Pós-Graduação em Ciência da Computação

Nicolas Melo de Oliveira

Otimização em Computadores Quânticos: *Formulação QUBO para o Problema CMO*

Universidade Federal de Pernambuco
posgrad@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife

2018

Nicolas Melo de Oliveira

Otimização em Computadores Quânticos: *Formulação QUBO para o Problema CMO*

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Ricardo Martins de Abreu Silva
Co-orientador: Wilson Rosa de Oliveira Jr

Recife
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

O48o Oliveira, Nicolas Melo de
Otimização em computadores quânticos: formulação QUBO para o problema CMO / Nicolas Melo de Oliveira. – 2018.
75 f.: il., fig., tab.

Orientador: Ricardo Martins de Abreu.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Redes neurais artificiais. 2. Otimização. 3. Computação quântica. I. Abreu, Ricardo Martins de (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2018-096

Nicolas Melo de Oliveira

Otimização em Computadores Quânticos: Formulação QUBO para o Problema CMO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

Aprovado em: 22/02/2018.

BANCA EXAMINADORA

_____

Prof. Dr. Silvio de Barros Melo
Centro de Informática/UFPE

_____

Prof. Dr. Tiago Alessandro Espínola Ferreira
Departamento de Estatística e Informática / UFRPE

_____

Prof. Dr. Ricardo Martins de Abreu
Centro de Informática / UFPE
(Orientador)

Dedico este trabalho à minha namorada, Gabriella, amiga e companheira, que foi porto seguro perante às dificuldades deste percurso.

AGRADECIMENTOS

Agradeço a todos que contribuíram direta e indiretamente para que eu pudesse concluir com sucesso mais esta etapa da minha vida acadêmica. Faço, aqui, um agradecimento especial ao professor e orientador Ricardo Martins de Abreu Silva por me conceder a oportunidade de executar esta pesquisa sob sua orientação. Igualmente, agradeço ao professor e co-orientador Wilson Rosa de Oliveira pelo apoio e conselhos dados durante o período de execução deste trabalho.

A todos os professores e funcionários do Centro de Informática/UFPE.

Aos meus colegas, de vida e de academia, pelos momentos de conversa e descontração.

À minha namorada, pela paciência necessária para me apoiar durante este caminho.

À FACEPE, pelo suporte financeiro.

"If we knew what it was we were doing, it wouldn't be called 'research', would it?"
(Albert Einstein)

RESUMO

A computação quântica é um paradigma computacional que tem motivado o aparecimento de diversas pesquisas que visam apresentar soluções para problemas, atualmente, considerados difíceis. O surgimento de algoritmos quânticos que operam mais rápido que seus análogos clássicos tem feito com que corporações como Google, NASA e IBM invistam nesse paradigma. Por isso, intensificou-se a busca por formulações alternativas para problemas, visando resolvê-los em um ambiente computacional quântico. Entre essas formulações, estão os trabalhos que abordam a resolução de problemas na computação quântica a partir de formulações QUBO (*Quadratic Unconstrained Binary Optimization*), um problema NP-difícil cujo princípio consiste em minimizar uma função quadrática. O formato QUBO é comumente utilizado na literatura quando da solução quântica de problemas de otimização e diversos autores têm recorrido à esta caracterização devido à sua aplicabilidade em uma considerável gama de problemas. Dessa forma, representar um dado problema como um problema QUBO implica diretamente que podemos executá-lo em um ambiente computacional quântico (genérico ou de propósito específico). O problema CMO (*Contact Map Overlap*) é definido como um problema de otimização combinatória NP-difícil que consiste na medida de semelhança entre pares de proteínas com base em seus respectivos mapas de contato. Em bioinformática, o estudo de problemas que buscam por funções similares entre estruturas biológicas, especialmente de proteínas, é um campo de grande interesse. Com isso, este trabalho de pesquisa aborda o problema CMO na conjectura da resolução dos problemas de otimização em computadores quânticos através da formulação QUBO dos mesmos. Além de fornecermos uma formulação QUBO para o problema de proteína CMO, resultados experimentais foram obtidos com o auxílio da ferramenta *qbsolv* e validaram esta abordagem como uma alternativa aos métodos clássicos existentes.

Palavras-chaves: Otimização combinatória. Computação quântica. Formulação QUBO. Problema CMO.

ABSTRACT

Quantum computing is a computational paradigm that has motivated the appearance of several researches aimed at presenting solutions to problems, currently, considered difficult. The emergence of quantum algorithms that operate faster than their classic analogues has made corporations such as Google, NASA, and IBM invest in this paradigm. Therefore, the search for alternative formulations for problems was intensified, aiming at solving them in a quantum computational environment. Among these formulations are the problem solving in quantum computing from QUBO (*Quadratic Unconstrained Binary Optimization*) formulations, a NP-hard problem whose principle is to minimize a quadratic function. QUBO format is commonly used in the literature when looking for quantum solution of optimization problems and several authors have resorted to this characterization due to its applicability in a considerable range of problems. Thus, representing a given problem as a QUBO problem directly implies that we can run it in a quantum computational environment (generic or specific purpose). The CMO (*Contact Map Overlap*) problem is defined as an NP-hard combinatorial optimization problem consisting of the measure of similarity between pairs of proteins based on their respective contact maps. In bioinformatics, the study of problems that seek similar functions between biological structures, especially proteins, is a field of great interest. Therefore, this research work addresses the CMO problem in the conjecture of solving optimization problems in quantum computers through their QUBO formulation. In addition to providing a QUBO formulation for the CMO protein problem, experimental results were obtained with the help of the *qbsolv* tool and validated this approach as an alternative to existing classical methods.

Key-words: Combinatorial optimization. Quantum computing. QUBO formulation. CMO problem.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMO	Contact Map Overlap
CQ	Computação Quântica
GPU	Graphics Processing Unit
IQ	Informação Quântica
MQ	Mecânica Quântica
QUBO	Quadratic Unconstrained Binary Optimization

SUMÁRIO

1 INTRODUÇÃO

Grandes avanços feitos na forma clássica de realizar computação em dispositivos digitais, como o advento de processadores multi-core e o uso de unidades de processamento gráfico (Graphics Processing Unit (GPU)) são exemplos que caracterizam uma época onde a demanda por mais poder computacional tem levado a tecnologia atual ao seu limite. Uma das barreiras encontradas por cientistas e pesquisadores diz respeito ao tamanho dos componentes que formam os chips de processamento. Por isso, caso a *Lei de Moore* (MOORE, 1998) continue se concretizando, tais componentes deverão chegar à escalas subatômicas em alguns anos. E esse é um dos cenários que faz com que se manifeste o interesse por modelos não-clássicos de computar, como a Computação Quântica (CQ).

Assim, devido aos resultados significantes de alguns algoritmos quânticos, superando seus correspondentes clássicos, a computação quântica tornou-se um paradigma que desperta elevado interesse de grandes corporações, como Google, Microsoft, NASA, Intel e IBM. O algoritmo de fatoração em tempo polinomial proposto por Shor (SHOR, 1994) e o algoritmo de busca em uma base de dados desordenada proposto por Grover (GROVER, 1996), os quais operam mais rápido que qualquer algoritmo clássico análogo conhecido, são exemplos primordiais da supremacia da computação quântica frente à computação clássica. Isto, claro, tem um significado importante quando falamos em solucionar problemas considerados difíceis de se computar com a tecnologia atual. Cabe ressaltar que a computação quântica possui mais de um modelo presente na literatura, porém, aqui, serão abordados apenas os principais: baseado em circuitos e baseado no teorema adiabático.

Problemas de otimização combinatória configuram-se como sendo um dos problemas mais promissores no que diz respeito à resolução em processadores quânticos. Tais problemas, que são (em geral) difíceis de serem computados pelo hardware clássico existente, possuem aplicações práticas e recorrentes em processos industriais onde busca-se minimizar ou maximizar o uso de algum recurso, por exemplo. Dessa forma, existe uma ampla capacidade de desenvolvimento de pesquisas que visam soluções alternativas para estes problemas, uma vez que mesmo problemas que não são originalmente de otimização podem ser modelados assim, para serem resolvidos com as abordagens quânticas propostas na literatura.

Portanto, a computação quântica surge como um campo de estudo que apresenta um grande potencial para a solução eficiente de problemas e é uma área de pesquisa extremamente ativa. Características peculiares como o paralelismo quântico - e outros fenômenos, até então, apenas observados nas leis da mecânica quântica, fazem com que a aplicação de tal paradigma computacional em problemas para os quais não se conhece soluções algorítmicas eficientes desperte o interesse de vários pesquisadores. Assim, com o desenvolvimento atual de projetos que envolvem computadores quânticos, intensificou-se

a busca por formulações para problemas que se enquadram nesse novo paradigma. Entre essas, estão os trabalhos que abordam a resolução de problemas na computação quântica a partir da reformulação deles como um problema de otimização binária irrestrita quadrática (Quadratic Unconstrained Binary Optimization (QUBO)). O formato QUBO, bem como o modelo Ising, são comumente utilizados na literatura quando da solução quântica de problemas de otimização e diversos autores ((BAPST et al., 2013), (BIAN et al., 2010), (HUA, 2016), (CAO et al., 2016), (LUCAS, 2014), (SU; TU; HE, 2016)) têm recorrido à esta caracterização devido à sua aplicabilidade em uma considerável gama de problemas. A "otimização binária quadrática irrestrita" é um problema NP-difícil (WANG; KLEINBERG, 2009) cujo princípio consiste em minimizar uma função quadrática. Logo, uma das implicações diretas do emprego de formulações QUBO é que podemos usar os computadores quânticos da empresa canadense D-Wave para resolver o problema em questão.

Em bioinformática, a análise das características de proteínas é uma área de estudos intensivos. Este interesse surge da possibilidade de identificar funções semelhantes entre estruturas tridimensionais de diferentes proteínas. Uma abordagem usada para reconhecer essas similaridades entre proteínas é baseada na comparação de seus mapas de contato, onde é feita a suposição inicial de que o alinhamento de certas estruturas entre pares desses mapas resulta, também, em funções similares entre as proteínas que eles representam. Assim, o problema Contact Map Overlap (CMO) é definido como um problema de otimização combinatória NP-difícil (GOLDMAN; ISTRAIL; PAPADIMITRIOU, 1999) que consiste na medida de semelhança entre pares de proteínas com base em seus respectivos mapas de contato. Estes, por sua vez, são representações das estruturas tridimensionais das proteínas e contém informações sobre resíduos e contatos entre estes - uma distância entre resíduos menor que um dado limiar configura um contato.

Entender completamente a física que rege o processo do arrefecimento quântico na computação tem sido um objetivo da comunidade científica. Aqui, nesta dissertação, aplicamos o princípio das formulações QUBO no problema CMO, tendo em vista sua resolução em computadores quânticos. Almejamos demonstrar a capacidade que a computação quântica, em conjunto com esse tipo de formulação, tem de solucionar problemas de otimização que envolvem milhares de variáveis binárias. Também, comparamos nossos resultados com aqueles obtidos através de *solvers* clássicos encontrados na literatura.

Cabe notar que existem algoritmos clássicos para o problema CMO que o resolvem com um desempenho melhor do que o abordado aqui, neste trabalho. Devido à ausência de acesso à uma máquina D-Wave, não fomos capazes de realizar comparações com um computador quântico real. A otimização quântica necessita de um considerável estudo adicional para se tornar algo prático e usual cientificamente. O avanço na técnica de arrefecimento quântico deve trazer melhorias com relação à coerência e controle dos qubits, tornando possível que problemas de maior relevância prática sejam solucionados de maneira eficiente.

Com este aparato estabelecido, este trabalho de pesquisa aborda o problema CMO na conjectura da resolução dos problemas de otimização em ambientes computacionais quânticos através da formulação QUBO dos mesmos. Ressalto que esta dissertação não tem por objetivo detalhar quaisquer minuciosidades acerca da realização física dos dispositivos quânticos em questão. Portanto, assumimos aqui a existência de processadores quânticos capazes de solucionar problemas caracterizados na sua forma QUBO. Tampouco, aqui neste trabalho, serão levados em consideração os tempos despendidos na execução dos experimentos - ainda que estes estejam expostos no Capítulo 4 apenas para complementar todo o processo realizado e detalhar os resultados obtidos. Ainda sobre o Capítulo 4, como não obtivemos acesso ao computador quântico a tempo de realizar os experimentos para esta dissertação, os resultados aqui expostos têm caráter puramente de validação da abordagem.

1.1 Objetivos

1.1.1 Geral

O objetivo geral deste trabalho consiste na investigação de formulações QUBO como caracterização para problemas de otimização, bem como na validação dessa abordagem para encontrar soluções em ambientes computacionais quânticos. Nosso objetivo, nesse primeiro momento, não é buscar por soluções ótimas ou superar as soluções clássicas existentes, mas sim mostrar que a abordagem CMO-QUBO pode produzir bons resultados e se qualificar como uma alternativa aos métodos existentes.

1.1.2 Específicos

Como objetivos específicos, tem-se os seguintes:

- Analisar metodologias quânticas para solução de problemas de otimização;
- Compreender a formulação QUBO para problemas de otimização;
- Desenvolver uma formulação QUBO para o problema CMO, visando, sobretudo, sua solução em ambientes computacionais quânticos;
- Fazer uso da ferramenta *qbsolv* para validar a abordagem utilizada;
- Analisar os resultados obtidos para ilustrar a viabilidade de tal alternativa para solução de problemas de otimização;
- Identificar potenciais estudos futuros no sentido de melhorar os resultados obtidos e refinar o trabalho desenvolvido.

1.2 Organização da dissertação

Esta dissertação está organizada de tal modo que apresente a base teórica conceitual que serve de alicerce para o problema aqui atacado para, então, abordar a solução proposta e a discussão acerca dos resultados obtidos. Sendo assim, nos parágrafos seguintes, estarão descritos brevemente os capítulos do presente trabalho.

No Capítulo 2, são expostos nas Seções 2.1, 2.2 e 2.4 os conceitos referentes à Computação Quântica, Formulações QUBO e ao problema CMO, respectivamente. Tais definições são imprescindíveis para uma razoável compreensão das características e peculiaridades destes assuntos. As principais propriedades da computação quântica são mostradas partindo do conceito básico de bits quânticos e explorando seus dois modelos mais relevantes: o baseado em circuitos e o baseado no teorema adiabático (juntamente com o arrefecimento quântico, utilizado para solucionar problemas de otimização). Também são fornecidas as definições dos modelos Ising e QUBO, este último usado como intermediário entre o problema a ser atacado e sua resolução em computação quântica. Por fim, é dada a caracterização do problema de similaridade de proteínas CMO, o qual é abordado no restante do texto como um exemplo de problema de otimização específico a ser reduzido ao problema QUBO para possibilitar sua solução em computadores quânticos. Também, neste capítulo, é dedicado um espaço às abordagens quânticas que objetivam resolver problemas de otimização. A intenção aqui é mostrar alguns trabalhos e possibilidades que abrem caminho para pesquisas que visam buscar alternativas aos *solvers* clássicos para esse tipo de problema. São descritos os principais métodos encontrados na literatura no que diz respeito ao uso da computação quântica para solucionar problemas de otimização, tanto no modelo de circuitos quanto no modelo adiabático.

O Capítulo 3 detalha a principal contribuição deste trabalho de pesquisa. Aqui, é apresentada e descrita a formulação QUBO que representa o problema CMO. São fornecidos detalhes de como a formulação foi desenvolvida, restrições do problema original a serem mapeadas no modelo QUBO e penalidades para criação da matriz QUBO. Após isso, um pequeno exemplo hipotético é descrito para ilustrar o processo de formação da matriz correspondente ao problema.

Detalhes sobre os resultados obtidos e os experimentos executados são fornecidos no Capítulo 4. Neste capítulo, todo o processo é detalhado: a base de dados utilizada, os *scripts* criados para a geração das matrizes (incluídos nos Apêndices A e B), a ferramenta *qbsolv* e seu algoritmo - usada para execução clássica dos experimentos com o objetivo de validar a formulação, mas que também possibilita a resolução de problemas em uma abordagem híbrida clássica/quântica em computadores da D-Wave - e as características do aquivo no formato .qubo (inserido como entrada para o *qbsolv*). Por fim, são apresentados os resultados dos experimentos e uma breve discussão acerca deles. Ressalto que estes resultados foram expostos em comparação com dois métodos clássicos bem estabelecidos na literatura.

Finalmente, o Capítulo 5 versa sobre as conclusões do trabalho desenvolvido durante esta pesquisa, bem como cita alguns caminhos que podem ser seguidos com relação a trabalhos futuros envolvendo a solução de problemas de otimização em computação quântica. Os Apêndices A, B e C exibem os códigos desenvolvidos para esta dissertação e são responsáveis, respectivamente, por: criar a matriz QUBO correspondente ao problema CMO, criar a matriz QUBO que corresponde ao problema CMO de acordo com a segunda abordagem, criar o arquivo .qubo a partir de uma matriz QUBO de entrada.

2 CONCEITOS E REFERENCIAIS TEÓRICOS

Objetivando fornecer uma fundamentação teórica suficiente para a compreensão da presente pesquisa, bem como visando ressaltar os principais aspectos de relevância para este trabalho, este capítulo fornece uma revisão dos textos fundamentais das áreas de interesse nas quais esta dissertação está inserida. Aqui, serão abordados os conceitos básicos e imprescindíveis para a assimilação do conteúdo dos demais capítulos desta dissertação. Tais conceitos estão fundamentados em referenciais teóricos disponíveis na literatura e permeiam a computação quântica (Seção 2.1), as formulações QUBO/Ising (Seção 2.2), a ferramenta *qbsolv* (Seção 2.3) e o problema CMO (Seção 2.4).

Este capítulo também dará espaço para trabalhos que fazem uso da computação quântica para resolver problemas de otimização fazendo uso, ou não, de formulações Ising/QUBO. Trabalhos que empregam o *qbsolv* para realizar experimentos nesse sentido também serão citados.

2.1 Computação quântica

Juntamente com a crescente demanda por poder computacional, também surgiram especulações sobre quais modelos alternativos ao modelo clássico vigente poderiam garantir a evolução da computação, haja vista que os limites da tecnologia atual parecem estar cada vez mais próximos. O interesse inicial pela computação quântica se deu em virtude da possibilidade de resolver os problemas presentes na mecânica quântica de forma mais eficiente que a computação clássica. Atualmente, cresce ainda mais a relevância da computação quântica, cujos efeitos provenientes da física quântica na computação podem ser usados, em determinados casos, para encontrar soluções para problemas difíceis de maneira mais eficiente que seus análogos clássicos. No decorrer de toda esta seção, serão explicitados alguns aspectos referentes aos dois modelos mais populares da computação quântica: o de circuitos quânticos e o adiabático. Além disso, metodologias que empregam a CQ para resolução de problemas de otimização também serão citadas.

Existem diversas abordagens para o estudo de computação e informação quântica. Em (MERMIN, 2003) é utilizada a analogia entre bits clássicos e quânticos para, com isso, explanar sobre computação quântica para cientistas da computação. Com a mesma abordagem (MERMIN, 2007) aprofunda conceitos básicos da computação quântica, como algoritmos, criptografia, correção de erros e protocolos de comunicação, e classifica o computador quântico como aquele cuja operação explora certas transformações especiais de seu estado interno, estas, garantidas pelas leis da Mecânica Quântica (MQ).

Em (JAEGER, 2007) denomina-se Informação Quântica (IQ) o transporte, armazenamento e processamento de informação utilizando sistemas de MQ. Neste livro é dado um

maior foco à teoria da informação quântica - a qual se configura como uma junção entre a mecânica quântica e a teoria da informação. Nele, são tratados temas como: correção de erro, entropia, criptografia, compressão de dados, operações quânticas, emaranhamento, protocolos quânticos, comunicação (clássica e quântica), algoritmos quânticos e teleporte quântico.

Na mesma linha do livro anterior, (NIELSEN; CHUANG, 2011) apresenta um material com um conteúdo abrangente sobre IQ. Entretanto, antes disso, são apontados diversos conceitos sobre MQ, CQ e ciência da computação, abordando, inclusive, perspectivas de realizações físicas para os computadores quânticos.

Outros livros-texto sobre CQ são: (MCMAHON, 2007), que, além dos conceitos usuais da computação quântica, fornece um material que aborda aplicações do emaranhamento, teoria da informação quântica e o modelo de computação quântica adiabática - outro ponto que deve ser ressaltado sobre este livro diz respeito à grande quantidade de exemplos e exercícios presentes em todos os capítulos, o que o torna uma boa alternativa para o ensino/aprendizado inicial da CQ; já em (YANOFSKY; MANNUCCI, 2008) tem-se uma abordagem totalmente voltada para cientistas da computação, onde não é esperado que o leitor tenha conhecimentos avançados prévios sobre matemática ou física - aqui, também são encontrados diversos exemplos ilustrando os conceitos expostos, bem como uma discussão sobre linguagens de programação para CQ.

Em (DEUTSCH; JOZSA, 1992), (GROVER, 1996) e (SHOR, 1994) são apresentados três dos algoritmos mais conhecidos da computação quântica. Deutsch, além de descrever o primeiro computador quântico universal, também desenvolveu o "*Algoritmo de Deutsch*" (DEUTSCH, 1985), que consiste em avaliar se uma dada função é contínua ou balanceada. Apesar de uma limitada aplicação técnica, este algoritmo exibe uma amostra de como os computadores quânticos podem ser exponencialmente mais eficientes em comparação aos computadores clássicos. Em (DEUTSCH; JOZSA, 1992) tem-se uma generalização do algoritmo de Deutsch. Por sua vez, o "*algoritmo de Grover*" (GROVER, 1996) representa uma ganho quadrático quando comparado ao análogo clássico na busca por elementos em uma lista desordenada. Já (SHOR, 1994) mostra em seu trabalho a descoberta de um algoritmo quântico em tempo polinomial para fatoração de números inteiros grandes em seus fatores primos. Tais explorações do poder computacional quântico possibilitaram o surgimento de estudos que se propuseram a resolver outros problemas computacionais de forma eficiente.

Para um estudo sobre os recursos matemáticos inerentes à informação e computação quântica, sugere-se a leitura do texto apresentado por (OHYA; VOLOVICH, 2011). Seu conteúdo engloba desde aspectos relativos às bases da probabilidade clássica e quântica, passando pelos fundamentos da teoria da comunicação quântica, entropia, emaranhamento e algoritmos quânticos, até culminar em aplicações para os problemas NP-completos, criptografia e teleporte quânticos.

Um computador quântico consiste numa máquina capaz de executar cálculos e operações computacionais baseando-se em propriedades e fenômenos da mecânica quântica. Em computadores clássicos a unidade de informação é o *bit*, o qual pode assumir os valores lógicos "0" ou "1". Analogamente, a unidade de informação quântica é o bit quântico, ou *qubit*. A um qubit podem ser atribuídos os valores lógicos "0", "1", ou qualquer superposição destes. Esta superposição consiste de uma combinação linear dos estados da base computacional descrita por um vetor com a seguinte forma básica, onde, para cada posição desse vetor, um dos estados está associado:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Na notação de Dirac, o vetor $|\psi\rangle$ é chamado de ***ket*** e seu conjugado transposto, $\langle\psi|$, de ***bra***. No estado quântico acima, α e β são amplitudes probabilísticas associadas aos estados, representadas por números complexos e que obedecem à seguinte regra de normalização:

$$|\alpha|^2 + |\beta|^2 = 1$$

Outra forma de interpretar um estado quântico é por meio da sua parametrização através de ângulos θ e ϕ:

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\phi}sen\frac{\theta}{2}|1\rangle$$

Com o que foi brevemente descrito acima, pode-se observar que em um sistema quântico é possível obter um alto grau de paralelismo computacional através da superposição de diversos estados. Esse paralelismo é uma das características que pode tornar o computador quântico uma alternativa eficiente aos computadores clássicos no que se refere à execução de algoritmos de ordem exponencial (classicamente) em tempo polinomial. Tal característica é expressa do seguinte modo:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i|i\rangle,$$

onde se tem n qubits e a_{2^n-1} amplitudes probabilísticas associadas a i estados. Dessa forma, n qubits podem representar, em um único estado, as 2^n combinações possíveis. Fisicamente, qubits são representados por qualquer objeto quântico que possua dois autoestados bem definidos. Na Figura 1 é mostrada uma interpretação gráfica geométrica de como pode ser representado um qubit através da *esfera de Bloch*.

O sistema quântico descrito até aqui está associado a um espaço vetorial complexo chamado *espaço de Hilbert*. Caso haja necessidade de representar mais de um qubit no

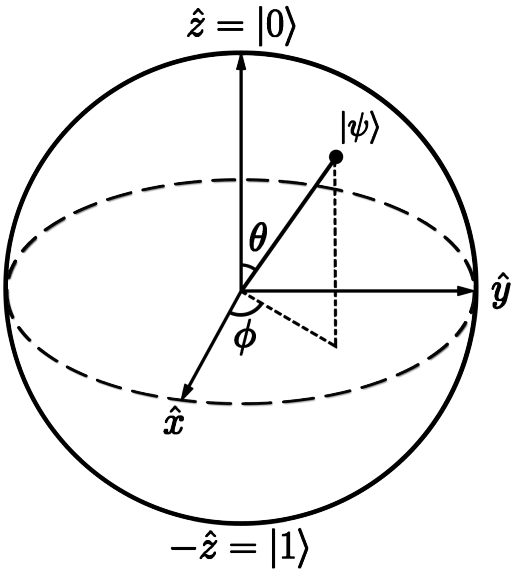


Figura 1 – Representação geométrica do *qubit* através da *esfera de Bloch*

espaço de Hilbert, é feito uso do produto tensorial, que se caracteriza como uma operação bilinear entre espaços vetoriais. A operação de dois ou mais qubits pelo produto tensorial é representada pelo símbolo $\otimes$. Como exemplo, tem-se a seguir o produto tensorial entre 2 qubits expresso em termos simbólicos e de forma matricial:

$$|0\rangle \otimes |0\rangle = |00\rangle; |0\rangle \otimes |1\rangle = |01\rangle; |1\rangle \otimes |0\rangle = |10\rangle; |1\rangle \otimes |1\rangle = |11\rangle$$

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}; |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Generalizando o produto tensorial de vetores representados por matrizes, sejam A e B vetores do espaço complexo, sendo $A \in \mathbb{C}^{m \times n}$ e $B \in \mathbb{C}^{p \times q}$, com $a_{1,1}$ até $a_{m,n}$ sendo os elementos que compõem A, $A \otimes B$ é dado por:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}B & a_{1,1}B & \cdots & a_{m,n}B \end{bmatrix},$$

e resulta em uma matriz de dimensões $mp \times nq$.

Contudo, nem sempre um estado quântico de 2 ou mais qubits pode ser representado dessa forma. Quando este estado não pode ser descrito como um produto tensorial é dito que o mesmo encontra-se *emaranhado*. Dado um sistema representado pelo estado $|\psi\rangle$, se não for possível decompor o mesmo através do produto tensorial de dois ou mais vetores, então há um *emaranhamento*. Como exemplo:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

o qual não pode ser representado por meio do produto tensorial de 2 qubits. Tal particularidade faz com que, uma vez realizada uma operação sobre um qubit desse sistema, mesmo que os outros qubits deste estejam fisicamente separados por uma grande distância, estes também sofrerão um efeito da operação aplicada.

Outro aspecto característico da CQ diz respeito a seus operadores. Sob a condição de estar em um "sistema quântico fechado", as operações sob um estado quântico são feitas por operadores unitários. Dado um operador $U : H \to H$, com H denotando o espaço de Hilbert, U é referido ser unitário quando seu inverso é igual ao seu conjugado transposto:

$$U^{-1} = U^{\dagger} \qquad ou \qquad UU^{\dagger} = U^{\dagger}U = I,$$

onde I designa o operador *identidade*. Agora, considere um registrador com n qubits no estado:

$$|\psi\rangle := \sum_{i=0}^{2^n-1} \alpha_i |i\rangle,$$

a aplicação de uma matriz unitária U de dimensão 2^n sobre este registrador, produz:

$$U|\psi\rangle = U\left(\sum_{i=0}^{2^n-1} \alpha_i |i\rangle\right) = \sum_{i=0}^{2^n-1} \alpha_i U|i\rangle.$$

Isto é, uma única aplicação de U realiza um número exponencial de operações em estados básicos. Este fenômeno é chamado de *paralelismo quântico*. Usualmente, a matriz de *Hadamard* é utilizada para gerar, num registrador, um estado quântico contendo a superposição de todos os estados básicos, todos com a mesma amplitude.

Uma restrição que é aplicada ao modelo de computar da CQ se refere à *medição* necessária para extrair qualquer informação resultante de um estado de qubits. Ao realizar esta operação, o estado colapsa e apenas um dos qubits que estavam em superposição será observado. Então, dado um estado quântico:

$$|\psi\rangle := \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \text{com} \sum_{i}^{2^{n-1}} |\alpha_i|^2 = 1$$

após ser realizada a medição, a probabilidade de se obter a saída $|i\rangle$ é $p_i = |\alpha_i|^2$.

Um ponto a ser relembrado é que a computação quântica não baseia-se em um único modelo. Duas principais metodologias estão sendo empregadas atualmente tanto para a concepção de algoritmos quânticos quanto para a construção de máquinas reais. O primeiro destes modelos é o fundamentado na utilização de portas lógicas quânticas para realizar a computação, análogo ao modelo clássico de computar. O segundo modelo diz respeito à evolução adiabática de um sistema físico. Essas duas abordagens serão detalhadas nas subseções a seguir.

2.1.1 Modelo de circuitos

A computação quântica de circuitos é análoga ao modelo computacional clássico baseado em circuitos. Dessa forma, a computação é realizada através de portas lógicas quânticas que executam uma sequência de transformações unitárias nos qubits. A Figura 2 exibe um pequeno exemplo de circuito quântico onde pode ser observada a saída $f(x, y, z)$ no último qubit.

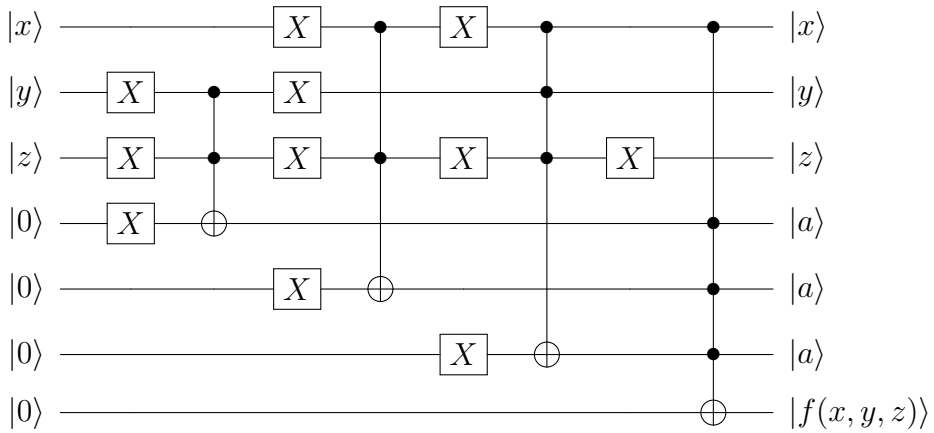


Figura 2 – Exemplo de circuito quântico

Algumas dificuldades inerentes a este modelo consistem em controlar os qubits, sendo necessário implementar algoritmos de correção que precisam de muitos qubits adicionais; o outro empecilho diz respeito à dimensão dos circuitos, que podem requerer milhares de qubits. Como forma de contornar esses problemas, tem-se em (WONG, 2012) um estudo sobre como otimizar circuitos quânticos.

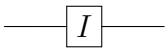
Com relação a este modelo, cabe a menção aqui à IBM, com seu projeto *IBM Q - Quantum Experience* (SANTOS, 2017) que, até a presente data, alega ter desenvolvido um computador quântico de circuitos com 20 qubits. Além deste, está em desenvolvimento

uma nova máquina com 50 qubits, o que possibilitaria resolver uma extensa gama de problemas nesta arquitetura.

Agora, serão apresentadas algumas portas lógicas quânticas comumente empregadas quando da criação de circuitos quânticos. Importante frisar que o comportamento das portas lógicas em questão pode ser visto, também, na computação quântica adiabática através da simulação da computação quântica de circuitos via processos adiabáticos. Tal peculiaridade, bem como uma prova de que a computação quântica adiabática é equivalente à computação quântica de circuitos, pode ser vista em (AHARONOV et al., 2008). Sobre as portas lógicas a serem exibidas, serão fornecidas suas formas matriciais, o resultado da ação sobre qubits e a representação no circuito. Para mais detalhes acerca das portas lógicas e circuitos quânticos, consultar os textos contidos em (MERMIN, 2003), (NIELSEN; CHUANG, 2011), (YANOFSKY; MANNUCCI, 2008) e (MCMAHON, 2007).

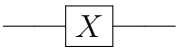
- Identidade - o mesmo qubit de entrada é obtido na saída do circuito:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \begin{aligned} I|0\rangle &= |0\rangle \\ I|1\rangle &= |1\rangle \end{aligned}$$



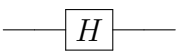
- Not - o qubit inicial tem seu valor invertido:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \begin{aligned} X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle \end{aligned}$$



- Hadamard - cria uma superposição dos estados da base:

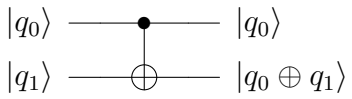
$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}, \qquad \begin{aligned} H|0\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$



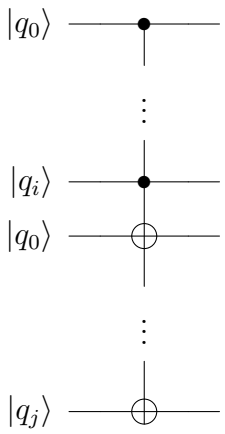
- Controlled-Not - porta lógica de 2 qubits na qual o qubit alvo, representado pelo sinal $\oplus$ no circuito, tem seu valor alterado caso o outro qubit (o de controle) tenha valor 1; caso contrário, nada é alterado:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$CNOT|0\rangle|0\rangle = |0\rangle|0\rangle \quad CNOT|1\rangle|0\rangle = |1\rangle|1\rangle$$
$$CNOT|0\rangle|1\rangle = |0\rangle|1\rangle \quad CNOT|1\rangle|1\rangle = |1\rangle|0\rangle$$

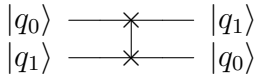


- Controlled-Gate - uma generalização de portas lógicas com bits controladores que pode incluir mais de um qubit como função de controle (i) e mais de um qubit como alvo (j):



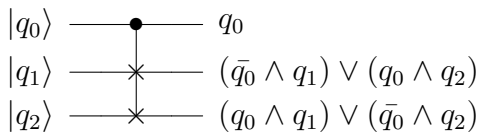
- Swap - porta lógica de 2 qubits que realiza uma troca entre os valores dos mesmos:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- Fredkin - porta lógica de 3 qubits que atua como uma *troca controlada*; dessa forma, caso o bit de controle (•) tenha valor 1 os outros dois qubits (×) tem seus valores trocados entre si:

$$Fredkin = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



2.1.1.1 Otimização em circuitos quânticos

Com relação a este tema, é suficiente expor o trabalho desenvolvido em (ZAHEDINEJAD; ZARIBAFIYAN, 2017), que fornece um resumo de 4 principais metodologias para a realização de otimização quântica no modelo de circuitos:

- (1) **Simulação de computação quântica com portas lógicas quânticas** (AHARONOV et al., 2008) e (DAM; MOSCA; VAZIRANI, 2001) - a ideia aqui é decompor a evolução unitária de um sistema quântico em uma sequência de portas lógicas quânticas simples, o que permitiria uma espécie de simulação do operador Hamiltoniano responsável por codificar e resolver um dado problema. Um exemplo desta abordagem pode ser visto em (BARENDS et al., 2016).

- (2) **Aplicabilidade da busca de Grover em otimização global** (BARITOMPA; BULGER; WOOD, 2005) - trata-se do uso da busca de Grover como uma sub-rotina para algoritmos de otimização global. Por ter sido mostrado em (ZALKA, 1999) que a busca de Grover é ótima para uma base de dados desestruturada, ela tem sido usada da seguinte forma: o processo inicia-se a partir de uma solução inicial e a busca de Grover busca por qualquer outra solução com um menor custo; a cada iteração o espaço de busca é dividido em dois subespaços para que a busca procure por soluções no espaço que possui estados cuja energia seja menor do que a alcançada pela solução

vigente. Um outro exemplo da aplicação deste método pode ser estudado em (LIU; KOEHLER, 2010).

- (3) **Algoritmos de otimização aproximada quântica** (FARHI; GOLDSTONE; GUTMANN, 2014) - algoritmo quântico geral que se propõe a encontrar soluções aproximadas para problemas de otimização combinatória. Aqui, os problemas de otimização combinatória são definidos como sendo "*especificados por n bits e m cláusulas. Cada cláusula é uma restrição sobre um subconjunto dos bits que é satisfeita por certas atribuições desses bits e não satisfeita para as outras atribuições*". Então, para *strings* de n bits, tem-se que a função objetivo é o número de cláusulas satisfeitas na seguinte forma: $C(z) = \sum_{\alpha=1}^{m} C_\alpha(z)$, onde $z = z_1 z_2 ... z_n$ e $C_\alpha(z) = 1$ se z satisfaz a cláusula α e 0 caso contrário. Assim, dado um inteiro $p \geq 1$ como parâmetro, o algoritmo apresentado é dependente deste e, dada uma instância de n bits, o objetivo consiste em encontrar uma string z, para a qual $C(z)$ se aproxima do máximo global da função. Portanto, com o uso da computação quântica, um estado quântico pode ser preparado para qualquer inteiro $p \geq 1$ e $2p$ ângulos. Já em (WECKER; HASTINGS; TROYER, 2016), o problema de aproximação do estado fundamental de um modelo Ising foi transformado em um problema supervisionado de aprendizagem de máquina. A função objetivo é representada da seguinte forma: $F_p(\boldsymbol{\theta}_Z, \boldsymbol{\theta}_X) = |\langle \boldsymbol{\theta}_X, \boldsymbol{\theta}_Z | \psi_g \rangle|$, onde $|\psi_g\rangle$ é o verdadeiro estado fundamental e $(\boldsymbol{\theta}_Z, \boldsymbol{\theta}_X)$ são os parâmetros de aprendizagem que determinam a evolução do sistema do estado inicial para $|\boldsymbol{\theta}_X, \boldsymbol{\theta}_Z\rangle$. Esta abordagem funciona bem para o problema de máxima satisfatibilidade (Max-SAT) e é mostrado em (BIAN et al., 2010) como transformar um problema QUBO em um problema Max-SAT com pesos, permitindo, assim, que problemas QUBO em geral possam ser resolvidos desta forma.

- (4) **Amostragem Metropolis quântica** (TEMME et al., 2011) - algoritmos quânticos podem se beneficiar da distribuição de amostras de níveis baixos de energia de um sistema quântico. Dessa forma, pode-se encontrar o estado fundamental do sistema, bem como estimar o estado fundamental de um Hamiltoniano no modelo Ising, ao iniciar o sistema em algum estado aleatório e usar a amostragem quântica para evoluí-lo em direção a um estado com menor energia. Em (TEMME et al., 2011) é possível analisar a realização do correspondente quântico do algoritmo de amostragem Metropolis em um sistema quântico.

Além de realizar uma síntese dos trabalhos que se propõem a resolver problemas de otimização combinatória no modelo de computação baseado em portas lógicas quânticas, Ehsan Zahedinejad fornece em (ZAHEDINEJAD; ZARIBAFIYAN, 2017) uma introdução a algoritmos quânticos projetados para resolver tipos específicos de problemas na sua forma QUBO, como: problemas de satisfação de restrições (CERF; GROVER; WILLIAMS, 2000), problemas de satisfatibilidade booleana (AMBAINIS, 2004), problemas de conjunto

independente maximal (DÖRN, 2005) e problemas de clusterização (LLOYD; MOHSENI; REBENTROST, 2013).

2.1.2 Modelo adiabático

A ideia do uso do teorema adiabático na computação quântica (FARHI et al., 2000) surgiu com o propósito de explorar as capacidades quânticas de computar a resolução de problemas de otimização combinatória. Em termos de computabilidade, a computação quântica adiabática é equivalente ao modelo de circuitos quânticos, o que pode ser visto em (AHARONOV et al., 2008).

Um algoritmo quântico adiabático é descrito por um sistema Hamiltoniano e uma computação neste modelo é especificada por dois estados do Hamiltoniano chamados H_0 e H_1: o estado inicial H_0 geralmente consiste numa superposição de todos os estados da base que representa o problema e, ao final da computação adiabática, tem-se o estado fundamental de H_1, que representa a solução requerida para o problema (AHARONOV et al., 2008). Na computação quântica adiabática, um procedimento computacional é descrito pela dependência de tempo contínua de um Hamiltoniano. Assim, velocidade de evolução de um sistema quântico adiabático em um estado $|\psi(t)\rangle$ no tempo t é descrita pela *equação de Schrödinger* (FARHI et al., 2000).

Definição: *O teorema quântico adiabático para computação diz que, se o computador quântico inicia no estado fundamental de H_0, precisa terminar no estado fundamental de H_1, desde que a transição entre eles seja lenta o suficiente.*

Aqui, a questão principal está em como determinar o quão lenta esta transição precisa ser para encontrar a solução para o problema. Em (ALBASH; LIDAR, 2018) fornece uma visão geral sobre os aspectos relativos à natureza dessa variação lenta no teorema adiabático. Para mais detalhes sobre computação quântica adiabática, (FARHI et al., 2000) fornece uma explanação de seu funcionamento por meio de um algoritmo para resolução de instâncias do problema da satisfatiblidade booleana. Já em (CROSSON et al., 2014) são expostos resultados derivados do estudo da computação quântica adiabática visando solucionar instâncias do problema MAX 2-SAT. Neste mesmo trabalho, são dados três métodos que aumentam a probabilidade de sucesso ao medir o final da evolução quântica.

Sobre este modelo, tem-se a D-Wave, uma empresa canadense que tem desenvolvido dispositivos que empregam o arrefecimento quântico no modelo adiabático de computação quântica para resolver problemas de vidros de spin Ising (BIAN et al., 2010). O processo de arrefecimento quântico minimiza a energia Ising ao evoluir do estado fundamental de um Hamiltoniano inicial para o estado fundamental do Hamiltoniano do problema. O arrefecimento quântico será explicado com mais detalhes na Seção 2.1.2.1.

Ainda, problemas de interesse científico prático já foram codificados e resolvidos (em instâncias simples) em dispositivos experimentais usando Hamiltonianos Ising. Cabe ressaltar que o modelo *Ising* tem sido frequentemente chamado de QUBO na literatura mais

matemática. Resumidamente, o QUBO é um problema matemático NP-difícil cujo propósito consiste em minimizar uma função objetivo quadrática. Truques úteis tem sido desenvolvidos para fixar os valores de alguns *spins* e decompor grandes problemas QUBO em pequenos. Detalhes sobre os modelos Ising e QUBO podem ser lidos na Seção 2.2.

Em (ALBASH; LIDAR, 2018) temos um extenso resumo das características, variações e possibilidades da computação quântica adiabática. Um importante ponto explorado pelo autor é a discussão sobre a complexidade de se executar algoritmos quânticos adiabáticos, em comparação, por exemplo, com o modelo de circuitos quânticos. Porém, o objetivo maior deste artigo é explorar os processos adiabáticos na computação desde a sua origem. Os tópicos abordados incluem: o teorema adiabático e suas variações, algoritmos para computação quântica adiabática, provas da universalidade do modelo adiabático, cálculos de complexidade e computação quântica adiabática estocástica. Esta última configura-se como sendo o modelo para o qual esta dissertação direciona, quando da solução de problemas de otimização.

Como parte do anseio em tornar a computação quântica, em especial o modelo adiabático desta, o cenário ideal para a resolução de problemas de otimização combinatória, a D-Wave lançou em 01/2017 o *qbsolv* (BOOTH; REINHARDT; ROY, 2017), ferramenta que resolve problemas grandes na forma QUBO ao particioná-los em subproblemas direcionados para execução em sistemas D-Wave. Além disso, o *qbsolv* pode fazer uso de um solver clássico no lugar do arrefecimento quântico (presente no D-Wave) para os subproblemas, o que permite a execução de experimentos para aqueles que não possuem acesso a tais máquinas. Particularidades sobre o funcionamento desta ferramenta podem ser explorados na Seção 2.3.

Por fim, existe um debate no meio acadêmico sobre a real eficiência da computação quântica adiabática em resolver problemas considerados difíceis para a computação clássica. Ainda não é sabido se esse modelo quântico de computação é (ou se algum dia será) capaz de resolver problemas NP-completos em tempo polinomial, por exemplo. Porém, devido às suas características, é possível que a computação adiabática ainda seja mais eficiente que a clássica, pelo menos para algumas classes de problemas. Discussões que abordam este tema podem ser lidas em (DICKSON; AMIN, 2011), (FARHI et al., 2012), (HEN; YOUNG, 2011) e (JÖRG et al., 2010).

2.1.2.1 Arrefecimento quântico

O arrefecimento quântico consiste de uma técnica presente em computadores quânticos adiabáticos para resolver problemas de otimização combinatória que explora tunelamento (BOIXO et al., 2013a) e emaranhamento (MCGEOCH, 2014). Essa estratégia busca pelo mínimo global de uma dada função objetivo sob um conjunto de soluções candidatas e é derivada da meta-heurística clássica chamada *arrefecimento simulado*.

Arrefecer um material resfriando-o lentamente é uma técnica antiga para aprimorar as propriedades de vidros, metais e aço que tem sido usada por mais de 4 milênios (BOIXO et al., 2013b). O processo de arrefecimento quântico minimiza a energia Ising ao evoluir o Hamiltoniano do seu estado fundamental inicial H_0 para o estado fundamental do problema H_P. O estado fundamental H_0 é um estado de superposição no qual todas as configurações de *spin* são igualmente prováveis, enquanto no final do processo as configurações de *spin* com menor energia com relação a H_P são mais prováveis de serem medidas (BIAN et al., 2014). Um Hamiltoniano é, basicamente, uma descrição matemática de algum sistema físico e ele o descreve em termos de energias. É, essencialmente, uma fórmula onde você insere o estado de seu sistema e o Hamiltoniano lhe dá a energia desse sistema, que pode ser clássico ou quântico. Para uma abordagem física de como realizar o arrefecimento quântico tem-se o texto encontrado em (BOIXO et al., 2013b).

O funcionamento do arrefecimento quântico se dá da seguinte maneira: pensando em um diagrama de energia, o ponto mais baixo de energia representa a sobreposição dos estados; quando o arrefecimento quântico se inicia, o diagrama é transformado em um cenário de diversos mínimos locais, correspondendo aos diferentes estados possíveis. A forma de controlar onde o arrefecimento irá terminar é aplicando um campo magnético externo ao qubit, fazendo com que a probabilidade de se obter um dos estados seja maior do que para os outros. O verdadeiro poder deste processo surge quando se cria um emaranhamento entre os qubits e eles começam a influenciar um ao outro (feito com um dispositivo chamado *acoplador*, onde os qubits são vistos como um único objeto). Para que o acoplamento "escolha" quais configurações deseja estar no final do arrefecimento, ele diminui a energia dos estados que tem o mesmo valor ou dos estados que tem valores opostos. Cada um dos qubits envolvidos no emaranhamento podem ter um campo magnético aplicado a ele, podendo o usuário escolher todos os valores para os campos e emaranhamentos (tanto relativos à direção como à força). O conjunto de campos magnéticos e o conjunto de acoplamentos define um panorama de energia. Por sua vez, o arrefecedor quântico faz com que o arrefecimento resolva o sistema e encontre a energia mínima deste panorama. Após todo o processo, a saída do arrefecimento é um estado de energia $\mathbf{s}$ composto por *spins* Ising para cada qubit $s_i \in \mathbf{s}$ de tal forma que $\mathbf{s}$ corresponde ao estado de mínima energia do sistema. Em um modelo Ising tem-se que $s_i \in \{-1, +1\}$ porém, de maneira equivalente, para um problema formatado como QUBO temos $s_i \in \{0, 1\}$. Na Figura 3 tem-se uma simples representação sobre a diferença entre o arrefecimento simulado e o quântico. Classicamente, é preciso saltar sobre a barreira de energia/custo que separa um mínimo local de uma região com custo menor, enquanto que, quanticamente, pode-se realizar o tunelamento através da mesma. Se a barreira for alta o suficiente, o salto térmico se torna muito difícil devido às características do algoritmo de arrefecimento simulado/térmico. No entanto, se a barreira for estreita o suficiente, o tunelamento quântico se torna mais eficiente.

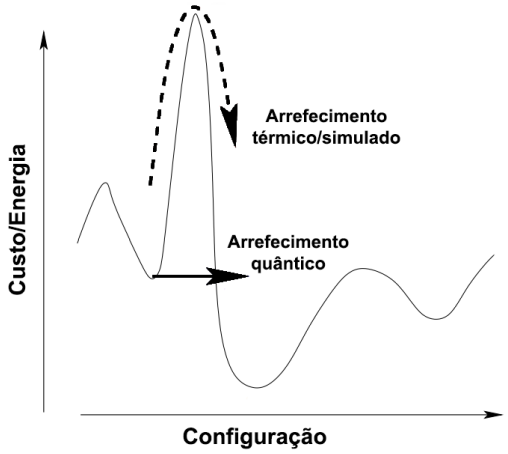


Figura 3 – Arrefecimento simulado × arrefecimento quântico

É natural suspeitar que o arrefecimento quântico seja similar ao arrefecimento simulado. Um modo de ver a diferença entre comportamento quântico e comportamento clássico nos qubits é a partir do congelamento - no arrefecimento o que acontece é que, em algum ponto do arrefecimento, a dinâmica cessa para cada qubit e então ele é fixado no estado 0 ou 1 e, a partir desse momento, o qubit não vai mais mudar seu estado. Porém, no comportamento clássico do arrefecimento, o bit muda de estado de forma dependente com a temperatura (quando a temperatura diminui, a dinâmica para). Por sua vez, o arrefecimento quântico é fracamente relacionado com a temperatura, ou seja, espera-se que a dinâmica continue a mesma, mesmo quando a temperatura é reduzida. Outra diferença entre os métodos é que, no processo simulado, a vizinhança permanece inalterada à medida que a busca é feita. Sendo assim, o parâmetro de temperatura é que determina a probabilidade de mudança de um estado para outro de maior energia. No entanto, no arrefecimento quântico, o parâmetro similar seria a força do campo de tunelamento, que determina o raio da vizinhança conforme o processo é executado, fazendo com que a vizinhança mude de acordo com a força do campo de tunelamento aplicada ao sistema.

Em (CAO et al., 2016) é apresentado como fazer uso do arrefecimento quântico para simular e solucionar o problema de cobertura de conjunto por pares (*Set Cover with Pairs*). Também é mostrado como incorporar o referido problema no grafo *chimera* que representa a topologia do hardware do sistema D-Wave. Uma revisão dos conceitos que permeiam a computação quântica adiabática em conjunto com o arrefecimento quântico para solução de problemas de otimização em vidros de spin pode ser vista em (DAS; CHAKRABARTI, 2008).

2.1.2.2 Otimização quântica adiabática

Na literatura, é comum a otimização quântica adiabática consistir em um processo de minimização de uma função objetivo associada a um modelo Ising ou QUBO. Tal ca-

racterística faz com que o estudo de problemas de otimização em um modelo quântico adiabático não fique restrito àqueles problemas que são originalmente de otimização, em um modelo clássico. Sendo assim, nesta seção, qualquer problema a ser resolvido via modelos Ising/QUBO em uma máquina quântica adiabática será considerado um problema de otimização neste contexto. Cabe frisar que a otimização quântica adiabática é realizada através do processo de arrefecimento quântico, o qual foi explicado na Seção 2.1.2.1.

Recentemente, o interesse na computação quântica adiabática tem aumentado devido ao surgimento de supostos computadores quânticos adiabáticos projetados para resolver problemas de computação modelados como vidros de *spin* Ising ou como um problema QUBO. De acordo com (LUCAS, 2014), esta abordagem pode ser realizada ao codificar a solução do problema em questão em um estado H_P do Hamiltoniano. Aliado a este estado, teríamos um outro H_0 que, tipicamente, é uma sobreposição de todos os estados possíveis para o problema. A questão aqui seria evoluir o Hamiltoniano a partir de H_0 adiabaticamente - com base no que foi explicitado na Seção 2.1.2, durante um certo tempo T, suficientemente grande para encontrar a solução do problema. A conexão da otimização quântica adiabática com o modelo Ising surge quando é necessário modelar H_P como uma versão quântica de um vidro de *spin* Ising.

Com relação às propostas de solução de problemas de otimização fazendo uso da computação quântica adiabática, tem-se alguns exemplos de trabalhos que abordam esta perspectiva. Em (BIAN et al., 2014) são discutidas técnicas para resolver problemas de otimização fazendo uso do arrefecimento quântico. Para isso, são investigadas questões inerentes a este modelo, tais como precisão limitada, temperatura e conectividade dos qubits. Ainda, são propostos algoritmos para mapear e decompor problemas grandes no modelo Ising e, posteriormente, em hardware quântico. Os métodos apresentados foram validados e tiveram seus resultados extraídos a partir da execução em uma máquina D-Wave.

Uma aplicação do mapeamento de problemas de otimização para hardware quântico com arrefecimento é mostrada em (BIAN et al., 2016), onde são explicitados algoritmos de mapeamento e imersão em hardware para o problema de diagnóstico de falha. São expostos, aqui, os métodos para imersão do problema em hardware quântico, possibilidades de pré-processamento e o mapeamento das restrições do problema no modelo Ising, juntamente com algoritmos de decomposição para problemas grandes, que incluem abordagens como “divisão e conquista”, por exemplo. Por fim, assim como no trabalho citado no item anterior, os testes para o problema de diagnóstico de falhas foram realizados em uma máquina D-Wave.

O artigo (MCGEOCH; WANG, 2013) trata de uma investigação experimental acerca do uso do arrefecimento quântico para resolver problemas de otimização. Para tanto, foram feitas comparações, a partir de instâncias de problemas NP-difíceis, com 3 *solvers* clássicos: CPLEX, METSlib e Akmaxsat. É argumentado que os resultados apresentados

referem-se, tão somente, ao desempenho de implementações específicas para conjuntos de entrada específicos, não podendo ser parâmetro para desempenho geral dos métodos estudados. Novamente, os testes com relação aos aspectos quânticos foram realizados em uma máquina D-Wave e mostraram o arrefecimento quântico alcançando ou melhorando os resultados das melhores soluções dos *solvers* clássicos.

Em (BAPST et al., 2013) é apresentado um longo estudo que se propõe a comparar, de forma eficiente, o desempenho da otimização quântica adiabática frente às soluções dadas por computadores clássicos. Inicialmente são fornecidas as definições que permeiam a computação quântica adiabática para, então, munir o leitor com variados exemplos clássicos de problemas de otimização. Com isso, o autor desenvolve o trabalho expondo métodos clássicos para resolver problemas de otimização e seus análogos na computação quântica adiabática, na perspectiva da solução de problemas formulados no modelo Ising. Os resultados mostraram que o uso do arrefecimento quântico da computação adiabática pode não ser muito interessante quando da busca por soluções ótimas. Mas, considerando um cenário onde boas soluções aproximadas são aceitas, este paradigma se torna promissor.

O trabalho exposto em (CROSSON et al., 2014) trata da solução para o problema MAX 2-SAT de tamanho $n = 20$ variáveis, distribuídas dentre 60 cláusulas de instâncias geradas aleatoriamente. A ideia consiste em buscar por uma atribuição de valores às variáveis tal que o número de cláusulas satisfeitas seja maximizado. Para este trabalho, o autor empregou algumas estratégias visando o aumento da probabilidade de sucesso do algoritmo adiabático, que são: diminuição do tempo da evolução, inicialização do sistema em estados excitados e a inclusão de um operador Hamiltoniano aleatório no decorrer da evolução. Para cada uma dessas estratégias, foram observadas as mudanças com relação à probabilidade de sucesso. Os resultados demonstraram que todas as 3 estratégias empregadas aumentaram a probabilidade de sucesso do algoritmo, indicando que novas estratégias devem ser utilizadas a fim de que a computação quântica adiabática se torne um paradigma atrativo para a solução desse tipo de problema.

Um processo que deve ser levado em consideração na otimização quântica adiabática é o *minor embedding* (CHOI, 2011), que consiste em uma técnica baseada em grafos para incorporar uma instância do problema no formato Ising ou QUBO na arquitetura de destino. Este procedimento mapeia relacionamentos entre qubits lógicos em qubits físicos. Dessa forma, analisar a eficiência da computação quântica adiabática para a solução de problemas de otimização envolve, também, a aplicação de um algoritmo eficiente para encontrar boas incorporações. Porém, é importante notar que encontrar uma incorporação perfeita para um grafo arbitrário é NP-difícil (ROBERTSON; SEYMOUR, 2004).

Em (YANG; DINNEEN, 2016) são fornecidos algumas heurísticas visando realizar esta tarefa em problemas com centenas de vértices, objetivando, com esses algoritmos, minimizar o número de qubits físicos necessários para representar os qubits lógicos do problema. Também em (YANG; DINNEEN, 2016) podem ser encontrados mais detalhes sobre a defi-

nição de um *minor embedding* e as etapas necessárias para se resolver um problema no formato QUBO/Ising com a computação quântica adiabática implementada nas máquinas D-Wave. Outras particularidades sobre como realizar, na prática, o *minor embedding* podem ser encontradas em (CHOI, 2008) e (CHOI, 2011).

2.2 Modelo Ising e formato QUBO

No modelo Ising o objetivo é minimizar a energia total de vidros de spin Ising através da atribuição de valores a estes por meio do uso de Hamiltonianos Ising com campo magnético transverso (sub-classe de Hamiltonianos). Em (BIAN et al., 2010) tem-se a seguinte formulação de um modelo Ising:

$$E(s) = \sum_{(i,j)\in neigh} J_{i,j} s_i s_j + \sum_{i\in V} h_i s_i$$

onde $s_i \in -1, +1$ representa o spin na i-ésima molécula, V representa o conjunto de todas as moléculas, $|V|$ é o número de moléculas, $s = [s1, ..., s_{|V|}]$ é uma coleção de spins, h_i é a força do campo aplicado à molécula i e $J_{i,j}$ age como campo de interação entre os spins vizinhos i e j e J é uma matriz $|V| \times |V|$. Já a otimização binária quadrática irrestrita (QUBO) trata-se de um problema de otimização NP-difícil (WANG; KLEINBERG, 2009) que objetiva minimizar uma função quadrática binária. De acordo com o exposto em (LUCAS, 2014), a versão quântica do Hamiltoniano de um modelo Ising clássico de N *spins* que codifica a solução para um dado problema P é:

$$H_P = H(\sigma_1^z, ..., \sigma_N^z)$$

onde σ_i^z é uma matriz de Pauli agindo no $i-$ésimo qubit, e J_{ij} e h_i são números reais.

Definição: Seja f uma função objetivo quadrática a ser minimizada, esta é representada como uma matriz triangular superior Q de dimensão $n \times n$, onde n é o número de variáveis do problema. Aqui, f tem a forma $x^T Q x$ e x é um vetor binário de tamanho n constituído das variáveis binárias. Assim, minimizar f significa encontrar uma atribuição de valores para as variáveis de x tal que $f(x)$ seja minimizada. Logo, o problema QUBO tem a seguinte forma:

$$min_x \sum_{i\leq j} x_i Q_{i,j} x_j$$

onde $x_i \in \mathbb{Z}_2$.

Igualmente ao formato QUBO, o modelo Ising busca pela minimização de uma função através da atribuição de valores a um dado vetor x. A diferença aqui consiste na não-binariedade das variáveis envolvidas no problema e na presença de variáveis lineares (além daquelas quadráticas, encontradas nos problemas QUBO). Devido à notável semelhança entre as funções objetivos de ambos os modelos, é natural buscar por formas de

relacioná-los. Dessa maneira, uma instância do problema QUBO pode ser, convenientemente, transformada em uma instância de um problema no modelo Ising de tal modo que $y_i = 2x_i - 1$, para todo $0 \leq i < n$, onde y_i e x_i são variáveis dos modelos Ising e QUBO, respectivamente.

No trabalho apresentado em (BEASLEY, 1998) tem-se a apresentação de dois algoritmos heurísticos para o problema de programação quadrática binária irrestrita UBQP - *Unconstrained Binary Quadratic Programming* baseados na busca tabu e no arrefecimento simulado. Nota-se, aqui, que, para a maioria dos problemas, a busca tabu apresenta um desempenho superior ao arrefecimento simulado; porém, para problemas muito grandes, tem-se o contrário. Um estudo introdutório mais detalhado ao modelo Ising por ser encontrado em (CIPRA, 1987). Para mapeamento de problemas de satisfatibilidade boolenana no modelo Ising do arrefecimento quântico, recomenda-se o trabalho exposto em (SU; TU; HE, 2016).

Em (LUCAS, 2014) são propostas formulações Ising para os 21 problemas NP-completos de Karp, incluindo problemas de particionamento, de cobertura, de coloração e ciclos Hamiltonianos, por exemplo. Em cada caso, o número de spins exigido é, no máximo, cúbico com relação ao tamanho do problema. Essas formulações podem ser úteis na concepção de algoritmos de otimização quântica adiabática.

Em (HUA, 2016), são apresentados métodos para construir eficientemente formulações QUBO para o problema de isomorfismo de grafo (um problema que não se sabe se é resolvível em tempo polinomial ou se é NP-completo). A formulação direta apresentada neste trabalho tem por objetivo mostrar que esta é mais prática que outras no que diz respeito à execução na arquitetura D-Wave. O artigo ainda apresenta uma formulação do Problema de Isomorfismo de Grafo em um Problema de Otimização de Programação Inteira, exibindo sua forma padrão e as restrições para o problema. A partir disso, é feita a conversão do Problema de Programação Inteira para um com apenas restrições binárias lineares. O próximo passo é construir uma matriz QUBO a partir da formulação do Problema de Programação Inteira. Posteriormente, é dada uma formulação QUBO direta. E, por último, é dada uma formulação alternativa baseada na redução do Problema de Isomorfismo de Grafo para o Problema do Clique.

2.3 Qbsolv

O *qbsolv* é uma ferramenta que foi desenvolvida para resolver grandes problemas na forma QUBO ao particioná-los em subproblemas, para execução em um sistema D-Wave (BOOTH; REINHARDT; ROY, 2017). O formato QUBO funciona como uma representação intermediária do problema. Cabe pontuar que, uma vez que o *qbsolv* utiliza de uma abordagem híbrida clássica-quântica, não é todo processamento que faz uso das propriedades do arrefecimento quântico. Sobre isto, pode-se fazer uso do arrefecimento quântico para solução

dos sub-problemas ou, ainda, da busca tabu, que consiste em uma heurística de busca local cujo funcionamento se dá através de um operador que, uma vez encontrada uma solução inicial, gera um certo número de outras possíveis soluções (vizinhança) que podem ser factíveis ou não. Assim, destas novas soluções, uma é escolhida como sendo a melhor e o processo se repete. Para evitar ciclos, uma lista tabu é utilizada para "guardar" os movimentos previamente realizados e é atualizada a cada passo do algoritmo (BEASLEY, 1998).

Sobre seu algoritmo, ele itera através de tentativas, com cada tentativa consistindo de um conjunto de chamadas ao *solver* subQUBO (*solver* que soluciona frações da instância QUBO original), idealmente o sistema D-Wave, para minimização global e uma chamada à busca tabu para minimização local. A natureza híbrida deste algoritmo explora as qualidades do *solver* D-Wave e da busca tabu. O arrefecimento quântico do sistema D-Wave explora diversas regiões do espaço de estados, mas sua precisão limitada restringe sua habilidade de atingir um valor mínimo exato rapidamente. Em contraste, a busca tabu pode, rapidamente, encontrar um mínimo exato dentro de uma vizinhança mas, às vezes, se esforça para escapar dessa vizinhança. De forma alternativa, o algoritmo pode ser visto como uma busca local de vizinhança grande, com melhorias tabu depois de cada iteração. No Algoritmo 1 (BOOTH; REINHARDT; ROY, 2017) está representado o algoritmo executado pela ferramenta.

Detalhes do algoritmo:

- A estrutura de dados-chave mantida é o vetor **index**, que ordena as variáveis por diminuição do impacto sobre o valor da solução
- O **impacto** de uma variável é o aumento no valor da função objetivo (ou seja, menos ótimo) que ocorre quando a variável é negada na solução atual
- Assumindo que a solução atual é um mínimo local, o valor não irá diminuir
- Espera-se que variáveis de maior impacto sejam mais fortemente determinadas
- Depois de encontrar um valor inicial (**best_solution**), solução (**solution**) e **index**, o loop principal repete as duas fases da computação
- A primeira fase divide a instância QUBO de entrada em subQUBOs que se ajustem a um dado sistema para, então, serem executadas. Cada subQUBO é solucionado individualmente e independentemente dos demais subQUBOs
- A constante **fraction** denota a fração da instância QUBO que é particionada em subQUBOs; tipicamente está num intervalo entre 0.05 e 0.15
- As variáveis de cada subQUBO são contíguas no vetor **index**

Algorithm 1 Partitioning algorithm implemented by `qbsolv`

Input: `QUBO` instance

```
# best_energy is the lowest value found to date
# best_solution is the solution bit vector corresponding to the lowest value so far
# index is the indices of the bits in the solution, sorted from most to least impact
# on value

# Get initial estimate of minimum value and backbone
solution ← random 0/1 vector
(best_energy, best_solution) ← TabuSearch(QUBO, solution)
index ← OrderByImpact(QUBO, best_solution)
passCount ← 0
solution ← best_solution

while passCount<numRepeats do
  change ← false
  for i = 0; i < fraction * Size(QUBO);i += subQUBOSize do
    # select subQUBO with other variables clamped
    sub_index ← i : i + subQUBOSize - 1
    subQUBO ← Clamp(QUBO, solution, index[sub_index])
    (sub_energy, sub_solution) ← DWaveSearch(subQUBO)
    # project onto full solution
    if (solution[sub_index] ≠ sub_solution) then
      solution[sub_index] ← sub_solution
      change ← true
    end if
  end for

  if not change then
    Randomize (solution[0 : i - 1])
  end if
  (energy, solution) ← TabuSearch(QUBO, solution)
  if energy < best_energy then
    best_energy ← energy
    best_solution ← solution
    passCount ← 0
  else
    passCount ++
  end if
  index ← OrderByImpact(QUBO, solution)
end while
```

Output: best_energy, best_solution

- Para otimizar sobre um conjunto de variáveis **S**, consideramos a instância QUBO nas variáveis **S** que resultam da fixação das variáveis que não estão em **S** aos seus valores na melhor solução atual

- Se a solução atual é x^*, então o novo subQUBO sobre **S** é

$$f_S(x_S) = \sum_{i \in S} (Q_{ii} + d_i) x_i + \sum_{\substack{i,j \in S \\ i<j}} Q_{ij} x_i x_j$$

  onde d_i é a contribuição para o termo linear em x_i dada pelas variáveis que estão grampeadas:

$$d_i = \sum_{j \notin S} (Q_{ij} + Q_{ji}) x_j^*$$

Dentre as motivações e vantagens do uso do *qbsolv* estão:

- Um problema estruturado para ser executado nesta ferramenta já encontra-se no formato adequado para execução nos sistemas D-Wave. Dessa forma, basta apenas selecionar o devido *solver* quântico dentro do *qbsolv* para que o problema será solucionado fazendo uso das propriedades quânticas presentes em tais sistemas

- Um dos objetivos claros do lançamento *open source* do *qbsolv* é o de estabelecer o QUBO como formato padrão para interfaces de otimização de alto nível. Além deste, tem-se a pretensão de, com o lançamento desta ferramenta, estimular o aumento dos trabalhos relacionados ao desenvolvimento de ferramentas e técnicas para mapear problemas reais no formato QUBO. Portanto, apesar dos resultados encorajadores, estes são vistos mais como uma validação da abordagem de criação de um *solver* geral que use o formato QUBO do que esta implementação em particular

- O problema QUBO é usado como uma representação intermediária útil para o problema que se deseja resolver com o arrefecimento quântico presente do D-Wave, isso porque trata-se de um conceito que corresponde de maneira muito próxima ao formato de hardware do D-Wave. Além disso, diversos problemas do mundo real podem ser mapeados dessa forma. Porém, devido ao tamanho que os problemas reais podem apresentar, por vezes não é possível mapear tais nos sistemas D-Wave. Com isso, a solução se dá através da divisão da instância do problema no formato QUBO em partições denominadas subQUBOs, as quais são resolvidas (por um solver clássico - busca tabu - ou com o arrefecimento quântico) e, posteriormente, tem seus resultados combinados para obtenção da solução do problema original

- Uma dada instância de entrada na forma QUBO precisa ser particionada e mapeada para um formato restrito, conhecido como *grafo Chimera*

O trabalho realizado por (ROSENBERG et al., 2016) apresenta objetivo semelhante ao *qbsolv*: pretende resolver subQUBOs em hardware de arrefecimento quântico quando este apresente vantagem. Uma diferença reside na implementação: enquanto (ROSENBERG et al., 2016) usa Python, o *qbsolv* é implementado em C.

Tendo em vista de seu recente lançamento, a ferramenta *qbsolv* tem sido empregada em um número ainda reduzido de trabalhos. Cabe ressaltar que, junto com seu lançamento, também foi disponibilizado um relatório técnico (ver (BOOTH; REINHARDT; ROY, 2017)). Tal relatório constitui documento de fundamental importância para aqueles que desejam fazer do *qbsolv* um intermediário entre os problemas de otimização na forma QUBO e a execução em plataformas quânticas - em específico, os computadores da D-Wave. Nele, serão encontrados detalhes acerca do algoritmo implementado e o desempenho observado em comparação com resultados estabelecidos na literatura. Por fim, apesar de objetivar a resolução de problemas de otimização fazendo uso das capacidades quânticas dos sistemas D-Wave, o *qbsolv* também pode ser utilizado como *solver* puramente clássico. Assim, a ferramenta pode ser útil tanto para resolução de problemas, como para validação de abordagens que visam ser executadas em sistemas quânticos através de formulações QUBO.

Em (USHIJIMA-MWESIGWA; NEGRE; MNISZEWSKI, 2017) é mostrado o uso e o desempenho do *qbsolv* para o problema de particionamento de grafos. A motivação deste referido trabalho se deu pela proposta de uma teoria de estrutura eletrônica baseada em grafos aplicada à simulações da dinâmica molecular quântica. Algo notado pelo autor foi que, para problemas envolvendo pequenos grafos, a execução direta em um sistema D-Wave seria adequada. Porém, em se tratando de problemas reais, os grafos alcançam milhares de vértices, impossibilitando sua aplicação direta em tais sistemas devido à limitação de *qubits*. A solução encontrada foi fazer uso da abordagem híbrida do *qbsolv* para grafos de tal magnitude. Os resultados obtidos foram comparados com os seguintes *frameworks* para particionamento de grafos já estabelecidos na literatura: METIS (KARYPIS; KUMAR, 1998) e KaHIP (SANDERS; SCHULZ, 2013). Foi observado que a execução do *qbsolv* demonstrou melhor desempenho quando comparado com os *frameworks* citados, igualando ou ficando muito próximo das melhores soluções já encontradas para o problema.

Já em (NEUKART et al., 2017), as características do arrefecimento quântico são utilizadas para resolver o problema de otimização de fluxo de tráfego. A escolha pelo *qbsolv* como ferramenta de experimentos se deu devido à pequena quantidade de *qubits* e a conectividade limitada presente no hardware atual das máquinas D-Wave. Os resultados demonstraram que o uso do *qbsolv* foi satisfatório em resolver o problema, criando um mapa de rotas cujo congestionamento foi reduzido de maneira significativa. Aqui, não foram levados em consideração os tempos de execução, uma vez que o objetivo era criar uma nova formulação para o problema e mostrar sua viabilidade em termos de qualidade. Apesar dos bons resultados, alguns parâmetros relacionados ao problema não foram pon-

derados, algo que o autor coloca como trabalho futuro.

2.3.1 O formato .qubo

O formato .qubo, como descrito em (BOOTH; REINHARDT; ROY, 2017), tem as seguintes características:

- O formato do arquivo de entrada é modelado sobre o formato de arquivo de Satisfatibilidade **DIMACS** e contém quatro tipos de linhas:
  - Comentários: denotados por um "**c**" na primeira coluna
  - Programa: denotado por um "**p**" na primeira coluna, uma linha única de programa por arquivo precisa ser a primeira linha não comentada no arquivo, e precisa conter os seguintes campos separados por espaço, na seguinte ordem:
    - **p**: a sentinela problema-linha
    - **qubo**: tipo de arquivo - **qubo** é o único valor atualmente suportado
    - **target**: topologia - atualmente suporta valores incluindo 0 ou **unconstrained**
    - **maxDiagonals**: o número de elementos diagonais no problema
    - **nDiagonals**: o número de elementos diagonais não-nulos no problema
    - **nElements**: o número de elementos não-nulos fora da diagonal no problema
  - Diagonal: Q_{ii} - uma tripla consistindo do número variável em ambas as posições primeira e segunda e o peso da variável, que pode ser qualquer inteiro ou valor flutuante, na terceira posição. Números variáveis, para ambas as linhas **Diagonal** e **Element**, precisam estar entre 0 e **maxDiagonals - 1** (inclusive)
  - Element: o Q_{ij} - uma tripla consistindo dos números de variável distinta das duas variáveis que o elemento conecta na primeira e segunda posições e a força da conexão, que pode ser um inteiro ou valor flutuante (com exceção do 0), na terceira posição
- O número de linhas **Diagonal** precisa ser igual ao valor **nDiagonals**; e o número de linhas **Element** precisa ser igual ao valor **nElements**. Enquanto, convencionalmente, todas as linhas **Diagonal** aparecem antes de qualquer linha **Element**, linhas **Diagonal** e **Element** podem estar livremente intercaladas no arquivo

2.4 Contact Map Overlap

Encontrar similaridades entre estruturas de proteínas é uma tarefa de fundamental importância em biologia molecular. Isto é feito com o objetivo de encontrar funções equivalentes

entre proteínas e classificá-las de acordo com alguma estrutura, por exemplo. Contudo, determinar qual parâmetro de similaridade deve ser usado não é algo trivial. Aqui, focaremos nossa atenção na abordagem que analisa mapas de contato de proteínas. O problema em questão é chamado de CMO (GODZIK; SKOLNICK, 1994), um problema NP-difícil (GOLDMAN; ISTRAIL; PAPADIMITRIOU, 1999) que tem sido extensivamente estudado e que possui notável relevância em bioinformática.

O mapa de contato de uma proteína é uma representação derivada da estrutura terciária tridimensional de uma proteína. Este mapa pode ser representado como um grafo onde os vértices são equivalentes aos resíduos (aminoácidos) das proteínas e as arestas são contatos entre dois resíduos. Este contato entre resíduos é determinado pela distância entre eles em um espaço tridimensional, a qual deve ser menos que um dado limiar - tipicamente, 7Å(ângstrons). É importante notar que o CMO também é usado como métrica para classificação de proteínas, como exposto em (ANDONOV et al., 2015) e (WOHLERS et al., 2014). Dessa forma, o problema CMO é assim definido:

Definição: Dados dois mapas de contato representados como grafos $G_A = (V_A, E_A)$ e $G_B = (V_B, E_B)$, o problema CMO busca por dois subconjuntos $S_A \subseteq V_A$ e $S_B \subseteq V_B$ com $|S_A| = |S_B|$ e uma bijeção f que preserva a ordem entre S_A and S_B, tal que a cardinalidade do conjunto $\mathcal{L}(S_A, S_B, f) = \{(u, v) \in E_A : u, v \in S_A, (f(u), f(v)) \in E_B\}$ seja maximizada (SILVA et al., 2014).

Na Figura 4 (CAPRARA et al., 2004) pode-se ver o perfeito alinhamento entre os mapas de contato de duas proteínas. Tal alinhamento gera um mapeamento entre os subconjuntos de resíduos (vértices) de ambos os mapas de contato (grafos) de tal maneira que a quantidade de contatos (arestas) presente nesse mapeamento é máxima e existe a preservação de ordem entre os resíduos.

A preservação de ordem mencionada na definição acima (também chamada de *propriedade de não-cruzamento*) faz referência à suposição de que o conjunto dos resíduos do mapa de contato está ordenado. Portanto, o problema CMO procura por um mapeamento um-pra-um entre S_A e S_B de tal modo que, para todo $u, v \in S_A$ e $(f(u), f(v)) \in E_B$, se $u < f(u)$ então $v < f(v)$ - o oposto também se aplica. É equivalente, para este problema, falar que procura-se por um número máximo de contatos em comum. A Figura 5 (ANDONOV; MALOD-DOGNIN; YANEV, 2011) mostra um pequeno exemplo do problema CMO e sua diferença entre o problema relacionado de grafos MCES (*Maximum Common Edge Subgraph*) onde, neste último, não há a exigência da preservação de ordem.

Uma vez que a definição do problema foi dada, serão expostos a seguir alguns trabalhos que exploram diferentes abordagens para solucionar o problema CMO. Dentre estes, tem-se a aplicação dos seguintes métodos: heurísticas, algoritmos de aproximação e algoritmos exatos.

Em (CAPRARA et al., 2004) são apresentados dois métodos baseados em *programação linear inteira* para resolver o CMO, *branch-and-cut* e *relaxação lagrangiana*. Estes, apesar

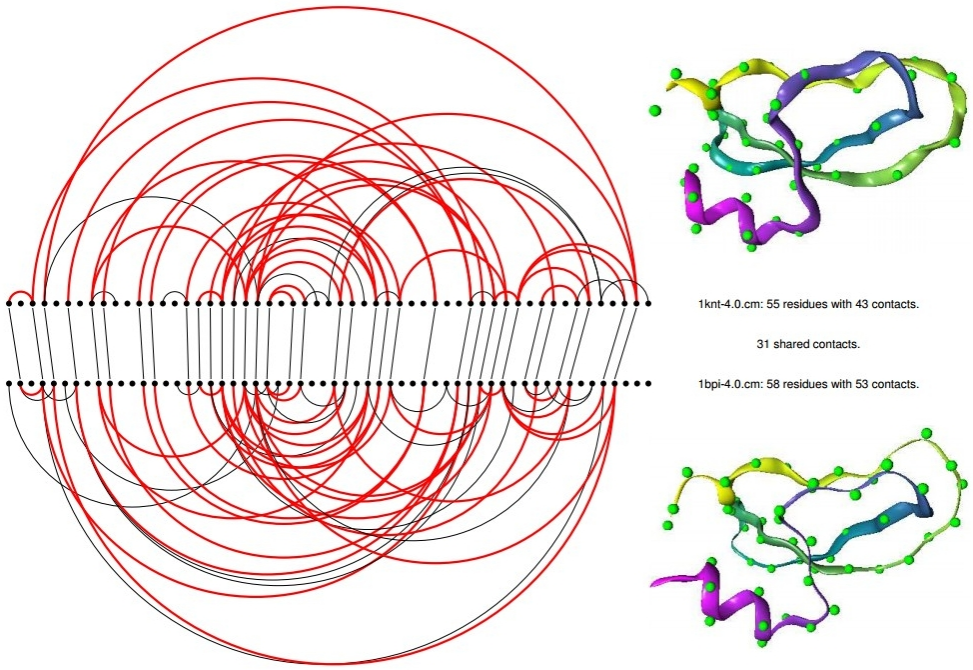


Figura 4 – Alinhamento entre duas proteínas para solução do problema CMO

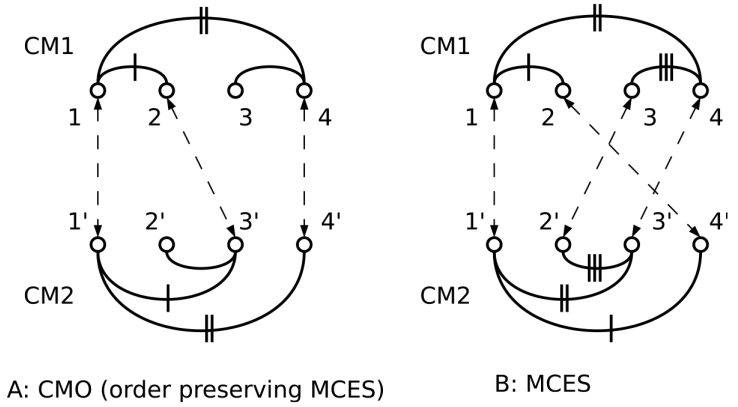


Figura 5 – Diferença entre os problemas CMO e MCES

de não garantirem resultados exatos, retornam soluções próximas das ótimas. Além destes método, são empregadas heurísticas de busca local e algoritmos genéticos como exemplos adicionais de tratamentos dados ao problema CMO. Como uma das formas de resolver o CMO, o autor o reduz para o problema MIS - *Maximum Independent Set* e mostra que seu algoritmo consegue encontrar soluções mesmo para instâncias com 10000 vértices. Cabe ressaltar que o autor não utiliza o CMO apenas para comparar as estruturas das proteínas mas, também, para determinar o quão similar elas são entre si. Os resultados obtidos aqui foram considerados bons, uma vez que, para boa parte dos pares de proteínas envolvidos nos experimentos, foi encontrada uma solução ótima (ou muito próxima desta). Destes resultados, foi verificado que a relaxação lagrangiana obtém melhores resultados e em menor tempo, quando comparada ao algoritmo de *branch-and-cut*. Com relação às

heurísticas, o algoritmo genético foi claramente superior, encontrando, para a maioria das instâncias testadas, soluções ótimas.

No trabalho exposto em (KRASNOGOR et al., 2005) tem-se a comparação de três algoritmos que resolvem o problema de maximizar o CMO entre mapas de contatos de proteínas: (1) a abordagem que integra buscas locais e evolucionárias, utilizada por (KRASNOGOR, 2004); (2) o algoritmo de relaxação lagrangiana, empregado por (CAPRARA; LANCIA, 2002); (3) o algoritmo LGA - *Local-Global Alignment*, aplicado por (ZEMLA, 2003). Dos resultados obtidos, foi verificado que a relaxação lagrangiana retornou as melhores soluções, seguida pelo algoritmo LGA e, por último, a busca evolucionária de (KRASNOGOR, 2004). Este resultado é o esperado, uma vez que a relaxação lagrangiana proposta por (CAPRARA; LANCIA, 2002) foi estritamente projetada para resolver o problema CMO, enquanto que o algoritmo de (KRASNOGOR, 2004) obteve resultados considerados bons, mesmo que este não tenha sido projetado com o objetivo de lidar com o CMO. Ainda assim, o autor pontua que o uso do algoritmo apresentado em (KRASNOGOR, 2004) pode ser útil devido a este não retornar apenas uma solução, mas um conjunto destas para o mesmo par.

Em (XIE; SAHINIDIS, 2007) é proposto um método exato para lidar com a maximização do CMO. Diferentemente de outras abordagens encontradas na literatura, aqui, o CMO não é reduzido a algum outro problema para ser solucionado. É realizado um estudo matemático das estruturas através de técnicas de otimização combinatória para, então, desenvolver um algoritmo *branch-and-bound* por meio de técnicas de redução. Além disso, tal algoritmo tem sua heurística de inicialização fundamentada na *relaxação lagrangiana*. Para execução dos experimentos, os autores consideraram 4 bancos de dados de proteínas contendo, entre eles, o utilizado nesta dissertação. Para aferimento do desempenho, foram realizadas comparações com outros 5 trabalhos da literatura e foi constatado um desempenho significativamente melhor do algoritmo proposto frente aos demais, tanto em relação ao tempo de execução quanto em relação à quantidade de instância resolvidas de maneira ótima.

Um algoritmo *branch-and-bound* exato obtido através de uma relaxação lagrangiana foi apresentado em (ANDONOV; YANEV; MALOD-DOGNIN, 2008). Tal algoritmo foi testado com a base de dados de proteínas proposta por (GODZIK; SKOLNICK, 1994) e mostrou resultados que superaram os melhores existentes até então, além de resolver algumas instâncias "difíceis" (pares com proteínas muito dissimilares de diferentes famílias) pela primeira vez. Para afirmar tal superioridade (em tempo e em qualidade), o autor confrontou seus resultados com aqueles obtidos por (CAPRARA et al., 2004) e (XIE; SAHINIDIS, 2007).

Por fim, cabe menção a duas publicações relativamente recentes, (ANDONOV; MALOD-DOGNIN; YANEV, 2011) e (WOHLERS et al., 2012). Em (ANDONOV; MALOD-DOGNIN; YANEV, 2011) o CMO é revisitado em face dos diversos métodos heurísticos e exatos que se

propõe a resolver o problema. Ali, é fornecida uma revisão de literatura sobre o assunto bem como uma modelagem matemática para o problema. Também são realizadas comparações entre alguns dos métodos citados, como: relaxação lagrangiana, subgradiente descendente e *branch-and-bound*, além de relacioná-los com outras abordagens propostas na literatura. Por sua vez, o trabalho exposto em (WOHLERS et al., 2012) se concentra em descrever um *web server* criado para realizar tarefas no sentido de comparar alinhamentos de estruturas de proteínas. As medidas de comparação utilizadas foram: CMO, PAUL, DALI e MATRAS.

Sobre o *web server* descrito em (WOHLERS et al., 2012), o mesmo foi utilizado para o download da ferramenta **A_purva**, a qual está integrada ao referido servidor. Tal ferramenta foi utilizada para obtenção dos limites inferior e superior para os pares de proteínas utilizados nos experimentos desta dissertação.

3 FORMULAÇÃO QUBO PARA O PROBLEMA CMO

Aqui, serão explicitados os detalhes referentes à formulação desenvolvida para solucionar o problema CMO caracterizado como uma matriz do problema QUBO. Para facilitar a leitura do texto a partir de agora, o problema aqui apresentado será referido como CMO-QUBO.

3.1 Formulação QUBO

O uso de formulações QUBO para encontrar soluções para problemas de otimização combinatória tem ganhado força com a popularização da computação quântica adiabática. Relembrando o que já foi detalhado na Seção 2.2, a otimização binária quadrática irrestrita é um problema NP-difícil cujo princípio consiste em minimizar uma função quadrática f definida como uma matriz triangular superior Q de dimensão $n \times n$, formalmente definida como:

$$x^* = \min_x \sum_{i \leq j} x_i Q_{(i,j)} x_j$$

onde $x_i \in \mathbb{Z}$, x é um vetor binário de n variáveis booleanas e x^* é o valor mínimo de f.

Dados dois mapas de contato de proteínas representados como grafos, a função objetivo para o problema CMO-QUBO consiste em minimizar a matriz resultante da seguinte formulação, composta de 3 operadores:

$$f(x) = H_{CMOP} = H_A + H_B + H_C$$

Mais especificamente:

$$H_A = A \sum_{0 \leq i < n_1} \left(1 - \sum_{0 \leq i' < n_2, i' \neq i} x_{i,i'}\right)^2 + A \sum_{0 \leq i' < n_2} \left(1 - \sum_{0 \leq i < n_1, i \neq i'} x_{i,i'}\right)^2$$

$$H_B = B \sum_{0 \leq i < n_1} \sum_{0 \leq i' < n_2} \sum_{i < j < n_1} \sum_{0 \leq j' < i'} x_{i,i'} x_{j,j'}$$

$$H_C = -C \sum_{ij \in E_1} \sum_{i'j' \in E_2} x_{i,i'} x_{j,j'}$$

onde são necessárias $n_1 \times n_2$ variáveis binárias, n_k é a cardinalidade do conjunto de vértices de G_k, i, j são vértices de G_1 e i', j' são vértices de G_2, E_k é o conjunto de arestas do grafo G_k, $x_{i,j}$ é o mapeamento do vértice $i \in G_1$ para o vértice $j \in G_2$, A, B e C são constantes positivas.

Cada um dos operadores mostrados acima visa garantir o cumprimento de alguma propriedade ou restrição relativa ao problema. Isto é feito na medida em que cada H é responsável por penalizar mapeamentos que ferem essas restrições. Dessa forma, cabe pontuar as penalidades aplicadas por H_A, H_B e H_C:

- H_A penaliza mapeamentos que ferem a bijeção entre $S_A \subseteq V_A$ e $S_B \subseteq V_B$, os quais contém os vértices que farão parte da solução para o problema. Ou seja, para cada vértice em S_A, ele só poderá ser mapeado para exatamente 1 vértice em S_B. O contrário também se aplica

- H_B penaliza os mapeamentos que ferem a propriedade de não-cruzamento. Assim, para cada mapeamento $x_{r,s}$, todo mapeamento $x_{q,t}$ tal que $r < q$ e $t < s$ será penalizado

- Por sua vez, o operador H_C tem um comportamento um tanto diferente. Ao passo que os outros dois operadores penalizam mapeamentos, este reforça bons mapeamentos. Como o objetivo do problema CMO é encontrar um conjunto $\mathcal{L}(S_A, S_B, f) = \{(u, v) \in E_A : u, v \in S_A, (f(u), f(v)) \in E_B\}$ tal que sua cardinalidade é maximizada, H_C reforça pares de mapeamentos que possuem aresta em comum

Como pode ser visto em (LUCAS, 2014), a existência das constantes positivas A, B e C se dá pois, em muitas formulações, faz-se necessário manter separadas as energias associadas com cada operador. Assim, torna-se útil o uso de tais constantes, uma vez que elas permitem uma escalabilidade de energia individual e facilitam o entendimento conceitual da formulação em questão.

Para determinar os valores de A, B, C e IC (presentes nos Apêndices A e B) foi utilizado um subconjunto das instâncias de forma a representar o conjunto total da base de dados. A partir disto, e tendo um conjunto de configurações em potencial, foi analisado qual destas resultava em um alto desempenho quando da execução no subconjunto de instâncias. Posteriormente, a escolha do uso de tal configuração se deu pela manutenção do desempenho quando exploradas as demais instâncias da base de dados. Ressalto, porém, que esses valores atribuídos não foram resultado de qualquer técnica estatística ou de configuração de parâmetros. Tal análise paramétrica poderá e deverá ser investigada em trabalhos futuros, visando uma correta atribuição de valores.

Como já dito anteriormente, as constantes A, B e C são úteis para permitir uma manipulação individual das energias e penalidades associadas a cada operador. Dessa forma, uma maneira encontrada para atribuir os valores a tais constates foi relacionar estes com o número de resíduos dos mapas de contato de cada par de proteínas. Já o valor de IC é utilizado para determinar, a partir de um dado resíduo da primeira proteína, um percentual de resíduos da segunda proteína para o qual o dado resíduo poderá ser mapeado - tanto para “frente” como para “trás”. Como, para esta dissertação,

foram empregadas apenas pares de proteínas com certa similaridade (pelo menos 77% em relação à quantidade de resíduos das proteínas do par), o valor de IC foi fixado em 10% dos resíduos da segunda proteína.

Com relação à formulação exposta nesta seção, assim como visto em (HUA, 2016) qualquer termo constante é ignorado, pois não afeta a obtenção da solução. Outro ponto a ser observado é que, como as variáveis são binárias, quaisquer variáveis $x_{i,i'}$ podem ser substituídas por $x^2_{i,i'}$ sem que seus valores sejam modificados.

3.1.1 Exemplo

Na Figura 6 são apresentadas duas representações em grafo dos mapas de contato de duas proteínas hipotéticas, apenas para ilustrar a abordagem adotada neste trabalho. Para este exemplo, temos as proteínas A e B e seus mapas de contato representados como G_A e G_B, respectivamente. O grafo G_A possui vértices $V_A = \{0, 1, 2, 3, 4\}$ e arestas E_A ={{0,1},{0,3},{0,4},{1,2},{1,3},{2,4},{3,4}}, já o grafo G_B possui vértices $V_B = \{0, 1, 2, 3\}$ e arestas E_B ={{0,1},{0,2},{1,2},{1,3},{2,3}}. Assim, teríamos $n_A \times n_B = 20$ variáveis para o problema QUBO, que são os mapeamentos possíveis entre os vértices de ambos os grafos, ou seja, $x = (x_{0,0}, x_{0,1}, x_{0,2}, x_{0,3}, x_{1,0}, x_{1,1}, x_{1,2}, x_{1,2}, ..., x_{4,3})$.

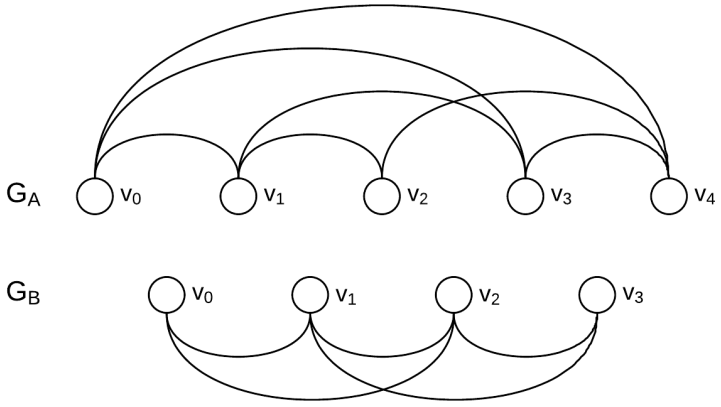


Figura 6 – Exemplo - alinhamento entre duas proteínas hipotéticas

A matriz QUBO (com penalidades $A = B = 5, C = 1$) para o exemplo da Figura 6 está representada na Tabela 2. Já o arquivo .qubo gerado através da execução do código contido no Apêndice C tem a forma exposta na Figura 7.

A minimização da matriz QUBO exposta na Tabela 2 retorna como resposta duas soluções ótimas para este exemplo:

$$x_1 = (0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1)$$

e

```
c .qubo file for execution in qbsolv
p qubo 0 20 0 165
c begin of diagonal elements
c end of diagonal elements
c begin of off-diagonal elements
0 1 10
0 2 10
0 3 10
0 4 10
0 5 -2
...
17 18 10
17 19 10
18 19 10
c end of off-diagonal elements
```

Figura 7 – Exemplo - arquivo .qubo

$$x_2 = (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1).$$

Os vetores-solução acima estão ilustrados, respectivamente, nas Figuras 8 e 9. Tais vetores resultam em um valor de CMO igual a 4.

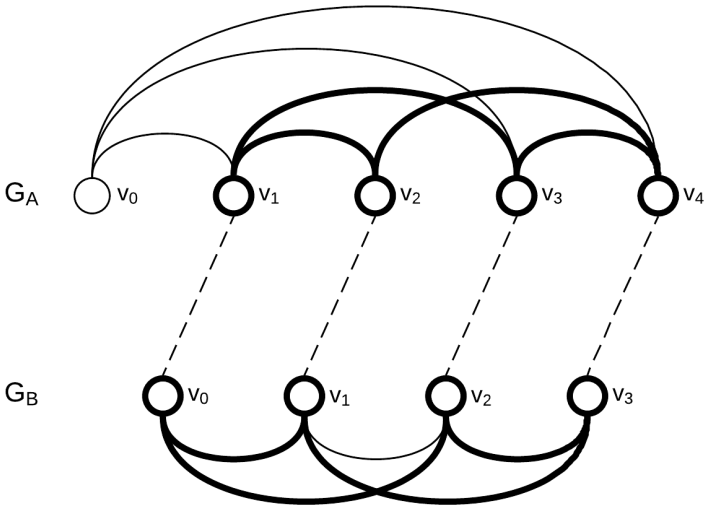


Figura 8 – Exemplo - solução (1). As arestas em negrito representam os contatos a serem considerados para o problema CMO. As linhas tracejadas indicam o mapeamento entre os vértices de G_A e G_B

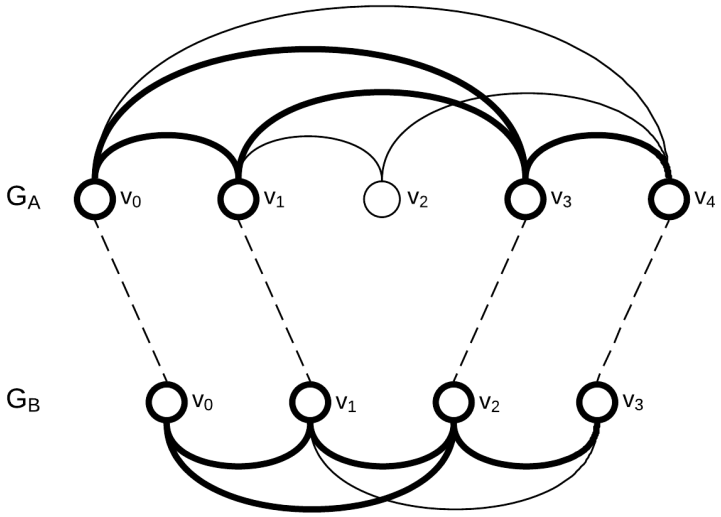


Figura 9 – Exemplo - solução (2). As arestas em negrito representam os contatos a serem considerados para o problema CMO. As linhas tracejadas indicam o mapeamento entre os vértices de G_A e G_B

variáveis	$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{4,0}$	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$
$x_{0,0}$	0	10	10	10	10	-2	-2	0	10	0	0	0	10	-2	-2	0	10	-2	-2	0
$x_{0,1}$	0	0	10	10	10	10	-2	-2	10	10	0	0	10	10	-2	-2	10	10	-2	-2
$x_{0,2}$	0	0	0	10	10	10	10	-2	10	10	10	0	10	10	10	-2	10	10	10	-2
$x_{0,3}$	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
$x_{1,0}$	0	0	0	0	0	10	10	10	10	-2	-2	0	10	-2	-2	0	10	0	0	0
$x_{1,1}$	0	0	0	0	0	0	10	10	10	10	-2	-2	10	10	-2	-2	10	10	0	0
$x_{1,2}$	0	0	0	0	0	0	0	10	10	10	10	-2	10	10	10	-2	10	10	10	0
$x_{1,3}$	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10
$x_{2,0}$	0	0	0	0	0	0	0	0	0	10	10	10	10	0	0	0	10	-2	-2	0
$x_{2,1}$	0	0	0	0	0	0	0	0	0	0	10	10	10	10	0	0	10	10	-2	-2
$x_{2,2}$	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	-2
$x_{2,3}$	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10
$x_{3,0}$	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	-2	-2	0
$x_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	-2	-2
$x_{3,2}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	2
$x_{3,3}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10
$x_{4,0}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10
$x_{4,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10
$x_{4,2}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
$x_{4,3}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 2 – Matriz QUBO

4 EXPERIMENTOS E RESULTADOS

4.1 Detalhes dos experimentos

Nesta seção serão detalhados todas as particularidades referentes aos experimentos executados para validação da formulação apresentada nesta dissertação. Todos os experimentos, tanto os códigos de criação das matrizes QUBO como a execução do *qbsolv*, foram realizados em uma máquina com as seguintes características: arquitetura x64, 8GB de memória RAM DDR3, processador Intel®Core™i7-5500U 2.40GHz, sistema operacional Linux Mint 18.1 Cinnamon. Ressalto, também, que toda a execução das ferramentas **Lagr** e **A_purva** foi realizada na referida máquina, sob as mesmas condições da execução dos códigos da abordagem CMO-QUBO. Uma vez que não obtivemos acesso ao computador quântico, os resultados não tem a pretensão de expor o real poder da computação quântica, mas sim mostrar que uma das abordagens por ela empregada (a de transformar problemas de otimização em problemas QUBO) pode produzir boas soluções.

4.1.1 Base de dados

Para a validação da abordagem apresentada no Capítulo 3 foram selecionados, a partir da base de dados de (GODZIK; SKOLNICK, 1994), 22 proteínas. As mesmas estão descritos na Tabela 3, a qual contém dados sobre a quantidade de resíduos e contatos das proteínas em questão. Durante a fase de experimentos, foi notada uma desigualdade entre os mapas de contato gerados para este trabalho e aqueles gerados internamente pela ferramenta **A_purva**. Esta recebe um arquivo .pdb e, internamente, cria o mapa de contato da proteína. Devido ao código fechado da ferramenta não foi possível atestar o motivo da discordância ao gerar os mapas de contato. Dessa forma, foram mantidos apenas os mapas de contato cujo número de resíduos e proteínas estivessem em concordância para ambas as abordagens CMO-QUBO e **A_purva**.

Os pares para os quais foram realizados os experimentos estão expostos na Tabela 4, onde são mostrados a ID do experimento, as proteínas do par e a similaridade entre elas (baseada na quantidade de resíduos). Como frisado por (CAPRARA et al., 2004), encontrar boas soluções para proteínas consideravelmente diferentes é uma tarefa bastante custosa para os algoritmos e métodos desenvolvidos até então. O texto apresentado em (CAPRARA et al., 2004) não deixa clara a diferença do custo computacional ao se observar proteínas similares e dissimilares. Em vista disso, os pares de mapas de contato empregados nos nossos experimentos são de proteínas substancialmente similares, no que diz respeito tanto à quantidade de resíduos quanto de contatos - não existe um consenso na escolha do quão similares devem ser as proteínas. Cabe pontuar, entretanto, que a escolha dos pares de

Proteína	Resíduos	Contatos	Proteína	Resíduos	Contatos
1b9bA	252	890	1b00A	122	394
1btmA	251	933	1bawA	105	367
1hti	248	861	1byoA	99	342
1treA	255	915	1byoB	99	338
1tri	239	830	1kdi	102	333
1ydvA	246	815	1nat	119	395
3ypiA	247	849	1rn1C	104	320
1b71	191	642	2pcy	99	344
1bcfA	158	524	3chy	128	431
1dpsA	159	513	4tmyA	118	418
1ier	174	595	4tmyB	118	419

Tabela 3 – Proteínas utilizadas nos experimentos

proteínas para este trabalho se deu em virtude da semelhança entre a quantidade de resíduos das mesmas (no mínimo, 77% de similaridade entre a quantidade de resíduos das proteínas do par).

4.1.2 Metodologia

Com base no que foi apresentado no Capítulo 2 e nas seções anteriores do presente capítulo, aqui será abordada a metodologia utilizada para a solução do problema CMO na sua forma QUBO. Uma vez que já foi apresentada sua formulação no Capítulo 3, a seguir serão detalhados os passos e escolhas feitas para possibilitar a execução dos experimentos a serem mostrados neste capítulo.

Inicialmente, foram obtidos os arquivos das proteínas (extensão .pdb) presentes na base de dados apresentada em (GODZIK; SKOLNICK, 1994). Com o auxílio do BioPython (COCK et al., 2009), o mapa de contato de cada proteína foi gerado a partir da cadeia α da estrutura secundária da proteína, com um limiar de 7.5Å(ângstrons) e ignorando resíduos consecutivos. Com o mapa de contato criado, foram elaborados *scripts* na linguagem de programação Python versão 2.7 (Apêndices A e B) responsáveis pela criação das matrizes QUBO a partir da formulação desenvolvida para este trabalho. Cabe mencionar que, para a criação das matrizes QUBO, os mapas de contato foram inicialmente transformados em grafos a partir do pacote NetworkX (HAGBERG; SCHULT; SWART, 2008). O passo final antes dos experimentos em si foi a criação do arquivo com extensão .qubo. O mesmo foi gerado a partir de um outro *script* (Apêndice C), também em Python, com suas devidas especificações e comentários para facilitar a leitura.

Com relação aos experimentos, estes foram executados com o auxílio da ferramenta *qbsolv*, a qual foi detalhada na Seção 2.3. Relembrando, o *qbsolv* recebe como entrada um arquivo .qubo e busca minimizar a função objetivo associada à matriz representada no arquivo. O processo de minimização pode ser realizado tanto em um ambiente clássico

(busca tabu), como em um ambiente quântico (arrefecimento quântico - mais especificamente, o ambiente quântico adiabático dos computadores da D-Wave). Apesar do uso desta ferramenta - criada para experimentos em máquinas D-Wave, um problema na sua forma QUBO já se encontra em um formato adequado para execução em outros dispositivos e ambientes quânticos.

Por fim, o *qbsolv* retorna um arquivo .qubout que contém uma amostragem dos menores valores obtidos para a função objetivo, bem como os vetores-solução que geraram estes valores. Outros detalhes, tais como tempo e tamanho da submatriz, podem ser configurados na execução para serem escritos neste mesmo arquivo de saída. Após isso, restou apenas checar se uma das soluções (ou mais de uma) era factível de acordo com as restrições para solucionar o problema e comparar o resultado com aqueles obtidos através das ferramentas **A_purva** (ANDONOV; YANEV; MALOD-DOGNIN, 2008), um *solver* que utiliza a estratégia *branch-and-bound* para resolver o problema CMO, e **Lagr** (CAPRARA et al., 2004).

Durante a execução dos códigos do Apêndice A, foi notada a necessidade em diminuir a dimensão das matrizes QUBO em virtude do alto tempo de processamento das mesmas, dificultando a dinâmica e ajuste dos experimentos. Além do alto tempo de processamento (cerca de 2 horas para cada par), houve ainda uma limitação da memória RAM disponível para armazenar as referidas matrizes. Devido a este entrave, os resultados expostos na próxima seção foram obtidos com as matrizes geradas através da execução dos códigos expostos no Apêndice B, os quais consideram apenas uma porcentagem dos mapeamentos possíveis. Essa escolha se deu, também, em função das características do problema: a observância da restrição de não-cruzamento citada na Seção 2.4 nos leva a perceber que mapear os resíduos iniciais de uma proteína com os finais de outra impossibilitaria boa parte dos mapeamentos possíveis, retornando, assim, uma solução ruim. Dessa forma, apesar de a formulação utilizada para os experimentos ter sido a mesma em ambos os casos, essa alteração na execução dos experimentos se fez necessária. Cabe pontuar que tal mudança possibilitou despender menos tempo para a realização dos experimentos sem afetar de maneira significativa a qualidade das soluções obtidas. A diferença entre as soluções obtidas através dos códigos dos Apêndices A e B não ultrapassou 10%.

ID	Proteína 1	Proteína 2	Similar.
1	1b9bA	1btmA	0.99
2	1b9bA	1htiA	0.98
3	1b9bA	1treA	0.98
4	1b9bA	1tri	0.94
5	1b9bA	1ydvA	0.97
6	1b9bA	3ypiA	0.98
7	1btmA	1htiA	0.98
8	1btmA	1treA	0.98
9	1btmA	1tri	0.95
10	1btmA	1ydvA	0.98
11	1btmA	3ypiA	0.98
12	1htiA	1treA	0.97
13	1htiA	1tri	0.96
14	1htiA	1ydvA	0.99
15	1htiA	3ypiA	0.99
16	1treA	1tri	0.93
17	1treA	1ydvA	0.96
18	1treA	3ypiA	0.96
19	1tri	1ydvA	0.97
20	1tri	3ypiA	0.96
21	1ydvA	3ypiA	0.99
22	1b71	1bcfA	0.82
23	1b71	1dpsA	0.83
24	1b71	1ier	0.91
25	1bcfA	1dpsA	0.99
26	1bcfA	1ier	0.90
27	1dpsA	1ier	0.91
28	1b00A	1bawA	0.86
29	1b00A	1byoA	0.81
30	1b00A	1byoB	0.81
31	1b00A	1kdi	0.83
32	1b00A	1nat	0.97
33	1b00A	1rn1C	0.85
34	1b00A	2pcy	0.81
35	1b00A	3chy	0.95
36	1b00A	4tmyA	0.96
37	1b00A	4tmyB	0.96
38	1bawA	1byoA	0.94
39	1bawA	1byoB	0.94
40	1bawA	1kdi	0.97
41	1bawA	1nat	0.88
42	1bawA	1rn1C	0.99
43	1bawA	2pcy	0.94
44	1bawA	3chy	0.82
45	1bawA	4tmyA	0.88
46	1bawA	4tmyB	0.88
47	1byoA	1byoB	1
48	1byoA	1kdi	0.97
49	1byoA	1nat	0.83
50	1byoA	1rn1C	0.95
51	1byoA	2pcy	1
52	1byoA	3chy	0.77
53	1byoA	4tmyA	0.83
54	1byoA	4tmyB	0.83
55	1byoB	1kdi	0.97
56	1byoB	1nat	0.83
57	1byoB	1rn1C	0.95
58	1byoB	2pcy	1
59	1byoB	3chy	0.77
60	1byoB	4tmyA	0.83
61	1byoB	4tmyB	0.83
62	1kdi	1nat	0.85
63	1kdi	1rn1C	0.98
64	1kdi	2pcy	0.97
65	1kdi	3chy	0.79
66	1kdi	4tmyA	0.86
67	1kdi	4tmyB	0.86
68	1nat	1rn1C	0.87
69	1nat	2pcy	0.83
70	1nat	3chy	0.92
71	1nat	4tmyA	0.99
72	1nat	4tmyB	0.99
73	1rn1C	2pcy	0.95
74	1rn1C	3chy	0.81
75	1rn1C	4tmyA	0.88
76	1rn1C	4tmyB	0.88
77	2pcy	3chy	0.77
78	2pcy	4tmyA	0.83
79	2pcy	4tmyB	0.83
80	3chy	4tmyA	0.92
81	3chy	4tmyB	0.92
82	4tmyA	4tmyB	1

Tabela 4 – Pares de proteínas utilizados nos experimentos

4.2 Resultados

Os resultados da execução das matrizes QUBO que representam o problema CMO estão expostos na Tabela 5. A coluna **Target** representa o limite superior para um dado par de proteínas e foi obtido com a execução do algoritmo *branch-and-bound* presente na ferramenta **A_purva**.

Nosso primeiro e principal objetivo para este trabalho era mostrar que poderíamos obter soluções relevantes para problemas reais e práticos através das formulações qubo. Isso, claro, visando a execução de tais problemas em um ambiente quântico. Dessa forma, não estávamos necessariamente procurando por soluções ótimas ou melhorar o desempenho de abordagens já bem estabelecidas na literatura. Assim, para avaliar a qualidade e a significância da abordagem aqui empregada, foram feitas comparações entre resultados obtidos com as ferramentas clássicas: **Lagr** (CAPRARA et al., 2004) e **A_purva** (ANDONOV; MALOD-DOGNIN; YANEV, 2011).

Os tempos de execução entre os experimentos da nossa abordagem e aqueles observados no algoritmo **Lagr** são mostrados na Tabela 7, sendo os mesmos expressos em segundos. Necessário frisar que a coluna **Tempo total** representa a soma de **Tempo pré** e **Tempo qbsolv**. Por sua vez, **Tempo pré** corresponde exclusivamente ao tempo de pré-processamento da abordagem CMO-QUBO, ou seja, antes da execução do *solver* propriamente dito. Assim, o pré-processamento inclui: leitura dos arquivos dos mapas de contato, geração dos grafos correspondentes, criação da matriz QUBO e, finalmente, a elaboração do arquivo .qubo.

ID	CMO-QUBO	Lagr	A_purva	Target	Porcentagem
1	839	839	839	839	100
2	788	788	788	788	100
3	810	810	810	810	100
4	754	754	754	754	100
5	768	768	768	768	100
6	766	777	777	777	98.58
7	464	790	790	790	58.73
8	816	818	818	818	99.75
9	744	762	762	762	97.64
10	780	780	780	780	100
11	530	791	791	791	67.00
12	318	786	786	786	40.46
13	748	748	748	748	100
14	781	781	781	781	100
15	790	791	791	791	99.87
16	410	739	739	739	55.48
17	765	772	772	772	99.09
18	671	770	770	770	87.14
19	529	716	716	716	73.88
20	747	747	747	747	100
21	560	754	754	754	74.27
22	399	456	456	456	87.50
23	313	380	380	380	82.37
24	354	469	469	469	75.48
25	344	398	398	398	86.43
26	490	490	490	490	100
27	362	404	404	404	89.60
28	160	171	171	171	93.57
29	147	151	154	154	95.45
30	148	150	155	155	95.48
31	147	160	162	162	90.74
32	338	338	338	338	100
33	143	153	156	159	89.94
34	150	153	158	158	94.94
35	352	352	352	352	100
36	343	343	343	343	100
37	343	343	343	343	100
38	311	311	311	311	100
39	308	308	308	308	100
40	287	287	287	287	100
41	164	169	170	170	96.47

Tabela 5 – Resultados do CMO-QUBO em comparação com o *Target* e com os algoritmos **Lagr** e **A_purva** *(continua...)*

ID	CMO-QUBO	Lagr	A_purva	Target	Porcentagem
42	133	145	145	141	94.33
43	313	313	313	313	100
44	160	176	178	178	89.89
45	165	173	173	173	95.37
46	163	173	173	173	94.22
47	335	335	335	335	100
48	297	297	197	297	100
49	144	153	154	154	93.51
50	128	142	142	142	90.14
51	332	332	332	332	100
52	154	157	157	158	97.47
53	152	158	158	159	95.60
54	148	158	158	159	93.08
55	294	294	294	294	100
56	141	151	153	153	92.16
57	136	140	141	141	96.45
58	331	331	331	331	100
59	151	156	158	158	95.57
60	150	157	157	158	94.94
61	151	157	157	158	95.57
62	145	156	160	160	90.62
63	136	143	145	138	98.55
64	292	292	292	292	100
65	155	164	166	166	93.37
66	153	163	163	164	93.29
67	150	162	162	163	92.02
68	141	154	156	153	92.16
69	150	150	155	155	96.77
70	358	358	358	358	100
71	364	364	364	364	100
72	364	364	364	364	100
73	132	143	144	142	92.96
74	142	162	161	162	87.65
75	139	161	158	158	87.97
76	139	159	158	158	87.97
77	149	159	159	160	93.12
78	155	158	159	160	96.87
79	156	159	159	160	97.50
80	372	372	372	372	100
81	371	371	371	371	100
82	414	414	414	414	100

Tabela 6 – Resultados do CMO-QUBO em comparação com o *Target* e com os algoritmos **Lagr** e **A_purva**

ID	Tempo pré	Tempo qbsolv	Tempo total	Tempo Lagr
1	55.13	285.86	340.99	112.31
2	51.94	274.23	326.17	239.88
3	54.65	372.19	426.84	178.89
4	45.83	255.27	301.10	186.69
5	50.70	247.34	298.04	129.56
6	50.28	262.05	312.33	249.02
7	50.01	361.49	411.50	221.01
8	73.88	259.99	333.87	247.75
9	45.59	244.59	290.18	185.10
10	49.76	271.61	321.37	146.04
11	49.77	344.06	393.83	290.41
12	53.07	348.49	401.56	752.71
13	44.69	230.79	275.48	276.24
14	48.70	256.92	305.62	55.05
15	48.90	248.02	296.92	100.70
16	47.51	406.66	454.17	350.67
17	51.52	241.94	293.46	331.07
18	51.23	404.85	456.08	423.22
19	45.40	316.78	362.18	233.58
20	45.10	226.56	271.66	322.18
21	47.79	290.28	338.07	117.55
22	12.14	119.68	131.82	829.70
23	11.93	132.06	143.99	1116.90
24	15.45	174.50	189.95	1801.59
25	8.41	71.04	79.45	161.96
26	10.80	66.06	76.86	237.02
27	10.86	95.28	106.14	703.90
28	2.41	22.67	25.08	565.96
29	1.96	14.01	15.97	557.28
30	1.97	21.64	23.61	562.27
31	2.44	16.67	19.11	570.86
32	2.84	19.40	22.24	29.10
33	2.48	14.55	17.03	494.60
34	1.96	19.95	21.91	527.99
35	3.45	17.15	20.60	11.07
36	2.87	17.30	20.17	12.75
37	2.86	19.73	22.59	11.59
38	1.54	5.48	7.02	8.15
39	1.54	5.91	7.45	8.60
40	1.88	12.98	14.86	19.47
41	2.17	33.14	35.31	437.96

Tabela 7 – Tempos de execução do CMO-QUBO em comparação com o algoritmo **Lagr**
(continua...)

ID	Tempo pré	Tempo qbsolv	Tempo total	Tempo Lagr
42	1.82	12.96	14.78	357.40
43	1.56	9.00	10.56	7.43
44	2.50	19.81	22.31	488.00
45	2.18	14.41	16.59	441.00
46	2.16	14.02	16.18	439.06
47	1.38	10.36	11.74	0.01
48	1.71	8.73	10.44	7.00
49	1.93	15.35	17.28	412.36
50	1.64	11.00	12.64	316.36
51	1.38	8.19	9.57	0.01
52	2.25	21.69	23.94	465.95
53	1.93	15.15	17.08	405.18
54	1.91	10.56	12.47	405.30
55	1.70	11.12	12.82	9.15
56	1.89	10.57	12.46	410.72
57	1.64	18.90	20.54	316.66
58	1.38	9.18	10.56	0.01
59	2.26	14.32	16.58	470.00
60	1.90	12.44	14.34	406.42
61	1.90	11.81	13.71	404.64
62	2.00	15.18	17.18	432.68
63	1.73	13.61	15.34	343.48
64	1.42	7.99	9.41	9.42
65	2.35	20.97	23.32	478.14
66	2.01	14.77	16.78	423.40
67	2.02	16.77	18.79	424.37
68	2.31	16.58	18.89	484.51
69	1.87	37.63	39.50	514.46
70	3.24	18.69	21.93	17.96
71	2.74	19.86	22.60	4.90
72	2.73	18.27	21.00	5.68
73	1.45	12.45	13.90	354.46
74	2.45	15.07	17.52	510.38
75	2.11	26.45	28.56	453.88
76	2.11	21.34	23.45	459.78
77	2.26	14.67	16.93	460.62
78	1.91	14.24	16.15	402.62
79	1.94	14.07	16.01	404.75
80	3.19	34.20	37.39	7.01
81	3.16	33.66	36.82	8.35
82	2.72	20.77	23.49	0.01

Tabela 8 – Tempos de execução do CMO-QUBO em comparação com o algoritmo **Lagr**

4.3 Discussão

A respeito destes resultados, não foi possível realizar testes em um computador quântico. Foram feitas tentativas de contato com a empresa D-Wave para execução de experimentos em suas máquinas, mas não obtivemos sucesso. Dessa forma, a validação da abordagem foi realizada no *qbsolv*, uma vez que este se prontifica a solucionar problemas na forma qubo. Caso venhamos a obter acesso a uma das máquinas adiabáticas, bastaria configurar o *qbsolv* para fazer uso do solver quântico para minimização da energia associada à matriz qubo do problema. Mas é importante notar que o objetivo aqui não é avaliar o desempenho referente ao tempo gasto na busca por soluções. Computação Quântica ainda é uma área que necessita de extensa pesquisa adicional para alcançar seu completo desenvolvimento. Espera-se que surjam novos algoritmos e técnicas que tirem total proveito das capacidades dos fenômenos quânticos na computação. Ademais, a solução exposta aqui nesta dissertação pode ser vista como sendo nosso primeiro passo nesta direção. Quanto à resolução de problemas no modelo de circuitos, a formulação QUBO apresentada no Capítulo 3 já a qualifica para ser resolvida neste modelo.

Com relação às ferramentas **Lagr** e **A_purva**, as mesmas foram executadas com as configurações padrão e seus códigos foram obtidos diretamente com os respectivos autores. Além disso, devido às características de ambos os algoritmos, foi necessário estipular um critério de parada, sendo escolhido o tempo de 1800s para isso. Apesar da existência de tais algoritmos, uma dificuldade encontrada para a realização dos experimentos se deu em consequência da escassez de detalhes dos resultados de outros métodos da literatura. Alguns dos métodos descritos na Seção 2.4 não possuem tabelas expositivas e nem disponibilizam os códigos desenvolvidos, o que prejudica o confronto de resultados.

A partir dos resultados expostos na Tabela 5, temos ainda os seguintes dados: dos 82 pares analisados, 30 (36.6%) alcançaram um valor perfeito de CMO; 18 (22%) alcançaram um valor entre 95% e 100% do valor ótimo; 17 (20.7%) ficaram entre 90% e 95%; 10 pares (12.2%) atingiram entre 80% e 90% do valor ótimo e apenas 7 pares (8.5%) ficaram obtiveram menos que 80% do valor pretendido. Sendo assim, 65 (79.3%) dos 82 pares obtiveram mais de 90% de acerto, enquanto que 17 (20.7%) obtiveram menos de 90% de acerto em comparação com o valor ótimo - esses dados estão explicitados em forma de gráfico na Figura 10. Do total de contatos envolvidos nos alinhamentos perfeitos entre os mapas de contato das proteínas (31130), a abordagem CMO-QUBO foi capaz de encontrar 28536 contatos, resultando numa porcentagem de acerto geral de 91.67%.

Como já pontuado, o objetivo aqui não é a comparação de desempenho a partir do tempo despendido para encontrar as soluções. Dessa forma, foi realizada apenas a comparação com uma abordagem bem estabelecida na literatura, o algoritmo lagrangiano **Lagr**. No que diz respeito a isto, tem-se os seguintes dados de tempo de execução (para todos os 82 pares):

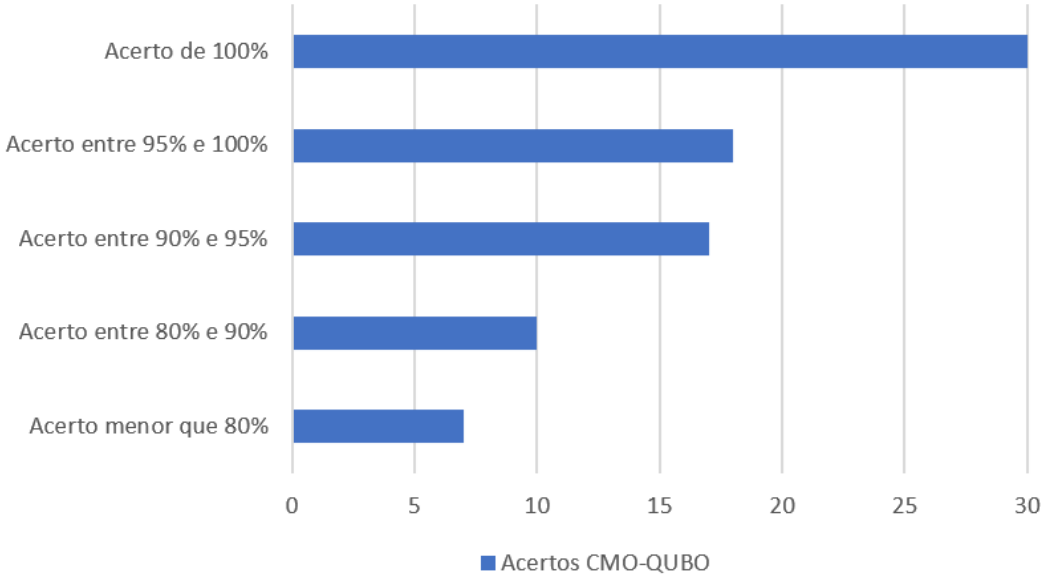


Figura 10 – Gráfico de acertos da abordagem CMO-QUBO em relação ao valor ótimo

- Os tempos referentes a abordagem **CMO-QUBO** são divididos em 2 fases, *pré-processamento* e *qbsolv*:
  - Para a etapa de *pré-processamento* foram despendidos cerca de 20min46s, sendo que 15.2s é o tempo médio para cada par;
  - A execução do *qbsolv* requereu cerca de 2h8min, sendo 1min34s o tempo médio para cada par;
  - O tempo total (*pré-processamento* + *qbsolv*) resultou em 2h29min, tendo o tempo médio de 1min49s para cada par.
- A execução do algoritmo **Lagr** despendeu um total de, aproximadamente, 7h10min, tendo cerca de 5min14s de tempo médio para cada par

É necessário pontuar que as soluções obtidas através da execução dos algoritmos **Lagr** e **A_purva** foram substancialmente melhores do que as da abordagem CMO-QUBO. Porém, os tempos apresentados pelo algoritmo **Lagr**, detalhados acima, foram consideravelmente maiores quando comparados com a nossa abordagem. Os ótimos tempos do **A_purva** e **Lagr** são decorrentes da sua formulação, a qual explora a estrutura matemática do problema CMO. O mesmo não acontece com a resolução do *qbsolv* apresentada nesta dissertação, uma vez que o *qbsolv* não tem conhecimento do problema inicial, apenas lidando com a representação qubo deste. Sendo assim, o CMO-QUBO entrega soluções razoáveis em um tempo melhor. Por consequência disto, entendemos que tal resultado qualifica e valida o uso de formulações QUBO para a solução de problemas específicos de otimização combinatória.

5 CONCLUSÕES E TRABALHOS FUTUROS

Foi dada uma formulação de QUBO para o problema CMO. Embora esta formulação tenha sido validada em um ambiente computacional clássico, deve-se notar que esta já se encontra em um formato que pode ser executado com facilidade em um computador quântico. Os resultados mostram que é válido usar formulações de QUBO para resolver o problema CMO e outros problemas de otimização, onde o objetivo é maximizar/minimizar uma função objetivo de forma combinatória.

Com relação aos resultados expostos no Capítulo 4 deste trabalho de pesquisa, eles indicam que o uso de formulações QUBO para resolução de problemas de otimização consegue alcançar bons resultados quando comparados aos já presentes na literatura. Como já dito, também no Capítulo 4, o objetivo aqui não se concentrava em verificar o desempenho da abordagem na avaliação do tempo de execução, mas sim em fornecer um caminho inicial para resolução de problemas particulares de otimização em um âmbito quântico. Ainda assim, os tempos de execução, tanto do pré-processamento quanto do *qbsolv* foram considerados razoáveis em termos computacionais, se comparados com aqueles observados na ferramenta **Lagr**. Portanto, cabe a análise do emprego de melhores estratégias para o processo de criação das matrizes QUBO e aplicação de pré-processamento nas matrizes QUBO já criadas, a fim de reduzir sua complexidade e dimensão, como em (LEWIS; GLOVER, 2017).

Finalmente, embora (LUCAS, 2014) tenha fornecido várias formulações de QUBO para problemas de computação importantes, essas formulações são genéricas e impraticáveis para problemas mais específicos. Esse fato é o que motiva e qualifica o desenvolvimento de formulações QUBO novas e eficientes com o objetivo de resolver problemas específicos em um computador quântico.

5.1 Considerações sobre otimização quântica

No artigo (MCGEOCH; WANG, 2013) tem-se uma comparação entre um computador quântico desenvolvido pela D-Wave e *solvers* clássicos, como o CPLEX, para a resolução de problemas de otimização combinatória. Apesar dos bons resultados apresentados, cabe ressaltar que uma máquina D-Wave resolve um tipo especial de problemas de otimização, os modelados como problemas QUBO ou no modelo Ising, onde não há restrições para a função objetivo. Por sua vez, o CPLEX não é construído para resolver especificamente esse tipo de problema. Assim, pode-se concluir que os sistemas D-Wave não estão sendo desenhados, até o momento, para competir com *solvers* clássicos como o CPLEX, uma vez que tem se configurado como um *solver* de propósito específico.

Alguns fatores devem ser levados em consideração para aqueles que desejam, no futuro, implementar algoritmos para problemas de otimização em ambientes quânticos. Por exemplo, a dimensão do sistema quântico onde serão executados os algoritmos é condição profundamente limitante, uma vez que a quantidade de *qubits* presente nos mesmos demarca a fronteira de tamanho do problema a ser resolvido. Uma alternativa a isto seria dividir o problema original em problemas menores ou, ainda, formular o problema de tal modo que ele possa ser incorporado no sistema subjacente. Porém, ambas as possibilidades poderiam acarretar um pré-processamento tão custoso que inviabilizaria o uso de tal abordagem.

Com isso em mente, é notado que a área de otimização quântica necessita ainda de uma quantidade substancial de pesquisa. Para que esta abordagem se torne uma alternativa atrativa em termos práticos para resolução de problemas considerados difíceis atualmente, novas técnicas e métodos precisam ser desenvolvidos e aprimorados.

5.2 Trabalhos futuros

O uso de formulações QUBO para resolver problemas em computação quântica, apesar da existência de algumas caracterizações nesse sentido, ainda carece de mais pesquisa para poder explorar os efeitos benéficos deste modelo computacional. Para tanto, ficam como trabalhos futuros e relacionados a esta dissertação:

- Análise de técnicas que resultem na diminuição da dimensão das matrizes QUBO para problemas grande. Um estudo sobre isso pode ser encontrado em (LEWIS; GLOVER, 2017)
- Uso de técnicas para escolhas de parâmetros objetivando ajustar os valores das constantes de penalidade empregadas na formulação CMO-QUBO
- Implementar outros algoritmos de busca para substituir a busca tabu do *qbsolv* e analisar o impacto nas soluções, bem como verificar o desempenho

5.3 Agradecimentos

Por fim, agradeço ao Professor Rumen Andonov, por fornecer os códigos da ferramenta **A_purva**, a qual foi útil para gerar os valores-*target* do problema CMO para as proteínas utilizadas nesta dissertação e ao Professor Giuseppe Lancia, pelos códigos da ferramenta **Lagr**. Agradeço, também, à FACEPE, pelo apoio financeiro.

REFERÊNCIAS

AHARONOV, D.; DAM, W. V.; KEMPE, J.; LANDAU, Z.; LLOYD, S.; REGEV, O. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, SIAM, v. 50, n. 4, p. 755–787, 2008.

ALBASH, T.; LIDAR, D. A. Adiabatic quantum computation. *Rev. Mod. Phys.*, American Physical Society, v. 90, p. 015002, Jan 2018. Disponível em: <https://link.aps.org/doi/10.1103/RevModPhys.90.015002>.

AMBAINIS, A. Quantum search algorithms. *ACM SIGACT News*, ACM, v. 35, n. 2, p. 22–35, 2004.

ANDONOV, R.; DJIDJEV, H.; KLAU, G. W.; BOUDIC-JAMIN, M. L.; WOHLERS, I. Automatic classification of protein structure using the maximum contact map overlap metric. *Algorithms*, Multidisciplinary Digital Publishing Institute, v. 8, n. 4, p. 850–869, 2015.

ANDONOV, R.; MALOD-DOGNIN, N.; YANEV, N. Maximum contact map overlap revisited. *Journal of Computational Biology*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 18, n. 1, p. 27–41, 2011.

ANDONOV, R.; YANEV, N.; MALOD-DOGNIN, N. An efficient lagrangian relaxation for the contact map overlap problem. *Algorithms in Bioinformatics*, Springer, p. 162–173, 2008.

BAPST, V.; FOINI, L.; KRZAKALA, F.; SEMERJIAN, G.; ZAMPONI, F. The quantum adiabatic algorithm applied to random optimization problems: The quantum spin glass perspective. *Physics Reports*, Elsevier, v. 523, n. 3, p. 127–205, 2013.

BARENDS, R.; SHABANI, A.; LAMATA, L.; KELLY, J.; MEZZACAPO, A.; HERAS, U. L.; BABBUSH, R.; FOWLER, A. G.; CAMPBELL, B.; CHEN, Y. et al. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, Nature Publishing Group, v. 534, n. 7606, p. 222, 2016.

BARITOMPA, W. P.; BULGER, D. W.; WOOD, G. R. Grover's quantum algorithm applied to global optimization. *SIAM Journal on Optimization*, SIAM, v. 15, n. 4, p. 1170–1184, 2005.

BEASLEY, J. E. Heuristic algorithms for the unconstrained binary quadratic programming problem. *London, UK: Management School, Imperial College*, v. 4, 1998.

BIAN, Z.; CHUDAK, F.; ISRAEL, R.; LACKEY, B.; MACREADY, W. G.; ROY, A. Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, Frontiers, v. 2, p. 56, 2014.

BIAN, Z.; CHUDAK, F.; ISRAEL, R.; LACKEY, B.; MACREADY, W. G.; ROY, A. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *arXiv preprint arXiv:1603.03111*, 2016.

BIAN, Z.; CHUDAK, F.; MACREADY, W. G.; ROSE, G. The ising model: teaching an old problem new tricks. *D-Wave Systems*, v. 2, 2010.

BOIXO, S.; ALBASH, T.; SPEDALIERI, F. M.; CHANCELLOR, N.; LIDAR, D. A. Experimental signature of programmable quantum annealing. *Nature communications*, Nature Publishing Group, v. 4, p. 2067, 2013.

BOIXO, S.; RØNNOW, T.; ISAKOV, S.; WANG, Z.; WECKER, D.; LIDAR, D.; MARTINIS, J.; TROYER, M. Quantum annealing with more than one hundred qubits. *arXiv preprint condmat/1304.4595*, v. 1304, 2013.

BOOTH, M.; REINHARDT, S.; ROY, A. *Partitioning optimization problems for hybrid classical/quantum execution*. [S.l.], 2017. DWave Technical Report.

CAO, Y.; JIANG, S.; PEROULI, D.; KAIS, S. Solving set cover with pairs problem using quantum annealing. *Scientific reports*, Nature Publishing Group, v. 6, 2016.

CAPRARA, A.; CARR, R.; ISTRAIL, S.; LANCIA, G.; WALENZ, B. 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. *Journal of Computational Biology*, Mary Ann Liebert, Inc., v. 11, n. 1, p. 27–52, 2004.

CAPRARA, A.; LANCIA, G. Structural alignment of large—size proteins via lagrangian relaxation. In: ACM. *Proceedings of the sixth annual international conference on Computational biology*. [S.l.], 2002. p. 100–108.

CERF, N. J.; GROVER, L. K.; WILLIAMS, C. P. Nested quantum search and structured problems. *Physical Review A*, APS, v. 61, n. 3, p. 032303, 2000.

CHOI, V. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, Springer, v. 7, n. 5, p. 193–209, 2008.

CHOI, V. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing*, Springer, v. 10, n. 3, p. 343–353, 2011.

CIPRA, B. A. An introduction to the ising model. *American Mathematical Monthly*, v. 94, n. 10, p. 937–959, 1987.

COCK, P. J.; ANTAO, T.; CHANG, J. T.; CHAPMAN, B. A.; COX, C. J.; DALKE, A.; FRIEDBERG, I.; HAMELRYCK, T.; KAUFF, F.; WILCZYNSKI, B. et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, Oxford University Press, v. 25, n. 11, p. 1422–1423, 2009.

CROSSON, E.; FARHI, E.; LIN, C. Y.-Y.; LIN, H.-H.; SHOR, P. Different strategies for optimization using the quantum adiabatic algorithm. *arXiv preprint arXiv:1401.7320*, 2014.

DAM, W. V.; MOSCA, M.; VAZIRANI, U. How powerful is adiabatic quantum computation? In: IEEE. *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. [S.l.], 2001. p. 279–287.

DAS, A.; CHAKRABARTI, B. K. Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, APS, v. 80, n. 3, p. 1061, 2008.

DEUTSCH, D. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 400, n. 1818, p. 97–117, 1985. ISSN 0080-4630.

DEUTSCH, D.; JOZSA, R. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 439, n. 1907, p. 553–558, 1992. ISSN 0962-8444.

DICKSON, N. G.; AMIN, M. Does adiabatic quantum optimization fail for np-complete problems? *Physical review letters*, APS, v. 106, n. 5, p. 050502, 2011.

DÖRN, S. Quantum complexity bounds for independent set problems. *arXiv preprint quant-ph/0510084*, 2005.

FARHI, E.; GOLDSTONE, J.; GUTMANN, S. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

FARHI, E.; GOLDSTONE, J.; GUTMANN, S.; SIPSER, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

FARHI, E.; GOSSET, D.; HEN, I.; SANDVIK, A.; SHOR, P.; YOUNG, A.; ZAMPONI, F. Performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs. *Physical Review A*, APS, v. 86, n. 5, p. 052334, 2012.

GODZIK, A.; SKOLNICK, J. Flexible algorithm for direct multiple alignment of protein structures and sequences. *Bioinformatics*, Oxford University Press, v. 10, n. 6, p. 587–596, 1994.

GOLDMAN, D.; ISTRAIL, S.; PAPADIMITRIOU, C. H. Algorithmic aspects of protein structure similarity. In: IEEE. *Foundations of Computer Science, 1999. 40th Annual Symposium on*. [S.l.], 1999. p. 512–521.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1996. (STOC '96), p. 212–219. ISBN 0-89791-785-5. Disponível em: <http://doi.acm.org/10.1145/237814.237866>.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA: [s.n.], 2008. p. 11–15.

HEN, I.; YOUNG, A. Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems. *Physical Review E*, APS, v. 84, n. 6, p. 061152, 2011.

HUA, R. *Adiabatic Quantum Computing with QUBO Formulations*. Tese (Doutorado) — MSc. Thesis, University of Auckland, 2016.

JAEGER, G. *Quantum Information: An Overview*. 1. ed. Springer, 2007. Hardcover. ISBN 0387357254. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387357254>.

JÖRG, T.; KRZAKALA, F.; SEMERJIAN, G.; ZAMPONI, F. First-order transitions and the performance of quantum algorithms in random optimization problems. *Physical review letters*, APS, v. 104, n. 20, p. 207206, 2010.

KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, SIAM, v. 20, n. 1, p. 359–392, 1998.

KRASNOGOR, N. Self generating metaheuristics in bioinformatics: The proteins structure comparison case. *Genetic Programming and Evolvable Machines*, Springer, v. 5, n. 2, p. 181–201, 2004.

KRASNOGOR, N.; LANCIA, G.; ZEMLA, A.; HART, W.; CARR, R.; HIRST, J.; BURKE, E. *A comparison of computational methods for the maximum contact map overlap of protein pairs*. [S.l.], 2005.

LEWIS, M.; GLOVER, F. Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis. *Networks*, Wiley Online Library, v. 70, n. 2, p. 79–97, 2017.

LIU, Y.; KOEHLER, G. J. Using modifications to grover's search algorithm for quantum global optimization. *European Journal of Operational Research*, Elsevier, v. 207, n. 2, p. 620–632, 2010.

LLOYD, S.; MOHSENI, M.; REBENTROST, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

LUCAS, A. Ising formulations of many np problems. *Frontiers in Physics*, Frontiers, v. 2, p. 5, 2014.

MCGEOCH, C. C. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing*, Morgan & Claypool Publishers, v. 5, n. 2, p. 1–93, 2014.

MCGEOCH, C. C.; WANG, C. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In: ACM. *Proceedings of the ACM International Conference on Computing Frontiers*. [S.l.], 2013. p. 23.

MCMAHON, D. *Quantum computing explained*. Hoboken, N.J. Wiley-Interscience: IEEE Computer Society, 2007. ISBN 978-0-470-09699-4. Disponível em: <http://opac.inria.fr/record=b1124180>.

MERMIN, N. D. From cbits to qbits: Teaching computer scientists quantum mechanics. *American Journal of Physics*, v. 71, p. 23–30, 2003. ArXiv:quant-ph/0207118.

MERMIN, N. D. *Quantum computer science : an introduction*. Cambridge: Cambridge University Press, 2007. Index inclus. ISBN 978-0-521-87658-2. Disponível em: <http://opac.inria.fr/record=b1122886>.

MOORE, G. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, v. 86, n. 1, p. 82–85, Jan 1998. ISSN 0018-9219.

NEUKART, F.; COMPOSTELLA, G.; SEIDEL, C.; DOLLEN, D. V.; YARKONI, S.; PARNEY, B. Optimizing traffic flow using quantum annealing and classical machine learning. *arXiv preprint arXiv:1708.01625*, 2017.

NIELSEN, M. A.; CHUANG, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. ed. New York, NY, USA: Cambridge University Press, 2011. ISBN 1107002176, 9781107002173.

OHYA, M.; VOLOVICH, I. *Mathematical Foundations of Quantum Information and Computation and Its Applications to Nano- and Bio-systems*. Springer Netherlands, 2011. (Theoretical and Mathematical Physics). ISBN 9789400701717. Disponível em: <https://books.google.com.br/books?id=0nags-BqlSQC>.

ROBERTSON, N.; SEYMOUR, P. Graph minors. xx. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, v. 92, n. 2, p. 325 – 357, 2004. ISSN 0095-8956. Special Issue Dedicated to Professor W.T. Tutte. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0095895604000784>.

ROSENBERG, G.; VAZIFEH, M.; WOODS, B.; HABER, E. Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications*, Springer, v. 65, n. 3, p. 845–869, 2016.

SANDERS, P.; SCHULZ, C. Think locally, act globally: Highly balanced graph partitioning. In: SPRINGER. *International Symposium on Experimental Algorithms*. [S.l.], 2013. p. 164–175.

SANTOS, A. C. O computador quântico da ibm e o ibm quantum experience. *Caderno Brasileiro de Ensino de Física*, v. 39, n. 1, 2017.

SHOR, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1994. (SFCS '94), p. 124–134. ISBN 0-8186-6580-7. Disponível em: <http://dx.doi.org/10.1109/SFCS.1994.365700>.

SILVA, R. M.; RESENDE, M. G.; FESTA, P.; VALENTIM, F. L.; JUNIOR, F. N. Grasp with path-relinking for the maximum contact map overlap problem. In: SPRINGER. *International Conference on Learning and Intelligent Optimization*. [S.l.], 2014. p. 223–226.

SU, J.; TU, T.; HE, L. A quantum annealing approach for boolean satisfiability problem. In: ACM. *Proceedings of the 53rd Annual Design Automation Conference*. [S.l.], 2016. p. 148.

TEMME, K.; OSBORNE, T. J.; VOLLBRECHT, K. G.; POULIN, D.; VERSTRAETE, F. Quantum metropolis sampling. *Nature*, Nature Research, v. 471, n. 7336, p. 87–90, 2011.

USHIJIMA-MWESIGWA, H.; NEGRE, C. F.; MNISZEWSKI, S. M. Graph partitioning using quantum annealing on the d-wave system. *arXiv preprint arXiv:1705.03082*, 2017.

WANG, D.; KLEINBERG, R. Analyzing quadratic unconstrained binary optimization problems via multicommodity flows. *Discrete Applied Mathematics*, Elsevier, v. 157, n. 18, p. 3746–3753, 2009.

WECKER, D.; HASTINGS, M. B.; TROYER, M. Training a quantum optimizer. *Physical Review A*, APS, v. 94, n. 2, p. 022309, 2016.

WOHLERS, I.; BOUDIC-JAMIN, M. L.; DJIDJEV, H.; KLAU, G. W.; ANDONOV, R. Exact protein structure classification using the maximum contact map overlap metric. In: SPRINGER. *International Conference on Algorithms for Computational Biology*. [S.l.], 2014. p. 262–273.

WOHLERS, I.; MALOD-DOGNIN, N.; ANDONOV, R.; KLAU, G. W. Csa: comprehensive comparison of pairwise protein structure alignments. *Nucleic acids research*, Oxford University Press, v. 40, n. W1, p. W303–W309, 2012.

WONG, R. G. An algorithm for quantum circuit optimization. California Polytechnic State University, 2012. Disponível em <http://digitalcommons.calpoly.edu/cscsp/16/>. Acesso em: 11 nov. 2015.

XIE, W.; SAHINIDIS, N. V. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA, v. 14, n. 5, p. 637–654, 2007.

YANG, Z.; DINNEEN, M. J. *Graph Minor Embeddings for D-Wave Computer Architecture*. [S.l.], 2016.

YANOFSKY, N. S.; MANNUCCI, M. A. *Quantum Computing for Computer Scientists*. 1. ed. New York, NY, USA: Cambridge University Press, 2008. ISBN 0521879965, 9780521879965.

ZAHEDINEJAD, E.; ZARIBAFIYAN, A. Combinatorial optimization on gate model quantum computers: A survey. *arXiv preprint arXiv:1708.05294*, 2017.

ZALKA, C. Grover's quantum searching algorithm is optimal. *Physical Review A*, APS, v. 60, n. 4, p. 2746, 1999.

ZEMLA, A. Lga: a method for finding 3d similarities in protein structures. *Nucleic acids research*, Oxford University Press, v. 31, n. 13, p. 3370–3374, 2003.

APÊNDICE A – CÓDIGO PYTHON CMOP2QUBO

Listing A.1 – Código em linguagem de programação Python para gerar a matriz QUBO para o problema CMO

```
import networkx as nx
from networkx import *
import sys , math
import numpy as np
import time
import itertools

def cmop2qubo():

    #reads proteins contact maps and creates their graphs
    protein1= ""
    protein2 = ""

    G1 = read_adjlist(protein1, create_using=nx.Graph(), nodetype=np.int16)
    G2 = read_adjlist(protein2, create_using=nx.Graph(), nodetype=np.int16)

    n1 = G1.order()
    n2 = G2.order()

    V1 = nodes(G1)
    V2 = nodes(G2)

    E1 = edges(G1)
    E2 = edges(G2)

    varsDict = {}

    #defines values of A, B and C
    A = int((n1+n2)/2)
    B = int((n1+n2)/2)
    C = int(((G1.number_of_edges() + G2.number_of_edges()) / 2) * 0.05)

    #creates variables dictionary
    index = 0
    for i in range(n1):
        for j in range(n2):
            varsDict[(i,j)] = index
            index += 1

    #initializes qubo matrix with zeros
    quboMatrix = np.zeros((n1*n2, n1*n2), dtype=np.int16)

    #HA part 1
    for i in range(n1):
        for iprime1 in range(n2):
            for iprime2 in range(n2):
                index1 = varsDict[(i, iprime1)]
                index2 = varsDict[(i, iprime2)]
```

```
49                  if(index1 != index2):
50                      quboMatrix[index1][index2] += A
51
52      #HA part 2
53      for iprime in range(n2):
54          for i1 in range(n1):
55              for i2 in range(n1):
56                  index1 = varsDict[(i1, iprime)]
57                  index2 = varsDict[(i2, iprime)]
58                  if(index1 != index2):
59                      quboMatrix[index1][index2] += A
60
61      #HB
62      for i in range(n1):
63          for iprime in range(n2):
64              for j in range(i+1, n1):
65                  for jprime in range(iprime):
66                          x1 = varsDict[(i, iprime)]
67                          x2 = varsDict[(j, jprime)]
68                          quboMatrix[x1][x2] += B
69                          quboMatrix[x2][x1] += B
70
71      #HC
72      for i in edges(G1):
73          for j in edges(G2):
74              x1 = varsDict[(i[0],j[0])]
75              x2 = varsDict[(i[1],j[1])]
76              quboMatrix[x1][x2] -= C
77              quboMatrix[x2][x1] -= C
78
79      #creates upper triangular qubo matrix
80      quboMatrix += np.triu(quboMatrix,1)
```

APÊNDICE B – CÓDIGO PYTHON CMOP2QUBO

Listing B.1 – Código em linguagem de programação Python para gerar a matriz QUBO para o problema CMO - 2ª abordagem

```
import networkx as nx
from networkx import *
import sys , math
import numpy as np
import time
import itertools

def generateQUBO():

    #reads proteins contact maps and creates their graphs
    protein1= ""
    protein2 = ""

    G1 = read_adjlist(protein1, create_using=nx.Graph(), nodetype=np.int16)
    G2 = read_adjlist(protein2, create_using=nx.Graph(), nodetype=np.int16)

    n1 = G1.order()
    n2 = G2.order()

    V1 = nodes(G1)
    V2 = nodes(G2)

    E1 = edges(G1)
    E2 = edges(G2)

    varsDict = {}

    #defines values  of A, B, C and percentage of mappings
    A = int((n1+n2)/2)
    B = int((n1+n2)/2)
    C = int(((G1.number_of_edges() + G2.number_of_edges()) / 2) * 0.05)
    ic = int(n2*0.1)

    #creates variables  dictionary with least mappings
    index = 0
    step = float(n2)/n1
    pos = 0
    for i in range(n1):
        start = max(0, pos - ic)
        end = min(n2, pos + ic + 1)
        for j in range(start, end):
            varsDict[(i,j)] = index
            index += 1
        pos = int((i+1) * step)

    #initializes qubo matrix with zeros
    quboMatrix = np.zeros((len(varsDict), len(varsDict)), dtype=np.int16)

```

```
    #HA part 1
    temp = []
    for i in range(n1):
        for v in varsDict.iterkeys():
            if(v[0]==i):
                temp.append(v)
        for k2 in temp:
            for k3 in temp:
                index1 = varsDict[k2]
                index2 = varsDict[k3]
                if(index1 != index2):
                    quboMatrix[index1][index2] += A
        temp = []

    #HA part 2
    temp = []
    for j in range(n2):
        for v in varsDict.iterkeys():
            if(v[1]==j):
                temp.append(v)
        for k2 in temp:
            for k3 in temp:
                index1 = varsDict[k2]
                index2 = varsDict[k3]
                if(index1 != index2):
                    quboMatrix[index1][index2] += A
        temp = []

    #HB
    vk = sorted(varsDict.keys())
    l = len(vk)
    for i in range(l):
        for j in range(i+1,l):
            if((vk[i][0]<vk[j][0]) and vk[i][1]>vk[j][1]):
                index1 = varsDict[vk[i]]
                index2 = varsDict[vk[j]]
                quboMatrix[index1][index2] += B
                quboMatrix[index2][index1] += B

    #HC
    for i in E1:
        for j in E2:
            if ((i[0],j[0]) in varsDict and (i[1],j[1]) in varsDict):
                index1 = varsDict[(i[0],j[0])]
                index2 = varsDict[(i[1],j[1])]
                quboMatrix[index1][index2] -= C
                quboMatrix[index2][index1] -= C

    #creates upper triangular qubo matrix
    quboMatrix += np.triu(quboMatrix,1)
```

APÊNDICE C – CÓDIGO PYTHON QUBO2FILE

Listing C.1 – Código em linguagem de programação Python para criação do arquivo .qubo

```
def qubo2file(quboMatrix, qubo_in):

    #gets the nonzero elements on diagonal and off-diagonal
    diagonal = np.diagonal(quboMatrix)
    s = len(diagonal)
    nz_diagonal = np.count_nonzero(diagonal)
    upper_tri = np.triu(quboMatrix,1)
    nz_elements = np.nonzero(upper_tri)

    #create the comments
    initial_comment = "c .qubo file for execution in qbsolv\n"
    p_line = "p qubo 0 " + str(s) + " " + str(nz_diagonal) + " " +
             str(len(nz_elements[0])) + "\n"
    c_line1 = "c begin of diagonal elements\n"
    c_line2 = "c end of diagonal elements\n"
    c_line3 = "c begin of off-diagonal elements\n"
    c_line4 = "c end of off-diagonal elements\n"

    aux1 = ""
    aux2 = ""

    #create diagonal lines
    for i in range(len(diagonal)):
        if(diagonal[i] != 0):
            diagonalLine = str(i) + " " + str(i) + " " + str(diagonal[i]) + "\n"
            aux1 += diagonalLine

    #create off-diagonal lines
    for i in range(len(nz_elements[0])):
        elementLine = str(nz_elements[0][i]) + " " + str(nz_elements[1][i]) + " " +
                      str(upper_tri[nz_elements[0][i]][nz_elements[1][i]]) + "\n"
        aux2 += elementLine

    #write in .qubo file
    with open('../qubos/in/' + qubo_in,'w') as file_qubo:
        file_qubo.write(initial_comment + p_line + c_line1)
        file_qubo.write(aux1)
        file_qubo.write(c_line2 + c_line3)
        file_qubo.write(aux2)
        file_qubo.write(c_line4)
```