4
Técnicas da Inteligência Computacional

A Inteligência Computacional (IC) é um ramo da ciência da computação que consiste em um conjunto de técnicas e algoritmos os quais são capazes de modelar o comportamento inteligente que existe de forma intrínseca nos sistemas biológicos. O interesse nesses "sistemas inteligentes" advém da sua habilidade em solucionar, de forma mais eficiente do que os métodos de otimização convencionais ou programação matemática, problemas consideravelmente complexos, nos quais existem múltiplos objetivos e múltiplas variáveis, as quais normalmente estão correlacionadas e fazem parte de um ambiente dinâmico, ou seja, que varia com o tempo. Dentre esses algoritmos inteligentes podemos destacar os seguintes: Redes Neurais Artificiais (RNA), Computação Evolucionária (CE), Otimização por Colônia de Formigas (ACO), Sistemas Imunes Artificiais (SIA) e Sistemas Fuzzy (SF).

4.1
Redes Neurais Artificiais (RNA)

Inspirada na estrutura e funcionamento do cérebro humano, uma Rede Neural Artificial (RNA) é um modelo matemático não-linear usado para realizar um mapeamento entre o espaço das variáveis de entrada e das variáveis de saída de conjunto de dados. Esse modelo tem sido aplicado principalmente aos problemas de classificação, otimização, reconhecimento de padrões, aproximação de funções e predição de séries temporais. As RNAs se diferenciam por três aspectos básicos: o modelo do neurônio do neurônio artificial, sua estrutura de interconexão (topologia) e o tipo de aprendizado.

Assim como o sistema nervoso é composto por bilhões de neurônios, a RNA também é formada por unidades elementares, denominadas neurônios artificiais ou processadores, que efetuam operações simples. Nas redes, esses neurônios encontram-se

interconectados e transmitem seus resultados aos processadores vizinhos. As RNAs são eficazes na elaboração de funções capazes de aproximas dados não-lineares, incompletos, com ruído ou compostos por exemplos contraditórios, sendo essa capacidade de modelar sistemas não lineares a sua principal vantagem sobre outros métodos de interpolação. Na maioria dos casos, uma RNA é um sistema adaptável que muda sua estrutura com base na informação externa ou interna da rede.

4.1.1
Estrutura de uma RNA

O Neurônio Artificial

O neurônio artificial *i*, tipicamente denominada de "elemento processador", é inspirado no neurônio biológico, possuindo um conjunto de entradas x_m (dendritos) e uma saída y_i (axônio), conforme ilustrado na Fig. 4.1. As entradas são ponderadas por pesos sinápticos w_{im} (sinapses), que determinam o efeito da entrada x_m sobre o processador *i*. Estas entradas ponderadas são somadas, fornecendo o potencial v_i do processador. A saída ou estado de ativação y_i do elemento processador *i* é finalmente calculada através de uma função cuja função é reforçar ou enfraquecer o sinal de entrada. Cada uma das entradas de ativação $\varphi(.)$ é então multiplicada pelos seus respectivos pesos e é feita a soma desses produtos. O resultado é fornece o potencial interno v_i do processador. O sinal de saída ou estado de ativação y_i do elemento processador *i* é calculada através de uma função especial, designada função de ativação Φ (.). O estado de ativação pode ser definido pela Eq. 4.1.

$$y_i = \phi\left(\sum_{j=1}^{m} x_j w_j + \theta_i\right) \qquad (4.1)$$

onde *m* é o número de entradas do neurônio *i* e θ_i é o termo de polarização (*bias*).

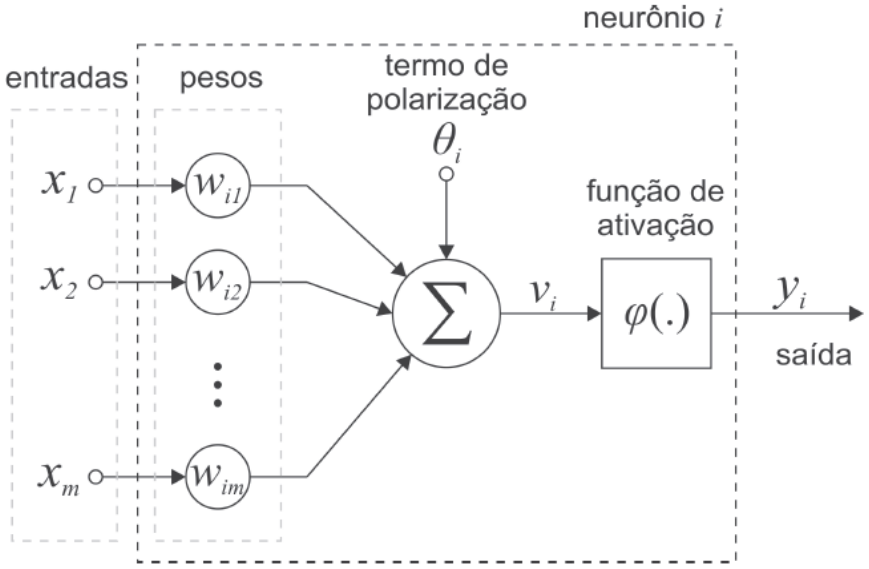

Figura 4.1: Representação gráfica de um neurônio artificial.

4.1.2
Função de Ativação

A função de ativação é responsável por majorar ou minorar a informação recebida por um neurônio antes de passá-la adiante. Qualquer função pode ser utilizada para tal fim, contanto que seja diferenciável. A diferenciabilidade é uma característica importante na teoria das redes neurais, possibilitando o seu uso em algoritmos de treinamento baseados no método do gradiente. Dentre as funções de ativação usuais têm-se as seguinte: linear, linear saturada, sinal, degrau, tangente hiperbólica e logística.

A função linear é geralmente utilizada quando a saída do neurônio pode atingir qualquer valor, ou seja, não possui valores limites. Quando há muitas entradas nesse neurônio, tal função pode produzir valores elevados nos resultados. Tal função é geralmente empregada nos neurônio da camada de saída.

As funções sigmóides representam uma família de funções cujo gráfico tem a forma de 's'. Tal forma é a mais comumente utilizada, sendo definida como uma função estritamente crescente que apresenta intervalos lineares e não-lineares em sua estrutura. Esta função possibilita o mapeamento de dados com comportamento semelhante. Um exemplo de função sigmóide é a função logística, em que os valores de saída dos neurônios são unipolares e estão contidos no intervalo [0,1]. Nessa função, λ é um parâmetro de suavização da curva. Variando-se esse parâmetro, obtém-se funções sigmóides com diferentes inclinações. Quando $\lambda \rightarrow \infty$, a função se transforma em um degrau unitário.

Algumas vezes é desejável que a função se estenda ao intervalo [-1,1]. Nesse caso, faz-se uso da função tangente hiperbólica. Essa função também pertence à família das sigmóides, mas é bipolar, possibilitando que o neurônio assuma valores de saída negativos.

4.1.3
Topologia

As topologias das RNA podem ser divididas em duas classes: recorrentes e não recorrentes. Redes não recorrentes são aquelas que não possuem realimentação das saídas para as entrada e por essa razão diz-se que elas não possuem memória. Essas redes são estruturadas em camadas, podendo ser formadas por uma única camada ou por múltiplas camadas. A estrutura desse tipo de rede pode ser vista na Fig. 3.2, onde os neurônios são representados por círculos (nós) e as conexões por retas (arcos). Essas redes contêm um conjunto de neurônios de entrada, uma camada de saída e uma ou mais camadas escondidas. A entrada não é considerada uma camada da rede, pelo fato de distribuir os padrões para a camada seguinte [81]. A camada de saída contém neurônios que fornecem o resultado da rede. As camadas que não possuem ligações diretas tanto com a entrada quanto com a saída são denominadas escondidas. No caso de redes não recorrentes, não existe conexões ligando um neurônio de uma camada a outro de uma camada anterior o da mesma camada.

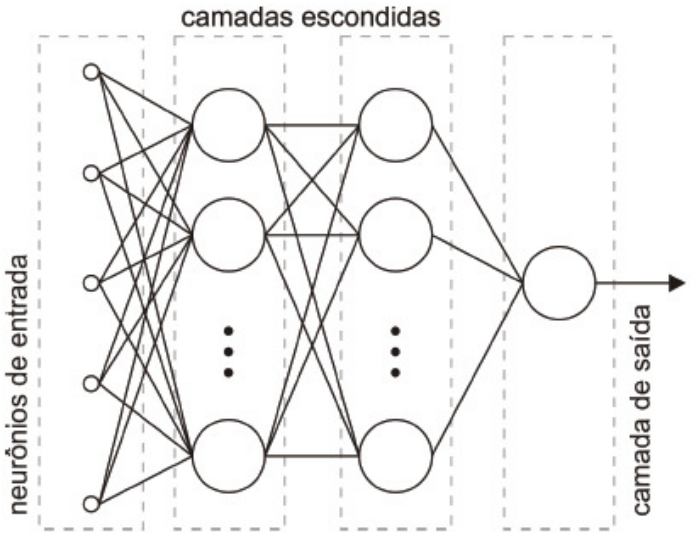

Figura 4.2: Estrutura de uma rede neural não recorrente.

As RNA recorrentes (Fig. 4.2) são redes em que as suas saídas realimentam as entradas, sendo determinadas pelas entradas atuais e pelas saídas anteriores. As redes recorrentes, quando organizadas em camadas, possuem interligações entre os neurônios da mesma camada e entre camadas não consecutivas, gerando interconexões bem mais complexas que aquelas das redes neurais não recorrentes.

As redes neurais recorrentes (Fig. 4.3) respondem a estímulos dinamicamente, isto é, após aplicar uma nova entrada, a saída é calculada e então realimentada para modificar a entrada. Para uma rede se tornar estável, este processo é repetido diversas vezes, produzindo pequenas mudanças nas saídas, até que estas tendam a ficar constantes. O fato de não se conseguir prever quais redes seriam estáveis foi um problema que preocupou os pesquisadores até o início da década de 80, quando Cohen e Grossberg provaram um teorema para definir quando as redes neurais recorrentes são estáveis [81]. Este teorema diz que, para as RNAs recorrentes alcançarem a estabilidade, é necessário que as funções de ativação sejam monotônicas, as conexões sejam simétricas e um neurônio não realimente a si próprio. Contribuições importantes foram dadas por John Hopfield [82], sendo que algumas configurações

passaram a ser chamadas de redes de Hopfield, em sua homenagem, e por Hinton e Sejnowski, que introduziram regras gerais porá o treinamento.

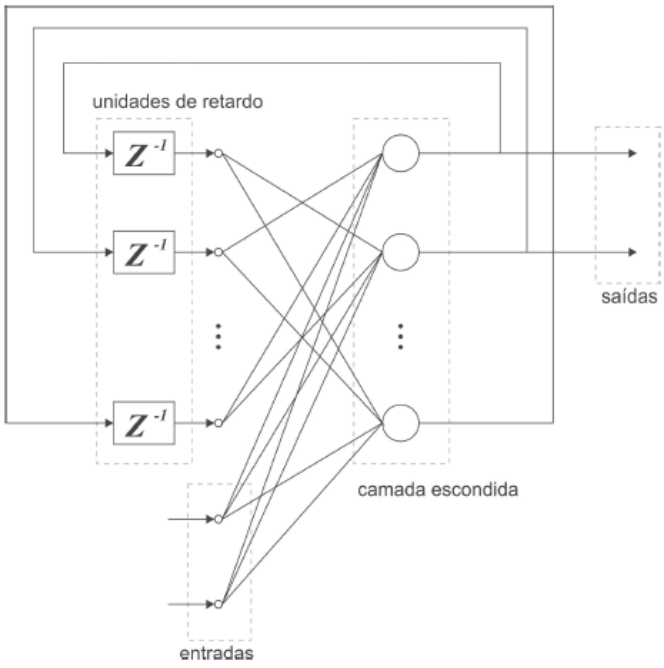

Figura 4.3: Exemplo de uma rede neural recorrente com duas entradas e duas saídas.

4.1.4
Tratamento de Dados

Como as ordens de grandeza dos parâmetros de cada entrada e saída podem ser distintas, é necessário empregar algum tipo de normalização antes de executar o treinamento da rede neural. Desta forma, o valor de uma variável não se torna mais significativo que o de outra quando as escalar forem diferentes. Uma variável que assume valores de 1 a 2500, por exemplo, não afetará o sistema mais do que outra com valores entre 0 e 0,5.

A normalização linear consiste em transformar os dados de modo que estejam em um determinado intervalo. Tal normalização é definida pela equação 4.2, onde *x* é o valor do dado real, *y* é o dado normalizado e os índices *max* e *min* representam os seus valores máximos e mínimos, respectivamente.

$$f(x) = y = (y_{max} - y_{min}) \frac{x - x_{min}}{(x_{max} - x_{min})} + y_{min} \quad (4.2)$$

Quando os dados se encontram concentrados em um subespaço do intervalo admissível de uma variável, utiliza-se um método para dispersá-los. Um dos métodos mais simples de se implementar é a normalização linear por partes. Tal normalização determina subintervalos a partir de um valor intermediário. Desse modo, a primeira parte dos dados é normalizada entre um dado intervalo [y_{min};y_{int}] e a segunda parte é normalizada em outro intervalo [y_{int}; y_{max}], conforme a Eq. 4.3.

$$f(x) = y = \begin{cases} (y_{int} - y_{min}) \frac{x - x_{min}}{(x_{int} - x_{min})} + y_{min} & se\ x \leq x_{int} \\ (y_{max} - y_{int}) \frac{x - x_{int}}{(x_{max} - x_{int})} + y_{int} & se\ x > x_{int} \end{cases} \quad (4.3)$$

4.1.5
Treinamento da RNA

O objetivo do treinamento de uma RNA é fazer com que a aplicação de um conjunto de entradas produza um conjunto de saídas desejado. Cada conjunto de entradas e saídas de dados desejadas, ou alvo, é chamado padrão de treinamento. O treinamento é realizado pela aplicação seqüencial dos vetores de entrada e, em alguns casos, também os de saída, enquanto os pesos da rede são ajustados de acordo com uma estratégia de treinamento pré-determinada. Durante o treinamento, os pesos da rede gradualmente convergem para determinados valores, de modo que a aplicação dos vetores de entrada produza as saídas necessárias. Os procedimentos de treinamento podem ser divididos em duas categorias: supervisionado e não supervisionado. Ambos usufruem de um conjunto de treinamento, contudo, o primeiro necessita de valores-alvo para a saída, enquanto o segundo não precisa.

O conjunto de treinamento modifica os pesos da rede de forma a produzir saídas que sejam consistentes, isto é, tanto a apresentação de um dos vetores de treinamento, como de um vetor que é suficientemente similar, irá produzir o mesmo padrão de saídas.

O treinamento supervisionado necessita de um par de vetores, composto das entradas e do vetor-alvo que se deseja obter como as respectivas saídas. Juntos, esses vetores são chamados de par de treinamentos ou vetor de treinamento. O treinamento da rede geralmente utiliza vários desses vetores.

Existe uma grande variedade de algoritmos de treinamento, tanto para o modelo supervisionado, quando para o não supervisionado. Entre estes, o mais difundido é o algoritmo de retropropagação [83] (*backpropagation*). A retropropagação contém duas etapas: (i) a primeira consiste na aplicação da regra-da-cadeia para a obtenção da derivada parcial da função de erro da rede em relação a cada um de seus pesos; (ii) a segunda é o algoritmo de atualização dos pesos, que consiste no método de descida por gradiente [84].

Em suma, para a realização do treinamento supervisionado, é preciso que haja um conjunto de padrões de entradas e suas respectivas saídas, além de uma função de erro (função de custo) para medir a diferença entre as saídas da rede e os valores desejados. Tal treinamento é iniciado com a apresentação e propagação de um padrão através da rede para obter as saídas. Uma vez calculadas, as saídas são comparadas com os respectivos valores-alvo e o erro é então calculado a partir de alguma métrica. O erro então é utilizado para atualizar os pesos de acordo com um algoritmo de minimização, conforme ilustrado na Fig. 4.4. Este processo de treinamento é repetido até que o erro, para todos os vetores de treinamento, tenha alcançado o nível especificado ou até que um número máximo de iterações seja atingido. Cada iteração desse processo é conhecida como época.

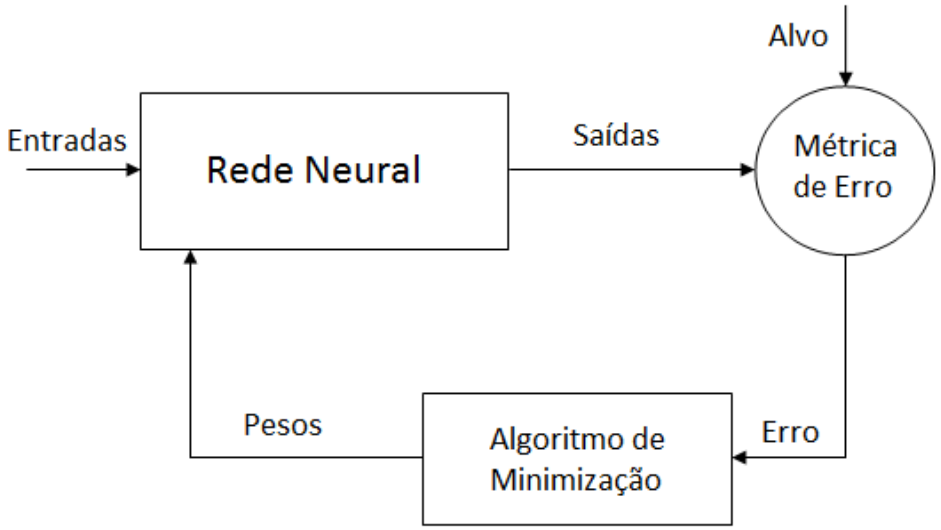

Figura 4.4: Diagrama esquemático do treinamento supervisionado de uma rede neural. (Adaptado de [85]).

4.2
Algoritmos Genéticos

Algoritmos Genéticos são, essencialmente, métodos estocásticos de busca e otimização inspirados nos conceitos da teoria de seleção natural das espécies, propostas por Darwin [86-90]. Os sistemas desenvolvidos a partir deste princípio são utilizados para procurar soluções de problemas complexos ou com espaço de soluções (espaço de busca) muito grande, os quais tornam os tornam problemas de difícil modelagem e solução, conforme mencionado no início do capítulo.

Estes algoritmos baseiam-se nos processos genéticos de organismos biológicos para procurar soluções ótimas. Para tanto, procede-se da seguinte maneira: cada possível solução de um problema é codificada em uma estrutura chamada de “cromossomo”, a qual é composta por uma cadeia de bits ou caracteres. Tais cromossomos representam indivíduos de uma população, que são evoluídos ao longo de várias gerações, de forma similar aos seres vivos, de acordo com os princípios de seleção natural e sobrevivência dos mais aptos, descrito pela primeira vez por Charles Darwin, em seu livro *A Origem das*

Espécies. Emulando estes processos, os Algoritmos Genéticos são capazes de “evoluir” soluções de problemas do mundo real.

Os cromossomos são então submetidos a um processo evolucionário que envolve quatro etapas fundamentais: avaliação, seleção, cruzamento e mutação. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos. Os AGs utilizam uma analogia direta deste fenômeno de evolução presente na natureza, onde cada indivíduo representa uma possível solução para um problema dado. A cada indivíduo atribui-se um valor de adaptação: sua aptidão, que indica a qualidade da solução representada por este indivíduo é boa com relação às outras soluções da população, cujo análogo em programação matemática é o resultado da função objetivo. Desta maneira, o termo População refere-se ao conjunto de todas as soluções com as quais o sistema trabalha. Aos indivíduos mais adaptados é dada a oportunidade de se reproduzir mediante cruzamentos com outros indivíduos da população, produzindo descendentes com características de ambas as partes. A mutação também tem papel significativo nesse processo, ao introduzir na população novos indivíduos, gerados de maneira aleatória.

O processo evolutivo começa com a criação aleatória dos indivíduos que formarão a população inicial. A partir de um processo de seleção baseado na aptidão de cada indivíduo, são escolhidos os indivíduos que irão compor a fase de reprodução, que cria novas soluções a partir de um conjunto de operadores genéticos. Deste modo, a aptidão do indivíduo determina o seu grau de sobrevivência e, assim, aumenta ou diminui as chances de que seu cromossomo faça parte das gerações seguintes. O processo básico de um Algoritmo Genético é resumido na Fig. 4.5 [91].

```
Início
t ← 1
P(t) ← IniciaPopulação(P(t))
Avalia (P(t))
enquanto (CondiçãoDeParada = falso) faça
    t ← t + 1
    P(t) ← SelecionaPopulação(P(t - 1))
    AplicaOperadores(P(t))
fim enquanto
```

Fig. 4.5: Estrutura básico de um Algoritmo Genético.

Há diversas formas de se determinar o final da evolução: o número de gerações, o número de indivíduos criados, ou ainda condicionar o algoritmo à obtenção de uma solução satisfatória, isto é quando se atingir um ponto ótimo. Outras condições de parada incluem o tempo de processamento e o grau de similaridade entre os elementos numa população (convergência). O processo evolucionário é então influenciado principalmente pelos seguintes fatores: (i) A representação ou codificação do problema; (ii) A função de avaliação; (iii) A inicialização da população; (iv) Os operadores de seleção, cruzamento e mutação.

4.2.1
Representação

A solução de um problema pode ser representada por um conjunto de parâmetros (genes), unidos para formar uma cadeia de valores (cromossomos); a este processo chama-se codificação. As soluções (cromossomos) são codificados através de uma seqüência formada por caracteres de um sistema alfabético. Originalmente, utilizou-se o alfabeto binário (0,1). Porém, novos modelos de AGs codificam as soluções com outros alfabetos como, por exemplo, números reais [92]. Assim, a representação é um aspecto fundamental na modelagem de um AG para a solução de um problema. Ela define a estrutura do cromossomo, com os

respectivos genes que o compõem, de maneira que este seja capaz de descrever todo o espaço de busca do problema. A decodificação do cromossomo consiste basicamente na construção de uma solução real do problema a partir de sua representação. Isto é, o processo de decodificação constrói a solução para que esta seja avaliada pelo problema.

4.2.2
Avaliação

A avaliação é a ligação entre o AG e o problema a ser solucionado. Ela é feita através de uma função que melhor representa o problema e tem por objetivo oferecer uma medida de aptidão de cada indivíduo na população corrente, que irá dirigir o processo de busca. Dado um cromossomo, a função de avaliação consiste me associar um valor numérico, o qual supõe-se proporcional à “utilidade” ou “habilidade”do indivíduo representado em solucionar o problema em questão. Muitas vezes a avaliação é realizada por um simulador ou código externo ao AG.

4.2.3
Operadores

Operadores genéticos são algoritmos que modificam os genes, possibilitando evoluir a solução. Existem duas classes de operadores: cruzamento e mutação. A geração de novos cromossomos pode ser realizada tanto pelo cruzamento quanto pela mutação, ou ainda pelos dois simultaneamente. Os operadores de mutação são responsáveis pela divergência das soluções, explorando o espaço de busca, enquanto os operadores de cruzamento fazem com que as soluções convirjam, tirando proveito de determinado subespaço de busca. Esses operadores variam de acordo com a representação utilizada.

4.3
Algoritmo de Colônia de Formigas

O Algoritmo de Colônia de Formigas, também conhecido como Otimização por Colônia de Formigas (ACO), é um tipo de meta-heurística inspirada no comportamento de uma colônia de formigas na sua busca por alimento (Fig. 4.6). A observação do comportamento coletivo de formigas reais, levou à proposição de um algoritmo para solução de problemas de otimização combinatória[1] do tipo NP-difícil [93-95].

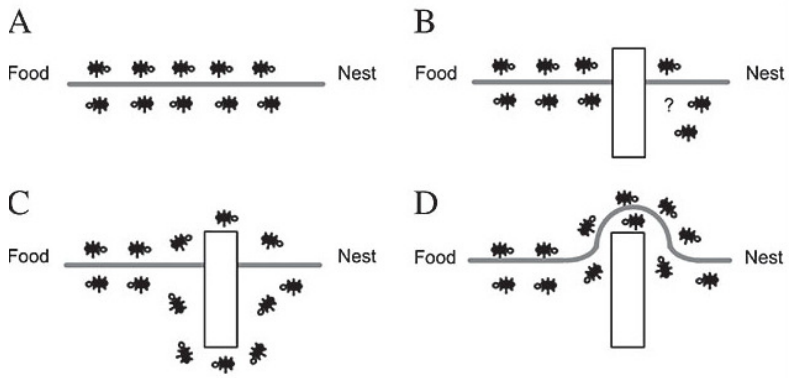

Figura. 4.6: Exemplo do comportamento de formigas reais na busca por alimento.

Algoritmos de otimização por colônia de formigas são considerados como parte de uma área conhecida como Inteligência de Enxame (IE), a qual é um campo de pesquisa que estuda métodos inspirados na observação do comportamento de enxames. Algoritmos desse tipo são compostos de indivíduos simples, capazes de cooperar uns com os outros e formar um todo auto-organização, isto é, sem qualquer tipo de controle central sobre os membros do enxame. Uma visão global dos princípios de auto-organização explorado por esses algoritmos pode ser encontrada em [96]. Em relação à Inteligência de Enxame, existe uma gama variada de algoritmos propostos na literatura [97].

[1] A otimização combinatória é um ramo da ciência da computação e da matemática aplicada que estuda problemas de otimização em conjuntos finitos. Em geral, é fácil listar os seus elementos e também testar se um dado elemento pertence a esse domínio. Ainda assim, a idéia ingênua de testar todos os elementos deste domínio na busca pelo melhor mostra-se inviável na prática, mesmo para instância de tamanho moderado.

4.3.1
Experimentos com Formigas Reais

Um dos experimentos clássicos realizados com formigas em laboratório é o chamado experimento com as duas pontes. Nesse caso, existem dois caminhos separando o formigueiro da fonte de alimento. No princípio não há feromônio em nenhuma das pontes. As formigas iniciam a sua busca explorando as redondezas do formigueiro e eventualmente cruzam uma das pontes e encontram a fonte de alimento. Durante o caminho de ida e volta entre o formigueiro e a fonte de alimento, as formigas depositam um marcador químico conhecido como feromônio na ponte utilizada. Inicialmente, cada formiga escolhe aleatoriamente uma das pontes. Entretanto, devido a flutuações estocásticas, após algum tempo, haverá mais feromônio depositado em uma das pontes do que em outra. Como as formigas tendem a escolher de forma probabilística a trilha com mais feromônio, a ponte que atender a essas condições irá atrair mais formigas. Um maior número de formigas provocará um aumento na concentração de feromônio, até que toda a colônia convirja em direção a uma única ponte.

Tal comportamento de uma colônia de formigas é classificado como autocatalítico, ou seja, é um processo que reforça a si próprio de forma a causar uma rápida convergência [97]. Esse mecanismo é explorado pelas formigas de forma a encontrar o menor caminho entre a fonte de alimento e o formigueiro. Isso foi demonstrado em outro experimento conduzido por Goss et al [98], no qual as duas pontes possuíam comprimentos diferentes: uma era significativamente maior do que a outra. Nesse caso, as flutuações estocásticas na escolha inicial da ponte ficaram muito reduzidas devido à interferência de um segundo mecanismo: as formigas que escolhiam por acaso a ponte mais curta eram as primeira a retornar ao formigueiro e ao fazê-lo, a probabilidade de escolher a ponte mais curta era maior, devido à maior concentração de feromônio. Portanto, graças a esse mecanismo, a trilha de formiga convergiu rapidamente para a segunda ponte. Esse método de comunicação indireta utilizado pelas formigas é denominado estigmergia.

4.3.2
Formigas Artificiais

Com base nos experimentos descritos acima, um modelo capaz de explicar o comportamento observado no experimento de pontes binárias foi desenvolvido por Goss et al [98]. Assumindo-se que depois de um tempo t desde o início do experimento, m_1 formigas escolheram a primeira ponte e m_2 formigas, a segunda, a probabilidade p_1 da (m+1)-ésima formigas escolher a primeira ponte é dada pela Eq. (4.4).

$$p_1(m+1) = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \tag{4.4}$$

onde os parâmetros k e h são necessários para ajustar o modelo aos dados experimentais. A probabilidade que a mesma (m+1)-ésima formigas escolha a segunda ponte é $p_2(m+1) = 1 - p_1(m+1)$. Simulações realizadas com o método Monte Carlo, executadas para testar a correspondência entre o modelo proposto e os dados reais demonstraram um bom ajuste para K ≈ 20 e h ≈ 2 [99].

Este modelo básico, que explicar o comportamento de formigas reais, pode ser usado como inspiração para o desenvolvimento de formigas artificiais para resolver problemas de otimização definidos de maneira similar. De forma análoga aos seus homônimos reais, formigas artificiais são capazes de simular a deposição de feromônio através da modificação apropriada das variáveis relacionadas a esse mecanismo e associadas às diferentes estados do sistema, os quais são percorridos pelas formigas durante a construção das soluções. Cada estado do sistema corresponde a um conjunto de valores específicos das variáveis que definem o problema.

Outro aspecto importante relativo ao comportamento de formigas reais e que pode ser aproveitado pelas formigar artificiais é o acoplamento entre o mecanismo de autocatálise e a avaliação implícita das soluções. Este último mecanismo significa que caminhos mais curtos

(os quais, no caso de formigas artificiais, correspondem a soluções com menor custo) são completados antes de caminhos mais longos, fazendo com que a concentração de feromônio na trilha aumenta mais rapidamente. O acoplamento entre esse dois mecanismos pode ser bem eficiente: quanto menor o caminho, mais feromônio será depositado e um maior número de formigas optará pelo caminho menor.

A estigmergia, juntamente com a avaliação implícita de soluções e com o comportamento autocatalítico, dá origem ao ACO. Os conceitos básicos do ACO são muito similares aos da sua inspiração biológica. Portanto, existem muitas semelhanças entre formigas reais e artificiais. Tanto as colônias de formigas reais quanto artificiais são compostas por uma população de indivíduos que trabalham juntos para atingir um determinado objetivo. Uma colônia é uma população de agente simples, independentes e assíncronos, que cooperam de forma a encontrar a melhor solução para o problema em questão. No caso de formigas reais, o problema é encontrar alimento, enquanto que no caso de formigas artificiais, o propósito é encontrar boas soluções para um dado problema de otimização. Uma única formiga, seja real ou artificial, é capaz de encontrar uma solução para tais problemas, mas somente a cooperação entre muitos indivíduos através da estigmergia possibilita encontrar boas soluções.

No caso de formigas reais, elas depositam e reagem a uma substância química chamada feromônio, a qual é depositada no solo durante a caminhada. Formigas artificiais vivem em um ambiente virtual e, conseqüentemente, são capazes de modificar apenas valores numéricos associados aos diferentes estados do problema e chamados analogamente de feromônios artificiais. Uma seqüência de valores de feromônio associados com estados específicos do problema é chamada de trilha de feromônio artificial. No ACO, essas trilhas de feromônio artificial são a única forma de comunicação entre as formigas. Um mecanismo similar à evaporação do feromônio físico que ocorre nas colônias de formigas reais permite que as formigas virtuais esqueçam o histórico passado de buscas e mantenham seu foco em novas direções.

Assim como as formigas reais, as formigas artificiais criam suas soluções seqüencialmente ao se moverem de um estado do problema para outro. As primeiras simplesmente caminham, escolhendo a sua direção com base na concentração local de feromônio e em uma política de decisão estocástica. As segundas também criam soluções passo a passo, movendo-se através dos estados do problema e decidindo o caminho a seguir estocasticamente a cada passo.

Existem, entretanto, algumas diferenças importantes entre as formigas reais e as artificiais:

- Formigas artificiais vivem em um ambiente discreto – elas se movem seqüencialmente através de um conjunto finito de estados do problema;
- A atualização do feromônio – a deposição e evaporação – não é executada exatamente da mesma maneira pelas formigas artificiais e reais. Algumas vezes essa atualização não é executada por todas as formigas artificiais, e geralmente apenas após a solução ter sido construída;
- Algumas implementações de formigas artificiais empregam mecanismos que não existem no caso de formigas reais. Exemplos incluem mecanismos de busca local, *look-ahead* [100], *backtracking* [101].

4.3.3
Problemas de Representação Real

A otimização combinatória – como o nome sugere -, trata da procura de combinações ou permutações ótimas das componentes disponíveis em um determinado problema. Assim, é necessário que o problema esteja dividido em um conjunto finito de elementos e o algoritmo de otimização combinatória então tentará encontrar a sua combinação ou permutação ótima. Muitos problemas do mundo real podem ser representados diretamente representados dessa forma. Existe, contudo, uma importante classe de problemas para o qual esse não é o caso: a

classe de otimização de problemas que requer a utilização de variáveis contínuas. Tais problemas só podem ser tratados pode algoritmos de otimização combinatória se a faixa de valores contínuos for convertida em conjuntos finitos. Nesses casos, algoritmos que possam naturalmente lidar com variáveis contínuas geralmente tem um desempenho superior.

Um conceito central na forma como o ACO trabalha é a construção incremental de soluções baseada na escolha probabilística dos componentes da mesma. No caso de problemas de otimização combinatorial, o conjunto de componentes da solução disponíveis é definido pela formulação do problema. A cada passo da montagem da solução, as formigas escolhem probabilisticamente cada um dos componentes c_i de um conjunto $N(s^p)$ de elementos disponíveis, de acordo com a Eq. 4.45.

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha}.\eta(c_{ij})^{\beta}}{\sum_{c_{ij}\in\mathbb{N}(s^p)} \tau_{ij}^{\alpha}.\eta(c_{ij})^{\beta}}, \qquad \forall\, c_{ij} \in \mathbb{N}(s^p) \tag{4.5}$$

onde τ_{ij}^{α} é o valor de feromônio associado com a componente c_{ij} e $\eta(.)$ é uma função que associa, a cada passo da construção da solução, um valor heurístico para cada componente $c_{ij} \in \mathbb{N}(s^p)$ de um possível solução. Os valores dados por está função são comumente chamados de informação heurística. Ademais, α e β são parâmetros positivos, cujos valores determinam a importância relativa do feromônio contra a informação heurística. As probabilidades associadas com os elementos do conjunto $N(s^p)$ compõem uma distribuição discreta de probabilidade (Fig. 4.7a) a qual uma formiga examina de forma a escolher o componente que será adicionado à solução corrente s^p.

A idéia fundamental envolvendo o ACO com representação real é a substituição da distribuição discreta de probabilidade por uma contínua, isto é, uma função densidade de probabilidade (FDP) (Fig. 4.7a). No ACO-$\mathbb{R}$, ao invés de escolher uma componente $c_{ij} \in \mathbb{N}(s^p)$, de acordo com a Eq. 4.4, uma formiga faz uma amostragem na FDP.

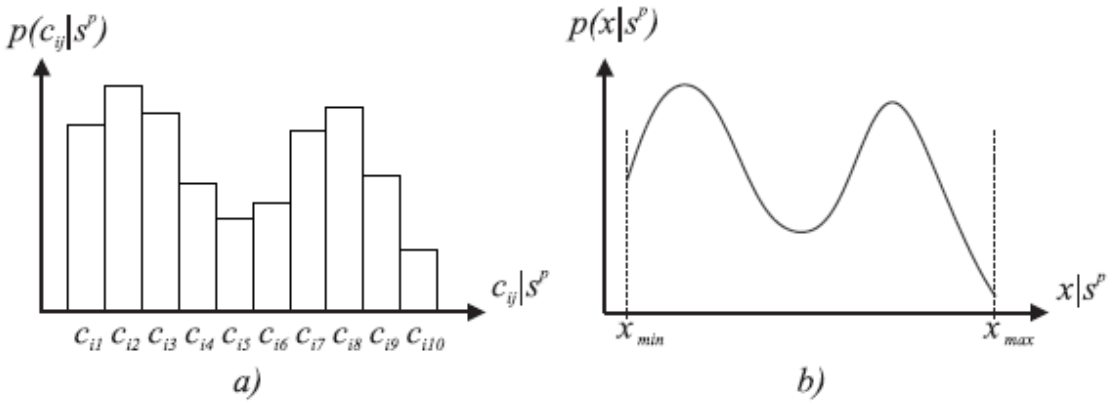

Figura. 4.7: (a) Distribuição discreta de probabilidade $p_d(c_{ij}|s^p)$ de um conjunto finito $\{c_{i1}, \ldots, c_{i10}\} \in N(s^p)$ de componentes disponíveis. (b) Função densidade de probabilidade contínua $p_d(x|s^p)$ em um possível intervalo $x \in [x_{min}, x_{max}]$.

A meta-heurística do ACO encontra soluções aproximadas em um problema de otimização iterando dois passos:

1. Soluções candidatas são construídas de uma forma probabilística aplicando-se uma distribuição de probabilidades sobre o espaço de busca;

2. As soluções candidatas são utilizadas para modificar a distribuição de probabilidades de uma forma que irá influenciar amostragem futura na direção de soluções de alta qualidade.

Algoritmos ACO para problemas de otimização combinatória empregam o modelo de feromônio para construir a solução de forma probabilística. Um modelo de feromônio consiste em um conjunto dos chamados parâmetros da trilha de feromônio. O valor numérico desses parâmetros reflete a experiência de busca do algoritmo e são usados para influenciar a criação da solução ao longo do tempo na direção de regiões do espaço de busca que contêm soluções de alta qualidade.

Ainda nesse contexto, os valores de feromônio estão associados a um conjunto finito de valores relacionados às decisões que as formigas devem tomar. Isso permite fazer uma representação na forma de uma tabela de feromônio. Isso não é possível no caso de variáveis contínuas,

uma vez que o número possível de valor não é finito. Portanto, o ACO-$\mathbb{R}$ emprega uma matriz de soluções como forma de descrever a distribuição de feromônio sobre o espaço de busca. A matriz de soluções contém uma quantidade completa de soluções para o problema. Enquanto o modelo de feromônio na otimização combinatorial pode ser visto como uma memória implícita do histórico de busca, uma matriz de soluções é uma memória explícita[2].

O fluxo do ACO-$\mathbb{R}$ funciona da seguinte maneira: no primeiro passo, a matriz de soluções é inicializada. Então, a cada iteração, um número de soluções é probabilisticamente construído pelas formigas. Essas soluções podem ser melhoradas por qualquer mecanismo de aperfeiçoamento como, por exemplo, técnicas de busca local ou gradiente.

4.3.4
Estrutura, Inicialização e Atualização da Matriz de Soluções

O ACO-$\mathbb{R}$ mantém um histórico dos seus processos armazenando as soluções em uma matriz de soluções T de dimensão $|T| = k$. Dado um problema de otimização com variáveis contínuas, n-dimensões e *k* soluções, o ACO-$\mathbb{R}$ armazena em T os valores das soluções geradas pelas n variáveis e o valor da função objetivo. O valor da i-ésima variável relativa à j-ésima solução é denotada por s_j^i. A estrutura da matriz de soluções pode ser vista na Fig. 4.5.

[2] A memória implícita é um tipo de memória na qual as experiências anteriores auxiliam no desempenho de uma tarefa sem uma percepção consciente das mesmas. A memória explícita é a recordação consciente e intencional de experiências e informações anteriores.

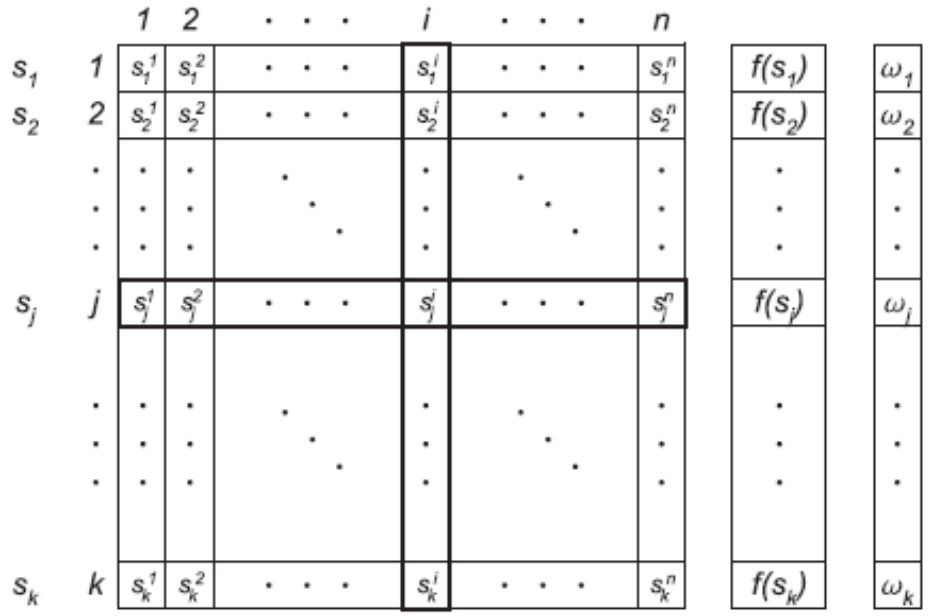

Figura. 4.5: Estrutura da matriz de soluções. As soluções na matriz são sorteadas de acordo com a sua qualidade, isto é, o valor da função objetivo f(s), portanto, a posição de uma solução na matriz sempre corresponde à sua classificação.

Antes da execução do algoritmo, a matriz é inicializada com *k* soluções aleatórias. A cada iteração do algoritmo, primeiramente, um conjunto de *m* é gerado pelas formigas e então acrescentado aquelas presentes em *T*. A partir desse conjunto de k + m soluções, as m piores são removidas. As k soluções restantes são sorteadas de acordo com a sua qualidade - o valor da função de avaliação – e armazenadas em uma nova matriz T. Dessa maneira, o processo de busca é influenciado na direção da região que contém as melhores soluções encontradas. As soluções contidas na matriz são sempre sorteadas com base na sua qualidade, assim, a melhor solução permanece no topo.

4.3.5
Construção de Soluções Probabilísticas

A construção de novas soluções pelas formigas é realizada de uma maneira incremental, variável por variável. Primeiro, uma formiga escolhe probabilisticamente uma das soluções na matriz. A probabilidade de escolher uma solução j é dada pela Eq. 4.6**.**

$$p_j = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r} \tag{4.6}$$

onde ω_j é um peso associado à solução *j*. Esse peso pode ser calculado utilizando-se várias fórmulas, dependendo do problema abordado. Uma das funções mais empregadas para esse fim é a função Gaussiana $g(\mu, \sigma) = \ g(1, qk)$ [102], descrita pela Eq. 4.7.

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(j-1)^2}{2q^2k^2}} \tag{4.7}$$

onde *q* é um parâmetro do algoritmo e *k* é o tamanho da matriz. A média da função Gaussiana é definida como 1, para que assim a melhor solução tenha o maior peso.

A escolha da função Gaussiana normalmente é atribuída às suas características de flexibilidade e não-linearidade Devido à sua não-linearidade, ela permite um controle flexível sobre os pesos. É possível atribuir uma elevada probabilidade para algumas poucas soluções principais, enquanto reduz-se significativamente a probabilidade das restantes.

As formigas tratam cada variável $i = 1, \dots, n$ do problema separadamente. Elas lêem o valor s_j^i da variável i da j-ésima solução escolhida e amostram as suas vizinhanças. Isto é realizado através da função densidade de probabilidade (FDP). Novamente, assim como no caso da escolha dos pesos, uma variedade de funções pode ser utilizada. A FDP $P(x)$ deve, entretanto, satisfazer a condição encerrada pela Eq. 4.8.

$$\int_{-\infty}^{+\infty} P(x) = 1 \tag{4.8}$$

Uma possível escolha para a FDP, segundo consta na literatura [102], é apresentada pela Eq. 4.9.

$$P(x) = g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (4.9)$$

Essa função possui dois parâmetros que devem ser definidos: μ e σ. Considerando-se a variável *i* da solução *j*, atribui-se a μ o valor da variável s_j^i analisada, $\mu \leftarrow s_j^i$. Em seguida, σ é determinado de acordo com a Eq. 4.10.

$$\sigma \leftarrow \xi \sum_{r=1}^{k} \frac{\left|s_r^i - s_j^i\right|}{k-1} \quad (4.10)$$

a qual representa a distância média entre a i-ésima variável da solução s_j e as i-ésimas variáveis de outras soluções na matriz, multiplicada por um parâmetro ξ[3]

[3] O parâmetro ξ tem um efeito similar ao da taxa de evaporação de feromônio no ACO clássico. Quanto maior o valor de ξ, menor a velocidade de convergência do algoritmo. Enquanto no ACO clássico a taxa de evaporação de feromônio influencia a memória de longa duração – isto é, as piores soluções são esquecidas rapidamente - o ξ no ACO-$\mathbb{R}$ influencia a forma como essa memória é utilizada – isto é, as novas soluções são consideradas como próximas às boas soluções conhecidas.