Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnológicas
Programa de Pós-Graduação em Ciência da Computação

ENRICO SILVA MIRANDA

META-APRENDIZAGEM APLICADA A PROBLEMAS DE MÁXIMA SATISFABILIDADE

São Luís
2017

ENRICO SILVA MIRANDA

META-APRENDIZAGEM APLICADA A PROBLEMAS DE MÁXIMA SATISFABILIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, **como parte dos requisitos necessários** para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Profº Alexandre César Muniz de Oliveira

São Luís

2017

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Miranda, Enrico Silva.
Meta-aprendizagem aplicada a problemas de máxima satisfabilidade / Enrico Silva Miranda. - 2017.
66 p.

Orientador(a): Alexandre César Muniz de Oliveira.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2017.

1. Máxima satisfabilidade. 2. Meta-aprendizagem. 3. Meta-heurísticas. I. Oliveira, Alexandre César Muniz de. II. Título.

A Deus e à minha família.

Agradecimentos

Agradeço a Deus, que iluminou o meu caminho durante esta longa caminhada.

Agradeço também à minha esposa Raquel, que de forma especial e carinhosa me deu força e coragem, me apoiando nos momentos de dificuldades. Quero agradecer à minha filha Nicole, que embora não tivesse conhecimento disto, iluminou de maneira especial os meus pensamentos me levando a buscar mais conhecimentos. E não deixando de agradecer aos meus pais, a quem agradeço todas as noites pela minha formação.

Ao Professor Alexandre, companheiro de caminhada ao longo desta jornada. Posso dizer que a minha formação, inclusive pessoal, não teria sido a mesma sem a sua pessoa.

Por fim, agradeço aos amigos e colegas que me apoiaram e acompanharam na jornada, em especial ao Eggo, Heygon, Theo, Elmer, Mucci e Iramar.

"Poucas ideias funcionam na primeira tentativa.
Iteração é a chave da inovação."
(Sebastian Thrun)

Resumo

Meta-aprendizado tem sido aplicado com sucesso em problemas de otimização, como o problema do Caixeiro Viajante (PCV) e Máxima Satisfabilidade (MaxSAT). Este último é um problema NP-Difícil, relevante para o estudo de problemas acadêmicos e industriais. No entanto, a maior parte da pesquisa atual no problema MaxSAT foca em métodos de solução exata. Devido à necessidade de soluções boas em um período de tempo reduzido, a utilização de meta-heurísticas é considerada neste trabalho. Além disso, propõe-se um *framework* de meta-aprendizagem para seleção de meta-heurísticas para o problema MaxSAT, o que inclui nova representação abstrata baseada em grafos, derivação de um novo conjunto de meta-características e definição de mecanismos de aprendizagem baseados em experiência obtida *a priori*. Experimentos comprovam que o arcabouço é eficaz para seleção de meta-heurística e de seus parâmetros para instâncias MaxSAT. As novas meta-características derivadas da representação baseada em grafo podem ser consideradas tão boas quanto o estado da arte atual. As medidas propostas de características de grafos podem ser aplicadas em trabalhos futuros a outras classes de problemas.

Palavras-chaves: Meta aprendizagem. Meta heurístcas. Máxima Satisfabilidade

Abstract

Meta-learning has been used with success in optimization problems, like the Traveling Salesman Problem (TSP) and the Maximum Satisfability Problem (MaxSAT). The latter is considered NP-Hard while also being relevant for academic and industrial problems. However, most of the research on the MaxSAT problem focuses of exact solution methods. Due to the need of generating good solutions on a limited time frame, this work considers the use of meta-heuristics. A meta-learning framework for meta-heuristic selection is also proposed for the MaxSAT problem, including a new representation based on graphs, new meta-features derived from this representation, the definition of machine learning mechanisms based on previous experience. Experiments show that the proposed outline is effective for selection of meta-heuristics and parameters for MaxSAT. The new meta-features derived are shown to be as good as the current state of the art. The graph meta-features proposed can be applied to other problems on the near future.

Keywords: Meta-learning. Meta-heuristics. Maximum satisfiability.

Lista de ilustrações

Lista de tabelas

Lista de Siglas

AE Algoritmo Evolutivo

AG Algoritmo Genético

ASP *Algorithm Selection Problem* - Problema da Seleção de Algoritmos

CG *Clause Graph* - Grafo de Cláusulass

EDA *Estimation of Distribution Algorithm* - Algoritmo de Estimativa de Distribuição

KNN *K Nearest Neighbors* - K vizinhos mais próximos

MaxSAT *Maximum Satisfability Problem* - Problema de Máxima Satisfabilidade

MLP *Multilayer Perceptron* - Perceptrons Multi Camadas

NCF *Normal Conjunctive Form* - Forma normal conjuntiva

PBIL *Population Based Incremental Learning* - Aprendizagem Incremental baseada em População

PSO *Particle Swarm Optimization* - Otimização por Enxame de Partículas

SVM *Support Vector Machines* - Máquinas de Vetor Suporte

VCG *Variable-Clause Graph* - Grafo de Cláusulas-Variáveis

VG *Variable Graph* - Grafo de Variáveis

WCG *Weighted Clause Graph* - Grafo de Cláusulas com pesos

WEKA *Waikato Environment for Knowledge Analysis*

WVG *Weighted Variable Graph* - Grafo de variáveis com pesos

Sumário

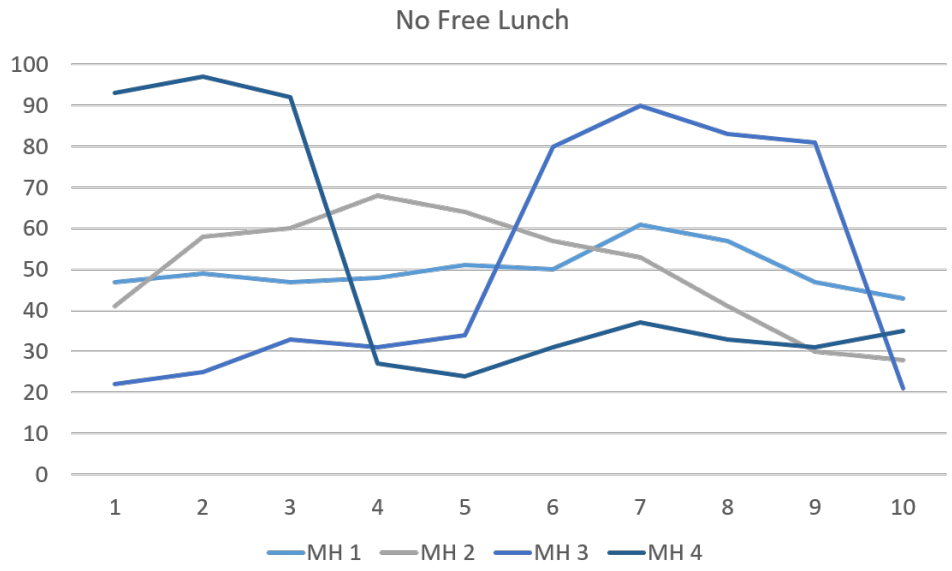
1 Introdução

O problema de Máxima Satisfabilidade, ou MaxSAT, é um tema de grande interesse da comunidade científica.Apesar de ser classificado como NP-Difícil, é fundamental para diversas aplicações práticas em diferentes áreas, incluindo Ciência da Computação e Engenharia Elétrica (ZHANG; SHEN; MANYà, 2003).

Em casos em que sejam necessárias soluções de qualidade para instâncias maiores ou casos com uma restrição temporal para a resposta, o uso de meta-heurísticas é recomendado como uma alternativa viável para a geração de soluções (TALBI, 2009).

Apesar de haver uma grande gama de meta-heurísticas e parametrização das mesmas, é comprovado pelo teorema *"No Free Lunch"* (WOLPERT; MACREADY, 1997) que não existe um algoritmo único que consiga apresentar a melhor solução para todas as instâncias possíveis de problemas. De acordo com este teorema, o ganho de performance em alguns problemas, que são obrigatoriamente um subconjunto de todos os problemas possíveis, gera obrigatoriamente uma redução de desempenho na resolução de outros problemas que não pertençam a esse subconjunto. Um exemplo de aplicação deste teorema está na Figura 1, em que algumas técnicas funcionam melhor em algumas instâncias, porém a performance média de todas é a mesma.

Figura 1 – *No Free Lunch*

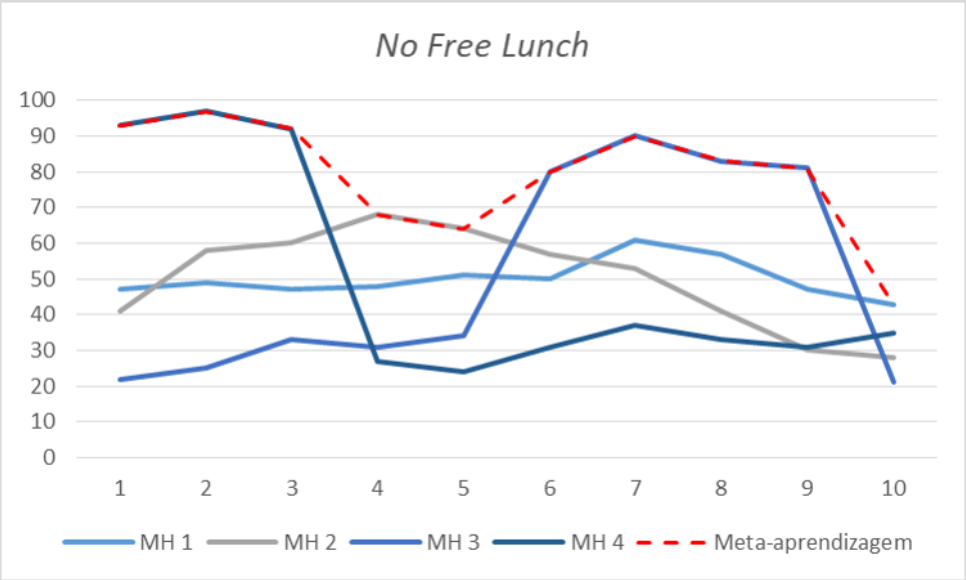


Fonte: Acervo do autor

Portanto, seria de grande valor acadêmico algoritmos capazes de caracterizar instâncias de problemas e determinar o melhor algoritmo para resolução de cada instância.

Com a utilização correta de cada algoritmo em cada problema, é possível selecionar algoritmos específicos para o problema. Desta forma, elimina-se a restrição de um único algoritmo aplicado a todas as instâncias, e consequentemente a restrição imposta pelo teorema *"No Free Lunch"*. Desta forma, caso seja possível estabelecer um oráculo que seja capaz de acertar o melhor algoritmo para todos os problemas, teríamos uma performance conforme apresentada na Figura 2. A performance da curva tracejada, através da seleção do algoritmo correto para cada problema, é capaz de obter um resultado superior a todas as técnicas individuais.

Figura 2 – *No Free Lunch* com seleção de algoritmo individual



Fonte: Acervo do autor

A seleção do melhor algoritmo pode ser modelado como um problema de aprendizagem de máquina (RICE, 1976). Mais especificamente, se for possível extrair características para cada instância de forma que problemas mais próximos sejam melhor resolvidos pelo mesmo algoritmo, a seleção se torna um problema de reconhecimento de padrões e classificação.

É importante notar que este aprendizado ou classificação não ocorre no nível do problema, ao contrário, deve trabalhar em nível superior ao de resolução do problema (meta-nível) (VILALTA; DRISSI, 2002). Enquanto no primeiro nível há o problema a ser resolvido e as meta-heurísticas que operam diretamente sobre ele, o meta-nível lida com dois problemas. O primeiro é o problema de determinar quais são as características representativas de uma instância (meta-características). O segundo problema é o de, dado um conjunto de meta-características, como selecionar o melhor método de resolução. Neste caso, as meta heurísticas podem ser vistas como classes para um problema de classificação (VILALTA; DRISSI, 2002).

Estratégias como essa têm sido usadas com sucesso em problemas NP-Difíceis, como o Caixeiro Viajante (KANDA et al., 2010) (KANDA, 2012). No entanto, os trabalhos verificados classificam apenas com relação a qual meta-heurística utilizar, sem realizar a variação de seus parâmetros.

Trabalhos anteriores também têm utilizado técnicas semelhantes para o problema MaxSAT, no entanto o foco destes tem sido em seleção de métodos de resolução exatos em menor tempo possível (ANSóTEGUI; MALITSKY; SELLMANN, 2014) (MATOS et al., 2008), ou na busca de quanto tempo um dado algoritmo resolveria uma instância específica (ZHANG; SHEN; MANYà, 2003).

1.1 SAT e MaxSAT

O problema de decisão de Satisfabilidade Booleana (SAT) possui extrema relevância para a comunidade científica. Em 1971, Cook provou a existência de uma classe de problemas de decisão denominados NP-Completos (COOK, 1971). Em seguida, provou que o SAT era NP-Completo. Por este motivo, pode-se considerar que o SAT é importante do ponto de vista histórico e fundamental em Ciência da Computação, dada a importância da definição de problemas NP-Completos.

O problema de Satisfabilidade Booleana é composto de um conjunto de cláusulas C e de um conjunto de variáveis binárias X. Cada cláusula $c_i \in C$ é uma cláusula expressa na forma normal conjuntiva (*Normal Conjunctive Form* - NCF). Esta fórmula é expressa como $c_i = (x_j \vee x_k ... \vee x_l)$, onde $x_j, x_k ... x_l \in X$. O SAT consiste em determinar se há ao menos um conjunto de valores que possa ser atribuído às variáveis de forma que todas as cláusulas sejam satisfeitas (BIERE; HEULE; MAAREN, 2009).

O MaxSAT, por sua vez, é o problema de otimização associado ao problema de decisão SAT. Enquanto o SAT busca responder se existe um conjunto de atribuições às variáveis que atende a todas as cláusulas, o MaxSAT busca responder qual a configuração de variáveis que atende o máximo de cláusulas possíveis, e qual o número máximo de cláusulas a ser atendido. A função objetivo a ser maximizada é $f_{obj} = \sum d_i$, onde $d_i = 0$ caso a cláusula c_i não seja satisfeita e $d_i = 1$ caso a cláusula c_i seja satisfeita (BIERE; HEULE; MAAREN, 2009).

1.2 Meta-heurísticas

Estratégias para calcular soluções ótimas quando baseadas em enumerações podem ser proibitivas em termos de tempo de execução, devido à natureza intratável de problemas NP-Difíceis (TALBI, 2009), vários deles de interesse industrial e científico.

Na prática, muitas vezes é possível aceitar soluções de qualidade *satisfatória*, que podem ser obtidas por meio de meta-heurísticas de otimização. Estes algoritmos representam uma família de técnicas de otimização por aproximação que têm ganhado grande popularidade nas últimas décadas (TALBI, 2009). Estas técnicas estão entre as mais promissoras e eficientes para os problemas NP-Difíceis. Elas podem gerar soluções satisfatórias dentro de uma janela de tempo aceitável para problemas complexos e este motivo explica o interesse crescente no ramo de meta-heurísticas.

No entanto, meta-heurísticas apresentam duas características bem definidas: ao contrário de algoritmos de otimização exatos, elas não garantem que a solução obtida seja a ótima em nenhum cenário. Além disso, ao contrário de algoritmos de aproximação, elas não definem também qual é a distância máxima que a solução encontrada está do ótimo global (TALBI, 2009).

1.3 Aprendizagem de Máquina e Meta Aprendizagem

Um programa de computador é dito capaz de aprender uma classe de tarefas T, se a métrica de desempenho P melhora com a experiência E (MITCHELL, 1997). Em outras palavras, o aprendizado de máquina pode ser exemplificado por um programa que consegue melhorar sua performance em uma tarefa com a aquisição de alguma experiência.

Para problemas muito complexos e com muitas variáveis, processos manuais são propensos a erros ou subavaliação de possibilidades de relações entre as características do problema. Nestes cenários, a utilização de técnicas de aprendizagem de máquina apresenta uma vantagem, pois a exploração deste espaço com muitas variáveis e possíveis inter relações são um dos pontos fortes destas técnicas (KOTSIANTIS, 2007).

Dentro da aprendizagem de máquina, destaca-se o campo de meta-aprendizagem, surgido a partir do problema de seleção de algoritmo: *Qual o algoritmo é mais provável de executar melhor para uma instância específica do problema?* (RICE, 1976).

As primeiras aplicações de meta-aprendizagem ou *meta-learning* foram em cima de problemas de classificação: dado uma instância do problema (um novo conjunto de dados), defina o melhor algoritmo para construção de classificador (THRUN; PRATT, 2012).

Meta-aprendizagem não tem sido muito explorada em otimização, porém especificamente para os problemas SAT e MaxSAT, tem havido um aumento de interesse significativo (SMITH-MILES, 2009). Para o MaxSAT há uma quantidade considerável de trabalhos que podem ser enquadrados como meta-aprendizagem (XU et al., 2007) (XU et al., 2008) (MALITSKY et al., 2013) (ANSÓTEGUI et al., 2016).

Conforme mencionado anteriormente, meta-aprendizado ocorre em nível superior

(meta-nível) ao de resolução do problema (VILALTA; DRISSI, 2002). No meta-nível, a representação de instâncias por meio de meta-características é usada para determinar o melhor algoritmo para aquela instância. Esta seleção precede o nível de resolução do problema propriamente dito.

No tocante a meta-características, ou seja, a representação de instâncias em nível de meta-aprendizado, recentes propostas têm se restringido a um subconjunto de características de SAT apresentadas em (NUDELMAN et al., 2004), contribuindo para uma certa homogenização. No que tange aos algoritmos utilizados, foram encontrados apenas seleção de algoritmos exatos (ANSÓTEGUI et al., 2016), sem qualquer registro na literatura SAT de seleção de meta-heurísticas.

1.4 Objetivos e Contribuições

O objetivo do trabalho é propor um *framework* de meta-aprendizagem que possibilite a extração de meta-características de instâncias do problema MaxSAT e consequente seleção de meta-heurísticas mais aptas a resolvê-las.

Para ser capaz de realizar essa classificação, propõe-se uma nova forma de representação do problema baseada em grafo com o intuito de melhorar a acurácia da seleção. Além disso, a utilização de meta-características baseadas em grafos sugere a possibilidade de aplicação da técnica para outros domínios.

Nesta abordagem, são utilizadas diversas técnicas de classificação e otimização para gerar um leque de opções que confirme a efetividade da técnica proposta. Para cada problema, será recomendada uma meta-heurística para resolução.

Neste trabalho, são investigadas diferentes formas de representação e diferentes meta-heurísticas com diversas operações e parâmetros. Para implementação, utiliza-se o *framework* de otimização *ParadisEO* (CAHON; MELAB; TALBI, 2004), em conjunto com a biblioteca de aprendizagem de máquina *WEKA* (HALL et al., 2009). A base de instâncias MAXSAT utilizada tem sido usada em competições internacionais (a *International MaxSAT Evaluation*). Para este trabalho, foi selecionada a base mais recente quando do início da pesquisa, sendo especificadamente a versão da competição de 2014, disponível em http://www.maxsat.udl.cat/14/benchmarks/index.html.

As contribuições deste trabalho pontualmente são:

- Proposta de uma nova forma de expressão do problema MaxSAT por meio de grafos, a partir da qual uma nova gama de meta-características para o problema foi derivada;
- Proposta de um arcabouço de meta-aprendizado aplicado a problemas MaxSAT extensível a outros problemas de otimização combinatória.

1.5 Organização do trabalho

Este documento está organizado em 6 capítulos, sendo este o primeiro, apresentando uma abordagem introdutória, a motivação e os objetivos do trabalho.

No Capitulo 2, são descritos os conceitos fundamentais relacionados ao problema MaxSAT, suas classificações na literatura. São definidos conceitos relativos a problemas de otimização e aos algoritmos e meta-heurísticas utilizadas na otimização. Além disso, serão tratadas as bases da aprendizagem de máquina, detalhando um pouco mais sistemas de classificação e algumas técnicas utilizadas.

No Capítulo 3, há a apresentação do conceito de aprendizagem de máquina e de meta-aprendizagem. Também é apresentado um breve histórico do campo, bem como aplicações para problemas de satisfabilidade e para outros problemas de otimização.

No Capítulo 4 é definida a arquitetura, bem como são descritos o *framework* e as novas formas de representação de problemas de máxima satisfabilidade para extração de novas medidas de caracterização do problema.

No Capítulo 5, são fornecidos detalhes do ambiente computacional utilizado. A representação computacional do problema e a definição dos principais operadores utilizados por cada meta-heurística são apresentados. Também são descritos os experimentos realizados e há uma discussão dos resultados obtidos e referências a trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, são descritos os conceitos fundamentais relacionados ao problema da Máxima Satisfabilidade, bem como meta-heurísticas de otimização e classificadores, com o objetivo de prover a base necessária para compreensão da proposta.

2.1 MaxSAT

O problema MaxSAT é formulado a partir de um conjunto de cláusulas C, formadas por associação de variáveis booleanas X, conectadas por operadores lógicos. Cada cláusula $c_i \in C$ é uma cláusula expressa na forma normal conjuntiva (*Normal Conjunctive Form* - NCF). Esta fórmula é expressa como $c_i = (x_j \vee x_k ... \vee x_l)$, onde $x_j, x_k ... x_l \in X$.

A função objetivo a ser maximizada é $f_{obj} = \sum d_i$, onde $d_i = 0$ caso a cláusula c_i não seja satisfeita e $d_i = 1$ caso a cláusula c_i seja satisfeita (BIERE; HEULE; MAAREN, 2009). Também é possível encontrar o problema como a minimização das cláusulas não satisfeitas, assumindo-se que $d_i = 0$ caso c_i seja satisfeita e $d_i = 1$ caso c_i não seja satisfeita. Normalmente o problema MaxSAT é tratado na literatura em sua forma de minimização das cláusulas não satisfeitas, que é adotada no presente trabalho.

2.1.1 Variações do MaxSAT

Uma variação do MaxSAT encontrada na literatura é o K-SAT, na qual todas as cláusulas possuem exatamente K variáveis. O 2-SAT, definido com exatamente duas variáveis em todas as cláusulas, é um problema de complexidade de resolução polinomial. No entanto, já foi provado que o K-SAT com $K \geq 3$ é equivalente ao SAT sem restrições, e portanto pertencente à classe de problemas NP-Completos (IMPAGLIAZZO; PATURI, 2001).

Variações usuais incluem o *Weighted MaxSAT*, em que cada cláusula tem um peso a ser levado em conta na otimização. O objetivo, neste caso, passa a ser determinar a atribuição das variáveis de forma que se maximize o peso total das cláusulas atendidas (JIANG; KAUTZ; SELMAN, 1995).

Uma outra variação encontrada na literatura é o *Partial MaxSAT*, em que existem *hard clauses*, que devem ser obrigatoriamente atendidas, e *soft clauses*, que não são obrigatórias. Neste caso, as *hard clauses* são restrições que devem ser atendidas, e o objetivo é atender o maior número possível de *hard clauses* (JIANG; KAUTZ; SELMAN, 1995).

Por fim, há também uma variação que combina o *Weighted MaxSAT* e o *Partial MaxSAT*, chamada *Weighted Partial MaxSAT*. Nesta variação, há a restrição das *hard clauses*, porém as *soft clauses* possuem um peso atribuído a elas, e o objetivo é alcançar o maior peso total possível de *soft clauses* atendidas (JIANG; KAUTZ; SELMAN, 1995).

2.1.2 Resolução Exata

Existem diversas técnicas e algoritmos para resolução do MaxSAT, a maioria baseadas em duas estratégias dominantes. A primeira estratégia é baseada em algoritmos de *branch-and-bound* e demais técnicas usuais para resolução de problemas de otimização inteira (ANSóTEGUI; MALITSKY; SELLMANN, 2014).

A segunda estratégia dominante é baseada em chamadas sucessivas a um *solver* do SAT decisório, de forma a estimar e por fim encontrar o número mínimo de cláusulas não satisfeitas (ANSóTEGUI; MALITSKY; SELLMANN, 2014).

2.1.3 Representação de instâncias MaxSAT

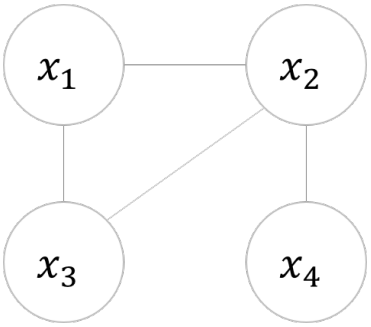
O problema MaxSAT pode ser codificado de diversos modos, tais como matrizes e vetores binários, visando atender a requisitos computacionais. No entanto, a representação por meio de grafos é de especial interesse neste trabalho. Tal representação é utilizada para resolver o problema MaxSAT como um problema de Satisfação de Restrições (NUDELMAN et al., 2004).

As formas de representação se tornam importantes no contexto da meta-aprendizagem, pois diferentes formas de representar o problema podem gerar medidas e características diferentes, e estas características são a base para o problema de aprendizagem de máquina. O conjunto de características, a ser extraído, deve permitir o agrupamento de instâncias MaxSAT que são resolvidas eficientemente por determinados algoritmos.

Existem três principais representações do problema SAT por meio de grafos.

A primeira destas representações, o grafo de variáveis (*Variable Graph* ou VG), é não-direcionado, construído a partir da instância, em que cada nó do grafo é a representação de uma variável. As arestas do grafo ocorrem entre dois nós que representam variáveis que aparecem em uma mesma cláusula. Desta forma, o conjunto de cláusulas $c_1 = x_1 \vee \neg x_2 \vee x_3$, $c_2 = \neg x_2 \vee \neg x_4$ e $c_3 = x_1 \vee \neg x_3$ gera o grafo que pode ser visto na Figura 3. Neste grafo, temos 4 nós, cada um correspondente a uma variável. Entre os nós que representam x_2 e x_4, por exemplo, há uma aresta, pois estas variáveis aparecem na mesma cláusula em c_2. Similarmente, a cláusula $c1$ gera três arestas, sendo uma entre (x_1, x_2), uma entre (x_2, x_3) e outra entre (x_1, x_3).

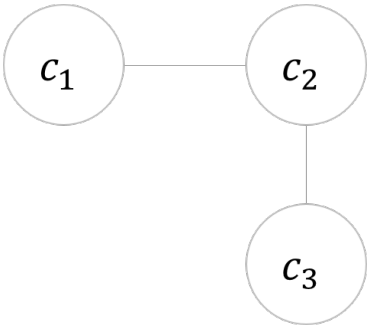
Figura 3 – Grafo de Variáveis (VG)



Fonte: Acervo do autor

Uma segunda representação, o grafo de cláusulas (*Clause Graph* ou CG), é igualmente não-direcionado, construído a partir da instância do problema, em que cada nó do grafo é a representação de uma cláusula. As arestas do grafo ocorrem entre dois nós que representam cláusulas que compartilhem uma mesma variável negada em sua formulação. Desta forma, o conjunto de cláusulas $c_1 = x_1 \vee \neg x_2 \vee x_3$, $c_2 = \neg x_2 \vee \neg x_3$ e $c_3 = x_1 \vee \neg x_3$ gera o grafo que pode ser visto na Figura 4. Neste grafo, há três nós, um para cada cláusula. Entre as cláusulas c_1 e c_2 há uma aresta, pois ambas possuem a variável negada $\neg x_2$. De forma análoga, há uma aresta entre c_2 e c_3, pelo compartilhamento da variável negada $\neg x_3$.

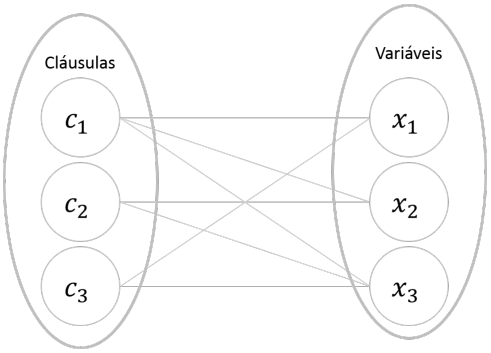
Figura 4 – Grafo de Cláusulas (CG)



Fonte: Acervo do autor

A última representação, o grafo de cláusulas-variáveis (*Variable-Clause Graph* ou VCG), é não-direcionado bipartido, construído a partir da instância do problema em que, no primeiro subconjunto, cada nó do grafo é uma variável, e no segundo subconjunto cada nó é uma cláusula. As arestas do grafo ocorrem entre dois nós sempre relacionando uma variável a uma cláusula. Não há arestas entre variáveis, nem arestas entre cláusulas, gerando a bipartição obrigatória do grafo. Desta forma, o conjunto de cláusulas $c_1 = x_1 \vee \neg x_2 \vee x_3$, $c_2 = \neg x_2 \vee \neg x_3$ e $c_3 = x_1 \vee \neg x_3$ gera o grafo que pode ser visto na Figura 5.

Figura 5 – Grafo de Cláusulas-Variáveis (VCG)



Fonte: Acervo do autor

2.2 Algoritmos de Otimização

Algoritmos de otimização exatos contêm estratégias que os habilite a obter soluções ótimas, desde que garantido os devidos tempos de execução, dependendo do tamanho da instância de um dado problema (BLUM; ROLI, 2003).

O conjunto de problemas NP-Difíceis contém problemas de otimização para os quais ainda não foram encontrados os algoritmos exatos que executem em tempo polinomial (PATURI et al., 2005). Algoritmos aproximados (ou heurísticos) podem gerar soluções de alta qualidade em um tempo satisfatório para o uso prático, todavia sem a garantia de otimalidade (RADER, 2010) (TALBI, 2009).

Entre os métodos exatos, existem algumas famílias principais de algoritmos, tais como programação dinâmica; *branch-and-bound* e derivados, desenvolvidos pela comunidade de pesquisa operacional, programação por restrições (*constraint programming*) (ANSóTEGUI; MALITSKY; SELLMANN, 2014); a família de algoritmos de busca A^*, desenvolvidos pela comunidade de inteligência artificial (TALBI, 2009); e a família de algoritmos de otimização baseados em gradiente (PRESS et al., 2007).

Os métodos não exatos se dividem entre métodos de aproximação e meta-heurísticas. Métodos de aproximação garantem que havendo uma solução ótima s, é possível obter uma solução aproximada a que satisfaça a desigualdade $(s-\epsilon) \leq a \leq (s+\epsilon)$, para qualquer valor escolhido de ϵ (TALBI, 2009).

Algoritmos meta-heurísticos são desenvolvidos para obter soluções que possam ser consideradas "boas"para instâncias de tamanhos tais, em que o tempo de execução de métodos exatos se torna proibitivo. No entanto, há um claro compromisso entre a qualidade da solução obtida e o esforço computacional dispendido pela meta-heurística; de forma que é sempre possível melhorar a solução à quanto maior for o tempo de execução dedicado ou capacidade computacional (OLIVEIRA, 2004).

Ao utilizar uma meta-heurística, existem duas abordagens contraditórias a serem levadas em conta durante o processo de busca. É necessário haver a exploração do espaço de busca, mantendo-se a diversidade de soluções candidatas e garantindo uma ampla amostragem do espaço de busca. Esse processo de diversificação tenta explorar todas as regiões do espaço de busca de forma igualitária. Porém, também é necessário manter a intensificação da busca, na proximidade das soluções candidatas de qualidade. Na estratégia de intensificação, os subespaços de busca mais promissores são explorados de forma mais detalhada na tentativa de encontrar soluções ainda melhores (OLIVEIRA, 2004).

É possível categorizar algoritmos de otimização de meta-heurísticas dependendo de vários fatores, qual sejam (TALBI, 2009):

- Solução única (*S-metaheuristic*) ou populacional (*P-metaheuristic*): meta-heurísticas de solução única, tais como *simmulated annealing* ou *Busca Tabu*, operam com única solução a cada iteração. Em contrapartida, meta-heurísticas populacionais trabalham com uma coleção de soluções candidatas a cada iteração. Estas famílias possuem características complementares: meta-heurísticas de solução única são mais orientadas a intensificar a busca e aproveitar o conhecimento da vizinhança; meta-heurísticas populacionais focam mais em exploração, tendo a capacidade de amostrar diferentes subespaços de busca.

- iterativa ou construtiva: meta-heurísticas construtivas começam de uma solução vazia, e a cada passo atribuem-se valores às variáveis do problema até obter-se uma solução completa. Heurísticas de vizinhança trabalham sempre com uma solução completa (*S-metaheuristic*) ou um conjunto de soluções completas (*P-metaheuristic*), explorando subespaços vizinhos.

Algoritmos Evolutivos (AE) são meta-heurísticas populacionais, que permite explorar mais de uma região do espaço de busca ao mesmo tempo (LINDEN, 2006), mantendo-se uma memória representativa das soluções encontradas ao longo da evolução por meio da sua população.

Nos casos em que essa memória de soluções e regiões promissoras é armazenada por meio dos próprios indivíduos e da população, o algoritmo é chamado populacional (TALBI, 2009). Neste caso, o conhecimento é passado de forma direta, por meio da interação entre os indivíduos nas diversas gerações. Dois algoritmos representativos desta classe são os algoritmos genéticos (GOLDBERG; HOLLAND, 1988) e a otimização por enxame de partículas (KENNEDY; EBERHART, 1997).

Algoritmos de Estimativa de Distribuição (*Estimation of Distribution Algorithms*), juntamente com Otimização por Colônia de Formigas (*Ant Colony Optimization*) (DORIGO; BIRATTARI; STUTZLE, 2006), formam uma família de algoritmos nos quais o

conhecimento é passado de forma indireta, por meio de um modelo probabilístico (*blackboard*) que representa uma população de soluções candidatas de qualidade. A cada geração, as melhores soluções amostradas são usadas para atualizar o *quadro negro* onde cada agente escreve suas descobertas visando compartilhar com os demais. Estes algoritmos EDA figuram entre os mais populares *blackboard* (PELIKAN; GOLDBERG; LOBO, 2002).

Segundo (BLUM; ROLI, 2003), as propriedades fundamentais que caracterizam as meta-heurísticas são:

- Meta-heurísticas são estratégias que guiam o processo de busca;
- O objetivo é explorar eficientemente o espaço de busca a fim de encontrar soluções ótimas ou quase-ótimas;
- Técnicas que constituem as meta-heurísticas variam dos procedimentos de busca local para processos complexos de aprendizagem;
- Algoritmos meta-heurísticos são aproximados e usualmente não-determinísticos;
- Incorporam usualmente mecanismos para escaparem de áreas confinadas do espaço de busca;
- Os conceitos básicos de meta-heurísticas permitem um nível abstrato de descrição;
- Meta-heurísticas não são especializadas em um determinado problema
- Conhecimento de domínio específico para controle de heurísticas de baixo nível pode ser empregado como estratégia de nível maior;
- Meta-heurísticas podem usar experiência ou mecanismo de memória para guiar a busca.

2.2.1 Algoritmo Genético

Algoritmos Genéticos (AG) são um tipo de algoritmo evolutivo (HOLLAND, 1975) que empregam uma estratégia adaptativa inspirada pela genética e evolução populacional, incluindo ideias como hereditariedade, frequência gênica, bem como pelo entendimento Mendeliano de estrutura, tal como cromossomos, genes e alelos, e dos mecanismos, tais como recombinação e mutação (BROWNLEE, 2011).

Sendo inspirado no processo de seleção natural, o AG considera que cada solução completa a ser trabalhada é um indivíduo. Este indivíduo tem uma medida chamada *fitness* que caracteriza o quão adequado o indivíduo está ao seu meio. Esta medida pode ser considerada como os valores atingidos na função objetivo a ser maximizada.

Em seguida, os melhores indivíduos são selecionados por algum mecanismo. Estes indivíduos passam por uma troca de material genético e contribuem para sua prole através da recombinação de genes. Além disso, é possível que haja erros aleatórios, denominados mutações, em diversos indivíduos.

Após este processo, todos ou alguns indivíduos da geração anterior morrem (são descartados), e a prole é considerada como uma nova geração do algoritmo. Com isto, espera-se que os indivíduos de cada geração ou iteração sucessiva estejam mais adaptados ao ambiente, ou seja, possuam um *fitness* melhor e portanto sejam soluções melhores para o problema de otimização.

O Algoritmo 1 mostra a estrutura de um Algoritmo Genético:

Algoritmo 1 Algoritmo Genético

Entrada: O tamanho da população P_{size}, $P_{crossover}$, $P_{mutacao}$
Saída: o melhor global $\mathbf{s}_{best}$.

```
1: Função objetivo f(x), x = (x_1, ..., x_d)^T
2: Gere a população inicial x_i (i = 1, 2, ..., n)
3: enquanto ¬ CriterioDeParada() faça
4:     Pais ← SelecionaPais(Populacao, P_size)
5:     Filhos ← ∅
6:     para cada Pai_1, Pai_2 ∈ Pais faça
7:         Filho_1, Filho_2 ← Crossover(Pai_1, Pai_2, P_crossover)
8:         Filhos ← Mutacao(Filho_1, Filho_2, P_mutacao)
9:     fim para
10:    AvaliaPopulacao(Filhos)
11:    s_best ← MelhorSolucao(Filhos)
12:    Populacao ← Substitui(Populacao, Filhos)
13: fim enquanto
14: retorne s_best
```

Cada uma das etapas descritas de seleção, cruzamento ou acasalamento e mutação utiliza-se de operadores definidos na literatura. Estes operadores são descritos a seguir.

2.2.1.1 Operadores de Seleção

Para os operadores de seleção, há uma gama de opções na literatura. Como as meta-heurísticas não se aplicam a apenas um problema, é importante ressaltar que esta lista não é extensiva, se resumindo aos operadores de seleção mais comumente encontrados na pesquisa literária realizada.

Um conceito importante para a escolha de operadores de seleção em algoritmos genéticos é a pressão seletiva. A pressão seletiva é definida como a relação entre a probabilidade de seleção do melhor indivíduo com a probabilidade de seleção do indivíduo médio.

Um dos primeiros operadores usados para seleção é chamado de Amostragem Estocástica com Reposição, usualmente chamado de "seleção por roleta"(BAKER, 1987). Neste método, a probabilidade de escolha de cada indivíduo pode ser descrita como $p_i = \frac{f_i}{\sum_{j \in P} f_j}$ onde f_n é a avaliação de *fitness* do n-ésimo indivíduo. Um dos problemas com este método de seleção é que ele somente funciona diretamente em problemas de maximização, sendo necessário realizar uma transformação para que ele possa também operar em problemas de minimização. Um outro ponto importante é que ele não permite o controle de parâmetros como a pressão seletiva; é possível haver um indivíduo dominante com uma chance de seleção próxima de 100%, assim como é possível que haja um equilíbrio e todos os indivíduos tenham chances semelhantes, especialmente em problemas com um *fitness* grande e pouca variação relativa, o que pode gerar uma estagnação da população.

De forma a propor uma técnica de seleção mais robusta, foi proposta uma técnica baseada no *ranking* de cada indivíduo (BÄCK; HOFFMEISTER, 1991). Nesta técnica, após os indivíduos da população serem avaliados de acordo com seu *fitness* real, eles são ordenados e recebem um valor de *dummy-fitness* de acordo com sua posição. O fato de que os valores de *dummy-fitness* são pré-determinados faz com que seja possível que este operador de seleção funcione sempre com uma pressão seletiva específica, por garantir que a chance do melhor indivíduo seja sempre a mesma. Dentro deste algoritmo, foram propostas duas principais variações, uma utilizando uma função linear de acordo com o *ranking* para o *dummy-fitness* e outra baseada em um *ranking* não linear. A técnica linear permite que sejam selecionadas pressões seletivas dentro do intervalo $[1.0 - 2.0)$, enquanto a técnica de *ranking* não linear permite a seleção de pressões seletivas no intervalo $[1.0 - (n - 2)]$ sendo n o tamanho da população. Como esta técnica se utiliza da ordenação dos indivíduos após avaliação da função objetivo, ela funciona bem para problemas de maximização e minimização, sem nenhuma necessidade de ajuste.

Uma outra técnica proposta é chamada de Seleção por Torneio Determinístico, ou simplesmente Seleção por Torneio. Nesta técnica, k indivíduos são selecionados de forma uniformemente aleatória da população. Dentre estes k indivíduos, o com o maior valor de *fitness* é selecionado. É possível perceber que à medida em que se aumenta o tamanho do torneio (k), a chance de que o melhor indivíduo esteja no torneio (e portanto vença) aumenta. Desta forma, há uma possibilidade de regulação indireta da pressão seletiva. Devido à complexidade da determinação exata da pressão seletiva neste método de seleção, o detalhamento não será exposto neste trabalho. Um estudo mais detalhado da determinação da pressão seletiva para o torneio determinístico pode ser vista em (BLICKLE; THIELE, 1996).

A última técnica é chamada de Seleção por Torneio Estocástico (GOLDBERG; DEB, 1991). Esta técnica é similar à Seleção por Torneio, com a diferença de que o torneio sempre composto de dois indivíduos selecionados uniformemente aleatoriamente, e que há

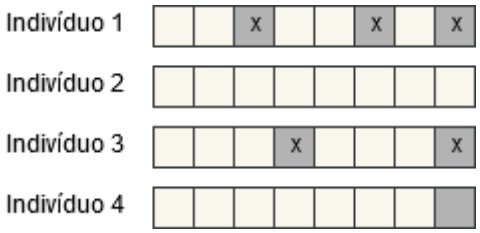
uma chance parametrizada de que um dos dois indivíduos seja selecionado aleatoriamente, ao invés de sempre selecionar o melhor. Com isso, também é possível controlar a pressão seletiva. Ela pode ser reduzida ao aumentar a chance de seleção de um indivíduo aleatório no torneio.

2.2.1.2 Operadores de Mutação

Dado que o problema a ser tratado neste trabalho é um problema de atribuição binária booleana, será realizada a descrição apenas de operadores de mutação que operam neste tipo de indivíduos. É importante notar que estes operadores não se aplicam a outros problemas, como problemas de atribuição de valores reais ou problemas de permutação e sequenciamento.

O primeiro operador de mutação utilizado, o *bitflip*, foi proposto originalmente em (GOLDBERG; HOLLAND, 1988). Este operador é aplicado em todos os indivíduos da população. Para cada gene de cada indivíduo há uma probabilidade p de que este gene seja alternado logicamente. Considerando $p = 0.125$, um exemplo do comportamento deste operador de mutação pode ser visto na Figura 6, onde os genes selecionados para mutação estão demarcados.

Figura 6 – Mutação por *bitflip*

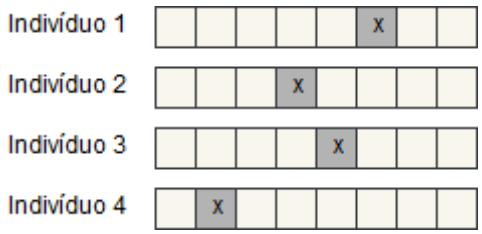


Fonte: Acervo do autor

Um algoritmo diferente para mutação é chamado de *bitflip* determinístico. Enquanto no *bitflip* original há a alteração média de $p\%$ dos genes, no *bitflip* determinístico sempre são alternados exatamente $p\%$ bits de cada indivíduo selecionado para mutação. É possível visualizar na Figura 7 um exemplo de aplicação deste operador para $p = 0.125$.

Apesar de os dois algoritmos alterarem a mesma quantidade de bits na média, o comportamento de ambos são bem diferentes. É perceptível pelas figuras que o algoritmo de *bitflip* possui chances independentes, enquanto o *bitflip* determinístico altera sempre a mesma quantidade de genes de cada indivíduo. Isto leva a diferenças na execução do algoritmo, considerando diversas iterações por meio de diversas gerações.

Figura 7 – Mutação por *bitflip* determinístico



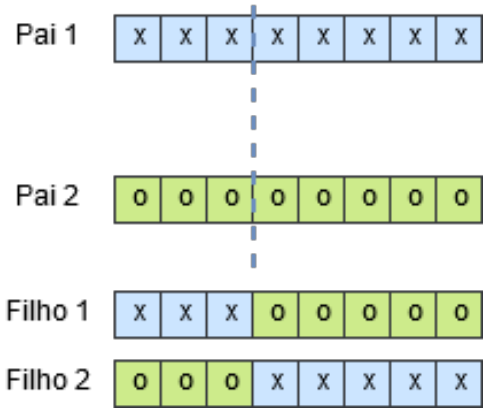
Fonte: Acervo do autor

2.2.1.3 Operadores de Cruzamento

Para a operação de cruzamento, também há uma vasta literatura com proposta de diferentes operadores. Alguns dos operadores da literatura funcionam apenas com problemas de atribuição real, há operadores que funcionam com mais de dois "pais". No entanto, para utilização neste trabalho são considerados apenas operadores que trabalhem com vetores de bits e com apenas dois "pais".

O primeiro operador de cruzamento proposto é o cruzamento simples de um único ponto (GOLDBERG; HOLLAND, 1988). Neste operador, um ponto aleatório de corte é selecionado. O primeiro filho recebe os genes do primeiro pai até o ponto de corte, e o restante dos genes são advindos do segundo pai. Similarmente, o segundo filho recebe a primeira parte do segundo pai até o ponto de corte, e recebe a segunda parte do primeiro pai. Um exemplo deste operador de cruzamento, com o ponto de corte tracejado, pode ser visto na Figura 8.

Figura 8 – Cruzamento simples de único ponto

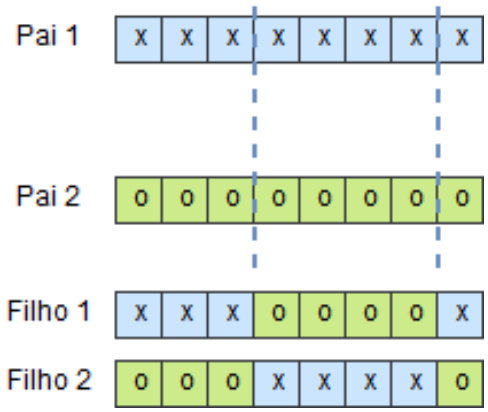


Fonte: Acervo do autor

O segundo operador de cruzamento é chamado cruzamento multi ponto. Nesta técnica, são selecionados n pontos de cruzamento. O primeiro filho inicia copiando os genes do primeiro pai, e toda vez que ele atinge um ponto de corte determinado ele alterna o pai

de origem dos genes. O segundo filho opera similarmente, porém iniciando sua sequência recebendo os genes do segundo pai (JONG; SPEARS, 1992). Este operador pode ser melhor visualizado na Figura 9, onde os pontos de cruzamento estão demarcados com uma linha tracejada. Para exemplificação, foi utilizado um cruzamento com dois pontos.

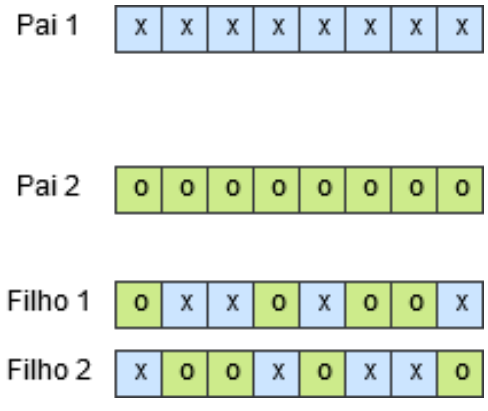
Figura 9 – Cruzamento multi ponto com dois pontos



Fonte: Acervo do autor

O último operador é chamado de cruzamento uniforme. Neste cruzamento, não há a determinação de pontos específicos de corte. Para cada gene, é realizada uma seleção aleatória entre os dois pais, gerando uma chance de 50% que cada gene venha de cada pai. Esta seleção é aleatoriamente independente, garantindo que os indivíduos gerados tenham uma chance uniforme de que cada gene tenha vindo do primeiro ou segundo pai (SYSWERDA, 1989). Este cruzamento pode ser visto na Figura 10, onde é possível visualizar que não há pontos pré-definidos de corte, cada gene pode vir de qualquer um dos dois pais.

Figura 10 – Cruzamento uniforme



Fonte: Acervo do autor

2.2.2 Enxame de Partículas

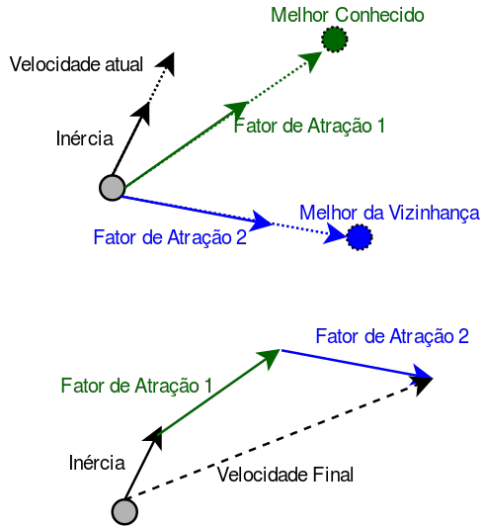
A otimização por enxame de partículas, ou PSO (*Particle Swarm Optimization)* é outra meta-heurística de uso geral. Esta técnica foi inspirada em enxames biológicos e se baseia em princípios de relacionamentos sociais e psicológicos (KENNEDY, 2010).

Uma característica importante desta heurística e que a destaca das demais meta-heurísticas populacionais baseadas em evolução é que não há a morte de nenhum indivíduo, da primeira iteração até a última.

Ao invés de mecanismos de morte e geração de indivíduos, os indivíduos são representados como pontos no espaço de busca. Estes indivíduos são chamados de *partículas*. Estas partículas estão em algum local específico do espaço de busca, e estão se deslocando durante a exploração deste espaço. Elas possuem uma certa inércia ou peso para o movimento, além de serem influenciadas pelo melhor local já visitado por ela e pelo melhor local já visitado globalmente. Desta forma, exibe um comportamento de clusterização com a criação de enxames, que tendem a concentrar em regiões promissoras do espaço de busca.

A operação de voo considerando o melhor conhecido pela partícula, o melhor da vizinhança que ela possui visualização e a posição atual pode ser vista na Figura 11. Para o exemplo contido na Figura, considera-se o peso $w < 1$, e os parâmetros de atração do ótimo conhecido $l_1 < 1$ e o parâmetro de atração do ótimo da vizinhança $l_2 < 1$. A nova velocidade durante o voo é a soma vetorial destas grandezas. A equação pode ser descrita por $V_{t+1} \leftarrow w * V_t + l_1 * (SELFBEST_i) + l_2 * (NEIGHBORBEST_i)$.

Figura 11 – Operação de voo no PSO



Fonte: Adatado de (EBERHART; SHI, 1998)

O Algoritmo 2 mostra a estrutura de um Algoritmo de Otimização por Enxame de Partículas:

Algoritmo 2 *Particle Swarm Optimization*

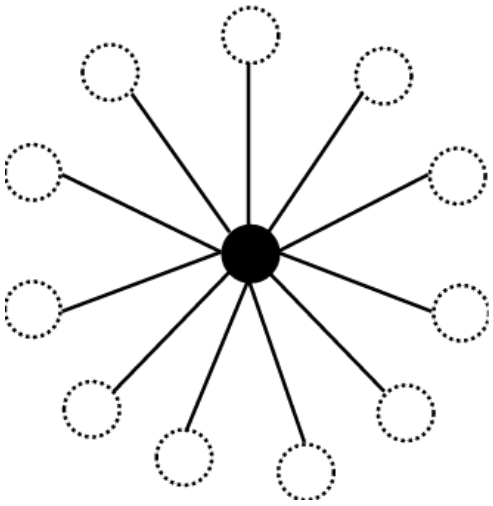
Entrada: O tamanho da população P_{size}, operador de vizinhança N
Saída: o melhor global $\mathbf{s}_{best}$.

1: Função objetivo $f(x)$, $\mathbf{x} = (x_1, ..., x_d)^T$
2: Gere os indivíduos iniciais $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
3: A posição inicial é a melhor conhecida; $best_i \leftarrow x_i$
4: Gere as velocidades iniciais v_i $(i = 1, 2, ..., n$
5: **enquanto** ¬ CriterioDeParada() **faça**
6: $nbest_i \leftarrow$ MelhorVizinhança(x_i, N)
7: Atualize a velocidade; $v_i \leftarrow w * v_i + l_1 * (selfBest_i) + l_2 * (neighborBest_i)$
8: Atualize a posição atual; $x_i \leftarrow$ OperadorVoo(x_i, v_i)
9: **se** $f(x_i) < f(best_i)$ **então**
10: $best_i \leftarrow x_i$
11: **fim se**
12: $\mathbf{s}_{best} \leftarrow$ MelhorSolucao(Populacao)
13: **fim enquanto**
14: **retorne** $\mathbf{s}_{best}$

2.2.2.1 Operadores de Vizinhança

O conceito original de influência do ótimo global foi mais tarde evoluído para incluir a possibilidade de diferentes topologias sociais, visando escapar da influência de algum ótimo local (KENNEDY, 1999).

Figura 12 – Vizinhança em Estrela



Fonte: Acervo do autor

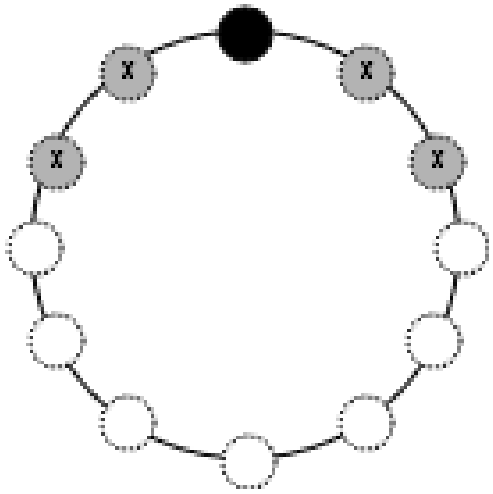
As principais topologias em uso na literatura são:

- Topologia em estrela (*Star Topology*): esta é a vizinhança original - todas as partículas estão conectadas a todas as demais partículas. Desta forma, o ótimo da vizinhança é

o ótimo global conhecido de toda a população. Um exemplo desta vizinhança em uma população de tamanho 12 pode ser visto na Figura 12.

- Topologia em anel (*Ring Topology*): nesta topologia, todos os indivíduos são colocados em uma estrutura em anel durante a criação. O tamanho da vizinhança é definido como um parâmetro configurável n. O ótimo global enxergado pela partícula passa a ser o melhor ponto conhecido dos indivíduos que estejam a uma distância de até $\frac{n}{2}$ no anel do indivíduo sendo avaliado. Esta topologia de vizinhança pode ser vista na Figura 13 para uma população de 12 indivíduos e tamanho de vizinhança $n = 4$, onde o indivíduo em avaliação pode ser visto em preto e os vizinhos estão demarcados. Os demais indivíduos não serão considerados na avaliação da vizinhança do indivíduo sinalizado.

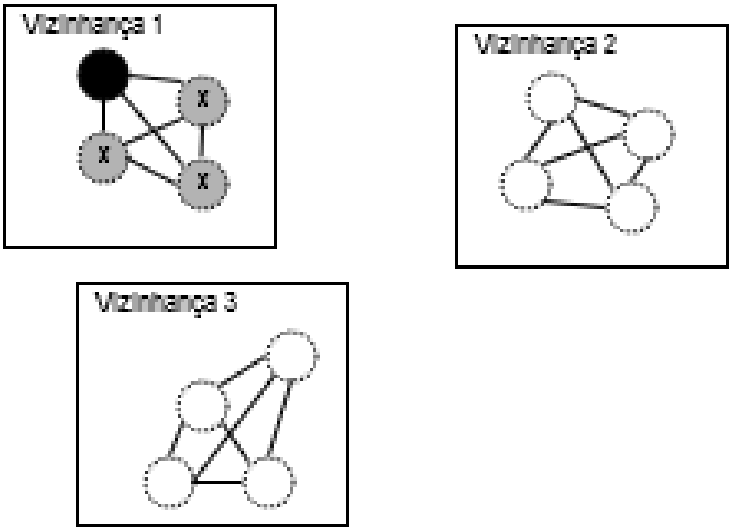
Figura 13 – Vizinhança em Anel



Fonte: Acervo do autor

- Topologia Linear (*Linear Topology*): esta topologia trabalha com diferentes vizinhanças sem sobreposição. Considerando o parâmetro configurável de tamanho de vizinhança como n e o número de indivíduos total da polução como n_{ind}, cada vizinhança terá associado a ela $\frac{n_{ind}}{n}$ indivíduos ou partículas. Cada vizinhança não tem conhecimento das demais vizinhanças, tornando esta topologia o equivalente a executar diversos algoritmos de PSO com populações menores em paralelo. Esta topologia pode ser vista na Figura 14 para uma população de 12 partículas e para $n = 4$, onde é possível ver a demarcação de 3 vizinhanças independentes. O indivíduo em avaliação está sinalizado de preto, e os vizinhos estão sinalizados. Todas as conexões entre todos indivíduos estão demarcadas na Figura 14 também.

Figura 14 – Vizinhança Linear



Fonte: Acervo do autor

2.2.2.2 Operador de Voo

Devido a sua característica de movimentação em um espaço de busca n-dimensional, o PSO foi desenvolvido originalmente para problemas de atribuição de valores reais. Desta forma, foi necessária a proposta de alterações para operar com esta meta-heurística em problemas de decisões binárias booleanas. Na adaptação, a posição (variável binária) passa a ser definida pela velocidade, que representa a chance de que aquela coordenada (ou variável no caso do MaxSAT) seja 0 ou 1. Este operador de voo para PSO foi proposto em (KENNEDY; EBERHART, 1997).

Neste operador, a posição atual (variável binária de decisão) terá valor **verdadeiro** caso $rand[0,1] <= sigmoid(V_i)$, sendo V_i a velocidade da partícula na coordenada i. Caso contrário, a variável terá valor **falso**.

2.2.3 Aprendizagem Incremental Baseada em População

Algoritmos de estimativa de distribuição (*Estimation of Distribution Algorithm* - EDA), são uma das técnicas de otimização por quadro negro. Nesta família de meta-heurísticas, o espaço de busca é explorado através da amostragem estatística por meio da geração de indivíduos, seguida por uma atualização das regras do modelo estatístico capaz de gerar os indivíduos (PELIKAN; GOLDBERG; LOBO, 2002).

Diversos algoritmos se encaixam dentro desta descrição. Os mais comuns são o algoritmo de otimização Bayesiano (*Bayesian Optimization Algorithm* - BOA), a aprendizagem incremental baseada em população (*Population Based Incremental Learning* - PBIL), o algoritmo genético compacto (*compact Genetic Algorithm* - cGA) e o algoritmo

de estimação de redes bayesianas (*Estimation of Bayesian Networks Algorithm* - EBNA) (PELIKAN; GOLDBERG; LOBO, 2002).

Uma das primeiras técnicas propostas da família dos EDA's foi o PBIL. Esta técnica foi proposta e opera em problemas de atribuição binária, e desta forma pode ser utilizada diretamente no problema MaxSAT (BALUJA; CARUANA, 1995).

Algoritmo 3 *Population Based Incremental Learning*

Entrada: O tamanho da população P_{size}, taxas de aprendizagem l_{best}, l_{worst}, quantidade de indivíduos a aprender n_{best}, n_{worst}
Saída: o melhor global $\mathbf{s}_{best}$.

1: Função objetivo $f(x)$, $\mathbf{x} = (x_1, ..., x_d)^T$
2: Gere a distribuição inicial $p_i = 0.5$ $(i = 1, 2, ..., n)$
3: **enquanto** ¬ CriterioDeParada() **faça**
4: Gere os indivíduos $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
5: AvaliaPopulacao(Populacao)
6: Atualize a probabilidade de cada gene; $p_i = (1 - \eta_b - \eta_w) * p_{i-1} + \eta_b * \frac{\sum_{n<n_b} p_n}{n_b} - \eta_w * \frac{\sum_{n>n_{ind}-n_w} p_n}{n_w}$
7: $\mathbf{s}_{best} \leftarrow$ MelhorSolucao(Populacao)
8: Descarte a população; $Populacao \leftarrow \phi$
9: **fim enquanto**
10: **retorne** $\mathbf{s}_{best}$

2.2.3.1 Codificação do Modelo

Nesta técnica, a codificação do modelo é bastante simples. O modelo é apenas um vetor de probabilidades. Ao gerar os indivíduos, é realizada a seleção aleatória de cada gene de acordo com a sua probabilidade de ser **verdadeiro**.

Inicialmente, este vetor possui todos os valores em 0.5, garantindo uma chance uniforme de seleção de cada variável entre **verdadeiro** e **falso**. Após a amostragem gerando indivíduos com esta regra, a probabilidade da variável ser verdadeira é ajustada de acordo com a performance dos indivíduos.

2.2.3.2 Regra de atualização do modelo

Para a atualização do modelo estatístico, foi proposto inicialmente uma regra simples, que levava em consideração apenas o indivíduo com o melhor *fitness*. Para cada uma das variáveis, a probabilidade de ser **verdadeira** era atualizada pela seguinte regra apresentada na Equação 2.1, onde p_i é a probabilidade da i-ésima variável assumir valor verdadeiro, η é um parâmetro configurável chamado de taxa de aprendizagem e $best_i$ é o valor obtido para aquele gene i no melhor indivíduo da população.

$$p_i = (1 - \eta) * p_{i-1} + \eta * best_i \tag{2.1}$$

Simplificadamente, essa regra ajustava para cada variável a chance dela ser positiva de acordo com o melhor indivíduo amostrado usando esta distribuição, aumentando ou diminuindo a probabilidade por um fator η.

Posteriormente, os mesmos autores propuseram um PBIL aditivo (BALUJA; CARUANA, 1995). Com esta nova proposta, o algoritmo poderia detectar regiões promissoras com mais de um indivíduo, e pode aprender tanto com indivíduos bons quando com indivíduos considerados ruins.

A nova regra de atualização de probabilidade proposta, chamada de PBIL aditivo, é apresentada na Equação 2.2, onde η_b é a taxa de aprendizagem considerada para os melhores indivíduos, η_w é a taxa de aprendizagem considerada para os piores indivíduos, e n_{ind}, n_b e n_w são respectivamente o número de indivíduos na população, e dois parâmetros adicionais configuráveis - n_b é o número de melhores indivíduos da população para aprender e n_w é o número de piores indivíduos da população para aprendizado.

$$p_i = (1 - \eta_b - \eta_w) * p_{i-1} + \eta_b * \frac{\sum_{n<n_b} p_n}{n_b} - \eta_w * \frac{\sum_{n>n_{ind}-n_w} p_n}{n_w} \tag{2.2}$$

É possível que a probabilidade de uma variável ser **verdadeira** atinja 100% ou 0%, o que efetivamente interromperia a exploração desta variável no espaço de busca, pois a variável irá assumir um valor fixo em todos os indivíduos gerados. Para evitar este efeito, há um parâmetro t, chamado de tolerância, com $0 \leq t < 0.5$, de forma que nenhuma probabilidade pode ser menor do que t ou maior do que $1 - t$, ficando limitadas ao intervalo $[t, 1 - t)$

2.3 Aprendizagem de Máquina

Segundo Mitchell (MITCHELL, 1997), um programa de computador é capaz de aprender uma classe de tarefas T, se a métrica de desempenho P melhora com a experiência E. Em outras palavras, o aprendizado de máquina pode ser definido em função de classe de tarefas, métrica de desempenho e experiência adquirida. Este tipo de definição faz com que os problemas de aprendizagem de máquina tenham proximidade e em alguns casos sobreposição com problemas de estatística computacional e otimização.

Uma dos grandes pontos fortes de técnicas de aprendizagem de máquina é que estas podem ser aplicadas a problemas complexos com muitas variáveis, onde pessoas são propensas a cometer erros ou não avaliar todas as possibilidades durante análises das relações entre diversas características do problema (KOTSIANTIS, 2007).

Há hoje na literatura três paradigmas de Aprendizagem de Máquina (KOTSIANTIS, 2007) (MITCHELL, 1997):

- **Aprendizagem Supervisionada**, ou *Supervised Learning*, é um paradigma que trata de problemas de aprendizagem de máquina onde cada instância do conjunto de treinamento tem um rótulo já pré-definido, e os algoritmos utilizam este rótulo em conjunto com as características dos exemplos como guias para reduzir o seu erro frente ao problema. O nome deriva de que a aprendizagem de máquina é realizada através da supervisão de uma entidade que atua como uma espécie de "professor", repassado para o algoritmo quais os erros e quais as respostas esperadas. Caso os rótulos das instâncias sejam contínuos, o problema é considerado como um problema de regressão. Caso os rótulos sejam categorizados (discretos e não ordenados) o problema é considerado um problema de classificação.

- **Aprendizagem não supervisionada**, ou *Unsupervised Learning*, é um paradigma utilizado para problemas de aprendizagem em que não há a rotulação prévia dos exemplos. Neste caso, não há uma referência anterior, de forma que o algoritmo não consegue medir seu nível de performance diretamente. Estes algoritmos trabalham com a inferência de relações entre os exemplos e a clusterização destes. Em geral, são utilizados para identificar padrões ocultos nos dados, sendo frequentemente utilizados em cenários de *Data Mining*.

- **Aprendizagem por reforço**, ou *Reinforcement Learning*, trata de casos em que o programa de computador interage com algum ambiente dinâmico em que ele necessite realizar alguma tarefa, tal como chegar com sucesso em um alvo, dirigir um carro ou vencer um oponente. O programa obtém um sinal de reforço do ambiente para as ações tomadas, de forma que ele consegue perceber recompensas e punições ao realizar suas ações no ambiente. No entanto, ele deve aprender uma estratégia ou política de ações de forma a atingir os objetivos finais. Além disso, ao contrário da aprendizagem supervisionada, é que apesar de receber um sinal de reforço ou punição do ambiente, o programa não sabe qual a melhor ação, e precisa aprender a melhor ação não somente no momento atual mas também considerando a sequencia de eventos necessária para atingir o objetivo.

2.3.1 Redes Neurais

Redes Neurais foram inspiradas pela biologia, a partir da qual foram propostos modelos computacionais de neurônios, como o proposto por McCulloch e Pitts (MCCULLOCH; PITTS, 1943), considerado a pedra fundamental dos estudos em redes neurais computacionais (HAYKIN, 1998).

Posteriormente, o modelo foi refinado com a introdução do *Perceptron* que é capaz de realizar qualquer tipo de aprendizado baseado em separação linear (ROSENBLATT,

1958), não sendo, portanto, capaz de realizar a classificação para problemas não linearmente separáveis.

Aprendizado computacional aplicado a problemas não linearmente separáveis progrediu com a introdução do algoritmo de Retro-propagação (*Backpropagation*) que permite a propagação do sinal de erro por meio de múltiplas camadas de neurônios, daí o nome dado à rede: Perceptron Multicamadas (*Multilayer Perceptron - MLP*) (LECUN et al., 1998). A propagação para determinar o erro dos neurônios das camadas escondidas é realizada através da Equação 2.3, onde e_i é a função erro para o i-ésimo neurônio, o conjunto M é o conjunto de neurônios conectados à saída do neurônio n para o qual se deseja calcular o erro, o_i é a saída do i-ésimo neurônio e w_{ij} é o peso da conexão entre os neurônios i e j.

$$e_n = \sum_{m \in M} \frac{\partial e_m}{\partial o_m} \cdot w_{nm} \tag{2.3}$$

Este método de aprendizagem supervisionado se baseia na utilização de técnicas de otimização por descida de gradiente para determinar o conjunto ideal de pesos entre todas as camadas de forma a minimizar o erro do neurônio de saída.

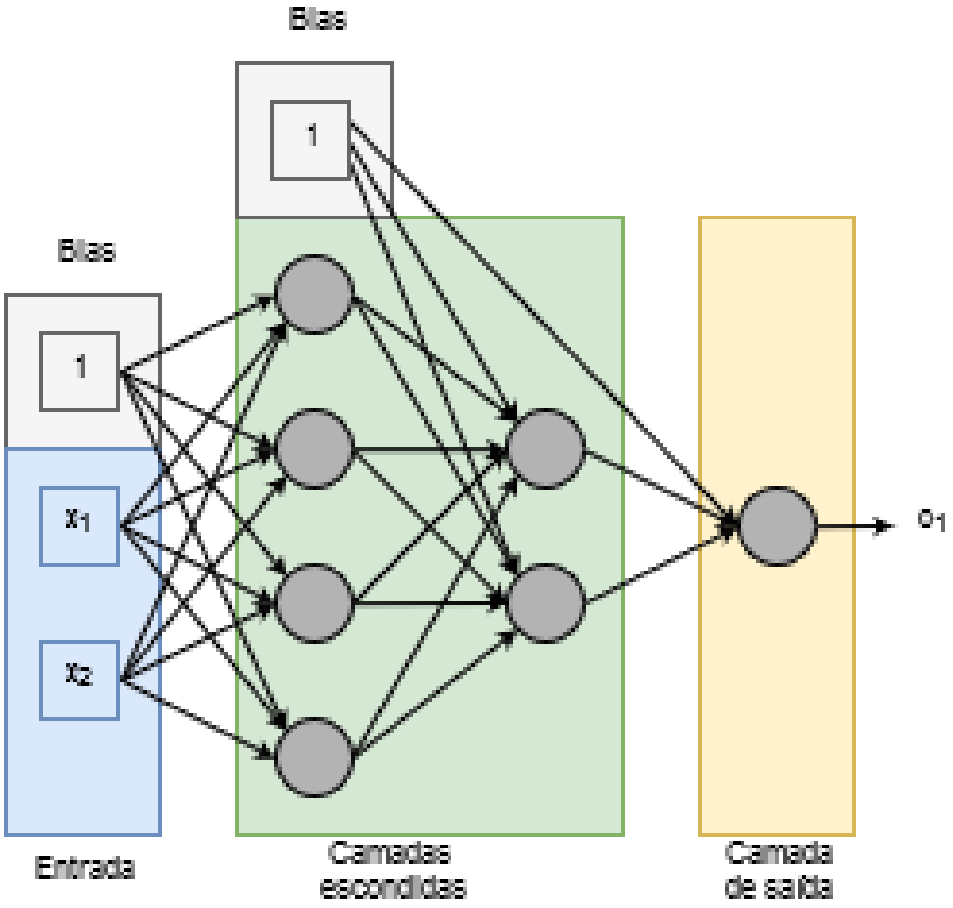
Uma MLP pode ser definida como um grafo direcionado onde cada nó é um neurônio e cada aresta representa a conexão entre os neurônios, sendo o peso da aresta a força da conexão. Uma rede MLP deve ter no mínimo uma camada de um ou mais perceptrons, sendo a última (ou a única, no caso de haver apenas uma camada) chamada de camada de saída (*Output Layer*). Adicionalmente, é possível que a rede tenha qualquer número de camadas intermediárias escondidas (DATT, 2012).

Em uma rede neural MLP totalmente conectada, cada variável de entrada é conectada a um neurônio da camada de entrada cuja saída é exatamente igual à entrada. Cada neurônio, seja da camada de entradas ou das camadas escondidas, é conectado a todos os neurônios da camada seguinte, sucessivamente até a camada de saída. Os neurônios da camada de saída são responsáveis pela saída da rede, e portanto a saída lida destes é o resultado final da classificação.

Um exemplo desta rede pode ser visto na Figura 15, onde as entradas são denominadas x_i e a saída da rede é denominada o_i. Os neurônios das camadas de entrada, das camadas escondidas e da camada de saída estão representados e identificados também, para melhor visualização.

A saída de cada um dos neurônios das camadas escondidas ou da camada de saída é dada pela equação $o_n = f(\sum w_{mn} * o_m + b_n)$, sendo w_{mn} o peso da aresta entre os nós m e n, b_n o *bias* do neurônio n, o_m a saída do neurônio m e sendo $f(\cdot)$ a função de ativação dos neurônios.

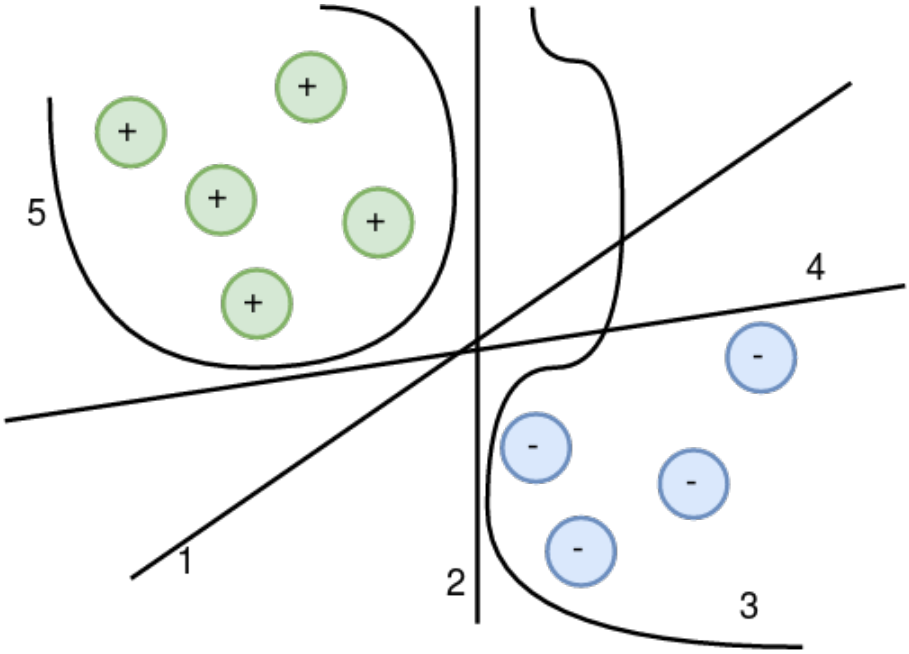
Figura 15 – Rede neural totalmente conectada



Fonte: Acervo do autor

Um dos pontos importantes de redes neurais é que elas desenvolvem hiperplanos que separem as classes do problema, porém elas não possuem nenhuma garantia com relação a esses hiperplanos. Desta forma, qualquer função que realize a separação é considerada tão apta quanto as demais. É possível perceber na Figura 16 que possui a curva de classificação gerada a partir de 5 redes neurais distintas. Todas as curvas possuem com uma performance de taxa de acerto de 100% na classificação, e portanto não há como direcionar a atualização dos pesos, possuindo todas a mesma performance para efeito de aprendizagem via rede neural.

Figura 16 – Redes Neurais com performance equivalente



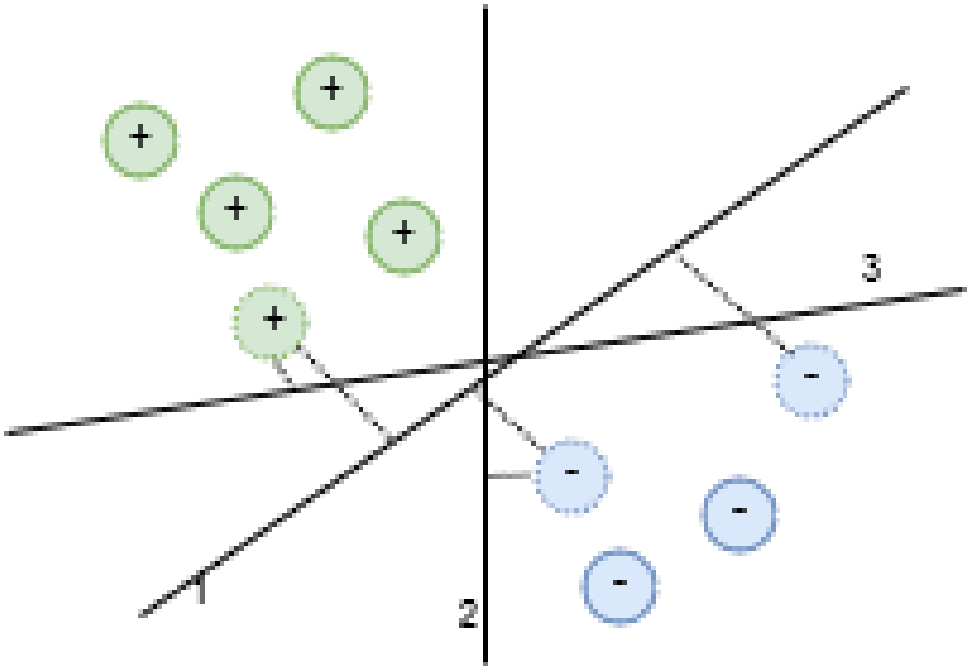
Fonte: Acervo do autor

2.3.2 Máquinas de Vetor Suporte

As máquinas de vetor suporte (*Support Vector Machines* - SVM) foram criadas de forma a tentar encontrar resultados que garantam uma certa estabilidade de classificação. Intuitivamente, é possível verificar que algumas curvas podem ser consideradas intuitivamente como mais ajustadas ao problema. Comparando a classificação das retas na Figura 17, geralmente a função proposta identificada pela reta número 1 possui uma distância melhor das duas classes, sendo considerada uma função com uma baixa probabilidade de erro futuro (HEARST et al., 1998).

Desta forma, foi criado o conceito de **margem**. A margem pode ser definida como a distância mínima do hiperplano de classificação ao ponto de fronteira mais próximo. Com este conceito, as três retas que resolvem o problema proposto na Figura 17 possuem margens diferentes, sendo a primeira delas considerada melhor por possuir a maior margem (HEARST et al., 1998).

Figura 17 – Separação Linear e margens



Fonte: Acervo do autor

As SVM formam hiperplanos que têm a maior margem possível, de forma a tentar reduzir o risco de erro de generalização do modelo construído. A técnica comum é encontrar os pontos de fronteira (vetores de suporte), e construir um hiperplano de forma intermediária entre estes pontos. Na Figura 17, os três vetores de suporte presentes foram sinalizados através de uma borda tracejada. O problema de maximização da margem é um problema de otimização quadrática, e portanto pode ser resolvido por técnicas comuns na literatura como descida de gradiente.

A garantia da máxima margem normalmente melhora a generalização para problemas linearmente separáveis, como pode ser visto por exemplo em (BYVATOV et al., 2003). No entanto, grande parte dos problemas do mundo real não pode ser modelado como linearmente separáveis. Neste caso, utiliza-se de funções chamadas *kernel functions*, que fazem a projeção dos dados para um espaço multidimensional com uma maior quantidade de dimensões, com o objetivo de obter um problema que seja linearmente separável num espaço de dimensões aumentadas, e o hiperplano é construído neste espaço de dimensões aumentadas (MEYER; WIEN, 2014).

2.3.3 K vizinhos mais próximos

O algoritmo de K vizinhos mais próximos (K *Nearest Neighbors* - KNN) foi proposto por (COVER; HART, 1967). Este algoritmo de classificação considera que o vetor de características é um vetor em um espaço n-dimensional, e armazena a posição e

classe de todo o conjunto de treinamento.

Ao realizar a classificação de uma instância não vista, o KNN calcula a distância para todos os pontos conhecidos, e seleciona os K vizinhos mais próximos. A instância é classificada como a classe mais comum dentre os K vizinhos (COVER; HART, 1967).

2.4 Considerações Finais

Neste capítulo, foi realizada a revisão bibliográfica e a apresentação de conceitos fundamentais. Reforça-se a importância da base providenciada pelas meta-heurísticas apresentadas, para aplicação no problema MaxSAT. Por fim, as técnicas de aprendizagem de máquina serão utilizadas para realizar a meta-aprendizagem, que será definida no Capítulo 3.

3 Meta-Aprendizagem

Neste capítulo, os principais conceitos inerentes à meta-aprendizagem são apresentados, bem como são discutidos algumas propostas encontradas na literatura para a aplicação desses conceitos ao problema MaxSAT.

3.1 Histórico

O campo da meta-aprendizagem surgiu a partir do problema de seleção de algoritmo. Este problema pode ser brevemente descrito como: *Qual algoritmo é mais provável de performar melhor para esta instância específica do problema?* (RICE, 1976), tendo sido criada esta definição por Rice em 1976.

Este processo iniciou-se com o uso de algoritmos de classificação. Portanto, as primeiras aplicações de meta-aprendizagem ou *meta-learning* foram nos próprios problemas de classificação - dada uma instância do problema, qual o melhor algoritmo para realizar a classificação (THRUN; PRATT, 2012).

Para realizar esta classificação de algoritmos, foi necessária a criação de um novo conjunto de características que descrevessem a instância e não mais o problema em si. Como a aprendizagem está operando em um segundo nível, chamado de nível meta, essas características foram chamadas de meta-características ou *meta-features* (SMITH-MILES, 2009).

O problema de meta-aprendizagem pode ser considerado como um problema de classificação, pois dado um conjunto de treinamento com características do conjunto de dados, e um rótulo com qual algoritmo de classificação este conjunto de dados obteve a melhor performance na classificação, é viável treinar algoritmos comuns de classificação para tentar encontrar qual a classe mais provável de um novo problema, ou qual o melhor algoritmo para resolver esta instância.

3.2 Meta-Aprendizagem em Otimização

O campo de meta-aprendizagem é bastante utilizado para aprendizagem de máquina, conforme visto na subseção anterior.

No entanto, há casos de usos de meta-aprendizagem em otimização. Um exemplo de aplicação direta de meta-aprendizagem em otimização com o uso de meta-heurísticas pode ser visto em (KANDA, 2012) e (KANDA et al., 2010). Nestes trabalhos, é realizado um estudo em uma base própria de problemas de caixeiro viajante com 4 meta-heurísticas.

Além disso, são propostas diversas meta-características específicas para problemas do caixeiro viajante e das meta-heurísticas utilizadas. É possível perceber que o resultado obtido é expressivamente melhor do que a utilização unicamente da melhor meta-heurística.

Além deste caso, há diversos casos na literatura de estudos da comunidade de meta-heurísticas na performance destas em problemas de otimização. Apesar de muitas vezes não ser utilizado o termo de meta-aprendizagem ou *meta-learning*, os conceitos estudados se qualificam dentro do tema (SMITH-MILES, 2009).

Nestes trabalhos, é realizada a descrição da instância por meio da análise de perfil da superfície de busca de uma instância específica e a extração de características que permitam realizar a seleção de algoritmo para o problema. Apesar destes esforços iniciais, não tem havido trabalhos para generalizar as ideias utilizadas na comunidade de meta-aprendizagem para aprendizagem de máquina.

Um outro ramo em que há um avanço similar à meta-aprendizagem são as hiper heurísticas, propostas por Burke (BURKE et al., 2003). Porém, nestes casos, é pretendido uma criação de uma combinação de diversas heurísticas mais simples, chamadas de heurísticas de baixo nível, de forma a tentar compor uma meta-heurística para cada problema, por meio da decisão de aplicação linear de um conjunto destes operadores ou da decisão dinâmica da aplicação dos mesmos (PAPPA et al., 2014).

3.3 Meta-Aprendizagem para MaxSAT

Apesar de dentro do ramo de otimização termos ainda uma utilização restrita de técnicas de meta-aprendizagem, especificamente dentro dos problemas SAT e MaxSAT há uma utilização maior destas técnicas (SMITH-MILES, 2009).

Considerando o problema MaxSAT, há uma competição internacional, chamada *MaxSAT evaluation*, que realiza os testes de diversos algoritmos para resolução de problemas MaxSAT. Dentro dos *solvers* utilizados na competição, existem os que utilizam uma abordagem tradicional, por meio da resolução de cláusulas conflitantes, propagação unitária, *branch-and-bound*, etc. Há também uma categoria, chamada de *portfolio solvers*, que se utilizam de meta-aprendizagem para seleção do melhor algoritmo para cada instância.

Um dos principais concorrentes neste quesito é o SATzilla (XU et al., 2007) (XU et al., 2008), sendo inclusive vencedor em competições anteriores. Neste trabalho, foi desenvolvido não somente um seletor de algoritmo com base em portfólio mas também um seletor de algoritmos de pré-processamento. Uma característica deste trabalho é que não foram utilizadas técnicas para classificação da instância e seleção de melhor algoritmo, mas foi proposto um modelo de regressão em que dadas as meta-características e um algoritmo, o SATzilla previa quanto tempo o algoritmo demoraria na resolução. Desta

forma, o SATzilla previa o tempo de execução de cada algoritmo candidato do portfólio, e após prever todos tomava a decisão, ao invés de observar as características da instância e tentar inferir diretamente qual o melhor algoritmo para resolução. Esta estratégia foi utilizada pelo SATzilla-07 (XU et al., 2007) e SATzilla-09 (XU et al., 2008).

Versões mais recentes do SATzilla, bem como outros *portfolio solvers* como o 3S e o CSHC utilizam-se de técnicas de classificação apenas, por apresentar performance superior conforme foi demonstrado em (XU et al., 2008). Os SATzillas mais modernos utilizam-se de técnicas de meta-aprendizagem de árvores com decisões binárias entre cada par de algoritmos, realizando uma votação da floresta para decisão do algoritmo. O CSHC (*Cost-sensitive hierarchical clustering*) utiliza-se de técnicas de clusterização para conseguir lidar com grande quantidade de instâncias / características e com uma grande quantidade de *solvers* base (uma dificuldade do SATzilla, uma vez que cria-se uma floresta com uma árvore para cada par de algoritmos) (MALITSKY et al., 2013).

Por fim, há algoritmos mais recentes como o *Instance Specific Algorithm Configuration* em que não é realizada somente a seleção do algoritmo, mas também uma tentativa de otimização dos parâmetros de configuração do algoritmo exato, por meio da clusterização das instâncias e da execução de um grande número de métodos exatos para estas instâncias (ANSÓTEGUI et al., 2016).

Na literatura, a utilização mais comum de meta aprendizagem para classificação em otimização é referente à seleção de algoritmos exatos. Esta tendência fica ainda mais evidenciada na utilização do MaxSAT, onde é perceptível que praticamente não há trabalhos relacionados à caracterização do problema para meta heurísticas, conforme pode ser visto nas Seções 3.2 e 3.3.

Além disto, meta heurísticas possuem uma grande quantidade de parâmetros configuráveis, geralmente podendo operar de forma bem diferente no que diz respeito a diversificação e intensificação, e também a como a interação dos diversos mecanismos, operadores e parâmetros de cada meta heurística alteram bastante o princípio de funcionamento (TALBI, 2009).

3.4 Meta-características da literatura

Um estudo bastante aprofundado de meta-características para o problema SAT pode ser visto em (NUDELMAN et al., 2004). Neste artigo, é realizada uma proposta extensa de meta-características para determinar a dificuldade do problema, com um conjunto total de 91 meta-características. Dentro das características, há desde características com base na lógica booleana, características das cláusulas, dos grafos utilizados para resolução, características de resposta às buscas locais gulosas e diversas outras.

Dentre estas características, em (NUDELMAN et al., 2004), há uma priorização e uma determinação de quais são mais importantes na determinação da dificuldade de resolução do SAT decisório.

Apesar da proposta inicial considerar apenas a dificuldade do SAT decisório, um subconjunto destas meta-características foram utilizadas em todos os trabalhos de MaxSAT.

Inicialmente, para o SATzilla-07, foram selecionadas apenas meta-características que não possuíam uma grande complexidade de custo computacional para cálculo. Este sub conjunto de 48 meta-características já apresentou uma redução significativa de meta-características (XU et al., 2008).

Porém, classificadores mais avançados têm utilizado uma quantidade menor de meta-características, como o ISAC+. Este utiliza um conjunto ainda menor de meta-características, com apenas 32 mais significativas. Empiricamente, verificou-se que esta quantidade de meta-características não prejudicava o desempenho do algoritmo. As características utilizadas foram medidas como a quantidade de variáveis, de cláusulas, proporções entre variáveis positivas e negativas e número de cláusulas em que uma variável aparece (ANSÓTEGUI et al., 2016).

3.5 Considerações Finais

Neste capítulo, apresentou-se o problema de seleção de algoritmos, bem como conceitos de meta-aprendizagem e aplicações específicas. Este entendimento servirá como base para compreensão da proposta, bem como percepção dos pontos propostos quando comparados a outras utilizações de meta aprendizagem na literatura.

4 Meta-Aprendizagem aplicada ao Problema MaxSAT

A proposta do *framework* de meta-aprendizagem compreende aspectos relativos à extração de meta-características, seleção e avaliação de meta-heurísticas para resolução de instâncias problemas que possam ser representados em grafos, sendo potencialmente aplicável a uma gama abrangente de problemas de otimização.

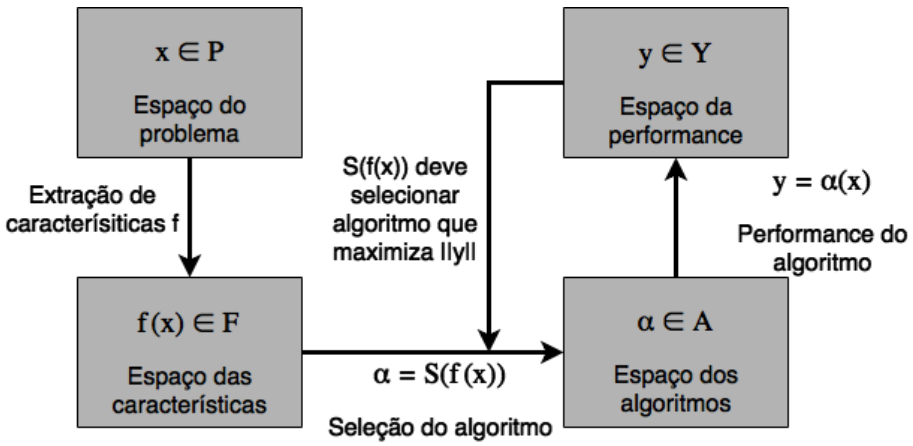
O teste do *framework* foi realizado na resolução de instâncias do problema MaxSAT. Para representação do problema em grafo, além das representações já existentes na literatura, foram propostas novas formas de representação do problema baseada em grafos.

No restante deste capítulo, é apresentada a extensão do problema de seleção de algoritmo dentro do *framework* proposto. Em seguida, é descrito o espaço de meta características, bem com as novas propostas para representação do MaxSAT por meio de grafos e é apresentado um detalhamento maior sobre a geração do espaço de algoritmos. Por fim, o mecanismo de aprendizagem é comentado com maior foco.

4.1 *Framework* genérico

Para o problema original de seleção de algoritmos, pode-se ver uma adaptação da proposta na Figura 18 (RICE, 1976).

Figura 18 – Problema de seleção de algoritmos



Fonte: Adaptado de (RICE, 1976)

Este é um conceito bastante genérico do problema e se aplica a qualquer tarefa de seleção de algoritmos, desde seleção de algoritmos para classificação, para otimização, para processamento de uma imagem entre outros domínios.

Além disso, o problema de seleção de algoritmo não necessariamente se utiliza de técnicas de aprendizagem de máquina para determinação da função de seleção de algoritmo, podendo ser utilizada qualquer forma de função. Por exemplo, em (RICE, 1976) é utilizado em um dos problemas a regressão pelo método de mínimos quadrados.

As aplicações de meta-aprendizagem com o viés de seleção de algoritmos, apresentadas no Capítulo 3, são sempre específicas para os problemas em questão, raramente abordadas de forma mais generalista.

A proposta deste trabalho é a formulação de um novo *framework* a partir da extensão do esquema apresentado na Figura 18. O *framework* proposto é extensível a vários problemas de otimização, incorporando algoritmos de aprendizagem de máquina para geração de modelos de decisão baseados em experimentos conhecidos *a priori*.

Nesse sentido, são propostas as seguintes alterações visando a um *framework* generalizado:

1. **Espaço de meta-características**: propõe-se a utilização de meta-características genéricas obtidas a partir de grafos. Desta forma, busca-se uma abrangência maior para aplicação da técnica proposta, dada a utilização ubíqua de grafos para representação de problemas de interesse acadêmico e industrial. Portanto, os seguintes desdobramentos são requeridos:

   - Definição de uma função $G(x)$, que realize a representação do problema por meio de grafos. Esta função deve permitir que até mesmo problemas que não são normalmente representados por grafos possam ser utilizados na abordagem. Para problemas como o Caixeiro Viajante, a representação via grafos é trivial, no entanto para problemas como MaxSAT é necessária a utilização de técnicas específicas separadamente para representação do problema. Estas técnicas de representação do MaxSAT via grafo foram vistas com detalhes na Seção 2.1.3
   - Extração de meta-características específicas de grafo. Desta forma, se torna possível a generalização da abordagem deste ponto em diante, pois as características extraídas dos grafos independem do problema que está sendo tratado.

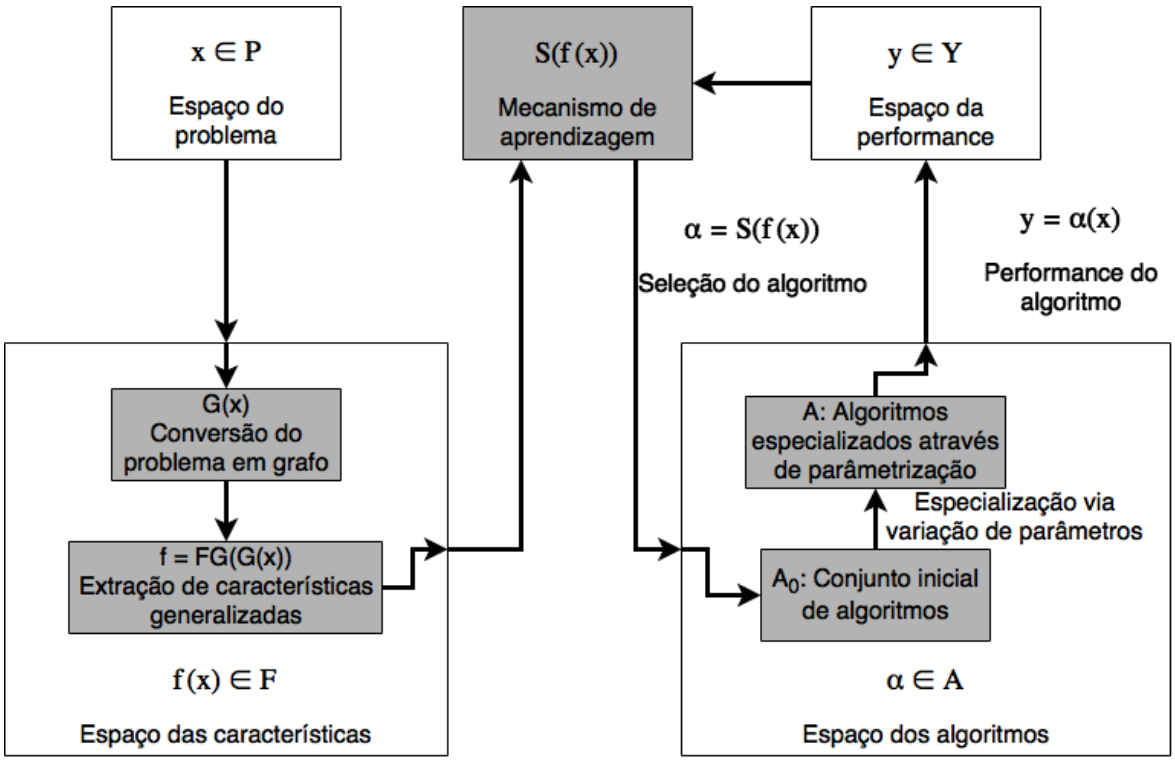
2. **Espaço de algoritmos**: propõe-se a extensão da mera seleção de algoritmos para um espaço amplo de algoritmos e diferentes configurações de parâmetros. Para tanto, os algoritmos que estiverem dentro do conjunto inicial de algoritmos A_0 devem ser instanciados com parâmetros específicos, permitindo a especialização antes da seleção e aplicação.

3. **Mecanismo de aprendizagem** propõe-se abordagem baseada em aprendizagem de máquina. Com isto, conforme a definição vista no Capítulo 3, na Seção 2.3, o sistema deve ser capaz de aprender com a experiência. Com isto, espera-se que o

mecanismo de aprendizagem seja capaz de futuramente selecionar os algoritmos corretos para diversos problemas e instâncias de problemas de otimização diferentes.

Estas alterações propostas podem ser vistas no diagrama estendido representado na Figura 19, onde as alterações no fluxo original estão em cinza.

Figura 19 – *Framework* proposto



Fonte: Acervo do autor

A partir destas alterações, espera-se que a abordagem seja extensível a diversos problemas de otimização e que a utilização de técnicas de aprendizagem de máquina seja capaz de selecionar eficientemente o algoritmo especializado para cada instância apresentada.

4.2 Espaço de meta-características

Não há atualmente na literatura uma representação que sirva exclusivamente para a geração de meta-características do problema MaxSAT. Todavia, verifica-se que a literatura já contempla propostas de extração de meta-características do problema utilizando grafos, como visto na Seção 2.1.3. Entretanto, as representações existentes, de certa forma, perdem alguma informação devido ao seu uso na resolução do problema, e não como forma de representar o conhecimento sobre a instância na sua totalidade.

A extração de conhecimento sobre a estrutura do problema é um subproduto da representação, e as características extraídas dos grafos não eram consideradas importantes para categorização do problema (NUDELMAN et al., 2004), frente a outras características presentes.

4.2.1 Proposta de representação baseada em grafos

Para este trabalho, é proposta uma expansão dos grafos apresentados na Seção 2.1.3, de forma a agregar informação sobre a estrutura do problema. Além disso, conforme descrito na Seção 4.1, a representação intermediária por grafos possibilita a expansão da aplicabilidade futura.

Para os grafos VG (Figura 3), as arestas deixam de ter pesos unitários e inclui-se informação referente à quantidade de cláusulas que as variáveis compartilham, sendo uma forma de sinalizar a força do relacionamento existente entre as variáveis. Este grafo foi nomeado Grafo de Variáveis com Pesos (*Weighted Variable Graph - WVG*).

Como exemplo, o conjunto de cláusulas $c_1 = x_1 \vee \neg x_2 \vee x_3$, $c_2 = x_2 \vee x_4$ e $c_3 = \neg x_1 \vee \neg x_2$ que foi utilizado para gerar o grafo da Figura 20a gera, com esta proposta, o grafo que ser visto na Figura 20b. Neste caso, é visível o peso 2 na aresta entre x_1 e x_2, indicando um relacionamento entre x_1 e x_2 diferente do existente entre x_1 e x_3.

Figura 20 – VG comparado a um WVG para um problema MaxSAT

(a) VG (b) WVG

Fonte: Acervo do autor

Para os grafos CG (Figura 4), foram inclusas arestas em nós representando variáveis compartilhadas positivas, de forma a significar o relacionamento entre as cláusulas e não somente as relações entre as restrições de variáveis negadas. Além disso, as arestas também deixam de ser unitárias - a aresta passa a ter o peso igual à quantidade de variáveis compartilhadas entre as cláusulas. Este grafo foi nomeado Grafo de Cláusulas com Pesos (*Weighted Clause Graph - WCG*).

A instância $c_1 = x_1 \vee \neg x_2 \vee x_3$, $c_2 = \neg x_2 \vee \neg x_3$ e $c_3 = x_1 \vee \neg x_3$, representada pelo grafo da Figura 21a, gera o grafo da Figura 21b. Neste caso, pode-se perceber a inclusão

do relacionamento entre c_1 e c_3 que compartilham duas variáveis que estão, no entanto negadas, em uma cláusula e não negadas na outra. Além desse relacionamento, percebe-se também o peso das arestas (c_1,c_2) bem como (c_1,c_3), indicando que estas cláusulas possuem uma relação diferente do par c_2, c_3, por exemplo.

Figura 21 – CG comparado a um WCG para um problema MaxSAT

(a) CG (b) WCG

Fonte: Acervo do autor

4.2.2 Caracterização das instâncias

Para a geração de meta-características, é necessária a extração de medidas a partir da formulação do problema. Um levantamento de características utilizadas com frequência na literatura, bem como uma análise da importância de cada uma pode ser vista em (NUDELMAN et al., 2004). No entanto, os estudos realizados são apenas para o problema SAT restrito a 3 variáveis por cláusula (3-SAT) e não para o MaxSAT.

Há artigos que se utilizam destas mesmas características para o problema SAT e MaxSAT, com sucesso. Desta forma, utilizam-se estes subconjuntos de características como estado da arte para comparação neste trabalho (NUDELMAN et al., 2004) (XU et al., 2008) (MALITSKY et al., 2013) (ANSÓTEGUI et al., 2016).

4.2.2.1 Características extraídas do problema booleano

Para o problema booleano, as características extraídas foram retiradas da lista de características consideradas mais importantes (NUDELMAN et al., 2004).

As características extraídas foram:

- Número de cláusulas - n_c;
- Número de variáveis - n_v;
- Razão de cláusulas por variáveis - $\frac{n_c}{n_v}$;
- Inverso da razão de cláusulas por variáveis - $\frac{n_v}{n_c}$;

- Razão de cláusulas por variáveis normalizada: $|4.27 - \frac{n_c}{n_v}|$. A relação de 4.27 cláusulas por variável é um valor que representa a dificuldade máxima de resolução do problema, sendo sua utilização explorada em (NUDELMAN et al., 2004);
- Relação entre ocorrências positivas e negativas de cada variável - calculada para cada variável e extraídas as seguintes medidas estatísticas como característica global da instância: média, desvio padrão, coeficiente de variação, mínimo, máximo, entropia de Shannon;
- Relação entre variáveis positivas e negativas em cada cláusula - calculada em cada cláusula e extraídas as seguintes medidas estatísticas como característica global da instância: média, desvio padrão, coeficiente de variação, mínimo, máximo, entropia de Shannon;
- Percentual de cláusulas com apenas uma variável (cláusulas unárias);
- Percentual de cláusulas com exatamente duas variáveis (cláusulas binárias); e
- Percentual de cláusulas com exatamente três variáveis;

4.2.2.2 Características propostas para extração de grafos

Para a extração de características de grafos, foram consideradas algumas medidas padrões de grafos. Segue abaixo o descritivo das medidas consideradas. É importante salientar que nenhuma destas medidas considera a estrutura específica do MaxSAT, sendo todas as meta características generalizáveis a qualquer grafo, com arestas de peso unitário ou não, com arestas direcionadas ou não.

- Número de nós;
- Número de vértices;
- Densidade do grafo;
- Grau de nó: calculada para cada nó e extraídas as seguintes características como características globais da instância: média, desvio padrão, coeficiente de variação, mínimo, máximo, entropia de Shannon; e
- *Weighted Clustering Coefficient* definido como o número de arestas de todos os nós vizinhos, dividido por $k(k+1)/2$, com k sendo o número de vizinhos (NUDELMAN et al., 2004). Calculado para cada nó e retiradas as seguintes medidas como características globais do grafo: média, desvio padrão, coeficiente de variação, mínimo, máximo, entropia de Shannon.

Além disso, no caso de grafos com pesos não-unitários propostos, adiciona-se a seguinte medida:

- Soma dos pesos das arestas do nó: calculada para cada nó e extraídas as seguintes características como características globais da instância: média, desvio padrão, coeficiente de variação, mínimo, máximo, entropia de Shannon

4.3 Espaço de algoritmos

Espaço de algoritmos passa a contemplar tuplas de algoritmos e seus parâmetros de configuração (tais como população, taxas de aprendizagem e operadores), tendo sido nomeado, portanto, como espaço estendido ou espaço de algoritmos-parâmetros. O conjunto de valores elegíveis é específico para cada parâmetro e pode se referir a expertise ou literatura recente.

Diferentemente de outras propostas voltadas para problemas MaxSAT, neste trabalho propõe-se meta-heurísticas como os algoritmos de otimização a serem selecionados durante a meta-aprendizagem devido ao seu desempenho previamente apurado, conforme visto na Seção 2.2.

Cada uma destas tuplas de algoritmos e parâmetros é considerada uma classe para o problema de aprendizagem. Desta forma, a geração de exemplos se dá por meio da execução de algoritmos diversos em uma ampla gama de instâncias.

4.4 Mecanismo de aprendizagem

A seleção de algoritmos de otimização baseada em aprendizagem de máquina implica na capacidade de ganho de desempenho em uma dada tarefa por meio da aquisição de experiência prévia (MITCHELL, 1997).

Para realização da seleção de algoritmo, para cada instância do problema é possível determinar que há uma ou mais tuplas de meta-heurísticas e parâmetros que resolvem da melhor forma este problema. Portanto, o problema se caracteriza como um problema de aprendizagem de máquina como aprendizagem supervisionada, onde para cada exemplo é possível obter a resposta correta. Além disso, o resultado esperado da aprendizagem para cada instância é discreto, sendo portanto o problema caracterizado como classificação.

A caracterização das instâncias do problema se dá através da extração de características apresentadas na Seção 4.2.2.2. Para determinação da classe deste exemplo, se faz necessária a execução de todos os algoritmos do espaço de algoritmos, de forma a ser possível identificar qual a tupla de algoritmo e parametrização que melhor resolve aquele problema, delimitando a classe da instância. Desta forma, ao se selecionar problemas

diversos, é possível montar um conjunto de treinamento capaz de representar adequadamente a gama de possíveis instâncias do problema representado pelo grafo, neste caso especificamente de problemas MaxSAT.

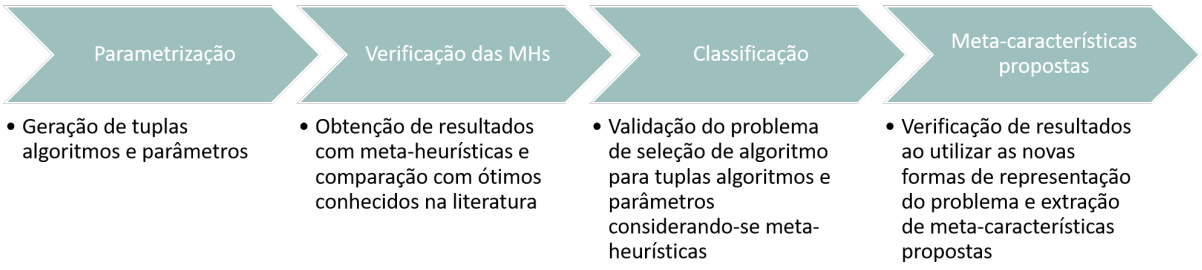
4.5 Considerações Finais

Neste capítulo foi exposto a proposta desenvolvida. Esta proposta é uma extensão ao problema de seleção de algoritmos, considerando-se a utilização de um *framework* genérico, a utilização de meta-características independentes do problema, a extensão do espaço de algoritmos e a utilização de mecanismos de aprendizagem de máquina para realizar a seleção.

5 Experimentos Computacionais

Neste Capítulo, o contexto experimental é detalhado, incluindo os operadores, representação e parâmetros de desempenho usados nas meta-heurísticas, bem como os parâmetros de desempenho dos classificadores. Faz-se uma verificação inicial se o conjunto de instâncias do MaxSAT pode ser considerado um problema de seleção de algoritmo. Em seguida, são apresentados resultados que permitem avaliar as diferentes formas de representar o problema em termos de meta-características. Esta proposta pode ser vista melhor em fases sequenciais na Figura 22.

Figura 22 – Sequência de Experimentos



Fonte: Acervo do autor

Os experimentos foram realizados em uma estação de trabalho, executando *Linux Mint*, versão 17.1, e *framework ParadisEO*, versão 2.0.1 (CAHON; MELAB; TALBI, 2004), compilado com G++ (C++11), versão 4.8.4, para implementação de meta-heurísticas. Para a implementação de classificadores, foi usada a biblioteca WEKA,versão 3.8.1 (HALL et al., 2009).

5.1 Meta-Heurísticas: codificação, parâmetros e operadores

Para cada meta-heurística, foram propostas 4 combinações de parâmetros diferentes, na seguinte configuração:

- Duas parametrizações com resultados bons obtidas após experimentação empírica;
- Uma parametrização com foco grande em intensificação; e
- Uma parametrização da meta-heurística com foco em diversificação.

Durante os experimentos, como propósito de obtenção de um conjunto de algoritmos e seus parâmetros bem adaptados a instâncias específicas, optou-se por limitar o

tempo de execução de cada meta-heurística. Sendo assim, como critério de parada das meta-heurísticas, considerou-se a utilização de 1.000 chamadas da função objetivo.

5.1.1 Algoritmos Genéticos

Dado que o problema do MaxSAT é um problema de atribuição de variáveis binárias, a escolha mais direta de um indivíduo é um vetor binário, onde cada posição ou gene pode assumir o valor 0 ou 1. O indivíduo deve possuir tantos genes quanto variáveis no problema. Desta forma, cada gene será mapeado diretamente para uma variável.

Para os algoritmos genéticos, foram considerados parâmetros com maior e menor populações, diferentes operadores de seleção com diferentes pressões seletivas (quando possível), e operadores de cruzamento e mutação com diferentes probabilidades e taxas de mutação. Os algoritmos genéticos utilizados podem ser vistos na Tabela 1.

As duas primeiras parametrizações foram escolhidas empiricamente após alguns poucos testes. A terceira parametrização tem um foco forte em diversificação, com uma população grande, uma taxa de mutação maior e uma quantidade de genes alteradas maior. Além disso, possui uma pressão seletiva menor pela grande chance de seleção de um indivíduo aleatório no torneio estocástico.

A quarta parametrização possui o foco em intensificação. Ela possui uma população um pouco mais restrita, com pouca chace de mutação e com um torneio com pressão seletiva muito grande, buscando atingir de maneira mais rápida ótimos locais.

Tabela 1 – Parâmetros dos algoritmos genéticos

Pop.	**Cruzamento**	**Taxa**	**Mutação**	**Taxa**	**Alteração**	**Seleção**
50	Um pt.	60%	*Bitflip* det.	5%	1%	Torneio - 2 ind.
70	Dois pts.	80%	*Bitflip* original	90%	1%	Torn. Estoc. 70%
500	Um pt.	60%	*Bitflip* det.	5%	10%	Torn. Estoc. 60%
70	Um pt.	80%	*Bitflip* original	75%	1%	Torneio - 5 ind.

5.1.2 Enxame de Partículas

Uma partícula possui algumas diferenças significativas para um indivíduo no algoritmo genético proposto na Seção 5.1.1. Uma partícula possui uma posição no hiperespaço, que deve possuir tantas dimensões quanto variáveis no problema e todas elas booleanas. Desta forma, a posição atual de uma dada partícula é uma atribuição das variáveis do problema. No entanto, uma partícula possui um componente adicional de velocidade, representado por um vetor de números reais com n_{var} dimensões, sendo n_{var} o número de variáveis na instância atual. Para evitar velocidades muito discrepantes e passos muito largos no espaço de busca da meta-heurística, a velocidade em cada dimensão ficou limitada entre $[-1,5; 1,5]$.

A escolha dos parâmetros para o PSO se deu com a utilização de conjuntos de parâmetros que possibilitassem o teste de diversos tamanhos de população, topologias de vizinhança, tamanho de vizinhança e parâmetros de voo (inércia, atração do ótimo conhecido - L1 - e atração do ótimo de vizinhança - L2). Os parâmetros testados podem ser vistos na Tabela 2. Assim como nos algoritmos genéticos, as duas primeiras parametrizações foram determinadas empiricamente.

A terceira parametrização é a focada em exploração ou diversificação, com um peso menor para os ótimos já conhecidos e uma inércia grande da partícula. Além disso, uma vizinhança em anel ajuda a reduzir a visibilidade de ótimos globais mais distantes, e uma população maior auxilia no processo de exploração do espaço de busca.

A quarta parametrização é focada em intensificação. Nela, temos pesos consideráveis de atração para os ótimos conhecidos, há uma topologia de vizinhança que permite que todos vejam o ótimo global, e há uma influência menor da inércia da própria partícula, favorecendo o foco em locais de maior concentração de ótimos conhecidos, intensificando a busca nestes locais.

Tabela 2 – Parâmetros testados no PSO

Pop	**Vizinhança**	**Tamanho**	**Inércia**	**L1**	**L2**
20	Linear	3	1,0	1,7	2,3
50	Linear	5	1,0	1,3	1,9
250	Ring	2	1,0	1,0	1,0
50	Estrela	-	0,8	2,0	2,5

5.1.3 PBIL

No PBIL, os indivíduos são codificados assim como no algoritmo genético representado na Seção 5.1.1. No entanto, o mecanismo de memória não é a população em si, mas um vetor com n_{var} posições de números contidos no intervalo $[0; 1]$, inicializado com todos os valores em $0,5$. A partir desse modelo probabilístico, a população deve ser gerada.

A escolha dos parâmetros para o PBIL foi realizada para avaliar diversos tamanhos de população, quantidade de indivíduos para aprendizagem e tolerâncias. Os parâmetros testados podem ser vistos na Tabela 3. Assim como nas demais meta-heurísticas, as duas primeiras parametrizações foram determinadas empiricamente de forma aleatória.

A terceira parametrização é a focada em exploração ou diversificação, com uma tolerância maior de forma que não se deixará de explorar nenhum local do espaço de busca. Além disso, é utilizada uma população maior por geração, e a aprendizagem se dá em diversos indivíduos.

A quarta parametrização é focada em intensificação. Nesta configuração, temos uma baixa tolerância, e a aprendizagem se dá com uma alta taxa e de apenas o indivíduo de

melhor performance. Com isso, a tendência do algoritmo é de se especializar e intensificar a busca apenas na região mais promissora encontrada, sem tentar encontrar novas regiões promissoras.

Tabela 3 – Parâmetros testados para o PBIL

População	n_b	η_b	n_w	η_w	**Tolerância**
50	1	0,1	0	0,01	0,01
50	1	0,1	0	0,01	0,05
200	5	0,05	5	0,10	0,3
30	1	0,1	0	0,01	0,01

5.2 Resolução do problema

Para determinar a efetividade das meta-heurísticas para o problema, se faz necessário comparar os resultados obtidos por estas com as melhores soluções conhecidas. Para problemas em que não são conhecidos os valores ótimos, considera-se como tal, para efeito de execução bem sucedida, a melhor solução obtida por qualquer meta-heurística durante os experimentos. A determinação da efetividade das meta-heurísticas se deu através da comparação do resultado médio obtido pelo conjunto meta-heurísticas e parâmetros em todas as execuções para as quais o ótimo é conhecido.

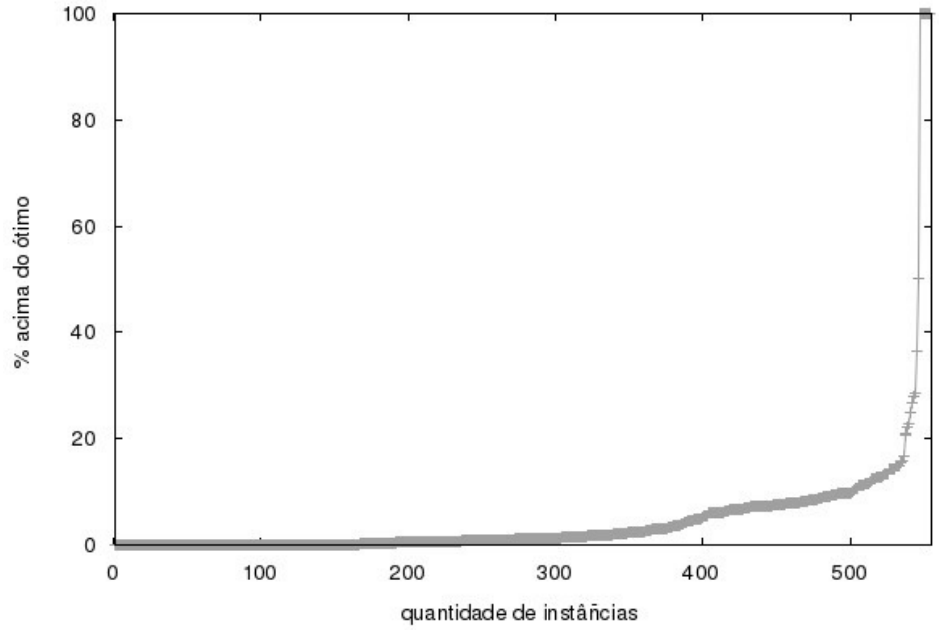
As instâncias do problema MaxSAT usadas possuem, em média 172 variáveis e 1.395 cláusulas, utilizadas como *benchmarks* em uma competição internacional. Apesar do tamanho do espaço de busca, as meta-heurísticas ficaram a apenas 4,81% da melhor solução conhecida, sendo esta ou a solução ótima, quando conhecida, ou a melhor solução obtida pelo conjunto de meta-heurísticas. As meta-heurísticas conseguiram atingir a melhor solução conhecida em 30,09% das instâncias.

Dos 555 problemas disponíveis, tem-se o seguinte desdobramento:

- Em 11 instâncias, todas as meta-heurísticas testadas atingiram o ótimo global; estes problemas foram removidos da base de teste;
- em 91 instâncias, não havia solução ótima conhecida, sendo considerado como ótimo global aquele conhecido a partir dos experimentos realizados;
- em 76 instâncias, houve apenas uma meta-heurística capaz de atingir o ótimo global disponível na literatura;
- no restante das 377 instâncias, a melhor meta-heurística obteve desempenho pior do que disponível na literatura.

Não houve o caso de mais de uma meta-heurística vencedora para uma dada instância, o que levaria a um caso de classificação *multi-label* (KANDA et al., 2010). Os testes foram realizados com a técnica *k-fold*, utilizando-se de 10 *folds* para teste de efetividade. Uma comparação de desempenho das melhores meta-heurísticas pode ser vista na Figura 23.

Figura 23 – Desempenho das meta-heurísticas



Fonte: Acervo do autor

É possível perceber que houve uma espécie de polarização dos problemas, com duas meta-heurísticas com uma performance bem superior às demais.

Tabela 4 – Meta-heurísticas e a quantidade de problemas em que foram as melhores

Meta-heurística	**# problems**
GA 50 pop, cruzamento 1-pt(60%), det. flip (5% taxa, 1% bits), 2-tourn.	344
GA 70 pop, cruzamento 1-pt(80%), *bitflip* orig (75% taxa, 1% bits), 5-tourn	177
PBIL 30 pop, $n_b = 1$, $\eta_b = 0,1$, $n_w = 0$, $tolerance = 0,01$	19
PBIL 50 pop, $n_b = 1$, $\eta_b = 0,1$, $n_w = 0$, $tolerance = 0,01$	4

5.3 Avaliação da proposta de meta-aprendizagem

Para gerar a base de dados da aprendizagem, foram executadas todas as meta-heurísticas em todos as instâncias disponíveis da base, com uma quantidade de iterações que possibilitasse obter significância estatística, quando fosse o caso. Instâncias para as quais não foi possível obter resultados estatisticamente relevantes foram descartadas. De

posse da solução obtida por cada meta-heurística para cada instância, foi atribuída uma classe a cada instância do problema. A classe de cada instância foi definida como sendo o melhor conjunto de meta-heurística e parâmetros para resolução.

5.3.1 Seleção de meta-heurísticas

Para que o problema de seleção de algoritmos a partir do conjunto de instâncias MaxSAT possa ser considerado um problema de classificação é necessário que seja possível a construção de um modelo de decisão para esse fim. Para verificar esta hipótese, foi considerado o conjunto de meta-características encontrado na literatura, descrito na Seção 4.2.2.1.

Como este conjunto já foi usado com sucesso na literatura para classificação do problema para algoritmos exatos, é um bom candidato para realizar a validação de que o MaxSAT pode ser classificado também para a seleção de meta-heurísticas como métodos de resolução.

Como base para a comparação, utilizou-se o classificador *Zero Rule*, que seleciona apenas a moda, ou a meta-heurística mais bem classificada. Para teste dos demais classificadores, foi realizado um teste de *t-student* para médias considerando amostras com variâncias diferentes (MONTGOMERY, 2001). Todos os resultados que obtiveram significância estatística ($p \leq 0,05$) estão marcados em negrito, conforme mostrado na Tabela 5.

Tabela 5 – Desempenho dos classificadores para meta-heurísticas

Dataset	Taxa de Acerto
Zero Rule	63,24%
MLP	**86,89%**
KNN (1)	**80,83%**
SVM	**86,68%**

Conforme pode ser observado, há uma indicação de que o problema é classificável também para meta-heurísticas. Todos os classificadores obtiveram resultados bastante superiores na seleção de meta-heurísticas quando comparados à utilização de apenas uma meta-heurística mais ajustada ao problema. Isso indica que a utilização de meta-aprendizagem para resolução de problemas considerando uma gama de meta-heurísticas configuradas de forma diferente pode ser mais eficiente do que a calibração fina de uma única meta-heurística.

5.3.2 Modelos de extração de meta-características

Foram realizados experimentos específicos para a avaliação dos modelos de extração de meta-características propostos no Capítulo 4. Os resultados foram obtidos a partir de

30 execuções, cada uma utilizando-se da técnica *k-fold*, com $k = 10$. Devido à utilização de meta-características extraídas do problema booleano, estas são consideradas como referência para a comparação.

Os resultados são apresentados na Tabela 6, com emprego do *test-t* para comparar individualmente cada uma das propostas de meta-características extraídas de grafos com as extraídas diretamente do problema booleano (MONTGOMERY, 2001). A hipótese testada no caso foi de que as médias não são equivalentes. Os resultados em negrito são estatisticamente significantes ($p \leq 0,05$), o que indica que a diferença entre a representação da coluna e a representação booleana da Seção 4.2.2.1 é significativa. Dentre as formas de representação propostas, aplicou-se o *framework* tanto nos grafos existentes na literatura descritos na Seção 2.1.3 (VCG, VG e CG) quanto os grafos propostos descritos na Seção 4.2.1 (WVG, WCG).

Tabela 6 – Taxa de acerto dos algoritmos de classificação

Algoritmo de Aprendizagem	Booleanas	VCG	VG	WVG	CG	WCG
Zero Rule	63,24	63,24	63,24	63,24	63,24	63,24
KNN	80,83	**82,13**	**80,34**	**78,72**	**82,05**	**82,21**
SVM	86,68	**85,76**	**85,44**	**85,70**	**85,01**	**86,17**
MLP	86,88	87,03	87,14	**87,18**	86,78	87,09

Como pode ser observado, as representações por grafos não apresentam, de forma geral, evidências de melhoria na taxa de acerto para todos os algoritmos de aprendizagem de máquina, exceto as redes neurais MLP. As representações por grafo foram positivas em 4 casos, neutras em 4 casos e negativas em 7 casos.

É importante ressaltar que o melhor resultado obtido foi usando meta-características extraídas do WVG (*Weighted Variable Graph*), o que demonstra que uma representação específica do problema no domínio de grafos pode melhorar a taxa de acertos quando comparada à métodos usuais da literatura.

5.3.3 Grafos ponderados

Foram realizados experimentos específicos para a avaliação dos grafos ponderados propostos WVG e WCG (*Weighted Variable Graph* e *Weighted Clause Graph*, respectivamente), quando comparados às suas versões originais sem pesos (VG e CG). Para tanto, foi utilizado o mesmo algoritmo de classificação sobre meta-características obtidas a partir dos 4 tipos de grafos.

Para o algoritmo *KNN*, pode-se observar na Tabela 7 que, no caso VG, a adição dos pesos acabou por prejudicar o desempenho do classificador. Para o CG, não houve diferença significativa.

Tabela 7 – Taxa de Acerto - KNN

	Sem Pesos	Com Pesos
VG	80,34	**78,72**
CG	82,05	82,21

Para as *SVM*, a Tabela 8 permite perceber que em ambos os grafos a proposta de adição de pesos foi capaz de aumentar a taxa de acerto. A taxa de acerto dos grafos com pesos foi comparada à dos grafos sem pesos através de um *test-t* (MONTGOMERY, 2001), e comprovou que a melhora da acurácia é estatisticamente significativa.

Tabela 8 – Taxa de Acerto - SVM

	Sem Pesos	Com Pesos
VG	85,44	**85,70**
CG	85,01	**86,17**

Para as *MLP*, pode-se perceber na Tabela 9 que, para o CG, a adição de pesos foi capaz de aumentar a taxa de acerto com significância estatística.

Tabela 9 – Taxa de Acerto - MLP

	Sem Pesos	Com Pesos
VG	87,14	87,18
CG	86,78	**87,09**

6 Conclusões e Trabalhos Futuros

O problema de Máxima Satisfabilidade, ou MaxSAT, é de grande interesse da comunidade científica tanto por ser fundamental para diversos problemas práticos, como por ter sido usando na definição de uma classe de problemas bastante estudada, os problemas NP-Completos (COOK, 1971).

Para obtenção de soluções de qualidade em um curto espaço de tempo, pode-se optar pela utilização de meta-heurísticas de otimização, todavia sujeitando-se a iminente necessidade de decidir qual delas e como ajustar seus parâmetros. Necessidade esta reforçada pelo Teorema *"No Free Lunch"* que declara que não existe algoritmo único que consiga apresentar a melhor solução para todas as instâncias possíveis de problemas, sendo o ganho de desempenho em um determinado subconjunto compensado pela perda de desempenho em outro subconjunto de instâncias.

Destaca-se então o campo de meta-aprendizagem, surgido a partir do problema de seleção do melhor algoritmo para uma única instância do problema, o que vem deslocar os esforços de pesquisa para a escolha automática de algoritmos melhores adaptados a cada instância, ao invés da explorar a melhor configuração de um único algoritmo a ser aplicado a todas instâncias.

Neste trabalho, foi proposto um *framework* de meta-aprendizagem capaz de extrair meta-características de instâncias do problema MaxSAT e selecionar meta-heurísticas de otimização mais aptas a resolvê-las. Para tanto, uma nova forma de representação do problema baseada em grafo foi proposta, acrescentando novas informações na forma de pesos, o que permite a derivação de novas meta-características. A utilização de uma representação intermediária do MaxSAT através de grafos já havia sido usada para resolução do problema e, incidentalmente, tem permitido a extração de meta-características, porém a literatura não aponta a existência de uma representação com o único objetivo de gerar meta-características do problema.

Foram obtidos resultados que comprovam que as meta-heurísticas utilizadas apresentam desempenhos competitivos para as instâncias utilizadas, bem como comprovam que o problema MaxSAT é classificável e, portanto, passível de ser usado para seleção de meta-heurísticas. Por fim, conclui-se também que a nova forma de representação do problema MaxSAT se equipara ou supera o estado da arte atual para classificação do MaxSAT.

Destaca-se ainda, como ponto forte da proposta, a evidente capacidade de generalização, por realizar uma abordagem baseada em grafos sem nenhum tipo de caracterização específica para o problema MaxSAT, excetuando-se a conversão deste para grafo. Desta

forma, é possível vislumbrar a aplicação em outros problemas, desde que representáveis por grafos e que tal representação mantenha-se expressiva o suficiente para capturar informações necessárias a classificação das instâncias.

Como trabalhos futuros, vislumbra-se principalmente três linhas de trabalhos:

- **Extensão de caracterização SAT:** Estudos de como a nova representação do problema pode ser combinada com as atuais de forma a melhorar a classificação considerando apenas o problema SAT; integrar a representação proposta com as existentes, buscando uma técnica conjunta que seja mais assertiva;
- **Utilização em outros problemas de grafos:** Aplicação do *framework* em outros problemas que possam ser representados por meio de grafos; avaliar a abordagem global, que inclui na base de aprendizagem todos os problemas trabalhados, não se limitando à aplicação em um único problema por vez; investigar conversores de representação ou geradores de meta-características que operem como *drives* ou filtros específicos para cada problema, mesmo aqueles que não são usualmente representados por grafos.

Com esta generalização, é possível haver um classificador que atue de forma geral para todos os principais problemas de otimização, reduzindo a necessidade de realização de uma configuração de parâmetros extensiva para diversos problemas. Com a generalização, deverá ser possível apresentar um problema novo ao meta-classificador, e este recomendar o método e os parâmetros mais adequados para a resolução daquela instância em específico.

Referências

ANSÓTEGUI, C.; GABÀS, J.; MALITSKY, Y.; SELLMANN, M. Maxsat by improved instance-specific algorithm configuration. *Artificial Intelligence*, Elsevier, v. 235, p. 26–39, 2016. Citado 5 vezes nas páginas 17, 18, 45, 46 e 51.

ANSóTEGUI, C.; MALITSKY, Y.; SELLMANN, M. Maxsat by improved instance-specific algorithm configuration. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2014. Disponível em: <http://4c.ucc.ie/~ymalitsky/Papers/AAAI-14.pdf>. Citado 3 vezes nas páginas 16, 21 e 23.

BÄCK, T.; HOFFMEISTER, F. Extended selection mechanisms in genetic algorithms. Citeseer, 1991. Citado na página 27.

BAKER, J. E. Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the second international conference on genetic algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987. p. 14–21. Citado na página 27.

BALUJA, S.; CARUANA, R. Removing the genetics from the standard genetic algorithm. In: *Machine Learning: Proceedings of the Twelfth International Conference*. San Francisco (CA): Morgan Kaufmann, 1995. p. 38–46. Citado 2 vezes nas páginas 35 e 36.

BIERE, A.; HEULE, M.; MAAREN, H. van. *Handbook of satisfiability*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009. v. 185. Citado 2 vezes nas páginas 16 e 20.

BLICKLE, T.; THIELE, L. A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 4, n. 4, p. 361–394, dez. 1996. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162/evco.1996.4.4.361>. Citado na página 27.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, set. 2003. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/937503.937505>. Citado 2 vezes nas páginas 23 e 25.

BROWNLEE, J. *Clever Algorithms: Nature-Inspired Programming Recipes*. 1st. ed. .: Lulu.com, 2011. ISBN 1446785068, 9781446785065. Citado na página 25.

BURKE, E.; KENDALL, G.; NEWALL, J.; HART, E.; ROSS, P.; SCHULENBURG, S. Hyper-heuristics: An emerging direction in modern search technology. In: *Handbook of metaheuristics*. .: Springer, 2003. p. 457–474. Citado na página 44.

BYVATOV, E.; FECHNER, U.; SADOWSKI, J.; SCHNEIDER, G. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of chemical information and computer sciences*, ACS Publications, v. 43, n. 6, p. 1882–1889, 2003. Citado na página 41.

CAHON, S.; MELAB, N.; TALBI, E.-G. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, Springer, v. 10, n. 3, p. 357–380, 2004. Citado 2 vezes nas páginas 18 e 55.

COOK, S. A. The complexity of theorem-proving procedures. In: ACM. *Proceedings of the third annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1971. p. 151–158. Citado 2 vezes nas páginas 16 e 63.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado 2 vezes nas páginas 41 e 42.

DATT, G. An evolutionary approach: analysis of artificial neural networks. *Int. J. Emerg. Technol. Adv. Eng*, Citeseer, v. 2, n. 1, 2012. Citado na página 38.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. *IEEE computational intelligence magazine*, IEEE, v. 1, n. 4, p. 28–39, 2006. Citado na página 24.

EBERHART, R. C.; SHI, Y. Comparison between genetic algorithms and particle swarm optimization. In: *Proceedings of the 7th International Conference on Evolutionary Programming VII*. London, UK, UK: Springer-Verlag, 1998. (EP '98), p. 611–616. ISBN 3-540-64891-7. Disponível em: <http://dl.acm.org/citation.cfm?id=647902.739129>. Citado na página 31.

GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, v. 1, p. 69–93, 1991. Citado na página 27.

GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning*, Springer, v. 3, n. 2, p. 95–99, 1988. Citado 3 vezes nas páginas 24, 28 e 29.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009. Citado 2 vezes nas páginas 18 e 55.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501. Citado na página 37.

HEARST, M. A.; DUMAIS, S. T.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. *IEEE Intelligent Systems and their Applications*, IEEE, v. 13, n. 4, p. 18–28, 1998. Citado na página 40.

HOLLAND, J. H. *Adaptation in natural and artificial systems*. Ann Arbor: MIT Press, 1975. 211 p. Citado na página 25.

IMPAGLIAZZO, R.; PATURI, R. On the complexity of k-sat. *Journal of Computer and System Sciences*, ., v. 62, n. 2, p. 367 – 375, 2001. ISSN 0022-0000. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022000000917276>. Citado na página 20.

JIANG, Y.; KAUTZ, H.; SELMAN, B. Solving problems with hard and soft constraints using a stochastic algorithm for max-sat. In: *1st International Joint Workshop on Artificial Intelligence and Operations Research*. Washington, DC: ., 1995. Citado 2 vezes nas páginas 20 e 21.

JONG, K. A. D.; SPEARS, W. M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, v. 5, n. 1, p. 1–26, Mar 1992. ISSN 1573-7470. Disponível em: <https://doi.org/10.1007/BF01530777>. Citado na página 30.

KANDA, J.; CARVALHO, A.; HRUSCHKA, E.; SOARES, C. Using Meta-learning to Classify Traveling Salesman Problems. IEEE, p. 73–78, out. 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5715216>. Citado 3 vezes nas páginas 16, 43 e 59.

KANDA, J. Y. *Uso de meta-aprendizado na recomendação de meta-heurísticas para o problema do caixeiro viajante*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado 2 vezes nas páginas 16 e 43.

KENNEDY, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: IEEE. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Washington, DC, 1999. v. 3. Citado na página 32.

KENNEDY, J. Particle swarm optimization. In: *Encyclopedia of Machine Learning*. .: Springer, 2010. p. 760–766. Citado na página 31.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In: IEEE. *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*. Orlando, FL, 1997. v. 5, p. 4104–4108. Citado 2 vezes nas páginas 24 e 34.

KOTSIANTIS, S. B. *Supervised Machine Learning: A Review of Classification Techniques*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007. 3–24 p. ISBN 978-1-58603-780-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1566770.1566773>. Citado 2 vezes nas páginas 17 e 36.

LECUN, Y.; BOTTOU, L.; ORR, G. B.; MüLLER, K.-R. Effiicient backprop. In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. London, UK, UK: Springer-Verlag, 1998. p. 9–50. ISBN 3-540-65311-2. Disponível em: <http://dl.acm.org/citation.cfm?id=645754.668382>. Citado na página 38.

LINDEN, R. *Algoritmos Genéticos, uma importante ferramenta de Inteligência Computacional*. Rio de Janeiro, RJ: Brasport, 2006. Citado na página 24.

MALITSKY, Y.; SABHARWAL, A.; SAMULOWITZ, H.; SELLMANN, M. Algorithm portfolios based on cost-sensitive hierarchical clustering. In: ROSSI, F. (Ed.). *IJCAI*. Beijing, China: IJCAI/AAAI, 2013. Citado 3 vezes nas páginas 17, 45 e 51.

MATOS, P. J.; PLANES, J.; LETOMBE, F.; MARQUES-SILVA, J. A MAX-SAT algorithm portfolio. In: *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*. IOS Press, 2008. p. 911–912. Disponível em: <https://doi.org/10.3233/978-1-58603-891-5-911>. Citado na página 16.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 37.

MEYER, D.; WIEN, F. T. Support vector machines. *The Interface to libsvm in package e1071*, 2014. Citado na página 41.

MITCHELL, T. M. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, v. 45, n. 37, p. 870–877, 1997. Citado 3 vezes nas páginas 17, 36 e 53.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. 5th ed. ed. New Jersey, USA: John Wiley, 2001. ISBN 0471316490,9780471316497. Citado 3 vezes nas páginas 60, 61 e 62.

NUDELMAN, E.; LEYTON-BROWN, K.; HOOS, H. H.; DEVKAR, A.; SHOHAM, Y. Understanding random sat: Beyond the clauses-to-variables ratio. In: *Principles and Practice of Constraint Programming–CP 2004*. .: Springer, 2004. p. 438–452. Citado 7 vezes nas páginas 18, 21, 45, 46, 50, 51 e 52.

OLIVEIRA, A. C. M. *Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — INPE - Instituto Nacional de Pesquisas Espaciais., 2004. Citado 2 vezes nas páginas 23 e 24.

PAPPA, G. L.; OCHOA, G.; HYDE, M. R.; FREITAS, A. A.; WOODWARD, J.; SWAN, J. Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, Springer, v. 15, n. 1, p. 3–35, 2014. Citado na página 44.

PATURI, R.; PUDLÁK, P.; SAKS, M. E.; ZANE, F. An improved exponential-time algorithm for k-sat. *Journal of the ACM (JACM)*, ACM, v. 52, n. 3, p. 337–364, 2005. Citado na página 23.

PELIKAN, M.; GOLDBERG, D. E.; LOBO, F. G. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, Springer, v. 21, n. 1, p. 5–20, 2002. Citado 3 vezes nas páginas 25, 34 e 35.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3. ed. New York, NY, USA: Cambridge University Press, 2007. ISBN 0521880688, 9780521880688. Citado na página 23.

RADER, D. J. *Deterministic Operations Research: Models and Methods in Linear Optimization*. New Jersey, USA: John Wiley & Sons, 2010. Citado na página 23.

RICE, J. R. The algorithm selection problem. *Advances in computers*, Elsevier, v. 15, p. 65–118, 1976. Citado 5 vezes nas páginas 15, 17, 43, 47 e 48.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 38.

SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, ACM, v. 41, n. 1, p. 6, 2009. Citado 3 vezes nas páginas 17, 43 e 44.

SYSWERDA, G. Uniform crossover in genetic algorithms. Morgan Kaufmann Publishers, Inc., 1989. Citado na página 30.

TALBI, E.-G. *Metaheuristics: from design to implementation*. New Jersey, USA: John Wiley & Sons, 2009. v. 74. Citado 6 vezes nas páginas 14, 16, 17, 23, 24 e 45.

THRUN, S.; PRATT, L. *Learning to learn*. .: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas 17 e 43.

VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, Springer, v. 18, n. 2, p. 77–95, 2002. Citado 2 vezes nas páginas 15 e 18.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 67–82, 1997. Citado na página 14.

XU, L.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Satzilla-07: the design and analysis of an algorithm portfolio for sat. In: SPRINGER. *International Conference on Principles and Practice of Constraint Programming*. USA, 2007. p. 712–727. Citado 3 vezes nas páginas 17, 44 e 45.

XU, L.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, v. 32, p. 565–606, 2008. Citado 5 vezes nas páginas 17, 44, 45, 46 e 51.

ZHANG, H.; SHEN, H.; MANYà, F. Exact Algorithms for MAX-SAT. *Electronic Notes in Theoretical Computer Science*, v. 86, n. 1, p. 190 – 203, 2003. ISSN 1571-0661. FTP'2003, 4th International Workshop on First-Order Theorem Proving (in connection with RDP'03, Federated Conference on Rewriting, Deduction and Programming). Disponível em: <http://www.sciencedirect.com/science/article/pii/S1571066104806637>. Citado 2 vezes nas páginas 14 e 16.