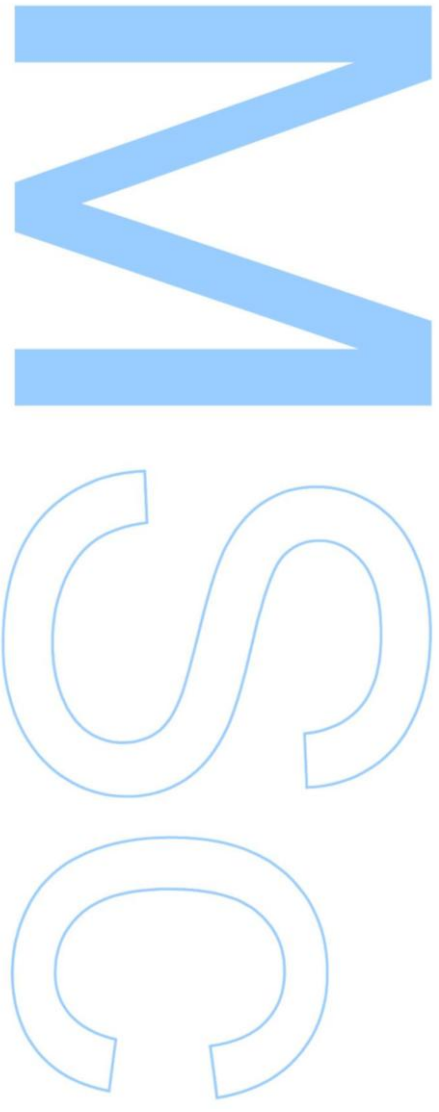
Melhoramentos na pesquisa de salas em software de geração de horários

Filipe David da Costa Martins
Mestrado integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2014

Orientador
Engº Rui Neves, Bullet Solutions – Sistemas de Informação

Coorientador
Prof. João Pedro Pedroso, DCC-FCUP



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Agradecimentos

Quero agradeçer a todas as pessoas que de alguma forma contribuiram de alguma forma para este projeto.

Ao meu orientador, o Professor João Pedro Pedroso pela disponibilidade e apoio durante a realização do estágio.

A todos os elementos da Bullet Solutions pelo bom ambiente e pela boa receção, no tempo que estive no estágio. Um agradecimento em particular ao Eng. Rui Neves, ao Eng. Armando Barbosa, ao Eng. Pedro Fernandes e ao Licínio Mendes pela disponibilidade que sempre demonstraram para me orientar e esclarer dúvidas durante o percuso de estágio.

A todos os meus amigos e colegas que me acompanharam nesta caminhada e a minha namorada Carla.

Por fim, a minha família e em especial aos meus pais e ao meu irmão, por tudo o que fizeram por mim.

Resumo

A geração de horários é um problema que as instituições de ensino superior enfrentam todos os anos.
A empresa Bullet Solutions desenvolveu uma aplicação que permite gerar os horários escolares de modo automático e otimizado.
O aumento da complexidade dos problemas e das combinações possíveis, resultantes de incrementos de turmas (alunos) e de docentes, obriga a que a empresa proceda constantemente a melhorias de performance nos seus algoritmos procurando que os mesmos sejam eficientes tanto quanto possível e que acompanhem as necessidades do cliente. Só assim será exequível a avaliação de um espaço de soluções cada vez maior, aumentando a probabilidade de encontrar a solução ótima.
O algoritmo é constituído por duas fases: a construção de uma solução inicial que respeite todas as restrições e que seja orientada segundo um conjunto de objetivos; e a otimização dessa solução inicial através da realização de trocas procurando melhorar o valor da função objetivo. Por sua vez, a fase de otimização encontra-se subdividida em quatro fases: normal; intensificação; melhorar atribuição de salas; e diversificação.

Este relatório de estágio pretende apresentar o estudo, as abordagens e as conclusões referentes ao algoritmo, denominado ”melhorar a atribuição de salas”. A fase inicial consistiu na compreensão do que estava implementado no algoritmo e na avaliação daquilo que poderia ser melhorado e otimizado. Após esta avaliação, foram aplicadas alterações e correções ao algoritmo pré-existente de forma a torná-lo mais eficiente e otimizado. Por último, avaliou-se a eficiência e eficácia das alterações efetuadas no algoritmo em questão.

O resultado final demonstrou uma melhoria da eficiência do algoritmo em análise, tanto do ponto de vista do tempo de execução como das soluções obtidas.

Abstract

Timetabling schedule creation is a problem that every higher education institutions deal with every year.
The company Bullet Solution developed an application that allows the creation of school timetables in an automatic and optimized way.
The increase in complexity of the problems and possible combinations, resulting from a raise in the number of classes and teachers, obliges the company to make continuous improvements in the algorithms performance so that it becomes as efficient as possible, to keep up with the client needs. Only this way will it be feasible to evaluate a rising solution space, increasing the probability of finding an improved solution.
The algorithm is constituted by two phases: the construction of an initial solution that respects all restrictions and is directed to a set of goals; and the optimization of that initial solution through the realization of changes to improve the value of the objective function. On the other hand, the optimization phase is subdivided in four other phases: normal; intensification; improving the classroom attribution; and diversification.

This internship report intends to present the study, the approaches and the conclusions concerning the part of the algorithm, named as “improve the classroom attribution”. The initial phase consisted in the understanding of what was implemented and in the evaluation of what could be improved and optimized in the pre-existing algorithm. After this evaluation, changes and corrections were applied to make this algorithm more efficient. Finally, the efficiency and efficacy of the effectuated changes in the algorithm were evaluated.

The final result has shown an improvement of the efficiency of the algorithm considered both in the execution time and in the obtained solutions.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A sociedade cada vez mais se preocupa com a gestão do tempo. Diversos meios tecnológicos têm sido utilizados de forma a automatizar processos. Esta automatização permite aumentar a produtividade, tornando assim, o processo mais eficiente, levando a uma redução de custos e a uma melhor distribuição de recursos para outras tarefas. Num mundo em que cada vez mais a palavra eficiência está presente, esta automatização tem um papel preponderante na agilização de muitos processos.

Na área do ensino, as Instituições de Ensino Superior (IES) deparam-se com o problema da criação de horários para as aulas antes de começar um novo ano académico. Os recursos humanos destas instituições ficam ocupados na tentativa de encontrar manualmente horários que satisfaçam todos os intervenientes neste problema.

O problema da geração automática de horários para IES é algo que tem atraído a comunidade científica desde 1960 na busca de novas soluções. Este problema é definido como a atribuição de um conjunto de aulas, que envolve professores e estudantes, a um conjunto de salas e de intervalos de tempo, satisfazendo um conjunto de restrições [1].

Este tipo de problema é normalmente associado à área de investigação operacional, mas também tem sido abordado nos últimos anos pela área da inteligência artificial com alguns algoritmos, como é o caso da pesquisa tabu, do *simulated annealing*, dos algoritmos genéticos e da satisfação de restrições [1].

A Bullet Solutions é uma empresa com larga experiência geração de horários. A empresa possui uma aplicação de geração de horários escolares para as IES, que se chama Bullet TimeTabler Education (BTTE), líder no mercado nacional, estando a funcionar nas dez maiores IES de Portugal.

A partir do mercado nacional conquistado, a internacionalização tornou-se um passo natural a seguir. Para tal tornou-se necessário tornar ainda mais eficiente o algoritmo da

geração de horários escolares, pois, quando se realiza a geração de horários para um número elevado de turmas/professores, o algoritmo atual poderá demorar algumas horas até se encontrar uma solução de qualidade para a instância em questão.

O algoritmo atual do BTTE, tem duas fases, a cada uma correspondendo um algoritmo próprio. A primeira fase é a construção de uma solução, chamada de solução inicial. A segunda fase é a de otimização da solução inicial, procurando melhorar diversos aspectos da solução que é encontrada na primeira fase.

Neste algoritmo não sendo possível uma abordagem ao problema completo, devido ao tamanho e complexidade, selecionou-se uma das fases de otimização, o algoritmo ”melhorar a atribuição de salas” (MAS), uma parte crítica do desempenho do software. Este algoritmo encontra-se relativamente contido e pode ser isolado do restante cálculo para efeitos de teste e propostas de melhoria, permitindo assim, uma análise mais aprofundada dos resultados.

Atualmente, a componente “melhorar a atribuição de salas” tem algoritmos próprios totalmente distintos das abordagens utilizadas nas restantes fases, uma vez que, o que se procura atingir são otimizações nas atribuições dos recursos físicos sem deslocar os eventos do espaço temporal a que foram atribuídas originalmente. A empresa, num dos estudos recentes que efetuou, identificou esta componente do algoritmo como sendo um dos pontos críticos, pois à medida que a dimensão das instâncias aumentava, o tempo de cálculo necessário para apresentar bons resultados tornava-se incompatível com a necessidade de apresentar soluções em tempo útil para os clientes.

1.1 Enquadramento da Empresa

A Bullet Solutions S.A. foi criada em 2006, estando localizada no Porto. A empresa presta serviços de consultoria e desenvolvimento de soluções informáticas de otimização para problemas combinatórios à medida [2].
A Bullet dispõem de soluções para diferentes áreas, como é o caso do ensino, da indústria e da saúde. Estas soluções vêm substituir processos que eram efetuados manualmente, e que se tornavam longos e complexos, resultando numa eficiência e produtividade reduzida. Uma das primeiras aplicações desenvolvidas pela empresa foi o Bullet TimeTabler Education, que se tornou a sua imagem de marca. Esta aplicação foi criada com base nas informações disponibilizadas por diversas IES (tal como, professores, salas, cursos, turmas e disciplinas). Desta forma, tornou-se possível criar um algoritmo que respeita um conjunto de restrições específicas (como por exemplo, um professor não poder dar mais de um número determinado de horas de aulas por dia) de maneira a minimizar a penalização dos objetivos propostos pelo cliente. Assim, obtém-se uma possível de solução de horário.

1.2 Objetivos

O objetivo do presente trabalho passa por melhorar a eficiência e eficácia do algoritmo MAS, que se encontra atualmente implementado no produto BTTE, documentando devidamente o algoritmo e todas as alterações efetuadas. As melhorias pretendidas para o algoritmo focam-se essencialmente no tempo de execução e no resultado da solução obtida.

Para atingir estes objetivos, definiram-se as seguintes fases para o projeto:

1. Analisar detalhadamente a implementação existente do algoritmo MAS;
2. Medir os tempos de execução e eficácia para diferentes problemas reais;
3. Analisar os pontos críticos existentes no algoritmo atual;
4. Conceber e implementar alterações ao algoritmo, originando assim um novo algoritmo;
5. Medir os tempos de execução e eficácia do novo algoritmo nas mesmas condições do ponto 2.

1.3 Estrutura do relatório

O presente relatório é constituído por mais cinco capítulos, para além deste.

O capítulo 2 que apresenta os principais conceitos do modelo de dados implementado no BTTE e tecnologias utilizadas neste trabalho.

O capítulo 3 descreve o algoritmo MAS implementado no BTTE e é realizada a análise de performance.

O capítulo 4 descreve as alterações e correções implementadas no algoritmo MAS.

O capítulo 5 apresenta os resultados conseguidos com o novo algoritmo MAS.

O capítulo 6 conclui o trabalho realizado e análise de possível trabalho futuro.

Capítulo 2

Preliminares

2.1 Conceitos principais

Esta secção apresenta alguns conceitos que se deve ter em conta para uma melhor compreensão do modelo usado nos algoritmos do BTTE.

Curso: pode ser constituído por diferentes planos curriculares.

Plano curricular: tem um conjunto de unidades curriculares e um conjunto de turmas.

Unidade curricular: designada também por disciplina ou cadeira, refere-se a cada uma das matérias lecionadas, encontram-se associadas a um plano curricular com objetivos de formação própria, tornando-se objeto de avaliação que se traduz numa classificação final.

Docente: leciona as unidades curriculares que a instituição de ensino tem para oferecer. Um docente também é designado como professor.

Turma: entende-se como o grupo de alunos do mesmo ano e curso que partilham exatamente o mesmo horário escolar. Pode também ser considerado como o ”horário de turma”, ou ainda, o ”perfil de horário” de um determinado ano e curso.

Aula: forma como a unidade curricular é lecionada, por exemplo, se a aula é teórica, teórico-prática ou prática. Uma aula pode ter um ou vários turnos e ocorrer num determinado período de tempo (semanas).

Turno: representa a quantidade de vezes que uma aula é repetida para diferentes turmas. Por exemplo, para as Turmas designadas por A, B e C que têm a unidade curricular de Matemática no seu horário. A unidade curricular de Matemática tem uma aula teórica, onde as três turmas se reúnem num único turno dessa aula, no entanto, as aulas práticas estão divididas em três turnos, onde um turno correspondente a cada

uma das turmas em questão.
Para este exemplo, podem se ter em conta dois pontos de vista o da turma e o da instituição, do ponto de vista da turma, esta só tem uma aula teórica e uma prática. Enquanto, do ponto de vista da instituição tem uma aula teórica e três aulas práticas.

Sala: espaço onde são lecionadas as aulas. As salas podem estar localizadas em diferentes edifícios e área geográficas.

Evento: constituído por um turno, uma ou mais salas e por um ou mais docentes. Um evento poderá ser alocado numa posição do horário, tendo em conta, uma duração específica e ocorrendo num conjunto de semanas, respeitando todas as restrições existentes.

Restrições *(hard constraints)***:** proibições, regras ou indisponibilidades de uma entidade que serão respeitadas. De seguida são apresentadas as restrições presentes no BTTE [3].

- para cada docente, turma ou sala só pode haver um evento atribuído em cada momento;
- cada sala só pode receber um número de alunos que respeite a sua capacidade;
- há alguns eventos que têm de ser contíguos;
- há alguns eventos que não podem ser lecionados no mesmo dia;
- há alguns eventos que têm de ser sobrepostos;
- há alguns eventos que têm de ser separados (podendo, ocorrer no mesmo dia);
- indisponibilidade da instituição, salas, turmas, unidades curriculares, docentes e turnos, ou seja, espaços temporais onde não podem ser atribuídos eventos.
- limites de horas diárias de eventos que podem ser atribuídas a docentes e turmas;
- limites de horas consecutivas de eventos que podem ser atribuídas a docentes e turmas;
- dentro de uma unidade curricular, têm de se respeitar a ordem de tipologias de eventos que ocorrem na semana (por exemplo, os eventos práticos só ocorrem depois dos teóricos);
- intervalo mínimo necessário entre eventos com início em dias diferentes, para docentes e turmas (para garantir que uma aluno ou docente que tenha aulas até tarde, não tem aulas muito cedo no dia a seguir);

- tempo mínimo entre eventos que implicam a deslocação entre edifícios para docentes ou turmas;
- número mínimo de dias livres para determinados docentes.

Objetivos *(soft constraints)***:** parâmetros que medem a qualidade de uma solução. Os objetivos podem ser entendidos como preferências de horário que se tentam cumprir, no entanto, estas podem não ser satisfeitas. A função objetivo resulta da penalização do incumprimento de objetivos e do relacionamento entre os objetivos. Os objetivos presentes no BTTE dizem respeito ao incumprimento das restrições que se encontram abaixo descritos [3].

- há eventos em dias diferentes que devem começar à mesma hora;
- há eventos que devem ser lecionados na mesma sala;
- há eventos que devem ser contíguos;
- há eventos que devem ser sobrepostos;
- em cada período do dia (manhã, tarde, pós-laboral), procurar colocar eventos de tipologias diferentes, nos horários das turmas;
- conseguir períodos livres (sem eventos atribuídos) nos horários das turmas e docentes;
- procurar garantir um mínimo de horas para um período do dia para turmas e docentes (de modo a evitar que estes se desloquem por pouco tempo);
- respeitar as preferências dos eventos para determinados períodos (eventos preferencialmente atribuídos a um período do dia);
- evitar furos nos horários de turmas, docentes e salas;
- evitar trocas de salas nos horários de turmas e docentes;
- evitar as alterações logísticas das salas (tentar que os eventos que forem atribuídos a uma sala tenham necessidades de equipamento o mais semelhante possível);
- utilização das salas mais indicadas para cada evento;
- respeitar as preferências da instituição, salas, turmas, unidades curriculares, docentes e turnos, ou seja, espaços temporais onde preferencialmente não devem ser atribuídos eventos;
- dentro de unidade curricular, devem ser respeitadas a ordem de tipologias de eventos que ocorrência na semana (por exemplo, os eventos práticos só ocorrem depois do teóricos);

- conseguir um número mínimo de dias livres para determinados docentes;
- docentes que devem lecionar eventos nos mesmos dias (docentes com horários parecidos).

O processo de geração de horários não é feito para um único aluno, mas sim, para um conjunto de alunos, ou seja, uma ou mais turmas. A atribuição de alunos às turmas ocorre num processo separado.
Alguns recursos como a instituição, turmas, professores, salas e unidades curriculares possuem horários que informam sobre as disponibilidades dos intervenientes. Os horários ajudam a perceber a que horas e a que dias da semana é possível alocar um determinado recurso. A disponibilidade dos horários está dividida de quatro formas: preferencial, não preferencial, a evitar e indisponível, com as cores verde, amarelo, cor-de-laranja e vermelho, respetivamente. A figura 2.1 representa a disponibilidade para alocação de um recurso.

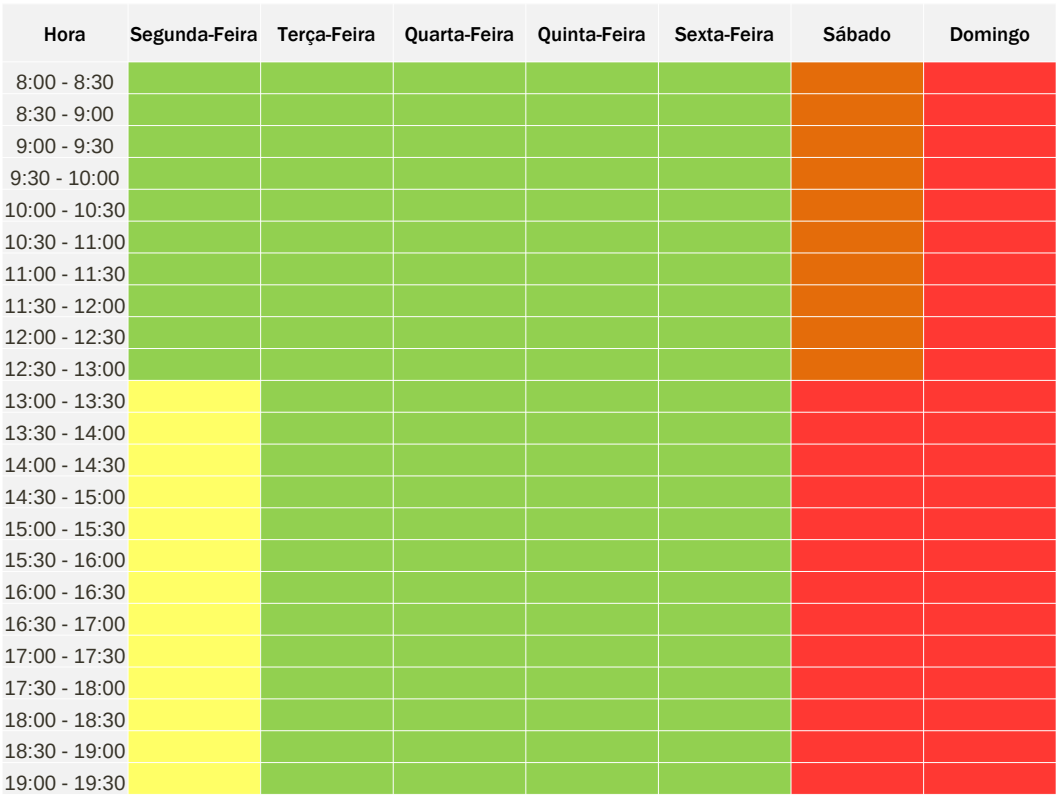


Figura 2.1: Disponibilidade dos horários

Os horários estão divididos em intervalos de tempo, a que cada intervalo é atribuído uma *slot* de tamanho fixo, que é definida pelo utilizador da aplicação. O tamanho de uma *slot*, no caso de ser trinta minutos, o horário terá de ser dividido em intervalos de tempo de trinta

minutos. Por exemplo, para se alocar um evento que tenha noventa minutos de duração, vai ocupar três *slots* no horário.
Os horários podem repetir-se por várias semanas, ou seja, têm como base um único horário para uma semana, que se repete em várias semanas. Ao aplicar alguma alteração num horário, realizam-se alterações em todas as semanas.

2.2 Algoritmo BTTE

O algoritmo do BTTE passa por duas fases distintas. A primeira fase entende-se como sendo a construção de uma solução inicial válida que respeite todas restrições. Uma solução válida é aquela que consegue alocar todos os eventos a serem lecionados. A segunda fase é a de otimização da solução inicial, composta por várias fases como mostra a figura 2.2.

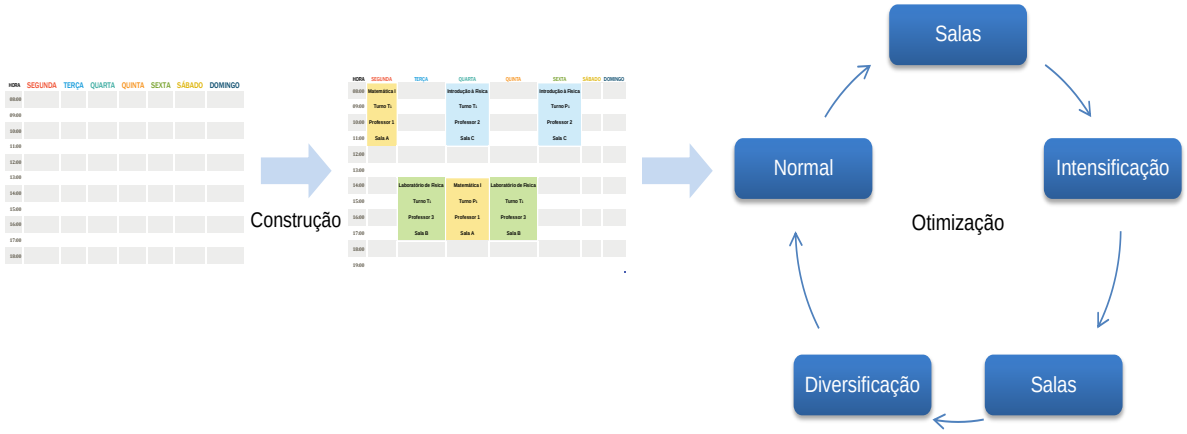


Figura 2.2: Fases constituintes do algoritmo do BTTE

2.2.1 Construção de uma Solução inicial

A criação de uma solução inicial é feita a partir de um horário vazio. Na construção da solução existe uma lista com todos os eventos que ainda não foram colocados no horário. No entanto, nem todos esse eventos serão colocados. Alguns desses eventos são eventos ”fantasmas”, criados para permitir uma maior flexibilidade ao algoritmo no seu espaço de pesquisa. Estes eventos, são combinações de recursos que deixaram de ser possíveis. Um exemplo de como os eventos se podem tornar fantasmas, é o caso de uma aula teórica com os turnos A e B (T_A e T_B) que possui dois professores possíveis (P_1 e P_2) para lecionar os turnos, cada um leciona um. Na lista de eventos possíveis de alocar estariam presentes

os eventos com as combinações P_1T_A, P_1T_B, P_2T_A e P_2T_B. Ao atribuir um dos eventos com a combinação P_1T_A, os eventos P_2T_A e P_1T_B tornam-se em eventos fantasma não sendo colocados no horário. O evento com a combinação P_2T_B é colocado por fim.
A escolha do evento a atribuir no horário é feita com base na urgência de colocação dos eventos, ordenando-os do mais urgente para o menos urgente, ou seja, dos eventos com menor possibilidade de colocação para os que têm maior possibilidade de colocação (por exemplo, a disponibilidade de todos os recursos envolvidos para cada evento). Com o evento mais urgente selecionado é necessário colocá-lo, para isso, é importante avaliar todos os objetivos e escolher o melhor local para inseri-lo, procurando assim, que essas escolhas não levem a uma limitação dos eventos que ainda restam colocar. No caso de não se conseguir colocar um evento, seleciona-se um que tenha recursos em comum, como professores, turmas ou salas, em que se vai trocando de recursos entre eventos até se conseguir colocar todos os eventos até agora inseridos, e atribuir o evento que não o tenha sido.
Após a colocação do evento, todos os mapas de dados e a lista de eventos são atualizados, sendo os critérios de urgência recalculados. Então, é escolhido o novo evento mais urgente. Isto ocorre repetidamente até todos os eventos estarem inseridos, respeitando sempre as restrições envolvidas. Quando todos os eventos tiverem sido colocados chega-se à solução inicial [3].

2.2.2 Otimização da Solução inicial

A fase de otimização inicia-se após a construção de uma solução inicial. Esta fase procura soluções de forma a melhorar a qualidade da solução, passando por quatro fases, nomeadamente, normal, intensificação, diversificação e melhorar salas. As três primeiras fases pesquisam novas soluções fazendo pequenas alterações na solução atual. Estas alterações são avaliadas podendo ser aceites ou não. As alterações são feitas com base em dois tipos de estrutura de vizinhança: as trocas diretas entre eventos e as trocas de eventos em dois passos.
No primeiro caso, é escolhido um evento pivô e são selecionados os potenciais locais de destino. Os locais podem estar livres ou ocupados por eventos. Os de destino que se encontram livres são vizinhos válidos. Os que estão ocupados, por um ou mais eventos, serão trocados diretamente do local de destino para o local de origem do evento do pivô, criando assim, um vizinho válido no caso da troca ser considerada possível. O segundo caso, é semelhante com o da primeira estrutura, mas tem a diferença de quando o local de destino está ocupado é feita uma avaliação mais completa às sobreposições existentes. Para os eventos com sobreposição de recursos são procurados locais de colocação alternativos para além do local original do evento pivô. A cada iteração, é aceite o vizinho com menor valor da função objetivo.

A fase normal, intensificação e diversificação têm papéis diferentes na procura de soluções. A fase normal, é uma pesquisa mais rápida, onde número de vizinhos criados a cada iteração é limitado, quer pelo número de eventos pivô usados quer pelo número reduzido de potenciais locais de destino a ser avaliados. Na fase de intensificação, é feita uma pesquisa com maior detalhe, retornando soluções de melhor qualidade que na fase normal, pois este tem um maior número de eventos pivôs a serem usados e avalia um conjunto de potenciais locais de destino para cada um dos pivôs. Na fase de diversificação, a função objetivo é adulterada temporariamente, durante um determinado número de iterações, isto permite evitar mínimos locais para tentar alternar para outra zona de pesquisa de soluções, tentando assim, encontrar melhores soluções.
A fase de melhorar salas têm como objetivo principal a otimização dos espaços atribuídos. Este algoritmo faz uma redistribuição das salas de aula, não ocorrendo alteração dos eventos na posição do horário, sendo utilizados apenas os objetivos relacionados com as salas. A fase de melhorar salas é realizada entre a fase normal e a fase de intensificação, e também entre, a fase de intensificação e a fase de diversificação. A fase normal e a de intensificação depois de um certo número de iterações sem melhorar nada, avançam para a fase de melhorar salas.
O processo de otimização pode acabar a pedido do utilizador, ou no caso de ser encontrada uma solução ótima, isto é, a penalização dos objetivos ser igual a zero [3].

2.3 Tecnologias

O projeto foi desenvolvido com as ferramentas e tecnologias que serão abordadas nesta secção.

2.3.1 Microsoft Visual Studio

Microsoft Visual Studio é um IDE desenvolvido pela Microsoft, que tem se tornado uma ferramenta completa, com o lançamento de versões e atualizações. Microsoft Visual Studio permite desenvolvimento de diferentes tipo de aplicações, como Windows Forms (aplicações para Microsoft Windows), aplicações Web e Windows Phone. Suporta várias linguagens de programação nativamente como C, C++, C# e Visual Basic [4].

2.3.2 C#

O C# (C-Sharp) é uma linguagem de programação simples, moderna, fortemente tipada e orientada a objetos. A linguagem C# foi desenvolvida pela Microsoft e aprovada como

padrão pela ECMA e ISO. Esta linguagem tem bases na linguagem C e será imediatamente familiar para os programadores de C, C + + , Java. O C-Sharp fornece suporte de princípios de engenharia de software, como verificação de tipagem forte, verificação dos limites dos *arrays*, deteção das tentativas de utilização de variáveis não inicializadas e recolha de lixo automática [5].

2.3.2.1 Stopwatch

As medições de performance realizadas no algoritmo a nível do tempo de execução foram realizadas com o auxílio da classe *Stopwatch*, que é disponibilizada pelo *.Net*.

A classe *Stopwatch* tem como função medir o tempo decorrido num intervalo, ou em vários. O contador utilizado pela classe *Stopwatch* depende do *hardware* instalado e do sistema operativo. Se estes suportarem *high-resolution performance counter*, então o *Stopwatch* utiliza esse contador (que tem um maior precisão), senão, a classe *Stopwatch* usa o contador do sistema para medir o tempo decorrido [6].

2.3.2.2 Sort

As ordenações realizadas no algoritmo foram aplicadas através do método de ordenação *Sort*. Este método faz o uso de três algoritmos de ordenação dependo [7]: se a partição da lista (parte de uma lista em avaliação), contém um número de elementos inferior a 16, é utilizado o algoritmo Insertion sort [8]; no caso do número de partições ser maior que 2 * log N (em que *N*, representa o número de elementos da lista que recebe), é utilizado o Heapsort [9]; e, não ocorrendo nenhuma das situações anteriores, este utiliza o Quicksort [10].

Capítulo 3

Algoritmo melhorar a atribuição de salas

A informação dentro das empresas é bastante importante para que não haja perda dos conhecimentos que estas adquirem ao longo dos anos. Nas empresas ligadas à programação, a documentação tem um papel importante para que qualquer programador perceba o código que está implementado em pouco tempo, e possa evoluir de forma a melhorar o que está feito.
Este capítulo tenta ajudar a perceber como o algoritmo MAS foi construído, uma vez que, por falta de documentação é uma tarefa difícil saber o que foi implementado. Este algoritmo vai ser apresentado desde como é integrado, iterado e como é feita a pesquisa das salas para os eventos.

3.1 Integração do algoritmo MAS

O algoritmo MAS não é implementado individualmente, encontra-se inserido entre fases de otimização. A integração e a iteração da fase a iterar encontra-se representado na figura 3.1.

O algoritmo ao finalizar a construção da solução inicial, inicia a fase de otimização, onde são exploradas as novas soluções viáveis, respeitando as restrições envolvidas.
Na fase de otimização existem dois momentos em que pára de iterar, que são representados pelos passos 3 e 5. No passo 3, se o valor dos objetivos for igual a zero pára de otimizar, pois a solução é ótima. No passo 5, a pedido do utilizador, que pode querer ver como se encontra a solução, o mesmo acontece. Esta iteração é o que vai ajudar a avançar no número de iterações necessárias em cada uma das fases. O passo 6 possui duas tarefas, a de mudar de fase de otimização e, a de iterar a fase de otimização a

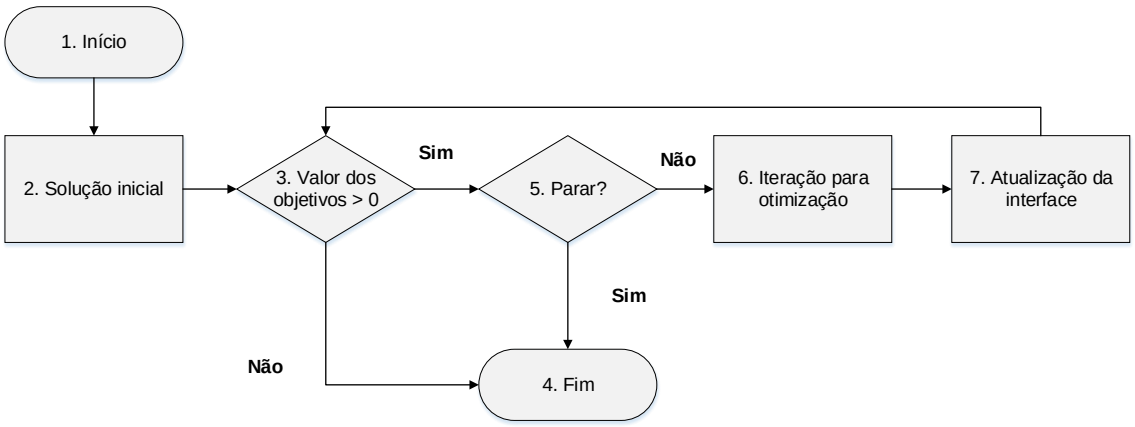


Figura 3.1: Representa o algoritmo que efetua a mudança de fase e a iteração de cada uma das fases de otimização.

ocorrer no momento. A primeira tarefa é a mudança de fase, onde se atribui a nova fase de otimização. Quando se inicia uma nova fase de otimização são carregados os dados que são precisos para esta nova fase da otimização decorrer. A segunda, tem a tarefa de otimizar os eventos alocados e avançar nestes.
A cada otimização é verificado, se é realizada ou não, a atualização da interface para o utilizador com a informação da evolução da solução, com base nos objetivos pretendidos. A atualização ocorre no caso de se conseguir obter soluções melhores ou iguais à melhor solução que se obteve até ao momento.

3.2 Executar melhorar salas

O algoritmo quando entra na fase de melhorar salas, vai reunir a informação necessária, como o valor da função objetivo das salas, a lista dos horários das turmas, a lista dos horários dos professores que serão ordenadas de forma aleatória, e, o número total de iterações para percorrer todos os horários das turmas e dos professores que é calculado da seguinte forma:

$$total = (t + d) \times p \times d$$

Onde *t* é o número de horários das turmas, o *d* representa o número de horários dos professores, *p* representa número de períodos do dia e, *d* o número de dias da semana.
O número de períodos do dia (podendo ser: manhã, tarde) e o número de dias da semana fazem parte do número total de iterações, uma vez que, se iteram todos os dias e todos os períodos destes nos horários.

A otimização é feita inicialmente nos horários das turmas e só depois nos horários dos professores. O horário como ponto de partida para otimização das salas encontra-se na primeira posição da lista das turmas, sendo selecionado o primeiro dia da semana do horário e, o primeiro período do dia.

Depois de iniciada a fase de melhorar as salas, a cada iteração, o algoritmo 1 vai ser chamado. Este algoritmo começa por avaliar, se continua ou não, na fase de melhorar as salas. No caso de não continuar, avança para a próxima fase de otimização (intensificação ou diversificação), isto acontece em três casos. Sendo o primeiro relativo ao tempo limite que é atingido, definido na aplicação. O segundo caso ao valor da função objetivo das salas ser igual a zero. Finalmente, o terceiro caso se percorreu todos os horários das turmas e professores da parte da manhã e da tarde de todos os dias da semana.

Algorithm 1: Algoritmo que explora as melhores salas e guarda a melhor solução.

```
1  if ContinuarFase() = false then
2      MudarFase ()
3  if AvaliaTurma() || AvaliaProfessor() then
4      aceitaSolução ← false
5      if QuemVaiMelhorar() = Turma then
6          aceitaSolução ← IteraTurma ()
7      else
8          aceitaSolução ← IteraProfessor ()
9      if aceitaSolução = true then
10         ValorFunçãoObjetivoDaCorrida () ← FunçãoObjetivoDaCorrida ()
11         GuardaSolucaoActualComoMelhorDaCorrida ()
12         if ValorFunçãoObjetivoDaCorrida() < MelhorValorFunçãoObjetivo() then
13             GuardaSolucaoActualComoMelhor ()
14             MelhorValorFunçãoObjetivo () ← ValorFunçãoObjetivoDaCorrida ()
15         ValorFunçãoObjetivo () ← ValorFunçãoObjetivoDaCorrida ()
16 else
17     Trocar(QuemVaiMelhorar())
```

Se o algoritmo continuar na fase de melhorar as salas é feita a validação de quem vai melhorar horário através do `AvaliaTurma` e do `AvaliaProfessor`. A função `AvaliaTurma` indica se os horários a melhorar são os das turmas e, se ainda não percorreu todos os horários das mesmas. A função `AvaliaProfessor` verifica se os horários a avaliar são os dos professores e, se não percorreu todos os horários destes. Após ter conhecimento sobre quem vai recair a otimização do horário é feita uma iteração para um período de um

dia no horário da turma através da função `IteraTurma` ou do professor através da função `IteraProfessor`, dependendo do horário que se está a melhorar, devolvendo se encontrou uma melhor solução de salas para o horário da turma ou do professor. Estas duas funções funcionam de forma muito parecida, sendo estas abordado na secção 3.2.1.
Ao longo da execução do algortimo são guardadas duas soluções, uma que é a melhor solução da corrida, e outra que é, a melhor solução atingida desde o início até ao ponto em que se encontra, pois isto, quando se passa na fase de diversificação pode-se obter soluções piores para depois conseguir melhores soluções.
O horário selecionado no caso de ser melhorado é realizada a avaliação dos pesos dos objetivos da corrida e guardadas as melhores soluções da corrida. O valor da função objetivo da corrida se é melhor que a solução medida até agora, guarda a solução com o melhor resultado, sendo posteriormente, atualizados os valores dos pesos dos objetivos. Uma das listas guardadas é a melhor solução da corrida para ter um ponto de partida sempre que se muda de fase.
Por fim, ao se avaliar todos os horários das turmas, passa-se a avaliar os horários dos professores através da função `Troca`.

3.2.1 Iteração no horários das turmas e dos professores

A forma de como é feita a iteração nos horários das turmas é representada pelo algoritmo 2, também podendo ser aplicado da mesma forma quando se está iterar os horários dos professores. O algoritmo MAS, após identificar sobre quem vai melhorar o horário, segue para o próximo passo, que será decidir se vai avaliar o período do dia no horário selecionado. Esta decisão é feita com base num valor de aceitação entre zero e um, definido pela empresa, que vai ser comparado com um valor aleatório. Se o valor aleatório for menor ou igual ao valor de aceitação, é feita a pesquisa de eventos através da função `Horário`. A função `Horário`, procura os eventos que ocorrem no horário de uma turma num dia da semana, para um determinado período desse dia.

No período do dia selecionado, os eventos serão ordenados pela hora de início.
Para os eventos selecionados é feita a pesquisa de melhores soluções, sendo realizada pela função `OtimizarSalas`, que devolve os eventos com as melhores salas encontradas, descrito na secção 3.3. A solução encontrada é analisada pela função `AceitaNovasSalas`. A função `AceitaNovasSalas` verifica se a solução não é igual à solução anterior, se esta for igual à anterior não é aplicada no horário, mas se a solução for diferente à anterior, é então aplicada no horário e, posteriormente, esta troca é avaliada quanto ao peso de todos os objetivos das salas. Este peso é comparado com o valor anterior do peso dos objetivos das salas, se este valor for maior ou igual, atualiza este peso, senão, reverte a solução.
Durante este processo, ao se verificar uma das seguintes condições é feita à atualização dos dados a iterar, como por exemplo: se o valor aleatório for maior do que o valor de

Algorithm 2: Iteração no horários das turmas.

output: aceitaSolução

```
1  aceitaSolução ← false
2  if ValorAleatório() <= AceitaDiaComProbilidade() then
3      turma ← ObterTurma ()
4      dia ← ObterDia ()
5      períodoDoDia ← ObterPeríodoDoDia ()
6      eventos ← Horário(turma, dia, períodoDoDia)
7      if eventos > 0 then
8          Ordena(eventos)
9          melhorListaEventos ← OtimizarSalas(eventos)
10         if AceitaNovasSalas(melhorListaEventos) then
11             aceitaSolução ← true
12 AtualizarDadosIterar ()
```

aceitação; da não existência de eventos a serem avaliados nesse intervalo de tempo; ou ainda, da finalização da otimização das salas para a lista de eventos. Esta atualização consiste no avanço do período do dia, no caso de se estar a avaliar os eventos da parte de manhã, avança-se para a parte da tarde. Continuamente, se o período do dia a avaliar for o da tarde, passa-se para o dia seguinte, este processo repete-se até ao final da semana. Ao percorrer o horário todo de uma turma ou de um professor, segue-se a próxima turma até percorrer todas as turmas, e depois, todos os professores. Nesta atualização é ainda incrementada uma iteração ao contador, para saber quantas iterações ainda faltam iterar. No fim, retorna se a solução que foi melhorada ou não.

3.3 Otimização das salas dos eventos

Esta secção descreve como é realizada a pesquisa e a validação de novas possíveis soluções. As novas soluções são obtidas através de pequenas alterações, relativamente, à solução anterior. Estas respeitam as restrições e tentam encontrar as melhores soluções do momento na sua vizinhança, com base nos objetivos. A figura 3.2 apresenta como isto é realizado.

A otimização das salas dos eventos recebe uma lista de eventos que se encontra situada num horário de uma turma ou de um professor, num período de um dia da semana. Esta lista de eventos é verificada se foi avaliada, se sim, retorna a lista de eventos com a melhor

solução de salas, este passo é utilizado quando a função de `OtimizarSalas` é chamada pelo passo 6.

No caso da lista de eventos ainda não ter sido avaliada, o próximo passo é o 4, onde é efetuada uma pesquisa de salas que sejam comuns a todos os eventos. Esta pesquisa realiza a procura de novas soluções vizinhas de forma evitar que as turmas ou os professores tenham que trocar de salas quando uma aula termina, abordado na secção 3.3.1. No passo 6 é feita a combinação de eventos da lista de eventos, como será apresentado em 3.3.2, tentando procurar melhores salas para cada uma dessas combinações de eventos, onde cada uma destas combinações chama recursivamente, este algoritmo de otimização, representado na figura 3.2. Quando uma lista de eventos tem tamanho igual a um, deixa de criar combinações de eventos.
No passo 7, após a pesquisa de novas possíveis salas para os eventos são guardadas as listas de eventos com as novas possíveis salas numa lista, presentes no passo 4 e no passo 6, no caso de se possuir mais que dois eventos.

O conjunto das listas de eventos com as novas possíveis salas vai ser iterada. A cada iteração para cada lista de eventos é criado um operador, que contém os objetivos e as restrições de cada uma destas listas de eventos com as novas possíveis salas. Neste operador também se guarda o estado atual dessa lista de eventos, constituído com os objetivos e as restrições para reverter a solução. A seguir à criação do operador, a lista de eventos é avaliada, verificando se respeita as restrições. A lista de eventos se não for válida passa à próxima lista com as novas salas.
No passo 11, é realizada a avaliação da lista de eventos atual, de forma a saber qual é o peso dos seus objetivos das salas para esta alocação. Ao se saber qual é o peso da solução antiga, falta saber qual é o peso da nova solução, para assim, se poder comparar as soluções. A lista de eventos com as novas possíveis salas, é então aplicada à solução com o auxílio do operador. Esta lista com as novas salas é avaliada quanto ao seu peso.
Depois da avaliação das salas anteriores e das novas da lista de eventos, a solução é revertida para a lista de eventos original. Esta reversão, é realizada para se comparar a lista de eventos original com as listas com as novas salas possíveis que ainda faltam avaliar.
No passo 14, guarda-se numa lista, a lista de eventos com o ganho associado. Este ganho é calculado através da subtração do peso da lista de eventos original, com o peso da lista de eventos com as novas salas. Seguidamente, avança-se para a próxima lista de eventos com as novas possíveis salas, até que todas as listas de eventos com as novas possíveis salas sejam avaliadas.
Após avaliadas, todas as listas com as novas possíveis salas são ordenadas as listas de eventos com os ganhos associados, em ordem crescente de grandeza, sendo guardado na lista de eventos avaliados à lista de eventos com o menor peso. Por fim, retorna a lista de eventos com as novas possíveis salas com o menor peso.

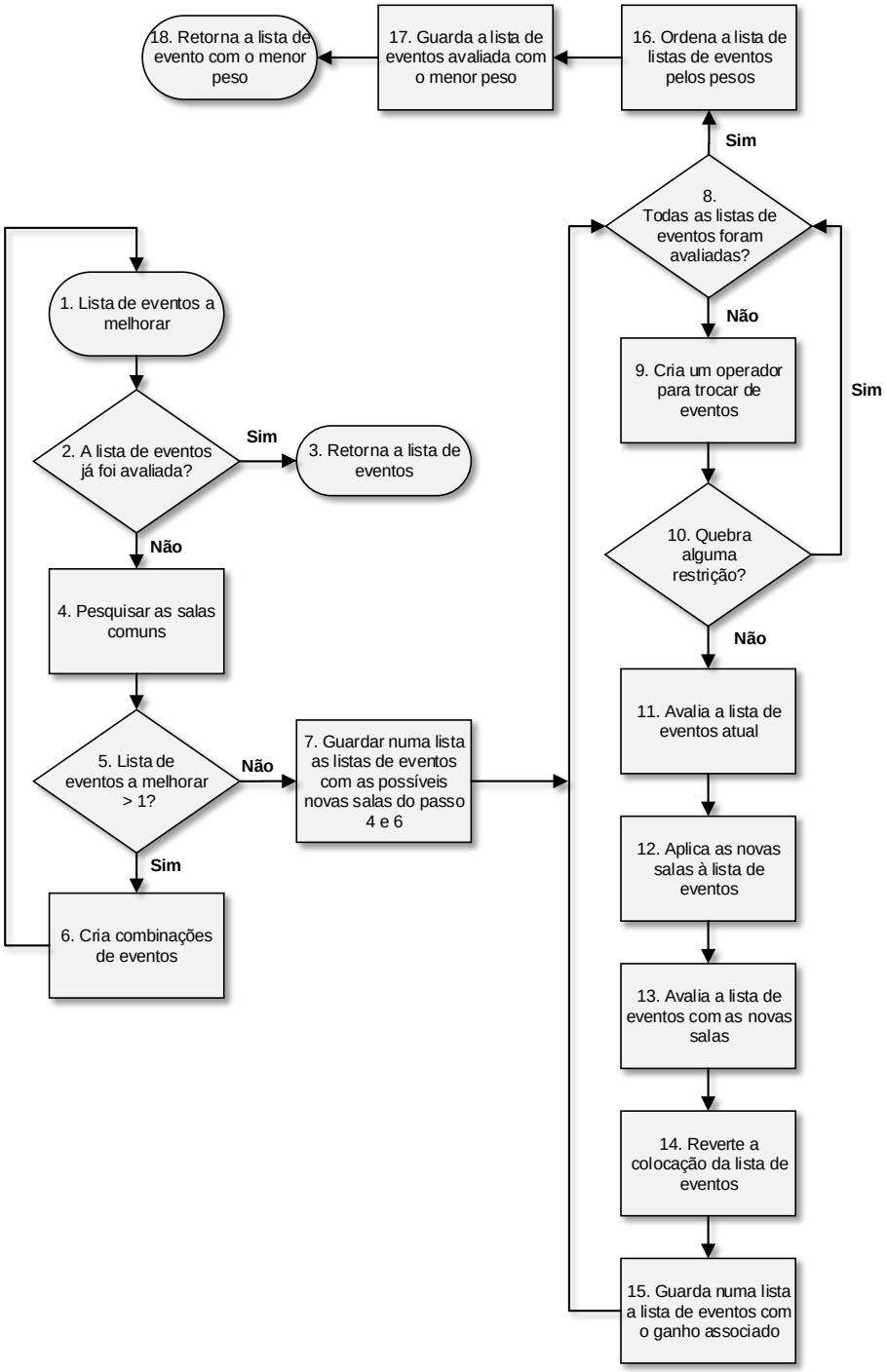


Figura 3.2: Otimização da salas da lista de eventos

3.3.1 Pesquisa de salas comuns aos eventos

Nesta estratégia realiza-se uma pesquisa de salas comuns a todos os eventos de um período do dia. As salas disponíveis para cada evento, estão ordenadas por ordem crescente do grau de adequação da sala ao evento, em que, este grau tenta aproximar a capacidade da sala e o número de alunos que o evento possui.
O seguinte exemplo serve para auxiliar numa melhor compreensão de como esta pesquisa é realizada. Na Turma X, à segunda-feira de manhã ocorrem três eventos (teóricos, T), como é apresentado na figura 3.3 A:

- a unidade curricular Matemática I tem a sala A atribuída e, tem as salas disponíveis $\{$A, B, C$\}$;
- a unidade curricular Introdução à Física com a sala C atribuída e dispõem das salas $\{$C, B$\}$;
- a unidade curricular Laboratório de Física que tem como sala B atribuída e tem as salas disponíveis $\{$C, B$\}$.

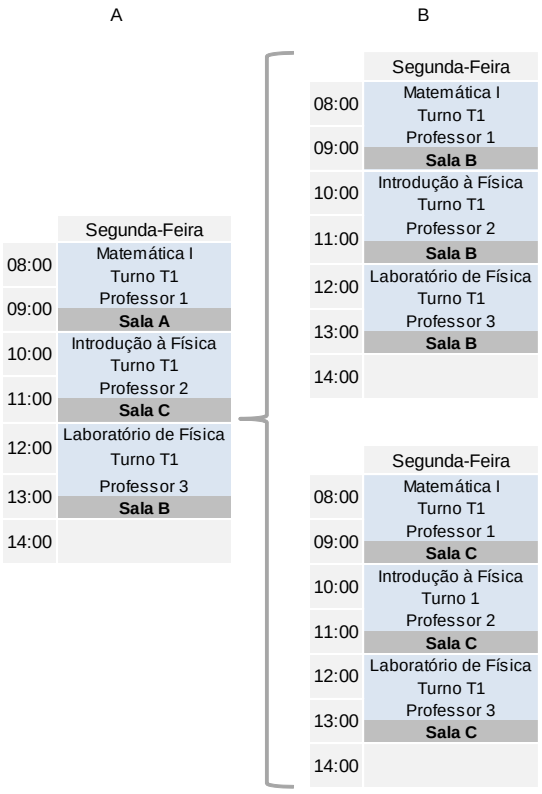


Figura 3.3: Representação da estratégia para pesquisa de sala comuns aos eventos que possuem apenas uma sala por evento.

Neste período do dia a Turma X tem de trocar de salas duas vezes, da sala A para a C e da sala C para a B, como mostra a figura 3.3 A. Nesta estratégia é realizada a procura de um conjunto de salas em que não seja necessário recorrer a nenhuma troca de salas para a Turma X, como é apresentado na figura 3.3 B. Para colocar todos estes eventos na mesma sala intersepta-se as salas disponíveis dos eventos obtendo as salas B e C como hipótese, encontra-se representado na figura 3.3 B.

Outro caso possível, é um evento ocorrer em mais do que uma sala ao mesmo tempo, como mostra a figura 3.4. A cada conjunto de salas disponíveis para cada evento é atribuído um grupo$_i$ (i = 1, 2, ..., n), por exemplo, a unidade curricular Introdução à Física, possui dois grupos. O grupo$_1$ composto pelas salas $\{$C, B$\}$ e o grupo$_2$ constituído pela sala $\{$K$\}$. O próximo exemplo serve para explicar como esta pesquisa é realizada. Numa Turma X à segunda-feira de manhã ocorrem três eventos (teóricos, T):

- a unidade curricular Matemática I com a sala $\{$A$\}$ atribuída e as salas disponíveis são $\{$A, B, C $\}$;
- a unidade curricular Introdução à Física com as salas $\{$C$\}$ e $\{$K$\}$ atribuídas, e dispõem das salas $\{$C, B$\}$ e $\{$K$\}$;
- a unidade curricular Laboratório de Física que tem como salas $\{$B$\}$ e $\{$R$\}$ atribuídas, e que tem como salas disponíveis $\{$C, B$\}$ e $\{$K, R$\}$.

À lista de eventos faz-se uma filtragem de salas comuns por grupos, comparando o anterior com o próximo, obtendo o grupo$_1$ com as salas $\{$B, C$\}$ e no grupo$_2$ com a sala $\{$K$\}$. A estes dois grupos, é aplicado o produto cartesiano, originando duas combinações de salas comuns possíveis. Sendo a primeira combinação, no grupo$_1$ o conjunto $\{$B$\}$ e no grupo$_2$ o conjunto $\{$K$\}$, e a segunda combinação possível, o grupo$_1$ o conjunto $\{$C$\}$ e no grupo$_2$ o conjunto $\{$K$\}$, como representado pelas duas soluções apresentadas na figura 3.4 B. Ao existirem salas comuns em grupos diferentes, algumas filtragens podem ser feitas, como por exemplo, no grupo$_1$ tem o conjunto de salas $\{$A, B$\}$, e no grupo$_2$ tem o conjunto de salas $\{$B, A$\}$. As combinações resultantes aceites do produto cartesiano são apenas AB, rejeitando as combinações AA, BB, BA, pois, se o evento escolhido ocorrer na sala A, então a segunda sala não pode ser a A, uma vez que, esse espaço vai estar ocupado. No caso de um evento ocorrer na sala A e B, ou vice-versa, a mesma situação ocorre daí apenas se obter uma combinação para o evento em questão.

Quando a lista de eventos têm só um evento, são utilizadas as melhores salas para esse evento, através do grau de adequação.

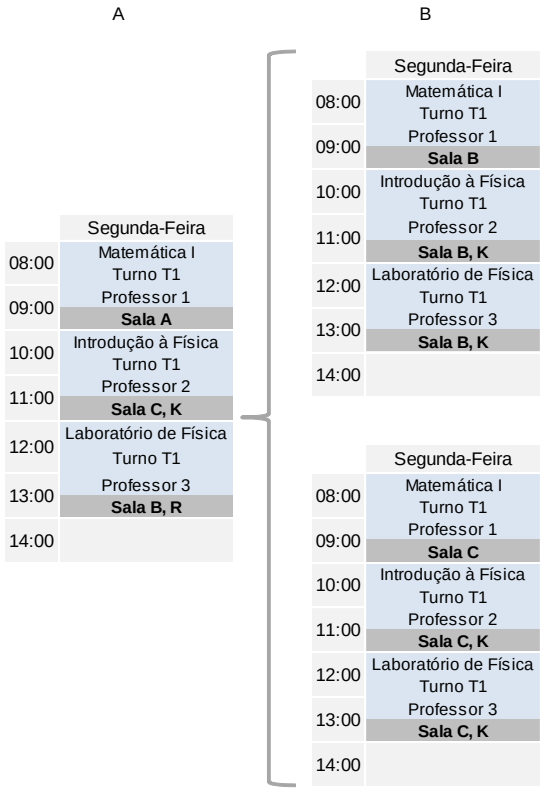


Figura 3.4: Representação da estratégia para pesquisa de sala comuns aos eventos que têm mais que uma sala por evento.

3.3.2 Pesquisa de salas criando combinações de eventos

O algoritmo de otimização de salas utiliza uma segunda estratégia de pesquisa de salas. Esta estratégia realiza uma pesquisa de salas que tenta agrupar os eventos a avaliar. As combinações de eventos são realizadas de modo a se conseguir obter o maior número de eventos consecutivos na mesma sala de aula. A figura 3.5 mostra como a combinação dos eventos é feita para a Turma X, já descrita na seção 3.3.1. A lista de eventos resulta em duas listas de combinações:

1. {*Matemática I*}, {*Introdução à Física, Laboratório de Física*};

2. {*Matemática I, Introdução à Física*}, {*Laboratório de Física*}.

A ideia desta pesquisa é dividir o problema em vários problemas menores e, assim, tentar resolver estes problemas com uso da recursividade, após estarem resolvidos é feita a união destes problemas, devolvendo a melhor solução possível (figura 3.6).

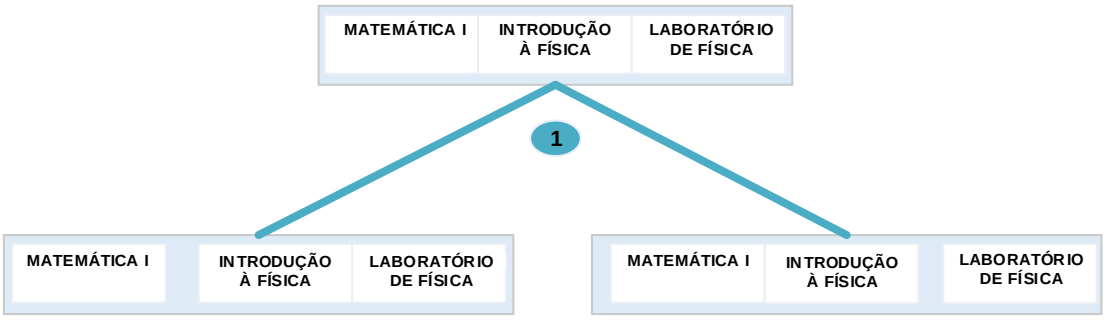


Figura 3.5: Criação da combinação dos eventos

As listas de combinações de eventos são iteradas, começando com a primeira lista de combinações, onde tenta pesquisar salas melhores para Matemática I e depois para $\{$ Introdução à Física, Laboratório de Física$\}$. O algoritmo de otimização de salas de eventos 3.3 é aplicado no evento com a unidade curricular Matemática I, este aplica a estratégia de pesquisar salas comuns, ficando então, com a lista de salas disponíveis A, B e C do evento, como explicado em 3.3.1. Devido à recursividade aplica-se novamente a criação de combinações de eventos, mas esta deixa de ser possível, pois a lista de eventos contém unicamente o evento de Matemática I, não existindo assim, mais combinações possíveis. A lista salas disponíveis são as do evento de Matemática I. O evento, por cada sala diferente verifica se respeita as restrições e avalia os objetivos das salas, devolvendo o evento de Matemática I com sala que tenha uma menor penalização, por exemplo, a sala A como mostra a figura 3.6 no passo 2.

Após a primeira parte da lista de combinações pesquisada, realiza-se a pesquisa melhores salas para o conjunto $\{$Introdução à Física, Laboratório de Física$\}$. O algoritmo otimização de eventos, é aplicado para o conjunto de eventos Introdução à Física, Laboratório de Física. A estratégia de pesquisar salas comuns é aplicada, ficando com as salas comuns B e C. A estratégia de criar combinações resulta numa lista de eventos, com eventos isolados [$\{$Introdução à Física$\}$, $\{$Laboratório de Física$\}$], representado pelo passo 4. O algoritmo otimização de salas, é chamado para o evento Introdução à Física, aplicando os mesmos passos de que quando se tinha só o evento de Matemática I, obtendo-se assim, a sala de aula com menor penalização para a Introdução à Física, por exemplo, a sala C, representada pelo passo 5. De seguida, os mesmo passos são repetidos para a aula de Laboratório de Física, obtendo a sala B, representado pelo passo 6.

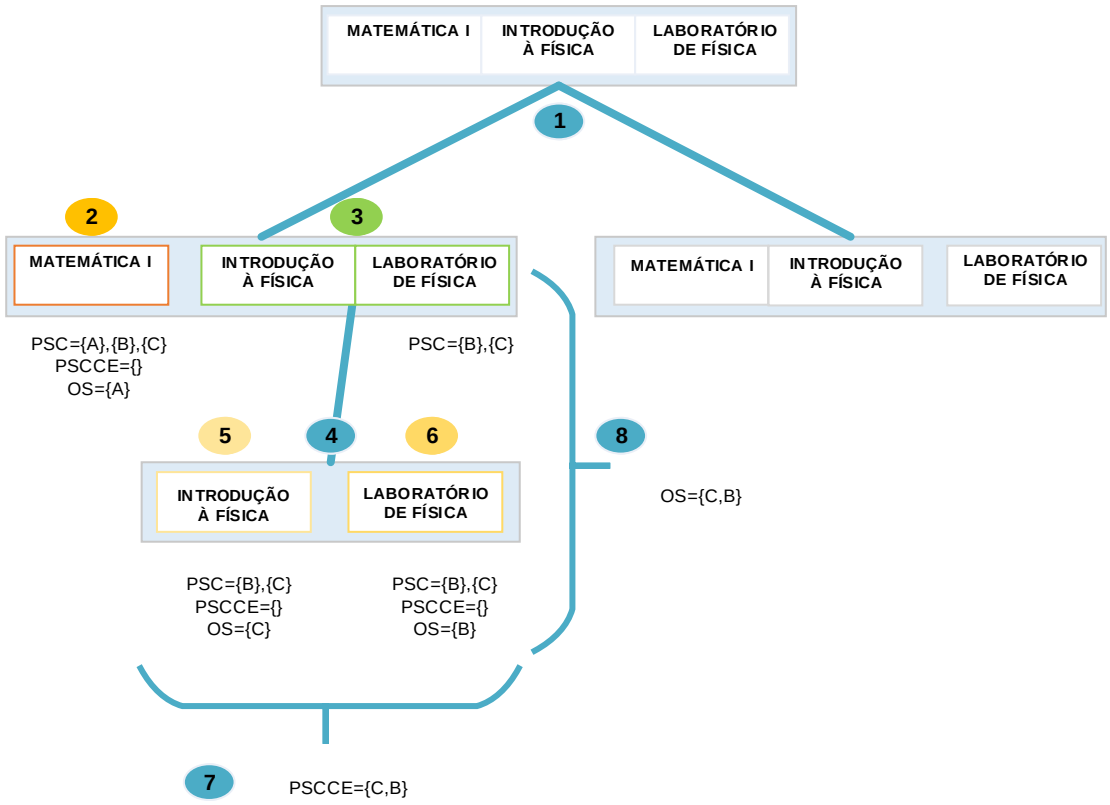


Figura 3.6: Representa parte da pesquisa de salas a combinações de eventos. PSC - pesquisa de salas comuns. PSCCE - pesquisa de salas criando combinações de eventos. OS - Otimização de salas de eventos.

Todas as combinações de eventos iteradas do passo 4, faz-se a união do resultado destas presentes no passo 5 e 6, obtendo assim, a sala C para a Introdução à Física e a sala B para Laboratório de Física, representado pelo passo 7. Esta união ocorre porque esta solução tem de ser avaliada no seu conjunto.

As novas soluções para Introdução à Física e para Laboratório de Física são as que resultaram da pesquisa de salas comuns (com duas possíveis salas a B e C para este eventos), e da pesquisa das salas com diferentes combinações de eventos (com uma possível solução, a sala C para a Introdução à Física e, a sala B para Laboratório de Física), devolvendo a melhor solução, representada no passo 8 (Introdução à Física a sala C e Laboratório de Física a sala B). Assim, para primeira lista de combinações, obtém-se a sala A para Matemática I , sala C para a Introdução à Física e a sala B para Laboratório de Física.

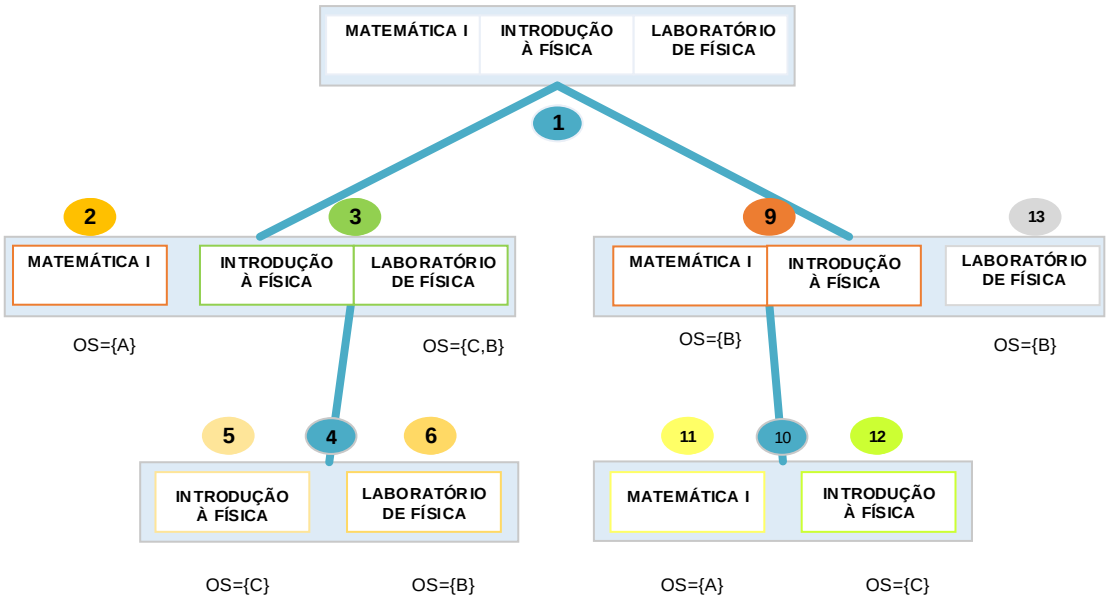


Figura 3.7: Representação da pesquisa de salas com as combinações de eventos. OS - Otimização de salas de eventos.

Após finalizada a pesquisa de salas para a primeira combinação é iniciada a pesquisa para a segunda lista de combinações de eventos. O conjunto de eventos {Matemática I, Introdução à Física} é iterado, representado no passo 9 da figura 3.7, e depois, o evento de Laboratório de Física, no passo 13. O algoritmo otimização de salas é chamado para a lista de eventos {Matemática I, Introdução à Física}, e é aplicado à estratégia de pesquisar salas comuns, ficando com a lista de salas comuns {B e C}. A estratégia de criar combinações é aplicada onde resultam os eventos isolados [{Matemática I} e {Introdução à Física}]. O algoritmo otimização de salas volta a ser aplicado para o evento de Matemática I, mas este já foi avaliado é então devolvida a melhor sala encontrada para este evento. O mesmo acontece para o evento de Introdução à Física. Depois de iteradas as combinações de eventos faz-se a união destas combinações resultando na sala A para Matemática I e na sala C para Introdução à Física.

As novas soluções obtidas para a Matemática I e para a Introdução à Física são as que resultaram da pesquisa de salas comuns e da pesquisa de salas com diferentes combinações de eventos são avaliadas, devolvendo a melhor solução para os eventos de Matemática I e Introdução à Física, então, fica-se com a sala B como a melhor alocação para os dois eventos. Na segunda lista de combinações, falta ainda pesquisar as salas para o evento de Laboratório de Física. O algoritmo otimização de salas é chamado para

o evento, mas esta já foi avaliada, retornando assim, a aula de Laboratório de Física com a melhor sala, sala B.

Concluindo assim, para primeira combinação de eventos, as salas são todas diferentes, onde no evento de Matemática I a melhor sala encontrada foi sala A, para o evento de Introdução à Física a melhor sala foi sala C, e, para Laboratório de Física a sala B. Na segunda combinação de eventos, a melhor sala encontrada foi a sala B para os três eventos.

3.4 Objetivos

Os objetivos que são utilizados pelo algoritmo MAS encontram-se referidos nesta secção.

3.4.1 Preferência dos horários das salas

A preferência do horário da sala está associada à disponibilidade de uma sala para alocação de um evento, num período de tempo. A cor amarela e laranja estão associada uma penalização, esta penalização atribuída é menor na cor amarela do que na cor laranja. A cor verde não tem nenhuma penalização, pois é o horário destinado para a atribuição dos eventos. Ao tentar atribuir um evento num período de tempo e não se encontrar uma colocação válida para o evento nas salas disponíveis sem ser nas cores com penalização, então o evento é atribuído nesse período de tempo com uma penalização.

3.4.2 Utilização das salas mais indicadas para cada aula

Utilização das salas mais indicadas para cada evento se refere à utilização das salas preferidas para o evento e se o número de alunos do evento é próximo da capacidade da sala de aula. Sendo atribuída uma penalização ao evento se a sala não é a preferida para ocorrer o evento e também é atribuída uma penalização no caso de capacidade da sala utilizada for muito maior do que o evento necessita, por exemplo a sala ter mais vinte e cinco lugares do que o evento necessita.

3.4.3 Evitar trocas entre salas de edifícios diferentes nos horários das turmas e dos docentes

Os eventos nos horários das turmas devem ocorrer em salas do mesmo edifício de modo a que as turmas não tenham de se deslocar de edifício, quando há uma troca de salas

entre edifícios necessária é atribuída uma penalização. O mesmo se aplica aos horários dos professores.

3.4.4 Evitar trocas de salas nos horários das turmas e dos docentes

Os eventos nos horários das turmas devem ocorrer na mesma sala de forma a que a turma não tenham de mudar de sala, quando é feita uma troca de de salas é atribuída uma penalização. O mesmo se aplica aos horários dos professores.

3.4.5 Minimizar a alternância de aulas no horário

Assim que um evento é atribuído a uma sala, todos os eventos referentes à mesma aula (por exemplo, aula teórica de Física) devem ser colocados todos seguidos, de forma a evitar as alterações de logística das salas para que os eventos que forem atribuídos a uma sala possuam o equipamento mais semelhantes possível entre si. No caso da aula não ser atribuída, de forma sucessiva, atribui-se uma penalização aos eventos que não ocorrem nas salas indicadas, mas também aos eventos que ocorrem indevidamente nessas salas.

3.4.6 Evitar furos nos horários das salas

As salas quando usadas num dia devem ser aproveitadas de forma a serem ocupadas com eventos, sempre que possível. Quando num horário de uma sala entre eventos existe um período de tempo sem eventos é atribuída uma penalização.

3.4.7 Aulas com repetição na mesma sala

As aulas com repetição devem acontecer na mesma sala, se a sala for diferente para algum desses eventos repetidos durante essa semana, é aplicada uma penalização ao evento, por cada sala diferente.

3.5 Análise da performance do algoritmo

Nesta secção procura-se mostrar quais os pontos que possam ser considerados como críticos na execução do algoritmo melhorar atribuição salas, anteriormente citado. A avaliação de performance é feita quanto ao tempo de execução e quanto à qualidade da solução

obtida. A máquina utilizada para a realização dos testes foi o Intel Core 2 Duo 3 GHz com 4 GB RAM.

A análise de performance foi realizada para as funções que pudessem ser mais relevantes na avaliação do tempo de execução de cada uma, para tal, foram utilizados cinco casos de estudo, sendo estes casos reais de IES. Estes casos variam quanto ao número de professores, edifícios, turmas e salas a avaliar.
O Caso 1 refere-se a um caso de pequena dimensão, que possui uma menor quantidade de horários de professores e turmas, sendo os tempos de execução medidos aos 5 e aos 8,65 minutos (momento em que o algoritmo termina de executar). À medida que se vai avançando nos casos de estudo (Caso 2, Caso 3, Caso 4, Caso 5) a quantidade de recursos envolvidos na geração vai aumentando, bem como o respetivo tempo de execução, pelo que as medições são efetuadas aos 30, 60 e 120 minutos.
Para cada caso efetuou-se vinte simulações, estando a geração limitada aos 120 minutos. A medição dos tempos de execução foi realizada com o auxílio da classe *Stopwatch*, os valores médios destes tempos encontram-se na tabela 3.1.

	Caso 1		**Caso 2**			**Caso 3**			**Caso 4**			**Caso 5**		
Tempo (minutos) / Funções	**5,00**	**8,65**	**30,00**(*)	**60,00**(*)	**120,00**(*)	**30,00**(*)	**60,00**(*)	**120,00**(*)	**30,00**(*)	**60,00**(*)	**120**(*)	**30,00** (*)	**60,00**(*)	**120,00**(*)
`OtimizarSalas`														
`SalasComuns`	0,03%	0,03%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
`CombinaçãoEventos`	0,01%	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
`Operador`	1,23%	1,09%	0,25%	0,24%	0,24%	0,44%	0,43%	0,43%	0,37%	0,32%	0,30%	0,22%	0,24%	0,23%
`Restrições`	36,58%	26,38%	80,10%	80,84%	79,58%	88,54%	88,33%	88,40%	67,23%	68,28%	62,24%	50,25%	55,66%	53,10%
`FunçãoObjetivo`	4,37%	3,98%	0,63%	0,63%	0,63%	0,29%	0,30%	0,30%	0,06%	0,06%	0,05%	0,12%	0,13%	0,13%
`Aplica`	6,08%	5,28%	0,52%	0,52%	0,51%	0,27%	0,27%	0,27%	0,06%	0,06%	0,06%	0,15%	0,16%	0,15%
`Reverte`	6,29%	5,45%	0,52%	0,52%	0,52%	0,27%	0,27%	0,27%	0,06%	0,07%	0,06%	0,15%	0,16%	0,15%
`AceitaNovasSalas`	0,24%	0,16%	0,15%	0,10%	0,06%	0,11%	0,08%	0,05%	0,10%	0,12%	0,10%	0,14%	0,15%	0,13%
`AtualizarInterface`	44,82%	57,27%	17,46%	16,92%	18,31%	9,76%	10,11%	10,16%	31,40%	30,28%	36,70%	48,19%	42,92%	45,62%
`Outras`(**)	0,34%	0,34%	0,37%	0,22%	0,14%	0,32%	0,21%	0,10%	0,72%	0,81%	0,48%	0,78%	0,58%	0,48%
`Total`	100,00%													

(*) Tempo limite para a execução do algoritmo.

(**) Guarda outras funções (como é o caso `FunçãoObjetivoDaCorrida`).

Tabela 3.1: Utilização do tempo do algoritmo MAS anteriormente implementado em percentagem do tempo de execução total.

Dois pontos críticos que se destacam na execução do algoritmo, apresentam-se sombreados na tabela 3.1. O primeiro ponto pela análise da tabela é a avaliação das restrições e o segundo o da atualização dos dados da interface para o utilizador. Através da análise da tabela pode-se verificar que estes são pontos críticos do algoritmo.
Outro ponto que se tem de ter em conta é referente aos objetivos, apesar dos tempos de execução não serem muito elevados, a validação das restrições que é feita inicialmente, evita a avaliação desnecessária dos objetivos.

Na tabela 3.2 encontra-se representado o número total de iterações necessárias para cada caso em estudo e as iterações respetivas.

	Caso 1		Caso 2			Caso 3			Caso 4			Caso 5		
Tempo (minutos)	**5**	**8,65**	**30**(*)	**60**(*)	**120**(*)	**30**(*)	**60**(*)	**120**(*)	**30**(*)	**60**(*)	**120**(*)	**30** (*)	**60**(*)	**120**(*)
Nº total de iterações	12096		32242			45010			118664			728644		
Iterações efetuadas	45,00%	100,00%	11,27%	21,83%	47,03%	3,94%	8,12%	5,96%	1,79%	3,13%	6,30%	0,03%	0,05%	0,15%

(*) Tempo limite para a execução do algoritmo.

Tabela 3.2: Iterações realizadas e o valor da otimização da conseguido da solução.

Nos casos avaliados pode-se verificar que o único caso que conseguiu concluir a análise de todos os horários foi o Caso 1, aos 8,65 minutos, como se pode observar na tabela 3.2, devido à menor quantidade de horários de professores e de turmas a otimizar. Os restantes casos apresentados encontram-se limitados a 120 minutos, não conseguindo tirar partido do aproveitamento máximo do algoritmo, pois o este não consegue iterar todos os horários. Quanto maior o número de horários a avaliar, menor o número de iterações que o algoritmo efetua-a, tornando também menor o aproveitamento obtido, pois o algoritmo despende muito tempo na avaliação de restrições e na atualização da interface para utilizador.

A figura 3.8 mostra o melhoramento conseguido quando o algoritmo MAS é executado. O aproveitamento é dado pela seguinte fórmula:

$$Aproveitamento\,(\%) = \frac{valorInicial - valorMAS}{valorInicial}$$

Onde $valorInicial$ é o valor inicial da função objetivo das salas e $valorMAS$ é o valor da função objetivo das salas, após aplicação do algoritmo MAS.

Através da análise dos objetivos e dos dados que são inseridos, o melhoramento da solução que se consegue obter não é muito acentuado, pois nem todos os eventos possuem o mesmo leque de salas disponíveis, isto leva a que, por exemplo, não se consiga atribuir todos os eventos de um horário à mesma sala, obtendo-se assim uma penalização.

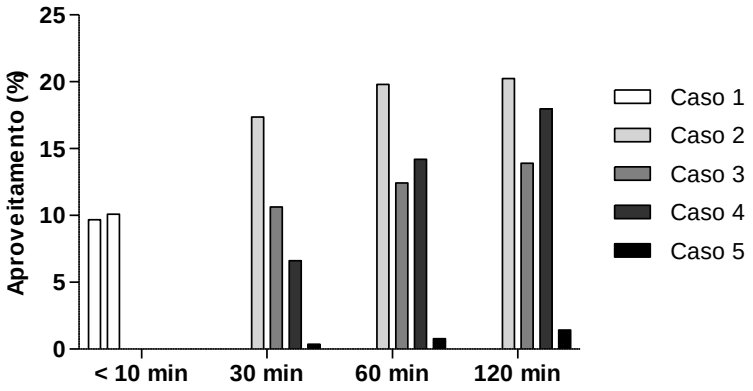


Figura 3.8: Melhoramento da solução em percentagem da penalização da solução inicial.

O algoritmo quando se encontra noutras fases, que não a de MAS, já utiliza a estratégia de atribuir as salas mais adequadas aos eventos em análise, tentando então, aproximar o número de alunos à capacidade das mesmas (sem nunca ultrapassar a sua capacidade), bem como alocar a sala preferencial para o evento em questão.

Capítulo 4

Implementação

Este capítulo descreve as alterações e melhorias que foram efetuadas no algoritmo MAS de forma a conseguir uma maior performance.

4.1 Otimização do código

Durante a análise do código anteriormente implementado no algoritmo persistiu a dificuldade em perceber o que realmente pertencia ou não ao algoritmo, pois este tinha código que não era utilizado. Outra das dificuldades para a perceção do código implementado foi existência de métodos extensos.

De modo a solucionar este tipo de problema procedeu-se a utilização de algumas boas práticas de implementação para reconstrução do código e implementação do novo código, como, a limpeza de código de implementações anteriores que não interessavam, evitar a criação de código repetido, a documentação do código implementado e minimização da dependência do código tornando os métodos mais curtos e simples [11]. Isto tem como objetivo tornar o código mais legível e de fácil manutenção no futuro.

O algoritmo à medida que se aumenta o número de entidades envolvidas para a otimização dos horários, o número de horários avaliados diminuía, para um intervalo de tempo definido. De forma a tentar conseguir aumentar o número de horário avaliados tentou-se sempre que possível utilizar instruções de código eficientes de forma a melhorar o tempo de execução [12].

4.2 Avaliação das restrições

O algoritmo melhorar atribuição de salas é uma parte do algoritmo do BTTE que tem como finalidade encontrar uma melhor afetação de salas para os eventos já alocados. Atualmente, o algoritmo na alocação de salas utiliza apenas os objetivos que estão relacionados com as salas, utilizando todas as restrições do problema original.
A validação das restrições tem um peso substancial no algoritmo durante o tempo de execução, como mostra a tabela 3.1. Após a análise às restrições verificou-se que só seria necessário utilizar as restrições referentes às trocas de salas de aula, pois os eventos não são movidos de posição no horário. Por exemplo, quando se atribui uma possível sala a um evento, não é necessário verificar se o professor tem disponibilidade, pois o professor foi alocado anteriormente ao evento, sendo feita a respetiva validação da restrição no momento dessa alocação.

O problema fica então com as seguintes restrições:

- nos eventos dos horários turmas e dos docentes que ocorrerem em salas de diferentes edifícios é necessário garantir que há um tempo mínimo de deslocação entre edifícios, para que os alunos e os docentes se possam deslocar antes do próximo evento a iniciar;
- cada sala só pode ter um evento atribuído em cada *slot* do horário.
- as salas só podem receber eventos com um número de alunos que respeitem as suas capacidades;
- indisponibilidade nos horários das salas, onde não é possível atribuir novos eventos.

Para esta redefinição do algoritmo foi necessário criar um novo operador que, em cada lista de eventos, adiciona os objetivos e as restrições (sendo apenas, usada a restrição do tempo mínimo de deslocação entre edifícios, pois as outras três encontram-se implícitas na estrutura de dados implementadas anteriormente) só das salas em análise. A função deste operador passa por guardar o estado anterior, facilitando assim, a reversão da solução. No momento em que, uma solução é aplicada ou revertida, torna-se necessário realizar a atualização das entidades que se encontram disponíveis ou ocupadas em cada momento.

4.3 Atualização de dados na interface gráfica

O algoritmo MAS, a cada iteração, após procurar uma nova solução, realizava a verificação se a interface para utilizador era ou não atualizada. A atualização dos dados era feita, no caso do valor da função objetivo ser menor ou igual ao melhor valor da função obtido até ao momento. Quando a solução encontrada tinha o mesmo peso que a solução

anterior, dois casos poderiam acontecer (figura 4.1 B e C), respetivamente o primeiro caso, a solução encontrada ser igual à solução anterior, e o segundo caso, a solução encontrada ser diferente da solução anterior. Ao acontecer qualquer um destes casos, os dados da interface eram atualizados.

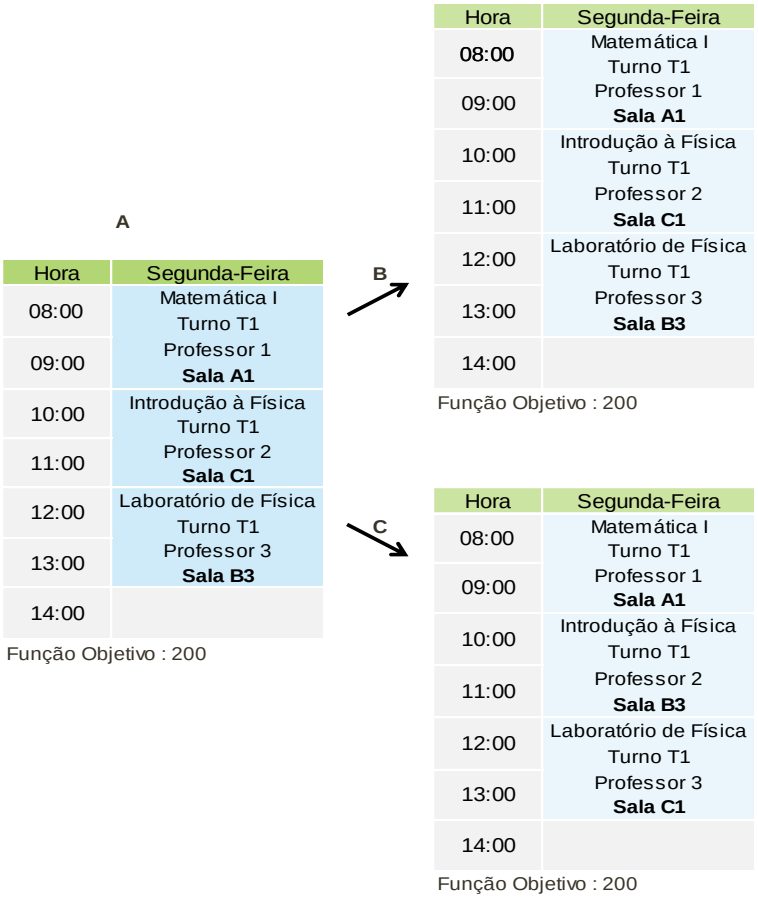


Figura 4.1: A solução A, pode dar origem a solução B ou C, com mesmo peso.

Alguns dos dados apresentados na interface ao utilizador são os pesos dos objetivos da solução. Assim sendo, era necessário realizar o cálculo para todos os objetivos do problema, de maneira a apresentar quais as melhorias conseguidas para o utilizador, tendo um peso elevado no tempo de execução, como se pode verificar na tabela 3.1.

A atualização dos dados para a interface foi assim alterada de forma a que, sempre que se obtivesse uma solução com o mesmo peso, só se realizava a atualização da interface, se a solução anterior fosse diferente da atual.

4.4 Objetivo evitar furos das salas

Neste objetivo existe um conjunto de valores que não entram no cálculo da penalização e nos valores que são contabilizados para este cálculo. Os valores que não contam para a penalização são 0 e -1. O valor 0 é atribuído às primeiras horas do dia até se encontrar o primeiro evento do dia, e a partir do último evento do dia, limpando assim, as penalizações das últimas horas do dia. O valor -1 é atribuído aos intervalos de tempo onde os eventos se encontram atribuídos nos horários. Os valores associados à penalização são 1 e 0,5. O valor 1 é atribuído quando entre eventos existe um intervalo de tempo sem eventos. O valor 0,5 é atribuído entre eventos em que existe um intervalo de tempo sem eventos perto da divisão do dia (por exemplo, paragem para almoço).

A forma como era percorrido o horário neste objetivo foi repensada, uma vez que, se encontrava a realizar iterações que podiam ser evitadas. A figura 4.2 A representa a forma como era atribuída a penalização quando ocorrem furos entre eventos num dia da semana de uma sala.

A

Hora	Segunda-Feira	Etapas 1º	2º	3º	4º	5º
8:00 - 8:30		**1**	1	1	**0**	0
8:30 - 9:00		**1**	1	1	**0**	0
9:00 - 9:30	**Química I**	**-1**	-1	-1	-1	-1
9:30 - 10:00	**Turno T1**	**-1**	-1	-1	-1	-1
10:00 - 10:30	**Professor 1**	**-1**	-1	-1	-1	-1
10:30 - 11:00	**Sala A**	**-1**	-1	-1	-1	-1
11:00 - 11:30		**1**	1	**0,5**	0,5	0,5
11:30 - 12:00		**1**	1	**0,5**	0,5	0,5
12:00 - 12:30		**1**	1	**0,5**	0,5	0,5
12:30 - 13:00		**1**	1	**0,5**	0,5	0,5
13:00 - 13:30		**1**	**0,5**	0,5	0,5	0,5
13:30 - 14:00		**1**	**0,5**	0,5	0,5	0,5
14:00 - 14:30	**Matemática I**	**-1**	-1	-1	-1	-1
14:30 - 15:00	**Turno T1**	**-1**	-1	-1	-1	-1
15:00 - 15:30	**Professor 1**	**-1**	-1	-1	-1	-1
15:30 - 16:00	**Sala A**	**-1**	-1	-1	-1	-1
16:00 - 16:30		**1**	1	1	1	1
16:30 - 17:00	**Física**	**-1**	-1	-1	-1	-1
17:00 - 17:30	**Turno T1**	**-1**	-1	-1	-1	-1
17:30 - 18:00	**Professor 2**	**-1**	-1	-1	-1	-1
18:00 - 18:30	**Sala A**	**-1**	-1	-1	-1	-1
18:30 - 19:00		**1**	1	1	1	**0**
19:00 - 19:30		**1**	1	1	1	**0**

B

Hora	Segunda-Feira	Etapas 1º	2º	3º	4º	5º
8:00 - 8:30		**1**	**0**	0	0	0
8:30 - 9:00		**1**	**0**	0	0	0
9:00 - 9:30	**Química I**	**-1**	-1	-1	-1	-1
9:30 - 10:00	**Turno T1**	**-1**	-1	-1	-1	-1
10:00 - 10:30	**Professor 1**	**-1**	-1	-1	-1	-1
10:30 - 11:00	**Sala A**	**-1**	-1	-1	-1	-1
11:00 - 11:30		**1**	1	1	1	**0,5**
11:30 - 12:00		**1**	1	1	1	**0,5**
12:00 - 12:30		**1**	1	1	1	**0,5**
12:30 - 13:00		**1**	1	1	1	**0,5**
13:00 - 13:30		**1**	1	1	**0,5**	0,5
13:30 - 14:00		**1**	1	1	**0,5**	0,5
14:00 - 14:30	**Matemática I**	**-1**	-1	-1	-1	-1
14:30 - 15:00	**Turno T1**	**-1**	-1	-1	-1	-1
15:00 - 15:30	**Professor 1**	**-1**	-1	-1	-1	-1
15:30 - 16:00	**Sala A**	**-1**	-1	-1	-1	-1
16:00 - 16:30		**1**	1	1	1	1
16:30 - 17:00	**Física**	**-1**	-1	-1	-1	-1
17:00 - 17:30	**Turno T1**	**-1**	-1	-1	-1	-1
17:30 - 18:00	**Professor 2**	**-1**	-1	-1	-1	-1
18:00 - 18:30	**Sala A**	**-1**	-1	-1	-1	-1
18:30 - 19:00		**1**	1	**0**	0	0
19:00 - 19:30		**1**	1	**0**	0	0

Figura 4.2: Horário com aulas de manhã e tarde. A, representa como era percorrido o horário para a atribuição da penalização anteriormente. B, representa alteração que foi implementada para percorrer os horários.

O horário era percorrido atribuindo na primeira etapa 4.2 A, o valor 1, ao intervalo de tempo

sem eventos e, -1, ao intervalo de tempo com um evento atribuído.
A segunda e terceira etapa tinham uma menor penalização, apesar de serem consideradas um furo, encontravam-se perto da hora de almoço. Na segunda, desde a divisão do dia (13:00) até ao primeiro evento da tarde (14:00), era substituindo o valor de 1 por 0,5. Na terceira etapa, desde a divisão do dia até se encontrar o último evento da manhã (que termina às 11:00), substituindo o valor de 1 por 0,5.
A quarta e quinta etapa, serviam para limpar as penalizações das primeiras e últimas horas do dia sem eventos. Na quarta etapa, era percorrido o horário desde início da manhã (8:00) até encontrar a primeiro evento do dia, substituindo o valor de 1 por 0. Na quinta etapa era iterado o horário de forma inversa, desde o último intervalo de tempo do dia (19:30) até ao último evento do dia, substituindo o valor de 1 por 0.

Na nova implementação, a nível de cálculo da penalização, o resultado final é igual ao que era obtido anteriormente. A primeira etapa da figura 4.2 B é igual à da figura 4.2 A. Na segunda e na terceira etapa, é realizada a limpeza de penalizações. Isto permite evitar a atribuição de penalização de 0,5, pois quando se tem uma o valor 0 nos intervalos de tempos, não é necessário recorrer a uma substituição de penalização. Na quarta e quinta etapa, penaliza-se o que se encontra perto da hora da divisão do dia.

A

Hora	Segunda-Feira	Etapas 1º	2º	3º	4º	5º
8:00 - 8:30		**1**	1	**0,5**	**0**	0
8:30 - 9:00		**1**	1	**0,5**	**0**	0
9:00 - 9:30		**1**	1	**0,5**	**0**	0
9:30 - 10:00		**1**	1	**0,5**	**0**	0
10:00 - 10:30		**1**	1	**0,5**	**0**	0
10:30 - 11:00		**1**	1	**0,5**	**0**	0
11:00 - 11:30		**1**	1	**0,5**	**0**	0
11:30 - 12:00		**1**	1	**0,5**	**0**	0
12:00 - 12:30		**1**	1	**0,5**	**0**	0
12:30 - 13:00		**1**	1	**0,5**	**0**	0
13:00 - 13:30		**1**	**0,5**	0,5	**0**	0
13:30 - 14:00		**1**	**0,5**	0,5	**0**	0
14:00 - 14:30	**Matemática I**	**-1**	-1	-1	-1	-1
14:30 - 15:00	**Turno T1**	**-1**	-1	-1	-1	-1
15:00 - 15:30	**Professor 1**	**-1**	-1	-1	-1	-1
15:30 - 16:00	**Sala A**	**-1**	-1	-1	-1	-1
16:00 - 16:30		**1**	1	1	1	1
16:30 - 17:00	**Física**	**-1**	-1	-1	-1	-1
17:00 - 17:30	**Turno T1**	**-1**	-1	-1	-1	-1
17:30 - 18:00	**Professor 2**	**-1**	-1	-1	-1	-1
18:00 - 18:30	**Sala A**	**-1**	-1	-1	-1	-1
18:30 - 19:00		**1**	1	1	1	**0**
19:00 - 19:30		**1**	1	1	1	**0**

B

Hora	Segunda-Feira	Etapas 1º	2º	3º
8:00 - 8:30		**1**	**0**	0
8:30 - 9:00		**1**	**0**	0
9:00 - 9:30		**1**	**0**	0
9:30 - 10:00		**1**	**0**	0
10:00 - 10:30		**1**	**0**	0
10:30 - 11:00		**1**	**0**	0
11:00 - 11:30		**1**	**0**	0
11:30 - 12:00		**1**	**0**	0
12:00 - 12:30		**1**	**0**	0
12:30 - 13:00		**1**	**0**	0
13:00 - 13:30		**1**	**0**	0
13:30 - 14:00		**1**	**0**	0
14:00 - 14:30	**Matemática I**	**-1**	-1	-1
14:30 - 15:00	**Turno T1**	**-1**	-1	-1
15:00 - 15:30	**Professor 1**	**-1**	-1	-1
15:30 - 16:00	**Sala A**	**-1**	-1	-1
16:00 - 16:30		**1**	1	1
16:30 - 17:00	**Física**	**-1**	-1	-1
17:00 - 17:30	**Turno T1**	**-1**	-1	-1
17:30 - 18:00	**Professor 2**	**-1**	-1	-1
18:00 - 18:30	**Sala A**	**-1**	-1	-1
18:30 - 19:00		**1**	1	**0**
19:00 - 19:30		**1**	1	**0**

Figura 4.3: Horário com manhã livre. A, representa como era percorrido o horário para a atribuição da penalização anteriormente. B, representa alteração que foi implementada para percorrer os horários.

Na figura 4.3 A o procedimento é igual ao que foi explicado anteriormente. No entanto, na figura 4.3 B ocorrem menos duas etapas necessárias para atribuição da penalização, devido a não haverem eventos da parte da manhã. Assim, deixa de ser preciso atribuir a penalização relativa à proximidade da hora de almoço, sendo também válido para as tardes livres.

A nova abordagem em comparação com abordagem anterior representada na primeira a figura 4.2 não é vantajosa, pois o número de iterações que se tem de fazer é o mesmo para 4.2 A e para 4.2 B. No entanto, o mesmo não se verifica quando se tem uma manhã ou tarde livre, conseguindo assim, uma redução do número de iterações necessárias para atribuição das penalizações.

4.5 Restrição tempo mínimo de deslocação entre edifícios

A restrição implementada anteriormente começava por retirar os eventos da solução do horário em que se estava à procura de novas soluções. Após a remoção dos eventos aplicava-se a nova lista de eventos, estes eram aplicados um a um no horário, como mostra a figura 4.4. À medida que se aplicava cada evento era analisado se estes respeitavam a restrição a avaliar.

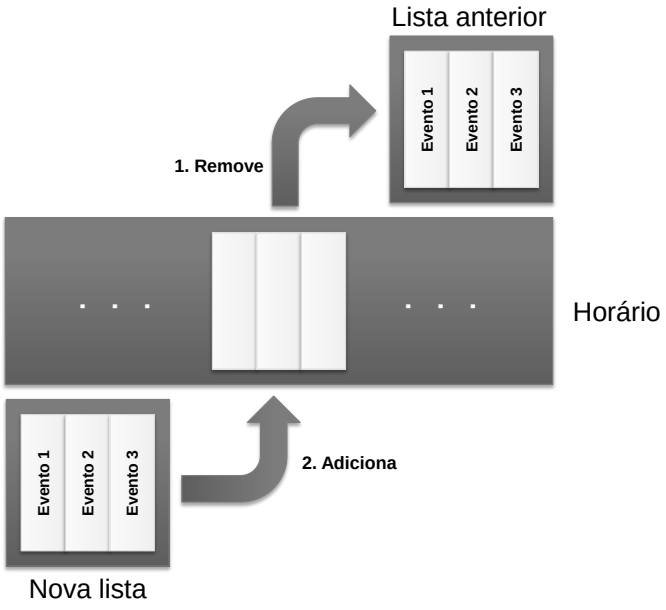


Figura 4.4: Restrição tempo mínimo de deslocação entre edifícios.

Quando algum dos eventos fosse aplicado no horário e não respeitasse a restrição, a

solução formada deixava de ser considerada válida e, era revertida para a solução no estado anterior (antes de se removerem os eventos).
Depois de percorrer toda a lista de eventos com as novas características, se a lista fosse válida, então, a solução era revertida para o estado anterior, devolvendo que a solução era válida.

Quando se aplicava um evento ao horário num dia, a restrição verificava se a(s) sala(s) onde o evento ocorria tinha(m) algum edifício em comum com o evento anterior, no caso de isto acontecer, a solução era considerada válida. No entanto, poderia não respeitar a ideia da restrição, como representado na figura 4.5.

Hora	Segunda-Feira
8:00 - 8:30	
8:30 - 9:00	
9:00 - 9:30	Matemática I
9:30 - 10:00	Turno T_1
10:00 - 10:30	Professor 1
10:30 - 11:00	**Sala A1,B1**
11:00 - 11:30	Física
11:30 - 12:00	Turno T_1
12:00 - 12:30	Professor 2
12:30 - 13:00	**Sala A2**

Figura 4.5: Horário onde sala B1 pertence ao edifício B e as salas A1 e A2 pertencem ao edifício A.O tempo mínimo deslocação entre o edifício A e B é de 30 minutos.

Os alunos que participem na aula de Matemática I na sala B não conseguem chegar a tempo para a próxima aula, que ocorre na sala A2 antes desta se iniciar, devido ao tempo de deslocação mínimo entre o edifício A e B. Anteriormente, esta situação ocorria sendo aceite pela restrição.
Esta verificação foi removida passando-se a respeitar a ideia para a qual a restrição foi inicialmente criada.
No caso dos eventos ocorrerem em diferentes edifícios é realizado o cálculo do tempo mínimo de deslocação entre edifícios. Este tempo é contado a partir do momento em que acaba o evento anterior até ao início do próximo evento.

Uma outra alteração realizada está relacionada com os novos eventos a colocar, possuírem exatamente os mesmos edifícios que os eventos a remover na mesma posição do horário. Isto é verificado logo no início da restrição evitando assim, a análise da restrição, que leva a um aumento da eficiência a nível de tempo de execução.

4.6 Ordenação dos horários

Anteriormente quando os horários das turmas e dos professores eram inicializados no algoritmo MAS, eram ordenados de forma aleatória em cada um dos conjuntos de horários.

Na nova implementação, a ordenação da lista de horários das turmas e da lista de horários dos professores, é realizada através do total dos pesos dos eventos de cada horário (figura 4.6). A ordenação aplicada tem como função melhorar o aproveitamento, no momento em que o algoritmo MAS não conclui o número de iterações que tem de efetuar. A ordenação realizada continua a ser feita nas duas listas avaliadas (turmas e professores) de forma separada, pois os alunos são o principal cliente da universidades. Assim, avaliando primeiro os horários das turmas e depois o dos professores.

Horários	Função objetivo
Turma A	1120
Turma B	1300
Turma C	1250
Turma D	1080

Horários	Função objetivo
Turma B	1300
Turma C	1250
Turma A	1120
Turma D	1080

Figura 4.6: Ordenação dos horários das turmas pelos peso dos eventos dos horários.

A ordenação é realizada através do método `Sort` da classe `List` [7]. A ordenação implementada só é aplicada na inicialização do algoritmo MAS, devido à avaliação do peso dos eventos de cada horário ao longo do algoritmo se tornar inviável, um pouco à semelhança do que acontece com atualização dos dados da interface para utilizador. Como esta ordenação só se realiza na inicialização, os horários que tem uma maior penalização na primeira iteração podem não o ser a segunda iteração, uma vez que, existem eventos comuns entre horários das turmas e dos professores, por exemplo a alteração de um evento para um horário de uma turma pode ser boa, mas para os outros horários das turmas/professores pode-o não o ser.

4.7 Colocação dos eventos em edifícios comuns

De forma a tentar melhorar a qualidade da solução foi implementada uma estratégia com base no objetivo evitar trocas de edifícios, de modo a que, se consiga colocar todos eventos de um horário no mesmo edifício, levando consequentemente, a uma diminuição da distância entre eventos de um horário de uma turma ou de um professor.
Uma boa solução atendendo aos objetivos do problema seria conseguir ter todos os eventos de uma turma ou de um professor na mesma sala, mas esta situação é muito difícil de acontecer, pois nem sempre as salas têm disponibilidade para os eventos que ocorrem, nem todos os eventos possuem as mesmas salas disponíveis. Assim, a estratégia que foi implementada tenta fazer com que a pesquisa de novas soluções seja mais abrangente.

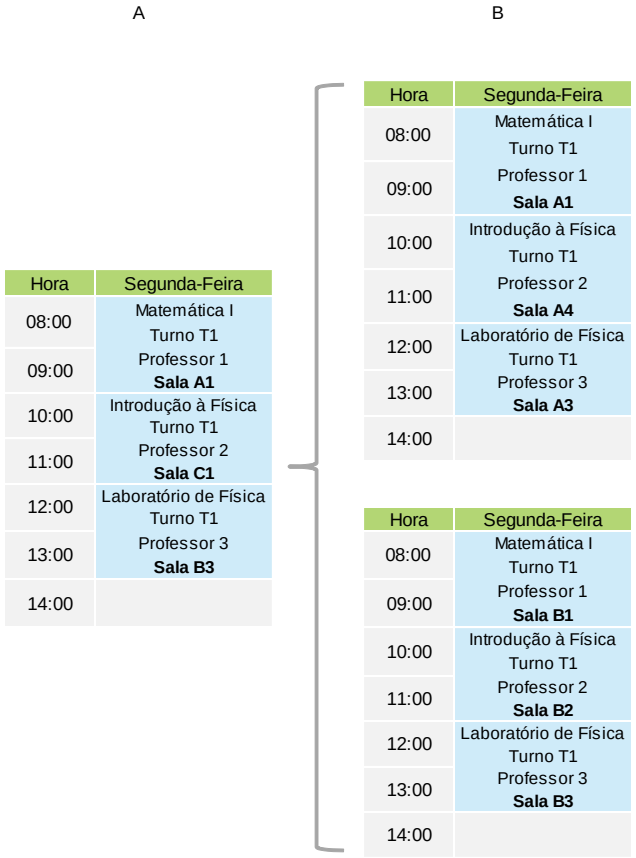


Figura 4.7: Estratégia para colocar os eventos em edifícios comuns.

As novas soluções conseguidas com esta estratégia estão focadas no objetivo evitar a troca de edifícios, de maneira a que se consiga obter um aumento na qualidade da solução dos objetivos que estão relacionados com as salas.

Esta estratégia foi inserida no algoritmo entre o passo 4 e 5, representada na figura 3.2, sendo utilizada na avaliação de pelo menos dois eventos, não sendo necessário realizar a procura de salas para um evento, existindo outra estratégia para este fim.
A procura de novas soluções, nesta estratégia começa por verificar se existe a possibilidade de colocar todos os eventos no mesmo edifício. No caso de se verificar se os eventos são associados às possíveis salas, devolve-se assim, essas soluções para posteriormente serem avaliadas.

O seguinte exemplo ajuda a uma melhor compreensão de como esta estratégia é realizada. Na Turma Y, à segunda-feira de manhã ocorrem três eventos (teóricos, T), como está apresentado na figura 4.7 A:

- a unidade curricular Matemática I tem a sala A1 atribuída, que dispõem da sala A1 do edifício A e da sala B1 do edifício B;
- a unidade curricular Introdução à Física com a sala C1 atribuída, que dispõem da sala C1 do edifício C, da sala B2 do edifício B e da sala A4 do edifício A ;
- a unidade curricular Laboratório de Física que tem como sala B3 atribuída, que dispõem da sala B3 do edifício B e da sala A3 do edifício A.

A figura 4.7 A representa um horário a ser otimizado, onde não se consegue colocar todos os eventos na mesma sala e tem-se a possibilidade da alocação dos eventos em edifícios comuns, para os eventos a avaliar da Turma Y. A figura 4.7 B mostra as duas novas possíveis soluções. Numa das soluções os eventos são atribuídos em salas do edifício A e na outra solução os eventos são atribuídos as salas do edifício B. Estas duas soluções são devolvidas para posteriormente serem avaliadas.

Capítulo 5

Resultados

5.1 Análise da performance do algoritmo e da qualidade

A análise de performance realizada nesta secção está relacionada com as otimizações e alterações que foram efetuadas ao algoritmo (que não tivessem impacto na forma como eram explorados as novas soluções), não se encontrando as alterações descritas na secção 4.6 e 4.7. Os casos de estudo utilizados são os mesmos citados anteriormente.
Após a implementação das alterações no algoritmo (denominado de *MAS alterado*), os casos de estudo, exceto o Caso 5, conseguem iterar todos os horários em alguns minutos, o que antes poderia demorar algumas horas com o algoritmo anteriormente implementado (*MAS inicial*). O Caso 5 continua a não conseguir iterar todos os horários, daí estar limitado aos 120 minutos, como se pode observar na tabela 5.1.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**		
MAS inicial	***8,65***	***120,00***(*)	***120,00***(*)	***120,00***(*)	***30,00***(*)	***60,00***(*)	***120,00*** (*)
MAS alterado	**1,10**	**2,70**	**2,80**	**6,00**	**30,00**(*)	**60,00**(*)	**120,00**(*)
Otimizar salas							
SalasComuns	0,16%	0,08%	0,08%	0,04%	0,01%	0,01%	0,02%
CombinaçãoEventos	0,04%	0,03%	0,03%	0,02%	0,00%	0,00%	0,00%
Operador	1,67%	2,18%	4,14%	1,92%	1,30%	1,82%	2,59%
Restrições	0,01%	0,00%	0,00%	0,11%	0,10%	0,13%	0,20%
FunçãoObjetivo	28,27%	35,38%	32,83%	13,80%	6,54%	8,84%	13,31%
Aplica	30,39%	19,85%	20,65%	10,06%	5,45%	7,27%	10,83%
Reverte	30,39%	19,71%	20,15%	9,57%	5,47%	7,27%	10,83%
AceitaNovasSalas	1,03%	3,65%	3,24%	11,64%	7,18%	8,38%	8,91%
AtualizarInterface	5,51%	12,16%	13,07%	29,72%	39,77%	33,16%	26,61%
Outras (**)	2,47%	6,85%	5,67%	23,05%	34,18%	33,09%	26,69%
Total	100,00%						

(*) Tempo limite para a execução do algoritmo.
(**) Guarda outras funções (como é o caso `FunçãoObjetivoDaCorrida`).

Tabela 5.1: Utilização do tempo do algoritmo MAS alterado em percentagem do tempo de execução total em minutos.

Com as modificações efetuadas no algoritmo, o tempo de execução encontra-se distribuído pelas funções. A redução do tempo de execução conseguida através das otimizações aplicadas, levaram a que o algoritmo consiga avaliar um maior número de soluções possíveis. Com estas alterações ao algoritmo, é necessário menos tempo para avaliar as restrições, e também, a atualização dos dados para interface do utilizador.
A avaliação de todos os objetivos continua a ter um peso elevado no Caso 5 quanto ao tempo despendido, pois os valores dos tempos obtidos para a atualização dos dados da interface e para guardar a melhor solução conseguida até ao momento, mantêm-se elevados.

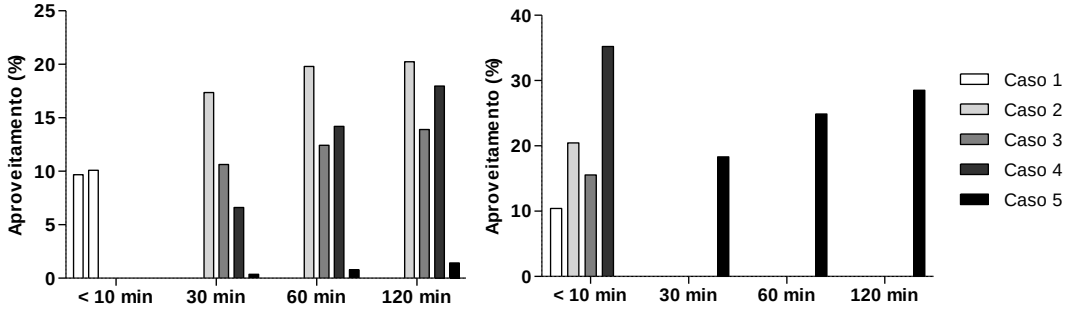
No caso 5, o algoritmo não consegue iterar todos horários, mas consegue obter uma maior percentagem de horários iterados, como se pode observar na tabela 5.2.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**		
Tempo (minutos)	**1,10**	**2,70**	**2,80**	**6,00**	**30,00**(*)	**60,00**(*)	**120,00**(*)
Nº total de iterações	12096	32242	45010	118664		728644	
Iterações efetuadas	100,00%	100,00%	100,00%	100,00%	5,20%	13,96%	42,15%

(*) Tempo limite para a execução do algoritmo.

Tabela 5.2: Iterações realizadas pelo algoritmo.

O melhoramento do tempo de execução do algoritmo resulta numa melhoria de eficiência da solução obtida. Uma vez que, se consegue iterar mais horários, há um aumento da qualidade da solução, como se pode observar na figura 5.1b.



(a) Algoritmo MAS inicial. (b) Algoritmo MAS alterado.

Figura 5.1: Melhoramento da solução em percentagem da penalização da solução inicial.

Através da análise da figura 5.1b, de uma forma geral, houve um aumento do aproveitamento nos casos estudados, sendo de salientar o Caso 5 com os melhores resultados de aproveitamento, seguindo-se do Caso 4, depois do Caso 2 e finalizando com o Caso 3. O

que manteve os valores de aproveitamento foi o Caso 1 em comparação com os valores da figura 5.1a. Outro aspeto relevante, observou-se um aumento do aproveitamento, no Caso 4 e 5.

5.2 Análise da performance do algoritmo e da qualidade

A análise de performance realizada nesta secção está relacionada com todas as alterações efetuadas (denominado de *MAS final*), incluindo as que têm impacto na forma como são explorados as novas soluções (secções 4.6 e 4.7).

Após as alterações no algoritmo referentes à ordenação e à estratégia de alocação dos eventos no mesmo edifício, tal como já tinha acontecido com a implementação da secção 5.1, para o Caso 1, 2, 3 e 4, o tempo de execução do algoritmo diminuiu em comparação com o algoritmo implementado inicialmente, sendo este tempo inferior a 10 minutos. No Caso 5, os tempos de execução mantiveram-se nos 120 minutos. Os tempos de execução avaliados podem ser observados na tabela 5.3.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**		
MAS inicial	***8,65***	***120,00***(*)	***120,00***(*)	***120,00***(*)	***30,00***(*)	***60,00***(*)	***120,00***(*)
MAS final	**1,10**	**2,70**	**3,75**	**5,90**	**30,00**(*)	**60,00**(*)	**120,00**(*)
Otimizar salas							
SalasComuns	0,15%	0,08%	0,11%	0,10%	0,03%	0,03%	0,03%
EdifíciosComuns	0,02%	0,01%	0,02%	0,01%	0,00%	0,00%	0,00%
CombinaçãoEventos	0,04%	0,03%	0,03%	0,02%	0,01%	0,01%	0,00%
Operador	1,65%	2,18%	3,19%	1,89%	8,53%	6,44%	3,82%
Restrições	0,01%	0,00%	0,00%	0,11%	0,71%	0,52%	0,30%
FunçãoObjetivo	27,95%	35,38%	25,07%	13,10%	23,73%	22,28%	18,19%
Aplica	30,06%	19,85%	15,94%	9,93%	17,77%	18,20%	15,78%
Reverte	29,98%	19,71%	15,55%	9,41%	17,68%	18,14%	15,77%
AceitaNovasSalas	1,03%	3,65%	5,73%	11,38%	11,21%	9,80%	9,61%
AtualizarInterface	5,98%	12,16%	29,78%	33,60%	11,43%	13,21%	19,31%
Outras (**)	1,99%	6,85%	4,52%	20,44%	8,90%	11,36%	17,18%
Total	100,00%						

(*) Tempo limite para a execução do algoritmo.
(**) Guarda outras funções (como é o caso `FunçãoObjetivoDaCorrida`).

Tabela 5.3: Utilização do tempo do algoritmo MAS final em percentagem do tempo de execução total em minutos.

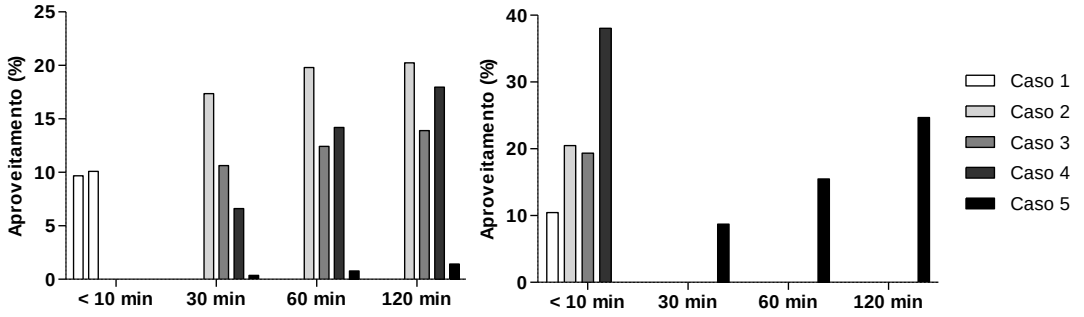
Com esta implementação também se conseguiu iterar mais horários na maioria dos casos, com exceção do Caso 5, como se pode verificar na tabela 5.4.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**		
Tempo (minutos)	**1,10**	**2,70**	**3,75**	**5,90**	**30,00**(*)	**60,00**(*)	**120,00**(*)
Nº total de iterações	12096	32242	45010	118664		728644	
Iterações efetuadas	100,00%	100,00%	100,00%	100,00%	14,57%	28,37%	62,47%

(*) Tempo limite para a execução do algoritmo.

Tabela 5.4: Iterações realizadas pelo algoritmo.

As alterações efetuadas no algoritmo resultam numa qualidade de solução superior à qualidade da solução inicialmente implementada. Os resultados da qualidade da solução obtida estão presentes na figura 5.2.



(a) Algoritmo MAS inicial. (b) Algoritmo MAS final.

Figura 5.2: Melhoramento da solução em percentagem da penalização da solução inicial.

Na análise da figura 5.2 pode-se observar um aumento similar no aproveitamento dos Casos 1, 2, 3 e 4, tanto em comparação com os dados da figura 5.2a como os da figura 5.1b. No Caso 5, o mesmo não se verifica, pois, comparando com os resultados da figura 5.2a, verificou-se um aumento do aproveitamento, enquanto em relação à figura 5.1b, houve um decréscimo da qualidade da solução. Este efeito pode dever-se à forma como a estrutura da procura de vizinhanças está a ser executada, uma vez que, a inserção da estratégia de procura de edifícios comuns foi inserida entre o passo 4 e 5, da figura 3.2, fazendo com que seja criada mais uma hipótese possível de salas. Esta hipótese poderá prevalecer no Caso 5 sobre as outras estratégias abordadas, este esquema pode ser observado na figura 5.3.

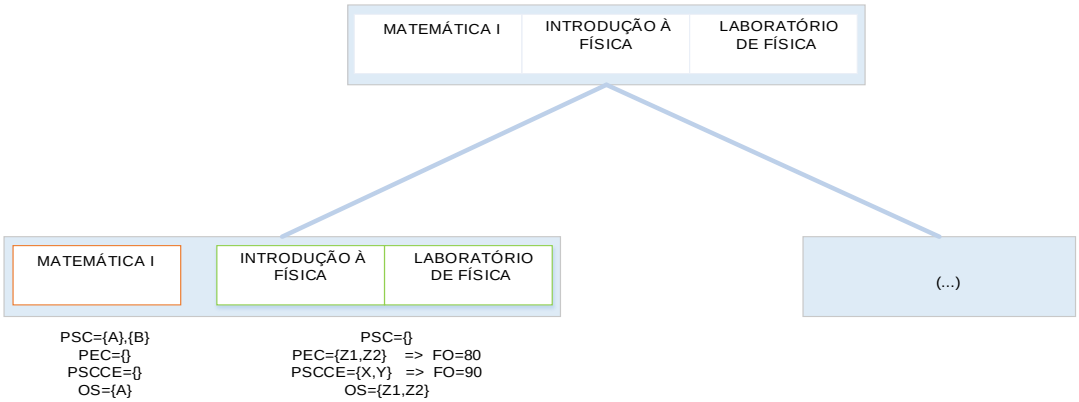


Figura 5.3: Representação esquematizada do possível causa da perda de qualidade da solução, observada no Caso 5. PEC - pesquisa de edifícios comuns. FO - função objetivo.

Na figura 5.3, está representada uma lista de eventos onde a estratégia de combinação de eventos, encontra a melhor sala para o evento de Matemática 1, a sala A, para o evento de Introdução à Física, a sala Z1, e, para o evento de Laboratório de Física, a sala Z2. Esta solução no seu conjunto teria um peso na função objetivo de 130, no caso de se ter escolhido a solução Matemática 1 com a sala A, Introdução à Física com a sala X e Laboratório de Física com a sala Y, a solução na sua totalidade teria um peso na função objetivo de 120, sendo a melhor hipótese a da última combinação de salas, mas esta foi rejeitada o que pode levar a perda da qualidade da solução.
Outra hipótese relacionada com esta problemática é quando os eventos Introdução à Física e Laboratório de Física são alocados no mesmo edifício em conjunto com o de Matemática 1 que decorre noutro edifício, podendo levar a que a restrição do tempo mínimo garantido não seja cumprida. Esta hipótese pode justificar o aumento no número de iterações executadas pelo algoritmo nos Casos 4 e 5.

A estes resultados pode-se reforçar que, com as alterações executadas no algoritmo conseguiu-se não só melhorar os tempos de execução dos diferentes casos, mas também, aumentar o seu aproveitamento. Através da observação dos resultados obtidos (figura 5.4) para a qualidade da solução podemos verificar que houve um aumento desta qualidade quando se realizaram as alterações no código 5.1 em todos os casos estudados. O último algoritmo implementado (Final) teve um aumento generalizado em comparação com o algoritmo inicialmente implementado (Inicial) em todos os casos, mas o mesmo não se verificou quando comparamos estes resultados com os dados obtidos com a implementação 5.1 (Alterado), apesar de se verificar um aumento da qualidade da solução no Caso 1, 2, 3 ,4, no Caso 5 observou-se um decréscimo da qualidade da solução obtida.

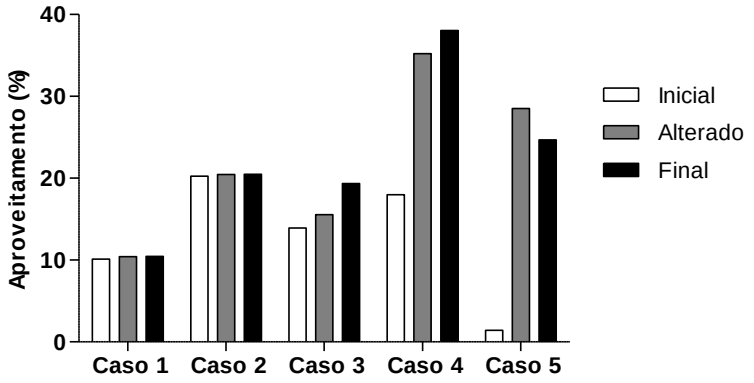


Figura 5.4: Melhoramento da solução em percentagem da penalização da solução inicial. Inicial - corresponde aos dados da implementação inicial 3.5. Alterado - corresponde aos dados da implementação 5.1. Final - corresponde aos dados da implementação 5.2.

Com a observação da figura 5.4, pode-se observar que o Caso 1 e 2, não tiveram grandes ganhos na qualidade da solução, talvez por não ser necessário realizar a trocas entre edifícios, pois, mantêm-se no mesmo edifício. No Caso 3 e 4 houve um aumento nas duas situações me relação ao algoritmo inicialmente implementado, pois conseguiu-se iterar um maior número de horários, levando a uma melhoria da qualidade da solução.

Capítulo 6

Conclusão

Ao longo deste estágio curricular, foi analisado e estudado um algoritmo de atribuição de salas em um problema de horários, onde foram realizadas alterações por forma a torna-lo mais eficiente.

Os objetivos iniciais propostos para a realização do presente trabalho eram: melhorar o tempo de execução e melhorar a qualidade de solução; ambos foram cumpridos. Quanto ao tempo de execução despendido na procura de novas soluções, conseguiu-se melhorá-lo nos casos avaliados. Relativante à qualidade da solução obtida após aplicação do algoritmo, observou-se uma melhoria acentuada na maioria dos casos. Através do cumprimento dos objetivos anteriores conseguiu-se melhorias nos horários iterados, pois atualmente consegue-se realizar mais iterações no processo de melhoramento do que com o algoritmo inicial.
A documentação do algoritmo era outro dos objetivos, esta encontra-se agora devidamente detalhada para facilitar o desenvolvimento de trabalhos posteriores.

6.1 Trabalho Futuro

Para trabalho futuro, para melhorar os resultados obtidos com este algoritmo, poderia criar-se mais estruturas de vizinhança de forma a que estas estejam relacionadas com os objetivos para melhorar a qualidade da solução final.
Outro aspeto a implementar, seria a criação de estruturas mais inteligentes, ou seja, o algoritmo com o auxilio de técnicas de estatística optar por utilizar a estratégia de vizinhança mais indicada para determinados modelos de dados e resultados obtidos. Além disto, outro ponto a considerar é a exploração e melhoramento do restante algoritmo de otimização.

Capítulo 7

Acrónimos

IES	Instituições de Ensino Superior
BTTE	Bullet TimeTabler Education
ECMA	European Computer Manufacturers Association
IDE	Integrated Development Environment
ISO	International Organization for Standardization
MAS	Melhorar a atribuição de salas

Referências

[1] A. Schaerf, “A Survey of Automated Timetabling,” *Artificial Intelligence Review*, vol. 13, no. 2, pp. 87–127, Apr. 1999, Consultado em Março de 2014.

[2] BulletSolutions, “A empresa,” http://www.bulletsolutions.com/, Consultado em Março de 2014.

[3] P. Fernandes, C. S. Pereira, and A. Barbosa, “Bullet TimeTabler Education: a real solution for automatic timetabling in Higher Education Institutions,” Aug. 2013, Consultado em Abril de 2014.

[4] Microsoft, “Visual studio professional with msdn,” http://www.visualstudio.com/en-US/products/visual-studio-professional-with-msdn-vs, Consultado em Agosto de 2014.

[5] E. International, “C# language specification,” *Jun. 2006. Retrieved Jan. 26, 2012*, Consultado em Abril de 2014.

[6] Microsoft, “Stopwatch class,” http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch(v=vs.110).aspx, Consultado em Agosto de 2014.

[7] ——, “List<t>.sort method,” http://msdn.microsoft.com/en-us/library/b0zbh7b6(v=vs.110).aspx, Consultado em Agosto de 2014.

[8] S. Devadas, “Lecture 3: Insertion sort, merge sort,” http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-3-insertion-sort-merge-sort/, May 2010, Consultado em Agosto de 2014.

[9] ——, “Lecture 4: Heaps and heap sort,” http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-4-heaps-and-heap-sort/, 2011, Consultado em Agosto de 2014.

[10] D. Gildea, “Chapter 7: Quicksort,” http://www.cs.rochester.edu/~gildea/csc282/slides/C07-quicksort.pdf, Consultado em Agosto de 2014.

[11] C. Diggins, “The principles of good programming,” http://www.artima.com/weblogs/viewpost.jsp?thread=331531, Consultado em Agosto de 2014.

[12] S. Allen, “C# optimizations,” http://www.dotnetperls.com/optimization, Consultado em Agosto de 2014.