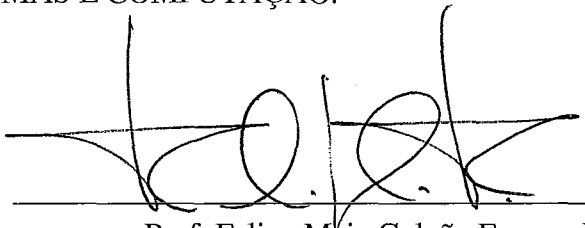
MAPEAMENTO E COMBINAÇÃO DE PROBLEMAS NP-DIFÍCEIS ATRAVÉS DE RESTRIÇÕES PSEUDO-BOOLEANAS PARA REDES NEURONAIS ARTIFICIAIS

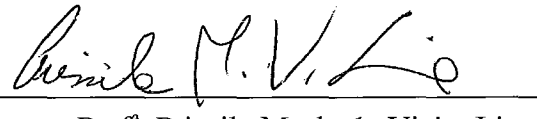
Glaucia da Conceição Pereira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

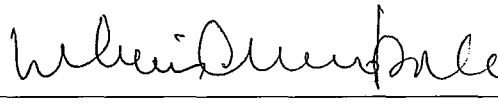
Aprovada por:



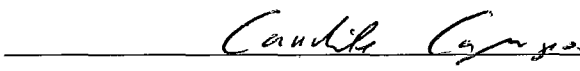
Prof. Felipe Maia Galvão França, Ph.D.



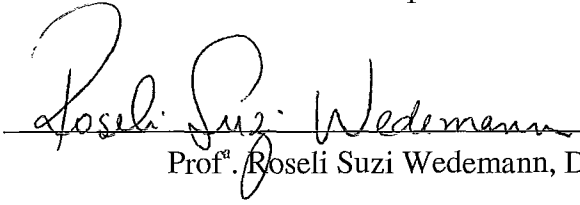
Profª. Priscila Machado Vieira Lima, Ph.D.



Prof. Valmir Carneiro Barbosa, Ph.D.



Prof. Carlile Campos Lavor, D.Sc.



Profª. Roseli Suzi Wedemann, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2006

PEREIRA, GLAUCIA DA CONCEIÇÃO

Mapeamento e Combinação de Problemas NP-Difíceis através de Restrições Pseudo-Booleanas para Redes Neuronais Artificiais [Rio de Janeiro] 2006

XII, 129 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE.

1 - Mapeamento de Satisfatibilidade para Minimização de Energia

2 - Redes Neuronais Artificiais

3 - SATyrus

4 - Predição de Conformações Moleculares Estáveis

5 - Problemas NP-Difíceis

I. COPPE/UFRJ II. Título (série)

"I am enough of an artist to draw freely upon my imagination. Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."

Albert Einstein

Agradecimentos

Gostaria inicialmente de agradecer a Deus, por me ajudar em todos os momentos de minha caminhada e a minha família que é a verdadeira base da minha vida. Obrigada pelo amor, dedicação, apoio e incentivo de sempre.

Agradeço aos meus orientadores Prof. Felipe M. G. França e Prof^a. Priscila M. V. Lima pelo incentivo, pela amizade e pela infraestrutura proporcionada para a realização deste trabalho. Às minhas orientadoras de graduação Prof ^{as}. Roseli S. Wedemann e Rosa M. V. de Figueiredo por todo apoio e consideração. Agradeço também ao Prof. Carlile C. Lavor pela grande contribuição no estudo de alguns problemas tratados neste trabalho e a Prof^a. Inês C. Dutra pela grande disposição em ajudar sempre.

Agradeço aos amigos da COPPE/UFRJ, por me ajudarem sempre que preciso, em especial à amiga Mariela.

Finalmente, obrigada aos colaboradores da UFRJ, que direta ou indiretamente contribuíram com a realização deste trabalho, em especial à Prof^a Lígia Barros, pelos conselhos pessoais e profissionais.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MAPEAMENTO E COMBINAÇÃO DE PROBLEMAS NP-DIFÍCEIS ATRAVÉS DE RESTRIÇÕES PSEUDO-BOOLEANAS PARA REDES NEURONAIS ARTIFICIAIS

Glaucia da Conceição Pereira

Setembro/2006

Orientadores: Felipe Maia Galvão França
Priscila Machado Vieira Lima

Programa: Engenharia de Sistemas e Computação

Este trabalho introduz parte do sistema computacional SATyrus. Este sistema representa a implementação de uma estratégia híbrida que combina lógica proposicional e redes neuronais, no tratamento de problemas complexos, em particular problemas NP-difíceis. Esta estratégia híbrida busca usar os benefícios provenientes da estratégia mapeamento de satisfatibilidade para minimização de energia (SMEM), que consiste em um método baseado em lógica proposicional para o mapeamento de problemas, definidos através de um conjunto de restrições, para funções de energia aliado ao mecanismo de busca global das redes neuronais estocásticas de alta ordem. Além disso, apresentamos a modelagem de algumas operações aritméticas, de alguns problemas NP-difíceis, de um modelo modificado para o problema das distâncias geométricas moleculares (MDGP) e introduzimos um novo modelo para o problema da predição das conformações moleculares estáveis (SMCPP) combinando um modelo clássico e um modelo geométrico (MDGP-Modificado). Por fim, para ilustrar o uso do sistema SATyrus o aplicamos na simulação de três problemas NP-difíceis e de três problemas aritméticos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MAPPING AND COMBINING NP-HARD PROBLEMS AS PSEUDO-BOOLEAN CONSTRAINTS INTO ARTIFICIAL NEURAL NETWORKS

Glaucia da Conceição Pereira

September/2006

Advisors: Felipe Maia Galvão França
Priscila Machado Vieira Lima

Department: Computer and Systems Engineering

This work introduces part of the SATyrus computational system. This system represents the implementation of a hybrid strategy that combines propositional logic and neural networks in the treatment of complex problems, in particular NP-hard problems. This hybrid strategy seeks to use the benefits from the Satisfiability Mapped into Energy Minimization strategy (SMEM), which presents a method based on propositional logic for the mapping of problems that can be defined through a set of restrictions into an Energy Function and from the mechanism of global search of Stochastic higher-Order Neural Networks. Moreover, we present the modeling of some arithmetic operations, some NP-Hard problems, a modified model of the Molecular Distance Geometry Problem (MDGP), and we introduce a new model to Stable Molecular Conformations Prediction Problem (SMCPP) combining a classical and a geometric (MDGP-Modified) models. Finally, to illustrate the use of the SATyrus system we apply it to three NP-hard problems and to three arithmetic problems.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A modelagem e a busca por soluções de problemas que possuem alto nível de complexidade apresentam inúmeras dificuldades, pois especificá-los através de um modelo matemático, capaz de descrever com precisão todas as características dos mesmos, pode ocasionar erros que diminuem significativamente a qualidade dos resultados obtidos. Por outro lado, espera-se que os métodos de busca utilizados sejam capazes de chegar a resultados precisos (idealmente o resultado ótimo) em um tempo computacional satisfatório (em geral de ordem polinomial). Neste contexto, por serem considerados problemas intratáveis e por serem utilizados na representação de muitas aplicações práticas, problemas NP-difíceis têm sido muito abordados na literatura. Neste caso, inúmeros trabalhos têm como objetivo encontrar estratégias de busca eficientes e ou apresentar modelos matemáticos que influenciem positivamente no processo de busca.

A respeito da modelagem, algumas abordagens descrevem os problemas de forma que seu conjunto de restrições seja expresso através de uma rede neuronal. Esta rede é definida por uma função de energia e seu conjunto de mínimos globais corresponde ao conjunto de soluções ótimas do problema. O problema do caixeiro

viajante (*Traveling Salesperson Problem* - TSP), por exemplo, foi definido em [1] através de uma rede de Hopfield cuja evolução determina o processo de busca. Posteriormente, verificou-se que a especificação apresentada em [1] estava incompleta e outros trabalhos, como os de Barbosa [2], Jones [3] e o de Carvalho e Barbosa [4], apresentaram a especificação completa, também através de uma rede de Hopfield. Além disso, Cheng e Baldi em [5] realizam a predição (montagem) de resíduos β-sheets de proteínas com o auxílio de redes neuronais recursivas. Estas redes são utilizadas para predizer a probabilidade de emparelhamento entre os pares de β-resíduos. O problema fundamental neste tipo de abordagem é que as funções de energia, que representam o espaço de soluções do problema estudado, são, em geral, geradas a partir do grafo associado ao problema. No caso do TSP e do problema da coloração de vértices (*Vertex Coloring Problem* - VCP) a construção das redes neuronais que os define é relativamente simples, porém em problemas complexos, como o problema da predição das conformações moleculares estáveis (*Stable Molecular Conformations Prediction Problem* - SMCPP), esta construção representa uma tarefa árdua podendo levar à introdução de erros adicionais a solução. Uma alternativa seria especificar tais problemas através de um conjunto de restrições essencialmente lógicas, esperando poder trabalhar com um conjunto de fórmulas da lógica proposicional. Este conjunto de fórmulas pode, por sua vez, ser mapeado para uma função de energia que representa uma rede de Hopfield, possívelmente de alta ordem. Em outras palavras, podemos obter, através de um método sistemático, uma rede neuronal a partir da especificação dos problemas.

Outra questão importante que será abordada neste trabalho é o estudo da geometria de compostos químicos, pois a estrutura tri-dimensional dos compostos está diretamente ligada à função química dos mesmos [6, 7]. Cheng e Baldi em [8],

por exemplo, fazem a predição de domínios protéicos utilizando, dentre outras coisas, informações sobre a estrutura secundária das proteínas. Além disso, o rápido crescimento da área de planejamento computacional de fármacos exige a disponibilidade de ferramentas computacionais cada vez mais precisas para que os bancos de dados possam ser constantemente atualizados. As informações fornecidas por estas ferramentas computacionais, obtidas por técnicas recentes de modelagem molecular [9], agilizam a análise da atividade biológica e das propriedades físico-químicas de compostos utilizados, dentre outras coisas, no desenvolvimento de novos agentes terapêuticos.

Neste contexto, os objetivos deste trabalho são:

1. Utilizar a estratégia SMEM como sintetizador de formulações exatas;
2. Utilizar uma estratégia de busca global, como é o caso das redes neuronais estocásticas;
3. Desenvolver um modelo para o problema da predição das conformações moleculares estáveis (SMCPP) que apresente em sua composição diferentes teorias.

A metodologia que utilizaremos para que os objetivos propostos possam ser alcançados será:

1. A implementação de uma estratégia híbrida, proposta em [10] e em [11]. Esta estratégia consiste na combinação entre a estratégia para a modelagem de problemas mapeamento de satisfatibilidade para minimização de energia

(SMEM) e o algoritmo de busca de uma rede de Hopfield estocástica (possivelmente de alta ordem) e deu origem ao sistema computacional SATyrus [12];

2. A ilustração da aplicação do sistema SATyrus através da apresentação dos resultados experimentais relativos à busca pelas soluções de dois problemas NP-difíceis, da combinação entre eles e de três problemas aritméticos que auxiliarão na construção do modelo para o SMCPP que apresentaremos neste trabalho;

3. A aplicação do sistema à modelagem de problemas complexos ilustrando sua capacidade em representar tais problemas através da união de problemas mais simples [11], em particular, na construção do problema híbrido caixeiro viajante colorido (*Map Coloring-TSP* - MC-TSP);

4. A apresentação de um modelo matemático, construído a partir da estratégia SMEM, para o problema das conformações moleculares estáveis (SMCPP), no qual este problema é tratado como um problema híbrido, resultado da combinação entre um modelo para o problema das distâncias geométricas moleculares (*Molecular Distance Geometry Problem* - MDGP) e de um modelo físico para o SMCPP. Estes modelos serão construídos a partir da combinação de redes que realizam a operação de soma e de redes que realizam a operação de produto de números na base 2.

Nossas principais contribuições foram:

1. O desenvolvimento do simulador neuronal que compõe o sistema computacional SATyrus;

2. A geração dos modelos matemáticos de algumas operações aritméticas binárias (soma / subtração, modulo e produto) através da estratégia SMEM;

3. A introdução de um modelo matemático, construído a partir da estratégia SMEM, para o problema das conformações moleculares estáveis (SMCPP), no qual este problema é tratado como um problema híbrido, isto é, uma combinação entre o modelo geométrico (MDGP-Modificado [13]) e o físico-químico clássico. Isto confirma a proposta de aplicação da estratégia SMEM na modelagem de problemas complexos vistos como problemas híbridos [11].

Veremos no Capítulo 2 algumas definições e notações que serão utilizadas ao longo deste trabalho. Em seguida, no Capítulo 3, apresentaremos o sistema computacional SATyrus, onde falaremos dos módulos que o compõem (um compilador lógico e um simulador neuronal) e apresentaremos os modelos matemáticos de seis problemas para ilustrar a aplicação da estratégia SMEM.

No Capítulo 4, apresentaremos a construção do modelo matemático proposto para o problema da predição das conformações moleculares estáveis (SMCPP). No Capítulo 5, mostraremos os resultados experimentais relativos ao desempenho do sistema SATyrus, em particular, do simulador neuronal apresentado neste trabalho, na busca de uma das possíveis soluções dos problemas apresentados no Capítulo 3. Finalmente, no Capítulo 6 apresentaremos nossas conclusões e propostas para trabalhos futuros.

Capítulo 2

Conhecimentos Preliminares

Neste capítulo falaremos das estratégias e conceitos que formam a base deste trabalho. Na Seção 2.1 mostraremos algumas propriedades das redes neuronais de Hopfield (binárias e estocásticas) e definiremos redes de alta ordem. Descreveremos, na Seção 2.2, o conjunto de passos que serão utilizados para transformar problemas especificados sob a forma de restrições em problemas de minimização de energia . Estes passos definem a estratégia mapeamento de satisfatibilidade para minimização de energia (SMEM) e foram utilizados por Pinkas [14] e, posteriormente, por Lima [10], na modelagem do problema do caixeiro viajante e por Lima *et al.* [11], na modelagem do problema da coloração de vértices e do problema híbrido MC-TSP, resultado da combinação dos dois problemas anteriores. Finalmente, na Seção 2.3, falaremos sobre algumas formas de modelar o problema da predição das conformações moleculares estáveis (SMCPP) com o objetivo de utilizá-las em um novo modelo para este problema [1].

[1] As Subseções 2.1.1 e 2.1.3 foram baseadas em [2].

2.1 Redes de Hopfield Estocásticas e de Alta Ordem

2.1.1 Redes Binárias de Hopfield

Um sistema dinâmico e discreto em relação ao tempo é definido por um conjunto de elementos interdependentes que têm seu comportamento interno descrito através da especificação de como valores (estados) associados a seus elementos variam com o tempo s e o domínio de s é o conjunto dos números naturais. Neste contexto, uma rede de autômatos A é um sistema dinâmico discreto em relação ao tempo, definido por um par (G, f), onde $G = (V, E)$ é um grafo não direcionado (Definição 2.1) que representa a estrutura da rede e f é a função de atualização que define a dinâmica da rede, pois ela é uma expressão matemática que determina a evolução, no tempo, dos estados de cada vértice $i \in V$.

Definição 2.1. *Um **Grafo Não Direcionado** que, por simplicidade, denotaremos por grafo, é um par ordenado $G = (V, E)$ onde V é um conjunto finito não vazio de vértices e E é um subconjunto de $\{(i,j)|i,j \in V\}$, tal que cada (i,j) é um par não ordenado, ou seja, $(i,j) = (j,i)$, denominado aresta. Além disto, se $(i,j) \in E \Rightarrow (p,l) \notin E$ e $(p,l) \in E \Rightarrow (i,j) \notin E$, então $i = p$ e $j = l$.*

Definimos $v_i(s)$, pertencente ao domínio D, como o estado de um vértice $i \in V$, no tempo (ou passo de atualização [2]) $s \geq 0$, sendo $v_i(0)$ o estado inicial do vértice i. Além disso, considerando $|V| = n$, chamamos de estado da rede de autômatos no tempo $s \geq 0$ e representamos por $\mathbb{N} = \mathbb{N}(s) = (v_1(s), v_2(s), \ldots, v_n(s))$, o conjunto dos estados de todos os vértices $i \in V$, $1 \leq i \leq n$ no passo de atualização $s \geq 0$.

[2]Um passo de atualização consiste na seleção de um ou mais vértices para atualização de seus estados.

Definição 2.2. *Seja $G = (V, E)$, grafo, e $H = (U, F)$ subgrafo de G.*

H é um ***Conjunto Independente****, denotado por I, se e somente se, para todo par $i, j \in U$, $(i, j) \notin E$.*

Definição 2.3. *Tome um grafo $G = (V, E)$ e um vértice $i \in V$.*

Chamaremos ***Vizinhança*** *de i em G e denotaremos por $N(i)$ o conjunto $\{j \in V | (i, j) \in E\}$, ou seja, $N(i)$ é o subconjunto de vértices de V que são adjascentes* [3] *a i em G.*

Definição 2.4. *Sejam $I_1, I_2, ...$ conjuntos independentes (Definição 2.2) no grafo $G = (V, E)$ associado a uma rede de autômatos A, tal que todo vértice em V aparece infinitamente freqüentemente na sequência $I_1, I_2, ...$ (segundo Barbosa [2], isso significa que existe uma constante $K \geq 0$, tal que todo vértice em V aparece em pelo menos um dos conjuntos $I_{K_0}, I_{K_0+1}, ...I_{K_0+K}$, para todo $K_0 > 0$).*

Tomando I_s como o conjunto de vértices em V, selecionados para atualização concorrente no passo $s > 0$, então temos uma ***Regra de Atualização Parcialmente Concorrente****, determinada pela equação 2.1.*

$$v_i(s) = \begin{cases} f(v_j(s-1);\ j \in \{i\}\ U\ N(i)),\ \textit{se } i \in I_s; \\ v_i(s-1),\ \textit{caso contrário.} \end{cases} \tag{2.1}$$

Para todo $i \in V$ e para todo $s \geq 0$, a regra de atualização parcialmente concorrente inicia em $v_i(0)$ e para $s > 0$ $v_i(s)$ assume a forma 2.1, tal que o termo $N(i)$, apresentado na Equação 2.1, representa a vizinhança do vértice i (Definição 2.3).

[3] Dado um grafo $G = (V, E)$, dizemos que dois vértices $i, j \in V$ são adjacentes ou vizinhos em G se existe aresta $(i, j) \in E$ definida por eles.

Neste contexto, uma **Rede Neuronal Binária de Hopfield**, ou simplesmente rede de Hopfield, é uma rede de autômatos, cuja regra de atualização é parcialmente concorrente, com domínio $D = \{0, 1\}$ e cujos elementos em V são denominados neurônios artificiais, ou apenas neurônios.

Em uma rede de Hopfield a regra de atualização para os estados dos neurônios $i \in V$, é dada através da Equação 2.2:

$$v_i := degrau(\sum_{j=1}^{n} w_{ij} v_j + e_i - \theta_i), \tag{2.2}$$

onde $degrau(y) = \left\{ \begin{array}{l} 0, \ se \ y \leq 0; \\ 1, \ se \ y > 0. \end{array} \right.$

θ_i é um limiar (threshold) ou fator de ativação, associado ao neurônio i. Em outras palavras, θ_i estabelece as k-uplas $(v_j, ..., v_{j'})$ de estados de vizinhos de i que, segundo a função de atualização *degrau* (Equação 2.2), tornam o estado de i 0 ou 1.

Em analogia à teoria relacionada aos neurônios biológicos, w_{ij} é o peso sináptico do neurônio j para o neurônio i. Este peso representa a influência da atividade do neurônio j sobre o neurônio i (que será nula caso i e j não sejam vizinhos em G) e e_i é o peso associado a uma entrada externa, de valor fixo e igual a 1, para o neurônio i.

O comportamento de uma rede binária de Hopfield é representado através de uma função de energia $E(v_1, ..., v_n)$, ou simplesmente E, que apresentamos através da Equação 2.3. A Figura 2.1 apresenta um exemplo de rede binária de Hopfield.

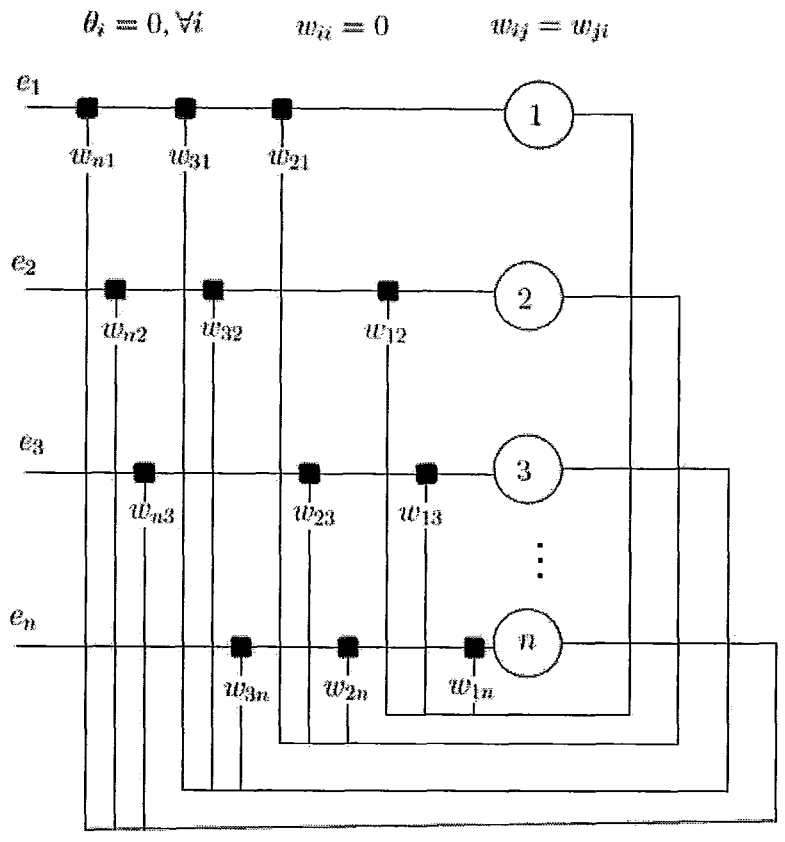


Figura 2.1: Arquitetura de uma rede binária de Hopfield com n neurônios.

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}v_iv_j - \sum_{i=1}^{n} e_iv_i + \sum_{i=1}^{n} \theta_iv_i \tag{2.3}$$

Teorema 2.1. *Suponha* $w_{ii} = 0$, $w_{ij} = w_{ji}$, *para todo* $i, j \in V$ *e, tome um subconjunto* I_s *de* V, *sendo tal que, se* $i, j \in I_s$ *então* $w_{ij} = 0$.

Se E^1 *e* E^2 *são, respectivamente, os valores de* E *imediatamente antes e depois dos neurônios em* V_s *serem atualizados concorrentemente, segundo a Equação 2.3, então* $\Delta\mathrm{E} = \mathrm{E}^2 - \mathrm{E}^1 \leq 0$.

Prova: Omitida. A prova deste teorema foi apresentada por Hopfield em [15].

O Teorema 2.1 apresenta as condições suficientes para que um mínimo de energia

Algoritmo 1 Rede Binária de Hopfield (Barbosa [2]).

1: Sejam: E a função de energia que define a rede, n o número de nós da rede, v_i' uma variável auxiliar para o ùltimo estado do neurônio $i \in V$ antes do passo de atualização corrente, (v_i o estado recebido após o passo de atualização corrente ser executado.
2: inicializar n, v_1), ..., e v_n
3: **repita**
4: **para** todo i em $\{1, ..., n\}$ **faça**
5: $v_i' := v_i(s)$
6: $v_i(s) := degrau(\sum_{j=1, j \neq i}^{n} w_{ij} v_j + e_i - \theta_i)$
7: **fim para**
8: **até** $v_i(s) = v_{i'}, \forall i \in \{1, ..., n\}$

da Função 2.3 seja atingido, quando neurônios não adjacentes em V são atualizados concorrentemente [4] e o Algoritmo 1 [2] apresenta um esquema simples para a simulação sequencial de uma rede binária de Hopfield.

Podemos observar que a Função 2.3 decresce monotonicamente e isso pode nos levar a mínimos locais de energia. Entretanto, desejamos aplicar esta estratégia na busca por soluções de problemas NP-difíceis. Neste caso, será necessário combiná-la com algum mecanismo que permita que mínimos globais sejam encontrados.

2.1.2 Redes de Hopfield Estocásticas

Como mencionamos na Subseção 2.1.1, a regra de atualização de estados de uma rede binária de Hopfield não garante que um mínimo global de energia seja encontrado. Neste caso, a combinação entre uma rede de Hopfield e o algoritmo de busca arrefecimento simulado (*Simulated Annealing* - SA), apresentado em [16], tem se mostrado promissora, pois possibilita o escape de mínimos locais.

O algoritmo de busca arrefecimento simulado (Algoritmo 2) é inspirado em um

[4] Neste trabalho, consideramos $|I_s|$, apresentado no Teorema 2.1, igual a um.

processo físico de aquecimento de metais, vidros, etc., seguido de um lento e gradual resfriamento para que o material se torne mais resistente, num ponto de energia mínima. Em analogia ao processo físico, o espaço de soluções de cada problema P é percorrido de forma que uma candidata a solução é escolhida com certa probabilidade e de acordo com uma com uma variável T (Temperatura). Quanto maior for T mais aleatória será a candidata a solução escolhida em relação à condidata imediatamente anterior. Uma das principais vantagens deste algoritmo é permitir que o espaço de soluções seja melhor explorado.

Algoritmo 2 Arrefecimento Simulado.

1: Admita a aplicação do algoritmo de arrefecimento simulado à busca por uma solução y de determinado problema P.
Sejam: y' uma variável auxiliar que representa uma candidata a solução no passo de busca corrente, y uma variável auxiliar que representa uma candidata a solução no passo de busca imediatamente anterior, $E_{y'}$ a energia associada a uma candidata a solução y', E_y a energia associada a uma candidata a solução y, T a temperatura, T_f a temperatura final.
2: inicializar y e T
3: **enquanto** $T \geq T_f$ **faça**
4: gere y'
5: **se** $E_{y'} < E_y$ **então**
6: y recebe y'
7: **senão**
8: y recebe y' *com probabilidade* $exp((E_y - E_{y'})/T)$
9: **fim se**
10: T recebe εT
11: **fim enquanto**

Em analogia à notação $v_i(s)$, utilizada para representar o estado de um neurônio $i \in V$ em determinado passo de atualização s, utilizaremos uma variável aleatória v_i [5] associada a cada $i \in V$. Cada variável v_i possuirá um valor no domínio $D = \{0, 1\}$ que denominaremos d_i. Desta forma, denotaremos cada elemento de D^n, de forma simplificada, como uma n-upla $(d_1, ..., d_n)$.

[5] Uma variável aleatória é uma quantidade que, sob certas circunstâncias, pode assumir valores diferentes.

Definição 2.5. *Considere um grafo $G = (V, E)$, uma vizinhança $N(i)$ de um vértice $i \in V$.*

$N(i)$ define uma ***Vizinhança Homogênea*** *em G, denotada por $Q(i)$, se e somente se $N(i)$ é tal que para cada par $i, j \in V$, se $j \in N(i)$ então $i \in N(j)$.*

Tome $\mathbb{V}$ como o conjunto de variáveis $v_1, ..., v_n$, tal que para cada $v_i \in \mathbb{V}$ esteja definido um conjunto de vizinhos $Q(v_i)$, de forma que uma vizinhança homogênea (Definição 2.5) seja obtida e, considerando um subconjunto C de $\mathbb{V}$, onde $\forall v_i, v_j \in C$, $v_j \in Q(v_i)$ (e conseqüentemente, $v_i \in Q(v_j)$), tome $\mathbb{C}$ como o conjunto de todos os possíveis $C \subseteq \mathbb{V}$. Definimos uma função de energia através da Equação 2.4.

$$E(d_1, ..., d_n) = \sum_{C \in \mathbb{C}} \mathbb{V}_C(d_1, ..., d_n) \tag{2.4}$$

$\forall (d_1, ..., d_n) \in D^n$.

Na Equação 2.4 $\mathbb{V}_C$ é chamado potencial e depende de todas as coordenadas em $(d_1, ..., d_n)$ que correspondem a variáveis em C ou é constante. Neste caso, tomando uma distribuição de probabilidade P sobre D^n, podemos apresentar as Definições 2.6 e 2.7 [2].

Definição 2.6. *$\mathbb{V}$ é um* ***Campo Aleatório de Markov*** *(MRF) em relação a Q e a P se:*

$$P(d_1, ..., d_n) > 0, \tag{2.5}$$

$\forall (d_1, ..., d_n) \in D^n$;

$$P(d_i|d_j; v_j \neq v_i) = P(d_i|d_j; v_j \in Q(v_i)), \tag{2.6}$$

$\forall v_i \in \mathbb{V}$.

Definição 2.7. $\mathbb{V}$ *é um* ***Campo Aleatório de Gibbs*** *(GRF) em relação a* Q *e a* P *se* P *é a distribuição de Boltzmann-Gibbs* (π)*, ou seja,*

$$P(d_1, ..., d_n) = \pi(d_1, ..., d_n) = \frac{\exp(-\mathrm{E}(d_1, ..., d_n)/T)}{\sum_{(\prime d_1, ..., \prime d_n) \in D^n} \exp(-\mathrm{E}(\prime d_1, ..., \prime d_n)/T)}, \tag{2.7}$$

$\forall (d_1, ..., d_n) \in D^n$.

Na Definição 2.6, a probabilidade condicional $P(d_i|d_j; v_j \neq v_i)$ significa a probabilidade da variável $v_i \in \mathbb{V}$ assumir o valor d_i, tal que cada variável $v_j \in \mathbb{V}, v_j \neq v_i$ possui o valor d_j. Enquanto $P(d_i|d_j; v_j \in Q(v_i))$ significa a probabilidade da variável $v_i \in \mathbb{V}$ assumir o valor d_i, tal que seus vizinhos v_j em Q possuem o valor d_j.

A definição de MRF (Definição 2.6) apresenta uma dependência local, pois cada v_i é avaliada com base nos valores de seus vizinhos v_j em $\mathbb{V}$. Por outro lado, o Teorema 2.2 condiciona esta dependência local ao fato da distribuição P ser a distribuição de Boltzmann-Gibbs π. Neste caso, $P(d_i|d_j; v_j \neq v_i)$ (Definição 2.6) pode ser determinada através da Equação 2.8. Nesta equação, E_i representa a energia apresentada na Equação 2.4, calculada em função da variável v_i.

$$P(d_i|d_j; v_j \neq v_i) = \frac{\exp(-E_i(d_1, ..., d_n)/T)}{\sum_{\prime d_i \in D} \exp(-E_i(d_1, ..., \prime d_i, ..., d_n)/T)} \tag{2.8}$$

Teorema 2.2. $\mathbb{V}$ *é um MRF em relação à uma vizinhança* Q *e a uma distribuição de probabilidade* $P \Leftrightarrow \mathbb{V}$ *é um GRF em relação à mesma vizinhança e a mesma distribuição de probabilidade.*

Prova: Omitida. A prova deste teorema se encontra em [17] *apud* [2] e parece ter sido apresentada pela primeira vez em [18].

O conjunto de variáveis em $\mathbb{V}$ deve ser percorrido de forma que seja garantido que todas elas tenham chance de se atualizar quando o número de passos de atualização tender ao infinito e de forma que a cada passo de atualização um novo valor seja aceito com certa probabilidade, determinada através da distribuição de probabilidade de Boltzmann-Gibbs. Neste caso, a simulação sequêncial (baseada no Algoritmo 1) utilizada neste trabalho respeita esta condição. Cada vez que uma variável v_i for visitada, segundo a ordem estabelecida no Algoritmo 1, seu valor será atualizado para d_i, de acordo com a distribuição de probabilidade π.

Em [19] é apresentado um teorema que garante que se todas as variáveis em $\mathbb{V}$ são visitadas infinitamente freqüentemente, em uma simulação que combina o algoritmo de arrefecimento simulado à distribuição de Boltmann-Gibbs, então, quando o número de passos de atualização tende ao infinito, a probabilidade de se atingir um ponto de mínimo global de energia tende a um. Neste contexto, uma **Rede de Hopfield Estocástica** é uma rede binária de Hopfield, tal que o estado de cada neurônio $i \in V$ está associado a uma variável aleatória $v_i \in \mathbb{V}$ e cuja atualização de estados é realizada de acordo com a distribuição de probabilidade de Boltzmann-Gibbs em associação com o algoritmo de arrefecimento simulado, de forma que todos os neurônios da rede tenham chance de se atualizar com freqüência infinita.

Algoritmo 3 Rede de Hopfield Estocástica (Barbosa [2]).

1: Sejam: E_i a função de energia que define a rede, cujo valor depende do estado $v_i \in D = \{0, 1\}$ do neurônio i, ε a velocidade do resfriamneto, T a temperatura, T_f a temperatura final e n o número de nós da rede.
2: inicializar n, T, (v_1, ... e v_n)
3: **enquanto** $T \geq T_f$ **faça**
4: **para** todo i em $\{1, ..., n\}$ **faça**
5: *Avaliar a probabilidade condicional*

$$P(d_i|d_j; v_j \neq v_i) = \frac{\exp(-E_i(d_1, ..., d_n)/T)}{\sum_{\prime d_i \in D} \exp(-E_i(d_1, ..., \prime d_i, ..., d_n)/T)}$$

para todo $d_i \in D$ *e escolher o valor de* v_i *de acordo.*
6: **fim para**
7: T recebe εT
8: **fim enquanto**

Como se trata de uma rede binária de Hopfield, na Equação 2.7 os possíveis valores de d_i devem pertencer a $D = \{0, 1\}$. Por isso, a regra de atualização de estados de uma rede de Hopfielde estocástica é expressa através das equações apresentadas em 2.9.

$$\begin{aligned} P(v_i = 0|v_j = d_j; i \neq j) &= 1/(1 + \exp(-\Delta E_i/T)); \\ P(v_i = 1|v_j = d_j; i \neq j) &= \exp(-\Delta E_i/T)/(1 + \exp(-\Delta E_i/T)); \end{aligned} \tag{2.9}$$

Portanto, podemos assumir que Geman e Geman apresentam em [19] um teorema segundo o qual a associação entre o algoritmo de arrefecimento simulado e uma rede binária de Hopfield (em uma simulação sequencial) garante, quando o número de passos de atualização tende ao infinito, a convergência para algum ponto de D^n que torna a energia da rede globalmente mínima.

A simulação sequencial, que representa a evolução de uma rede de Hopfield estocástica, é apresentada através do Algoritmo 3 [2].

2.1.3 Redes de Hopfield de Alta Ordem

Nas subseções anteriores vimos que a função de energia que determina o comportamento de uma rede binária de Hopfield é expressa através da Equação 2.3. Esta equação especifica a influência das ligações entre pares de neurônios e do comportamento de cada neurônio na dinâmica da rede. Entretanto, os problemas que apresentaremos neste trabalho e em geral, grande parte dos problemas pertencentes a classe NP-difícil, ao serem mapeados para problemas de minimização de energia, geram redes que sofrem a influência de ligações entre três ou mais neurônios (Figura 2.2). Tais ligações são chamadas de ligações de alta ordem.

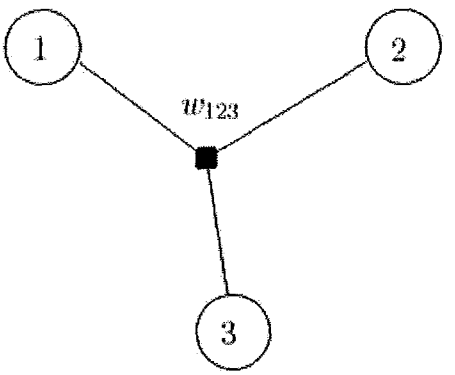


Figura 2.2: Ligação de alta ordem: Número de neurônios participantes, ou *aridade* = 3.

Uma ligação é dita de **Alta Ordem** se nela figuram três ou mais vértices simultaneamente. Analogamente, uma rede, em particular uma rede de Hopfield, é dita de alta ordem se ela apresentar uma ou mais ligações de alta ordem (Figura 2.3).

A Equação 2.10 apresenta uma possível forma para uma função de energia que representa a evolução de uma rede de Hopfield de alta ordem.

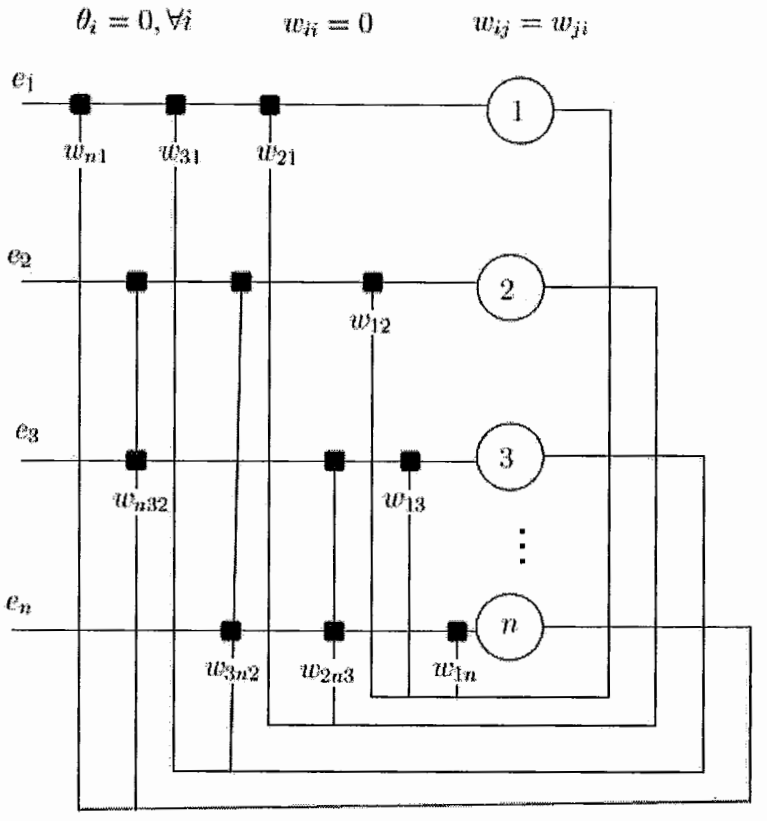


Figura 2.3: Arquitetura de uma rede binária de Hopfield com n neurônios e uma ligação de alta ordem entre os neurônios 2, 3 e n. As arestas omitidas representam ligações com peso nulo.

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\cdots\sum_{k=1}^{n} w_{ij\ldots k}v_i v_j \ldots v_k - \sum_{i=1}^{n} e_i v_i + \sum_{i=1}^{n} \theta_i v_i \qquad (2.10)$$

Observe que a energia apresentada na Equação 2.10 pode ser descrita através de potenciais $\mathbb{V}_C$ (Equação 2.4). Neste caso, a evolução probabilística mostrada na Subseção 2.1.3 permite que mínimos globais de energia sejam encontrados em redes de alta ordem.

2.2 SMEM: Mapeamento de Satisfatibilidade para Minimização de Energia

Nesta seção falaremos sobre a estratégia mapeamento de satisfatibilidade para minimizaçãao de energia. Esta estratégia pode ser utilizada no mapeamento de problemas, especificados por conjuntos de restrições, em uma função de energia. Neste caso, uma rede de Hopfield pode ser utilizada na busca do valor mínimo da função obtida. Na Subseção 2.2.1 (baseada em [20]) falaremos de alguns conceito relacionados à lógica proposicional e definiremos o problema da satisfatibilidade e na Subseção 2.2.2 descreveremos o mapeamento realizado pela estatégia SMEM, a partir da definição dos operadores H e H^*.

2.2.1 Lógica Proposicional: Conceitos Básicos

A Linguagem da Lógica Clássica Proposicional

A linguagem proposicional é uma linguagem formal que busca representar partes do discurso de forma clara (sem ambigüidades) e precisa.

Definição 2.8. *Um **Alfabeto Proposicional** é um conjunto de símbolos que se subdivide em três classes.*

Conectivos ou Operadores Lógicos*: Apresentados na Tabela 2.2.1;*

Símbolos Auxiliares*: "(" e ")";*

Símbolos Proposicionais*: Qualquer letra maiúscula, sozinha ou acompanhada por um índice (por exemplo: A, B,..., Z, A_1, B_j).*

Tabela 2.1: Conectores Lógicos.

Símbolo	Significado
$\neg$	Negação
$\wedge$	Conjunção
$\vee$	Disjunção
$\rightarrow$	Implicação
$\leftrightarrow$	Equivalência

As sentenças mais simples, denominadas proposições atômicas, são representadas por símbolos proposicionais do alfabeto apresentado na Definição 2.8. As sentenças mais complexas são construídas com o auxílio dos operadores lógicos.

Definição 2.9. *Uma **Fórmula bem Formada** φ, ou simplesmente fórmula, pode ser definida, indutivamente, pelas seguintes condições:*

i) Qualquer símbolo proposicional é uma fórmula;

ii) Se φ_1 e φ_2 são fórmulas então $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \wedge \varphi_2)$, $(\neg\varphi_1)$, $(\varphi_1 \rightarrow \varphi_2)$, $(\varphi_1 \leftrightarrow \varphi_2)$ tambem o são;

iii) Nada mais é fórmula.

Qualquer símbolo proposicional é uma **Fórmula Atômica** ou átomo (A, B, etc.) e um **Literal** l é um átomo (literal positivo) ou sua negação (literal negativo), por exemplo A e $\neg$B.

Definição 2.10. *Uma fórmula φ está na* ***Forma Normal Conjuntiva*** *(CNF) se e somente se ela está na seguinte forma $\varphi = \varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n$, tal que $\varphi_i, i = 1...n$ é uma disjunção de literais ($\varphi_i = l_1 \vee l_2 \vee ... \vee l_m$).*

Semântica da Lógica Clássica Proposicional

A semântica da lógica proposicional consiste na atribuição de significado às fórmulas da linguagem, associando à elas um valor (*valor − verdade*) em $D_l = \{Verdadeiro\ (V), Falso\ (F)\}$. A Tabela 2.2.1 exemplifica a atribuição de valores a símbolos proposicionais.

Tabela 2.2: Tabela verdade da negação, da disjunção, da conjunção, da implicação e da equivalência.

A	B	¬ A	A ∨ B	A ∧ B	A → B	A ↔ B
F	*V*	*V*	*V*	*F*	*V*	*F*
F	*F*	*V*	*F*	*F*	*V*	*V*
V	*V*	*F*	*V*	*V*	*V*	*V*
V	*F*	*F*	*V*	*F*	*F*	*F*

A atribuição de valores-verdade aos símbolos proposicionais é realizada através de uma função de atribuição, ou simplesmente atribuição, $cv : \mathfrak{B} \rightarrow D_l$, onde $\mathfrak{B}$ é o conjunto dos símbolos proposicionais (Exemplo 2.11).

Ex.:

$$\begin{array}{l} cv(A) = F; \\ cv(B) = V. \end{array} \tag{2.11}$$

Podemos estender a atribuição cv para que ela possa ser aplicada no conjunto de todas as fórmulas da linguagem proposicional, que denotaremos por $\mathfrak{W}$. Neste contexto, definimos uma função $CV : \mathfrak{W} \rightarrow D_l$ satisfazendo às seguintes condições:

1. $CV(A) = cv(A)$, se $A \in \mathfrak{B}$.

2. $CV(\neg\varphi) = V$, se $CV(\varphi) = F$; F, caso contrário.

3. $CV(\varphi_1 \wedge \varphi_2) = V$, se $CV(\varphi_1) = CV(\varphi_2) = V$; F, caso contrário.

4. $CV(\varphi_1 \vee \varphi_2) = F$, se $CV(\varphi_1) = CV(\varphi_2) = F$; V, caso contrário.

5. $CV(\varphi_1 \rightarrow \varphi_2) = F$, se $CV(\varphi_1) = V$ e $CV(\varphi_2) = F$; V, caso contrário.

Seja φ uma fórmula e Γ um conjunto de fórmulas:

1. Uma atribuição de valores-verdade CV satisfaz φ se e somente se $CV(\varphi) = V$. Analogamente, CV satisfaz Γ se e somente se CV satisfaz cada membro de Γ;

2. Γ é satisfatível se e somente se existe uma atribuição CV que satisfaz Γ. Caso contrario, Γ é insatisfatível.

Considere uma fórmula lógica φ. O **Problema da Satisfatibilidade** (SAT) consiste em apresentar, caso exista, um conjunto de valores-verdade, que atribuídos por CV aos átomos de φ a torna verdadeira. Este conjunto de valores-verdade é chamado de *modelo* para φ. A Tabela 2.2.1 apresenta um modelo para a fórmula $\varphi = (\neg A \vee \neg B) \wedge (\neg A \vee B) \wedge (A \vee \neg B)$, através de sua tabela verdade.

Tabela 2.3: Tabela verdade da fórmula lógica φ. Neste caso o modelo para φ é $CV(A) = F$ e $CV(B) = F$.

A	B	$\neg A \vee \neg B$	$\neg A \vee B$	$A \vee \neg B$	$(\neg A \vee \neg B) \wedge (\neg A \vee B) \wedge (A \vee \neg B)$
F	V	V	V	F	F
F	F	V	V	V	V
V	V	F	V	V	F
V	F	V	F	V	F

2.2.2 SMEM: Operadores Pseudo-Booleanos

Vários trabalhos fazem uso da equivalência entre o problema de satisfatibilidade da lógica proposicional e o problema de minimização de energia, dentre eles, o de Gadi Pinkas [14] apresentou uma forma de mapear problemas complexos, especificados através de conjuntos de fórmulas lógicas, em problemas de minimização de energia. Pinkas mostrou que o mínimo global de uma função de energia, obtida a partir de um conjunto de fórmulas lógicas, é equivalente a um modelo que satisfaça este conjunto de fórmulas. Posteriormente, Lima [10] utilizou esta estratégia na modelagem do problema do caixeiro viajante. Uma outra forma de mapear satisfatibilidade para redes de autômatos é dada em [21], sendo aplicada ao problema de escalonamento de recursos.

A estratégia SMEM é composta, basicamente, por dois estágios. No primeiro estágio o conjunto de restrições que representam o problema a ser modelado é especificado através de um conjunto de fórmulas φ_i na forma normal conjuntiva e o problema original é mapeado para um problema de satisfatibilidade. O segundo estágio corresponde à aplicação de um operador H^* [6] sobre a negação das fórmulas lógicas que definem um problema de satisfatibilidade e que foram geradas

[6] O operador H^* é baseado no operador H, cuja definição apresentaremos a seguir (Equações 2.12).

no primeiro estágio. O operador H^* mapeia nossas fórmulas em uma função de energia, determinando um nova transformação no problema original, que passa a ser um problema de minimização de energia.

Considere um problema P especificado através de uma função de energia E, que depende de um conjunto $\mathbb{V}$ de variáveis aleatórias $v_i, i = 1...n$ e cujo conjunto de mínimos corresponde ao conjunto de soluções de P. Um **Problema de Minimização de Energia** consiste em determinar um conjunto de valores em um domínio D, que atribuídos às variáveis em $\mathbb{V}$ tornam a função E globalmente mínima.

O conjunto de regras que especificam o mapeamento realizado pelo operador H é apresentado através do conjunto de Equações 2.12.

$$\begin{aligned} &H(V) = 1; \\ &H(F) = 0; \\ &H(A) = A; \\ &H(\neg A) = 1 - H(A); \\ &H(A \wedge B) = H(A) \times H(B); \\ &H(A \vee B) = H(A) + H(B) - H(A \wedge B). \end{aligned} \tag{2.12}$$

O operador H^* utiliza as mesmas regras de transição de H, exceto em fórmulas disjuntivas, pois neste caso $H^*(A \vee B) = H(A) + H(B)$. O motivo para esta mudança é que em fórmulas $\varphi = \varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n = \bigwedge_{i=1}^{n} \varphi_i$, tal que $\varphi_i = l_1 \vee l_2 \vee ... \vee l_m$, a aplicação de H sobre a negação de φ geraria um termo que cresce exponencialmente com o valor de n, pois $H(\neg\varphi) = \sum_{i=1}^{n}(-1)^{i-1}\varphi_1 \times ... \times \varphi_i$. Por outro lado, o termo obtido a partir da aplicação do operador H^* equivale a $\sum_{i=1}^{n} \varphi_i$. Neste caso, a aplicação de H^* gera um número bem menor de parcelas, se comparado ao operador H e o termo obtido a partir de sua aplicação computa o número de fórmulas φ_i que são satisfeitas (verdadeiras).

Ressaltamos que em restrições de viabilidade [7], como a satisfatibilidade de uma fórmula φ deve gerar um decréscimo de energia o operador H^* deve ser aplicado à negação de φ. Portanto, quando uma restrição for satisfeita φ será verdadeira e, conseqüentemente, $H^*(\neg\varphi)$ será igual a zero (0). Além disso, a estratégia SMEM permite que símbolos não pertencentes ao alfabeto proposicional sejam agregados às fórmulas lógicas. Estes símbolos representam valores numéricos que fornecem determinadas informações sobre o problema mapeado (P) e que no termo de energia obtido fará parte do peso neuronal da rede utilizada no processo de busca por uma solução de P. No TSP por exemplo, a restrição de otimalidade possui um símbolo $dist_{ij}$ que armazena o valor da distância entre as cidades i e j. Por outro lado, cada parcela da função de energia possui uma constante multiplicativa (α, β, etc.) que integra o peso da ligação neuronal que ela representa. Estas constantes multiplicativas representam o nível de importância, ou grau de precedência, das restrições a que elas estão associadas, em relação às demais restrições. Cada constante multiplicativa deve ser estimada buscando garantir que o custo de se infringir pelo menos uma das restrição às quais ela está associada seja sempre maior do que o decréscimo de energia correspondente ao atendimento de todas as restrições que possuem ordem de precedência menor do que ela. Em outras palavras, uma restrição não pode ser infringida, mesmo para o atendimento de todas as restrições que são menos significativas. No VCP em um grafo G, por exemplo, a constante α associada à menor precedência é igual a um limite superior para $\chi(G)$, ou seja, α é igual ao número total n de cores $+h$ [8] e as constantes subsequentes são iguais ao produto entre as constantes multiplicativas menos significativas e o número de fórmulas que

[7] Dado um problema P, especificado através de conjuntos de restrições, o conjunto de restrições de viabilidade contém as restrições que garantem a consistência de uma possível solução e o conjunto de restrições de otimalidade apresenta as restrições necessárias para que uma solução seja viável.

[8] h pequeno e maior que zero(0)

representam restrições menos significativas $+h$.

Por fim, em fórmulas φ que representam restrições de otimalidade o termo de energia é obtido através de $H^*(\varphi)$, pois o custo associado às restrições de otimalidade deve ser adicionado à função de energia. No TSP por exemplo, a restrição de otimalidade que diz que, se duas cidades ocupam posições consecutivas no percurso o custo do caminho entre elas deve ser considerado é especificada pela fórmula $\varphi_4 = \vee_i \vee_{j,j\neq i} \vee_k (dist_{ij}(v_{ik} \wedge v_{j(k+1)}))$ então $H^*(\varphi_4) = \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} \sum_{k=1}^{n} dist_{ij}(v_{ik}v_{j(k+1)}))$. Observe que este termo computa o custo dos percursos candidatos a solução do TSP em cada iteração do processo de busca.

O Exemplo 2.13 ilustra o mapeamento realizado pelo operador H^* através de sua aplicação na fórmula, que representa a restrição do problema do caixeiro viajante que determina que duas cidades não podem ocupar a mesma posição em um percurso e a Figura 2.4 apresenta a rede neuronal resultante.

Ex.:

$$
\begin{aligned}
&\forall i, \forall j, \forall k | 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n, j \neq i : \neg(v_{ik} \wedge v_{jk}) \\
&\Leftrightarrow \wedge_i \wedge_{j,j\neq i} \wedge_k \neg(v_{ik} \wedge v_{jk}) = \varphi_2 \\
&\Rightarrow \neg\varphi_2 \ = \vee_i \vee_{j,j\neq i} \vee_k (v_{ik} \wedge v_{jk}) \\
&H^*(\neg\varphi_2) = \sum_{i=1}^{n} H(\vee_{j,j\neq i} \vee_k (v_{ik} \wedge v_{jk})) \\
&= \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} H(\vee_k (v_{ik} \wedge v_{jk})) \\
&E_{\varphi_2} = \beta \times \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} \sum_{k=1}^{n} v_{ik}v_{jk}
\end{aligned}
\tag{2.13}
$$

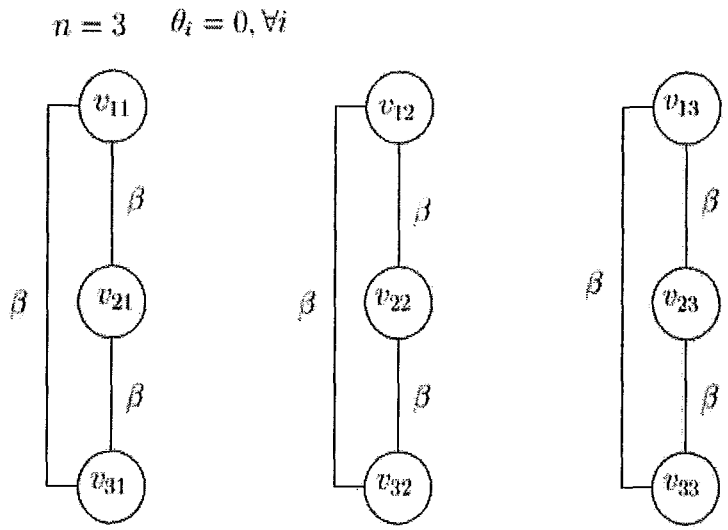


Figura 2.4: Parte da rede neuronal que simula o problema do caixeiro viajante (E_{φ_2}).

2.3 Geometria Molecular: A Predição de Conformações Moleculares Estáveis

Nesta seção descreveremos dois modelos matemáticos para o problema da predição das conformações moleculares estáveis. Estes modelos serão combinados, no Capítulo 4, dando origem a um novo modelo para o SMCPP.

Na Subseção 2.3.1 apresentaremos algumas considerações iniciais sobre modelagem clássica, quântica e geométrica e falaremos das vantagens e desvantagens inerentes à aplicação de cada uma delas. Na Subseção 2.3.2, descreveremos um modelo clássico para o SMCPP que consiste na geração de uma função potencial baseada na teoria da mecânica clássica [22]. Em seguida, na Subseção 2.3.3, enunciaremos o problema da geometria das distâncias moleculares segundo [23] e descreveremos um dos modelos que o representa, apresentado em [24].

2.3.1 Considerações Iniciais

O problema da predição de conformações moleculares estáveis pode ser representado através de modelos baseados na mecânica clássica ou na mecânica quântica. No primeiro caso, os modelos matemáticos obtidos são baseados fundamentalmente na teoria atômica de Rutherford-Bohr e são utilizados na predição da geometria, ou conformação estável, de moléculas grandes, devido ao baixo custo computacional. Por outro lado, os modelos quânticos são construídos, principalmente, a partir das teorias de De Broglie, Heisenberg e Schrodinger, sendo utilizados nos casos em que é necessário uma precisão maior na predição da estrutura molecular. Entretanto, o custo computacional associado a modelos quânticos é bem mais alto, em função do elevado número de informação que devem ser consideradas.

Existem casos em que o números de componentes do sistema molecular e o nível de complexidade da interação entre estes componentes, faz com que ele se encontre em uma faixa de transição entre o domínio clássico e quântico e em outros casos é necessário analisar partes de um mesmo composto químico sob a dinâmica quântica, devido ao grau de precisão que se quer alcançar em determinadas partes do composto. Nestes casos, a geometria destes sistemas moleculares, seria descrita mais precisamente pela composição de leis da teoria da mecânica quântica e clássica. Todavia, limitações computacionais associadas ao custo relativo ao elevado número de informações que devem ser analisadas em um modelo quântico faz com que muitos trabalhos tenham como base apenas a representação clássica. Em particular, a representação clássica baseada em leis físicas, pois ela pode ser considerada relativamente simples.

A representação física do problema da predição da geometria molecular é feita através de uma função, denominada potencial, que descreve de forma clássica as interações entre as particulas envolvidas no sistema (podemos encontrar alguns exemplos de funções potenciais em [25] e em [26].

Outra forma de representar o problema da predição de conformações moleculares estáveis é através de modelos baseados na alocação de um conjunto de pontos, representando átomos, no espaço euclidiano. Estes modelos consistem, fundamentalmente, na predição de ângulos de torção a partir de um conjunto de distâncias, representando ligações covalentes, entre pares de pontos consecutivos. Neste caso, o problema tratado é uma variante do SMCPP e chama-se problema da geometria das distâncias moleculares (MDGP) [24, 27]. Uma das vantagens desta forma de tratar o SMCPP é o baixo custo computacional, se comparado às abordagens citadas no início desta subseção, pois os átomos são vistos como pontos no espaço, reduzindo o número de informações que devem ser analisadas e determinadas durante a busca por uma solução do problema. Entretanto, como este modelo é construído a partir de um conjunto de informações relativamente símples e que, por isso, não reflete de forma precisa a natureza do composto químico cuja geometria deve ser determinada, para cada instância do MDGP há um conjunto de formas espaciais que são soluções ótimas, mas apenas uma delas é uma solução ótima para a instância do SMCPP correnpondente. Portanto, o problema fundamental desta abordagem é encontrar, dentre suas soluções ótimas, a solução ótima do SMCPP correspondente. Uma alternativa seria adicionar termos clássicos e ou quânticos a este modelo.

2.3.2 Dinâmica Molecular: Modelo Clássico

Nesta subseção descreveremos um modelo clássico inspirado em [28] e em [29]. Este modelo utiliza uma representação física baseada em osciladores harmônicos, descrevendo o comportamento de sistemas moleculares como se as interações entre as partículas que o compõem fossem essencialmente elásticas. O principio básico deste modelo é que muitos comportamentos oscilatórios expressam a ação de forças restauradoras que tendem a trazer ou manter o sistema, sobre o qual elas atuam, no seu estado de equilíbrio. Tais forças restauradoras são basicamente do tipo elásticas, obedecendo, portanto, a lei de Hooke.

De acordo com a lei de Hooke, a intensidade da força elástica resultante F, que atua sobre um sistema, é diretamente proporcional ao deslocamento $\Delta pos = pos_s - pos_0$, onde pos_0 representa a posição inicial (ou de equilíbrio) do sistema no instante $s = 0$ e pos_s representa a posição do sistema em um instante $s > 0$. Matematicamente temos $F = K \times \Delta pos$, onde K é o fator de restauração. Um sistema que se comporta desta forma é o sistema massa-mola (Figura 2.5), que consiste em uma mola de fator de restauração, ou constante elástica K, presa por uma das extremidades a um ponto fixo e pela outra extremidade a um corpo rígido com massa de valor m.

No sistema massa-mola o ponto de equilíbrio pos_0 é, geralmente, fixado na origem ($pos_0 = 0$) e sempre que há uma tentativa de tirar o sistema deste ponto, surge uma força restauradora F que tende a trazê-lo de volta ao ponto de equilíbrio.

O comportamento dos comprimentos das ligações químicas covalentes pode ser descrito por uma função potencial aproximada a de Hooke, pois cada um destes

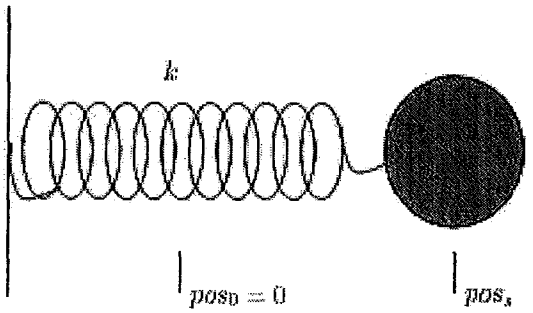


Figura 2.5: Sistema massa-mola.

comprimentos oscila próximo ao seu valor de equilíbrio, seguindo a mesma dinâmica de um sistema de massas unidas por molas. Na Figura 2.6 apresentamos um exemplo da ligação covalente, em um sistema molecular SM, cujo modelo é baseado em interações elásticas. As linhas pontilhadas representam os valores de distância que oscilam próximo ao valor de equilíbrio, especificado através da linha sólida.

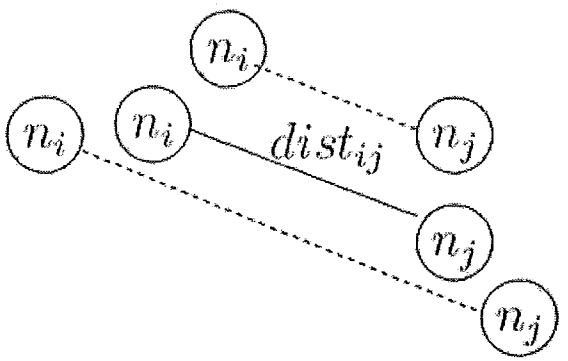


Figura 2.6: Exemplo da ligação covalente.

Outra classe de interações que pode ser descrita por um potencial harmônico é a variação dos ângulos entre pares de ligações covalentes consecutivas e dos ângulos de torção. Neste caso, K representa a constante de Hooke para a restituição do ângulo de equilíbrio α_{ikj} entre cada par de ligações covalentes $(a_i, a_k), (a_k, a_j) \in L$ e do ângulo de torção $\alpha_{ikj'}$. A Figura 2.7 mostra um exemplo de ângulo entre ligações covalentes consecutivas e a Figura 2.8 mostra um exemplo de ângulo de torção

$\alpha_{ikj'}$ entre os planos definidos pelos vértices n_i, n_k, n_j e $n_k, n_j, n_{j'}$. O equilíbrio é estabelecido por forças elásticas e os arcos pontilhados representa os ângulos que oscilam próximo aos valores de equilíbrio, especificados através dos arcos sólidos.

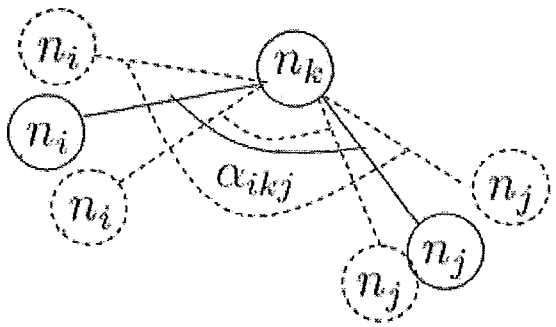


Figura 2.7: Exemplo de ângulo entre ligações covalentes consecutivas.

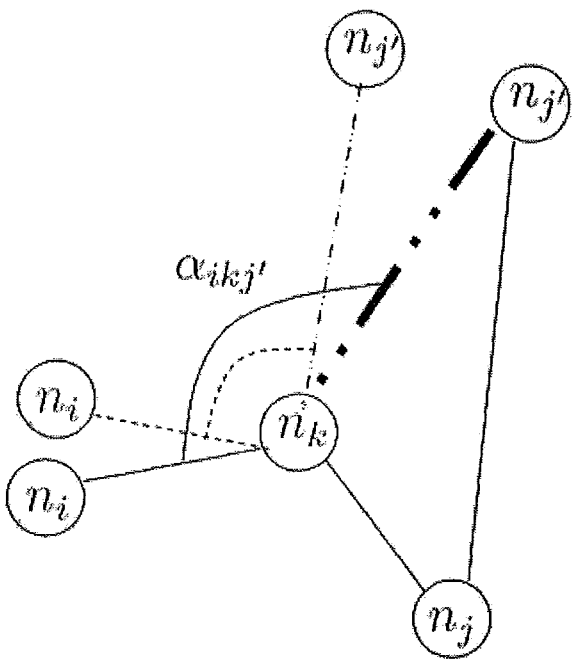


Figura 2.8: Exemplo de ângulo de torção.

As interações que acabamos de descrever dão origem aos seguintes termos:

Termos Harmônicos - Descrevem ligações covalentes entre pares de átomos e ângulos entre ligações covalentes vizinhas;

Termo Torcional - Descreve rotações em torno de ligações covalentes.

Além disso, temos alguns termos que descrevem as interações polares e, ou iônicas que estejam influenciando a dinâmica do composto químico cuja geometria deve ser estimada [9]. São eles:

Termo Relativo às Forças de Dispersão e Indução Dipolar - Descreve a influência mútua entre átomos que induzem a polarização da região do composto químico em que a interação entre eles ocorre (Figura 2.9).

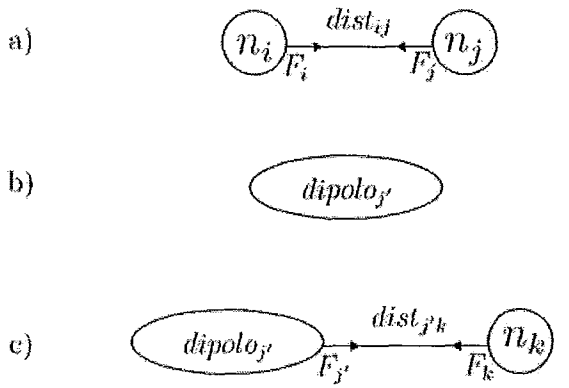


Figura 2.9: Interação polar: a) Átomos com diferença de eletronegatividade se atraem; b) Tais átomos se unem formando um dipolo; c) Este dipolo pode, por sua vez, interagir com outros átomos ou dipolos.

Termo Relativo às Atrações e Repulsões Eletrostáticas - Descreve interações iônicas, considerando a relação entre a carga elétrica dos átomos ionizados (Figura 2.10).

Matematicamente, este modelo físico pode ser representado através da energia

[9]Estes termos descrevem interações entre átomos não ligados, sob a hipotese da impenetrabilidade das nuvens eletrônicas.

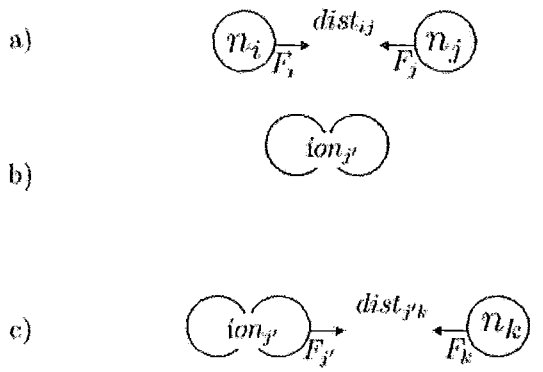


Figura 2.10: Interação iônica: a) Átomos com grande diferença de eletronegatividade se atraem; b) A diferença de eletronegatividade é tão marcante que os átomos se ionizam; c) O íon resultante pode, por sua vez, interagir com outros átomos ou íons.

potencial $E = E_{covalente} + E_{plano} + E_{torcao} + E_{polar} + E_{eletrostatica}$.

$$E_{covalente} = \frac{1}{2} \times K_{ij}(dist_{ij} - dist_{ij}^0)^2 \tag{2.14}$$

Na Equação 2.14 K_{ij} é a constante de Hooke associada à ligação covalente (a_i, a_j), $dist_{ij}$ é o comprimento da ligação em um instante qualquer e $dist_{ij}^0$ representa o comprimento de equilíbrio da ligação.

$$E_{plano} = \frac{1}{2} \times K'_{ij}(\alpha_{ij} - \alpha_{ij}^0)^2 \tag{2.15}$$

Na Equação 2.15 K'_{ij} é a constante de Hooke para a restituição do ângulo de equilíbrio entre duas ligações covalentes consecutivas (a_i, a_k) e (a_k, a_j), α_{ij} é o ângulo em um instante qualquer e α_{ij}^0 representa o ângulo de equilíbrio.

$$E_{torcao} = \frac{1}{2} \times K_{ikj} \times [1 + cos(n \times \theta - \delta)] \tag{2.16}$$

Em geral, o termo potencial associado a predição dos ângulos de torção consiste no termo mais baixo de uma expansão co-seno [22] (Equação 2.16). Segundo Pascutti [29] K_{ikj} é a constante que define a altura da barreira de rotação, n é o número de mínimos para a torção de uma ligação química específica, θ é o ângulo diedral para a ligação central em uma seqüência de quatro átomos e δ é a defasagem no ângulo diedral.

A Equação 2.17 é determinada pelo termo relativo às forças de dispersão e indução dipolar e é especificado através do potêncial de Lennard-Jones. A variável ϵ representa o limite entre a barreira atrativa e a repulsiva e $dist_{ij}^{0}$ representa a distância de equilíbrio entre os átomos a_i e a_j.

$$E_{polar} = 4 \times \epsilon \times [(\frac{dist_{ij}^{0}}{dist_{ij}})^{12} - (\frac{dist_{ij}^{0}}{dist_{ij}})^{6}] \tag{2.17}$$

A Equação 2.18 representa o termo relativo às atrações e repulsões eletrostáticas e é determinado pela lei de Coulomb. As variáveis q_i e q_j representam as cargas dos átomos a_i e a_j.

$$E_{eletrostatica} = \frac{(q_i \times q_j)}{4 \times \pi \times dist_{ij}} \tag{2.18}$$

2.3.3 O Problema da Geometria das Distâncias Moleculares

Neste trabalho, trataremos o problema da geometria das distâncias moleculares como sendo a predição da estrutura tri-dimensional de compostos químicos, baseada

em um conjunto esparso de distâncias, obtidas a partir da ressonância magnética nuclear [23].

Consideremos um sistema molecular $SM = (\mathcal{A}, L)$, tal que $\mathcal{A}$ é um conjunto de átomos a_i e L é o conjunto de ligações entre pares de átomos a_i e a_j em $\mathcal{A}$. Podemos associar o sistema molecular SM a um grafo $G = (V, E)$ de forma que cada vértice n_i em V represente um átomo a_i em $\mathcal{A}$ e as arestas $(n_i, n_j) \in E$ representem as ligações $(a_i, a_j) \in L$. Neste caso, podemos, na Definição 2.11, apresentar formalmente o problema da geometria das distâncias moleculares.

Definição 2.11. *Seja $G = (V, E)$ o grafo associado a um sistema molecular SM. Tomemos um subconjunto E_c de E que contenha todos e apenas os pares $(n_i, n_j) \in E$ que representem ligações covalentes $(a_i, a_j) \in L$, de tal forma que possamos definir um conjunto S, cujos elementos são distâncias $dist_{ij} \in \mathbb{R}$ associadas a cada uma das ligações covalentes representadas em E_c.*

O Problema da Geometria das Distâncias Moleculares *consiste na predição de um conjunto de coordenadas $(x_i, y_i, z_i) \in \mathbb{R}^3$, que determinam a disposição espacial de cada $n_i \in V$, tal que $|\|(x_i, y_i, z_i) - (x_j, y_j, z_j)\| - dist_{ij}| \leq \varepsilon$ para todo $(n_i, n_j) \in E_c$ e para algum $\varepsilon \geq 0$.*

Barbosa, Lavor e Raupp apresentam em [24] um modelo para o MDGP no qual consideram uma sequência $S_n = (n_1, ..., n_n)$ de pontos em $\mathbb{R}^3$, tal que a distância entre pontos consecutivos (n_i, n_{i+1}) é a distância euclidiana associada à ligação covalente $(a_i, a_{i+1}) \in L$, cada três pontos consecutivos (n_i, n_{i+1}, n_{i+2}) determinam um ângulo $\alpha_{ii+1i+2}$ entre ligações covalentes consecutivas e cada quatro pontos consecutivos $(n_i, n_{i+1}, n_{i+2}, n_{i+3})$ determinam um ângulo $\alpha_{ii+1i+3}$ entre os planos Π_1 e Π_2, determinados pelos átomos n_i, n_{i+1}, n_{i+2} e $n_{i+1}, n_{i+2}, n_{i+3}$, respectivamente. O ângulo

$\alpha_{ii+1i+3}$ é chamado ângulo de troção e é determinado através do ângulo entre as normais que cortam Π_1 e Π_2.

Todas as distâncias associadas a ligações covalentes e todos os ângulos entre ligações covalentes consecutivas são fixados em seu valor de equilíbrio. Neste caso, a energia potencial é uma função dos ângulos de torção $\alpha_{ii+1i+3}, i = 1...n-3$, ou seja, $E = f(\alpha_{124}, ..., \alpha_{n-3n-2n})$. Portanto, o problema original se transforma em encontrar o conjunto de valores que atribuídos a $(\alpha_{124}, ..., \alpha_{n-3n-2n})$ tornam a função E globalmente mínima. Neste caso, ε será considerado como $|E_{calculada} - E_{otima}| = 0.0816608225$.

Capítulo 3

SATyrus: Uma Estratégia Híbrida para o Tratamento de Problemas NP-Difíceis

3.1 O Sistema SATyrus

A implementação da estratégia híbrida que combina a estratégia SMEM e redes de Hopfield resultou no sistema computacional SATyrus, que é composto por dois módulos principais. O primeiro módulo consiste em um compilador, desenvolvido por Morveli-Espinoza e apresentado em [30] e em [12]. Este compilador converte um conjunto de restrições lógicas, especificadas através de uma linguagem especial, definida por Lima e Morveli-Espinoza, em uma função de energia, que representa uma rede neuronal de Hopfield, possivelmente de alta ordem. O segundo módulo, um simulador neuronal, desenvolvido e apresentado através deste trabalho e em [12], é responsável pela construção de uma rede neuronal, a partir da função de energia compilada e pela simulação estocástica da evolução da rede no tempo, ou passos de atualização, através do algoritmo de busca global arrefecimento simulado combinado

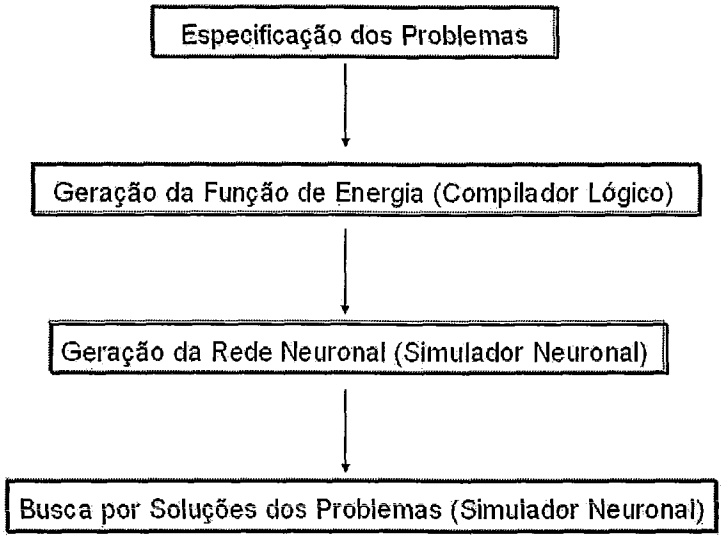


Figura 3.1: Processo de modelagem, compilação e simulação dos problemas.

à distribuição de probabilidade de Boltzmann-Gibbs.

A idéia principal consiste em facilitar a especificação de problemas complexos através de redes neuronais. Portanto, o processo que se inicia com a definição do conjunto de restrições que especificam tais problemas e que termina com a obtenção de uma solução para os mesmos, pode ser resumido através da Figura 3.1.

Inicialmente, cada problema P é especificado através de um conjunto de restrições essencialmente lógicas (criação dos arquivos de entrada para o compilador). Estas restrições são compiladas para uma função de energia por um processo que simula o mapeamento realizado pela estratégia SMEM. Em seguida, a função de energia resultante (entrada para o simulador) é utilizada na geração da rede neuronal cuja evolução determina a busca por uma das soluções de P e, finalmente, a busca é realizada e uma solução, possivelmente ótima, é apresentada.

3.2 O Compilador

O compilador lógico, desenvolvido em [30], é responsável pelo mapeamento de problemas sob a forma de restrições para uma função de energia. Tais restrições são especificadas através de uma combinação entre fórmulas lógicas e constantes multiplicativas que refletem determinadas características do problema em estudo.

Falaremos da linguagem SATyrus, ou seja, da linguagem reconhecida pelo compilador e que deve ser utilizada na composição do arquivo de restrições que especificam cada problema, ao descrevermos as entradas do sistema.

3.2.1 Arquivo de Restrições (Principal)

O arquivo de restrições contém a descrição do problema que deve ser mapeado, sob a forma de fórmulas essencialmente lógicas. Ele se divide em três partes:

1. Definição das estruturas neuronais que compõem a rede;

   Este trecho engloba a declaração das estruturas neuronais e das variáveis que auxiliarão no processo de compilação. A declaração de cada uma delas envolve a atribuição de valores que determinam, por exemplo, o tamanho das dimensões de uma estrutura definida por uma matriz bi-dimensional. Existem três formas de atribuição:

   (a) Atribuição direta de um valor inteiro ou de uma soma ou subtração, de valores ou variáveis, a uma variável (Exemplo 3.1).

Ex.:

$$\begin{aligned} &num1 = 4; \\ &num2 = num1 - 1; \\ &num3 = num1 + num2; \end{aligned} \tag{3.1}$$

(b) Atribuição de um valor inteiro, de uma variável, de uma soma ou subtração (de valores ou variáveis) ou de um intervalo pertencente a $\mathbb{N}^*$ a dimensão de uma estrutura (Exemplo 3.2). Existem algumas variações e extensões destas declarações e de todas as outras que serão apresentadas, como por exemplo ($pos(1 <= i <= 4,\ 1 <= j <= 5; i \neq j);$). Entretanto, nos limitaremos a descrever as formas principais. Para maiores detalhes consulte [30].

Ex.:

$$\begin{aligned} &pos(4,4); \\ &num1 = 4; \\ &pos(num1, num1); \\ &pos(num1 - 2, 4); \\ &pos(1 <= i <= 3, 1 <= j <= 6); \end{aligned} \tag{3.2}$$

(c) Atribuição de valores a estruturas auxiliares que representam matrizes de valores inteiros e que armazenam informações sobre a natureza do problema estudado, associadas aos pesos neuronais. Esta atribuição é feita através de um arquivo texto (.txt)(Exemplo 3.3).

Ex.:

$$dist(num1, num1);$$
$$dist\ read\ from\ tsp1.txt; \tag{3.3}$$

2. Descrição do conjunto de restrições que especificam o problema;

A especificação dos problemas é feita a partir de dois tipos de restrições, de viabilidade e de otimalidade. Neste caso, devemos utilizar a palavra reservada *group*, precedida pelas palavras reservadas *integrity*, para indicar se determinada restrição é de viabilidade, ou *optimality*, para indicar se determinada restrição é otimalidade. A palavra reservada *type*, seguida pelo nome da precedência, define o identificador da precedência que será atribuida a restrição que estiver sendo declarada. Este bloco de declarações deve ser finalizado com o sinal (:). A palavra reservada $forall$, seguida por uma lista de índices separados por vírgula e entre chaves, estabelece os índices, ou indexadores, que serão utilizados na restrição corrente. Em seguida, deve ser declarada a lista de intervalos a que pertence cada indexador, finalizando este bloco de declarações com o sinal (:)(Exemplo 3.4).

Ex.:

$$integrity\ group\ type\ int1 :\ foralli, j;$$
$$1 <= i <= num, 1 <= j <= num :\ pos[i][j]; \tag{3.4}$$

A última parte da especificação de uma restrição consiste na definição da fórmula lógica que a representa. Analisando o Exemplo 3.5 podemos observar que cada literal da fórmula é representado pelo identificador da posição que ele ocupa na estrutura que o representa (ex.: $pos[i][j]$), precedido pela palavra

reservada *not*, caso seja um literal negativo. As palavras reservadas *or* e *and* representam disjunções e conjunções e admite-se a utilização de constantes multiplicativas que farão parte de pesos neuronais e que, no Exemplo 3.5, representamos através da variável $dist[i][k]$.

Ex.:

$$\begin{array}{l} (not\ pos[i][j]\ or\ not\ pos[k][j]); \\ dist[i][k]\ (pos[i][j]\ and\ pos[k][j+1]); \end{array} \tag{3.5}$$

3. Definição do nível de prioridade de cada restrição;

   O nível de prioridade define o grau de importancia de cada restrição em relação às demais. Cada penalidade, ou precedência, recebe um identificador na declaração das restrições e, neste ponto, é definido seu grau de prioridade, através de números em $\mathbb{N}^*$, como mostra o Exemplo 3.6. A mais baixa prioridade recebe nível 0.

   Ex.:

$$\begin{array}{l} penalty\{ \\ wta\ is\ level\ 2; \\ int1\ is\ level\ 1; \\ costo\ is\ level\ 0;\} \end{array} \tag{3.6}$$

A Figura 3.2 apresenta o arquivo de restrições utilizado no TSP.

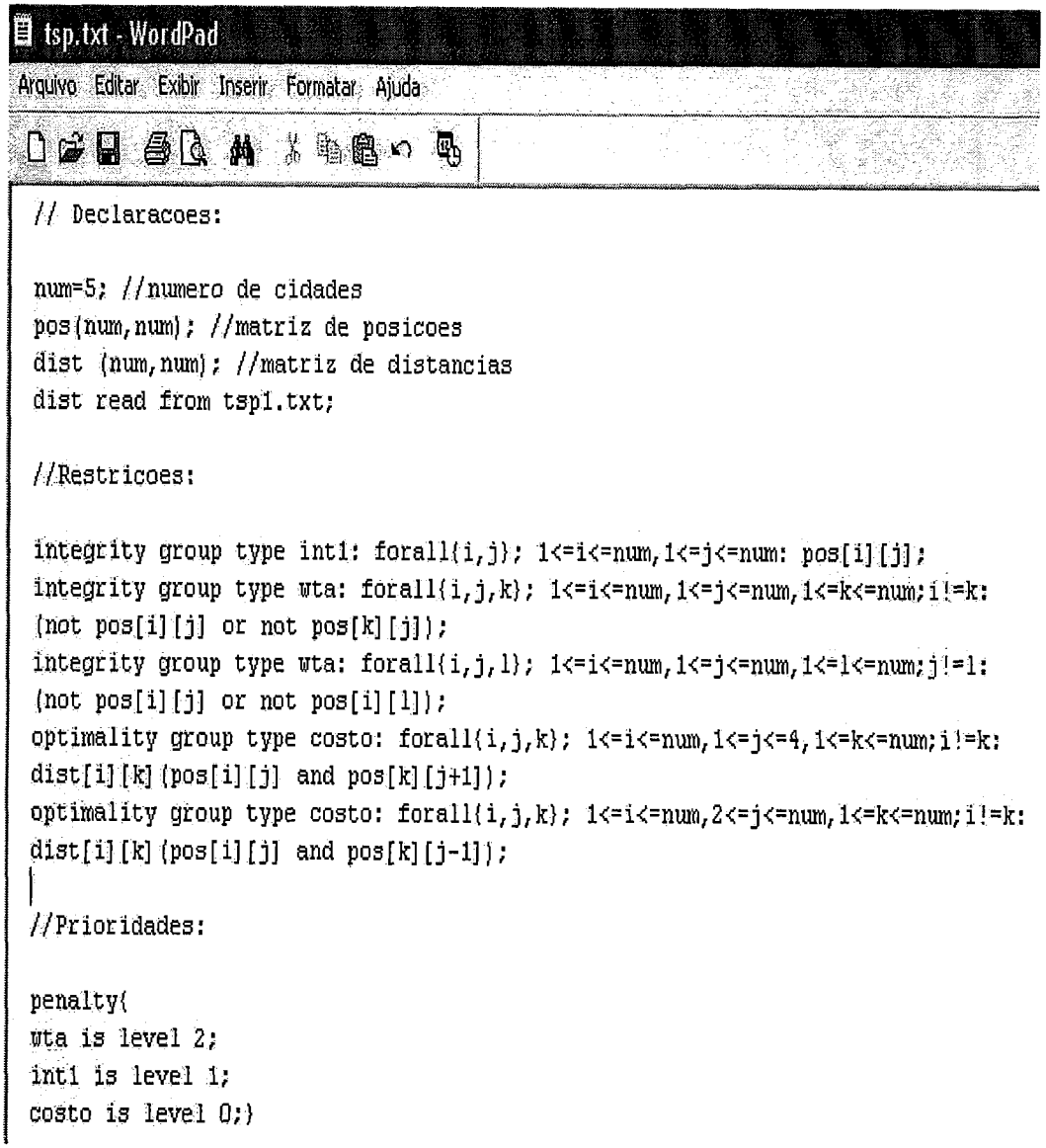


Figura 3.2: Arquivo de restrições do problema do caixeiro viajante (TSP).

3.2.2 Arquivos de Inicialização Neuronal

Os arquivos de inicialização neuronal apresentam um conjunto de valores de inicialização que podem ser atribuídos aos neurônios da rede. A especificação destes valores é feita através da 4-upla (coordenadas do neurônio na estrutura em que ele foi declarado - seu estado inicial - se ele é fixo ou não - sua freqüência de operação). As opções de inicialização são apresentadas a seguir.

- O estado inicial do neurônio (0 ou 1). Este atributo pode ser inicializado através de um menu que é apresentado ao usuário no início da compilação;
- Se ele deve ou não se atualizar (ser fixo $\rightarrow$ 1 ou não $\rightarrow$ 0 e por default é inicializado com valor 0);
- A freqüência de operação, ou seja, quais as chances que o neurônio terá de se atualizar em relação aos demais neurônios da rede. Este valor pertence a $\mathbb{N}^*$ e por default é inicializado com valor 1.

A Figura 3.3 apresenta um exemplo de arquivo de inicialização neuronal utilizado no TSP. O nome do arquivo deve ser igual ao nome da estrutura que, no exemplo, se chama *pos* e foi declarada no arquivo de restrições da Figura 3.2.

Figura 3.3: Arquivo de inicialização da estrutura neuronal *pos*.

3.3 O Simulador

O simulador neuronal, implementado durante a realização deste trabalho, é composto por dois estágios. No primeiro estágio são criados os neurônios que compõem a rede de Hopfield, a partir de um conjunto de informações de inicialização associado ao problema P que se deseja simular. Esta rede será utilizada na busca por uma solução de P e o conjunto de informações de inicialização é determinado pelo usuário antes da compilação e armazenado pelo compilador (descrito na Seção 3.2). Posteriormente, a função de energia gerada pelo compilador é percorrida e cada uma de suas parcelas dá origem a ligações neuronais que completam a geração da rede. Esquematicamente temos:

1. O simulador recebe uma lista com o número de neurônios da rede, que nesta seção denotaremos por n, com seu nome e com seu estado. Com estas informações, cada neurônio é criado e armazenado em um vetor $Network$ que representa a rede neuronal (Figura 3.4).

2. A função de energia E é recebida e cada uma de suas parcelas é percorrida, de forma que, tomando como exemplo $(E = \alpha \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} v_i v_j + \beta \sum_{i=1}^{n} v_i)$ com $\alpha = 500$ e $\beta = 1000$, os neurônios possam armazenar a representação das conexões que completam a estrutura da rede.

   Cada neurônio armazena a representação das conexões, das quais ele participa, através de seu ΔE (Figura 3.5) e esta representação será utilizada posteriormente, durante o processo de simulação, em sua decisão local de atualizar ou não seu estado.

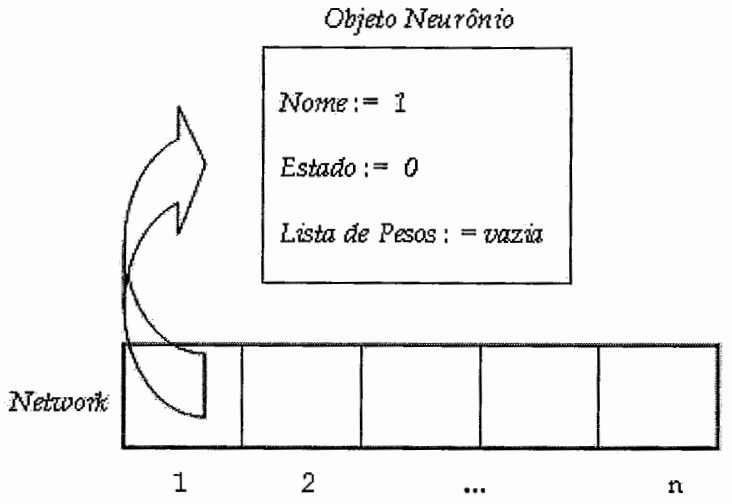


Figura 3.4: Esquema de geração dos neurônios da rede.

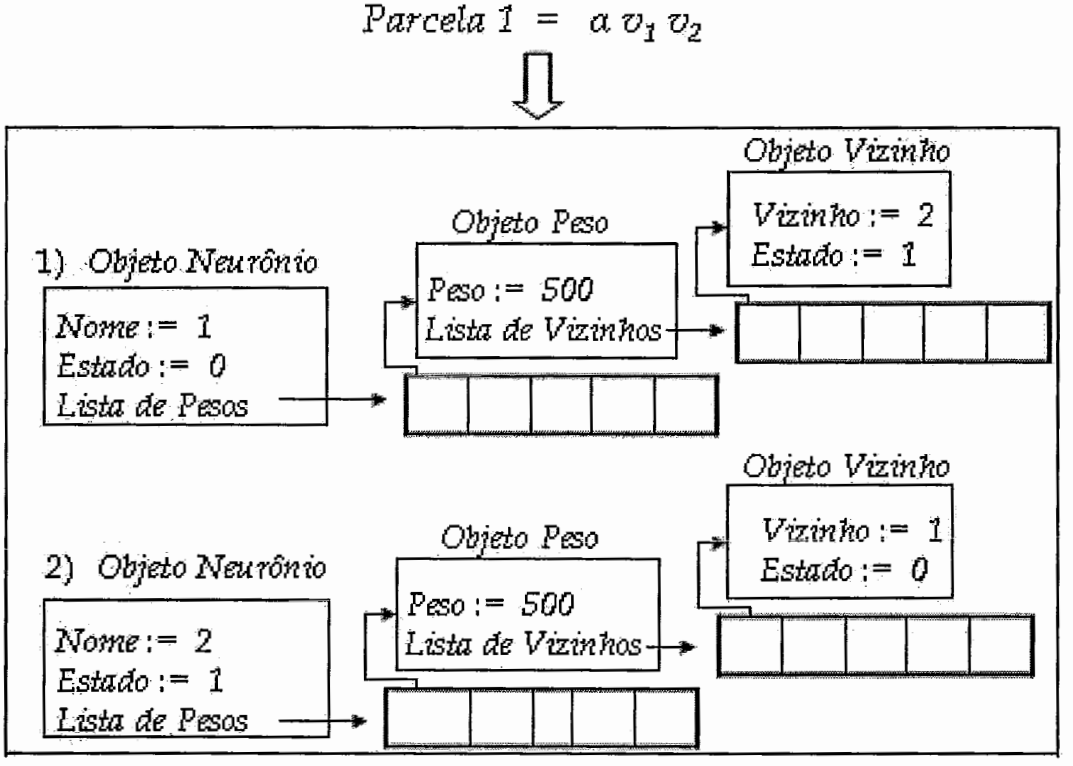


Figura 3.5: Esquema de geração da representação das ligações neuronais.

No segundo estágio, através da simulação da evolução da rede de Hopfield criada no primeiro estágio, um mínimo global de energia, que representa uma solução ótima para P, é procurado. Esta busca é realizada a partir da implementação do Algoritmo 3, descrito no Capítulo 2.

3.4 Mapeamento de Problemas NP-Difíceis para Minimização de Energia

Nesta seção ilustraremos a aplicação da estratágia SMEM, descrevendo a modelagem de três problemas NP-difíceis.

3.4.1 O Problema do Caixeiro Viajante

Seja $G = (V, E)$ um grafo não direcionado. Associemos os vértices $i, j \in V$ a cidades e cada aresta $(i, j) \in E$ a um caminho entre as cidades i e j, onde $dist_{ij}$ é a distância entre as cidades i e j, ou seja, é o custo associada à aresta $(i, j) \in V$ e $|V| \geq 3$. O problema do caixeiro viajante consiste em determinar o ciclo Hamiltoniano de custo mínimo em G. Entretanto, o conjunto de restrições utilizado, neste trabalho, para especificar o TSP não determina que o percurso inicie e termine na mesma cidade, por isso, para garantir que um ciclo seja obtido, utilizaremos o seguinte artifício:

> "Uma das cidades (A) deve ser replicada (inclusive os custos associados a ela) e fixada como a primeira cidade no percurso. Sua réplica A', por sua vez, será fixada como a última cidade no percurso.
>
> (...) O custo entre A e A' será considerado como 0 (zero)." [10], página 73.

Nesta subseção utilizaremos $n = |V| + 1$, pois neste caso, em um problema com $|V|$ cidades, utilizaremos na verdade $|V| + 1$ cidades, pois há uma cidade extra.

A rede neuronal que utilizaremos na simulação do TSP será constituída de uma estrutura com $n \times n$ neurônios, que será uma matriz bidimensional. Nesta estrutura as linhas representam as cidades e as colunas representam as possíveis posições de cada cidade no percurso. Além disso, utilizaremos uma matriz auxiliar $n \times n$, chamada *dist*, para armazenar as distâncias associadas as arestas em E. Seus componentes farão parte de pesos neuronais.

Mapeamento de Restrições para Satisfatibilidade

Nossa rede neuronal será constituída por neurônios binários v_{ik} , onde i representa uma cidade em V e k representa a posição ocupada pela i-ésima cidade no percurso e o comportamento da rede, que será o reflexo do comportamento de cada neurônio, será especificado através de um conjunto de restrições de viabilidade e de um conjunto de restrições de otimalidade.

- Restrições de Viabilidade:

  i) Todas as n cidades devem participar do percurso:

  $\forall i, \exists k | 1 \leq i \leq n, 1 \leq k \leq n : v_{ik} \quad \Leftrightarrow \quad \wedge_i \vee_k v_{ik} = \varphi_1$

  ii) Duas cidades não podem ocupar a mesma posição no percurso:

  $\forall i, \forall j, \forall k | 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n, j \neq i : \neg(v_{ik} \wedge v_{jk})$

  $\Leftrightarrow \wedge_i \wedge_{j, j \neq i} \wedge_k \neg(v_{ik} \wedge v_{jk}) = \varphi_2$

  iii) Uma cidade não pode ocupar mais de uma posição no percurso:

  $\forall k, \forall k', \forall i | 1 \leq k \leq n, 1 \leq k' \leq n, 1 \leq i \leq n, k' \neq k : \neg(v_{ik} \wedge v_{ik'})$

  $\Leftrightarrow \wedge_k \wedge_{k', k' \neq k} \wedge_i \neg(v_{ik} \wedge v_{ik'}) = \varphi_3$

- Restrição de Otimalidade:

  iv) Se duas cidades ocupam posições consecutivas no percurso, então o custo do caminho entre elas deve ser considerado:

  $forall\ i,\ j,\ k | 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n, i \neq i' : dist_{ij}(v_{ik} \wedge v_{j(k+1)})$

  $\Leftrightarrow \vee_i \vee_{j,j \neq i} \vee_k (dist_{ij}(v_{ik} \wedge v_{j(k+1)})) = \varphi_4$

Estas restrições possuem uma ordem de precedência, que é expressa através de constantes multiplicativas. Neste problema as restrições ii) e iii) estão no topo da hierarquia, seguidas pela restrição i) e no fim, encontramos as restricões de otimalidade. A mais alta precedência será expressa através da constante β, em seguida teremos a constante α.

$$\begin{aligned} d &= \max\{dist_{ij}\}; \\ \alpha &= (n \times d) + h; \\ \beta &= ((n^3 - 2n^2 + n) \times \alpha) + h. \end{aligned} \tag{3.7}$$

As equações em 3.7 foram obtidas da seguinte forma:

- α representa o menor nível de precedência e é o limite superior para o custo de um percurso. Em outras palavras, α é o produto entre o número total de cidades em um percurso e a maior distância entre duas cidades consecutivas somado a um pequeno valor $h > 0$;

- β representa o nível de precedência imediatamente superior a α e é o produto entre α e o total de fórmulas com precedência α.

Satisfatibilidade Mapeada para Minimização de Energia

Utilizaremos o operador H^*, descrito no Capítulo 2, para mapear as fórmulas da lógica proposicional, geradas na seção anterior, no conjunto $D = \{0, 1\}$.

Subdividiremos nossa função de energia em duas partes:

- A primeira se refere às restrições de viabilidade e será representada por E_v;
- A segunda se refere às restrições de otimalidade e será representada por E_o.

Primeiro geraremos $E_v = \sum_{i=1}^{3} H^*(\neg\varphi_i)$:

- Dado $\varphi_1 = \wedge_i \, (\vee_k \, (v_{ik})) \qquad \Rightarrow \qquad \neg\varphi_1 = \vee_i(\wedge_k(\neg v_{ik}\,))$

$$H^*(\neg\varphi_1) = \sum_{i=1}^{n} H(\wedge_k(\neg v_{ik}))$$

$$= \sum_{i=1}^{n} \prod_{k=1}^{n} H(\neg v_{ik})$$

$$= \sum_{i=1}^{n} \prod_{k=1}^{n} (1 - v_{ik})$$

$$= \sum_{i=1}^{n} (1 - v_{i1})(1 - v_{i2})(1 - v_{i3})...(1 - v_{in})$$

$$\Rightarrow \sum_{i=1}^{n} \left[\begin{array}{l} (-1)^{n-n} + (-1)^{n-(n-1)} \sum_{k=1}^{n} v_{ik} + \\ (-1)^{n-(n-2)} \sum_{k=1}^{n} \sum_{k'=1\,,\,k'\neq k}^{n} v_{ik} v_{ik'} + \\ (-1)^{n-(n-3)} \sum_{k=1}^{n} \sum_{k'=1}^{n} \sum_{k''=1\,,\,k''\neq k'\neq k}^{n} v_{ik} v_{ik'} v_{ik''} + ... + \\ (-1)^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} \cdots \sum_{k^{(n-1)}=1\,,\,k^{(n-1)}\neq ...\neq k'\neq k}^{n} v_{ik} v_{ik'} ... v_{ik^n} \end{array} \right|$$

$$\Rightarrow \left\{ \begin{array}{l} \sum_{i=1}^{n} (-1)^{n-n} + (-1)^{n-(n-1)} \sum_{i=1}^{n} \sum_{k=1}^{n} v_{ik} \\ +(-1)^{n-(n-2)} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1\,,\,k'\neq k}^{n} v_{ik} v_{ik'} \\ +(-1)^{n-(n-3)} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} \sum_{k''=1\,,\,k''\neq k'\neq k}^{n} v_{ik} v_{ik'} v_{ik''} \\ +... + (-1)^{n} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} \\ \quad \cdots \sum_{k^{(n-1)}=1\,,\,k^{(n-1)}\neq ...\neq k'\neq k}^{n} v_{ik} v_{ik'} ... v_{ik^n} \end{array} \right| \qquad (3.8)$$

- Dado $\varphi_2 = \wedge_i \wedge_{j,j\neq i} \wedge_k \neg(v_{ik} \wedge v_{jk})$

$$\Rightarrow \quad \neg\varphi_2 = \vee_i \vee_{j,j\neq i} \vee_k (v_{ik} \wedge v_{jk})$$

$$H^*(\neg\varphi_2) = \sum_{i=1}^{n} H(\vee_{j,j\neq i} \vee_k (v_{ik} \wedge v_{jk}))$$

$$= \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} H(\vee_k (v_{ik} \wedge v_{jk}))$$

$$\Rightarrow \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} \sum_{k=1}^{n} v_{ik} v_{jk} \tag{3.9}$$

- Dado $\varphi_3 = \wedge_k \wedge_{k',k'\neq k} \wedge_i \neg(v_{ik} \wedge v_{ik'})$

$$\Rightarrow \quad \neg\varphi_3 = \vee_k \vee_{k',k'\neq k} \vee_i (v_{ik} \wedge v_{ik'})$$

$$H^*(\neg\varphi_3) = \sum_{k=1}^{n} H(\vee_{k',k'\neq k} \vee_i (v_{ik} \wedge v_{ik'}))$$

$$= \sum_{k=1}^{n} \sum_{k'=1,k'\neq k}^{n} H(\vee_i (v_{ik} \wedge v_{ik'}))$$

$$\Rightarrow \sum_{k=1}^{n} \sum_{k'=1,k'\neq k}^{n} \sum_{i=1}^{n} v_{ik} v_{ik'} \tag{3.10}$$

Agora geraremos $E_o = \sum_{i=4}^{4} H^*(\varphi_i)$:

- Dado $\varphi_4 = \vee_i \vee_{j,j\neq i} \vee_k (dist_{ij}(v_{ik} \wedge v_{j(k+1)}))$

$$H^*(\varphi_4) = \sum_{i=1}^{n} H(\vee_{j,j\neq i} \vee_k (dist_{ij}(v_{ik} \wedge v_{j(k+1)})$$

$$= \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} H(\vee_k(dist_{ij}(v_{ik} \wedge v_{j(k+1)})$$

$$= \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{k=1}^{n} dist_{ij} H(v_{ik} \wedge v_{j(k+1)})$$

$$\Rightarrow \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{k=1}^{n} dist_{ij}(v_{ik} v_{j(k+1)})) \tag{3.11}$$

Como a restrição iii) determina que a ativação de um neurônio v_{ik} inibe a ativação de $v_{ik'}, \forall k \neq k'$ o termo de energia gerado por $H^*(\neg\varphi_1)$ se resume a $1 - v_{ik}$ (Equação 3.12).

$$\sum_{i=1}^{n} \sum_{k=1}^{n} (1 - v_{ik}) \tag{3.12}$$

Finalmente, obtemos nossa função de energia através da associação entre as Equações 3.12, 3.9, 3.10 e 3.11 com seus respectivos pesos, expressos através das constantes multiplicativas definidas na Equação 3.7.

$$E = E_v + E_o$$

$$= \beta \left[\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ik} v_{jk} + \sum_{i=1}^{n} \sum_{k'=1}^{n} v_{ik} v_{ik'} \right]$$

$$+\alpha \left[\sum_{i=1}^{n} \sum_{k=1}^{n} (1 - v_{ik}) \right]$$

$$+ \left[\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} dist_{ij}(v_{ik} v_{i(k+1)}) \right]$$

3.4.2 O Problema da Coloração de Vértices

Seja $G = (V, E)$ um grafo não direcionado. O problema da coloração de vértices consiste em encontrar o número cromático de G, denotado por $\chi(G)$.

Nesta seção $n = |V|$ e nossa rede neuronal será composta por três estruturas de neurônios binários, uma matriz bidimensional vc com $n \times n$ neurônios, um vetor cor que possui n neurônios e uma matriz bidimensional viz com $n \times n$ neurônios.

Mapeamento de Restrições para Satisfatibilidade

Na primeira estrutura que compõe nossa rede (a matriz vc), as linhas representam os vértices $i \in V$ e as colunas representam as possíveis cores que os vértices podem possuir. Sendo assim, os neurônios nesta estrutura são definidos por vc_{ik}, tal que $\forall i \in V$ temos que k representa a cor associada ao vértice i.

No vetor cor os neurñios são definidos por cor_k, este vetor contém todas as cores que podem ser atribuídas a um vértice. Além disso, utilizaremos neste problema uma estrutura viz_{ij}que auxiliará na modelagem das restrições, estabelecendo a vizinhança $N(i)$ de cada $i \in V$. Os componentes desta estrutura participarão de pesos neuronais.

A seguir, apresentamos os conjuntos de restrições que especificam o problema da coloração de vértices.

- Restrições de Viabilidade:

  i)Todos os n vértices devem possuir uma cor:

$\forall i, \exists_k | 1 \leq i \leq n, 1 \leq k \leq n : vc_{ik} \Rightarrow \wedge_i(\vee_k(vc_{ik}))$

ii)Dois vértices adjacentes não podem possuir a mesma cor:

$\forall i, \forall i', \forall_k | 1 \leq i \leq n, 1 \leq i' \leq n, 1 \leq k \leq n, i \neq i' : \neg(viz_{ii'}) \vee \neg(vc_{ik} \wedge vc_{i'k})$

$\Rightarrow \wedge_i \wedge_{i', i' \neq i} \wedge_k (\neg(viz_{ii'}) \vee \neg(vc_{ik} \wedge vc_{i'k}))$

iii)Um vértice não pode possuir mais de uma cor:

$\forall i, \forall_k, \forall_{k'} | 1 \leq i \leq n, 1 \leq k' \leq n, 1 \leq k \leq n, 1 \leq k' \leq n, k \neq k' :$

$\neg(v_{ik} \wedge v_{ik'})$

$\Rightarrow \wedge_i \wedge_k \wedge_{k', k' \neq k} \neg(vc_{ik} \wedge vc_{ik'})$

iv)Todas as cores que são atribuídas aos vértices devem estar na matriz *cor*:

$\forall i, \forall k | 1 \leq i \leq n, 1 \leq k \leq n : \neg(vc_{ik}) \vee (cor_k)$

$\Rightarrow \wedge_i \wedge_k (\neg(vc_{ik}) \vee (cor_k))$

- Restrição de Otimalidade:

  v)O custo associado a atribuição de cores aos vértices deve ser considerado:

  $forall\ k | 1 \leq k \leq n : cor_k \Rightarrow \vee_k cor_k$

Assim como no TSP, utilizaremos constantes multiplicativas α e β para indicar a ordem de precedência das restrições (Equações em 3.13). Neste caso, β representa a mais alta precedência, seguida por α. Além disso, α é igual a um limite superior para $\chi(G)$, ou seja, α é igual ao número total n de cores $+h$ e β é igual ao produto entre α e o número de fórmulas que representam restrições de nível $\alpha + h$.

$$\begin{aligned} c &= \max\{cor_k\} = 1; \\ \alpha &= ((n \times c) + h; \\ \beta &= ((n^3 - 2n^2 + n) \times \alpha) + h. \end{aligned} \tag{3.13}$$

Satisfatibilidade Mapeada para Minimização de Energia

Primeiro geraremos a função de energia relativa as restrições de viabilidade $E_v = \sum_{i=1}^{4} H^*(\neg\varphi_i)$:

- Dado $\varphi_1 = \wedge_i (\vee_k (v c_{ik}))$ $\Rightarrow$ $\neg\varphi_1 = \vee_i(\wedge_k(\neg v c_{ik}))$

$$H^*(\neg\varphi_1) = \sum_{i=1}^{n} H(\wedge_k(\neg v c_{ik}))$$

$$\Rightarrow \left\{ \begin{array}{l} \sum_{i=1}^{n}(-1)^{n-n} + (-1)^{n-(n-1)} \sum_{i=1}^{n} \sum_{k=1}^{n} v c_{ik} \\ +(-1)^{n-(n-2)} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1\ ,\ k'\neq k}^{n} v c_{ik} v c_{ik'} \\ +(-1)^{n-(n-3)} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} \sum_{k''=1\ ,\ k''\neq k'\neq k}^{n} v c_{ij} v c_{ik'} v c_{ik''} \\ +\ldots + (-1)^{n} \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} \\ \quad \ldots \sum_{k^{(n-1)}=1\ ,\ k^{(n-1)}\neq\ldots\neq k'\neq k}^{n} v c_{ik} v c_{ik'} \ldots v c_{ik^n} \end{array} \right| \qquad (3.14)$$

Devido a restrição iii), a Equação 3.14 se transforma na Equação 3.15.

$$\sum_{i=1}^{n} \sum_{k=1}^{n} (1 - v c_{ik}) \qquad (3.15)$$

- Dado $\varphi_2 = \wedge_i \wedge_{i',i'\neq i} \wedge_k (\neg(v i z_{ii'}) \vee \neg(v c_{ik} \wedge v c_{i'k}))$

$$\Rightarrow \quad \neg\varphi_2 = \vee_i \vee_{i',i'\neq i} \vee_k (v i z_{ii'} \wedge v c_{ik} \wedge v c_{i'k})$$

$$H^*(\neg\varphi_2) = \sum_{i=1}^{n} H(\vee_{i',i'\neq i} \vee_k (v i z_{ii'} \wedge v c_{ik} \wedge v c_{i'k})$$

$$\Rightarrow \sum_{i=1}^{n} \sum_{i'=1,i'\neq i}^{n} \sum_{k=1}^{n} v i z_{ii'} v c_{ik} v_{i'k} \qquad (3.16)$$

- Dado $\varphi_3 = \wedge_k \wedge_{k',k'\neq k} \wedge_i \neg(vc_{ik} \wedge vc_{ik'})$

$$= \wedge_k \wedge_{k',k'\neq k} \wedge_i ((\neg vc_{ik}) \vee (\neg vc_{ik'}))$$

$$\Rightarrow \quad \neg\varphi_3 = \vee_k \vee_{k',k'\neq k} \vee_i (vc_{ik} \wedge vc_{ik'})$$

$$H^*(\neg\varphi_3) = \sum_{k=1}^{n} H(\vee_{k',k'\neq k} \wedge_i (vc_{ik} \wedge vc_{ik'}))$$

$$= \sum_{k=1}^{n} \sum_{k'=1,k'\neq k}^{n} H(\wedge_i (vc_{ik} \wedge vc_{ik'}))$$

$$\Rightarrow \sum_{k=1}^{n} \sum_{k'=1,k'\neq k}^{n} \sum_{i=1}^{n} vc_{ik} vc_{ik'} \tag{3.17}$$

- Dado $\varphi_4 = \wedge_i \wedge_k (\neg(vc_{ik}) \vee (cor_k) \quad \Rightarrow \quad \neg\varphi_4 = \vee_i \vee_k (vc_{ik} \wedge (\neg(cor_k)))$

$$H^*(\neg\varphi_4) = \sum_{i=1}^{n} H(\vee_k (vc_{ik} \wedge (\neg(cor_k))))$$

$$\Rightarrow \sum_{i=1}^{n} \sum_{k=1}^{n} vc_{ik}(1 - cor_k) \tag{3.18}$$

Agora geraremos $E_o = \sum_{i=5}^{5} H^*(\varphi_i)$:

- Dado $\varphi_5 = \vee_k cor_k$

$$H^*(\varphi_5) = \sum_{k=1}^{n} H(cor_k)$$

$$\Rightarrow \sum_{k=1}^{n} cor_k \tag{3.19}$$

A seguir, apresentamos nossa função de energia.

$$E = E_v + E_o$$

$$= \beta \left[\begin{array}{l} \sum_{i=1}^{n} \sum_{i'=1}^{n} \sum_{k=1}^{n} vc_{ik} vc_{i'k} viz_{ii'} \\ + \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} vc_{ik} vc_{ik'} \\ + \sum_{i=1}^{n} \sum_{k=1}^{n} v_{ik}(1 - cor_k) \end{array} \right]$$

$$-\alpha[\sum_{i=1}^{n} \sum_{k=1}^{n} vc_{ik}]$$

$$+\sum_{k=1}^{n} cor_k$$

3.4.3 O Problema do Caixeiro Viajante Colorido

Considere um conjunto de cidades distribuídas em regiões, tal que cada região está associada a uma cor e, seja $M = (V, E_1, E_2)$ um multigrafo não direcionado, onde cada vértice em V representa uma cidade, as arestas em E_1 representam pares de cidades situadas em regiões adjacentes e as arestas em E_2 representam a existência de caminho direto (sem passar por outras cidades) entre pares de cidades. Neste caso, cada aresta $(i, j) \in E_2$ está associada a um custo $dist_{ij}$, como no problema do caixeiro viajante.

O problema do caixeiro viajante colorido consiste em encontrar o percurso de custo mínimo e uma coloração no multigrafo M de forma que cidades em regiões adjacentes possuam cores distintas e cidades na mesma região possuam a mesma cor.

Nossa rede neuronal será composta por duas matrizes bidimensionais (v e vc), cada uma com $n \times n, n = |V| + 1$ neurônios binários. Na primeira, cada posição ik indica que a cidade i está na posição k de um percurso e na segunda, que para cada posição jc a cidade j recebeu a cor c em uma coloração.

Temos também um vetor *cor*, que possui n neurônios binários, que representam as cores que podem ser atribuídas a vértices em V. Além disso, temos duas matrizes auxiliares, cada uma com $n \times n$ posições. A primeira matriz auxiliar se chama *dist* e é utilizada para armazenar as distâncias associadas as arestas em E_2 e a segunda se chama *viz* e armazena a vizinhança associada a E_1. Os componentes destas matrizes auxiliares participam de pesos neuronais.

Mapeamento de Restrições para Satisfatibilidade

O conjunto de restrições que definem este problema é formado pela união dos conjuntos de restrições do problema do caixeiro viajante e do problema da coloração de vértices, considerando uma restrição de otimalidade adicional (x). Neste caso, temos um total de sete restrições de integridade e três restrições de otimalidade.

x) O custo associado ao cruzamento de fronteiras deve ser considerado:

$$forall\ i,\ k,\ j,\ c,\ c' | 1 \leq i \leq n, 1 \leq k \leq n-1, 1 \leq j \leq n, 1 \leq c \leq n, 1 \leq c' \leq n, i \neq j, c \neg c' : (v_{ik} \wedge v_{j(k+1)} \wedge vc_{ic} \wedge vc_{jc'})$$

$$\Rightarrow \vee_i \vee_k \vee_j \vee_c \vee_{c'} (v_{ik} \wedge v_{j(k+1)} \wedge vc_{ic} \wedge vc_{jc'})$$

As constantes multiplicativas que participarão dos pesos neuronais são apresentadas em 3.20 e a função de energia resulta do mapeamento das dez restrições anteriores combinadas com suas constantes multiplicativas.

$$\begin{aligned} d &= \max\{dist_{ij}\}; \\ \alpha &= ((n^3 - 2n^2 + n) \times d) + h; \\ \beta &= ((n^5 - n^4 - n^3 + n^2 + 1) \times \alpha) + h; \\ \gamma &= ((n+1) \times \beta) + h; \\ \sigma &= ((2n^3 - n^2 + n + 1) \times \gamma) + h. \end{aligned} \tag{3.20}$$

A ordem de precedência para as restrições do TSP são:

- restrição (i) $\Rightarrow$ ordem de precedência γ;
- restrição (ii) $\Rightarrow$ ordem de precedência σ;
- restrição (iii) $\Rightarrow$ ordem de precedência σ.

A ordem de precedência para as restrições do VCP são:

- restrição (i) $\Rightarrow$ ordem de precedência γ;
- restrição (ii) $\Rightarrow$ ordem de precedência γ;
- restrição (iii) $\Rightarrow$ ordem de precedência σ;
- restrição (iv) $\Rightarrow$ ordem de precedência γ;
- restrição (v) $\Rightarrow$ ordem de precedência β.

Finalmente, a ordem de precedência associada à restrição (x) é determinada por α.

3.5 Mapeamento de Aritmética Binária para Satisfatibilidade

Nesta seção apresentaremos a especificação de algumas operações aritméticas na base 2 (soma/subtração, produto e módulo) através de restrições e seu mapeamento para satisfatibilidade.

Alguns destes problemas serão utilizados na geração do modelo matemático que iremos propor para o problema da predição das conformações moleculares estáveis (Capítulo 4).

3.5.1 O Somador

O problema somador consiste em, dados dois números inteiros na base dois, determinar o resultado da soma entre eles.

Seja $G = (V, E)$ o grafo não direcionado que representa a rede associada a este problema. Esta rede é composta por quatro estruturas neuronais. São elas:

*number*1(b) e *number*2(b) Estas estruturas contêm os números binários cuja soma deve ser calculada. A última posição de cada uma delas ($b = bits$) representa o bit mais significativo que, neste caso, será o bit de sinal;

resultado(b) Esta estrutura armazena o resultado da soma entre *number*1(b) e *number*2(b);

vaium($b + 1$) Esta estrutura armazena o vai um da soma bit a bit entre *number*1(b) e *number*2(b);

Neste caso, $n = |V|$ e b representa o número de bits utilizados na representação dos números armazenados em $number1(b)$ e em $number2(b)$.

Mapeamento das Restrições para Satisfatibilidade

O conjunto de restrições que especificam o problema somador foi obtido a partir da descrição do procedimento de cálculo binário, relativo a soma. Ou seja, este conjunto de restrições, de viabilidade, especifica as regras para que um bit seja 0 ou 1, simulando uma máquina de cálculo binário. A Figura 3.6 apresenta o esquema de cálculo utilizado para a descrição da soma binária e a Tabela 3.1 apresenta a tabela verdade utilizada na geração das regras, ou restrições, que especificam a soma. Cada linha da tabela verdade representa uma fórmula lógica e a partir da primeira linha, por exemplo, podemos obter uma das restrições que determinam quando um bit da soma ou do vai um deve ser zero (0). Neste caso, a fórmula relativa a não ativação de um bit na posição b da estrutura $resultado$ será $(\neg vaium_b \vee \neg number1_b \vee \neg number2_b \vee \neg resultado_b)$.

$vaium \rightarrow$	0	0	1	1	0
$number1 \rightarrow$	0	0	1	1	
$number2 \rightarrow$	0	0	1	1	
$resultado \rightarrow$	0	1	1	0	
$b \rightarrow$	5	4	3	2	1

Figura 3.6: Exemplo do processo de cálculo de uma soma binária entre dois números com 4 bits($bits = 4$).

Neste problema todas as restrições possuem a mesma ordem de precedência ($\alpha = 1$), pois todas são fundamentais para que o cálculo seja feito corretamente. A seguir, apresentamos o conjunto de restrições que especificam o somador.

Tabela 3.1: Tabela verdade a partir da qual o conjunto de restrições que especificam a soma é gerado.

$number1_b$	$number2_b$	$vaium$	$resultado_b$	$vaium_{b+1}$
0	0	0	0	0
0	0	0	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Restrições de Viabilidade:

  i - iv) Restrições relativas à ativação dos bits do resultado da soma, armazenado em $resultado(b)$:

  $\forall b | 1 \leq b \leq bits : (vaium_b \vee number1_b \vee \neg number2_b \vee resultado_b)$

  $\Rightarrow \wedge_b (vaium_b \vee number1_b \vee \neg number2_b \vee resultado_b)$

  $\forall b | 1 \leq b \leq bits : (vaium_b \vee \neg number1_b \vee number2_b \vee resultado_b)$

  $\Rightarrow \wedge_b (vaium_b \vee \neg number1_b \vee number2_b \vee resultado_b)$

  $\forall b | 1 \leq b \leq bits : (\neg vaium_b \vee number1_b \vee number2_b \vee resultado_b)$

  $\Rightarrow \wedge_b (\neg vaium_b \vee number1_b \vee number2_b \vee resultado_b)$

  $\forall b | 1 \leq b \leq bits : (\neg vaium_b \vee \neg number1_b \vee \neg number2_b \vee resultado_b)$

  $\Rightarrow \wedge_b (\neg vaium_b \vee \neg number1_b \vee \neg number2_b \vee resultado_b)$

  v - viii) Restrições relativas à não ativação dos bits do resultado da soma, armazenado em $resultado(b)$:

  $\forall b | 1 \leq b \leq bits : (vaium_b \vee number1_b \vee number2_b \vee \neg resultado_b)$

  $\Rightarrow \wedge_b (vaium_b \vee number1_b \vee number2_b \vee \neg resultado_b)$

$\forall b | 1 \leq b \leq bits : (\neg vaium_b \vee \neg number1_b \vee number2_b \vee \neg resultado_b)$

$\Rightarrow \wedge_b(\neg vaium_b \vee \neg number1_b \vee number2_b \vee \neg resultado_b)$

$\forall b | 1 \leq b \leq bits : (\neg vaium_b \vee number1_b \vee \neg number2_b \vee \neg resultado_b)$

$\Rightarrow \wedge_b(\neg vaium_b \vee number1_b \vee \neg number2_b \vee \neg resultado_b)$

$\forall b | 1 \leq b \leq bits : (vaium_b \vee \neg number1_b \vee \neg number2_b \vee \neg resultado_b)$

$\Rightarrow \wedge_b(vaium_b \vee \neg number1_b \vee \neg number2_b \vee \neg resultado_b)$

ix - xii) Restrições relativas à ativação dos bits do vai um da soma, armazenado em $vaium(b+1)$:

$\forall b | 1 \leq b \leq bits : (\neg vaium[b] \vee \neg number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

$\Rightarrow \wedge_b(\neg vaium[b] \vee \neg number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

$\forall b | 1 \leq b \leq bits : (\neg vaium[b] \vee \neg number1[b] \vee number2[b] \vee vaium_{b+1})$

$\Rightarrow \wedge_b(\neg vaium[b] \vee \neg number1[b] \vee number2[b] \vee vaium_{b+1})$

$\forall b | 1 \leq b \leq bits : (vaium[b] \vee \neg number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

$\Rightarrow \wedge_b(vaium[b] \vee \neg number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

$\forall b | 1 \leq b \leq bits : (\neg vaium[b] \vee number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

$\Rightarrow \wedge_b(\neg vaium[b] \vee number1[b] \vee \neg number2[b] \vee vaium_{b+1})$

xiii - xvii) Restrições relativas à não ativação dos bits do vai um da soma, armazenado em $vaium(b+1)$:

$\forall b | 1 \leq b \leq bits : (vaium_b \vee number1_b \vee number2_b \vee \neg vaium_{b+1})$

$\Rightarrow \wedge_b(vaium_b \vee number1_b \vee number2_b \vee \neg vaium_{b+1})$

$\forall b | 1 \leq b \leq bits : (vaium_b \vee number1_b \vee \neg number2_b \vee \neg vaium_{b+1})$

$\Rightarrow \wedge_b(vaium_b \vee number1_b \vee \neg number2_b \vee \neg vaium_{b+1})$

$$\forall b | 1 \leq b \leq bits : (vaium_b \vee \neg number1_b \vee number2_b \vee \neg vaium_{b+1})$$

$$\Rightarrow \wedge_b (vaium_b \vee \neg number1_b \vee number2_b \vee \neg vaium_{b+1})$$

$$\forall b | 1 \leq b \leq bits : (\neg vaium_b \vee number1_b \vee number2_b \vee \neg vaium_{b+1})$$

$$\Rightarrow \wedge_b (\neg vaium_b \vee number1_b \vee number2_b \vee \neg vaium_{b+1})$$

3.5.2 O Produto Binário

O problema produto binário consiste em determinar o resultado do produto entre dois números inteiros na base dois.

Seja $G = (V, E)$ o grafo não direcionado que representa a rede associada a este problema. Esta rede é composta por cinco estruturas neuronais. São elas:

$number1(b)$ **e** $number2(b)$ Estas estruturas contêm os números binários cujo produto deve ser calculado. A última posição de cada uma delas ($b = bits$) representa o bit mais significativo que, neste caso, será o bit de sinal;

$parcela(l, b)$ Esta estrutura armazena as parcelas do produto binário entre $number1(b)$ e $number2(b)$;

$resultado(l, b)$ Esta estrutura armazena a soma das parcelas do produto binário entre $number1(b)$ e $number2(b)$ (para $1 \leq l \leq bits - 1$) e o resultado do produto em $l = bits$;

$vaium(l, b + 1)$ Esta estrutura armazena o vai um da soma bit a bit entre as parcelas armazenadas em $resultado(l, b), 1 \leq l \leq bits$;

Neste caso, $n = |V|$, l representa o número de parcelas intermediárias produzidas durante o cálculo do produto binário e b representa o número de bits utilizados na representação dos números armazenados em $number1(b)$ e em $number2(b)$.

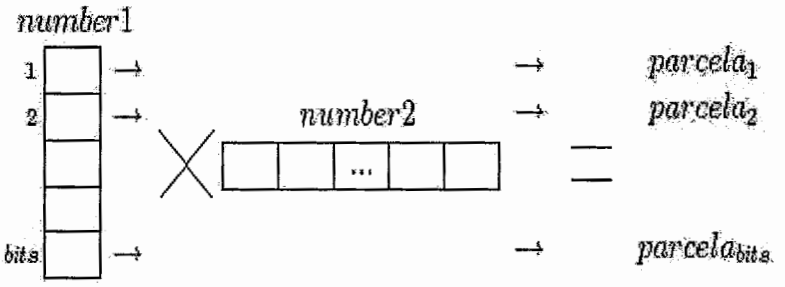
Mapeamento das Restrições para Satisfatibilidade

O conjunto de restrições que especificam o problema produto binário foi obtido a partir da descrição do procedimento de cálculo binário, relativo ao produto. Ou seja, este conjunto de restrições, de viabilidade, especifica as regras para que um bit seja 0 ou 1, simulando uma máquina de cálculo binário. A Figura 3.7 a) apresenta o esquema de cálculo utilizado para a descrição da geração das parcelas do produto binário e a Figura 3.7 b) apresenta o esquema de cálculo utilizado para a descrição da geração da soma das parcelas do produto binário. A partir da Figura 3.7 a) obtemos a restrição que diz que se $number1_b = 0$ então $parcela_{bb'} = 0$, $b' = 1...bits$, senão $parcela_{bb'} = number2_{b'}$, $b' = 1...bits$. Em outras palavras, cada uma das parcelas do produto entre $number1$ e $number2$ é nula ($number1_b = 0$) ou é igual a $number2$ ($number1_b = 1$). Na Figura 3.7 b) vemos que o resultado do produto é obtido somando todas as parcelas apresentadas na Figura 3.7 b) e o resultado da operação de multiplicação entre $number1$ e $number2$ é armazenada em $resultado_{bits-1}$.

Neste problema todas as restrições possuem a mesma ordem de precedência ($\alpha = 1$), pois todas são fundamentais para que o cálculo seja feito corretamente. A seguir, apresentamos o conjunto de restrições que especificam o somador.

Todas as restrições possuem ordem de precedência $\alpha = 1$. Além disso, não mostraremos as restrições que determinam quando um bit, relativo ao vai um ou ao

a)



b)

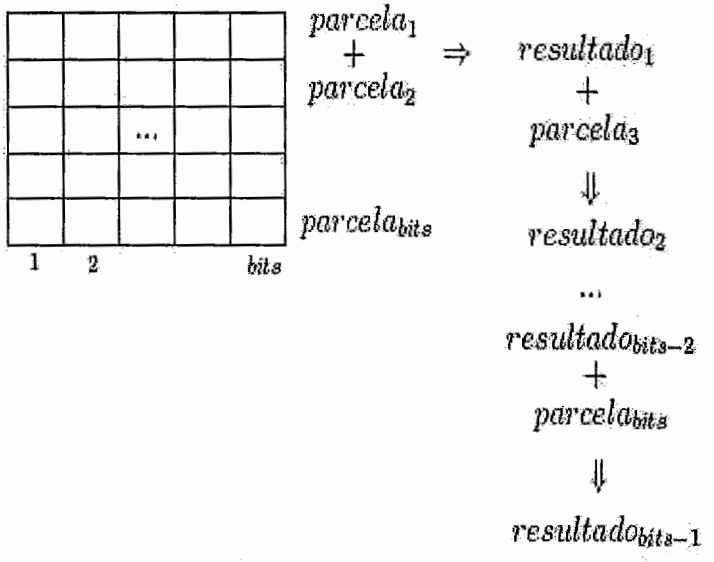


Figura 3.7: Exemplo do processo de cálculo do produto binário (*number*1 × *number*2). Apresentamos em a) o processo de geração e em b) a soma das parcelas do produto.

resultado da soma entre as parcelas do produto, deve ou não ser ativado, pois estas restrições são baseadas no problema somador, apresentado na Subseção 3.5.1. A seguir, apresentamos o conjunto de restrições que especificam o produto binário.

- Restrições de Viabilidade:

  i) Restrições relativas à ativação dos bits de cada parcela produzida durante o cálculo do produto e armazenadas em $resultado(l, b)$:

  $\forall l \forall b \forall v | 1 \leq l \leq bits, 1 \leq b \leq bits + 1 - l, v = l - 1 :$

  $(parcela_{l(b+v)} \vee \neg number1_l \vee \neg number2_b)$

  $\Rightarrow \wedge_l \wedge_b \wedge_v (parcela_{l(b+v)} \vee \neg number1_l \vee \neg number2_b)$

  ii - iv) Restrições relativas à não ativação dos bits de cada parcela produzida durante o cálculo do produto e armazenadas em $resultado(l, b)$:

  $\forall l \forall b \forall v | 1 \leq l \leq bits, 1 \leq b \leq bits + 1 - l, v = l - 1 :$

  $(\neg parcela_{l(b+v)} \vee number1_l \vee number2_b)$

  $\Rightarrow \wedge_l \wedge_b \wedge_v (\neg parcela_{l(b+v)} \vee number1_l \vee number2_b)$

  $\forall l \forall b \forall v | 1 \leq l \leq bits, 1 \leq b \leq bits + 1 - l, v = l - 1 :$

  $(\neg parcela_{l(b+v)} \vee number1_l \vee \neg number2_b)$

  $\Rightarrow \wedge_l \wedge_b \wedge_v (\neg parcela_{l(b+v)} \vee number1_l \vee \neg number2_b)$

  $\forall l \forall b \forall v | 1 \leq l \leq bits, 1 \leq b \leq bits + 1 - l, v = l - 1 :$

  $(\neg parcela_{l(b+v)} \vee \neg number1_l \vee number2_b)$

  $\Rightarrow \wedge_l \wedge_b \wedge_v (\neg parcela_{l(b+v)} \vee \neg number1_l \vee number2_b)$

3.5.3 O Módulo da Diferença

O problema módulo da diferença consiste em determinar o módulo da soma entre dois números na base dois, um deles pertencente a $\mathbb{Z}_+$ e o outro a $\mathbb{Z}_-$. Observe que este problema, assim como o produto apresentado na Subseção 3.5.2, utiliza um conjunto de somadores, especificados pelas restrições que foram definidas na Subseção 3.5.1. Neste caso, apenas as restrições que definem o módulo do resultado da soma serão apresentadas.

Seja $G = (V, E)$ o grafo não direcionado que representa a rede associada ao módulo da diferença. Esta rede é composta por oito estruturas neuronais. São elas:

$number1(b)$ e $number2(b)$ Estas estruturas contêm os números binários cujo módulo deve ser calculado. A última posição de cada uma delas ($b = bits$) representa o bit mais significativo que, neste caso, será o bit de sinal;

$Sresultado(b)$ Esta estrutura armazena o resultado da soma entre $number1(b)$ e $number2(b)$;

$Svaium(b+1)$ Esta estrutura armazena o vai um da soma bit a bit entre $number1(b)$ e $number2(b)$;

$um(b)$ Esta estrutura armazena o número um na base dois. Ela deverá ser utilizada no cálculo do módulo da soma armazenada em $Sresultado(b)$;

$complemento1(b)$ Esta estrutura armazena o resultado do complemento a um do valor armazenado em $Sresultado(b)$, ou seja, ele guarda $\neg Sresultado(b), b = 1...bits$ e deverá ser utilizada no cálculo do módulo de $Sresultado(b)$;

$Mresultado(b)$ Esta estrutura armazena o resultado do módulo da soma entre $number1(b)$ e $number2(b)$. Este valor é calculado a partir do complemento a um do valor em $Sresultado(b)$ somado ao valor em $um(b)$, se $Sresultado(b)$ é negativo e é igual a $Sresultado(b)$ caso contrário;

$Cvaium(b+1)$ Esta estrutura armazena o vai um da soma bit a bit entre $Sresultado(b)$ e $um(b)$;

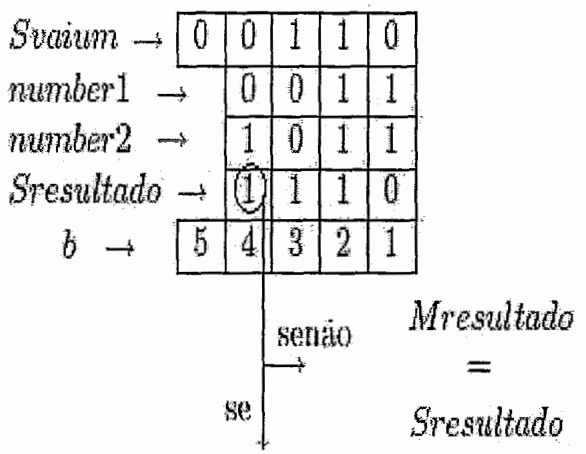
Neste caso, $n = |V|$ e b representa o número de bits utilizados na representação dos números armazenados em $number1(b)$ e em $number2(b)$.

Mapeamento das Restrições para Satisfatibilidade

O conjunto de restrições que especificam o problema módulo da diferença foi obtido a partir da descrição do procedimento de cálculo binário, relativo ao módulo. Em outras palavras, nosso conjunto de restrições especifica o processo de soma binária e seu complemento a um, somado ao número binário um. A Figura 3.8 apresenta o esquema de cálculo utilizado para a descrição do módulo. Se o resultado da soma (*Sresultado*) é negativo (bit de sinal, circulado na Figura 3.8, igual a um 1) então é realizado o complemento a dois da soma, ou seja, o complemento a um somado ao número binário um 1. Por outro lado, se o resultado da soma for positivo (bit de sinal igual a zero 0) o módulo *Mresultado* recebe o valor da soma.

Neste problema as restrições também possuem ordem de precedência $\alpha = 1$. A seguir, apresentamos o conjunto de restrições que especificam o problema módulo da diferença.

a)



b)

Cvaium →	0	0	0	1	0
*complemento*1 →		0	0	0	1
um →		0	0	0	1
Mresultado →		0	0	1	0
b →	5	4	3	2	1

Figura 3.8: Ilustração do processo de cálculo do módulo entre dois números binários armazenados em *number*1 e em *number*2. Em a) apresentamos o processo de soma e em b) o cálculo do complemento a dois. A realização do complemento a dois está condicionada ao bit de sinal do resultado da soma possuir valor 1.

- Restrições de Viabilidade:

  i) Restrições relativas ào cálculo do módulo a partir da soma armazenada em $Sresultado(b)$:

  1. Se o resultado da soma é maior ou igual a zero então a soma é igual ao seu módulo:

     $\forall b | 1 \leq b \leq bits : (Sresultado_{bits} \vee$

     $(\neg Mresultado_b \wedge \neg Sresultado_b) \vee (\neg Mresultado_b \wedge \neg Sresultado_b))$

     $\Rightarrow \wedge_b (Sresultado_{bits} \vee$

     $(\neg Mresultado_b \wedge \neg Sresultado_b) \vee (\neg Mresultado_b \wedge \neg Sresultado_b))$

  2. Se o resultado da soma é menor que zero então:

     (a) Calcula-se o complemento a um. Ou seja, $complemento_b$ recebe $\neg Sresultado_b$ $\forall b = 1...bits$:

     $\forall b | 1 \leq b \leq bits : \neg Sresultado_{bits} \vee$

     $(\neg complemento_b \wedge Sresultado_b) \vee (complemento_b \wedge Sresultado_b)$

     $\Rightarrow \wedge_b \neg Sresultado_{bits} \vee$

     $(\neg complemento_b \wedge Sresultado_b) \vee (complemento_b \wedge Sresultado_b)$

     (b) $Mresultado$ recebe a soma entre $complemento1$ e um.

     O conjunto de restrições que especificam esta operação será omitido, pois baseia-se no conjunto de restrições definido na Subseção 3.5.1.

Capítulo 4

O Problema da Predição das Conformações Moleculares Estáveis

Neste capítulo, buscaremos descrever o problema da predição das conformações moleculares estáveis, através da estratégia SMEM, apresentando-o como um problema de minimização de energia. Na Seção 4.1 falaremos de nossas motivações e de como o modelo para o SMCPP será construído e, na Seção 4.2, descreveremos este modelo. Finalmente, na Seção 4.3, apresentaremos o mapeamento realizado através da estratégia SMEM na geração da função de energia que especifica a rede neuronal associada ao SMCPP.

4.1 Considerações Iniciais

No Capítulo 3 utilizamos o argumento de que em problemas como o TSP e o VCP, a construção das redes neuronais que os define é relativamente simples, porém em problemas mais complexos esta construção pode levar a introdução de erros adicionais a solução. O SMCPP se ajusta perfeitamente a esta situação, pois

a construção da rede neuronal que o representa seria extremamente complicada. Outro ponto importante, apresentado no Capítulo 3 e exemplificado através do MC-TSP, na Seção 3.4.3, é o fato de que a estratégia SMEM permite que problemas complexos sejam construídos através da combinação dos subproblemas que os compõem. Esta característica será particularmente importante na formação do modelo matemático que apresentaremos, pois a possibilidade de adicionar informações a um modelo previamente estabelecido, resultado imediato da característica da estratégia SMEM que acabamos de citar, nos permite descrever o problema da predição das conformações moleculares estáveis combinando diferentes modelos que, por sua vez, podem ser subdivididos em problemas mais simples. Neste trabalho iremos propor a combinação entre um modelo geométrico e um modelo clássico ao qual poderemos adicionar termos baseados em leis da teoria da mecânica quântica. Além disso, os modelos que combinaremos serão construídos a partir da combinação entre os problemas aritméticos apresentados no Capítulo 3 e o modelo geométrico que apresentaremos (MDGP-Modificado [13]) será formulado com base na modelagem descrita na Subseção 2.3.3.

Iniciaremos o processo de combinação entre modelos baseados em restrições da teoria quântica e clássica, objetivando a construção de um modelo mais flexível, que possa ser aplicado a sistemas moleculares de pequeno, médio e grande porte e que apresente um grau de precisão aceitável quando o sistema molecular, cuja geometria deve ser determinada, apresentar regiões onde a estimativa de sua forma espacial exija um grau de precisão mais alto. Além disso, esperamos que a simulação deste problema, através de uma estratégia heurística resistente ao ruído (redes neuronais), compense o fato desta estratégia apresentar alta complexidade de armazenamento.

4.2 O Modelo Proposto

No Capítulo 2, Seção 2.3, descrevemos dois modelos utilizados na predição de estruturas moleculares. O primeiro modelo considera interações físicas e o segundo trata este problema como um problema geométrico. Cada um deles se aplica de acordo com o conjunto de informações inicialmente conhecidas e com o grau de precisão que se espera alcançar.

Nesta seção iremos propor um modelo para o SMCPP que combina o modelo físico descrito na Subseção 2.3.2 e um modelo para o MDGP que iremos propor na Subseção4.2.1.

4.2.1 MDGP-Modificado

Considere o problema da geometria das distâncias moleculares sobre um sistema molecular $SM = (\mathcal{A}, L)$, tal como apresentado na Definição 2.11 (detalhes sobre o MDGP podem ser encontrados em [24] e em [23]). Moré e Wo mostram em [27] que para o PGDM que estamos considerando a busca por um $\varepsilon > 0$ ótimo é um problema NP-difícil. Neste caso, por simplicidade assumiremos $\varepsilon = 0$.

O modelo que estamos propondo para a especificação do MDGP, assim como o modelo que descrevemos na Subseção 2.3.3, considera as distâncias $dist_{ij}$ como sendo distâncias euclidianas e fixa os ângulos planos (ou ângulos entre ligações covalentes consecutivas) em seu valor de equilíbrio. Por outro lado, para simplificar a especificação do problema através de fórmulas lógicas, associaremos distâncias e coordenadas espaciais com ângulos planos e com ângulos de torção através da mesma

lei geométrica (lei dos co-senos para ângulos planos). Portanto, nosso modelo, que foi elaborado durante a realização deste trabalho e apresentado em [13], se baseia nos seguintes termos:

Termo Relativo à Distância Euclidiana Este termo especifica a relação entre as coordenadas espaciais de cada vértice $n_i \in V$ e a estimativa da distância $dist_{ij}$ entre cada par $n_i, n_j \in E$, apresentando as restrições que garantem a consistência do valor estimado para a distância entre pares de átomos $a_i, a_j \in \mathcal{A}$ e da estimativa da posição espacial de cada um destes átomos. Este termo pode ser representado matematicamente através da Equação 4.1.

$$\begin{aligned} &dist_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2, \\ &\forall i = 1..|V| - 1,\ j = i + 1..|V|. \end{aligned} \tag{4.1}$$

A Figura 4.1 mostra a representação gráfica da disposição espacial do grafo G, que representa um sistema molecular SM com 5 átomos e 4 ligações covalentes e destaca a interação covalente entre dois átomos a_i, a_j quaisquer. As linhas sólidas representam as distâncias conhecidas, ou seja, as distâncias associadas a ligações covalentes, enquanto as linhas pontilhadas definem as distâncias estimadas.

Termo Relativo à Lei dos Co-senos Este termo apresenta as restrições que garantem a consistência da estimativa de cada ângulo de torção com base em sua relação com os ângulos planos inicialmente fixados. Estas restrições especificam as interações entre as distâncias $dist_{ij}, dist_{kj}$ e $dist_{ki}$ para cada grupo de três vértices $n_i, n_j, n_k \in V$, selecionados dois a dois, e os respectivos ângulos entre as arestas em E que são definidas por eles. Este termo pode ser representado matematicamente através da Equação 4.1.

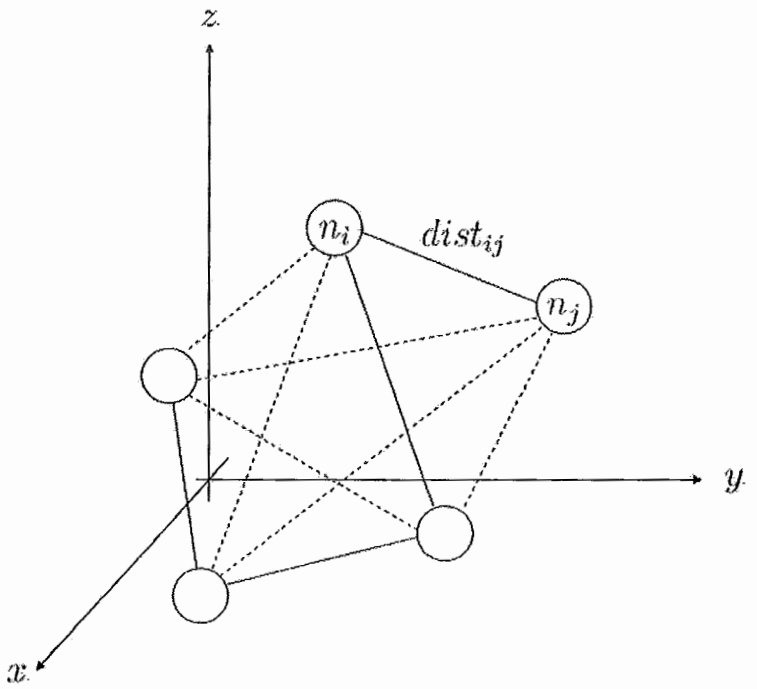


Figura 4.1: Representação gráfica da disposição espacial do grafo G, associado a um sistema molecular SM.

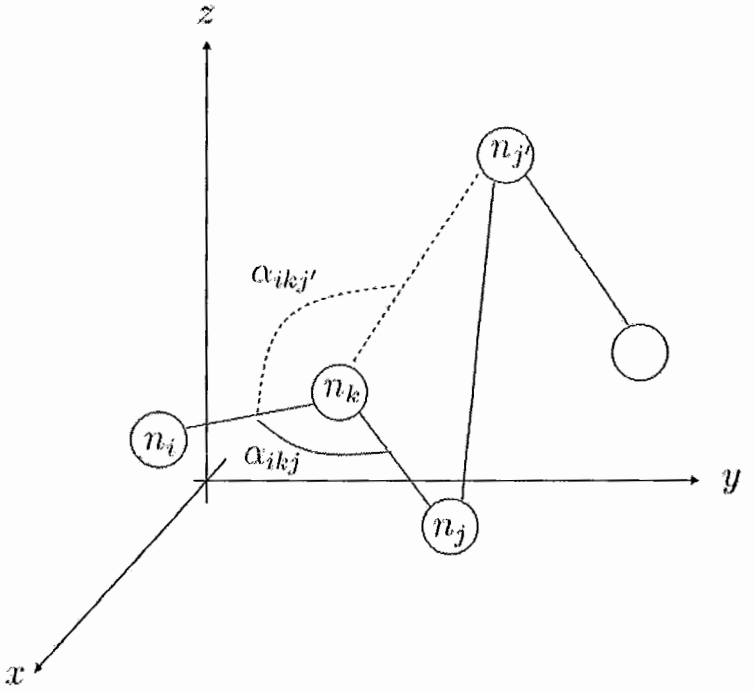


Figura 4.2: Exemplo de ângulos planos e de torção em um sistema molecular SM.

$$dist_{ij}^2 = dist_{kj}^2 + dist_{ki}^2 - (2 \times dist_{kj} \times dist_{ki} \times cos(\alpha_{ikj})),$$
$$\forall i = 1..|V| - 1,\ j = i + 1..|V|,\ k = 1..|V|,\ tal\ que\ k \neq j\ e\ k \neq i. \tag{4.2}$$

A Figura 4.2 apresenta um exemplo de ângulo de torção $\alpha_{ikj'}$ e um exemplo de ângulo plano α_{ikj}, para determinados $i, j, j', k \in \{1, ..., |V|\}$. As variáveis α_{ikj} e $\alpha_{ikj'}$ representam o ângulo entre duas ligações covalentes consecutivas e o ângulo de torção formado pela interseção entre os planos Π_{ikj} (definido por $n_i n_k n_j$) e $\Pi_{ikj'}$ (definido por $n_i n_k n_{j'}$), respectivamente.

4.2.2 Associação entre os Modelos MDGP-Modificado e Físico

Nesta seção combinaremos o modelo físico descrito na Subseção 2.3.2 e o modelo geométrico que propomos na Subseção 4.2.1. Utilizaremos as relações descritas pelo modelo geométrico na criação das restrições de viabilidade, que devem garantir a consistência na predição dos ângulos de torção, a partir de distâncias e ângulos conhecidos e estimados. O modelo físico será utilizado na especificação das restrições de otimalidade, que auxiliarão na predição de ângulos e distâncias através da introdução de informações que refletem algumas características do composto químico, cuja disposição espacial deve ser determinada. Em nosso modelo, o conjunto de restrições de viabilidade será determinado pelos Termos relativos à distância euclidiana e à lei dos co-senos, apresentados na Seção 4.2.1. Estes termos são representados pelas Equações 4.1 e 4.2, respectivamente e estas equações expressam as interações entre ângulos, distâncias e coordenadas espaciais dos átomos em um sistema molecular SM qualquer. O conjunto de restrições de otimalidade será definido através dos

Termos relativos às forças de dispersão e indução dipolar e às atrações e repulsões eletrostáticas apresentados no Capítulo 2 (Seção 2.3.2). Estes termos descrevem as interações polares e ou iônicas que estejam influenciando a dinâmica do composto químico em estudo e são especificados pelas Equações 2.17 e 2.18. A união destes dois conjuntos de restrições forma o modelo que estamos propondo para o problema da predição das conformações moleculares estáveis.

4.2.3 Redução da Complexidade de Armazenamento

Neste ponto, devemos estabelecer uma forma de diminuir a complexidade de tempo e de espaço da rede neuronal que vamos gerar. Observe que a complexidade de espaço do modelo que apresentamos na Subseção 4.2.2 é exponencial, pois estamos considerando a distância entre todos os pares de átomos e os ângulos entre todos os pares de distâncias consecutivas. A partir deste momento, selecionaremos o conjunto de informações realmente necessário, de forma que o desempenho da busca por uma solução do SMCPP não seja comprometido. Neste contexto, Lavor apresenta em [31] uma estratégia que realiza a predição de conformações moleculares estáveis a partir de um conjunto base CB com 3 átomos. Estes átomos são posicionados e os demais são alocados iterativamente, tendo como referência o conjunto base inicialmente fixado. Estamos propondo uma variação desta estratégia, pois com o objetivo de aproveitar ao máximo os dados inicialmente conhecidos, ou seja, as distâncias associadas a ligações covalêntes e os ângulos planos, utilizamos um conjunto base dinâmico. Neste caso, para $G = (V, E)$ representando um sistema molecular SM, tal que $V = \{n_1, ..., n_k, ..., n_n\}$, $E_c = \{(n_1, n_2), (n_2, n_3), ..., (n_{k-1}, n_k)$ $, (n_k, n_{k+1}), ..., (n_{n-1}, n_n)\}$ e sendo conhecidos os ângulos planos α_{ikj}, consideramos a seguinte sucessão de conjuntos base:

1. $CB = \{n_1, n_2, n_3\}$ corresponde a alocação do vértice n_4;

2. $CB = \{n_2, n_3, n_4\}$ corresponde a alocação do vértice n_5;

   ...

3. $CB = \{n_i, n_{i+1}, n_{i+2}\}$ corresponde a alocação do vértice n_{i+3};

   e assim sucessivamente até que o vértice n_n seja alocado.

A utilização destes conjuntos base na alocação dos vértices que representam os átomos de um sistema molecular SM, reduz o número de informações a serem consideradas durante a simulação do SMCPP, pois iremos priorizar apenas as informações necessárias durante a estimativa dos ângulos de torção e o conjunto CB nos ajuda a descobrir quais são estas informações. Definindo genericamente $CB = \{n_i, n_{i+1}, n_{i+2}\}$ de forma que o vértice n_{i+3} (i varia de 1 até $n-3$) seja alocado, a aplicação da lei dos co-senos para ângulos planos deve ser feita com base no plano definido pelos vértices n_i, n_{i+1} e n_{i+3}. Neste caso, as informações necessárias para a alocação de n_{i+3}, com base na predição do ângulo de torção $\alpha_{ii+1i+3}$, são as distâncias $dist_{ii+1}$, $dist_{ii+3}$ e $dist_{i+1i+3}$ e o ângulo $\alpha_{ii+1i+3}$. Na Figura 4.3 a), por exemplo, ao alocar o vértice n_4 o conjunto base CB será $\{n_1, n_2, n_3\}$. Isto significa que devemos aplicar a lei dos co-senos para ângulos planos com base no plano definido pelos vértices n_1, n_2 e n_4 e que o ângulo que deve ser estimado é α_{124}. Portanto, as informações relevantes para são, além do próprio ângulo, as distâncias $dist_{14}, dist_{12}$ e $dist_{24}$. Além disso, as informações conhecidas inicialmente serão utilizadas para aumentar o grau de confiança nas estimativas realizadas.

Ressaltamos que a Figura 4.3, a) e b), apresenta os conjuntos CB utilizados na alocação dos vértices n_4 e n_5, respectivamente. Definindo o ângulo de torção, associado ao vértice n_{i+3} que está sendo alocado, como o ângulo entre o plano definido

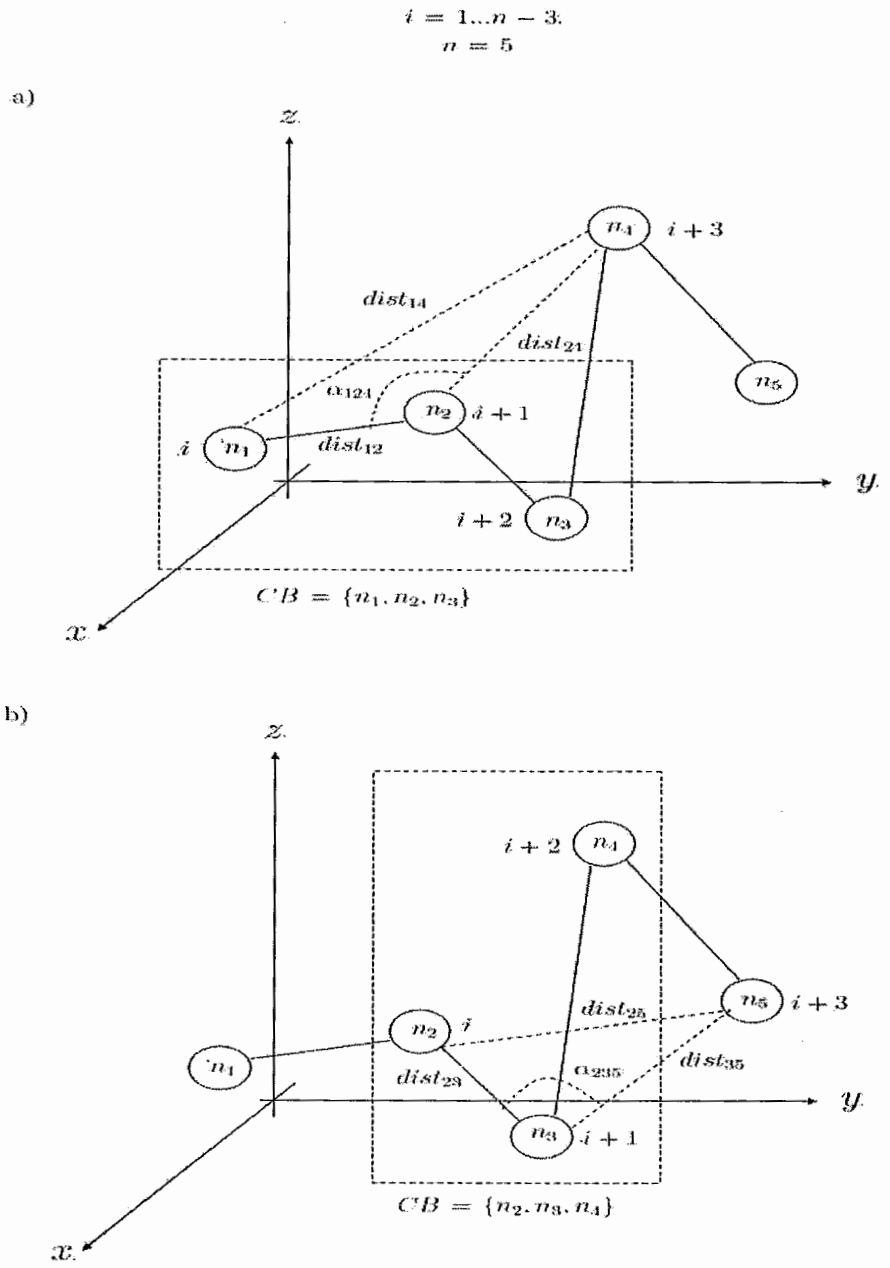


Figura 4.3: Exemplo de utilização do conjunto base CB.

pelo conjunto $CB = \{n_i, n_{i+1}, n_{i+2}\}$ atual e o plano definido por $\{n_{i+1}, n_{i+2}, n_{i+3}\}$ e sabendo que a predição do ângulo de torção $\alpha_{ii+1i+3}$ será realizada com base na lei dos co-senos para ângulos planos, podemos determinar o conjunto de informações necessárias nesta predição. Neste exemplo estas informações estão especificados através das linhas e arcos pontilhados.

4.3 A Estratégia SMEM Aplicada ao Problema da Predição das Conformações Moleculares Estáveis

O próximo passo será, na Subseção 4.3.1, descrever nossa rede neuronal. Falaremos sobre as estruturas utilizadas e, em seguida, na Subseção 4.3.2, apresentaremos o conjunto de restrições que especificam o SMCPP, convertendo-o para um problema de satisfatibilidade. Posteriormente, na Subseção 4.3.3, nosso conjunto de restrições, que neste ponto, passa a ser um conjunto de fórmulas lógicas, será convertido para uma função de energia que especifica as ligações que, por sua vez, completam a definição da arquitetura da rede neuronal que utilizaremos para simular o SMCPP.

4.3.1 A Arquitetura da Rede

Na Subseção 2.3.3, o sistema molecular $SM = (\mathcal{A}, L)$, utilizado na especificação do problema da predição das conformações moleculares estáveis, foi definido através de um grafo $G = (V, E)$. Da mesma forma, representaremos a rede neuronal que simula o SMCPP através de um grafo $G' = (V', E')$. Neste grafo, os elementos de

V' são os neurônios que constituem a rede, enquanto as arestas em E' representam as ligações entre k-uplas de vértices de V'. O estado de um neurônio em V' é representado através da variável $v_{i_1 i_2 ... i_k}$, onde a k-upla de índices $(i_1 i_2 ... i_k)$ varia com a dimensão da matriz que contém o neurônio que estiver sendo considerado. Por simplicidade, não faremos distinção entre um neurônio e seu estado (ambos estão sendo denotados por $v_{i_1 i_2 ... i_k}$). Além disso, a rede que utilizamos na simulação do SMCPP é constituída por seis estruturas neuronais principais que, por sua vez, requerem algumas estruturas neuronais auxiliares.

- Estruturas neuronais principais:

  $dist(i, j, b)$ Esta estrutura contém os neurônios que representam a distância, em bits b, entre pares de átomos $a_i, a_j \in \mathcal{A}$;

  $quaddist(i, j, l, b)$ Esta estrutura contém os neurônios que representam o quadrado da distância, em bits b, entre pares de átomos $a_i, a_j \in \mathcal{A}$;

  $cosalfa(i, j, k, b)$ Esta estrutura contém os neurônios que representam o co-seno, em bits b, dos ângulos de torção e dos ângulos entre pares de ligações covalêntes consecutivas em L, definidos por $(a_i, a_k), (a_k, a_j) \in L$;

  $pcartesiano(i, l, b)$ Esta estrutura contém os neurônios que representam as coordenadas cartesianas l ($l = 1 \rightarrow x$, $l = 2 \rightarrow y$ e $l = 3 \rightarrow z$), em bits b, de cada um dos átomos $a_i \in \mathcal{A}$;

  $LCosenos(i, j, k, b)$ Esta estrutura contém o valor, em bits b, da Equação 4.2.

  $DEuclidiana(i, j, b)$ Esta estrutura contém o valor, em bits b, da Equação 4.1.

- Estruturas neuronais auxiliares:

Estruturas utilizadas em cálculos binários

$um(b)$;

$menosdois(b)$;

Estruturas auxiliares na predição do quadrado das distâncias

$quaddistparcela(i, j, l, b)$;

$quaddistvaibit(i, j, l, b)$;

Estruturas auxiliares na predição da lei dos co-senos

$LCosenosprodvaibit(i, j, k, l, b)$;

$LCosenosprod(i, j, k, l, b)$;

$LCosenosprodsum(i, j, k, l, b)$;

$LCosenossum1(i, j, k, b)$;

$LCosenossum1vaibit(i, j, k, b)$;

$LCosenosvaibit(i, j, k, b)$;

Estruturas auxiliares na predição das distâncias euclidianas

$sum(i, j, l, b)$;

$sumvaibit(i, j, l, b)$;

$prodparcela(i, j, c, l, b)$;

$prod(i, j, c, l, b)$;

$prodvaibit(i, j, c, l, b)$;

$sumprod(i, j, b)$;

$sumprodvaibit(i, j, b)$;

$Deuclidianavaibit(i, j, b)$;

Lei de Coulomb e termo de Lennard-Jones

$ionico(i, j)$;

$polar(i, j)$;

Além das estruturas neuronais que definem os componentes da rede, utilizamos duas estrutuuturas auxiliares. Estas estruturas são responsáveis pelo armazenamento de dados numéricos, experimentais, que refletem a natureza do composto químico cuja geometria deve ser estimada. Estes dados formarão, junto com constantes multiplicativas, o peso das ligações nas quais eles influenciam. A primeira de nossas estruturas auxiliares é um vetor chamado *carga*. Ele armazena o valor, no domínio dos reais, da carga elétrica de cada átomo a_i que participa de ligaçẽs iônicas. A segunda estrutura auxiliar, a matriz bidimensional *disteq*, armazena a distância de equilíbrido entre cada par de átomos $a_i, a_j \in \mathcal{A}$ que se polarizam.

A seguir, na Subseção 4.3.2, apresentaremos o SMCPP sob a forma de restrições e o transformaremos em um problema de satisfatibilidade.

4.3.2 Mapeamento de Restrições para Satisfatibilidade

Nesta subseção, especificaremos o comportamento da rede através de um conjunto de restrições de viabilidade e de um conjunto de restrições de otimalidade. O primeiro conjunto apresenta as restrições que garantem a consistência de uma possível solução e o segundo apresenta as restrições necessárias para que uma solução seja ótima. Nossos conjuntos de restrições de viabilidade e de otimalidade serão determinados com base no grafo $G = (V, E)$, associado a um sistema molecular $SM = (\mathcal{A}, L)$ qualquer.

- Restrições de Viabilidade:

i) O quadrado da distância, associada a uma aresta $(n_i, n_j) \in E$, é equivalente ao somatório do quadrado da diferença entre as coordenadas dos vértices incidentes a esta aresta. Matematicamente temos $dist_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2$.

$\forall i, \forall j \forall b | 1 \leq i \leq n-3, |i+1 \leq j \leq i+3, 1 \leq b \leq bits :$

$(\neg quaddist_{i,j,n,b} \wedge \neg DEuclidiana_{i,j,b}) \vee (quaddist_{i,j,n,b} \wedge DEuclidiana_{i,j,b})$

$\Leftrightarrow \varphi_1 = \vee_i \vee_j \vee_b$

$(\neg quaddist_{i,j,n,b} \wedge \neg DEuclidiana_{i,j,b}) \vee (quaddist_{i,j,n,b} \wedge DEuclidiana_{i,j,b})$

ii) O co-seno do ângulo formado por pares de arestas $(n_i, n_k), (n_k, n_j) \in E$ é equivalente a razão entre o somatório do quadrado das distâncias associadas as arestas $(n_i, n_k), (n_k, n_j), (n_i, n_j) \in E$ e o produto das distâncias associadas as arestas que o define. Matematicamente temos $cos(\alpha_{ikj}) = \frac{1}{2} \times \frac{dist_{kj}^2 + dist_{ki}^2 - dist_{ij}^2}{2 * dist_{kj} \times dist_{ki}})$. Entretanto, representaremos esta restrição da seguinte forma: $dist_{ij}^2 = dist_{kj}^2 + dist_{ki}^2 - (2 \times dist_{kj} \times dist_{ki} * cos(\alpha_{ikj}))$.

$\forall i, \forall j, \forall k, \forall b | 1 \leq i \leq n-3, j = i+3, k = i+1, 1 \leq b \leq bits :$

$(\neg quaddist_{i,j,n,b} \wedge \neg LCosenos_{i,j,k,b}) \vee (quaddist_{i,j,n,b} \wedge LCosenos_{i,j,k,b})$

$\Leftrightarrow \varphi_2 = \vee_i \vee_j \vee_k \vee_b$

$(\neg quaddist_{i,j,n,b} \wedge \neg LCosenos_{i,j,k,b}) \vee (quaddist_{i,j,n,b} \wedge LCosenos_{i,j,k,b})$

iii) Toda distância que corresponde a uma aresta em E deve ser diferente de zero:

$\forall i, \forall j, \exists b | 1 \leq i \leq n-3, |i+1 \leq j \leq i+3, 1 \leq b \leq bits : dist_{i,j,b}$

$\Leftrightarrow \varphi_3 = \vee_i \vee_j \wedge_b dist_{i,j,b}$

O conjunto de restrições de viabilidade utilizado na especificação do SMCPP é, na verdade, bem mais complexo, pois as restrições aqui apresentadas se desdobram em outros conjuntos de restrições. Isto ocorre porque as operações de soma, produto, etc., utilizadas nas Equações 4.1 e 4.2, são simuladas através de redes semelhantes às dos problemas aritméticos apresentados no Capítulo 3 e o resultado destas simulações é armazenado na estrutura correspondente. $DEuclidiana(i, j, b)$ no caso da Equação 4.1 e $LCosenos(i, j, k, b)$ no caso da Equação 4.2. Entretanto, por simplicidade, apenas as restrições principais foram apresentadas.

- Restrições de Otimalidade:

iv, v) Se dois vértices $v_i, v_j \in V$ estão associados a átomos que se relacionam através de interações iônicas ou polares, então a distância entre eles deve respeitar as relações impostas pela lei de Coulomb ou pelo termo de Lennard-Jones, respectivamente [1]:

$forall\ i,\ j,\ b | 1 \leq i \leq n-3, | i+1 \leq j \leq i+3, 1 \leq b \leq bits :$

$(\neg ionico_{i,j} \vee const_1(dist_{ijb}))$

$\Leftrightarrow \varphi_4 = \vee_i \vee_j \vee_b (\neg ionico_{i,j} \vee const_1(dist_{ijb}))$

$\forall i, \forall j, \forall b | 1 \leq i \leq n-3, | i+1 \leq j \leq i+3, 1 \leq b \leq bits :$

$(\neg polar_{i,j} \vee const_2(dist_{ijb}))$

$\Leftrightarrow \varphi_5 = \vee_i \vee_j \vee_b (\neg polar_{i,j} \vee const_2(dist_{ijb}) \vee const_3(dist_{ijb}))$

[1] Nas restrições que representam a lei de Coulomb e o termo de Lennard-Jones utilizamos constantes multiplicativas ($const_1, const_2$ e $const_3$) que farão parte do peso neuronal, quando as restrições forem mapeadas para uma função de energia.

As constantes utilizadas nas restrições iv) e v) são calculadas a partir do conjunto de equações 4.3:

$$\begin{aligned} const_1 &= (\frac{4\times\pi\times 2^{b-1}}{carga_i \times carga_j}); \\ const_2 &= (\frac{2^{12*(b-1)}}{disteq_{ij}^{1} 2\times\epsilon}); \\ const_3 &= (-\frac{2^{6*(b-1)}}{disteq_{ij}^{6}\times\epsilon}). \end{aligned} \tag{4.3}$$

Estas restrições possuem uma ordem de precedência, que é expressa através de constantes multiplicativas. No SMCPP as restrições de viabilidade i), ii) e iii) e o conjunto de restriçẽs em que elas se desdobram estão no topo da hierarquia, seguidas pelas restrições de otimalidade iv) e v). A mais alta precedência será expressa através da constante γ e a mais baixa precedência será representada pela constante β (equações apresentadas em 4.4).

$$\begin{aligned} &\gamma = (\text{número } \textit{total de} \text{ cláusulas } \beta \ \times \ (\beta \ + \ 1)) + h \\ &\beta = 1. \end{aligned} \tag{4.4}$$

A seguir, na Subseção 4.3.3, nosso problema será convertido em um problema de minimização de energia.

4.3.3 Satisfatibilidade Mapeada para Minimização de Energia

Utilizaremos o operador H^*, descrito no Capítulo 2, para mapear as fórmulas lógicas geradas na subseção anterior no conjunto $D = \{0, 1\}$. Subdividiremos nossa função de energia em duas partes:

- A primeira se refere às restrições de viabilidade e será representada por E_v;
- A segunda se refere às restrições de otimalidade e será representada por E_o.

Primeiro geraremos $E_v = \sum_{i=1}^{3} H^*(\neg\varphi_i)$:

- Dado $\varphi_1 = \vee_i \vee_j \vee_b$

$(\neg quaddist_{i,j,n,b} \wedge \neg DEuclidiana_{i,j,b}) \vee (quaddist_{i,j,n,b} \wedge DEuclidiana_{i,j,b})$

$\Rightarrow \neg\varphi_1 = \wedge_i \wedge_j \wedge_b$

$(quaddist_{i,j,n,b} \vee DEuclidiana_{i,j,b}) \wedge (\neg quaddist_{i,j,n,b} \vee \neg DEuclidiana_{i,j,b})$

Aplicando o operador H^* a fórmula lógica dada por $\neg\varphi_1$, obtemos a Equação 4.5.

$$H^*(\neg\varphi_1) = \sum_{i=1}^{n-3} \sum_{j=i+1}^{i+3} \sum_{b=1}^{bits} ((quaddist_{i,j,n,b} DEuclidiana_{i,j,b}) + (1 - quaddist_{i,j,n,b})(1 - DEuclidiana_{i,j,b})) \quad (4.5)$$

- Dado $\varphi_2 = \vee_i \vee_j \vee_k \vee_b$

$(\neg quaddist_{i,j,n,b} \wedge \neg LCosenos_{i,j,k,b}) \vee (quaddist_{i,j,n,b} \wedge LCosenos_{i,j,k,b})$

$\Rightarrow \neg\varphi_2 = \wedge_i \wedge_j \wedge_k \wedge_b$

$(quaddist_{i,j,n,b} \vee LCosenos_{i,j,k,b}) \wedge (\neg quaddist_{i,j,n,b} \vee \neg LCosenos_{i,j,k,b})$

Aplicando o operador H^* a $\neg\varphi_2$, obtemos a Equação 4.6.

$$H^*(\neg\varphi_2) = \sum_{i=1}^{n-3} \sum_{j=i+3} \sum_{k=i+1} \sum_{b=1}^{bits} ((quaddist_{i,j,n,b} LCosenos_{i,j,k,b}) + (1 - quaddist_{i,j,n,b})(1 - LCosenos_{i,j,k,b})) \quad (4.6)$$

- Dado $\varphi_3 = \vee_i \vee_j \wedge_b dist_{i,j,b}$

$\Rightarrow \neg\varphi_3 = \wedge_i \wedge_j \vee_b \neg dist_{i,j,b}$

Aplicando o operador H^* a $\neg\varphi_3$, obtemos a Equação 4.7.

$$H^*(\neg\varphi_3) = \sum_{i=1}^{n-3} \sum_{j=i+1}^{i+3} \prod_{b=1}^{bits} (1 - dist_{i,j,b}) \tag{4.7}$$

Agora geraremos $E_o = \sum_{i=3}^{4} H^*(\neg\varphi_i)$:

- Dado $\varphi_4 = \vee_i \vee_j \vee_b (\neg ionico_{i,j} \vee const_1(dist_{ijb}))$

$\Rightarrow \neg\varphi_4 = \wedge_i \wedge_j \wedge_b \vee_i \vee_j \vee_b (ionico_{i,j} \wedge const_1(\neg dist_{ijb}))$

Aplicando o operador H^* a $\neg\varphi_4$, obtemos a Equação 4.6.

$$H^*(\neg\varphi_4) = \sum_{i=1}^{n-3} \sum_{j=i+1}^{i+3} \sum_{b=1}^{bits} (ionico_{i,j} + const_1(1 - dist_{ijb})) \tag{4.8}$$

- Dado $\varphi_5 = \vee_i \vee_j \vee_b$

$(\neg polar_{i,j} \vee const_2(dist_{ijb}) \vee const_3(dist_{ijb})$

$\Rightarrow \neg\varphi_5 = \wedge_i \wedge_j \wedge_b$

$\vee_i \vee_j \vee_b (polar_{i,j} \wedge const_2(\neg dist_{ijb}) \wedge const_3(\neg dist_{ijb})$

Aplicando o operador H^* a $\neg\varphi_5$, obtemos a Equação 4.6.

$$H^*(\neg\varphi_5) = \sum_{i=1}^{n-3} \sum_{j=i+1}^{i+3} \sum_{b=1}^{bits} (polar_{i,j} + const_2(1 - dist_{ijb}) + const_3(1 - dist_{ijb})) \quad (4.9)$$

Finalmente, nossa função de energia é o resultado da combinação das Equações 4.5, 4.6, 4.7, 4.8 e 4.9 multiplicadas por seus respectivos pesos, expressos através da Equação 4.4.

Capítulo 5

Resultados Experimentais

Neste capítulo apresentaremos os resultados experimentais da simulação do TSP, do VCP e do MC-TSP (com $|V| = 4$, 8 e 16) e dos problemas somador (considerando dois, três e quatro números), produto (entre dois números) e módulo (da diferença entre pares de números de sinais opostos). Simulamos 15 instâncias de cada problema e os resultados obtidos foram comparados com as soluções encontradas pela função *fmincon*, do pacote de otimização, do programa MATLAB versão 6.5.0.180913*a*. A função *fmincon* tenta encontrar um mínimo de uma função escalar de n variáveis que possuem uma estimativa inicial utilizando métodos de programação não linear [32]. Além disso, o método de busca empregado pela função *fmincon*, assim como a busca estocástica realizada pelo sistema SATyrus, não garante que um mínimo global seja encontrado. Os experimentos não tiveram por objetivo uma validação exaustiva e abrangente dos métodos e sim, a obtenção de respostas relativas a uma avaliação limitada do comportamento so sistema SATyrus.

Todos os testes foram realizados em um processador Pentium IV de 2.4 GHz, com 512 MB de memória RAM e sistemas operacionais Linux Fedora CORE 1, Kernel 2.4.22 e Windows XP Professional.

Aos parâmetros de inicialização do simulador SATyrus atribuímos os seguintes valores:

$T_{inicial} = 10.000$;

$T_{final} \rightarrow 0$;

$\varepsilon = \{0.9,\ 0.99,\ 0.999\}$; (em média)

As soluções obtidas através do sistema SATyrus foram melhores do que as soluções obtidas pela função *fmincon*, mesmo nas instâncias em que uma solução viável não foi encontrada. Por outro lado, a dificuldade em encontrar uma solução ótima aumentou, significativamente, com o número de componentes da rede. Observamos que neste caso as constantes multiplicativas (α, β, etc.) associadas às restrições de menor nível na escala de prioridade assumiram, durante o processo de cálculo automático, valores bem menos significativos do que as demais. Uma solução imediata foi calcular experimentalmente o valor da constante associada ao menor nível de precedência, mas supomos que a discrepância entre o valor atribuído a constante multiplicativa que representa o nível de precedência inicial e as demais esteja tornando a superfície da função de energia difícil de ser percorrida e, conseqüentemente, dificultando o processo de busca. Outro fator crítico é que em muitas instâncias utilizadas nos testes a diferença entre o pior custo e o custo ótimo foi pequena (da ordem das unidades).

As Tabelas 5 e 5 apresentam o desempenho do sistema SATyrus e do programa MATLAB (função *fmincon*) na busca pelas soluções dos problemas descritos no Capítulo 3.

Tabela 5.1: Resultados Experimentais: Problemas NP-Difíceis.

TSP					
	SATyrus			MATLAB (*fmincon*)	
Número de Cidades	Número de Neurônios da Rede	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
4	25	15	15	12	2
8	81	15	12	10	2
16	289	15	14	7	1

VCP					
	SATyrus			MATLAB (*fmincon*)	
Número de Vértices	Número de Neurônios da Rede	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
4	20	15	9	12	2
8	72	13	7	9	1
16	272	13	6	7	1

MC-TSP					
	SATyrus			MATLAB (*fmincon*)	
Número de Cidades	Número de Neurônios da Rede	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)	Soluções Viáveis Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
4	45	13	7	9	1
8	153	12	7	7	1
16	561	12	4	4	1

Tabela 5.2: Resultados Experimentais: Operações Aritméticas.

SOMADOR			
	SATyrus		MATLAB (*fmincon*)
Quantidade de Números {Bits (b)}	Número de Neurônios da Rede	Soluções Ótimas Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
$2\{4-16\}$	$4b+1$	14	8
$3\{4,8,16\}$	$7b+2$	12	7
$4\{4,8,16\}$	$10b+3$	9	7

PRODUTO			
	SATyrus		MATLAB (*fmincon*)
Quantidade de Números {Bits}	Número de Neurônios da Rede	Soluções Ótimas Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
$2\{4,8,16\}$	$3b^2+b-1$	13	6

MÓDULO			
	SATyrus		MATLAB (*fmincon*)
Quantidade de Números {Bits}	Número de Neurônios da Rede	Soluções Ótimas Encontradas (Dentre 15)	Soluções Ótimas Encontradas (Dentre 15)
$2\{4,8,16\}$	$9b+2$	13	8

A Figura 5.1apresenta o gráfico da evolução da energia e o resultado da simulação de uma instância do problema do caixeiro viajante com 16 cidades.

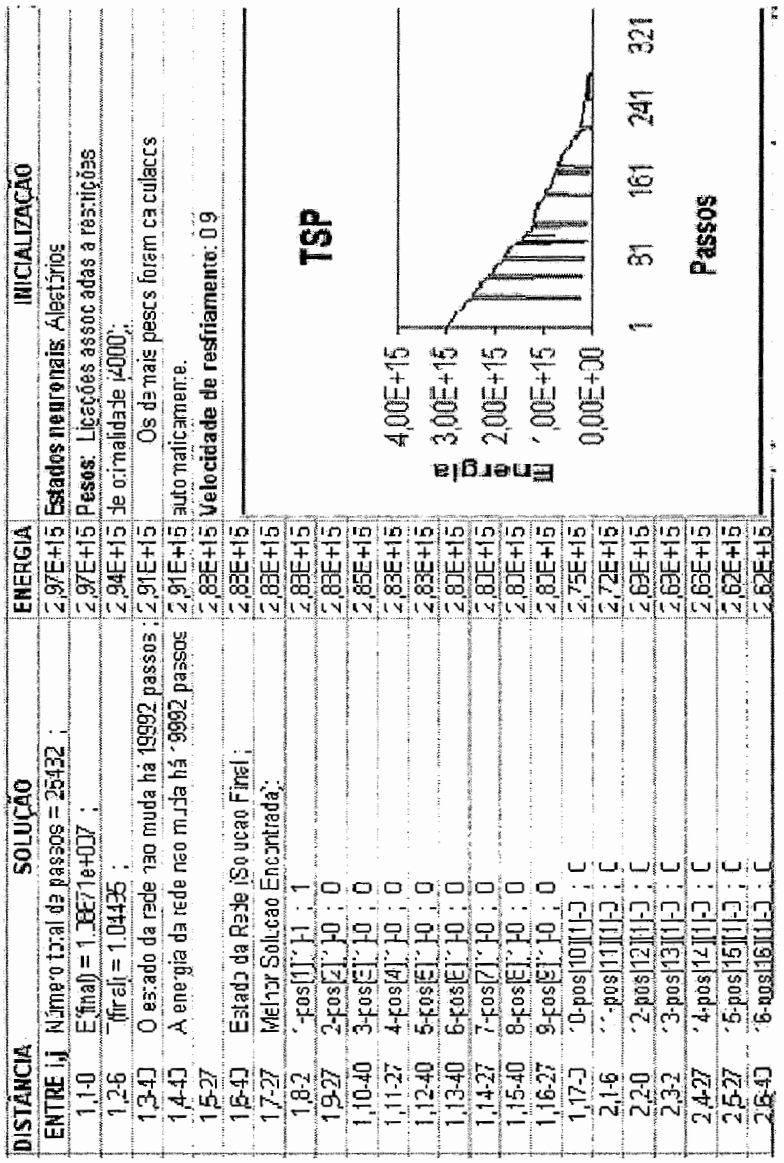


Figura 5.1: Parte da solução para uma instância do TSP com 16 cidades e o gráfico da evolução da energia da rede em função dos passos de atualização.

Capítulo 6

Conclusão

Neste trabalho, apresentamos a implementação de uma estratégia híbrida que combina a estrategia SMEM e o mecanismo de busca associado às redes de Hopfield estocásticas e propusemos um modelo híbrido para o problema da predição das conformações moleculares estáveis. Verificamos que a estratégia SMEM utiliza mecanismos lógicos que permitem modelar de forma natural conjuntos de restrições, que definem os problemas que desejamos resolver, através de uma rede neuronal artificial, possivelmente de alta ordem e representada por uma função de energia.

Diante da dificuldade de tratar problemas NP-difíceis, esta estratégia auxilia minimizando as complicações que envolvem a especificação destes problemas, ao convertê-los de forma natural, em um problema de minimização de energia. Além disso, ela possibilita o tratamento de problemas híbridos. Isto foi visto através da especificação do problema MC-TSP e do problema SMCPP. Outro fator importante que motivou a escolha da estratégia SMEM para ser utilizada na modelagem dos problemas estudados neste trabalho é que ao gerar uma função de energia a partir da especificação de um problema estamos obtendo sua formulação exata a partir da descrição dos mesmos.

Outro ponto interessante é que a estratégia SMEM tem se mostrado promissora em modelos que combinam linguagem pseudo-natural e fórmulas matemáticas. Em outras palavras, esta estratégia tem permitido a associação entre descrição matemática formal (fórmulas matemáticas) e linguagem pseudo-natural (restrições especificadas através de dialeto comum) em um mesmo modelo.

Nossas principais contribuições foram:

1. O desenvolvimento do simulador neuronal que compõe o sistema computacional SATyrus [12];
2. A modelagem de algumas operações aritméticas binárias (soma / subtração, modulo e produto), do VCP e do MC-TSP através da estratégia SMEM;
3. A proposta de um modelo matemático para o SMCPP, no qual este problema é tratado como um problema híbrido que combina um modelo clássico e um modelo geométrico (MDGP-Modificado [13]).

Como possíveis trabalhos futuros apontamos:

1. O desenvolvimento de um simulador neuronal distribuído, possivelmente em uma grade computacional (*GRID*);
2. O uso e combinação de outras técnicas heurísticas na simulação dos problemas mapeados através da estratégia SMEM. Como exemplo podemos citar os Algoritmos Genéticos [33];
3. O aperfeiçoamento do modelo matemático para o problema das conformações moleculares estáveis (SMCPP) proposto neste trabalho, a partir do acréscimo de restrições que especifiquem interações quânticas;

4. O estudo de métodos de suavização de superfícies [34], esperando que estas técnicas possam ser utilizadas com o objetivo de facilitar o processo de busca nos casos em que a funções de energia possuir uma superfície difícil de ser percorrida.

Referências Bibliográficas

[1] Hopfield, J.J., Tank, D.W. "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, v. 52, n. 3, pp. 141-152, Jul. 1985.

[2] Barbosa, V.C., *Massively Parallel Models of Computation: Distributed Parallel Processing in Artificial Intelligence and Optimisation*. West Sussex, Ellis Horwood, 1993.

[3] Jones, A.J., *Models of Living Systems: Evolution and Neurology*. Lecture Notes, Department of Computing - Imperial College of Science,Technology and Medicine, London, UK, 1994.

[4] Carvalho, L.A.V., Barbosa, V. C., *Towards a Stochastic Neural Model for Combinatorial Optimization*. Technical Report ES-196/89, COPPE/UFRJ, Rio de Janeiro, Brasil, 1989.

[5] Cheng, J., Baldi, P. "Three-Stage Prediction of Protein Beta-Sheets by Neural Networks, Alignments, and Graph Algorithms". In: *Proceedings of the 2005 Conference on Intelligent Systems for Molecular Biology - ISMB 05*, v. 21, pp. 75-84, Detroit, Jun. 2005.

[6] Cheng, J., Baldi, P. "A Machine Learning Information Retrieval Approach to Protein Fold Recognition", *Bioinformatics*, v. 22, n. 12, pp. 1456-1463, Mar. 2006.

[7] McAllister, S.R., Floudas, C.A. "Structure Prediction of Alpha-Helical Proteins". In: *Proceedings of the International Symposium on Mathematical and Computational Biology - BIOMAT 2005*, v. 3515, pp. 265-288, Rio de Janeiro, 2006.

[8] Cheng, J., Baldi, P. "DOMpro: Protein Domain Prediction Using Profiles, Secondary Structure, Relative Solvent Accessibility, and Recursive Neural Networks", *Data Mining and Knowledge Discovery*, v. 13, n. 1, pp. 1-10, Jul. 2006.

[9] Van, G.W.F., Berendsen, H.J.C. "Computer Simulation of Molecular Dynamics: Metodology, Applications and Pespectives in Chemistry". *Angew. Chem. Int.*, v. 29, n. 9, pp. 992-1023, Jan. 1990.

[10] Lima, P.M.V., *Resolution-Based Inference on Artificial Neural Network*. Ph.D. dissertation, Department of Computing - Imperial College of Science,Technology and Medicine, London, UK, 2000.

[11] Lima, P.M.V., Pereira, G.C., Morveli-Espinoza, M.M.M., et al. "Mapping and Combining Combinatorial Problems into Energy Landscapes via Pseudo-Boolean Constraints". In: *Lecture Notes in Computer Science*, v. 3704, pp. 308-317, Berlin, Oct. 2005.

[12] Lima, P.M.V., Morveli-Espinoza, M.M.M., Pereira, G.C., et al. "SATyrus: A SAT-based Neuro-Symbolic Architecture for Constraint Processing". In: *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems - HIS'05*, v. 1, pp. 137-142, Los Alamitos, Nov. 2005.

[13] Lima, P.M.V., Pereira, G.C., Morveli-Espinoza, M.M.M., et al. "Mapping Molecular Geometry Problems into Pseudo-Boolean Constraints". In: *Proceedings*

of the 4th. International Workshop on Genomic Databases - IWGD'05, Rio de Janeiro, Nov. 2005.

[14] Pinkas, G., *Logical Inference in Symmetric Neural Networks*. Ph.D. dissertation, Sever Institute of Technology - Washington University, Saint Louis, Missouri, USA, 1992.

[15] Hopfield, J.J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". *Proceedings of the National Academy of Sciences of the United States of American*, v. 79, n. 8, pp. 2554-2558, Apr. 1982.

[16] Kirkpatrick, S., Gellat, Jr.C.D., Vecchi, M.P. "Optimization via Simulated Annealing", *Science*, v. 220, n. 4598, pp. 671-680, May. 1983.

[17] Griffeath, D., "Introduction to Randon Fields". In: Kennedy, J.G., Snell, J.L., Knapp, A.W. (eds), *Denumerable Markov Chains*, pp. 425-458, New York, USA, Springer-Verlag, 1976.

[18] Grimmett, G.R. "A Theorem about Random Fields", *Bull. of the London Mathematical Society*, v. 5, pp. 81-84, 1973.

[19] Geman, S., Geman, D. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence - PAMI*, v. 6, n. 6, pp. 721-741, Nov. 1984.

[20] Enderton, H.B., *A Mathematical Introduction to Logic*. New York, Academic Press, 1972.

[21] Barbosa, V.C., Gafni, E. "A Distributed Implementation of Simulated Annealing", *Parallel and Distributed Computing*, v. 6, n. 2, pp. 411-434, Apr. 1989.

[22] BrooksIII, C.L., Karplus, M., Pettitt, B.M. *Proteins. A Theoretical Perspective of Dynamics, Structure, and Thermodynamics*, v. 71, *Advances in Chemical Physics*, New York, USA, John Wiley & Sons, 1988.

[23] Yoon, J.M., Gad, Y., Wu, Z., *Mathematical Modeling of Protein Structure Using Distance Geometry*. Technical Report TR00-24, Department of Computer Applied Maths, Rice University, Houston, USA, 2000.

[24] Barbosa, H., Lavor, C., Raupp, F. "A GA-Simplex Hybrid Algorithm for Global Minimization of Molecular Potential Energy Functions", *Annals of Operations Research*, v. 138, n. 1, pp. 189-202, Sep. 2005.

[25] Brooks B., Karplus M. "Harmonic Dynamics of Proteins: Normal Modes and Fluctuations in Bovine Pancreatic Trypsin Inhibitor". *National Academy of Sciences*, v. 80, n. 21, pp. 6571-6575, Nov. 1983.

[26] Weiner, S.J., Kollman, P.A., Case, U.C., at al. "A New Force Field for Molecular Simulation of Nucleic Acid and Proteins", *American Chemical Society*, v. 106, n. 3, pp. 765-784, 1984.

[27] Moré, J., Wu, Z., Vecchi, M.P. "Distance Geometry Optimization for Protein Structures", *Global Optimization*, v. 15, n. 3, pp. 219-234, Oct. 1999.

[28] Santos, D.J.V.A., *Estudos Teóricos da Química de Interfaces*. Tese de D.Sc., Faculdade de Ciências - Universidade do Porto, Porto, Portugal, 2003.

[29] Pascutti, P.G., "Introdução à Modelagem e Dinâmica Molecular". In: Pascutti, P.G. (org), *Introdução à Modelagem e Dinâmica Molecular*, v. 1, pp. 1-38, 2002.

[30] Morveli-Espinoza, M.M.M., *Compilando Resolução de Problemas para Minimização de Energia*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2006.

[31] Lavor, C., "On Generating Instances for the Molecular Distance Geometry Problem". In: *Global Optimization: from Theory to Implementation*, v. 84, *Nonconvex Optimization and its Applications*, Springer-Verlag, pp. 405-414, 2006.

[32] Han, S.S.P. "A Globally Convergent Method for Nonlinear Programming", *Optimization Theory and Applications*, v. 3, n. 3, pp. 279-309, Jul. 1977.

[33] Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York, Addison-Wesley, 1989.

[34] Xavier, A.E., Chaves, A.M.V. "Problemas MINIMAX: Uma Alternativa de Resolução Via Suavização". In: *IX Congreso Latinoiberoamericano de Investigacion Operativa*, v. 32, pp. 33, Buenos Aires, Sep. 1998.

[35] Szwarcfiter, J.L., *Grafos e Algoritmos Computacionais*. Rio de Janeiro, Campus, 1984.

[36] Garey, M.R., Jhonson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, W.H. Freeman and Company, 1979.

Apêndice A

Problemas NP-Difíceis

[1]Um *Problema Não-Deterministicamente Polinomial* é um problema computável cujas soluções até então conhecidas são de ordem exponencial e não se sabe se existe uma solução melhor, de complexidade polinomial. O conjunto de todos os problemas não-deterministicamente polinomiais é denominado como a classe de problemas NP. Por outro lado, o subconjunto da classe NP que contém os problemas para os quais pode-se encontrar solução em tempo polinomial é a classe denotada por P.

Algorítmos com complexidade polinomial são considerados computacionalmente tratáveis, pois requerem um tempo de execução limitadado por uma função polinomial.

Um problema é dito NP-Completo se ele é representante da classe de problemas NP, de forma que os outros problemas da classe são redutíveis a ele em tempo polinomial. Neste caso, dizemos que um problema é NP-Difícil se ele possui a característica de representatividade de um problema NP-Completo, porém não se exigindo sua pertinência a NP. A Figura A.1 apresenta a relação entre as classes de problemas P, NP, NP-Completo e NP-difícil.

[1]Este capítulo se baseia em [35] e em [36].

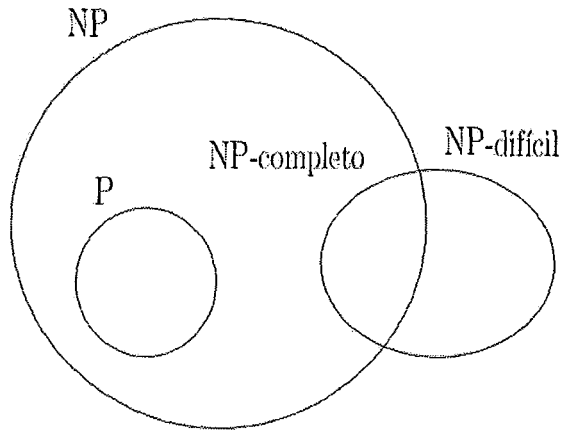


Figura A.1: Relação entre as classes de problemas P, NP, NP-completo e NP-difícil.

Apêndice B

Código para Compilação dos Problemas Simulados

TSP

```
// Declaracoes:

num=5;

// Estruturas:

pos(num,num);

dist (num,num);

dist read from tsp1.txt;

//Restricoes de Viabilidade:

integrity group type int1:
forall{i,j; 1¡=i¡=num,1¡=j¡=num:
pos[i][j];
```

```
integrity group type wta:
forall{i,j,k};1¡=i¡=num,1¡=j¡=num,1¡=k¡=num;i!=k:
(not pos[i][j] or not pos[k][j]);
integrity group type wta:
forall{i,j,l};1¡=i¡=num,1¡=j¡=num,1¡=l¡=num;j!=l:
(not pos[i][j] or not pos[i][l]);

    //Restricoes de Otimalidade:

    optimality group type costo:
forall{i,j,k};1¡=i¡=num,1¡=j¡=num-1,1¡=k¡=num;i!=k:
dist[i][k](pos[i][j] and pos[k][j+1]);

    //Prioridades:

    penalty {
wta is level 2;
int1 is level 1;
costo is level 0;}
```

```
    //Declaracoes:

    num=4;

    //Estruturas:

    neigh(num,num);
vc(num,num);
col(num);

    //Restricoes de Viabilidade:

    integrity group type int1:
forall{i,k};1¡=i¡=num,1¡=k¡=num:
vc[i][k];
integrity group type wta:
forall{i,l,k};1¡=i¡=num,1¡=l¡=num,1¡=k¡=num;i!=l:
(not neigh[i][l] or not vc[i][k] or not vc[l][k]);
integrity group type wta:
forall{i,k,m};1¡=i¡=num,1¡=k¡=num,1¡=m¡=num;k!=m:
(not vc[i][k] or not vc[i][m]);

    //Restricoes de Otimalidade:

    optimality group type wta:
forall{i,k};1¡=i¡=num,1¡=k¡=num:
(not vc[i][k] or col[k]);
optimality group type costo:
```

```
forall{k};1¡=k¡=num: col[k];

    //Penalidades:

    penalty {
wta is level 2;
int1 is level 1;
costo is level 0;}
```

```
    //Declaracoes:

    num=4;

    //Estruturas:

    neigh(num,num);
vc(num,num);
col(num);
pos(num,num);
dist(num,num);


dist read from tsp1.txt;

    //Restricoes de Viabilidade:

    integrity group type gama:
forall{i,k};1¡=i¡=num,1¡=k¡=num:
vc[i][k];
integrity group type gama:
forall{i,l,k};1¡=i¡=num,1¡=l¡=num,1¡=k¡=num;i!=l:
(not neigh[i][l] or not vc[i][k] or not vc[l][k]);
integrity group type sigma:
forall{i,k,m};1¡=i¡=num,1¡=k¡=num,1¡=m¡=num;k!=m:
(not vc[i][k] or not vc[i][m]);
integrity group type gama:
```

forall{i,j};1≤i≤num,1≤j≤num:

pos[i][j];

integrity group type sigma:

forall{i,j,k};1≤i≤num,1≤j≤num,1≤k≤num;i!=k:

(not pos[i][j] or not pos[k][j]);

integrity group type sigma:

forall{i,j,l};1≤i≤num,1≤j≤num,1≤l≤num;j!=l:

(not pos[i][j] or not pos[i][l]);

//Restricoes de Otimalidade:

optimality group type gama:

forall{i,k};1≤i≤num,1≤k≤num:

(not vc[i][k] or col[k]);

optimality group type beta:

forall{k};1≤k≤num:

col[k];

optimality group type costo:

forall{i,j,k};1≤i≤num,1≤j≤num-1,1≤k≤num;i!=k:

dist[i][k](pos[i][j] and pos[k][j+1]);

optimality group type alfa:

forall{i,k,j,c,s; 1≤i≤num,1≤k≤num-1,1≤j≤num,1≤c≤num,1≤s≤num-1;i!=j,c!=s:

(pos[i][k] and pos[j][k+1] and vc[i][c] and vc[j][s]);

//Penalidades:

penalty {

sigma is level 4;

gama is level 3;

beta is level 2;

alfa is level 1;

costo is level 0;}

```
    //Declaracoes:

    bits=2;

    //Estruturas:

    number1(bits);
number2(bits);
resultado(bits);
vaium(bits+1);

    //Restricoes:

    integrity group type alfa:
forall{b};1¡=b¡=bits:
(vaium[b] or number1[b] or not number2[b] or resultado[b]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(vaium[b] or not number1[b] or number2[b] or resultado[b]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(not vaium[b] or number1[b] or number2[b] or resultado[b]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(not vaium[b] or not number1[b] or not number2[b] or resultado[b]);
integrity group type alfa:
```

forall{b};1¡=b¡=bits:

(vaium[b] or number1[b] or number2[b] or not resultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not vaium[b] or not number1[b] or number2[b] or not resultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

not vaium[b] or number1[b] or not number2[b] or not resultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(vaium[b] or not number1[b] or not number2[b] or not resultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not vaium[b] or not number1[b] or not number2[b] or vaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not vaium[b] or not number1[b] or number2[b] or vaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(vaium[b] or not number1[b] or not number2[b] or vaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not vaium[b] or number1[b] or not number2[b] or vaium[b + 1]);

integrity group type alfa: forall{b};1¡=b¡=bits:

vaium[b] or number1[b] or number2[b] or not vaium[b + 1]);

```
integrity group type alfa:
forall{b};1¡=b¡=bits:
(vaium[b] or number1[b] or not number2[b] or not vaium[b + 1]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(vaium[b] or not number1[b] or number2[b] or not vaium[b + 1]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(not vaium[b] or number1[b] or number2[b] or not vaium[b + 1]);

    //Penalidades:

    penalty {
alfa is level 0;}
```

```
    //Declaracoes:

    num=3;
bits=6;
bitsmenos=bits-1;
bitsmais=bits+1;
fixo=1;
parcelasumprod=2;

    //Estruturas:

    number1(1¡=b¡=bits);
number2(1¡=b¡=bits);
parcela(1¡=l¡=bits,l¡=b¡=bits);
vaibit(1¡=l¡=bitsmenos,1¡=b¡=bitsmais);
resultado(1¡=l¡=bitsmenos,1¡=b¡=bits);

    //Restricoes:

    integrity group type alfa:
forall{l,b,v};1¡=l¡=bits,1¡=b¡=bitsmais-l;v=l-1:
(parcela[l][b+v] or not number1[l] or not number2[b]);
integrity group type alfa:
forall{l,b,v};1¡=l¡=bits,1¡=b¡=bitsmais-l;v=l-1:
(not parcela[l][b+v] or number1[l] or number2[b]);
integrity group type alfa:
```

forall{l,b,v};1≤l≤bits,1≤b≤bitsmais-l;v=l-1:

(not parcela[l][b+v] or number1[l] or not number2[b]);

integrity group type alfa:

forall{l,b,v};1≤l≤bits,1≤b≤bitsmais-l;v=l-1:

(not parcela[l][b+v] or not number1[l] or number2[b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(not vaibit[l][b] or resultado[l-1][b] or not parcela[l+1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(vaibit[l][b] or resultado[l-1][b] or not parcela[l+1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(vaibit[l][b] or not resultado[l-1][b] or parcela[l+1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(not vaibit[l][b] or resultado[l-1][b] or parcela[l+1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(vaibit[l][b] or resultado[l-1][b] or parcela[l+1][b] or not resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(not vaibit[l][b] or not resultado[l-1][b] or parcela[l+1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2≤l≤bits-1,l+1≤b≤bits:

(not vaibit[l][b] or resultado[l-1][b] or not parcela[l+1][b] or not resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(vaibit[l][b] or not resultado[l-1][b] or not parcela[l+1][b] or not resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,1¡=b¡=l:

(resultado[l-1][b] or not resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,1¡=b¡=l:

(not resultado[l-1][b] or resultado[l][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or parcela[a-1][b] or not parcela[a][b] or resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or parcela[a-1][b] or not parcela[a][b] or resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or not parcela[a-1][b] or parcela[a][b] or resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or parcela[a-1][b] or parcela[a][b] or resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or parcela[a-1][b] or parcela[a][b] or not resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or not parcela[a-1][b] or parcela[a][b] or resultado[a-1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or parcela[a-1][b] or not parcela[a][b] or not resultado[a- 1][b]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or not parcela[a-1][b] or not parcela[a][b] or not resultado[a- 113 1][b]);

integrity group type alfa:

forall{i,j,p,q,a; 1¡=i¡=num-1,i+1¡=j¡=num,1¡=p¡=num-1,p+1¡=q¡=num;a=fixo:

(parcela[a][a] or not resultado[a][a]);

integrity group type alfa:

forall{i,j,p,q,a; 1¡=i¡=num-1,i+1¡=j¡=num,1¡=p¡=num-1,p+1¡=q¡=num;a=fixo:

(not parcela[a][a] or resultado[a][a]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(not vaibit[l][b] or not resultado[l-1][b] or not parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(not vaibit[l][b] or not resultado[l-1][b] or parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(vaibit[l][b] or not resultado[l-1][b] or not parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(not vaibit[l][b] or resultado[l-1][b] or not parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(vaibit[l][b] or resultado[l-1][b] or parcela[l+1][b] or not vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(vaibit[l][b] or resultado[l-1][b] or not parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits:

(vaibit[l][b] or not resultado[l-1][b] or parcela[l+1][b] or vaibit[l][b + 1]);

integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,l+1¡=b¡=bits: not vaibit[l][b] or resultado[l-1][b] or parcela[l+1][b] or not vaibit[l][b + 1])};114 integrity group type alfa:

forall{i,j,p,q,l,b; 2¡=l¡=bits-1,1¡=b¡=l+1: not vaibit[l][b]; integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or not parcela[a-1][b] or not parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or not parcela[a-1][b] or parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or not parcela[a-1][b] or not parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or parcela[a-1][b] or not parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or parcela[a-1][b] or parcela[a][b] or not vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or parcela[a-1][b] or not parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(vaibit[a-1][b] or not parcela[a-1][b] or parcela[a][b] or vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,b,a; 2¡=b¡=bits;a=parcelasumprod:

(not vaibit[a-1][b] or parcela[a-1][b] or parcela[a][b] or not vaibit[a-1][b + 1]);

integrity group type alfa:

forall{i,j,p,q,f; 1¡=i¡=num-1,i+1¡=j¡=num,1¡=p¡=num-1,p+1¡=q¡=num;f=fixo:

not vaibit[f][f];

//Penalidades:

penalty {

alfa is level 0;}

```
    //Declaracoes:

    bits=2;

    //Estruturas:

    number1(bits);
number2(bits);
svaium(bits+1);
sresultado(bits);
complemento(bits);
cresultado(bits);
um(bits);
mresultado(bits);
vaium(bits+1);

    //Restricoes:

    integrity group type alfa:
forall{b};1¡=b¡=bits:
(svaium[b] or number1[b] or not number2[b] or sresultado[b]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
(svaium[b] or not number1[b] or number2[b] or sresultado[b]);
integrity group type alfa:
forall{b};1¡=b¡=bits:
```

(not svaium[b] or number1[b] or number2[b] or sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or not number1[b] or not number2[b] or sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or number1[b] or number2[b] or not sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or not number1[b] or number2[b] or not sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or number1[b] or not number2[b] or not sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or not number1[b] or not number2[b] or not sresultado[b]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or not number1[b] or not number2[b] or svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or not number1[b] or number2[b] or svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or not number1[b] or not number2[b] or svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or number1[b] or not number2[b] or svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or number1[b] or number2[b] or not svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or number1[b] or not number2[b] or not svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(svaium[b] or not number1[b] or number2[b] or not svaium[b + 1]);

integrity group type alfa:

forall{b};1¡=b¡=bits:

(not svaium[b] or number1[b] or number2[b] or not svaium[b + 1]);

integrity group type beta:

forall{b,f};1¡=b¡=bits;f=bits:

(sresultado[f] or (mresultado[b] and sresultado[b]) or (not mresultado[b] and not sresultado[b]));

integrity group type beta:

forall{{b,f};1¡=b¡=bits;f=bits:

(not sresultado[f] or (mresultado[b] and not complemento[b]) or (not mresultado[b] and complemento[b]));

integrity group type beta:

forall{b,f};1¡=b¡=bits;f=bits:

(not sresultado[f] or (mresultado[b] and cresultado[b]) or (not mresultado[b] and not cresultado[b])); integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or complemento[b] or not um[b] or cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or not complemento[b] or um[b] or cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or complemento[b] or um[b] or cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or not complemento[b] or not um[b] or cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or complemento[b] or um[b] or not cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or not complemento[b] or um[b] or not cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or complemento[b] or not um[b] or not cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or not complemento[b] or not um[b] or not cresultado[b]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or not complemento[b] or not um[b] or vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or not complemento[b] or um[b] or vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or not complemento[b] or not um[b] or vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or complemento[b] or not um[b] or vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or complemento[b] or um[b] or not vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or complemento[b] or not um[b] or not vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(vaium[b] or not complemento[b] or um[b] or not vaium[b + 1]);

integrity group type alfa: forall{b}; 1<=b<=bits:

(not vaium[b] or complemento[b] or um[b] or not vaium[b + 1]);

penalty {

alfa is level 0;

beta is level 1;}

SMCPP

CD em anexo.