UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

CAROLINE SALIB CANTO

A META-HEURÍSTICA POR COLÔNIA DE FORMIGAS PELO ALGORITMO MIN-MAX ANT SYSTEM APLICADA AO PROBLEMA DE QUADRO DE HORÁRIOS ESCOLAR

CRICIÚMA

2018

CAROLINE SALIB CANTO

A META-HEURÍSTICA POR COLÔNIA DE FORMIGAS PELO ALGORITMO MIN-MAX ANT SYSTEM APLICADA AO PROBLEMA DE QUADRO DE HORÁRIOS ESCOLAR

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientadora: Profa. Dra. Merisandra Côrtes de Mattos Garcia

CRICIÚMA

2018

CAROLINE SALIB CANTO

A META-HEURÍSTICA POR COLÔNIA DE FORMIAS PELO ALGORITMO MIN-MAX ANT SYSTEM AP

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Criciúma, 25 de junho de 2018.

BANCA EXAMINADORA

Orientador: Profa. Dra. Merisandra C. de Mattos Garcia - Orientador

Prof. Dr. Cristian Cechinel (UFSC – Araranguá)

Prof. Me. Luciano Antunes(UNESC)

À minha mãe que me deu todo o apoio e conforto nas horas mais difíceis.

AGRADECIMENTOS

Agradeço aos meus familiares e amigos, que me apoiaram e me confortaram nesta trajetória, tiveram paciência comigo e souberam respeitar os momentos nos quais eu precisei ficar sozinha para focar no desenvolvimento desta pesquisa.

Alguns amigos tiveram um papel importante para que eu pudesse finalizar meu trabalho e merecem um agradecimento especial, como o Giovani, que me cobrou bastante, e me ajudou mais ainda quando precisei, lendo artigos e fazendo vídeo chamadas nos domingos para tentar me ajudar nas partes mais complexas do desenvolvimento do algoritmo.

Agradeço também ao meu amigo Helder, que não me deixou desistir em nenhum momento, e me ajudou bastante com a apresentação final e com a entrega da parte escrita quando eu não tinha como ir até a universidade.

Algumas partes dos estudos sobre ACO foram muito complexas, com muitas equações avançadas e difíceis de entender, e por isto agradeço também a Aline, sem ela eu jamais teria entendido o funcionamento do algoritmo MMAS.

A minha orientadora Meri, que me deu todo apoio quando precisei me afastar da universidade, sendo totalmente flexível quanto as orientações e ainda sim me dando toda atenção necessária, mesmo a distância, para que eu pudesse entregar um trabalho bem escrito e concluir minha pesquisa.

“A ignorância gera mais frequentemente confiança do que o conhecimento: são os que sabem pouco, e não aqueles que sabem muito, que afirmam de uma forma tão categórica que este ou aquele problema nunca será resolvido pela ciência.”

Charles Darwin

RESUMO

A geração de quadro de horários nas escolas é um problema clássico de otimização combinatória que se constitui em um fator crítico de qualidade para qualquer instituição de ensino. O software de gestão escolar i-Educar é um projeto feito em comunidade, utilizado por diversos municípios em todo o Brasil para auxílio na gestão de escolas públicas. Considerando a complexidade na elaboração de grade horária de forma manual e a dificuldade de obtenção de soluções ótimas tem tempo computacional aceitável, o presente trabalho propõe o uso da meta-heurística de otimização por colônia de formigas para gerar quadros de horários com dados do software de gestão escolar i-Educar. Para isto, foi implementada uma API que permite ler os dados do i-Educar, e importar estes para a base de dados do protótipo. Dentre os métodos de colônia de formigas, empregou-se o algoritmo *Min-Max Ant System* para geração da grade horária. Os resultados foram positivos, podendo ser gerado grade horária de qualidade com tempo satisfatório, afirmando então, que método MMAS com busca local é um bom candidato para resolução de problemas de otimização combinatória, podendo gerar bons resultados e com poucas violações das restrições difíceis.

Palavras-chave: Meta-heurística. Inteligência Coletiva. Colônia de formigas. Horário escolar. Min-Max Ant System.

ABSTRACT

The generation of school timetable is a classic problem of combinatorial optimization that constitutes a critical factor of quality for any educational institution. The i-Educar is a community-based software for school management, used by several cities throughout Brazil to assist in the management of public schools. Considering the complexity of manual elaboration and the difficulty of obtaining good solutions, the present work proposes the use of the optimization metaheuristic based on ant colony optimization to generate timetables with data from the i-Educar. For this, an API was implemented that allows reading the i-Educar data and importing these into the prototype database. Among the ant colony methods, the Min-Max Ant System algorithm was used to generate the hourly grid. The results were positive, being able to generate hourly quality grid with satisfactory time, affirming, then, that MMAS method with local search is a good candidate for solving problems of combinatorial optimization, being able to generate good results and with few violations of the restrictions.

Palavras-chave: Meta-heuristic. Collective Intelligence. Ant Colony. Timetable. Min-Max Ant System.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACF	Algoritmos em Colônia de Formiga
ACO	*Ant Colony Optimization*
ACS	*Ant Colony System*
ACT	Autoridade para as Conduções de Trabalho
AE	Algoritmos Evolutivos
AG	Algoritmo Genético
ANCOTT	*Ant Colony Based Timetabling Tool*
API	*Application Programming Interface*
AS	*Ant System*
BWAS	*Best-Worst Ant System*
BWACS	*Best-Worst Ant Colony System*
CE	Computação Evolutiva
EE	Estratégias Evolutivas
GRASP	*Greedy Randomized Adptative Search Procedure*
HCV	*Hard Constrains Violations*
HTTP	*Hypertext Transfer Protocol*
IA	Inteligência Artificial
ITC	*International Timetabling Competition*
LS	*Local Search*
MMAS	*Min-Max Ant System*
PCV	Problema do Caixeiro Viajante
PE	Programação Evolutiva
PG	Programação Genética
PCMV	Problema de Cobertura Multi-Veículo
PRV	Problema de Roteamento de Veículos
PSO	*Particle Swarm Optimization*
RBAS	*Rank-Based Ant System*
RNA	Redes Neurais Artificiais
SC	Sistemas de Classificação
SCV	*Soft Constrains Violations*
UCTP	*University Course Timetabling Problem*
VNS	*Variable Neighborhood Search*

SUMÁRIO

1 INTRODUÇÃO

A inteligência e a forma como os seres de cada espécie pensam tem sido um dos principais objetos de estudo na área tecnológica durante milhares de anos. A área de Inteligência Artificial (IA) nasceu logo após a segunda guerra mundial, no ano de 1956, e tenta não apenas compreender, mas também construir entidades inteligentes, baseando-se na forma como cada indivíduo pensa (RUSSEL; NORVIG, 2013). A Inteligência Computacional (IC), denominada originalmente de Inteligência Artificial (IA), vai além da perspectiva de compreensão do pensamento, pois também procura construir entidades artificiais inteligentes. A estratégia geral adotada pela área de IC é aplicar técnicas de aproximação e um método de busca que possa gerar (parcialmente) uma solução otimizada para um dado problema (KRUSE et al, 2013, tradução nossa).

A meta-heurística, conforme colocado por Santos (2016), é um método utilizado para a resolução de problemas de otimização que geralmente não podem ser resolvidos por heurísticas tradicionais. A meta-heurística tem como característica diversificar o campo de soluções, onde ao encontrar um ótimo local é realizado um processo de afastamento para buscar uma nova solução em outro local, como uma combinação de escolhas aleatórias e resultados encontrados anteriormente (BARBOSA, 2014). Uma otimização, do ponto de vista computacional, é um conjunto de praticas que auxiliam na busca de melhores caminhos para montar soluções e resolver problemas de interesse para os seres humanos (OLIVEIRA, 2013). Segundo Lobo (2005), denomina-se um problema de otimização, quando dentre todas as soluções possíveis, é considerada a de menor ou maior valor da função objetivo, variando-se no caso de problemas de minimização ou maximização. De acordo com Serapião (2009), nos últimos tempos, algoritmos inspirados em inteligência coletiva, vem sendo utilizados para resolver problemas de busca e otimização nas quais as abordagens tradicionais não conseguem encontrar boas soluções.

A inteligência coletiva, também conhecida como inteligência de enxames ou inteligência de colônias, de acordo com Bembem e Santos (2013), visa o reconhecimento de habilidades distribuídas a um grupo de indivíduos. Algoritmos baseados em inteligência coletiva e meta-heurísticas vêm sendo utilizados para

resolver diversos problemas de busca e otimização, onde as abordagens tradicionais, como programação matemática, têm dificuldades em solucionar. Estes algoritmos tendem a serem conduzidos de uma forma evolutiva, inicia-se na obtenção da população inicial, e então se aplica métodos de busca local para melhorar a solução, considerando que os indivíduos são evoluídos e que haja troca de informações entre eles. Este processo conduzirá os indivíduos em direção a uma solução ótima (SERAPIÃO, 2009).

Um dos problemas de otimização combinatória que pode ser estudado em conjunto com meta-heurísticas e inteligência coletiva, é a elaboração de quadro de horários, problema conhecido pelo nome de *Timetabling Problem*. No Brasil, a elaboração de quadro de horários nas escolas se da a todo início de ano letivo, dando trabalho aos gestores escolares na hora de executar esta tarefa de forma que haja qualidade em seus resultados (ALVES, 2010). Muitas vezes, para realização desta tarefa, gestores tem o auxilio de algum software de gestão escolar, como por exemplo o i-Educar.

O i-Educar é um software público de gestão escolar que pode ser utilizado, modificado e distribuído livremente por qualquer pessoa ou organização. Por ser um software livre e consequentemente gratuito, além dos benefícios já disponíveis dentro da plataforma, o uso do software gera economia, como no caso citado por Brasil (2015) onde a prefeitura do município de Monte Alegre do Rio Grande do Norte economizou cerca de R$ 2,4 milhões no período de um ano com o uso desta ferramenta. Uma característica interessante, e relacionada diretamente a pesquisa que será realizada neste trabalho**,** é que o software possui um módulo para geração de quadro de horários, no entanto isto é realizado manualmente e não de uma forma automatizada.

A elaboração de grade horária feita de forma manual gera desperdício do tempo dos gestores e nem sempre dispõe de resultados satisfatórios, o que pode ser prejudicial para o rendimento dos alunos, pois pode ocorrer sobrecarga de aulas acumuladas, ou pode ocorrer a necessidade de deslocamento para salas muito distantes (ALVES, 2010; NEUKIRCHEN, 2015).

Além do problema de geração de quadro de horários para escolas (*School Timetabling*), que será o foco de estudo desta pesquisa, também são conhecidos os problemas de geração de quadro de horários para universidades

(*University Timetabling*) e geração de quadro de horários para exames (*Examination timetabling*). O desenvolvimento de métodos computacionais para solucionar estes tipos de problemas atrai a atenção de pesquisadores de diversas áreas da comunidade científica (NEUKIRCHEN, 2015). Um dos temas que vem sendo bastante pesquisado para resolução do problema de *timetabling*, é o uso de meta-heurísticas, como na pesquisa de Melo (2003) e Mikuska (2015), que utilizaram Algoritmo Genético (AG); Tilahun e Ong (2015) que criaram um algoritmo de otimização chamado “Prey-predator Algorithm” baseado em técnicas meta-heurísticas e Patrick e Godswill (2016) que empregaram a otimização por colônia de formigas.

O algoritmo de otimização por colônia de formigas, do inglês *Ant Colony Optimization* (ACO), é uma meta-heurística utilizada em problemas de otimização combinatória. Esta técnica foi introduzida por Dorigo et al. (1992) como *Ant System* (AS), porém ainda não fazia o uso de busca local. Mais tarde, Dorigo e Gambardela (1997) desenvolveram a primeira versão do ACO, com o propósito de buscar soluções para o problema do caixeiro viajante (PESSOA, 2015). O algoritmo ACO e a técnica AS foram baseados no comportamento real das formigas quando estão em busca coletiva de fontes de alimento. Segundo Pessoa (2015) as formigas que viajam pelo caminho mais curto retornam à colônia mais rapidamente, desta forma o trajeto percorrido contém uma maior concentração de feromônio. Essa trilha atrai as outras formigas e com o tempo todos os indivíduos da colônia passam a trilhar pelo mesmo caminho, tornando-o otimizado e mais curto (ALLEGRINI; OLIVIERI, 2011). O algoritmo desenvolvido por Dorigo e Gambardela é justificado pela distribuição de *x* formigas em *y* locais, de forma que cada formiga percorra um caminho que passe apenas uma única vez pelo local (PESSOA, 2015). De acordo com Razan (2014), o algoritmo tem duas etapas principais, uma destinada à atualização da trilha de feromônio e a outra à comparação dos modelos gerados.

Sendo assim, esta pesquisa propõe encontrar um modelo dentre os estudos de otimização por colônia de formigas que resolva o problema da geração de quadro de horários de forma automática. Para isto, será realizado o estudo sobre a meta-heurística com foco em inteligência coletiva, e também a implementação do algoritmo de otimização *Min-Max Ant System (MMAS)* baseado na colônia de formigas. Os dados que serão utilizados neste protótipo serão coletados por meio de

uma API que será empregada no Software Público de Gestão Escolar i-Educar.

1.1 OBJETIVO GERAL

Desenvolver o algoritmo de otimização baseado em colônia de formigas *Min-Max Ant System* para geração de quadro de horários escolares.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) descrever o problema de geração de quadro de horários, inteligência coletiva e meta-heurística baseada em colônia de formigas;
b) empregar API no i-Educar que retorne os dados das instituições de ensino que utilizam o software;
c) aplicar o algoritmo de otimização por colônia de formigas, *Min-Max Ant System*, para geração de grade horária escolar;
d) gerar quadro de horários de forma automática para municípios que utilizem o Software Público de Gestão Escolar i-Educar.

1.3 JUSTIFICATIVA

As soluções para problemas de alocação de horas são poucas, visto a sua complexidade e as restrições, sendo difícil determinar as soluções que minimizam os custos, atendem às preferências dos funcionários, distribuam turnos de forma equitativa e que satisfaçam as restrições no local de trabalho (ERNST et al., 2004, tradução nossa). A literatura direciona-se a abordagens meta-heurísticas para resolução de problemas de alocação de horas. Embora soluções por meio de meta-heurística possam não garantir um resultado ótimo, Santos (2016) afirma que geralmente produzem uma boa solução, viável para a maioria dos dados de entrada em um limitado tempo de execução. Na prática, existem muitos problemas de otimização que necessitam de uma atenção especial, e em consequência, tem-se vários métodos e algoritmos para a resolução desses problemas. Algoritmos meta-

heurísticos são uma boa escolha para resolver problemas de otimização (ERNST et al., 2004, tradução nossa, TILAHUN; ONG, 2015, tradução nossa).

Há décadas, de acordo com Oliveira, Vianna e Vianna (2012), a elaboração de quadro de horários tem sido um problema de otimização abordado nas pesquisas, isso devido à dificuldade de obtenção de soluções ótimas em tempos computacionais aceitáveis e também ao fato de que as instituições de ensino enfrentam dificuldade na elaboração de boas grades de horários, considerando os requisitos pedagógicos, pessoais e institucionais envolvidos. Algumas destas pesquisas são as de Neukirchen (2015) sobre geração de quadro de horários para os cursos de computação da UFRGS; Santos (2016) que estudou sobre a alocação de horários dos funcionários que satisfaça as leis trabalhistas de uma loja varejista; Patrick e Godswill (2016) que utilizaram a otimização por colônia de formigas para resolver o problema de grade horária em cursos universitários; Soria-alcaraz et al. (2014) que empregaram hiper-heurística para resolução de problemas de grade horária para cursos em instituições educacionais, e também nos trabalhos realizados na UNESC, como o de Melo (2003) que avalia um sistema de geração de grade horária que utiliza o método de busca por algoritmo genético e o de Rosa (2015) que compara o algoritmo de busca e satisfação de restrições e o algoritmo genético em um sistema para geração de horário escolar. Visto que existe um número considerável de pesquisas que tenta resolver o problema de grade horária, pode-se afirmar que mesmo sendo um contratempo antigo, ainda hoje se procura por métodos para solucionar o problema de forma automatizada.

O Software Público de Gestão Escolar i-Educar é um exemplo de software que gera manualmente o quadro de horários, utilizando-se dos dados da instituição que já foram cadastrados no sistema. Esta ferramenta é usada por diversos municípios do Brasil, em função de ser de origem livre, o que permite que qualquer um possa modificá-lo de acordo com suas necessidades. Os trabalhos existentes que visam resolver o problema de alocação de horários para escolas, em geral, se propõem a atender um problema específico, sendo difícil adaptá-lo em outra instituição (OLIVEIRA; VIANNA; VIANNA, 2012). Elaborado de maneira manual, o quadro de horários criado no i-Educar é relevante, como informado por Brasil (2015), que apresenta o caso do município de Monte Alegre no Rio Grande do Norte, que após a elaboração manual do quadro de horários realizada no i-Educar, conseguiu

elevar a taxa de aproveitamento dos recursos do quadro de professores e horas de aula para 95%, e com isto um ganho potencial de economia em 22% na folha de pagamento. Mesmo com o ganho que um município pode ter montando seu quadro de horários de forma manual, um processo automatizado com uso de algum algoritmo de otimização poderia elevar estes números.

A otimização por colônia de formigas, do inglês *Ant Colony Optimization* (ACO), tem alcançado resultados consistentes para problemas de otimização, como *timetabling problem* (problema de alocação horária) (DORIGO; MANIEZZO; COLORNI, 1996, tradução nossa). Dorigo, Birattari e Stutzle (2006) contam que a ACO tem sido aplicada com sucesso para a resolução de vários problemas NP-completos e NP-difíceis[1] de otimização combinatória. A inteligência coletiva, em especial nesta pesquisa, a da colônia de formigas, tem sido base para o estudo e criação de novas técnicas computacionais que auxiliam na busca de soluções para problemas de otimização. Alguns exemplos de sua aplicabilidade são os trabalhos de Patrick e Godswill (2016) que utilizaram ACO como técnica de otimização para *University Timetabling*, ou o de Ghellere (2012), em que aplica o algoritmo *Standard Ant Clustering Algotithm* na *Shell Orion Data Mining Engine* para o agrupamento em *data mining*, e também o de Pessoa (2015) que utilizou ACO para desenvolvimento de modelos quimiométricos[2].

A variação de ACO dada pelo algoritmo *Min-Max Ant System* (MMAS) mostra-se como uma boa escolha quando se trata de problemas de *timetabling,* pois obteve bons resultados em pesquisas que visavam resolver o problema de geração de quadro de horários para universidades, como nos trabalhos de Hicks, Pongcharoen e Thepphakorn (2014); Knowles, Sampels e Socha (2002), e Ugat et al. (2018). Como o problema da presente pesquisa é semelhante ao dos trabalhos citados, acredita-se que o algoritmo de MMAS é uma boa opção para ser aplicado neste contexto.

---

[1] Problemas denominados NP-difíceis são aqueles em que não podem ser resolvidos de forma ótima dentro de um tempo computacional aceitável, e que por tanto, para obter boas soluções deve-se utilizar métodos de aproximação que retornam soluções quase ótimas em um tempo relativamente curto (DORIGO; STÜTZLE, 2004).

[2] Quimiometria é o termo utilizado para o ato de aplicar métodos matemáticos com dados de origem química (ADAMS, 2004).

1.4 ESTRUTURA DO TRABALHO

Este trabalho é dividido em oito capítulos. O primeiro capítulo apresenta uma breve descrição dos assuntos que serão abordados neste trabalho, podendo tomar conhecimento dos objetivos bem como a justificativa da realização desta pesquisa.

No segundo capítulo, é realizado um estudo sobre meta-heurística e algumas das técnicas já existentes, que são utilizadas para obtenção de resultados satisfatórios em diferentes tipos de problemas de otimização. São citados alguns exemplos de problemas comuns de otimização como atestado médico e o jogo de xadrez, onde não se é possível obter um resultado exato, mas uma solução com um custo computacional baixo ou uma taxa de falhas reduzidas são consideradas como um resultado aceitável.

O terceiro capítulo aborda a Computação Natural, explicando a computação evolutiva, inteligência coletiva e o método de meta-heurística escolhido para esta pesquisa, a otimização por colônia de formigas, que se baseia no comportamento de formigas reais quando em busca do seu alimento.

Referente ao quarto capítulo, é apresentado o método de otimização por colônia de formigas, contando de forma breve a história de criação deste método, a evolução dele conforme novas pesquisas foram surgindo e ilustrando como o uso da técnica pode ser utilizado e aplicado em problemas de otimização com um mínimo esforço computacional e obtendo bons resultados. O algoritmo *Min-Max Ant System* também é abordado neste capítulo, mostrando-se como uma boa opção para resolução de problemas de *timetabling* quando utilizado em conjunto com técnicas de busca local.

O quinto capítulo compreende o problema de alocação de quadro de horários nas escolas, bem como a complexidade e o custo computacional deste tipo de problema. Somando-se a isso é fala-se do problema de alocação de quadro de horários, já recorrente no Software Público de Gestão Escolar i-Educar.

Na parte seis são apresentados os trabalhos correlatos que são relacionados com os assuntos abordados nessa pesquisa, visando tornar claro a relevância científica dos assuntos abordados nesta pesquisa.

O capítulo sete, fala sobre o trabalho desenvolvido e a metodologia

utilizada para o desenvolvimento do trabalho, bem como os recursos necessários para implementação. Nesta mesma seção são realizados a apresentação e discussão dos resultados, tendo a reprodução dos testes e conclusão em cima dos resultados obtidos. Com os resultados obtidos, também é realizado a discussão destes com relação aos da literatura.

Por fim, no capítulo oito, tem-se a conclusão da pesquisa e propostas para trabalhos futuros.

2 PROBLEMAS DE OTIMIZAÇÃO E O USO DE META-HEURÍSTICA

A ciência da otimização visa encontrar os extremos, máximo ou mínimo, de uma determinada função. Os extremos são conhecidos por determinar a melhor maneira em que o sistema deve ser construído, no sentido de que o mesmo deve reproduzir um custo mínimo, máxima eficiência energética, ou mínima taxa de falhas. Em um contexto prático, o uso da otimização ajudará a determinar as melhores configurações possíveis para a construção de sistemas que consistem em solucionar problemas de interesse dos seres humanos como, por exemplo, custos minimizados de fabricação de um determinado produto, otimização da rota que será utilizada por algum veículo para entregas de carga, ou até mesmo a elaboração de grade horária que, de acordo com as suas restrições, pode ser algo complexo e com tempo indeterminado para uma tomada de decisão (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013; NEUKIRCHEN, 2015; OLIVEIRA, 2016).

> Considera-se o problema de otimização combinatória de minimização 1 dado por P = (S, Ω, f), onde: S é o conjunto de soluções candidatas; Ω é um conjunto de restrições; e f é a função objetivo que associa um custo f (s) a cada solução candidata $s \in S$. O objetivo é encontrar uma solução $s* \in S$ globalmente ótima, i.e., que seja factível (respeita todas as restrições em Ω) e tenha o menor custo dentre todas as soluções candidatas (MULATI; CONSTANTINO; SILVA, 2013).

Mesmo em problemas distintos, o uso de otimização pode aperfeiçoar a entrega de resultados, podendo ser diferenciado por meio do método adotado para ser implementado. A escolha do método é fundamental para obter um melhor resultado, já que cada um pode trazer conclusões diferentes. Segundo Oliveira (2016), esta escolha depende de uma série de características do problema que está sendo analisado, podendo levar ao sucesso ou insucesso do estudo, visto que uma solução pode ser ideal para um tipo de problema e não ser adequada para outro.

Segundo Goldbarg, Goldbarg e Luna (2016), uma heurística é uma técnica da computação que visa buscar uma solução aceitável utilizando um esforço computacional razoável, podendo proporcionar eficiência computacional no desempenho do algoritmo, sendo o tempo de tamanha importância, tolera-se a possibilidade de a solução encontrada não ser ao todo ótima. Luger (2013) apresenta alguns exemplos de problemas no qual se faz necessário o uso de métodos de heurísticas para resolução, como o problema do diagnóstico médico e

do jogo de xadrez:

a) **diagnóstico médico:** nem sempre é possível se obter uma solução exata, por causa de ambiguidades ao formar-se o problema nos dados disponíveis. Os médicos usam heurística como auxílio ao formular um plano de tratamento, pois um dado conjunto de sintomas pode ter várias causas possíveis. A visão também é um exemplo de um problema heurístico, pois cenas visuais são frequentemente ambíguas, permitindo múltiplas interpretações, como no caso da ilusão de ótica;
b) **jogo de xadrez:** é possível ter uma solução exata, porém o custo computacional pode ser impeditivo. Isto se deve ao crescimento do espaço de estados ser combinacionalmente excessivo com o número de estados possíveis aumentado exponencialmente, ou factorialmente, com a profundidade da busca. Nestes casos as técnicas de busca convencionais por profundidade ou em amplitude podem não conseguir encontrar a solução em um intervalo de tempo justo, enquanto as heurísticas enfrentam esta mesma complexidade de forma diferente, com uma busca guiada pelo caminho mais promissor, desconsiderando os estados menos promissores e seus descendentes. Um algoritmo heurístico pode enfrentar essa complexidade combinatória e encontrar uma solução mais aceitável e, por fim, mais rápida.

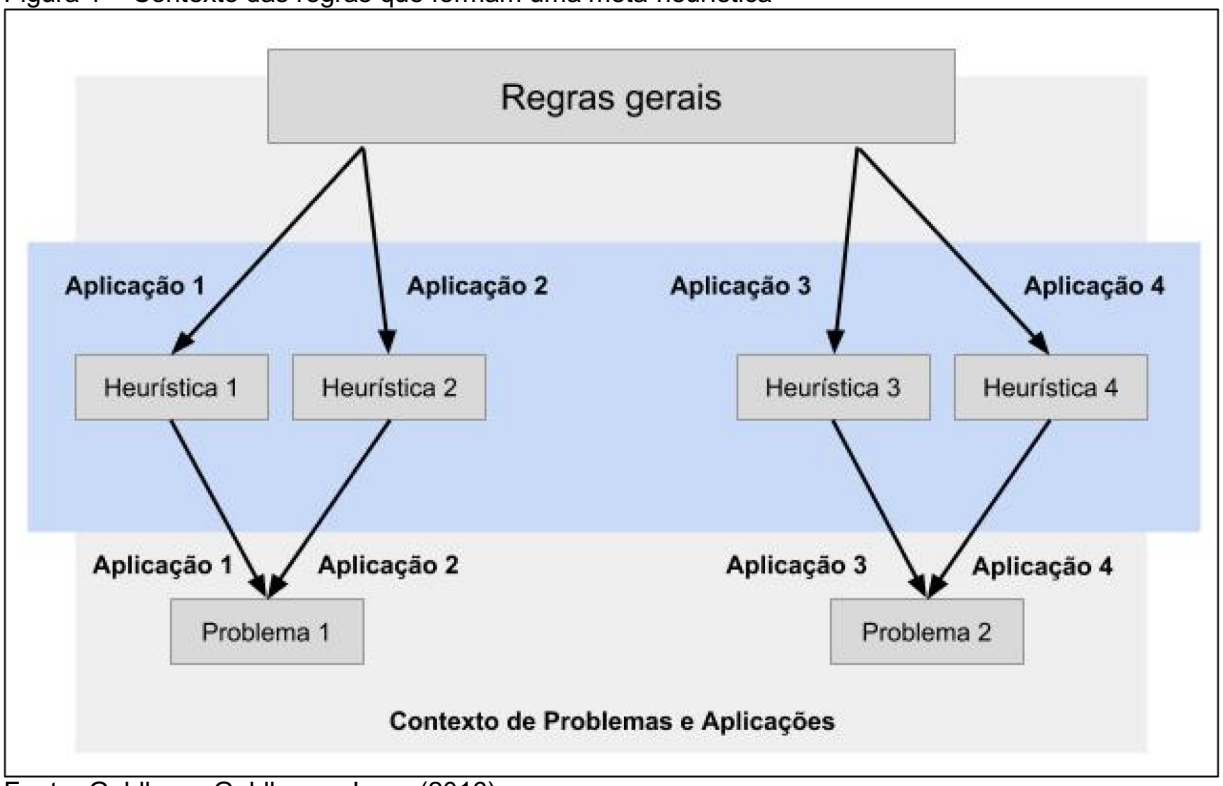
O desenvolvimento de procedimentos heurísticos deu-se devido à necessidade de resolver problemas de otimização sem dispor de garantias teóricas ou do alcance de uma solução exata, tão pouco uma proximidade a esta. Neste sentido, a lógica da heurística não é de que a solução obtida seja melhor ou superior a vasta maioria das soluções, mas de que a mesma seja uma boa solução (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

Com os vastos problemas de otimização, às vezes faz-se necessário o uso de técnicas mais complexas, que usam não apenas um método heurístico, mas uma combinação deles, desenvolvendo então uma estratégia mais completa e com maior eficiência na exploração do espaço de busca, e para isto, utiliza-se o método da meta-heurística.

A meta-heurística é definida como um conjunto de conceitos algorítmicos que podem ser aplicados por meio de algoritmos heurísticos para resolução

satisfatória de diferentes tipos de problemas NP-difíceis. Meta-heurísticas também são boas alternativas para resolução de problemas de otimização em um tempo computacional razoável. De modo geral o uso destas estratégias computacionais não garante encontrar uma solução ótima, mas estatísticas mostram que elas têm alcançado soluções de qualidade com tempo computacional aceitável (KOULINAS; KOTSIKAS; ANAGNOSTOPOULOS, 2014, tradução nossa; MULATI; CONSTANTINO; SILVA, 2013). Uma vantagem em relação ao uso de técnicas meta-heurísticas, é que sua arquitetura, quando formada a partir de um tema comum, pode servir de base para vários projetos. De acordo com Goldbarg, Goldbarg e Luna (2016), a arquitetura geral para uma meta-heurística é capaz de guiar ou inspirar a criação de algoritmos heurísticos para diferentes situações, aplicações e problemas. Na figura 1, dentro de uma abordagem meta-heurística (regras gerais) é possível derivar mais de uma heurística para mais de um problema.

Figura 1 – Contexto das regras que formam uma meta-heurística



Fonte: Goldbarg, Goldbarg e Luna (2016).

As regras que compõem o contexto de uma meta-heurística muitas vezes são baseadas em fenômenos naturais, como por exemplo, as meta-heurísticas bioinspiradas que derivam suas regras para formação de heurísticas que

reproduzem computacionalmente processos biológicos. O uso de heurísticas inspiradas no ramo da computação natural, baseados em populações e na adaptação dos seres vivos, conforme observado na natureza, são populares para resolver problemas de busca e otimização, obtendo inclusive melhor desempenho do que projetos que seguem abordagens tradicionais como a programação matemática (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013; GOLDBARG; GOLDBARG; LUNA, 2016; SERAPIÃO, 2009).

3 COMPUTAÇÃO NATURAL

A computação natural é uma área de pesquisa relacionada ao subcampo da inteligência artificial denominado Aprendizado de Máquina (AM) que se dedica ao desenvolvimento de técnicas computacionais que permitem ao computador aperfeiçoar seu desempenho em determinada tarefa por meio de aprendizado. A computação natural deu origem a várias técnicas bem-conceituadas na área da computação como, por exemplo, redes neurais artificiais[3], computação quântica[4], autômatos celulares[5], geometria fractal[6] e também a computação evolutiva (FACELI et al, 2011).

3.1 COMPUTAÇÃO EVOLUTIVA

A Computação Evolutiva (CE) é uma área da computação bioinspirada que se baseia nos conceitos de computação natural. Ela se inspira em processos que ocorrem na natureza para desenvolvimento de algoritmos que podem ser utilizados em problemas reais (FACELI et al, 2011; GOLDBARG; GOLDBARG; LUNA, 2016).

Goldbarg, Goldbarg e Luna (2016) citam como características condicionais da computação evolutiva:

a) ser realizada por meio de um processo iterativo;

b) basear-se em uma população;

c) possuir internamente uma arquitetura de processamento paralelo;

d) corresponder a um processo de busca guiada;

e) empregar os princípios de Darwin de seleção natural de acúmulos de variações genéticas.

---

[3] As redes neurais artificiais são uma abstração das redes neurais biológicas, e seu funcionamento é semelhante à forma como os seres humanos buscam solucionar problemas (ENGELBRECHT, 2007, tradução nossa).

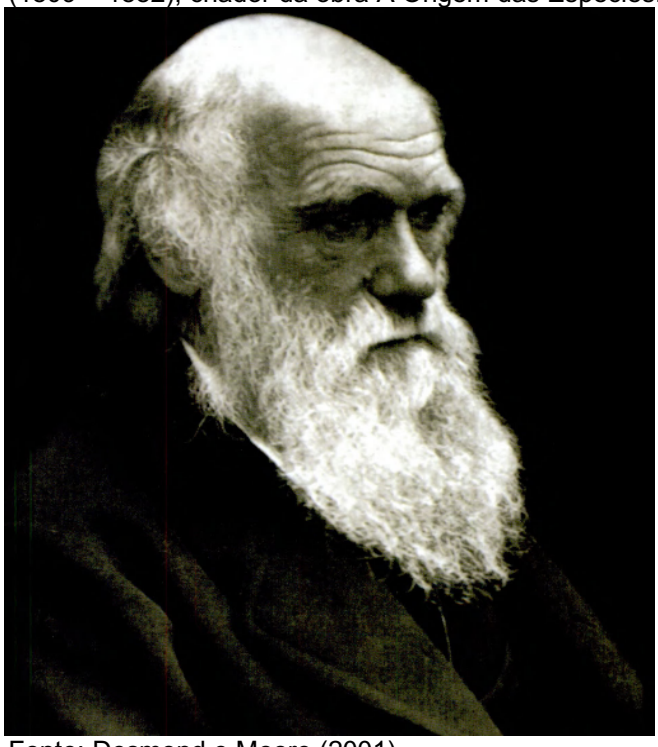
[4] A computação quântica é uma ciência que estuda a mecânica quântica na Ciência da Computação (MATTIELO et al., 2012).

[5] Autômatos celulares são sistemas evolutivos formados por uma rede de células, onde cada uma pode estar em um número finito de estados que variam de acordo com as regras determinísticas PEIXOTO; BARROS, 2004).

[6] A geometria fractal é um campo da matemática que estuda o comportamento dos fractais, conhecido como figuras geométricas não-euclidianas (ASSIS et al., 2008).

Até a chegada do século XIX a única teoria aceita sobre as diferentes espécies de vegetais e animais na terra era o criacionismo, que dizia que todos os animais e vegetais eram criados por Deus. No entanto, o criacionismo não continha um esclarecimento que fosse pertinente em relação a extinção de algumas espécies e aparecimento de outras. Para complementar a teoria do criacionismo sem entrar em conflito com a Bíblia, surgiu então uma nova teoria denominada “catastrofismo”, que condizia com a teoria do criacionismo e acrescentava os relatos de catástrofes naturais como explicação para as mudanças entre as espécies (DARWIN, 1859; PILA, 2015). Em 1859, o cientista inglês Charles Darwin (figura 2) publicou sua famosa obra “A origem das espécies”, afirmando que os indivíduos de uma população são diferentes, e devido as suas diferenças, alguns deles são mais adaptáveis ao ambiente do que os outros, e com isso, têm mais chances de sobreviver e se reproduzir. Estas características vantajosas são herdadas pelos indivíduos das próximas gerações, tornando-se através do tempo características predominantes na população. A obra de Darwin é referência para estudos e entendimento do evolucionismo, fazendo com que esta teoria tenha ficado conhecida também pelo nome de *Darwinismo*.

Figura 2 - Cientista inglês Charles Robert Darwing (1809 – 1882), criador da obra A Origem das Espécies.



Fonte: Desmond e Moore (2001).

A teoria da evolução dita por Darwin não se aplica apenas a organismos biológicos, mas sim para todos os tipos de organismos que se reproduzem de forma que envolva herança e viabilidade. Então entende-se que a forma de seleção natural poderia ocorrer de forma não-biológica, conforme se estuda na Computação Evolutiva. Segundo Gaspar-Cunha, Takahashi e Antunes (2013), nos últimos 40 anos a ideia de construir heurísticas inspiradas no mecanismo de adaptação dos seres vivos foi se afirmando gradativamente, conforme o número de respostas adequadas para problemas mais complexos foi aumentando. A consequência do uso destas heurísticas bioinspiradas foi a produção em coletivo de soluções para problemas de adaptação, baseada em ações simples entre os indivíduos que interagem entre si e com o meio ambiente.

A evolução das espécies em seu ambiente natural pode acontecer por meio de processo evolutivo, busca por alimento (coordenada por colônia de formigas ou bando de pássaros), ou também por resposta do sistema imunológico de mamíferos à invasão de agentes infecciosos. Estes formatos de evolução envolvem de forma essencial a realização de buscas com um amplo espaço de alternativas e tarefas que não apenas são de elevada complexidade, mas que também mudam com o passar do tempo, exigindo uma contínua adaptação de seus mecanismos. As soluções para problemas de adaptação dos seres que habitam o planeta tratam-se de mecanismos muito bem-sucedidos, pois foi o que permitiu aos seres evoluírem e se manterem, após bilhões de anos de luta pela sobrevivência. Por esses motivos, acredita-se que a estrutura de tais mecanismos naturais de adaptação possa ser transposta com sucesso em técnicas computacionais de adaptação destinadas a tratar problemas de otimização de alta complexidade (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

A computação evolutiva é uma área ampla e que envolve os Algoritmos Genéticos, Programação Genética, Estratégias Evolutivas, Programação Evolutiva e os Sistemas de Classificação. Cada um desses representa seus indivíduos e suas respectivas funções de forma singular, mas de acordo com a teoria de Darwin, sendo a base semelhante para todas. Embora as áreas sejam diversas, não há uma singularidade a respeito da aplicabilidade de forma que não haja uma junção dessas técnicas, ou seja, quando se fala de computação evolutiva, as aplicações tendem a

ser modelos híbridos, mas que consideram o melhor que cada conceito pode oferecer (FACELI et al., 2011; PILA, 2015).

Além da computação evolutiva, existem outras heurísticas que podem ser mais adequadas para algumas classes de problemas. A compreensão de novas heurísticas baseadas em processos da natureza foi o que inspirou outros algoritmos da computação natural com técnicas diferentes aos da evolução das espécies. Um exemplo disto é o algoritmo baseado em colônia de formigas, que adota o mecanismo da inteligência coletiva (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

3.2 INTELIGÊNCIA COLETIVA

A inteligência coletiva, do inglês *Collective Intelligence*, é um ramo da Computação Natural[7] que aborda uma forma de inteligência que surge no contexto de populações de indivíduos que podem se comunicar e agir com um razoável grau de autonomia. A inteligência coletiva também é conhecida na área da computação como inteligência de enxames ou inteligência de colônias (GOLDBARG; GOLDBARG; LUNA, 2016).

Indivíduos de uma colônia se organizam de forma relativamente simples, onde o principal aspecto de seu comportamento é copiar a ação de seu vizinho quando o mesmo obteve sucesso. Com base nesta conduta, a colônia consegue descobrir ótimos lugares ou regiões dentro de um espaço de busca de alta dimensão (ENGELBRECHT, 2007, tradução nossa).

Algoritmos baseados em inteligência coletiva geralmente manipulam indivíduos simples que atuam de forma auto-organizada. Millonas (1993) explica que os sistemas baseados em inteligência coletiva seguem cinco princípios:

a) **proximidade:** os indivíduos da população devem interagir entre si;

b) **qualidade:** os indivíduos devem ser capazes de avaliar a interação entre si e com o ambiente;

c) **diversidade:** o sistema deve ser capaz de reagir a ações inesperadas;

[7] A computação natural procura identificar e replicar funções dos processos naturais da vida (GOLDBARG; GOLDBARG; LUNA, 2016).

d) **estabilidade:** modificações no ambiente nem sempre deverão fazer com que haja uma adaptação do indivíduo;

e) **adaptabilidade:** os indivíduos devem ser capazes de se adaptarem a possíveis mudanças no ambiente e também da população.

Com o passar dos anos, verificou-se, pelas pesquisas realizadas, que algumas heurísticas podem ser mais específicas que outras dependendo da classe do problema a ser estudado. Isso motivou o estudo da computação natural para resolução de problemas ir além da evolução das espécies, sendo explorado outros processos da natureza mais relevantes para tratar de problemas com características específicas, tais como o algoritmo *shuffled frog-leaping*, que se inspira em um grupo de sapos pulando no pântano em busca de comida; o algoritmo de colônia de abelhas, baseado na organização social das abelhas; o algoritmo de otimização de enxame de partículas, do inglês *Particle Swarm Optimization* (PSO), motivado pelo comportamento social dos rebanhos de aves, e a otimização por colônia de formigas, inspirado no comportamento das formigas a procura de alimentos (ENGELBRECHT, 2007, tradução nossa; GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013; LINS, 2012; SERAPIÃO, 2009). Para esta pesquisa, será estudada a otimização por colônia de formigas.

3.3 OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

As colônias de formigas serviram como inspiração para o desenvolvimento de diversas meta-heurísticas, dentre elas a mais estudada e de maior sucesso é uma técnica chamada Otimização por Colônia de Formigas, do inglês *Ant Colony Optimization* (ACO). A técnica de otimização ACO surgiu da observação do comportamento de formigas reais, que apesar da pouca visão, conseguiam uma rota curta quando em busca do alimento. O principal aspecto do comportamento é a comunicação que ocorre entre as formigas da colônia por meio do depósito de feromônios[8] nas trilhas percorridas (DORIGO et al., 2006; FACELI et al., 2011; MULATI; CONSTANTINO; SILVA, 2013).

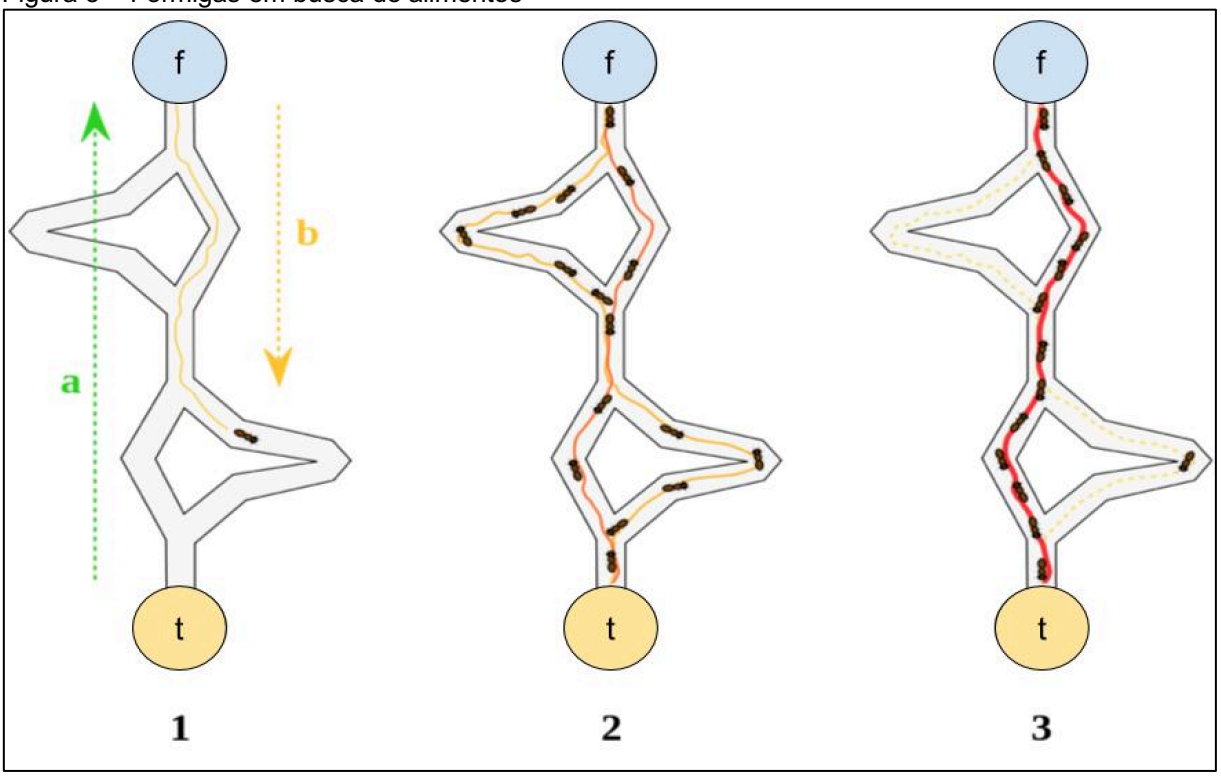
Uma colônia de formigas é organizada em grupos com funções distintas e

[8] O feromônio é uma substância química cujo odor é uma forma de sinalização, no caso das formigas, sinaliza o rastro para a fonte de comida (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

que variam de acordo com seu tamanho, como proteger a colônia, buscar por alimentos ou transportá-los. Durante a busca por alimentos, as formigas exploram de maneira aleatória uma determinada região e ao longo de sua trajetória deixam rastros de feromônios. Estes rastros criam inúmeras rotas que podem ser renovadas regularmente, dependendo do caminho que as formigas percorrem. Desta forma se uma outra formiga encontrar uma rota melhor, os feromônios da rota antiga desaparecerão gradativamente, fazendo com que ao longo do tempo esta rota sofra uma evaporação natural, e a nova rota ganhe uma concentração maior do feromônio, tornando-se então a escolha mais popular entre as formigas. Como as formigas que encontraram o melhor caminho devem voltar antes das que escolheram um caminho pior, então é provável que tenha uma concentração maior de feromônio no menor caminho e, provavelmente, será o caminho seguido pelas próximas formigas (BLUM, 2005; COPPIN, 2010; FACELI et al., 2011; GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013). A evaporação do feromônio ocorre de forma lenta, significando que uma vez depositado em diferentes regiões, estas poderão ser exploradas (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

Na figura 3 é possível visualizar o comportamento das formigas na busca do alimento, entre duas rotas (a, b), o rastro de feromônio é mais forte onde as formigas tendem a passar mais vezes, independente da rota, e portanto, ao longo do tempo o caminho com mais ferômonio se torna a rota mais utilizada pelas formigas.

Figura 3 – Formigas em busca de alimentos



Fonte: Gaspar-cunha, Takahashi e Antunes (2013).

Na computação, a meta-heurística ACO é considerada um método de busca construtivo, onde uma população, neste caso a colônia de formigas, constrói soluções de forma cooperativa para determinado problema de otimização. As formigas guardam em sua memória a informação das experiências anteriores, que podem ser computacionalmente convertidas em matrizes ou grafos contendo informações heurísticas (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013).

O processo da auto-organização[9], de acordo com Goldbarg, Goldbarg e Luna (2016), pode ser usado para descrever como funciona a inteligência coletiva na área da meta-heurística, tendo como partes do processo de auto-organização (figura 4):

a) ***feedback* positivo:** são regras do comportamento que contribuem para a criação de estruturas do sistema, como por exemplo, com recrutamento e reforço. O *feedback* positivo pode gerar uma tendência

[9] Trata-se de um princípio relevante tanto na física e na química como na biologia e na antropologia. Os sistemas auto organizados são caracterizados por não apresentarem um comportamento previsível formado pelas interações locais e dos indivíduos participantes (GOLDBARG; GOLDBARG; LUNA, 2016).

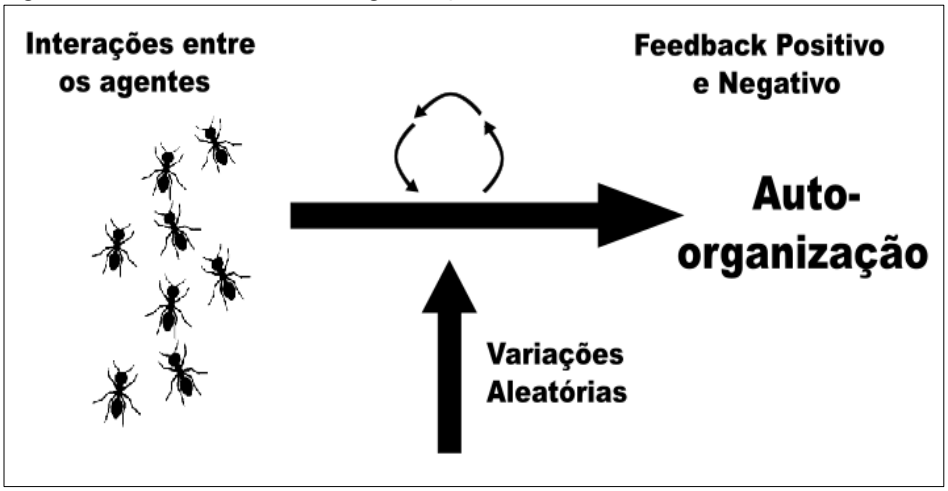
a fazer com que o sistema se redirecione a favor deste *feedback* obtendo escolhas não previstas;

b) ***feedback* negativo:** é apresentado em forma de exaustão, saturação ou competição, podendo estabilizar ou reduzir a variabilidade interna. Ação oposta ao *feedback* positivo;

c) **amplificação de flutuações aleatórias:** são fontes internas de variação no sistema, como os caminhos aleatórios, erros, mudanças aleatórias de atribuições de tarefas e comunicações cruzadas;

d) **múltiplas interações:** o processo requer um número mínimo de indivíduos para que estes possam ter interações de forma a tolerar erros e imprecisão de informações. Os indivíduos são autônomos e robustos.

Figura 4 – Processo de auto-organização.



Fonte**:** Goldbarg, Goldbarg e Luna (2016).

Conhecido o processo de auto-organização, é possível compreender como extrair um processo natural baseado em agentes reais como as formigas em suas colônias para sistemas científicos dentro da área da computação. Gaspar-Cunha, Takahashi e Antunes (2013), citam as formigas como insetos sociais que são capazes de se auto-organizarem e comunicarem por meio da estigmergia podendo realizar tarefas de extrema complexidade. A estigmergia é um fenômeno que se refere a ação de um determinado agente deixar sinais no meio ambiente que poderão ser recebidos por outros agentes, geralmente de uma mesma espécie,

proporcionando uma comunicação entre eles, que pode afetar as ações subsequentes do grupo. No caso das formigas esta sinalização é feita pelo depósito de feromônio na rota. De acordo com Goldbarg, Goldbarg e Luna (2016), sem a forma de comunicação promovida pela estigmergia, as formigas executariam buscas cegas com medidas de decisões aleatórias, caso as informações gerais fossem predominantes, as formigas iriam se dirigir para uma solução local. Graças a estrutura da comunicação entre as formigas e o depósito de feromônio, pode-se intensificar e diversificar a busca.

Um exemplo de sucesso da utilização da meta-heurística de otimização de colônia de formigas é o uso que os engenheiros têm feito para possibilitar aos mesmos um melhor caminho para rotear cabos em uma rede de comunicações. Devido as formigas ficarem continuamente varrendo a rede, este método é capaz de lidar bem com mudanças no ambiente, como por exemplo, obstruções e novas rotas (COPPIN, 2010).

Conforme dito por Souza, Lopes e Januario (2016), uma formiga artificial constrói probabilisticamente uma solução para o problema considerando o feromônio já depositado no ambiente e, em seguida, deposita feromônio com base na solução construída. Segundo Goldbarg, Goldbarg e Luna (2016), as formigas artificiais se afastam da inspiração biológica pelo menos nos seguintes pontos:

a) as formigas artificiais vivem em um mundo muito discreto;
b) possuem um estado interno capaz de guardar suas ações executadas no passado;
c) a quantidade depositada de feromônio é em função da qualidade da solução corrente;
d) os algoritmos podem ser enriquecidos com habilidades extras, como otimização local e *backtracking*[10].

O comportamento da colônia de formigas acompanhado nos estudos de ACO proporcionaram aos cientistas da computação o desenvolvimento de algoritmos para solução de problemas de otimização. Dos algoritmos que se baseiam no comportamento das formigas, aqueles que são criados a partir dos princípios de

[10] A expressão backtracking, ou busca do retrocesso, tem Ideia de retornar pelo caminho já percorrido em busca de novas soluções, através de uma busca em profundidade, alterando os valores para a variável apresentada em cada retrocesso (RUSSEL; NORVING, 2013).

ACO geralmente se destacam e são amplamente reconhecidos, em função da diversidade de problemas em que foram aplicados (DORIGO; STÜTZLE, 2004).

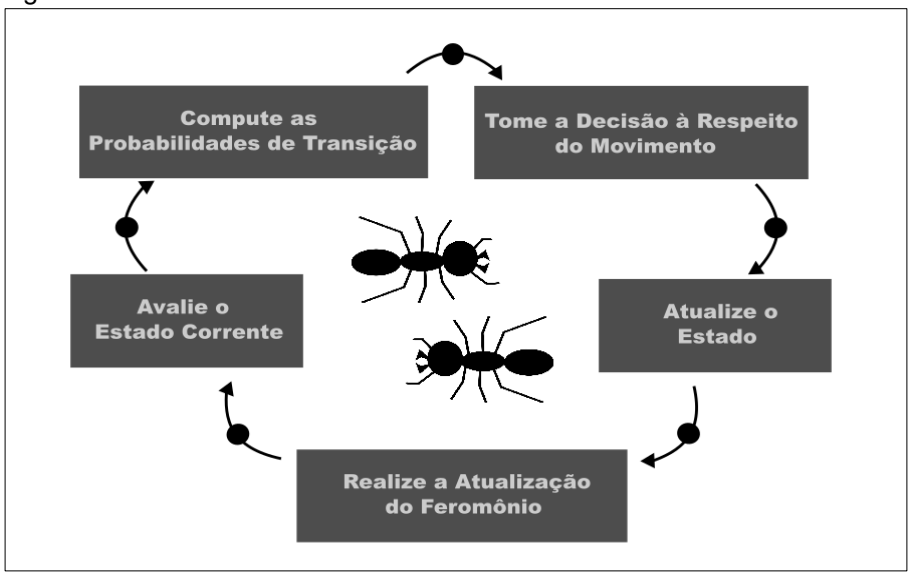
4 ALGORITMOS DE OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

Em função da analogia com a computação natural, os primeiros experimentos baseados na meta-heurística por colônia de formigas foram aplicados em problemas de otimização associado com o espaço físico como, por exemplo, o Problema do Caixeiro Viajante (PCV). Este foi o primeiro problema a ser abordado pela ACO por ter o cenário ideal para utilização da técnica (CONSTANTINO, tradução nossa; DORIGO; MANIEZZO; COLORNI, 1996, tradução nossa; DORIGO; GAMBARDELLA, 1997, tradução nossa; GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013; MULATI; SILVA, 2013).

> O PCV pode ser descrito como o problema de um vendedor que necessita sair de sua cidade, percorrer todas as cidades contidas em uma área geográfica e retornar a sua cidade de origem de tal maneira que o percurso total seja mínimo e cada cidade seja visitada uma única vez. O PCV pode ser representado como um grafo completo G = (V, E), sendo V o conjunto de cidades e E o conjunto de arestas que conectam todas as cidades. A cada aresta (*i,j*) é atribuído um valor d*ij* que representa a distância entre as cidades *i* e *j* (MULATI; CONSTANTINO; SILVA, 2013).

O ciclo construtivo dos Algoritmos Baseados em Colônia de Formiga (ACF) apresentado por Goldbarg, Goldbarg e Luna (2016), pode ser exemplificado na figura 5. Cada bloco do ciclo representa estratégias que podem ser igualmente encontradas na implementação do algoritmo, como a atualização do feromônio e a computação da probabilidade. Na figura 6 tem-se um exemplo de implementação da meta-heurística ACO proposto por Dorigo, Caro e Gambardella (1999).

Figura 5 – Ciclo construtivo dos ACFs.



Fonte: Goldbarg, Goldbarg e Luna (2016).

Os blocos de código da figura 6 apresentam a implementação de uma meta-heurística ACO. Cada bloco permite o desenvolvimento de estratégias específicas para problemas computacionais estáticos ou dinâmicos. As funções das figuras são nomeadas de acordo com as ações que executam, e no caso do método *acoes_extras* seriam executadas as ações extras de busca. A memória dos rastros de feromônios ficam registrados na matriz *a*. O depósito dos feromônios é executado, arco após arco, no momento em que a formiga toma sua decisão por meio do método *depositar_feromonio_arco_visitado*, ou tardiamente caso esta estratégia seja adotada.

Ao executar a estratégia tardia, todos os arcos são marcados de uma única vez passando por uma avaliação, na qual caso a solução formada pela formiga não seja atrativa, nenhum arco é marcado. As funções *calcular_probabilidade_por_transicao* e *evaporar_feromonio* são mecanismos utilizados para evitar uma concentração prematura do feromônio. A medida que as formigas executam seus passos, ou tardiamente quando as soluções construídas pelas formigas forem comparadas, a atualização do feromônio poderá ocorrer simultaneamente (DORIGO; CARO; GAMBARDELLA, 1999; GOLDBARG; GOLDBARG; LUNA, 2016).

Figura 6 – Pseudocódigo ACF.

```
def aco_meta_heuristica
    while criterio_parada_nao_satisfeito do
        gerar_formigas_e_ativar
        evaporar_feromonio
        acoes_extras # opcional
    end
end

def gerar_formigas_e_ativar
    while existe_recursos_disponiveis do
        cria_nova_formiga
        nova_formiga_ativa
    end
end

def nova_formiga_ativa
    iniciar_formiga
    m = atualizar_memoria_formiga

    while estado_atual <> estado_alvo do
        a = ler_tabela_local_roteamento_formiga
        p = calcular_probabilidade_por_transicao(a,m,restricoes)
        proximo_estado = aplicar_politica_decisao(p,restricoes)
        mover_proximo_estado(proximo_estado)

        if atualizacao_feromonio_online
            depositar_feromonio_arco_visitado
            atualizar_tabela_roteamento_formiga
        end

        m = atualiza_estado_interno
    end

    if atualizacao_tardia_feromonio_online
        avaliar_solucao
        depositar_feromonio_todos_arcos_visitados
        atualizar_tabela_roteamento_formiga
    end

    morrer
end
```

Fonte: Dorigo, Caro e Gambardella (1999, tradução nossa).

O *Ant System* (AS) foi o primeiro algoritmo desenvolvido com base em ACO, tendo sua primeira implementação no artigo de Dorigo, Maniezzo e Colorni (1991). O AS passou por diversas customizações, visando a melhora dos resultados, diminuição do esforço computacional e sua extensão para problemas multiobjetivo e de domínios contínuos (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013). Dorigo, Birattari e Stutzle (2006), montaram uma lista (tabela 1) com as principais variações de implementações bem-sucedidas de ACO.

Tabela 1 - Principais variações do algoritmo ACO

Algoritmo	Primeiras referências
Ant System (AS)	Dorigo et al. (1991)
Elitist AS (EAS)	Dorigo (1992)
Ant-Q	Gambardella e Dorigo (1995)
Ant Colony System (ACS)	Dorigo e Gambardella (1997a,b)
Max-Min AS (MMAS)	Stützle e Hoos (1996)
Rank-Based AS (ASrank/RBAS)	Bullnheimer et al. (1997)
ANTS	Maniezzo (1999); Maniezzo e Carbonaro (2000)
Best-Worst AS (BWAS)	Cordón et al. (2000)
Hyper-Cube AS (HCAS)	Blum et al. (2001)

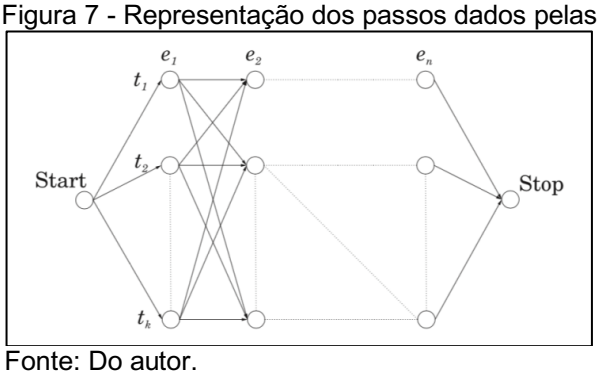
Fonte: Dorigo, Birattari e Stutzle (2006).

4.1 ALGORITMO MIN-MAX ANT SYSTEM MMAS

O algoritmo chamado de Min-Max Ant System (MMAS) é uma variação de ACO, derivado do algoritmo *Ant System*, que se mostra viável para resolução de problemas de otimização combinatória difíceis (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2013; STÜTZLE, 2000, tradução nossa).

Para o caso de resolução de problema de grade horária em universidades (UCTP), considera-se que cada formiga siga uma lista de eventos e então faça a escolha de um *timeslot*. Cada evento pode ser posto apenas uma vez em um mesmo *timeslot*, porém pode haver mais de um evento em um mesmo *timeslot*, sendo que em cada passo a formiga pode escolher uma possível transição (SOCHA; KNOWLES; SAMPELS, 2002, tradução nossa). Uma representação de como as formigas realizam seus passos é dada na figura 7.

Figura 7 - Representação dos passos dados pelas formigas



Fonte: Do autor.

Na figura 8 pode ser encontrado uma representação de um algoritmo MMAS, onde é possível visualizar a rotina de iteração das formigas dentro da colônia, passado pelas fases de construção da solução, busca local (opcional), ordenação da solução e atualização do feromônio.

Figura 8 - Algoritmo Max-Min AS (MMAS)

```
1   para cada colonia faça
2     para cada formiga faça
3       ConstruirSolução
4       AplicarBuscaLocal (opcional)
5     fim para
6
7     OrdenarSoluções
8     AtualizarFerominio
9   fim para
10
```

Fonte: Do autor.

Na fase de construção cada formiga escolhe para qual destino ir de acordo com uma função de probabilidade (1) que avalia a distância da cidade e a quantidade de feromônio em cada aresta (i, j). Os valores de α e β são argumentos que regulam a influência da matriz de feromônio τ_{ij}, e da informação heurística η_{ij}, (DORIGO; STÜTZLE, 2004, tradução nossa).

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum \iota \in \mathrm{N}_{i}^{k}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}} \quad (1)$$

Na equação (2) é possível ver a função de probabilidade adaptada ao problema de alocação de quadro de horários para universidades, onde o valor de $\tau_{(e_i,t)}$ são referentes ao evento e *timeslot,* e o valor de A_{i-1} representa a formiga anterior (SOCHA; KNOWLES; SAMPELS, 2002, tradução nossa).

$$p_{e_i,t}(\tau(A_{i-1}),\eta(A_{i-1})) = \frac{(\tau_{e_i,t}(A_{i-1}))^{\alpha}.(\eta_{e_i,t}(A_{i-1}))^{\beta}}{\sum_{\theta \epsilon \mathrm{T}}(\tau_{e_i,\theta}(A_{i-1}))^{\alpha}.(\eta_{e_i,\theta}(A_{i-1}))^{\beta}} \quad (2)$$

De acordo com Gaspar-Cunha, Takahashi e Antunes (2013), a deposição do feromônio é dada conforme a equação (3), onde o valor do $\Delta\tau_{ij}^{melhor}$ diz respeito a deposição nas arestas da melhor rota.

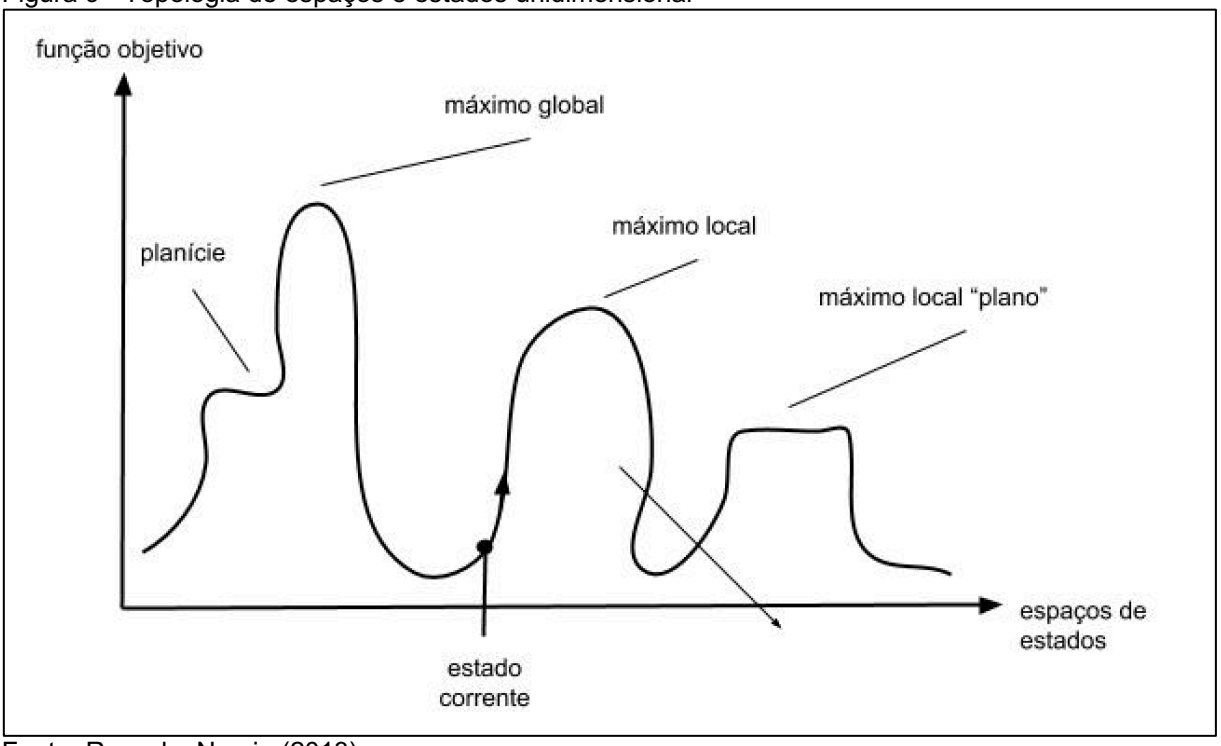
$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{melhor} \quad (3)$$

Para soluções do problema de alocação de grade horária em universidades com MMAS, Dorigo e Stützle (2004), contam que o rastro de feromônio τ_{ij} refere-se à alocação de eventos *i* para *timeslots j*, sem o uso da informação heurística. No entanto é necessário que os eventos sejam pré-ordenados e alocados nos *timeslots* de acordo com a probabilidade. Quando finalizado a fase de construção, a solução é melhorada com o uso de um algoritmo de Busca Local.

A busca local, do inglês *Local Search* (LS), é um método de busca que opera usando apenas o estado atual, e em geral se move apenas para os vizinhos deste estado, geralmente não guardando o histórico do trajeto que foi percorrido. Algoritmos de busca local, por sua vez, são úteis para resolver problemas de otimização, que não se adaptam a algoritmos de busca padrão, já que o objetivo é encontrar o melhor estado de acordo com uma função objetivo (RUSSEL; NORVIG, 2013).

Segundo Russel e Norvig (2013), para entender a busca local é possível considerar uma topologia de espaços e estados (figura 9), onde tem-se uma posição que é definida pelo estado e uma elevação que é definida pelo custo obtido pela função heurística ou pela função objetivo. Dado que a elevação corresponde ao custo, o objetivo é encontrar o vale mais baixo (mínimo global), se a elevação corresponde a função objetivo, então o objetivo será encontrar o pico mais alto (máximo global). Um algoritmo de busca local completo sempre encontra um objetivo, caso ele exista, e um algoritmo de busca local ótimo sempre encontra um mínimo/máximo global.

Figura 9 - Topologia de espaços e estados unidimensional



Fonte: Russel e Norvig (2013).

Um exemplo de algoritmo que aplica busca local é o chamado subida de encosta, do inglês *Hill Climbing Algorithm*, que consiste em um laço repetitivo contínuo que se move de forma crescente, ou seja, encosta acima. Como o algoritmo não mantém um histórico dos caminhos visitados é considerado apenas o nó atual e o dos vizinhos imediatos. Segundo Russel e Norvig (2013), é como tentar alcançar o topo do Monte Everest em meio a um nevoeiro denso e durante uma crise de amnésia.

Na figura 10 é possível visualizar um exemplo da implementação de um algoritmo de subida de encosta. Basicamente, gera-se uma solução aleatória, calcula-se a função heurística ou função objetivo, e se a solução de um vizinho for melhor que a solução atual, a mesma é substituída pela nova solução.

Figura 10 - Pseudocódigo subida de encosta

```
1   Gera uma solução aleatória para x
2   Calcula o valor da função objetivo para f(x) da solução gerada
3
4   until criterio_parada
5     if solucao_nova_melhor_que_atual
6       subjstitui_solucao_atual_pela_solucao_nova
7     end
8   end
```

Fonte: Brabazon, Oneill e Mcgarraghy (2015, tradução nossa).

5 QUADRO DE HORÁRIOS NAS ESCOLAS

O problema de geração de quadro de horários, também conhecido pelo nome de *timetabling problem*, é um clássico dentro das meta-heurísticas, justamente por ser um problema de otimização onde nem sempre é necessário um resultado absoluto, mas sim uma melhor solução possível em tempo computacional aceitável (PATRICK; GODSWILL, 2016, tradução nossa; SOCHA; KNOWLES; SAMPELS, 2002, tradução nossa; UGAT; MONTEMAYOR, MANLIMOS, 2018, tradução nossa).

Neukirchen (2015) determina a geração de quadro de horários como um problema combinatório restrito e classificado como NP-completo. Schaerf (1991) lista as vertentes do problema de *timetabling* em rede educacional:

a) geração de quadro de horários para escola (*school timetabling);*
b) geração de quadro de horários para universidades (*university timetabling*);
c) geração de quadro de horários para enxames (*examination timetabling*).

O processo de geração de quadro de horários para escolas, o qual é o foco desta pesquisa, geralmente é feito de forma manual nas instituições de ensino e consiste em relacionar turmas, disciplinas e professores nos respectivos horários ou dias que tem disponibilidade. Este processo pode parecer simples, mas após estudar com mais profundidade, percebe-se que quando feito de forma manual, pode ser bem complexo, desgastante e demorado, além da possibilidade de falhas. Alves (2010) conta que o quadro de horários é elaborado, na maioria dos casos, de forma manual, o que gera desperdício do tempo dos responsáveis e, nem sempre o resultado é satisfatório.

Segundo Neukirchen (2015) a geração de um quadro de horários não otimizado, pode inclusive ser prejudicial para o rendimento dos alunos, como por exemplo, uma sobrecarga de aulas em um mesmo dia, dificultando o aprendizado dos estudantes, ou então a necessidade de deslocamento para salas muito distantes, o que pode gerar atrasos e perda de conteúdos. A figura 11 representa um exemplo de quadro de horários.

Figura 11 - Exemplo de grade horária em escolas

TURMA	HORARIO	SEG	TER	QUA	QUI	SEX
1 Ano A	07h às 08h	PT Vera Lucia	MAT Ederson Souza	ING Luciana	GEO Maria Silva	ED. FISICA João P.
	08h às 09h	PT Vera Lucia	MAT Ederson Souza	ING Luciana	ED. FISICA João P.	ED. FISICA João P.
	09h às 10h	MAT Ederson Souza	PT Vera Lucia	ED. FISICA João P.	ED. FISICA João P.	ART Maicon R.
	10h às 11h	MAT Ederson Souza	PT Vera Lucia	MAT Ederson Souza	CIE Pedro Melo	ART Maicon R.
	11h às 12h	GEO Maria Silva	GEO Maria Silva	PT Vera Lucia	CIE Pedro Melo	REL Joana L.
1 Ano B	07h às 08h	ED. FISICA João P.	GEO Maria Silva	MAT Ederson Souza	ING Luciana	MAT Ederson Souza
	08h às 09h	ED. FISICA João P.	ED. FISICA João P.	MAT Ederson Souza	ING Luciana	MAT Ederson Souza
	09h às 10h	ART Maicon R.	ED. FISICA João P.	PT Vera Lucia	ED. FISICA João P.	PT Vera Lucia
	10h às 11h	ART Maicon R.	CIE Pedro Melo	PT Vera Lucia	MAT Ederson Souza	PT Vera Lucia
	11h às 12h	REL Joana L.	CIE Pedro Melo	GEO Maria Silva	PT Vera Lucia	GEO Maria Silva

Fonte: Do autor.

Na figura 11 é possível visualizar o problema de grade horária, tendo como exemplo as disciplinas que devem ser distribuídas em períodos (horários) para cada turma da escola, e que para isso, devem ser considerados os professores que ministram estas disciplinas e a sua disponibilidade de horário.

Souza, Maculan e Ochi (2001), citam como alguns dos requisitos para a qualidade do resultado final de um quadro de horários:

a) o professor não pode estar alocado para ministrar aula em mais de uma turma ao mesmo tempo;

b) uma turma não pode ter mais de uma aula alocada no mesmo horário;

c) cada professor tem que cumprir a sua carga horária semanal;

d) um professor só pode ser alocado em horários nos quais ele tem disponibilidade;

e) uma turma não pode ter mais do que dois horários de aula de uma mesma matéria em um mesmo dia;

f) tentar atender ao máximo o período de duas aulas consecutivas, quando houver mais de um horário de alguma matéria;

g) a agenda do professor deve ser tão compacta quanto possível.

A geração da grade horária constitui-se em uma das atribuições dos gestores escolares. No desempenho das atividades de gestão escolar, têm-se algumas ferramentas que auxiliam neste processo, como por exemplo, o software público i-Educar.

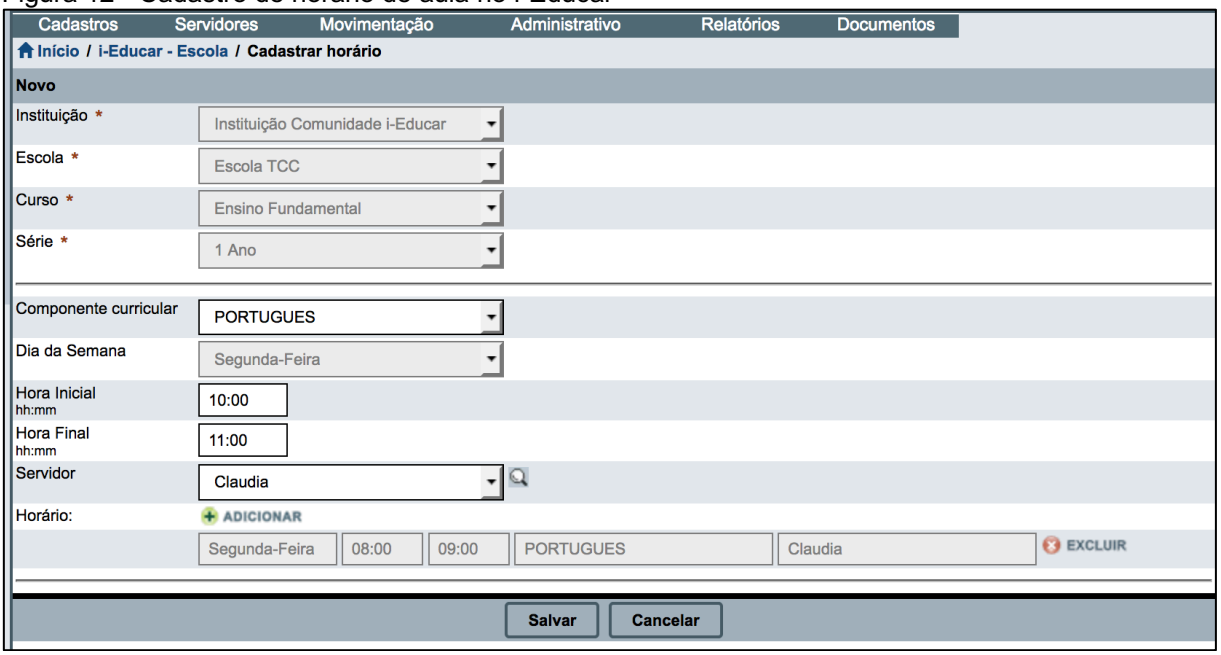
5.1 SOFTWARE PÚBLICO DE GESTÃO ESCOLAR I-EDUCAR

O i-Educar é um software público de gestão escolar que pode ser utilizado e distribuído livremente por qualquer pessoa ou organização, tendo uso da versão 2 da Licença Pública Geral do GNU, do inglês, General Public License (GPL)[11]. Ele foi desenvolvido na prefeitura de Itajaí no estado de Santa Catarina (BRASIL, 2008; GIROTO, 2014).

No início de cada período letivo, é necessário realizar a alocação dos horários dos professores e das turmas de alunos. A elaboração do quadro de horários é uma tarefa complexa que requer um tempo considerável e dedicação dos profissionais da rede de ensino. Inclusive no i-Educar, a elaboração do quadro de horários torna-se também difícil, pois requer um esforço dos usuários para tomar as decisões sem o auxílio de uma ferramenta automatizada (ALVES, 2010). Como é possível ver na figura 12 e na figura 13 onde são realizados os passos para criação manual do quadro de horários.

[11] A GPL versão 2 é uma licença de software que da o direito a qualquer pessoa de copiar e distribuir cópias de um software, link da tradução: https://creativecommons.org/licenses/GPL/2.0/legalcode.pt.

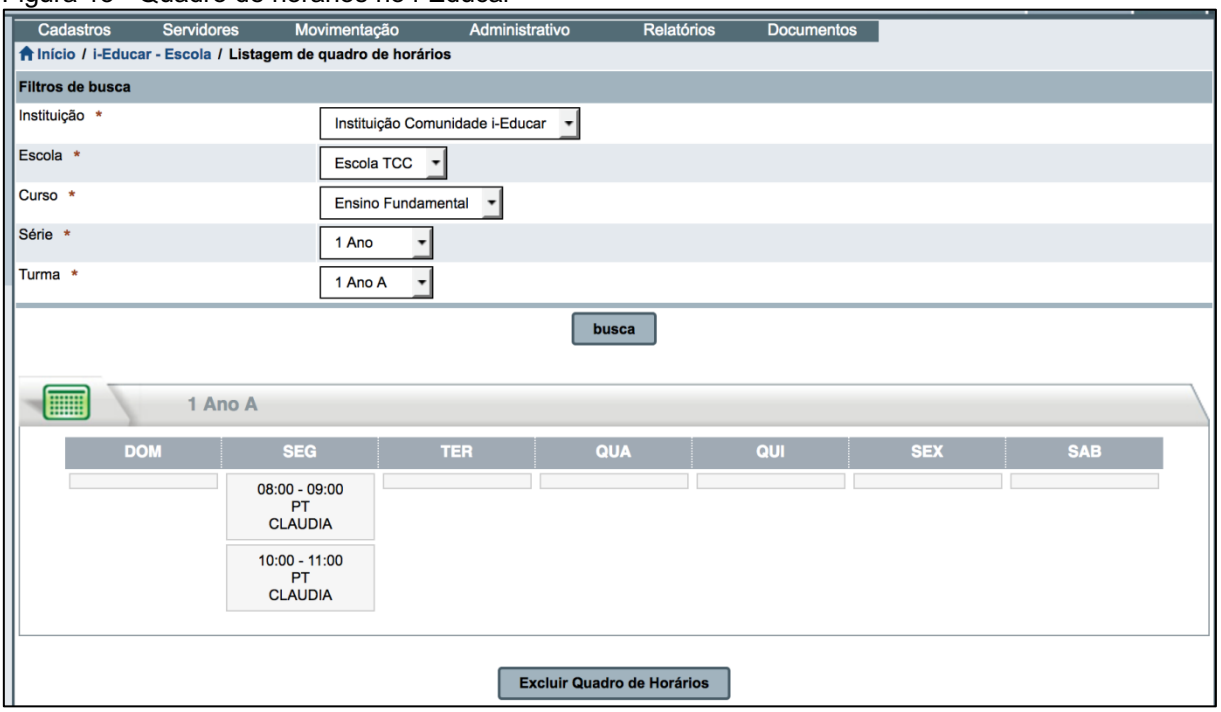
Figura 12 - Cadastro de horário de aula no i-Educar



Fonte: i-Educar (2018).

Na figura 12 é possível visualizar a tela de cadastro de um horário de aula no i-Educar, sendo necessário informar os horários um a um, bem como o professor e a disciplina lecionada. Para que o professor esteja disponível para dar aula, ele também deve ser alocado com horas disponíveis para cada escola em que o mesmo irá trabalhar. Em uma rede de ensino com muitas escolas, este é um trabalho que pode levar não só horas como dias, pois cada horário tem que ser cadastrado separadamente, tendo que consultar a disponibilidade de cada professor, sem recursos automáticos que combinam as matérias no melhor horário. Na figura 13 tem-se o quadro de horários tomando forma, com os dois horários que foram inseridos na segunda-feira.

Figura 13 - Quadro de horários no i-Educar



Fonte: i-Educar (2018).

O i-Educar é utilizado em muitas escolas públicas de municípios brasileiros, o que faz com que uma melhoria no modo como é gerado o quadro de horários no software também reflita em todas estas escolas, e em consequência pode ocasionar uma melhoria na gestão educação como um todo. Cesar (2012) levantou uma lista dos municípios que utilizam o i-Educar, como pode ser visto na tabela 2. Dados atualizados dos municípios que estão atualmente utilizando o software podem ser encontrados no Portal do Software Público[12].

[12] Link da wiki do i-Educar no Portal do Software Público onde é possível encontrar informações atualizadas pela comunidade dos municípios que utilizam o i-Educar: https://softwarepublico.gov.br/gitlab/i-educar/i-educar/wikis/quem-utiliza.

Tabela 2 - Municípios que utilizam o i-Educar

Estado	Instituição
Alagoas	Município de Arapiraca Município de Maceió
Bahia	Município de Irecê Município de São Francisco do Conde
Minas Gerais	Município de Montes Claros
Pará	Município de Pacajá
Paraíba	Colégio da Polícia Militar Município de Natuba
Rio Grande do Norte	Município de Mossoró
Município de Mossoró	Município de Cacoal
Santa Catarina	Município de Araranguá Município de Balneário Gaivota Município de Florianópolis Município de Içara Município de Itajaí Município de Jacinto Machado Município de Laguna Município de Maracajá Município de Meleiro Município de Nova Veneza Município de Praia Grande Município de Santa Rosa do Sul Município de São João do Sul Município de Sombrio Município de Timbé do Sul

Fonte: Cesar (2012).

6 TRABALHOS CORRELATOS

Várias técnicas foram desenvolvidas para realizar a elaboração automática de quadro de horários utilizando meta-heurísticas. Dentre as técnicas de meta-heurísticas, os algoritmos de otimização por colônia de formigas se destacam por terem motivado bons resultados em pesquisas nos últimos anos. A seguir, tem-se algumas pesquisas que são relacionadas aos temas estudados neste trabalho.

6.1 ESTRATÉGIA DE OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS PARA RESOLVER O PROBLEMA DE GERAÇÃO DE QUADRO DE HORÁRIOS PARA CURSOS DE UNIVERSIDADES

O artigo escrito por Kenekayoro Patrick e Zipamone Godswill, publicado na revista *British Journal of Mathematics & Computer Science* no ano de 2016, fala dos problemas de geração de quadro de horários na rede de ensino superior.

Nesta pesquisa, os autores buscam resolver o problema de geração de quadro de horários para cursos de universidades, do inglês *University Course Timetabling Problem* (UCTP), e para alcançar este resultado os autores empregam o sistema de colônia de formigas utilizando o conjunto de dados fornecidos pela Competição Internacional de *Timetabling*, do inglês *International Timetabling competition* (ITC), que podem ser aplicados para realizar um teste de qualidade de seu protótipo. Patrick e Godswill (2016), contam que várias técnicas de meta-heurísticas obtiveram sucesso quando testadas com os dados do ITC, mas que poucas vezes a técnica de otimização por colônia de formigas foi utilizada, principalmente no problema de UCTP.

Os autores concluíram que em termos de qualidade, o modelo do algoritmo proposto na pesquisa não alcança os melhores resultados se comparado com a maioria, mas com algumas exceções. Das 21 instâncias utilizadas para testar, 17 delas tiveram pouca violação das restrições, menores do que a média de todos os algoritmos comparados (PATRICK; GODSWILL, 2016, tradução nossa).

6.2 UM SISTEMA DE FORMIGAS MAX-MIN APLICADO AO PROBLEMA DE GRADE HORÁRIA EM CURSOS DE UNIVERSIDADES

Artigo escrito por Krzysztof Socha, Joshua Knowles, e Michael Sampels, publicado pela revista *Ant Algorithms* no ano de 2002.

O referente trabalho implementa o algoritmo baseado em colônia de formigas MMAS para resolver o problema de geração de grade horária em universidades, envolvendo três tipos de violações complexas e três tipos de violações simples. Além da implementação do algoritmo MMAS, os autores também propuseram fazer o uso de uma rotina de busca local para chegar mais próximo de uma solução ótima. Segundo Socha, Knowles e Sampels (2002), a busca local é determinística e termina quando um local ótimo for alcançado.

Este modelo foi avaliado pelo autor por meio de onze instâncias de três níveis do problema, sendo os níveis pequeno, médio e grande. Os resultados demonstraram que é possível construir grades horárias melhor de forma significativa utilizando um algoritmo de sistema de formigas com busca local e soluções iniciais aleatórias. Como resultado da pesquisa, descobriu-se que algoritmos baseados em sistema de formiga são capazes de lidar com problemas com múltiplas restrições, e que somado ao uso da busca local pode trazer uma melhora adicional no desempenho. Comparações indicam que o algoritmo MMAS, construído neste trabalho, é competitivo com outras meta-heurísticas que foram desenvolvidas para resolver este tipo de problema (SOCHA; KNOWLES; SAMPELS, 2002, tradução nossa).

6.3 DEFINIÇÃO E IMPLEMENTAÇÃO DE UMA FUNÇÃO DE AVALIAÇÃO PARA UM SISTEMA DE GERAÇÃO DE GRADE HORÁRIA UTILIZANDO ALGORITMO GENÉTICO

Trabalho de Conclusão de Curso desenvolvido pelo aluno Rodrigo Meurer Melo para obtenção do título de Bacharel em Ciência da Computação, realizado na Universidade do Extremo Sul Catarinense (UNESC) em Criciúma – SC no ano de 2003.

No presente trabalho o autor desenvolveu um protótipo, o qual ele chama de URANO, com a proposta de resolver o problema de geração de quadro de horários em instituições de ensino. Como base para resolução do problema, fora utilizado o sistema de geração de quadro de horários já desenvolvido anteriormente na Universidade do Planalto Catarinense (UNIPLAC) em 2001, o qual apresentava alguns problemas no resultado como coincidência de horários e disponibilidade dos professores para as aulas. Melo (2003), justifica os problemas encontrados no sistema desenvolvido pela UNIPLAC por uma definição incompleta da função de avaliação, que é relacionada ao algoritmo genético ao atribuir o grau de adaptação de cada cromossomo dentro da população.

Na pesquisa concluiu-se que o método de buscas por Algoritmos Genéticos pode ser uma forma eficiente para resolver este problema de otimização. O autor construiu uma solução refinada e com uma fase de testes que seleciona com maior precisão a melhor solução gerada pelo algoritmo, eliminando completamente o problema de coincidência de horários que existia anteriormente. Com este resultado, podem ser criados novos quadros horários de forma simples e que mesmo não sendo perfeito atendem todas as restrições predefinidas pelos coordenadores dos cursos na entrada do sistema (MELO, 2003).

6.4 FERRAMENTA BASEADA EM COLÔNIA DE FORMIGAS PARA RESOLVER PROBLEMAS DE GRADE HORÁRIA EM INSTITUIÇÕES DE ENSINO

Este artigo escrito por Thatchai Thepphakorn, Pupong Pongcharoen e Chris Hicks, foi publicado em março de 2014 na revista *International Journal of Production Economics*.

Para este trabalho, foi construído um protótipo chamado de Ferramenta de *Timetabling B*aseada em Colônia de Formigas, do inglês *Ant Colony Based Timetabling Tool* (ANCOTT), que tem como objetivo resolver os problemas de geração de quadro de horários de palestras, seminários, sessões práticas, exames e aulas em instituições de ensino. Segundo Thepphakorn, Pongcharoen e Hicks (2014), a proposta foi de utilizar estratégias de busca local como o Sistema de Formigas Melhor-Pior, do inglês *Best-Worst Ant System* (BWAS), e o Sistema de Colônia de Formigas Melhor-Pior, do inglês *Best-Worst Ant Colony System*

(BWACS), pois estas técnicas ajudam a encontrar com eficiência um cronograma otimizado com o baixo número de violações e restrições suaves.

As técnicas de BWAS e BWACS foram utilizadas para resolver problemas de tempo, e novas pesquisas locais foram incluídas para aumentar a eficiência do protótipo. Após realizar um estudo por meio de ferramentas de análise e estatística, verificou-se que a eficiência do BWAS foi melhorada usando configurações de parâmetros otimizados. O desempenho com o uso de BWAS e BWACS em termos de qualidade da solução obtida e sua velocidade foram melhores do que as variações originais de ACO para geração de quadro de horários. Conclui-se então que o BWAS produziu resultados melhores em problemas pequenos, enquanto o BWACS para problemas maiores, sendo que com o uso das pesquisas locais o tempo computacional foi mais longo (THEPPHAKORN; PONGCHAROEN; HICKS, 2014, tradução nossa).

6.5 OTIMIZAÇAO DE COLONIA DE FORMIGAS PARALELA NO PROBLEMA DE TIMETABLING EM CURSO UNIVERSITARIO E DE FACULDADE EM MSU-IIT

Artigo escrito por Earth Ugat, Jennifer Montemayor, Mark Manlimos e Dante Dinawanao, publicado pela revista *GSTF Journal on Computing (JoC)*, no ano de 2018, nos periódicos da *Mindano State University-Iligan Institute of Technology (MSU-IIT)*, localizada na cidade de Iligan em Filipinas.

Este trabalho apresenta um algoritmo paralelo de otimização de colônia de formigas MMAS como abordagem para solução do problema de grade horária em curso/faculdade em universidades, do inglês, *The University Course-Faculty Timetabling Problem* (UCFTP), em sua própria instituição de ensino, a MSU-IIT. Ugat et al. (2018), cita o problema de UCFTP como a versão de maior dificuldade quando se trata de problemas de otimização de grade horária para universidades, contendo em função da delegação de salas de aula para disciplinas disponíveis, incluindo o cronograma e pessoal adequado do corpo docente, levando em consideração algumas restrições como capacidade de sala de aula, localização e limitação dos colaboradores envolvidos.

No final, os autores desenvolveram uma aplicação para gerar a grade

horária de forma automática utilizando o Erlang/OTP, uma linguagem funcional especializada em concorrência e sistemas distribuídos, baseando-se nos princípios de ACO e MMAS. Suas pesquisas confirmaram que cada configuração de ACO tem um comportamento distinto, mesmo em diferentes instâncias do problema com variados tamanhos. Em função disso, afirmam que é difícil encontrar, no futuro, uma solução que seja ótima para todos os casos de UCFTP, pois as configurações deverão ser calibradas e as características do problema examinadas para cada caso real do problema (UGAT et al, 2018).

6.6 ANÁLISE COMPARATIVA DE UM ALGORITMO DE BUSCA E SATISFAÇÃO DE RESTRIÇÕES E ALGORITMO GENÉTICO EM UM SISTEMA PARA GERAÇÃO DE HORÁRIO ESCOLAR

Trabalho de Conclusão de Curso do aluno Tágner Formanski Rosa para obtenção do grau de Bacharel em Ciência da Computação na Universidade do Extremo Sul Catarinense (UNESC) em Criciúma – SC, no ano de 2015.

O estudo tem como objetivo desenvolver um protótipo utilizando o software de algoritmo genético Cronos, para obtenção de uma análise comparativa de desempenho de busca e satisfação de restrições no problema de grade horária escolar. De acordo com Rosa (2015), uma das formas de melhorar o desenvolvimento do quadro de horários é com uso de algoritmos de busca de satisfação de restrições de *backtracking*, pois buscam soluções viáveis e com menor custo de processamento, e algoritmos genéticos, que por sua vez buscam as melhores soluções, eliminando as que não atendem as necessidades.

O autor identificou que os protótipos desenvolvidos com *backtracking* e algoritmo genético apresentaram fatores qualitativos similares em todas as categorias analisadas, sendo que para avaliar a qualidade da solução implementada, foi utilizada a norma de qualidade ISO/IEC 9126 – Parte 1 (NRB 13596), que fornece um modelo que compreende amplas características de qualidade de software. No caso de eficiência do algoritmo proposto, o protótipo feito em *backtracking* se saiu melhor na subcategoria relacionada ao tempo (ROSA, 2015).

7 O ALGORITMO MIN-MAX ANT SYSTEM APLICADO NA GERAÇÃO DE QUADRO DE HORÁRIOS EM SOFTWARE DE GESTÃO ESCOLAR

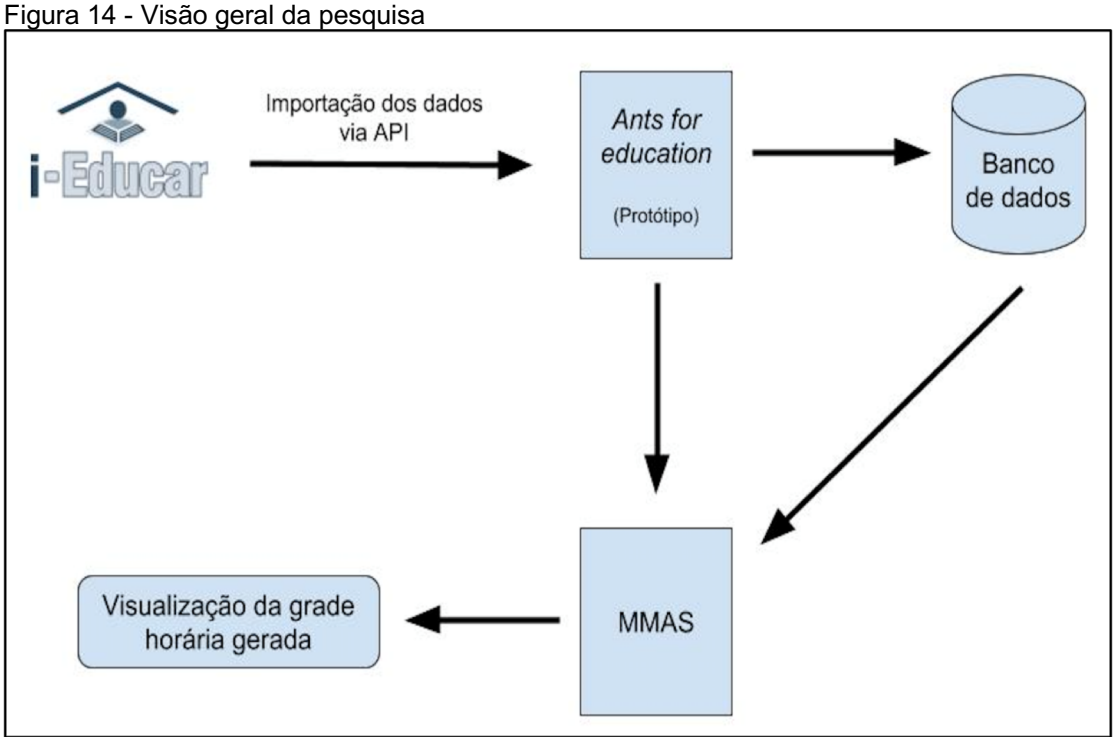
Algoritmos baseados em colônia de formigas têm sido utilizados para resolver problemas de otimização, como por exemplo, o do caixeiro viajante, do roteamento de veículos e da geração de grade horária. Considerando isso, nesta pesquisa, dentre os algoritmos baseados em colônia de formigas, implementou-se o MMAS. Este algoritmo tem tido bons resultados em problemas de alocação de horários em cursos universitários, para problemáticas não muito extensas, conforme os estudos realizados nas *pesquisas de* Hicks, Pongcharoen e Thepphakorn (2014); Knowles, Sampels e Socha (2002), e Ugat et al. (2018).

Nesta pesquisa, a implementação do algoritmo MMAS aplica os conceitos de otimização por colônia de formigas no protótipo chamado de *Ants for Education*, fazendo com que seja gerada uma opção de grade horária para uma escola, e que ao final cada turma tenha todos os períodos de aulas preenchidos e com os professores disponíveis alocados.

Para a realização desta implementação, os dados utilizados para a elaboração da grade horária foram gerados simulando os cadastros realizados no i-Educar pelas escolas que o utilizam, sendo estes dados fictícios. Como várias escolas utilizam este software de gestão, no futuro elas mesmas podem gerar automaticamente o quadro de horários por meio do protótipo.

Na figura 14 é possível ter uma visão geral desta pesquisa, que consiste em consultar uma Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API), no i-Educar, importar os dados para o banco de dados do protótipo, executar o algoritmo MMAS que vai processar os dados da escola e no final apresentar um quadro de horários.

Figura 14 - Visão geral da pesquisa



Fonte: Do autor.

7.1 METODOLOGIA

A pesquisa desenvolvida é aplicada e de base tecnológica, definida a partir do conhecimento adquirido, onde, para alcançar os seus objetivos, foram realizadas as atividades de levantamento bibliográfico; avaliação dos dados escolares necessários; desenvolvimento de uma API no i-Educar para extração de dados; importação de dados escolares; algoritmo MMAS; realização de testes; análise de desempenho e qualidade.

Na etapa do levantamento bibliográfico foram compreendidos os assuntos envolvidos nesta pesquisa, tais como os de problemas de otimização, o uso de meta-heurísticas, computação natural, otimização por colônia de formigas, o algoritmo MMAS e a problemática da geração de quadros de horários nas escolas que utilizam o software i-Educar.

7.1.1 Base de dados no i-Educar para geração de grade horária

O i-Educar possui a estrutura necessária para gerenciar as informações dentro de uma instituição de ensino, como cadastro de escolas, cursos, séries,

turmas e disciplinas. A vantagem de ter este software disponível na comunidade, é que o algoritmo MMAS implementado no protótipo deve trabalhar para otimizar dados similares ao de casos reais, como das escolas que utilizam o software de fato e precisam gerar seus quadros de horários de forma manual.

Nesta pesquisa, para coletar os dados do i-Educar, implementou-se uma API no código-fonte disponível na comunidade[13], utilizando a linguagem de programação PHP, para expor alguns *endpoints*[14] onde é possível resgatar por meio do método GET[15] do HTTP[16], informações direto da base de dados do programa. Como medida de segurança, é necessário gerar um código de identificação que fica oculto no i-Educar e deve ser informado como parâmetro da requisição no momento da coleta dos dados. No caso de invalidade do código de identificação, não é possível obter os dados da escola. A lista de *endpoints* criados no i-Educar para extração dos dados pode ser visualizada na tabela 3.

Tabela 3 - *Endpoints* criados no i-Educar para extração de dados

Resource	URL *ENDPOINT*
turmas	http://url_ieducar/module/api/turma
turmas-disciplinas	http://url_ieducar/module/api/turma
cursos	http://url_ieducar/module/api/curso
escolas	http://url_ieducar/module/api/escola
series	http://url_ieducar/module/api/serie
servidores	http://url_ieducar/module/api/servidor
servidores-disciplinas	http://url_ieducar/module/api/servidor
servidores-escolas	http://url_ieducar/module/api/servidor

Fonte: Do autor.

7.1.2 Importação de dados

Para fazer uso dos dados do i-Educar, foi efetuada uma importação de dados que deu início ao desenvolvimento do protótipo, que teve sua implementação em linguagem de programação Ruby 2.4.3, utilizando o *framework* Rails na versão 5.0.6, com a IDE RubyMine 2018.1. A escolha da tecnologia Ruby on Rails se deu

[13] O código-fonte da implementação realizada, nesta pesquisa, para integrar os dados do i-Educar para o protótipo pode ser encontrada em (https://github.com/carolinesalib/ieducar_tcc/tree/api-quadro-horario).

[14] Termo utilizado para representar a extremidade de uma URL, que no caso de uma API, representa o seu recurso.

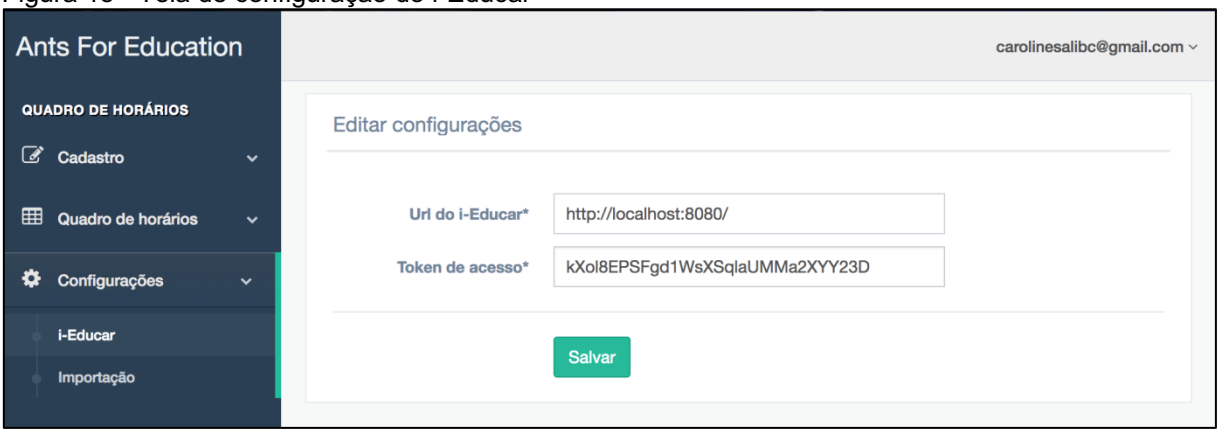
[15] É um tipo de método do HTTP utilizado para receber informações de um determinado recurso (MOLINARI, 2017).

[16] A definição completa do protocolo HTTP pode ser encontrada em(https://tools.ietf.org/html/rfc2616).

devido as facilidades que este *framework* proporciona, podendo ser criado toda a estrutura do protótipo de forma rápida com os recursos de automação do Rails, ganhando tempo para a implementação da meta-heurística de otimização por colônia de formigas, que é o foco de estudo nesta pesquisa. O banco de dados escolhido foi o PostgreSQL 10, por ser um gerenciador de banco de dados de código aberto, também bastante utilizado na comunidade de Ruby on Rails.

Anterior a importação, configuraram-se os dados necessários (figura 15) para que o protótipo pudesse acessar a API do i-Educar. Para isto, foi informada a URL do i-Educar, que deve estar rodando em algum servidor web, e também forneceu-se o código de acesso (*token*) para que fosse possível acessar os dados expostos na API.

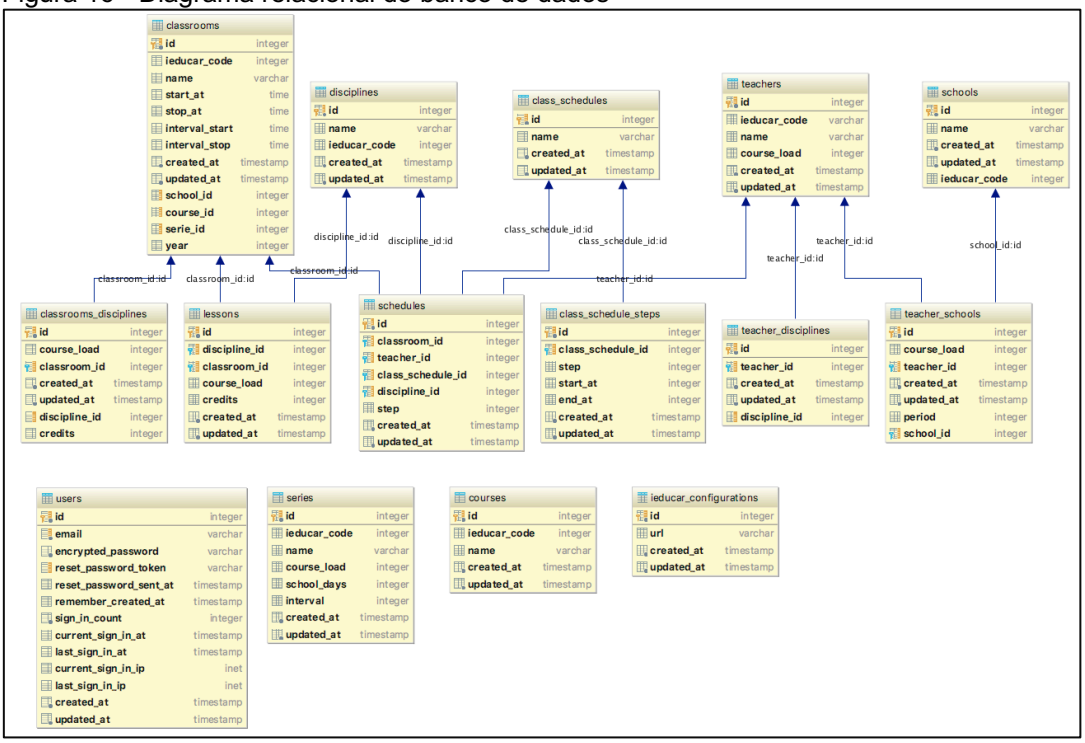
Figura 15 - Tela de configuração do i-Educar



Fonte: Do autor.

Para a realização da importação, é necessário o acesso deste item que está disponível na seção de configuração no menu do protótipo. A importação consiste em acessar cada *endpoint* disponível no i-Educar, tratar os dados, e inseri-los no banco de dados do protótipo. A estrutura do banco de dados pode ser visualizada no diagrama relacional (figura 16).

Figura 16 - Diagrama relacional do banco de dados



Fonte: Do autor.

Em relação aos dados do diagrama relacional (figura 16), uma breve explicação das principais tabelas e colunas que foram criadas, tendo como base os dados importados da plataforma i-Educar, e que são utilizadas para a geração da grade horária (tabela 4).

Tabela 4 - Principais tabelas e colunas usadas no desenvolvimento do quadro de horários

Tabela	Coluna	Descrição
classrooms**:** armazena as turmas	*ieducar_code* *name*	Código de identificação no i-Educar Nome da turma
disciplines: armazena as disciplinas	*ieducar_code* *name*	Código de identificação no i-Educar Nome da disciplina
lessons: relaciona turmas com disciplinas	*discipline_id* *classroom_id* *credits*	Código referente a disciplina Código referente a turma Quantidade de aulas da disciplina na turma
teachers**:** armazena professores	*ieducar_code* *name* *carga horária*	Código de identificação no i-Educar Nome do professor Carga horária do professor
teacher_disciplines**:** relaciona professores com disciplinas	*teacher_id* *discipline_id*	Código referente ao professor Código referente a disciplina

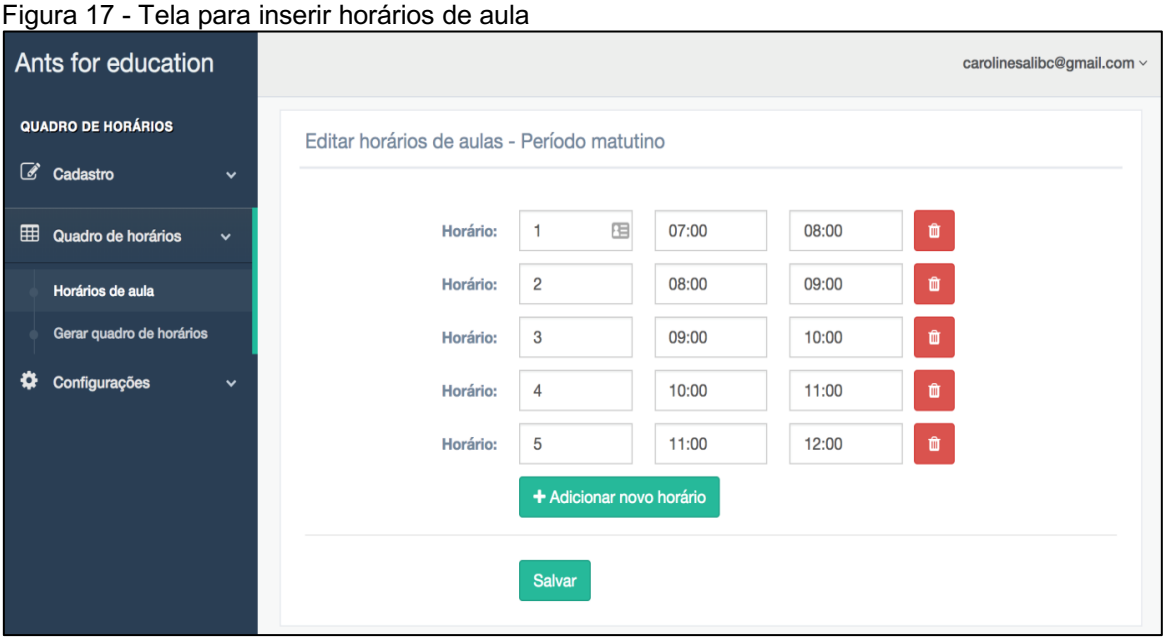
Fonte: Do autor.

7.1.3 Desenvolvimento do protótipo

Para melhor entendimento sobre os passos de desenvolvimento do protótipo, alguns aspectos do funcionamento geral devem ser entendidos, como por exemplo, para considerar uma solução factível, esta deve respeitar as seguintes restrições: não possui professor alocado no mesmo *timeslot* em mais de um evento; cada turma deve ter todos os *timeslots* alocados com eventos.

Definiu-se para esta implementação, que seria melhor ter os horários de aulas fixos, desta forma não seria necessário considerar o horário em si, somente a quantidade de aulas que as turmas teriam por período. Na figura 17 apresenta-se um exemplo de definição dos horários para o período matutino. Neste exemplo, as turmas têm cinco aulas por dia.

Figura 17 - Tela para inserir horários de aula

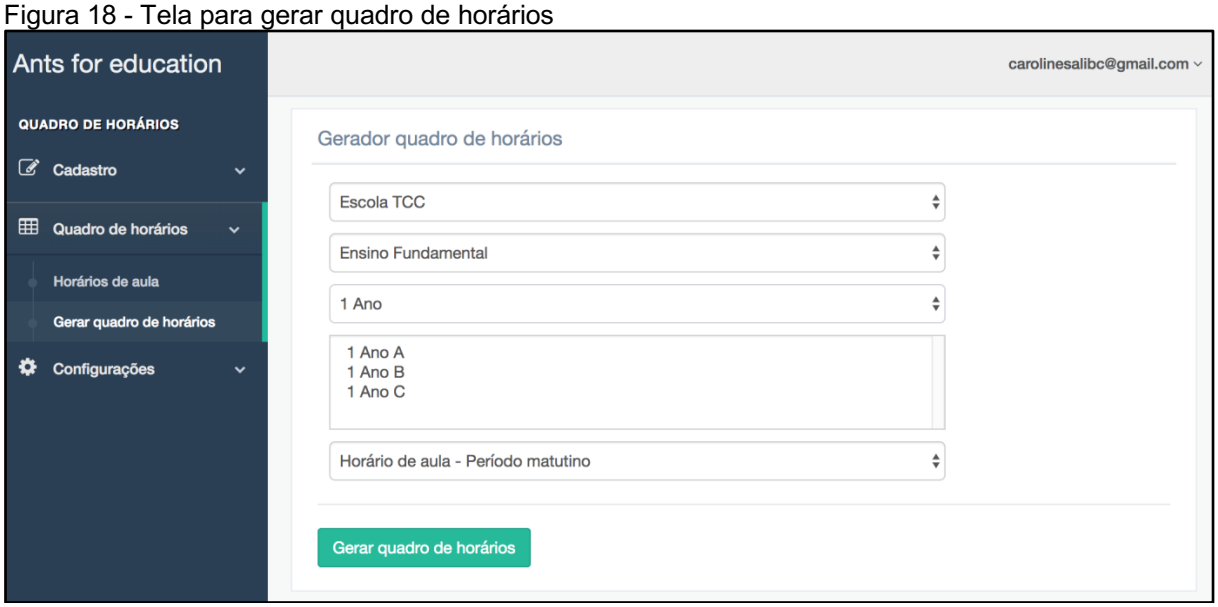


Fonte: Do autor.

Para a geração da grade horária, foi implementado uma tela (figura 18) para selecionar quais as turmas devem ser consideradas para a execução do algoritmo, sendo possível filtrá-las por escola, curso e série. O campo horário de aula representa quais os períodos e dias que estão sendo considerados para gerar o quadro de horários, sendo estes os dados que foram inseridos na tela da figura 17. Ao clicar no botão *Gerar Quadro de Horários*, uma instância do algoritmo MMAS

será iniciada, passando como argumento o conjunto de turmas selecionadas e a quantidade de aulas que cada turma tem por dia.

Figura 18 - Tela para gerar quadro de horários



Fonte: Do autor.

Ao clicar no botão para gerar o quadro de horários (figura 18), uma instância da classe MMAS que foi implementada é gerada. Na figura 19 é possível visualizar a estrutura da classe MMAS, que contém alguns valores fixos como o número de formigas que fora adicionado como 10, número de tentativas, também adicionado como 10, e o tempo limite no qual o algoritmo irá ficar em modo de execução, sendo um total de 90 segundos. O número de formigas, tentativas e tempo limite foram escolhidos com base na literatura, com referência aos trabalhos de Knowles, Sampels e Socha (2002) e Simanjuntak et al. (2012).

Figura 19 – Visão geral da classe MMAS

```
1   class MMAS
2     attr_reader :problem, :ants
3
4     NUMBER_OF_ANTS = 10
5     NUMBER_OF_TRIES = 10
6     TIME_LIMIT_SECONDS = 90
7
8     def initialize(classrooms, days, periods)
9       @problem = Problem.new(classrooms, days, periods)
10    end
11
12    def generate…
55    end
56
57    def time_passed?(time_start)…
61    end
62
63    def generate_ants…
65    end
66  end
```

Fonte: Do autor.

Logo na inicialização do MMAS, uma instância da classe *Problem* é criada, é esta classe que irá conter alguma das principais ações do algoritmo MMAS, como a inicialização dos eventos[17] e *timeslots*, evaporação e limitação do valor mínimo e máximo da matriz de feromônio. Na figura 20 tem-se os métodos existentes na classe problema. Nota-se que na inicialização da classe, algumas variáveis são iniciadas com dados pré-definidos, com base nos trabalhos de aos trabalhos de Knowles, Sampels e Socha (2002) e Simanjuntak et al. (2012), como a evaporação do feromônio, os valores mínimo e máximo do feromônio. Estes valores podem ser sobescritos na criação da classe caso seja preciso testar com outras variações de valores.

[17] Nesta implementação, um evento considera-se uma aula, ou seja, a relação de uma disciplina, lecionada por um professor em uma determinada turma e horário, sendo este último representado pelo *timeslot.*

Figura 20 – Visão geral da classe *Problem*

Fonte: Do autor.

Na Figura 21 é possível ter uma visão geral da classe *Ant*, que representa a formiga do algoritmo MMAS. Uma formiga é capaz de se mover, e depositar o feromônio que irá influenciar as outras formigas.

Quando a formiga se move, os professores são alocados aos eventos conforme as disciplinas que os mesmos lecionam. Uma formiga em movimento percorre todos os eventos existentes, associando estes a *timeslots* de forma randômica, com influencia da soma do feromônio para o evento em questão.

Figura 21 - Visão geral da classe *Ant*

```ruby
class Ant
  attr_accessor :solution

  ALPHA = 1.0
  BETHA = 2.0

  def move!(problem)
    @problem = problem
    @solution = Solution.new(@problem)
    @problem = @solution.assign_teachers(@problem)

    @problem.total_events.times do |event_index|
      sum_pheromone_for_event = @problem.sum_pheromone_for_event(event_index)
      random = Random.rand(sum_pheromone_for_event)

      total = 0.0
      timeslot = nil

      @problem.total_timeslots.times do |timeslot_index|

        total += @problem.event_timeslot_pheromone[[event_index, timeslot_index]]

        if total >= random
          timeslot = timeslot_index
          break
        end
      end

      @problem.events[event_index].timeslot = timeslot
    end

    @problem
  end

  def deposite_pheromone…
  end

  def probability(event_index, timeslot_index)…
  end

  def heuristic_information(hcv)…
  end
end
```

Fonte: Do autor.

Outra classe importante para o funcionamento do MMAS é a nomeada como *Solution* (figura 19), que representa a solução, é nesta classe que os professores são alocados nos eventos, sendo realizado o cálculo das violações de restrição. Esta classe também é responsável pelo método de busca local, que dado uma melhor solução irá tratar de melhorar a qualidade da mesma.

Figura 22 - Visão geral da classe *Solution*

```
  1  class Solution
  2    attr_accessor :hard_constraints_violations, :soft_constraints_violations
  3    attr_reader :problem
  4
  5    def initialize(problem = nil)…
 10    end
 11
 12    def assign_teachers(problem)…
 21    end
 22
 23    def calcule_hard_constraints_violations…
 33    end
 34
 35    def compute_feasibility…
 56    end
 57
 58    def local_search(problem)…
 79    end
 80
 81    def move(problem, event_index, current_hcv)…
 95    end
 96
 97    def next_timeslot(event)…
108    end
109
110    def look_for_empty_timeslot(classroom)…
122    end
123
124    def event_hard_constraints_violations(problem, event)…
130    end
131
132    def timeslots_without_events_hcv…
149    end
150
151    def duplicated_teacher_timeslot_hcv(all_events, current_event)…
163    end
164  end
```

Fonte: Do autor.

Na figura 23 é possível visualizar o método *generate*, que tem como responsabilidade criar as soluções e gerar um quadro de horários com as informações recebidas (turmas, períodos e dias). Este método possui um critério de parada, quando o número de tentativas for alcançado, e caso o tempo limite seja excedido, o algoritmo é forçadamente parado gerando uma exceção.

Ao iniciar o método, as formigas são criadas de acordo com a quantidade pré-definida. Cada vez que uma destas formigas se move, ela passa por todos os eventos e relaciona este com um *timeslot.* Ao terminar de se mover, é verificado se a solução criada pela formiga é melhor que a solução global que fora criada, de forma

aleatória, no início do método. Caso a solução gerada pela formiga atual tenha um número menor de violações a regra geral, esta solução vira a nova solução global, chamada de *best_solution*.

Figura 23 - Método *generate*

```
def generate
  time_start = Time.now
  generate_ants
  best_solution = Solution.new(@problem)

  NUMBER_OF_TRIES.times do
    @problem.reset_pheromone

    raise TimeLimitError if time_passed?(time_start)

    best_fitness = 99999
    ant_index = nil
    @ants.each_with_index do |ant, index|
      @problem = ant.move!(@problem)

      @problem.evaporate_pheromone

      fitness = ant.solution.calcule_hard_constraints_violations
      if fitness < best_fitness
        best_fitness = fitness
        ant_index = index
      end
    end

    @problem = @ants[ant_index].solution.local_search(@problem)

    feasible = @ants[ant_index].solution.compute_feasibility

    unless feasible
      @ants[ant_index].solution.calcule_hard_constraints_violations
      if @ants[ant_index].solution.hard_constraints_violations <= best_solution.hard_constraints_violations
        best_solution = @ants[ant_index].solution
        best_solution.soft_constraints_violations = 99999
      end
    end

    @ants[ant_index].solution = best_solution

    @problem.pheromone_min_max
    @problem = @ants[ant_index].deposite_pheromone
  end

  @problem
end
```

Fonte: Do autor.

Ao encontrar a melhor solução, o algoritmo executa uma busca local do tipo subida de encosta (figura 24) com o objetivo de melhorar o resultado da mesma. Basicamente, a busca local embaralha todos os eventos e passa por cada um, verificando se a solução do vizinho mais próximo é melhor. O método *move* é responsável por procurar o vizinho no *timeslot* mais próximo e calcular o número de

restrições que este vizinho tem para o evento atual. Se o evento tiver menos restrições no *timeslot* mais próximo do que tem no *timeslot* atual, então o evento vizinho passa a ser o evento atual. A busca local é executada 10 vezes ou até que a solução global seja factível. Este critério de parada foi empregado para que o algoritmo não demande muito tempo de execução na busca local, já que a mesma é executada de acordo com o número de tentativas.

Figura 24 - Algoritmo de busca local

```ruby
def local_search(problem)
  events_indexes = [*0..problem.total_events - 1].shuffle

  feasible = compute_feasibility
  step = 0

  until feasible || step == 10
    step += 1
    events_indexes.each do |event_index|
      event = problem.events[event_index]
      current_hcv = event_hard_constraints_violations(problem, event)

      next if current_hcv.zero?

      problem = move(problem, event_index, current_hcv)
    end
    feasible = compute_feasibility
  end

  problem
end

def move(problem, event_index, current_hcv)
  neighbour_problem = problem
  neighbour_event = neighbour_problem.events[event_index]

  neighbour_event.timeslot = next_timeslot(neighbour_event)
  neighbour_event_hcv = event_hard_constraints_violations(neighbour_problem, neighbour_event)

  if neighbour_event_hcv < current_hcv
    neighbour_problem.events[event_index] = neighbour_event

    return neighbour_problem
  end

  problem
end

def next_timeslot(event)
  empty_timeslot_classroom = look_for_empty_timeslot(event.lesson.classroom)
  return empty_timeslot_classroom unless empty_timeslot_classroom.nil?

  if event.timeslot < @problem.total_timeslots - 1
    event.timeslot += 1
  else
    event.timeslot = 0
  end

  event.timeslot
end
```

Fonte: Do autor.

O algoritmo que verifica se a solução é factível pode ser visto na figura 25.

Figura 25 - Algoritmo para computar solução factível

```
def compute_feasibility
  @problem.events.each do |event|
    if duplicated_teacher_timeslot_hcv(@problem.events, event) > 0
      return false
    end
  end

  # all timeslots must have an event for all classrooms
  if timeslots_without_events_hcv > 0
    return false
  end

  true
end
```

Fonte: Do autor.

No cálculo do número de restrições, considera-se as fáceis e difíceis. Nesta implementação, consideraram-se apenas as restrições difíceis, que consiste em verificar quantos professores estão alocados no mesmo *timeslot*, ou seja, quantos professores estão em dois locais ao mesmo tempo, e quantos *timeslots* não possuem eventos. Quanto maior o número destas restrições, pior a qualidade da solução como um todo. O algoritmo que calcula as restrições difíceis pode ser visualizado na figura 26.

Figura 26 - Algoritmo que calcula o número de restrições difíceis

```
def calcule_hard_constraints_violations
  hard_constraints_violations = 0

  @problem.events.each do |event|
    hard_constraints_violations += duplicated_teacher_timeslot_hcv(@problem.events, event)
  end

  hard_constraints_violations += timeslots_without_events_hcv

  @hard_constraints_violations = hard_constraints_violations
end
```

Fonte: Do autor.

Ao final de cada tentativa do algoritmo MMAS, a matriz de feromônio é atualizada, substituindo:

a) os valores menores que o valor mínimo, pelo valor mínimo;

b) os valores maiores que o máximo, pelo valor máximo.

A delimitação do feromônio pelo valor máximo e mínimo é feita para evitar a estagnação do algoritmo, onde o feromônio poderia tomar uma proporção muito

grande já no início da execução fazendo com que todas as formigas escolham o mesmo percurso.

Logo após a limitação do feromônio, a sua deposição é feita para a relação de eventos e *timeslots* que fizeram parte do caminho da melhor formiga.

7.1.4 Realização de testes

Para a realização dos testes, verificou-se a importância do desempenho do algoritmo em relação ao seu tempo de execução, bem como a qualidade do resultado, que leva em consideração principalmente se existe choque de horários entre os professores e *timeslots* vazios. Para realizar os experimentos, foi necessário inserir alguns dados no i-Educar, como escola, curso, série, turmas, disciplinas e professores, e importar estes dados para o protótipo. Os experimentos foram realizados com quantidades de 3 e 5 turmas, contendo de 5 a 10 disciplinas diferentes com um professor atrelado a cada disciplina. Os resultados podem ser vistos de forma detalhada no capítulo de apresentação e discussão dos resultados.

7.2 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Com objetivo de compreender a eficiência do algoritmo na elaboração de quadro de horários foram realizados três experimentos com quantidades diferentes de turmas e disciplinas, variando de 3 a 5 turmas com 5 a 10 disciplinas. A relação dos experimentos realizados pode ser vista na tabela 5.

Tabela 5 - Experimentos realizados

Número do experimento	Número de turmas	Número de disciplinas
1	3	5
2	3	10
3	5	10

Fonte: Do autor.

7.2.1 Experimentos de geração de grade horária

Para o primeiro experimento utilizou-se uma turma com cinco disciplinas,

sendo cada uma com um professor (tabela 6). A coluna créditos representa a quantidade de aulas daquela disciplina que cada turma tem durante a semana.

Tabela 6 - Relação de turmas, disciplinas e professores

Turma	Disciplina	Créditos	Professor
1 ano A	PORTUGUES	5	MARIA
	MATEMATICA	5	JOAO
	CIENCIAS	5	MARCOS
	GEOGRAFIA	5	DAVID
	HISTORIA	5	CLAUDIA
1 ano B	PORTUGUES	5	MARIA
	MATEMATICA	5	JOAO
	CIENCIAS	5	MARCOS
	GEOGRAFIA	5	DAVID
	HISTORIA	5	CLAUDIA
1 ano C	PORTUGUES	5	MARIA
	MATEMATICA	5	JOAO
	CIENCIAS	5	MARCOS
	GEOGRAFIA	5	DAVID
	HISTORIA	5	CLAUDIA

Fonte: Do autor.

Os parâmetros utilizados para a geração do quadro de horários podem ser vistos na tabela 7, considerando os critérios expostos anteriormente dentro da metodologia na seção de desenvolvimento do protótipo.

Tabela 7 - Lista de parâmetros

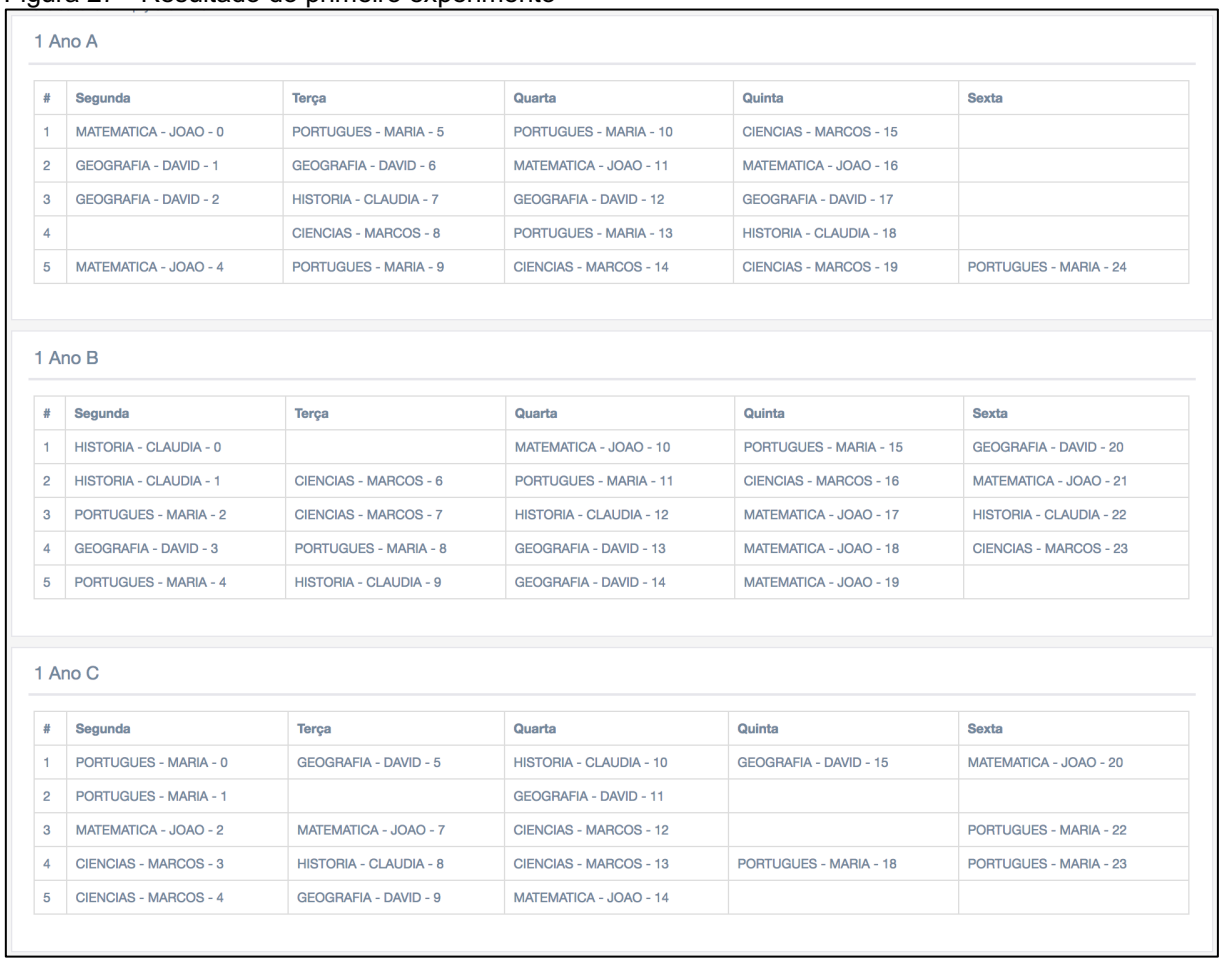
Parâmetro	Valor
Número de formigas	10
Número de tentativas	10
Tempo limite de execução	90 s
Evaporação do feromônio	1.0
Feromônio mínimo	0.1
Feromônio máximo	8.0

Fonte: Do autor.

A primeira execução do algoritmo com os dados da tabela 6 e os parâmetros da tabela 7 levou um tempo total de 5,89 segundos, tendo 13 *timeslots* vazios (o algoritmo não conseguiu alocar um evento dentro do tempo), porém nenhum professor teve choque de horários. É possível verificar o quadro de horários gerado na figura 27. Em relação ao tempo de execução, deve levar-se em consideração que a máquina utilizada na realização dos testes contém um

processador com 2,7 GHz da Intel modelo Core i5 e 8 GB de memória RAM DDR3. A mesma máquina foi utilizada para os demais testes, portanto considera-se sempre a mesma configuração de hardware quanto ao desempenho dos experimentos.

Figura 27 - Resultado do primeiro experimento



1 Ano A

#	Segunda	Terça	Quarta	Quinta	Sexta
1	MATEMATICA - JOAO - 0	PORTUGUES - MARIA - 5	PORTUGUES - MARIA - 10	CIENCIAS - MARCOS - 15	
2	GEOGRAFIA - DAVID - 1	GEOGRAFIA - DAVID - 6	MATEMATICA - JOAO - 11	MATEMATICA - JOAO - 16	
3	GEOGRAFIA - DAVID - 2	HISTORIA - CLAUDIA - 7	GEOGRAFIA - DAVID - 12	GEOGRAFIA - DAVID - 17	
4		CIENCIAS - MARCOS - 8	PORTUGUES - MARIA - 13	HISTORIA - CLAUDIA - 18	
5	MATEMATICA - JOAO - 4	PORTUGUES - MARIA - 9	CIENCIAS - MARCOS - 14	CIENCIAS - MARCOS - 19	PORTUGUES - MARIA - 24

1 Ano B

#	Segunda	Terça	Quarta	Quinta	Sexta
1	HISTORIA - CLAUDIA - 0		MATEMATICA - JOAO - 10	PORTUGUES - MARIA - 15	GEOGRAFIA - DAVID - 20
2	HISTORIA - CLAUDIA - 1	CIENCIAS - MARCOS - 6	PORTUGUES - MARIA - 11	CIENCIAS - MARCOS - 16	MATEMATICA - JOAO - 21
3	PORTUGUES - MARIA - 2	CIENCIAS - MARCOS - 7	HISTORIA - CLAUDIA - 12	MATEMATICA - JOAO - 17	HISTORIA - CLAUDIA - 22
4	GEOGRAFIA - DAVID - 3	PORTUGUES - MARIA - 8	GEOGRAFIA - DAVID - 13	MATEMATICA - JOAO - 18	CIENCIAS - MARCOS - 23
5	PORTUGUES - MARIA - 4	HISTORIA - CLAUDIA - 9	GEOGRAFIA - DAVID - 14	MATEMATICA - JOAO - 19	

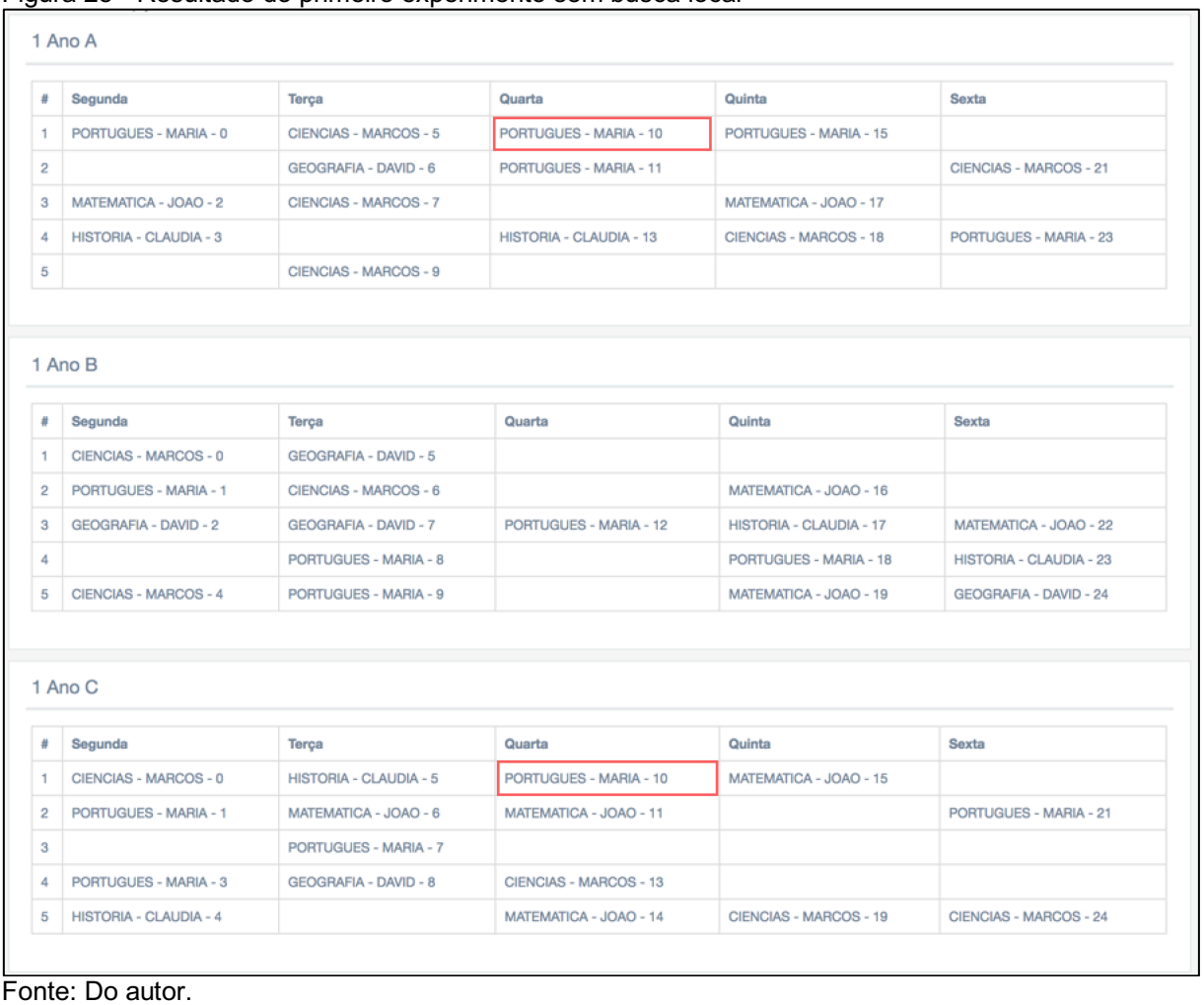
1 Ano C

#	Segunda	Terça	Quarta	Quinta	Sexta
1	PORTUGUES - MARIA - 0	GEOGRAFIA - DAVID - 5	HISTORIA - CLAUDIA - 10	GEOGRAFIA - DAVID - 15	MATEMATICA - JOAO - 20
2	PORTUGUES - MARIA - 1		GEOGRAFIA - DAVID - 11		
3	MATEMATICA - JOAO - 2	MATEMATICA - JOAO - 7	CIENCIAS - MARCOS - 12		PORTUGUES - MARIA - 22
4	CIENCIAS - MARCOS - 3	HISTORIA - CLAUDIA - 8	CIENCIAS - MARCOS - 13	PORTUGUES - MARIA - 18	PORTUGUES - MARIA - 23
5	CIENCIAS - MARCOS - 4	GEOGRAFIA - DAVID - 9	MATEMATICA - JOAO - 14		

Fonte: Do autor.

Sem o uso da busca local o tempo de execução pode ser um pouco menor, neste caso com um pouco mais de 1 segundo de diferença, porém como pode ser visto na figura 28, o resultado do quadro de horários apresenta um número de *timeslots* consideravelmente maior do que sem o uso de busca local, tendo praticamente o dobro de *timeslots* vazios. Além disso, uma perda na qualidade das alocações pode ser considerada, como por exemplo, a professora Maria, que está alocada para ensinar a disciplina de português na quarta-feira em duas turmas diferentes no mesmo período.

Figura 28 - Resultado do primeiro experimento sem busca local



1 Ano A

#	Segunda	Terça	Quarta	Quinta	Sexta
1	PORTUGUES - MARIA - 0	CIENCIAS - MARCOS - 5	PORTUGUES - MARIA - 10	PORTUGUES - MARIA - 15	
2		GEOGRAFIA - DAVID - 6	PORTUGUES - MARIA - 11		CIENCIAS - MARCOS - 21
3	MATEMATICA - JOAO - 2	CIENCIAS - MARCOS - 7		MATEMATICA - JOAO - 17	
4	HISTORIA - CLAUDIA - 3		HISTORIA - CLAUDIA - 13	CIENCIAS - MARCOS - 18	PORTUGUES - MARIA - 23
5		CIENCIAS - MARCOS - 9			

1 Ano B

#	Segunda	Terça	Quarta	Quinta	Sexta
1	CIENCIAS - MARCOS - 0	GEOGRAFIA - DAVID - 5			
2	PORTUGUES - MARIA - 1	CIENCIAS - MARCOS - 6		MATEMATICA - JOAO - 16	
3	GEOGRAFIA - DAVID - 2	GEOGRAFIA - DAVID - 7	PORTUGUES - MARIA - 12	HISTORIA - CLAUDIA - 17	MATEMATICA - JOAO - 22
4		PORTUGUES - MARIA - 8		PORTUGUES - MARIA - 18	HISTORIA - CLAUDIA - 23
5	CIENCIAS - MARCOS - 4	PORTUGUES - MARIA - 9		MATEMATICA - JOAO - 19	GEOGRAFIA - DAVID - 24

1 Ano C

#	Segunda	Terça	Quarta	Quinta	Sexta
1	CIENCIAS - MARCOS - 0	HISTORIA - CLAUDIA - 5	PORTUGUES - MARIA - 10	MATEMATICA - JOAO - 15	
2	PORTUGUES - MARIA - 1	MATEMATICA - JOAO - 6	MATEMATICA - JOAO - 11		PORTUGUES - MARIA - 21
3		PORTUGUES - MARIA - 7			
4	PORTUGUES - MARIA - 3	GEOGRAFIA - DAVID - 8	CIENCIAS - MARCOS - 13		
5	HISTORIA - CLAUDIA - 4		MATEMATICA - JOAO - 14	CIENCIAS - MARCOS - 19	CIENCIAS - MARCOS - 24

Fonte: Do autor.

No segundo experimento foi utilizado a mesma quantidade de turmas, porém com o dobro de disciplinas para cada turma, como pode ser visto na tabela 8.

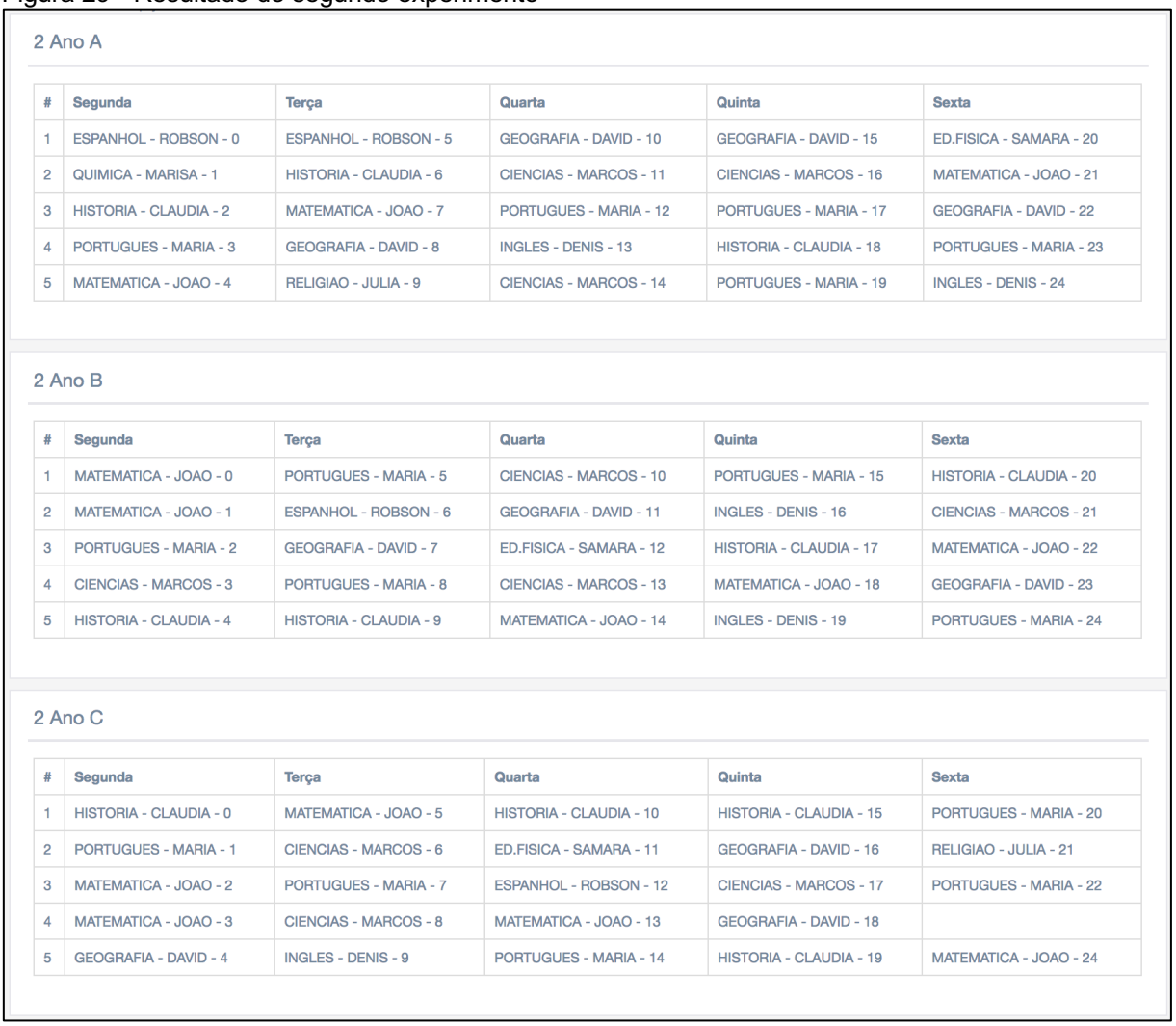
Tabela 8 - Relação de turmas, disciplinas e professores (segundo experimento)

Turma	Disciplina	Créditos	Professor
2 ano A	PORTUGUES	3	MARIA
	MATEMATICA	3	JOAO
	CIENCIAS	3	MARCOS
	GEOGRAFIA	3	DAVID
	HISTORIA	3	CLAUDIA
	INGLES	2	DENIS
	ESPANHOL	2	ROBSON
	ED. FISICA	2	SAMARA
	RELIGIAO	2	JULIA
	QUIMICA	2	MARISA
2 ano B	PORTUGUES	3	MARIA
	MATEMATICA	3	JOAO
	CIENCIAS	3	MARCOS
	GEOGRAFIA	3	DAVID
	HISTORIA	3	CLAUDIA
	INGLES	2	DENIS
	ESPANHOL	2	ROBSON
	ED. FISICA	2	SAMARA
	RELIGIAO	2	JULIA
	QUIMICA	2	MARISA
2 ano C	PORTUGUES	3	MARIA
	MATEMATICA	3	JOAO
	CIENCIAS	3	MARCOS
	GEOGRAFIA	3	DAVID
	HISTORIA	3	CLAUDIA
	INGLES	2	DENIS
	ESPANHOL	2	ROBSON
	ED. FISICA	2	SAMARA
	RELIGIAO	2	JULIA
	QUIMICA	2	MARISA

Fonte: Do autor.

O segundo experimento levou um total de 8,27 segundos para executar, e teve apenas um *timeslot* que não fora alocado com evento. Nesta segunda realização dos testes, apesar de ter mais disciplinas e professores, nenhum professor foi alocado para dar aula em dois períodos ao mesmo tempo (figura 29).

Figura 29 - Resultado do segundo experimento



2 Ano A

#	Segunda	Terça	Quarta	Quinta	Sexta
1	ESPANHOL - ROBSON - 0	ESPANHOL - ROBSON - 5	GEOGRAFIA - DAVID - 10	GEOGRAFIA - DAVID - 15	ED.FISICA - SAMARA - 20
2	QUIMICA - MARISA - 1	HISTORIA - CLAUDIA - 6	CIENCIAS - MARCOS - 11	CIENCIAS - MARCOS - 16	MATEMATICA - JOAO - 21
3	HISTORIA - CLAUDIA - 2	MATEMATICA - JOAO - 7	PORTUGUES - MARIA - 12	PORTUGUES - MARIA - 17	GEOGRAFIA - DAVID - 22
4	PORTUGUES - MARIA - 3	GEOGRAFIA - DAVID - 8	INGLES - DENIS - 13	HISTORIA - CLAUDIA - 18	PORTUGUES - MARIA - 23
5	MATEMATICA - JOAO - 4	RELIGIAO - JULIA - 9	CIENCIAS - MARCOS - 14	PORTUGUES - MARIA - 19	INGLES - DENIS - 24

2 Ano B

#	Segunda	Terça	Quarta	Quinta	Sexta
1	MATEMATICA - JOAO - 0	PORTUGUES - MARIA - 5	CIENCIAS - MARCOS - 10	PORTUGUES - MARIA - 15	HISTORIA - CLAUDIA - 20
2	MATEMATICA - JOAO - 1	ESPANHOL - ROBSON - 6	GEOGRAFIA - DAVID - 11	INGLES - DENIS - 16	CIENCIAS - MARCOS - 21
3	PORTUGUES - MARIA - 2	GEOGRAFIA - DAVID - 7	ED.FISICA - SAMARA - 12	HISTORIA - CLAUDIA - 17	MATEMATICA - JOAO - 22
4	CIENCIAS - MARCOS - 3	PORTUGUES - MARIA - 8	CIENCIAS - MARCOS - 13	MATEMATICA - JOAO - 18	GEOGRAFIA - DAVID - 23
5	HISTORIA - CLAUDIA - 4	HISTORIA - CLAUDIA - 9	MATEMATICA - JOAO - 14	INGLES - DENIS - 19	PORTUGUES - MARIA - 24

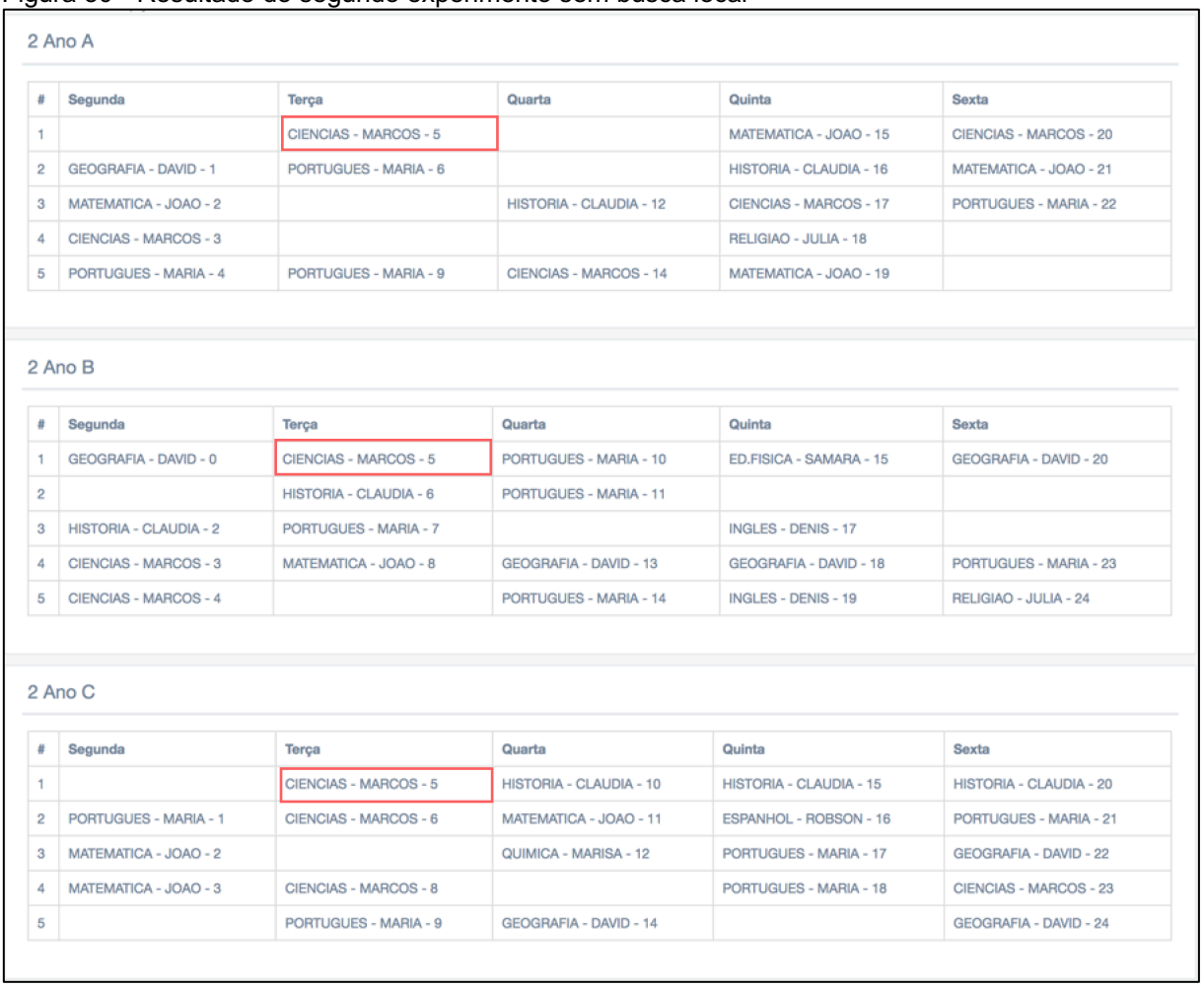
2 Ano C

#	Segunda	Terça	Quarta	Quinta	Sexta
1	HISTORIA - CLAUDIA - 0	MATEMATICA - JOAO - 5	HISTORIA - CLAUDIA - 10	HISTORIA - CLAUDIA - 15	PORTUGUES - MARIA - 20
2	PORTUGUES - MARIA - 1	CIENCIAS - MARCOS - 6	ED.FISICA - SAMARA - 11	GEOGRAFIA - DAVID - 16	RELIGIAO - JULIA - 21
3	MATEMATICA - JOAO - 2	PORTUGUES - MARIA - 7	ESPANHOL - ROBSON - 12	CIENCIAS - MARCOS - 17	PORTUGUES - MARIA - 22
4	MATEMATICA - JOAO - 3	CIENCIAS - MARCOS - 8	MATEMATICA - JOAO - 13	GEOGRAFIA - DAVID - 18	
5	GEOGRAFIA - DAVID - 4	INGLES - DENIS - 9	PORTUGUES - MARIA - 14	HISTORIA - CLAUDIA - 19	MATEMATICA - JOAO - 24

Fonte: o autor.

Sem o uso da busca local, o segundo experimento foi executado em 5,82 segundos e teve 19 *timeslots* vazios, além de que entre os professores dos eventos, alguns estão alocados para dar aula no mesmo período e no mesmo dia, como no caso do professor Marcos de Ciências, alocado em três turmas no primeiro período da terça-feira.

Figura 30 - Resultado do segundo experimento sem busca local



2 Ano A

#	Segunda	Terça	Quarta	Quinta	Sexta
1		CIENCIAS - MARCOS - 5		MATEMATICA - JOAO - 15	CIENCIAS - MARCOS - 20
2	GEOGRAFIA - DAVID - 1	PORTUGUES - MARIA - 6		HISTORIA - CLAUDIA - 16	MATEMATICA - JOAO - 21
3	MATEMATICA - JOAO - 2		HISTORIA - CLAUDIA - 12	CIENCIAS - MARCOS - 17	PORTUGUES - MARIA - 22
4	CIENCIAS - MARCOS - 3			RELIGIAO - JULIA - 18	
5	PORTUGUES - MARIA - 4	PORTUGUES - MARIA - 9	CIENCIAS - MARCOS - 14	MATEMATICA - JOAO - 19	

2 Ano B

#	Segunda	Terça	Quarta	Quinta	Sexta
1	GEOGRAFIA - DAVID - 0	CIENCIAS - MARCOS - 5	PORTUGUES - MARIA - 10	ED.FISICA - SAMARA - 15	GEOGRAFIA - DAVID - 20
2		HISTORIA - CLAUDIA - 6	PORTUGUES - MARIA - 11		
3	HISTORIA - CLAUDIA - 2	PORTUGUES - MARIA - 7		INGLES - DENIS - 17	
4	CIENCIAS - MARCOS - 3	MATEMATICA - JOAO - 8	GEOGRAFIA - DAVID - 13	GEOGRAFIA - DAVID - 18	PORTUGUES - MARIA - 23
5	CIENCIAS - MARCOS - 4		PORTUGUES - MARIA - 14	INGLES - DENIS - 19	RELIGIAO - JULIA - 24

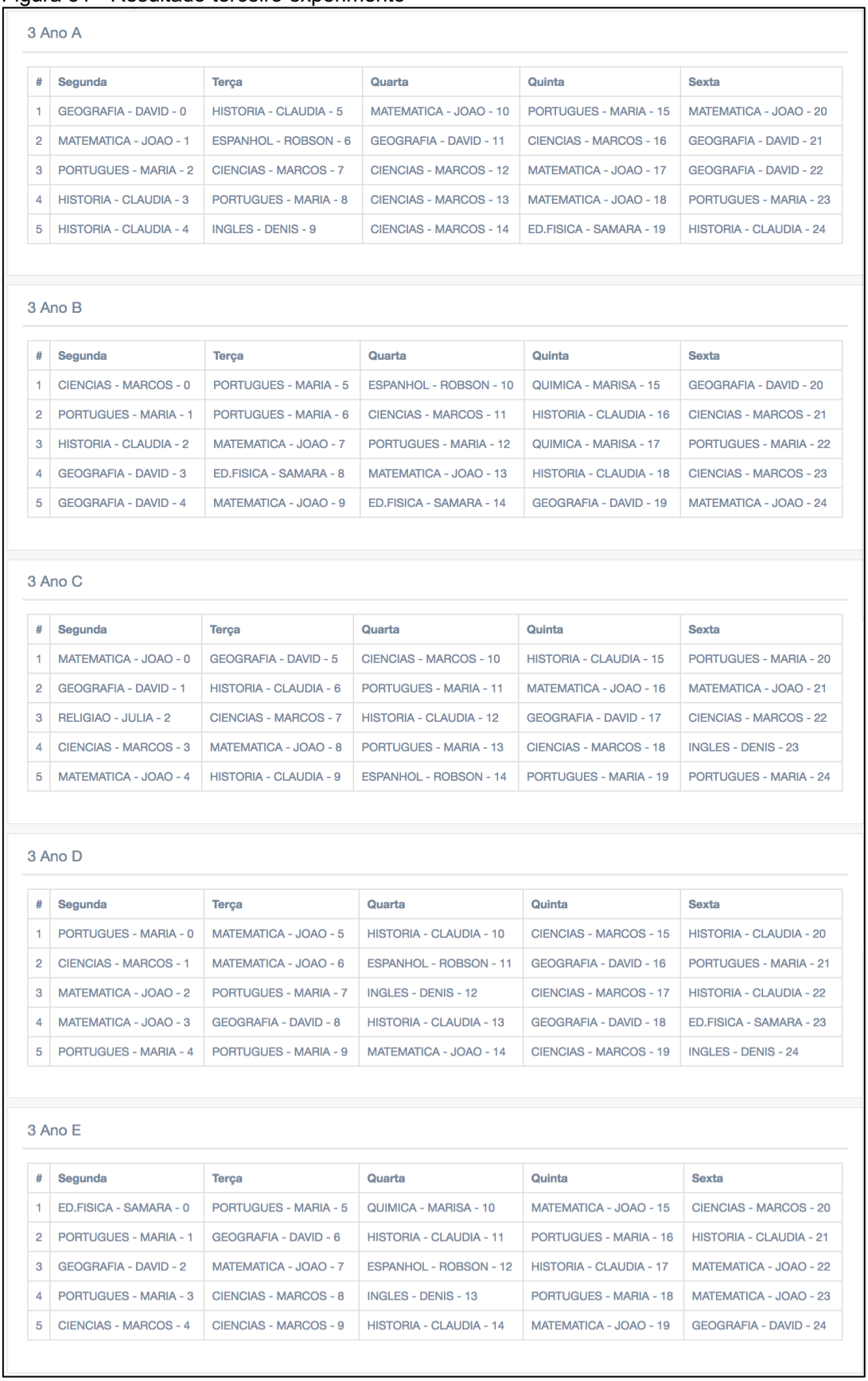
2 Ano C

#	Segunda	Terça	Quarta	Quinta	Sexta
1		CIENCIAS - MARCOS - 5	HISTORIA - CLAUDIA - 10	HISTORIA - CLAUDIA - 15	HISTORIA - CLAUDIA - 20
2	PORTUGUES - MARIA - 1	CIENCIAS - MARCOS - 6	MATEMATICA - JOAO - 11	ESPANHOL - ROBSON - 16	PORTUGUES - MARIA - 21
3	MATEMATICA - JOAO - 2		QUIMICA - MARISA - 12	PORTUGUES - MARIA - 17	GEOGRAFIA - DAVID - 22
4	MATEMATICA - JOAO - 3	CIENCIAS - MARCOS - 8		PORTUGUES - MARIA - 18	CIENCIAS - MARCOS - 23
5		PORTUGUES - MARIA - 9	GEOGRAFIA - DAVID - 14		GEOGRAFIA - DAVID - 24

Fonte: Do autor.

O mesmo experimento da tabela 8 foi realizado com mais turmas, obtendo resultados muito parecidos (figura 31). Devido ao comportamento do algoritmo, cada vez que ele é executado, mesmo utilizando os mesmos parâmetros de inicialização, sempre será gerado um resultado diferente, ou seja, por mais que os parâmetros sejam os mesmos, a solução pode as vezes ser perfeita ou não. Esta divergência ocorre, pois, os resultados são sempre gerados de forma aleatória no inicio do algoritmo, criando uma solução “ruim”, e é a partir desta solução que ocorre a otimização, partindo sempre de um resultado pré-definido, aleatório e único.

Figura 31 - Resultado terceiro experimento



3 Ano A

#	Segunda	Terça	Quarta	Quinta	Sexta
1	GEOGRAFIA - DAVID - 0	HISTORIA - CLAUDIA - 5	MATEMATICA - JOAO - 10	PORTUGUES - MARIA - 15	MATEMATICA - JOAO - 20
2	MATEMATICA - JOAO - 1	ESPANHOL - ROBSON - 6	GEOGRAFIA - DAVID - 11	CIENCIAS - MARCOS - 16	GEOGRAFIA - DAVID - 21
3	PORTUGUES - MARIA - 2	CIENCIAS - MARCOS - 7	CIENCIAS - MARCOS - 12	MATEMATICA - JOAO - 17	GEOGRAFIA - DAVID - 22
4	HISTORIA - CLAUDIA - 3	PORTUGUES - MARIA - 8	CIENCIAS - MARCOS - 13	MATEMATICA - JOAO - 18	PORTUGUES - MARIA - 23
5	HISTORIA - CLAUDIA - 4	INGLES - DENIS - 9	CIENCIAS - MARCOS - 14	ED.FISICA - SAMARA - 19	HISTORIA - CLAUDIA - 24

3 Ano B

#	Segunda	Terça	Quarta	Quinta	Sexta
1	CIENCIAS - MARCOS - 0	PORTUGUES - MARIA - 5	ESPANHOL - ROBSON - 10	QUIMICA - MARISA - 15	GEOGRAFIA - DAVID - 20
2	PORTUGUES - MARIA - 1	PORTUGUES - MARIA - 6	CIENCIAS - MARCOS - 11	HISTORIA - CLAUDIA - 16	CIENCIAS - MARCOS - 21
3	HISTORIA - CLAUDIA - 2	MATEMATICA - JOAO - 7	PORTUGUES - MARIA - 12	QUIMICA - MARISA - 17	PORTUGUES - MARIA - 22
4	GEOGRAFIA - DAVID - 3	ED.FISICA - SAMARA - 8	MATEMATICA - JOAO - 13	HISTORIA - CLAUDIA - 18	CIENCIAS - MARCOS - 23
5	GEOGRAFIA - DAVID - 4	MATEMATICA - JOAO - 9	ED.FISICA - SAMARA - 14	GEOGRAFIA - DAVID - 19	MATEMATICA - JOAO - 24

3 Ano C

#	Segunda	Terça	Quarta	Quinta	Sexta
1	MATEMATICA - JOAO - 0	GEOGRAFIA - DAVID - 5	CIENCIAS - MARCOS - 10	HISTORIA - CLAUDIA - 15	PORTUGUES - MARIA - 20
2	GEOGRAFIA - DAVID - 1	HISTORIA - CLAUDIA - 6	PORTUGUES - MARIA - 11	MATEMATICA - JOAO - 16	MATEMATICA - JOAO - 21
3	RELIGIAO - JULIA - 2	CIENCIAS - MARCOS - 7	HISTORIA - CLAUDIA - 12	GEOGRAFIA - DAVID - 17	CIENCIAS - MARCOS - 22
4	CIENCIAS - MARCOS - 3	MATEMATICA - JOAO - 8	PORTUGUES - MARIA - 13	CIENCIAS - MARCOS - 18	INGLES - DENIS - 23
5	MATEMATICA - JOAO - 4	HISTORIA - CLAUDIA - 9	ESPANHOL - ROBSON - 14	PORTUGUES - MARIA - 19	PORTUGUES - MARIA - 24

3 Ano D

#	Segunda	Terça	Quarta	Quinta	Sexta
1	PORTUGUES - MARIA - 0	MATEMATICA - JOAO - 5	HISTORIA - CLAUDIA - 10	CIENCIAS - MARCOS - 15	HISTORIA - CLAUDIA - 20
2	CIENCIAS - MARCOS - 1	MATEMATICA - JOAO - 6	ESPANHOL - ROBSON - 11	GEOGRAFIA - DAVID - 16	PORTUGUES - MARIA - 21
3	MATEMATICA - JOAO - 2	PORTUGUES - MARIA - 7	INGLES - DENIS - 12	CIENCIAS - MARCOS - 17	HISTORIA - CLAUDIA - 22
4	MATEMATICA - JOAO - 3	GEOGRAFIA - DAVID - 8	HISTORIA - CLAUDIA - 13	GEOGRAFIA - DAVID - 18	ED.FISICA - SAMARA - 23
5	PORTUGUES - MARIA - 4	PORTUGUES - MARIA - 9	MATEMATICA - JOAO - 14	CIENCIAS - MARCOS - 19	INGLES - DENIS - 24

3 Ano E

#	Segunda	Terça	Quarta	Quinta	Sexta
1	ED.FISICA - SAMARA - 0	PORTUGUES - MARIA - 5	QUIMICA - MARISA - 10	MATEMATICA - JOAO - 15	CIENCIAS - MARCOS - 20
2	PORTUGUES - MARIA - 1	GEOGRAFIA - DAVID - 6	HISTORIA - CLAUDIA - 11	PORTUGUES - MARIA - 16	HISTORIA - CLAUDIA - 21
3	GEOGRAFIA - DAVID - 2	MATEMATICA - JOAO - 7	ESPANHOL - ROBSON - 12	HISTORIA - CLAUDIA - 17	MATEMATICA - JOAO - 22
4	PORTUGUES - MARIA - 3	CIENCIAS - MARCOS - 8	INGLES - DENIS - 13	PORTUGUES - MARIA - 18	MATEMATICA - JOAO - 23
5	CIENCIAS - MARCOS - 4	CIENCIAS - MARCOS - 9	HISTORIA - CLAUDIA - 14	MATEMATICA - JOAO - 19	GEOGRAFIA - DAVID - 24

Fonte: Do autor.

Na tabela 9 tem-se uma comparação dos resultados obtidos para cada experimento. Em relação aos dados da tabela, compreende-se que dentre os testes realizados, o algoritmo teve resultados mais satisfatórios quando existiam mais turmas e disciplinas em sua composição. Entende-se também, que o uso da busca local se torna um elemento indispensável em relação a qualidade da solução.

Tabela 9 - Relação de tempo de execução e *timeslots* sobre os experimentos

Número do experimento	Número de turmas	Número de disciplinas	Tempo de execução	*Timeslots* vazios
1	3	5	5,89	13
1 (busca local)	3	5	4,91	27
2	3	10	8,27	1
2 (busca local)	3	10	5,82	19
3	5	10	8,11	0

Fonte: Do autor.

7.2.2 Análise de desempenho e qualidade

Para a análise de desempenho e qualidade foi feito um comparativo considerando a quantidade de turmas, disciplinas, formigas, tentativas, feromônio mínimo, feromônio máximo e a evaporação do feromônio. Com a variação destes valores, foi analisado qual o tempo de execução e o número de restrições violadas, do inglês *Hard Constrains Violations* (HCV). Este tipo de comparação permite compreender a influência dos parâmetros na obtenção de bons resultados, e mediante a estes dados, pode ser tomado direcionamentos a fim de obter um melhor desempenho ou uma melhor qualidade do resultado final.

Na tabela 10 é representada uma comparação dos testes realizados com diferentes parâmetros e quantidades de informações. As duas colunas ao final da tabela representam o tempo que levou para gerar a solução e a quantidade de restrições que o algoritmo não conseguiu resolver no final de sua execução.

Tabela 10 - Análise de desempenho e qualidade

Quantidade turmas	Quantidade disciplinas	Quantidade formigas	Quantidade tentativas	Feromônio mínimo	Feromônio máximo	Evaporação	Tempo execução	HCV
5	10	10	10	0,1	8	1	24,61	6
5	10	10	10	0,0078	3,3	0,3	25,36	6
5	10	5	5	0,1	8	1	9,25	10
5	10	5	5	0,0078	3,3	0,3	9,44	10
3	10	10	10	0,1	8	1	7,52	2
3	10	10	10	0,0078	3,3	0,3	8,68	2
3	10	5	5	0,1	8	1	2,35	2
3	10	5	5	0,0078	3,3	0,3	3,09	2
3	5	10	10	0,1	8	1	5,84	25
3	5	10	10	0,0078	3,3	0,3	5,53	25
3	5	5	5	0,1	8	1	2,20	25
3	5	5	5	0,0078	3,3	0,3	2,20	25

Fonte: Do autor.

Após a análise de desempenho e qualidade, percebeu-se que a variação do parâmetro de quantidade de formigas e tentativas pode influenciar no tempo de execução do algoritmo, sendo que em alguns casos o algoritmo levou mais que o dobro do tempo de execução quando modificado o número formigas e tentativas de 5 para 10, não tendo diferenças quanto ao número de restrições, com exceção do exemplo com 5 turmas e 10 disciplinas, onde o número de restrições violadas aumentou de 6 para 10 quando diminuiu o número de formigas e tentativas de 10 para 5. Referente aos parâmetros feromônio mínimo, feromônio máximo e evaporação, compreende-se que estes não tiveram tanta influência no tempo de execução e quantidade de restrições, tendo uma diferença de menos de um segundo no tempo de execução em alguns experimentos, e nenhuma diferença quanto ao número de HCVs.

7.2.3 Discussão dos resultados

As técnicas de otimização por colônia de formigas têm sido cada vez mais empregadas em estudos para resolução de problemas de otimização combinatória, como no caso da geração de grade horária, onde destaca-se os métodos com MMAS e Busca Local em problemas de *University Course Timetabling Problem* (UCTP). Alguns trabalhos realizados para resolução de UCTP com MMAS são encontrados nas pesquisas de Hicks, Pongcharoen e Thepphakorn (2014); Knowles, Sampels e Socha (2002)

No trabalho de Knowles, Sampels e Socha (2002), foi utilizado o algoritmo

MMAS com uma Busca Local básica, confirmando ser um tratamento bastante simples para o problema de horários. Na implementação dos autores, eles utilizaram os parâmetros de feromônio mínimo 0.0078, feromônio máximo 3.3 e evaporação 0.3, sendo que nos experimentos realizados nesta pesquisa com estes valores, como pode ser visto na tabela 10, teve-se uma perda pequena de desempenho. Confirma-se então que os valores de feromônio mínimo 0.1, feromônio máximo 8 e evaporação 1, para o modelo de quadro de horários utilizado nesta pesquisa, teve um resultado melhor do que o modelo representado na pesquisa de Knowles, Sampels e Socha (2002).

Já na pesquisa de Simanjuntak et al. (2012), foi utilizado o algoritmo MMAS com Busca Local. Com uso dos parâmetros de feromônio mínimo 0.1, feromônio máximo 3.3 e evaporação 0.3, os autores tiveram um bom desempenho na execução do algoritmo quando para resolução de problemas pequenos de quadro de horários, levando um total de 0.4 segundos nesta categoria. No presente trabalho, o tempo mínimo para execução do algoritmo foi de 2.2 segundos, o que leva a crer que o algoritmo desenvolvido por Simanjuntak et al. (2012) teve um desempenho melhor para o problema proposto em relação a velocidade de execução.

No contexto brasileiro, existem iniciativas para resolução de grade horária nas instituições de ensino, utilizando algoritmos genéticos (MELO, 2003), método *Greedy Randomized Adptative Search Procedure* (GRASP) (SCALDELAI; VITOR, 2014) e *Iterated Local Search, Discrete Particle Swarm Optimization* e *Jump Frog Optimization* (ALVES, 2010).

Ao que compete a trabalhos científicos brasileiros, não foi encontrado trabalhos que resolvam problema de grade horária por meio do uso de MMAS, porém em comparação com os trabalhos que estudam GRASP, percebe-se que os resultados obtidos por meio do uso de MMAS são também satisfatórios.

Com relação aos trabalhos que utilizam a mesma tecnologia, o MMAS com Busca Local, pode-se confirmar que o uso da técnica traz bons resultados para problemas pequenos de grade horária, com tempo satisfatório levando alguns poucos segundos para execução completa do algoritmo. O uso desta metodologia também enfatiza que o uso da busca local se faz necessário para que se encontre resultados de melhor qualidade.

8 CONCLUSAO

A fim de se obter melhores resultados em problemas de otimização pode-se aplicar técnicas de inteligência coletiva, como por exemplo, algoritmos de colônia de formigas. Este tem um papel importante na tomada de decisão quando se trata de uma otimização combinatória, como no caso do problema de geração de quadro de horários.

Após os estudos sobre o problema de geração de grade horária, entendeu-se que por meio de abordagens da inteligência coletiva, é possível que sejam encontradas boas soluções com base na tomada de decisão ao alocar eventos em *timeslots,* respeitando ao máximo as restrições empregadas.

Com o desenvolvimento da API no i-Educar foi possível coletar dados para a implementação do algoritmo baseado em otimização por colônia de formigas. Fazendo uso do algoritmo MMAS, buscou-se um trajeto otimizado ao alocar eventos em *timeslots* que melhor se adequam a situação atual deste evento. A aplicação da busca local, do tipo subida de encosta, no algoritmo de MMAS fez com que a qualidade dos resultados melhorasse de forma significativa, fazendo da junção dos dois algoritmos, uma boa solução baseada na otimização por colônia de formigas para geração de quadro de horários.

Avaliou-se que o tempo de execução foi satisfatório em todos os testes, percebendo que em casos com mais variações de disciplinas, obteve-se resultados melhores e com poucas violações das restrições difíceis. Neste aspecto, afirma-se que o presente algoritmo é um bom candidato para resolver problemas de otimização combinatória de médio porte, com poucas restrições difíceis.

Mesmo levando em consideração as dificuldades que foram encontradas ao longo desta pesquisa, os objetivos foram alcançados, possibilitando que os municípios que operam com software i-Educar possam utilizar o protótipo desenvolvido para gerar quadro de horários de forma automática e otimizada.

Com base no conhecimento adquirido nesta pesquisa, propõe-se então algumas sugestões de trabalhos futuros a fim de dar continuidade no desenvolvimento do protótipo para geração de grade horária no i-Educar:

a) estudar e implementar outros algoritmos de ACO, tais como *Ant System* (AS), *Best-Worst Ant System* (BWAS), *Rank-Based Ant System*

(RBAS)...;

b) estudar e implementar outras formas de inteligência coletiva, como por exemplo: colônia de abelhas, otimização por enxame de partículas, *shuffled frog-leaping*;
c) comparar soluções implementadas com outros algoritmos para fazer uma análise de desempenho e qualidade;
d) expandir a solução para que gere o quadro de horários para uma escola inteira.

REFERÊNCIAS

ADAMS, Mike J. **Chemometrics in Analytical Spectroscopy.** 2. ed. [s.l.]: Royal Society Of Chemistry, 2004. 238 p. (RSC Analytical Spectroscopy Series (Book 8)).

ALLEGRINI, Franco; OLIVIERI, Alejandro C.. A new and efficient variable selection algorithm based on ant colony optimization. Applications to near infrared spectroscopy/partial least-squares analysis. **Analytica Chimica Acta**, [s.l.], v. 699, n. 1, p.18-25, ago. 2011. Elsevier BV/.

ALVES, Reginaldo Heidder de Jesus. **Metaheurísticas Aplicadas ao Problema de Horário Escolar.** 2010. 68 f. Dissertação (Mestrado) - Curso de Mestrado em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2010. Disponível em: <http://www.files.scire.net.br/atrio/cefet-mg-ppgmmc_upl/THESIS/22/reginaldoheidderdejesusalves.pdf>. Acesso em: 21 ago. 2016.

ASSIS, Thiago Albuquerque de et al. Geometria fractal: propriedades e características de fractais ideais. **Revista Brasileira de Ensino de Física**, [s.l.], v. 30, n. 2, p.2304.1-2304.10, 2008. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s1806-11172008000200005.

BARBOSA, Roberto Cavalcante. **Aplicação Da Metaheurística Algoritimo Genético Na Otimização Das Rotas De Entregas Da Distribuição Física De Produtos No Município De Fortaleza.** 2014. 91 f. Dissertação (Mestrado) - Curso de LogÍstica e Pesquisa Operacional, Universidade Federal do Ceará, Fortaleza, 2014. Disponível em: <http://repositorio.ufc.br/handle/riufc/15038>. Acesso em: 24 ago. 2016.

BEMBEM, Angela Halen Claro; SANTOS, Plácida Leopoldina V. Amorim da Cos. Inteligência coletiva: um olhar sobre a produção de Pierre Lévy. **Perspectivas em Ciência da Informação**, [s.l.], v. 18, n. 4, p.139-151, dez. 2013.

BLUM, Christian. Ant colony optimization: Introduction and recent trends. **Physics Of Life Reviews**, [s.l.], v. 2, n. 4, p.353-373, dez. 2005. Elsevier BV.

BLUM, Christian; ROLI, Andrea; DORIGO, Marco. HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization. Mic'2001 - 4th Metaheuristics International Conference, Belgium, v. 20, n. 16, p.399-403, jun. 2001.

BULLNHEIMER, Bernd; HARTL, Richard F.; STRAUSS, Christine. A new rank based version of the Ant System. A computational study. 1997.

BRABAZON, Anthony; O'NEILL, Michael; MCGARRAGHY, Seán. **Natural Computing Algorithms.** London: Springer, 2015.

BRASIL. Ministério da Educação. Sistema público permite redução de gastos em cidade potiguar. 2015. Disponível em: <http://portal.mec.gov.br/ultimas-noticias/209-564834057/30851-sistema-publico-permite-reducao-de-gastos-em-cidade-potiguar>. Acesso em: 29 ago. 2016.

BRASIL. **Software Público tem programa de gestão educacional para municípios.** 2008. Disponível em: <http://www.planejamento.gov.br/assuntos/logistica-e-tecnologia-da-informacao/noticias/software-publico-tem-programa-de-gestao>. Acesso em: 31 maio 2018.

CESAR, Andréia Luiza Corrêa. **IMPACTOS SOCIAIS E ECONÔMICOS DA ADOÇÃO DO SOFTWARE PÚBLICO i-EDUCAR NA GESTÃO ESCOLAR.** 2012. 84 f. Tese (Doutorado) - Curso de Administração, Universidade de Brasília, Brasília, 2012.

COPPIN, Ben. **Inteligência Artificial.** Rio de Janeiro: Ltc, 2010. 636 p. Tradução e revisão técnica por Jorge Duarte Pires Valério.

CORDON, Oscar et al. A new aco model integrating evolutionary computation concepts: the best-worst ant system. **Iridia - Université Libre de Bruxelles,** Belgium, p.22-29, fev. 2000.

DARWIN, Charles. **The Origin of Species.** 6. ed. London: John Murray, 1859. 438 p.

DESMOND, Adrian; MOORE, James. **Darwin:** A vida de um evolucionista atormentado. 4. ed. São Paulo: Geração Editorial, 2001. 796 p.

DORIGO, Marco; GAMBARDELLA, Luca Maria. **Ant colonies for the travelling salesman problem. Biosystems**, [s.l.], v. 43, n. 2, p.73-81, jul. 1997.

DORIGO, M.; GAMBARDELLA, L.m.. Ant colony system: a cooperative learning approach to the traveling salesman problem. **Ieee Transactions On Evolutionary Computation**, [s.l.], v. 1, n. 1, p.53-66, abr. 1997. Institute of Electrical and Electronics Engineers (IEEE).

DORIGO, Marco. Optimization, learning and natural algorithms. **Ph. D. Thesis, Politecnico di Milano**, Italy, 1992.

DORIGO, Marco; MANIEZZO, Vittorio; COLORNI, Alberto. Positive feedback as a

search strategy. **Technical Report**, Milan, Italy, jun. 1991.

DORIGO, M.; MANIEZZO, V.; COLORNI, A.. Ant system: optimization by a colony of cooperating agents. **Ieee Transactions On Systems, Man And Cybernetics, Part B (cybernetics)**, [s.l.], v. 26, n. 1, p.29-41, 1996. Institute of Electrical and Electronics Engineers (IEEE).

DORIGO, Marco; BIRATTARI, Mauro; STUTZLE, Thomas. Ant colony optimization. **Ieee Computational Intelligence Magazine**, [s.l.], v. 1, n. 4, p.28-39, nov. 2006. Institute of Electrical and Electronics Engineers (IEEE).

DORIGO, Marco; CARO, Gianni di; GAMBARDELLA, Luca M.. Ant Algorithms for Discrete Optimization. **Artificial Life**, [s.l.], v. 5, n. 2, p.137-172, abr. 1999. MIT Press - Journals. http://dx.doi.org/10.1162/106454699568728.

DORIGO, Marco; STÜTZLE, Thomas. Ant colony optimization. Cambridge: **Mit Press**, 2004.

ENGELBRECHT, Andries P.. **Computational Intelligence**: An Introduction. 2. ed. London: West Sussex, 2007. 597 p.

ERNST, A.t et al. Staff scheduling and rostering: A review of applications, methods and models. **European Journal Of Operational Research**, [s.l.], v. 153, n. 1, p.3-27, fev. 2004. Elsevier BV.

FACELI, Katti et al. **Inteligência Artificial**: Uma Abordagem de Aprendizado de Máquina. Rio de Janeiro: Ltc, 2011. 378 p.

GAMBARDELLA, Luca M.; DORIGO, Marco. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. **Machine Learning Proceedings,**[s.l.], p.252-260, 1995. Elsevier.

GASPAR-CUNHA, António; TAKAHASHI, Ricardo; ANTUNES, Carlos Henggeler. **Manual de computação evolutiva e metaheurística**. Belo Horizonte: Editora Ufmg, 2013. 453 p.

GHELLERE, Samuel Lodetti. **Algoritmo Standard Ant Clustering Algorithm Na Tarefa De Clusterização Da Shell Orion Data Mining Engine.** 2012. 124 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, 2012.

GIROTO, Diego Bachim. **AVALIAÇÃO DE SISTEMAS DE GESTÃO ESCOLAR SEGUNDO O MODELO DE AVALIAÇÃO DE PREPARO PARA NEGÓCIOS OPENBRR.** 2014. 60 f. Tese (Doutorado) - Curso de Ciência da Computação,

Universidade Federal de Lavras, Lavras, 2014.

GOLDBARG, Marco Cesar; GOLDBARG, Elizabeth Gouvêa; LUNA, Henrique Pacca Loureiro. **Otimização Combinatória e Meta-heurística: Algoritmos e aplicações.** Rio de Janeiro: Estúdio Castellani, 2016. 392 p.

LUGER, George F.. **Inteligência Artificial.** 6. ed. São Paulo: Person Education do Brasil Ltda., 2013. 614 p.

KOULINAS, Georgios; KOTSIKAS, Lazaros; ANAGNOSTOPOULOS, Konstantinos. A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. **Information Sciences**, [s.l.], v. 277, p.680-693, set. 2014. Elsevier BV.

KRUSE, Rudolf et al. Computational intelligence: a methodological introduction. **Springer Science & Business Media**, 2013.

LINS, Hertz Wilton de Castro. **Análise e síntese de antenas e superfícies seletivas de frequência utilizando computação evolucionária e inteligência de enxames**. 2012. 89 f. Tese (Doutorado em Automação e Sistemas; Engenharia de Computação; Telecomunicações) - Universidade Federal do Rio Grande do Norte, Natal, 2012.

LOBO, Eduardo Luiz Miranda. **Uma Solução Do Problema De Horário Escolar Via Algoritmo Genético Paralelo**. 2005. 95 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2005.

MANIEZZO, Vittorio. Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. **Informs Journal On Computing**, [s.l.], v. 11, n. 4, p.358-369, nov. 1999. Institute for Operations Research and the Management Sciences (INFORMS).

MANIEZZO, Vittorio; CARBONARO, Antonella. An ANTS heuristic for the frequency assignment problem. **Future Generation Computer Systems**, [s.l.], v. 16, n. 8, p.927-935, jun. 2000. Elsevier BV.

MATTIELO, Felipe et al. DECIFRANDO A COMPUTAÇÃO QUÂNTICA. Caderno de Física da Uefs, Santana, v. 10, n. 2, p.31-44, dez. 2012.

MELO, Rodrigo Meurer. **Definição e Implementação De Uma Função De Avaliação Para Um Sistema De Geração De Grade Horária Que Utiliza Como Método De Busca Algoritmo Genético.** 2003. 69 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, 2003.

MILLONAS, Mark M.. ·Swarms, Phase Transitions, and Collective Intelligence. **Sfi Working Paper.** [s. L.], p. 1-10. jun. 1993.

MIKUSKA, Márcia Inês Schabarum. **Uma Proposta Baseada Em Algoritmo Genético Para O Problema Timetable Escolar Compacto.** 2015. 105 f. Dissertação (Mestrado) - Curso de Métodos Numéricos em Engenharia, Setores de Tecnologia e de Ciências Exatas, Universidade Federal do ParanÁ, Curitiba, 2015. Disponível em: <http://acervodigital.ufpr.br/handle/1884/41889>. Acesso em: 25 ago. 2016.

MOLINARI, Willian. **Desconstruindo a Web:** As tecnologias por trás de uma requisição. São Paulo: Casa do Código, 2017. 155 p.

MULATI, Mauro Henrique; CONSTANTINO, Ademir Aparecido; SILVA, Anderson Faustino da. Otimização por Colônia de Formigas. **Meta-heurísticas em Pesquisa Operacional**, [s.l.], p.53-68, 9 maio 2013. Omnipax.

NEUKIRCHEN, Fábio Viriato Perez. **Um estudo de caso sobre a geração de quadros de horários nos cursos de Ciência da Computação e Engenharia da Computação da UFRGS**. 2015. 56 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

OLIVEIRA, Ioná Maghali Santos de. **Inteligência De Enxames Aplicada Ao Problema De Otimização De Recargas De Reatores Nucleares A Água Pressurizada.** 2013. 100 f. Tese (Doutorado) - Curso de Engenharia Nuclear, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013. Disponível em: <https://www.nuclear.ufrj.br/DScTeses/teses2013/Tese_Iona_Maghali.pdf>. Acesso em: 30 ago. 2016.

OLIVEIRA, Janio; VIANNA, Dalessandro; VIANNA, Marcilene. Uma Heurística GRASP+VND para o Problema de Programação de Horário Escolar. **Sistemas & Gestão**, [s.l.], v. 7, n. 3, p.326-335, 2012. LATEC.

PATRICK, Kenekayoro; GODSWILL, Zipamone. Greedy Ants Colony Optimization Strategy for Solving the Curriculum Based University Course Timetabling Problem. **British Journal of Mathematics & Computer Science**, [s.l.], v. 14, n. 2, p.1-10, 10 jan. 2016. Sciencedomain International.

PEIXOTO, M.s.; BARROS, L.c.. Um Estudo de Autômatos Celulares para o Espalhamento Geográfico de Epidemias com Parâmetro Fuzzy. **Tema - Tendências em Matemática Aplicada e Computacional**, [s.l.], v. 5, n. 1, p.125-134, 13 abr. 2004. Brazilian Society for Computational and Applied Mathematics (SBMAC).

PESSOA, Carolina de Marco. **Aperfeiçoamento do algoritmo colônia de**

formigas para o desenvolvimento de modelos quimiométricos. 2015. 131 f. Dissertação (Mestrado em Engenharia) – Universidade Federal do Rio Grande do Sul, Porto Alegre.

PILA, Adriano Donizete. História e terminologia a respeito da computação evolutiva. **Revista de Ciências Exatas e Tecnologia**, [s. L.], v. 1, n. 1, p. 40-50, 2015.

RANZAN, Cassiano. **Desenvolvimento de modelos quimiométricos utilizando o algoritmo de otimização colônia de formigas.** 2014. 150 f. Tese (Doutorado) - Curso de Programa de Pós-graduação em Engenharia Química., Universidade Federal do Rio Grande do Sul., Porto Alegre, 2014. Disponível em: <http://www.lume.ufrgs.br/handle/10183/101212>. Acesso em: 25 ago. 2016.

ROSA, Tágner Formanski. A**nálise Comparativa De Um Algoritmo De Busca E Satisfação De Restrições E Algoritmo Genético Em Um Sistema Para Geração De Horário Escolar.** 2014. 106 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, 2015.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial**. Rio de Janeiro: Elsevier Editora Ltda., 2013.

SANTOS, Izabel Cristina Vilela. **Utilização da Metaheurística Simulated Annealing para a otimização da programação de turnos dos funcionários de uma loja varejista.** 2016. 32f. Monografia (Graduação em Engenharia de Produção) – Instituto de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, 2016.

SCALDELAI, Dirceu; VITOR, Adriano. Gerador de Grade de Horário Escolar por meio de Múltiplas Metaheurísticas GRASP. **Proceeding Series Of The Brazilian Society Of Applied And Computational Mathematics**, [s.l.], p.1-6, 19 dez. 2014. SBMAC. http://dx.doi.org/10.5540/03.2014.002.01.0112.

SCHAERF, Andrea. A Survey of Automated Timetabling. **Artificial Intelligence Review,** [s.l.], v. 13, n. 2, p.87-127, 1999. Springer Nature.

SERAPIÃO, Adriane Beatriz de Souza. Fundamentos de otimização por inteligência de enxames: uma visão geral. **Sba Controle & Automação**, [s.l.], v. 20, n. 3, p.271-272, set. 2009. FapUNIFESP (SciELO).

SIMANJUNTAK, Tennov et al. **UNIVERSITY COURSE TIMETABLING USING ANT COLONY OPTIMIZATION**. Laguboti: Politeknik Informatika del, 2012.

SOCHA, Krzysztof; KNOWLES, Joshua; SAMPELS, Michael. A MAX-MIN Ant System for the University Course Timetabling Problem. **Ant Algorithms**, [s.l.], p.1-13, 2002. Springer Berlin Heidelberg.

SOUZA, M.j.f.; MACULAN, N.; OCHI, L.s.. Uma Heurística para o Problema de Programação de Horários em Escolas. **Tendências em Matemática Aplicada e Computacional,** [s.l.], v. 2, n. 1, p.213-222, 14 out. 2001. Brazilian Society for Computational and Applied Mathematics (SBMAC).

SOUZA, Vitor; LOPES, Ramon; JANUARIO, Tiago. Proposta de um Algoritmo Híbrido baseado em Colônia de Formigas para o Problema de Roteamento de Veículos com Restrições de Cobertura. **Abakós**, [s.l.], v. 5, n. 1, p.3-17, 29 nov. 2016. Pontificia Universidade Catolica de Minas Gerais.

SORIA-ALCARAZ, Jorge A. et al. Effective learning hyper-heuristics for the course timetabling problem. **European Journal Of Operational Research**, [s.l.], v. 238, n. 1, p.77-86, out. 2014. Elsevier BV.

STÜTZLE, Thomas; HOOS, Holger H. – **Ant System. Future Generation Computer Systems**, [s.l.], v. 16, n. 8, p.889-914, jun. 2000. Elsevier BV. http://dx.doi.org/10.1016/s0167-739x(00)00043-1.

STÜTZLE, Thomas; HOOS, Holger H. Improving the Ant System: A detailed report on the MAX–MIN Ant System. **FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA–96–12**, 1996.

TILAHUN, Surafel Luleseged; ONG, Hong Choon. Prey-Predator Algorithm: A New Metaheuristic Algorithm for Optimization Problems. **International Journal of Information Technology & Decision Making,** [s.l.], v. 14, n. 06, p.1331-1352, nov. 2015. World Scientific Pub Co Pte Lt.

THEPPHAKORN, Thatchai; PONGCHAROEN, Pupong; HICKS, Chris. An ant colony based timetabling tool. **International Journal Of Production Economics,** [s.l.], v. 149, p.131-144, mar. 2014. Elsevier BV.

UGAT, Earth B. et al. Parallel Ant Colony Optimization on the University Course-Faculty Timetabling Problem in MSU-IIT Distributed Application in Erlang/OTP. **GSTF Journal on Computing (JoC)**, [s.l.], v. 1, n. 4, jan. 2018.

VERÇOSA, Luis Felipe; LOIOLA, Eliane. Uma investigação do problema de elaboração de grade horária (timetabling problem). **Revista de Engenharia e Pesquisa Aplicada,** [s. L.], v. 2, n. 1, p.188-197, 2016.

APÊNDICE(S)

APÊNDICE A - Artigo

A Meta-Heurística Por Colônia De Formigas Pelo Algoritmo Min-Max Ant System Aplicada Ao Problema De Quadro De Horários Escolar

Caroline Salib Canto, Merisandra Côrtes de Mattos Garcia

Universidade do Extremo Sul Catarinense (UNESC)
Criciúma – SC – Brazil

carolinesalibc@gmail.com; mem@unesc.net

Abstract. *The generation of school timetable is a classic problem of combinatorial optimization that constitutes a critical factor of quality for any educational institution. The i-Educar is a community-based software for school management, used by several cities throughout Brazil to assist in the management of public schools. Considering the complexity of manual elaboration and the difficulty of obtaining good solutions, the present work proposes the use of the optimization metaheuristic based on ant colony optimization to generate timetables with data from the i-Educar. For this, an API was implemented that allows reading the i-Educar data and importing these into the prototype database. Among the ant colony methods, the Min-Max Ant System algorithm was used to generate the hourly grid. The results were positive, being able to generate hourly quality grid with satisfactory time, affirming, then, that MMAS method with local search is a good candidate for solving problems of combinatorial optimization, being able to generate good results and with few violations of the restrictions.*

Resumo. *A geração de quadro de horários nas escolas é um problema clássico de otimização combinatória que se constitui em um fator crítico de qualidade para qualquer instituição de ensino. O software de gestão escolar i-Educar é um projeto feito em comunidade, utilizado por diversos municípios em todo o Brasil para auxílio na gestão de escolas públicas. Considerando a complexidade na elaboração de grade horária de forma manual e a dificuldade de obtenção de soluções ótimas tem tempo computacional aceitável, o presente trabalho propõe o uso da meta-heurística de otimização por colônia de formigas para gerar quadros de horários com dados do software de gestão escolar i-Educar. Para isto, foi implementada uma API que permite ler os dados do i-Educar, e importar estes para a base de dados do protótipo. Dentre os métodos de colônia de formigas, empregou-se o algoritmo Min-Max Ant System para geração da grade horária. Os resultados foram positivos, podendo ser gerado grade horária de qualidade com tempo satisfatório, afirmando então, que método MMAS com busca local é um bom candidato para resolução de problemas de otimização combinatória, podendo gerar bons resultados e com poucas violações das restrições difíceis.*

1. Introdução

A meta-heurística, conforme colocado por Santos (2016), é um método utilizado para a resolução de problemas de otimização que geralmente não podem ser resolvidos por heurísticas tradicionais. A meta-heurística tem como característica diversificar o campo de so-

luções, onde ao encontrar um ótimo local é realizado um processo de afastamento para buscar uma nova solução em outro local, como uma combinação de escolhas aleatórias e resultados encontrados anteriormente (BARBOSA, 2014). Uma otimização, do ponto de vista computacional, é um conjunto de praticas que auxiliam na busca de melhores caminhos para montar soluções e resolver problemas de interesse para os seres humanos (OLIVEIRA, 2013). Segundo Lobo (2005), denomina-se um problema de otimização, quando dentre todas as soluções possíveis, é considerada a de menor ou maior valor da função objetivo, variando-se no caso de problemas de minimização ou maximização. De acordo com Serapião (2009), nos últimos tempos, algoritmos inspirados em inteligência coletiva, vem sendo utilizados para resolver problemas de busca e otimização nas quais as abordagens tradicionais não conseguem encontrar boas soluções.

A inteligência coletiva, também conhecida como inteligência de enxames ou inteligência de colônias, de acordo com Bembem e Santos (2013), visa o reconhecimento de habilidades distribuídas a um grupo de indivíduos. Algoritmos baseados em inteligência coletiva e meta-heurísticas vêm sendo utilizados para resolver diversos problemas de busca e otimização, onde as abordagens tradicionais, como programação matemática, têm dificuldades em solucionar. Estes algoritmos tendem a serem conduzidos de uma forma evolutiva, inicia-se na obtenção da população inicial, e então se aplica métodos de busca local para melhorar a solução, considerando que os indivíduos são evoluídos e que haja troca de informações entre eles. Este processo conduzirá os indivíduos em direção a uma solução ótima (SERAPIÃO, 2009).

Um dos problemas de otimização combinatória que pode ser estudado em conjunto com meta-heurísticas e inteligência coletiva, é a elaboração de quadro de horários, problema conhecido pelo nome de *Timetabling Problem*. No Brasil, a elaboração de quadro de horários nas escolas se da a todo início de ano letivo, dando trabalho aos gestores escolares na hora de executar esta tarefa de forma que haja qualidade em seus resultados (ALVES, 2010). Muitas vezes, para realização desta tarefa, gestores tem o auxilio de algum software de gestão escolar, como por exemplo o i-Educar.

O i-Educar é um software público de gestão escolar que pode ser utilizado, modificado e distribuído livremente por qualquer pessoa ou organização. Por ser um software livre e consequentemente gratuito, além dos benefícios já disponíveis dentro da plataforma, o uso do software gera economia, como no caso citado por Brasil (2015) onde a prefeitura do município de Monte Alegre do Rio Grande do Norte economizou cerca de R$ 2,4 milhões no período de um ano com o uso desta ferramenta. Uma característica interessante, e relacionada diretamente a pesquisa realizada neste trabalho, é que o software possui um módulo para geração de quadro de horários, no entanto isto é realizado manualmente e não de uma forma automatizada.

A elaboração de grade horária feita de forma manual gera desperdício do tempo dos gestores e nem sempre dispõe de resultados satisfatórios, o que pode ser prejudicial para o rendimento dos alunos, pois pode ocorrer sobrecarga de aulas acumuladas, ou pode ocorrer a necessidade de deslocamento para salas muito distantes (ALVES, 2010; NEUKIRCHEN, 2015).

O algoritmo de otimização por colônia de formigas, do inglês Ant Colony Optimization (ACO), é uma meta-heurística utilizada em problemas de otimização combinatória. Esta técnica foi introduzida por Dorigo et al. (1992) como Ant System (AS), porém ainda não fazia o uso de busca local. Mais tarde, Dorigo e Gambardela (1997) desenvolveram a primeira versão do ACO, com o propósito de buscar soluções para o problema do caixeiro viajante (PES-

SOA, 2015). O algoritmo ACO e a técnica AS foram baseados no comportamento real das formigas quando estão em busca coletiva de fontes de alimento. Segundo Pessoa (2015) as formigas que viajam pelo caminho mais curto retornam à colônia mais rapidamente, desta forma o trajeto percorrido contém uma maior concentração de feromônio. Essa trilha atrai as outras formigas e com o tempo todos os indivíduos da colônia passam a trilhar pelo mesmo caminho, tornando-o otimizado e mais curto (ALLEGRINI; OLIVIERI, 2011). O algoritmo desenvolvido por Dorigo e Gambardela é justificado pela distribuição de x formigas em y locais, de forma que cada formiga percorra um caminho que passe apenas uma única vez pelo local (PESSOA, 2015).

2 O Algoritmo Min-Max Ant System aplicado na geração de quadro de horários em software de gestão escolar

Dentre os algoritmos baseados em colônia de formigas, nesta pesquisa escolheu-se implementar o Min-Max Ant System (MMAS). Este algoritmo tem tido bons resultados em problemas de alocação de horários em cursos universitários, para problemáticas não muito extensas, conforme os estudos realizados nas *pesquisas de* Hicks, Pongcharoen e Theppha-korn (2014); Knowles, Sampels e Socha (2002), e Ugat et al. (2018).

Nesta pesquisa, a implementação do algoritmo MMAS aplica os conceitos de otimização por colônia de formigas no protótipo chamado de *Ants for Education*, fazendo com que seja gerada uma opção de grade horária para escolas, e que ao final cada turma tenha todos os períodos de aulas preenchidos e com os professores disponíveis alocados.

Os dados utilizados para a elaboração da grade horária foram gerados simulando os cadastros realizados no i-Educar pelas escolas que o utilizam, sendo estes dados fictícios. Como várias escolas utilizam este software de gestão, no futuro elas mesmas podem gerar automaticamente o quadro de horários por meio do protótipo.

Na figura 1 é possível ter uma visão geral desta pesquisa, que consiste em consultar uma Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API), no i-Educar, importar os dados para o banco de dados do protótipo, executar o algoritmo MMAS que vai processar os dados da escola e no final apresentar um quadro de horários.

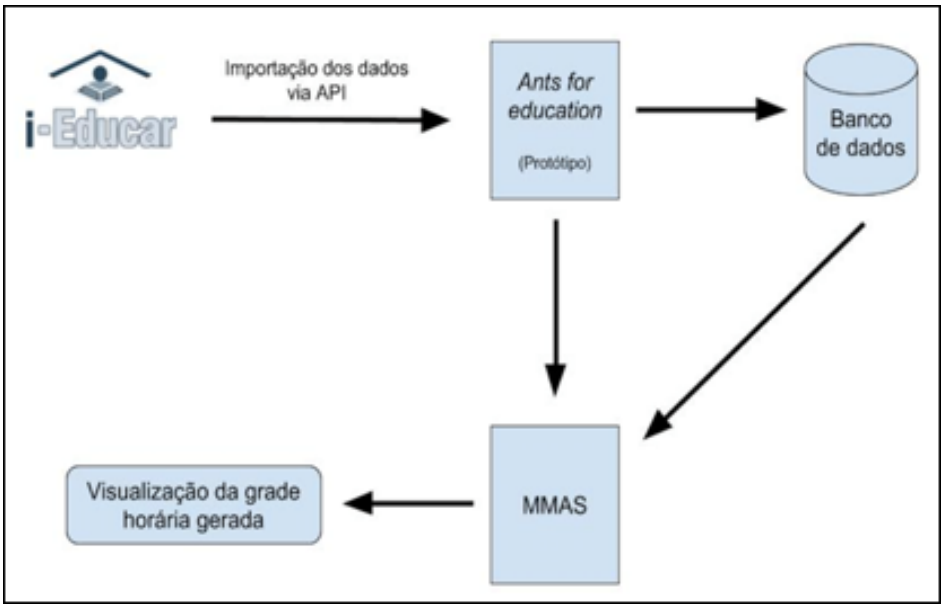


Figura 1. Visão geral da pesquisa

2.1 Importação dos dados do i-Educar

Para coletar os dados do i-Educar, implementou-se uma API no código-fonte disponível do software[1], expondo algumas informações como escolas, cursos, séries, turmas, professores e disciplinas, direto da base de dados do programa. Antes de realizar a importação, configuraram-se os dados necessários para que o protótipo possa acessar a API do i-Educar. Para isto, foi informa-se a URL do i-Educar, que deve estar rodando em algum servidor web, e também se fornece o código de acesso (*token*) para que seja possível acessar os dados expostos na API.

2.2 Desenvolvimento do protótipo

Para a geração da grade horária, foi implementado uma tela (figura 2) para selecionar quais as turmas devem ser consideradas para a execução do algoritmo, sendo possível filtrá-las por escola, curso e série. Ao clicar no botão *Gerar Quadro de Horários*, uma instância do algoritmo MMAS será iniciada, passando como argumento o conjunto de turmas selecionadas e a quantidade de aulas que cada turma tem por dia.

Figura 2. Tela para gerar quadro de horários

Ao clicar no botão para gerar o quadro de horários, uma instância da classe MMAS que foi implementada é gerada. Na figura 3 é possível visualizar a estrutura da classe MMAS, que contém alguns valores fixos como o número de formigas que fora adicionado como 10, número de tentativas, também adicionado como 10, e o tempo limite no qual o algoritmo irá ficar em modo de execução, sendo um total de 90 segundos. O número de formigas, tentativas e tempo limite foram escolhidos com base na literatura, com referência aos trabalhos de Knowles, Sampels e Socha (2002) e Simanjuntak et al. (2012).

[1] https://github.com/carolinesalib/ieducar_tcc/tree/api-quadro-horario

```
class MMAS
  NUMBER_OF_ANTS = 5
  NUMBER_OF_TRIES = 5
  TIME_LIMIT_SECONDS = 90

  def initialize(classrooms, days, periods)
    @problem = Problem.new(classrooms, days, periods)
  end

  def generate
    time_start = Time.now
    generate_ants
    best_solution = Solution.new(@problem)

    NUMBER_OF_TRIES.times do
      @problem.reset_pheromone

      raise TimeLimitError if time_passed?(time_start)

      best_fitness = 99999
      ant_index = nil
      @ants.each_with_index do |ant, index|
        @problem = ant.move!(@problem)

        @problem.evaporate_pheromone

        fitness = ant.solution.calcule_hard_constraints_violations
        if fitness < best_fitness
          best_fitness = fitness
          ant_index = index
        end
      end

      @problem = @ants[ant_index].solution.local_search(@problem)

      feasible = @ants[ant_index].solution.compute_feasibility

      unless feasible
        @ants[ant_index].solution.calcule_hard_constraints_violations
        if @ants[ant_index].solution.hard_constraints_violations <= best_solution.hard_constraints_violations
          best_solution = @ants[ant_index].solution
          best_solution.soft_constraints_violations = 99999
        end
      end

      @ants[ant_index].solution = best_solution

      @problem.pheromone_min_max
      @problem = @ants[ant_index].deposite_pheromone
    end

    @problem.hcv = best_solution.calcule_hard_constraints_violations

    @problem
  end

  def time_passed?(time_start)
    return true if Time.now >= time_start + TIME_LIMIT_SECONDS

    false
  end

  def generate_ants
    @ants = Array.new(NUMBER_OF_ANTS, Ant.new)
  end
end
```

Figura 3. Visão geral do algoritmo MMAS

2.3 Realização de testes

Para a realização dos testes, verificou-se a importância do desempenho do algoritmo em relação ao seu tempo de execução, bem como a qualidade do resultado, que leva em consideração principalmente se existe choque de horários entre os professores e *timeslots* vazios. Os experimentos foram realizados com quantidades de 3 e 5 turmas, contendo de 5 a 10 disciplinas diferentes com um professor atrelado a cada disciplina. Os resultados podem ser vistos de forma detalhada na sessão de apresentação e discussão dos resultados.

2.4 Apresentação e discussão dos resultados

Com objetivo de compreender a eficiência do algoritmo na elaboração de quadro de horários foram realizados três experimentos com quantidades diferentes de turmas e disciplinas, variando de 3 a 5 turmas com 5 a 10 disciplinas. A relação dos experimentos realizados pode ser vista na tabela 1.

Tabela 1 - Experimentos realizados

Número do experimento	Número de turmas	Número de disciplinas
1	3	5
2	3	10
3	5	10

Os parâmetros utilizados para a geração do quadro de horários podem ser vistos na tabela 2, considerando os critérios expostos anteriormente dentro da metodologia na seção de desenvolvimento do protótipo.

Tabela 2 - Lista de parâmetros

Parâmetro	Valor
Número de formigas	10
Número de tentativas	10
Tempo limite de execução	90 s
Evaporação do feromônio	1.0
Feromônio mínimo	0.1
Feromônio máximo	8.0

Na tabela 3 tem-se uma comparação dos resultados obtidos para cada experimento. Em relação aos dados da tabela, compreende-se que dentre os testes realizados, o algoritmo teve resultados mais satisfatórios quando existiam mais turmas e disciplinas em sua composição. Entende-se também, que o uso da busca local se torna um elemento indispensável em relação a qualidade da solução.

Tabela 3 - Resultados obtidos

Número do experimento	Número de turmas	Número de disciplinas	Tempo de execução	*Timeslots vazios*
1	3	5	5,89	13
1 (busca local)	3	5	4,91	27
2	3	10	8,27	1
2 (busca local)	3	10	5,82	19
3	5	10	8,11	0

2.5 Análise de desempenho e qualidade

Para a análise de desempenho e qualidade foi feito um comparativo considerando a quantidade de turmas, disciplinas, formigas, tentativas, feromônio mínimo, feromônio máximo e a evaporação do feromônio. Com a variação destes valores, foi analisado qual o tempo de execução e o número de restrições violadas, do inglês *Hard Constrains Violations* (HCV).

Este tipo de comparação permite compreender a influência dos parâmetros na obtenção de bons resultados, e mediante a estes dados, pode ser tomado direcionamentos a fim de obter um melhor desempenho ou uma melhor qualidade do resultado final.

Na tabela 4 é apresentado uma comparação dos testes realizados com diferentes parâmetros e quantidades de informações. As duas colunas ao final da tabela representam o tempo que levou para gerar a solução e a quantidade de restrições que o algoritmo não conseguiu resolver no final de sua execução.

Tabela 4 - Análise de desempenho e qualidade

Quantidade turmas	Quantidade disciplinas	Quantidade formigas	Quantidade tentativas	Feromônio mínimo	Feromônio máximo	Evaporação	Tempo execução	HCV
5	10	10	10	0,1	8	1	24,61	6
5	10	10	10	0,0078	3,3	0,3	25,36	6
5	10	5	5	0,1	8	1	9,25	10
5	10	5	5	0,0078	3,3	0,3	9,44	10
3	10	10	10	0,1	8	1	7,52	2
3	10	10	10	0,0078	3,3	0,3	8,68	2
3	10	5	5	0,1	8	1	2,35	2
3	10	5	5	0,0078	3,3	0,3	3,09	2
3	5	10	10	0,1	8	1	5,84	25
3	5	10	10	0,0078	3,3	0,3	5,53	25
3	5	5	5	0,1	8	1	2,2	25
3	5	5	5	0,0078	3,3	0,3	2,2	25

Após a análise de desempenho e qualidade, percebeu-se que a variação do parâmetro de quantidade de formigas e tentativas pode influenciar no tempo de execução do algoritmo, sendo que em alguns casos o algoritmo levou mais que o dobro do tempo de execução quando modificado o número formigas e tentativas de 5 para 10, não tendo diferenças quanto ao número de restrições, com exceção do exemplo com 5 turmas e 10 disciplinas, onde o número de restrições violadas aumentou de 6 para 10 quando diminuiu o número de formigas e tentativas de 10 para 5. Referente aos parâmetros feromônio mínimo, feromônio máximo e evaporação, compreende-se que estes não tiveram tanta influência no tempo de execução e quantidade de restrições, tendo uma diferença de menos de um segundo no tempo de execução em alguns experimentos, e nenhuma diferença quanto ao número de HCVs.

2.6 Discussão dos resultados

As técnicas de otimização por colônia de formigas têm sido cada vez mais empregadas em estudos para resolução de problemas de otimização combinatória, como no caso da geração de grade horária, onde destaca-se os métodos com MMAS e Busca Local em problemas de *University Course Timetabling Problem* (UCTP).

No trabalho de Knowles, Sampels e Socha (2002), foi utilizado o algoritmo MMAS com uma Busca Local básica, confirmando ser um tratamento bastante simples para o problema de horários. Na implementação dos autores, eles utilizaram os parâmetros de feromônio mínimo 0.0078, feromônio máximo 3.3 e evaporação 0.3, sendo que nos experimentos reali-

zados nesta pesquisa com estes valores, como pode ser visto na tabela 10, teve-se uma perda pequena de desempenho. Confirma-se então que os valores de feromônio mínimo 0.1, feromônio máximo 8 e evaporação 1, para o modelo de quadro de horários utilizado nesta pesquisa, teve um resultado melhor do que o modelo representado na pesquisa de Knowles, Sampels e Socha (2002).

Já na pesquisa de Simanjuntak et al. (2012), foi utilizado o algoritmo MMAS com Busca Local. Com uso dos parâmetros de feromônio mínimo 0.1, feromônio máximo 3.3 e evaporação 0.3, os autores tiveram um bom desempenho na execução do algoritmo quando para resolução de problemas pequenos de quadro de horários, levando um total de 0.4 segundos nesta categoria. No presente trabalho, o tempo mínimo para execução do algoritmo foi de 2.2 segundos, o que leva a crer que o algoritmo desenvolvido por Simanjuntak et al. (2012) teve um desempenho melhor para o problema proposto em relação a velocidade de execução.

No contexto brasileiro, existem iniciativas para resolução de grade horária nas instituições de ensino, utilizando algoritmos genéticos (MELO, 2003), método *Greedy Randomized Adptative Search Procedure* (GRASP) (SCALDELAI; VITOR, 2014) e *Iterated Local Search, Discrete Particle Swarm Optimization* e *Jump Frog Optimization* (ALVES, 2010).

Ao que compete a trabalhos científicos brasileiros, não foi encontrado trabalhos que resolvam problema de grade horária por meio do uso de MMAS, porém em comparação com os trabalhos que estudam GRASP, percebe-se que os resultados obtidos por meio do uso de MMAS são também satisfatórios.

Com relação aos trabalhos que utilizam a mesma tecnologia, o MMAS com Busca Local, pode-se confirmar que o uso da técnica traz bons resultados para problemas pequenos de grade horária, com tempo satisfatório levando alguns poucos segundos para execução completa do algoritmo. O uso desta metodologia também enfatiza que o uso da busca local se faz necessário para que se encontre resultados de melhor qualidade.

3. Conclusão

A fim de se obter melhores resultados em problemas de otimização pode-se aplicar técnicas de inteligência coletiva, como por exemplo, algoritmos de colônia de formigas. Este tem um papel importante na tomada de decisão quando se trata de uma otimização combinatória, como no caso do problema de geração de quadro de horários.

Após os estudos sobre o problema de geração de grade horária, entendeu-se que por meio de abordagens da inteligência coletiva, é possível que sejam encontradas boas soluções com base na tomada de decisão ao alocar eventos em *timeslots,* respeitando ao máximo as restrições empregadas.

Com o desenvolvimento da API no i-Educar foi possível coletar dados para a implementação do algoritmo baseado em otimização por colônia de formigas. Fazendo uso do algoritmo MMAS, buscou-se um trajeto otimizado ao alocar eventos em *timeslots* que melhor se adequa a situação atual deste evento. A aplicação da busca local, do tipo subida de encosta, no algoritmo de MMAS fez com que a qualidade dos resultados melhorasse de forma significativa, fazendo da junção dos dois algoritmos, uma boa solução baseada na otimização por colônia de formigas para geração de quadro de horários.

Avaliou-se que o tempo de execução foi satisfatório em todos os testes, percebendo que em casos com mais variações de disciplinas, obteve-se resultados melhores e com poucas violações das restrições difíceis. Neste aspecto, afirma-se que o presente algoritmo é um bom

candidato para resolver problemas de otimização combinatória de médio porte, com poucas restrições difíceis.

4. Referências

ALLEGRINI, Franco; OLIVIERI, Alejandro C.. A new and efficient variable selection algorithm based on ant colony optimization. Applications to near infrared spectroscopy/partial least-squares analysis. **Analytica Chimica Acta**, [s.l.], v. 699, n. 1, p.18-25, ago. 2011. Elsevier BV/.

ALVES, Reginaldo Heidder de Jesus. **Metaheurísticas Aplicadas ao Problema de Horário Escolar.** 2010. 68 f. Dissertação (Mestrado) - Curso de Mestrado em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2010.

BARBOSA, Roberto Cavalcante. **Aplicação Da Metaheurística Algoritimo Genético Na Otimização Das Rotas De Entregas Da Distribuição Física De Produtos No Município De Fortaleza.** 2014. 91 f. Dissertação (Mestrado) - Curso de LogÍstica e Pesquisa Operacional, Universidade Federal do Ceará, Fortaleza, 2014.

BRASIL. Ministério da Educação. Sistema público permite redução de gastos em cidade potiguar. 2015. Disponível em: <http://portal.mec.gov.br/ultimas-noticias/209-564834057/30851-sistema-publico-permite-reducao-de-gastos-em-cidade-potiguar>. Acesso em: 29 ago. 2016.

DORIGO, Marco; GAMBARDELLA, Luca Maria. **Ant colonies for the travelling salesman problem. Biosystems**, [s.l.], v. 43, n. 2, p.73-81, jul. 1997.

DORIGO, Marco. Optimization, learning and natural algorithms. **Ph. D. Thesis, Politecnico di Milano**, Italy, 1992.

LOBO, Eduardo Luiz Miranda. **Uma Solução Do Problema De Horário Escolar Via Algoritmo Genético Paralelo**. 2005. 95 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2005.

MELO, Rodrigo Meurer. **Definição e Implementação De Uma Função De Avaliação Para Um Sistema De Geração De Grade Horária Que Utiliza Como Método De Busca Algoritmo Genético.** 2003. 69 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, 2003.

NEUKIRCHEN, Fábio Viriato Perez. **Um estudo de caso sobre a geração de quadros de horários nos cursos de Ciência da Computação e Engenharia da Computação da UFRGS**. 2015. 56 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

OLIVEIRA, Ioná Maghali Santos de. **Inteligência De Enxames Aplicada Ao Problema De Otimização De Recargas De Reatores Nucleares A Água Pressurizada.** 2013. 100 f. Tese (Doutorado) - Curso de Engenharia Nuclear, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

PESSOA, Carolina de Marco. **Aperfeiçoamento do algoritmo colônia de formigas para o desenvolvimento de modelos quimiométricos**. 2015. 131 f. Dissertação (Mestrado em Engenharia) – Universidade Federal do Rio Grande do Sul, Porto Alegre.

SANTOS, Izabel Cristina Vilela. **Utilização da Metaheurística Simulated Annealing para a otimização da programação de turnos dos funcionários de uma loja varejista.** 2016.

32f. Monografia (Graduação em Engenharia de Produção) – Instituto de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, 2016.

SCALDELAI, Dirceu; VITOR, Adriano. Gerador de Grade de Horário Escolar por meio de Múltiplas Metaheurísticas GRASP. **Proceeding Series Of The Brazilian Society Of Applied And Computational Mathematics**, [s.l.], p.1-6, 19 dez. 2014. SBMAC. http://dx.doi.org/10.5540/03.2014.002.01.0112.

SERAPIÃO, Adriane Beatriz de Souza. Fundamentos de otimização por inteligência de enxames: uma visão geral. **Sba Controle & Automação**, [s.l.], v. 20, n. 3, p.271-272, set. 2009. FapUNIFESP (SciELO).

SIMANJUNTAK, Tennov et al. **UNIVERSITY COURSE TIMETABLING USING ANT COLONY OPTIMIZATION**. Laguboti: Politeknik Informatika del, 2012.

SOCHA, Krzysztof; KNOWLES, Joshua; SAMPELS, Michael. A MAX-MIN Ant System for the University Course Timetabling Problem. **Ant Algorithms**, [s.l.], p.1-13, 2002. Springer Berlin Heidelberg.

THEPPHAKORN, Thatchai; PONGCHAROEN, Pupong; HICKS, Chris. An ant colony based timetabling tool. **International Journal Of Production Economics,** [s.l.], v. 149, p.131-144, mar. 2014. Elsevier BV.