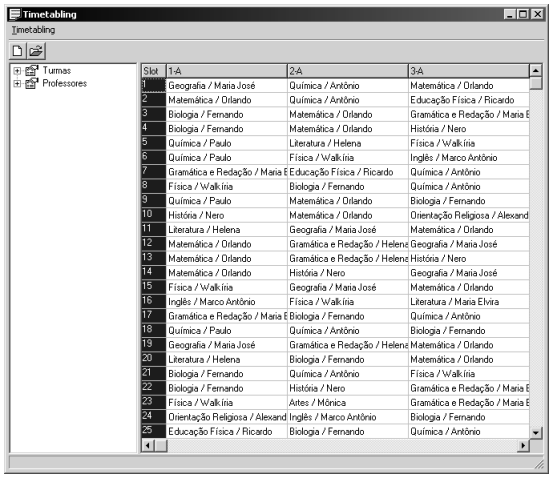

Figura 13: Solução do problema do ponto de vista global

1. Ampliar o conjunto de restrições empregadas no resolvedor de forma a tornar o sistema ainda mais utilizável na prática. Pretende-se, principalmente, adaptar o resolvedor às especificações do agendamento das escolas públicas do Estado de São Paulo. A nossa equipe já entrou em contato com uma escola pública estadual[7] onde já se estuda a possibilidade de uso do sistema por esta escola no agendamento das atividades do ano que vem, que se realizará no final do mês de janeiro.

2. Dar prosseguimento aos estudos do problema de agendamento, ampliando seu escopo. Existem várias vertentes possíveis a serem tratadas: o agendamento interativo [MB02], alocação de professores às respectivas turmas, além do agendamento voltado para universidades (*university timetabling*).

3. Está-se estudando a possibilidade de se construir heurísticas híbridas para melhorar a qualidade das soluções encontradas. Pode-se, por exemplo, criar um algoritmo híbrido usando a Busca Tabu aqui apresentada e Algoritmos Genéticos, desenvolvido para a

[7] A Escola Estadual Barão Geraldo de Rezende, localizada na cidade de Campinas.

solução do agendamento em um projeto paralelo. Assim pode-se aliar as qualidades de cada uma das heurísticas e suprir os seus pontos fracos mediante a hibridização.

4. Por fim, pretende-se, além de realizar a hibridização das heurísticas, implementar um resolvedor que seja capaz usar processamento paralelo, o que pode reduzir drasticamente o tempo de busca por soluções.

Com estes trabalhos, pretende-se ampliar as fronteiras da pesquisa, partindo-se para implementações mais sofisticadas usando heurísticas híbridas e que executam paralelamente, ou mesmo buscando estender as capacidades do resolvedor para usá-lo no agendamente universitário. Além do avanço nas atividades de pesquisa, buscar-se-á a utilização prática dos frutos deste trabalho, através do uso efetivo do sistema por escolas de ensino fundamental e médio.

A 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.

A.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

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 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

A.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.[8]. 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.

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.

[8] 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

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.

A.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

[BT93] Roberto BATTITI and Giampietro TECCHIOLLI. The reactive tabu search. 1993.

[CAN98] Marco CANTÙ. *Dominando o Delphi 4: A Bíblia*. Makron Books, 1998.

[GL97] Fred GLOVER and Manuel LAGUNA. *Tabu Search*. Kluwer Academic Publishers, 1 edition, 1997.

[GLO90] Fred GLOVER. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[GOL01] Hans-Joachim GOLTZ. Combined automatic and interactive timetabling using constraint logic programming. 2001.

[HTW95] Allain HERTZ, Eric TAILLARD, and Dominique De WERRA. A tutotial on tabu search. Technical report, Département de Mathématiques EPFL, 1995.

[MB02] Tomás MULLER and Roman BARTÁK. Interactive timetabling: Concepts, techniques, and practical results. Technical report, Charles University, 2002.

[MS98] Kim MARRIOT and Peter J. STUCKEY. *Programing with Constraints: An Introduction*. The MIT Press, 1998.

[NSA01] Thiago Ferreira NORONHA, Marcelo Mariano SILVA, and Dario José ALOISE. Uma abordagem sobre estratégias metaheurísticas. Technical report, Universidade Federal do Rio Grande do Norte, 2001.

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

[SCH96] Andrea SCHAERF. Tabu search techniques for large high-school timetabling problems. Technical report, Centrum voor Wiskunde en Informatica, 1996.

[SG01] Andrea SCHAERF and Luca Di GASPERO. Local search techiniques for education timetabling problems. Technical report, Università di Udine, 2001.

[SIL03] Rafael Lamare SILVEIRA. Aplicação da busca tabu ao problema de agendamento escolar. Technical report, Relatório de Atividades do Primeiro Semestre - Processo FAPESP número 02/09208-9, 2003.

Índice Remissivo