Algoritmos sociais de busca

Emílio Bergamim Júnior

Instituto de Geociências e Ciências Exatas - UNESP

2024

Conteúdo da aula

- Computação inspirada pela natureza
- Otimização por enxame de partículas
- Otimização por colônia de formigas

Computação inspirada pela natureza

- Na última aula, discutimos algoritmos inspirados tanto pela física (recozimento simulado) como pela biologia evolutiva (algoritmo genético)
- Estes algoritmos refletem um tipo de inteligência que não é essencialmente humana
  - No recozimento simulado, utiliza-se uma propriedade observada em processos metalúrgicos na qual o resfriamento gradual de um material leva-o a uma configuração desejada
  - Já o algoritmo genético faz uso da ideia de que a combinação e mutação de indivíduos produz novos indivíduos melhores adaptados a um ambiente
- É importante ressaltar que estas são apenas metáforas de forma a justificar o funcionamento de tais algoritmos, já que não é conhecida uma justificativa matemática ou algoritmicamene robusta para o funcionamento dos mesmos

Princípios variacionais

- Na descrição de sistemas naturais, muitas vezes se emprega a ideia de que o comportamento observado se dá através da otimização de uma função.
- Mesmo não sabendo a expressão exata destas funções, você já pode ter ouvido algumas das seguintes expressões
  - Princípio de **mínima ação**
  - Sobrevivência do **mais apto**
  - Princípio de **máxima entropia**
- Isso é um recurso útil tanto de modelagem como do ponto de vista de uma explicação: justificar que os fenômenos observados são dessa forma pois são **ótimos** ajuda a retirar a impressão de que os mesmos sejam fruto de mero acaso.

Busca social

- Hoje discutiremos algoritmos inspirados por comportamentos sociais
- Estes baseiam-se na observação de que determinadas tarefas são impossíveis para um único indivíduo, mas plausíveis para um grupo destes
- Além disso, tal comportamento é **emergente** da combinação de características individuais, não sendo oriundo de uma peça central que dirige todo o processo.
- Um exemplo desse tipo de comportamento é aquele exibido por um bando de aves voando de forma sincronizada, que forma um padrão macroscópico que não pode ser compreendido em termos dos indivíduos que constituem o grupo
- Uma simulação desse comportamento pode ser vista na biblioteca do NetLogo

Enxames de partículas

- Como discutimos na última aula, desejamos minimizar uma função real $f(\boldsymbol{x})$ cujo domínio é um espaço $d-$dimensional.
- Assim como nos algoritmos de busca por feixes, serão armazenadas n soluções parciais do problema que serão referidas como as **posições** das partículas e denotadas por $\{\boldsymbol{x}_i\}_{i=1}^{n}$
- Cada partícula armazena também sua melhor posição. Isto é, a posição $\boldsymbol{x}_i^*$ que possui o menor valor observado de f.
- A cada iteração do algoritmo, a posição de uma partícula é atualizada de acordo com sua velocidade $\boldsymbol{v}_i$ através da fórmula

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1). \tag{1}$$

- Após essa atualização, se cabível, atualiza-se também $\boldsymbol{x}_i^*$

A expressão da velocidade

Sendo $\boldsymbol{x}_{opt}(t)$ a melhor posição dentre todas as posições do enxame, a cada iteração a velocidade de uma partícula é atualizada através da expressão

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + \boldsymbol{c}_1 \odot [\boldsymbol{x}_i^*(t) - \boldsymbol{x}_i(t)] + \boldsymbol{c}_2 \odot [\boldsymbol{x}_{opt}(t) - \boldsymbol{x}_i(t)], \quad (2)$$

onde $\boldsymbol{a} \odot \boldsymbol{b}$ denota o vetor obtido do produto elemento por elemento de $\boldsymbol{a}$ e $\boldsymbol{b}$.

- O primeiro termo da expressão denota a tendência a permanecer na direção da iteração anterior
- O segundo termo é o aprendizado individual da partícula, encorajando-a a retornar à sua melhor posição
- Já o terceiro reflete o aprendizado coletivo do enxame, influenciando a particular a seguir para a melhor posição do enxame

Exemplo: função de Rastrigin bidimensional

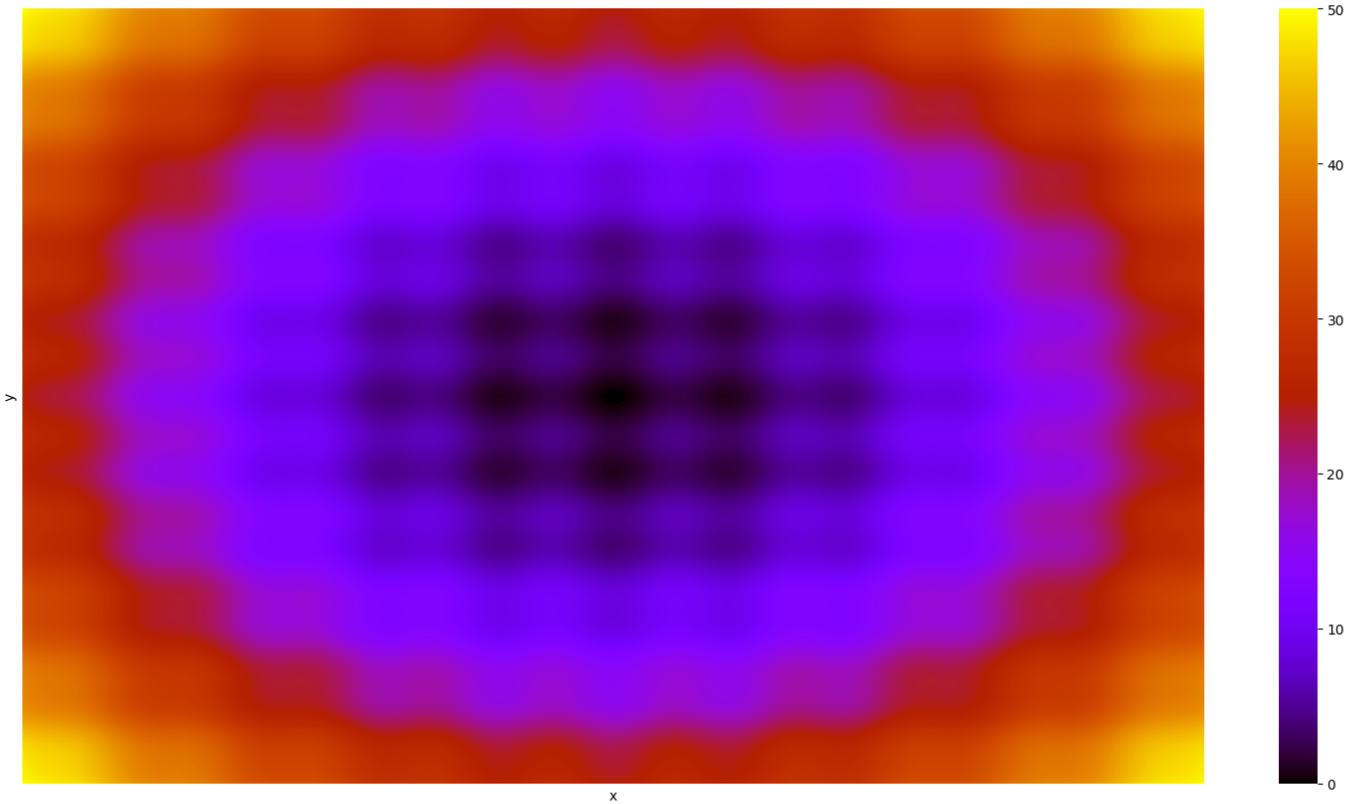

Figura: Função de Rastrigin em duas dimensões.

Resultados

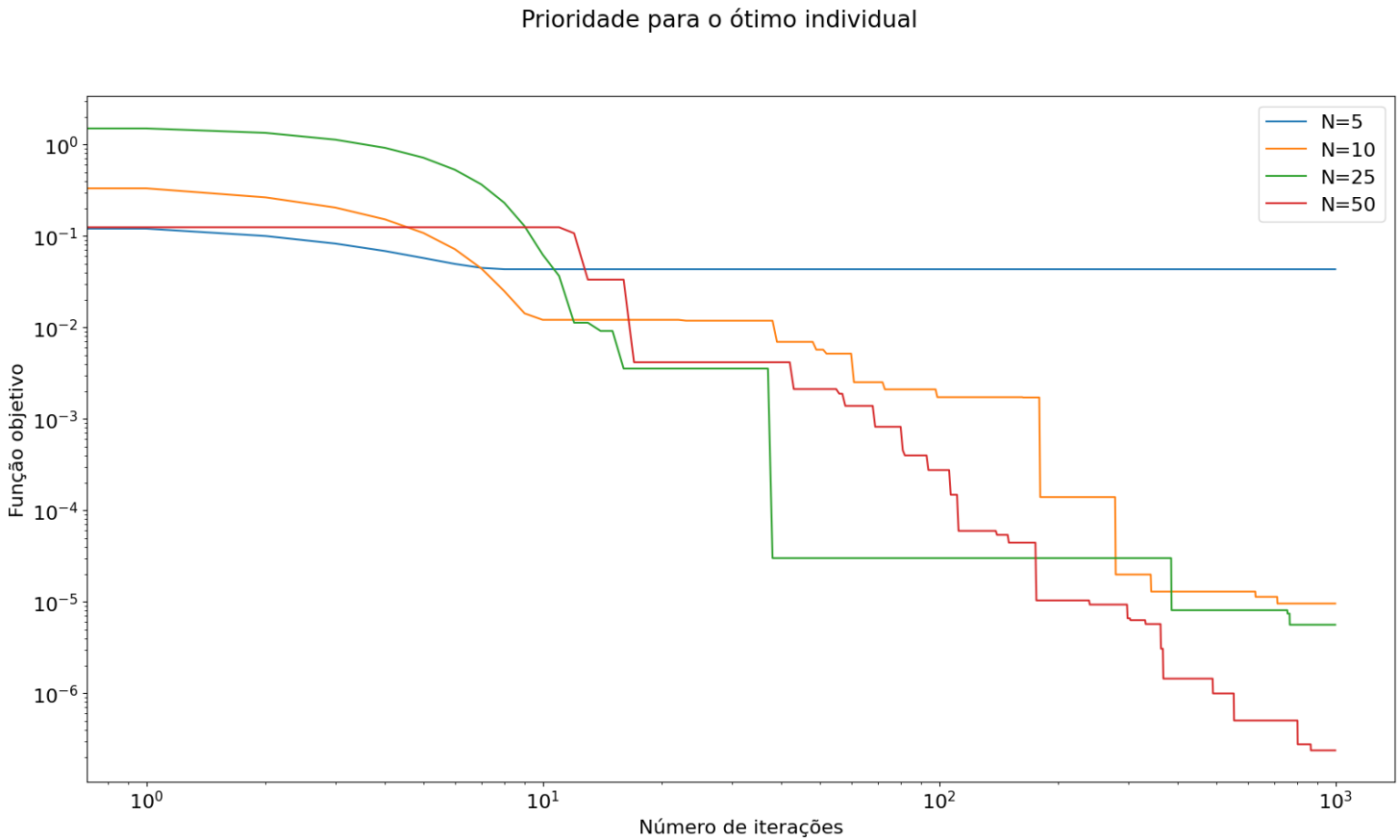

Figura: Resultados para diferentes tamanhos do enxame com coeficientes privilegiando os ótimos individuais das partículas durante atualização da velocidade.

Resultados

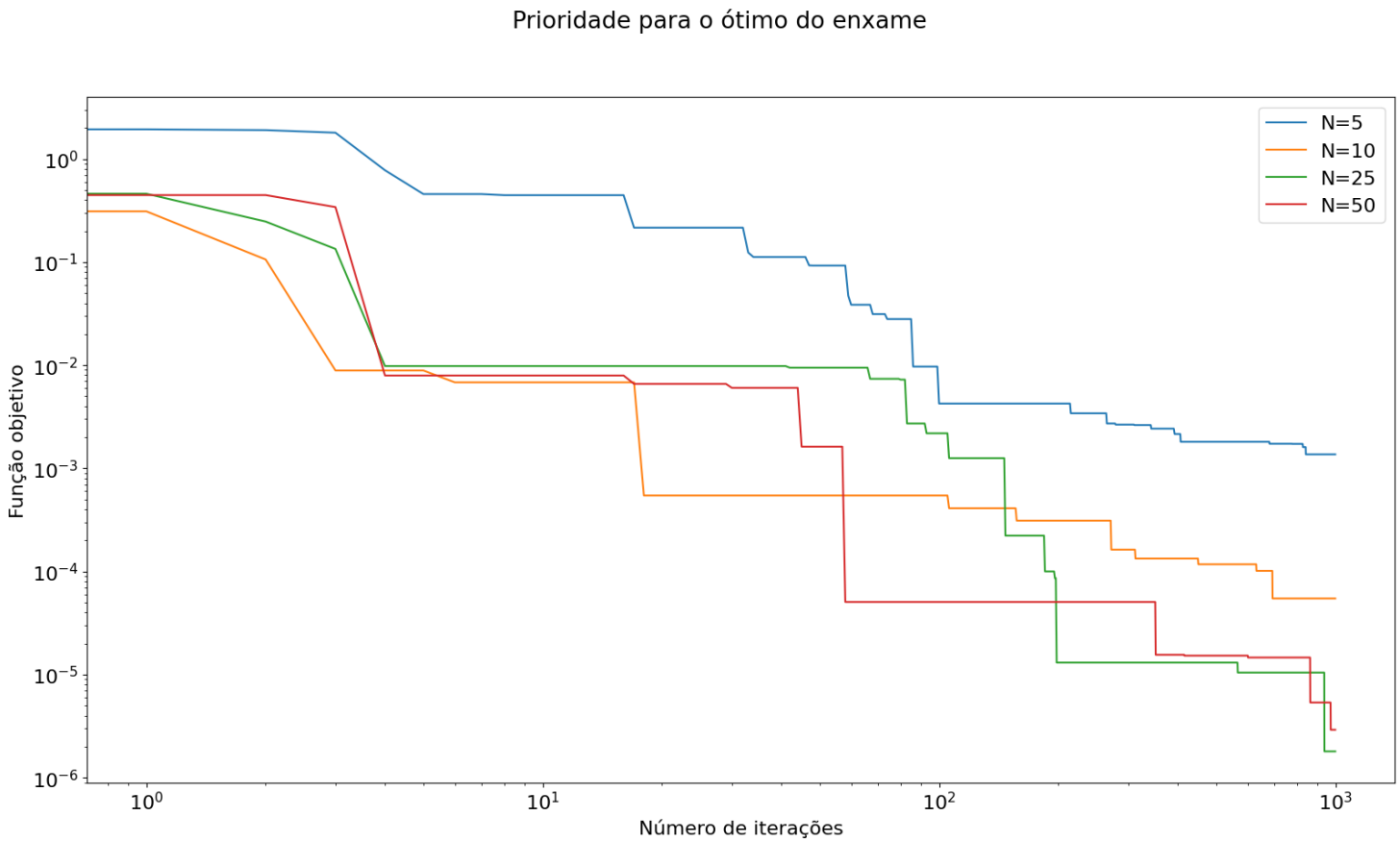

Figura: Resultados para diferentes tamanhos do enxame com coeficientes privilegiando o ótimo do enxame durante atualização da velocidade.

Controle de velocidade I

Note que partículas que divergem significativamente das melhores posições (tanto individuais como coletivas) produzirão velocidades com altas magnitudes, acarretando em deslocamentos grandes de uma iteração para outra do algoritmo.

- Esse comportamento garante uma maior exploração do espaço de busca como um todo
- Porém, também impede que uma região de interesse (onde possivelmente encontra-se o mínimo global) seja explorada de forma a refinar as posições
- Assim, velocidades podem ser limitadas de forma a garantir que a variação de posição entre iterações sucessivas não seja muito alta. Uma forma simples de fazer isso é
  1. Se $||\boldsymbol{v}_i(t+1)|| > v_{max}$, faça

$$\boldsymbol{v}_i(t+1) \leftarrow v_{max} \frac{\boldsymbol{v}_i(t+1)}{||\boldsymbol{v}_i(t+1)||} \tag{3}$$

Controle de velocidade II

- Outra forma é incluir um termo de inércia na expressão da velocidade:

$$\boldsymbol{v}_i(t+1) = W\boldsymbol{v}_i(t) + \boldsymbol{c}_1 \odot [\boldsymbol{x}_i^*(t) - \boldsymbol{x}_i(t)] + \boldsymbol{c}_2 \odot [\boldsymbol{x}_{opt}(t) - \boldsymbol{x}_i(t)], \quad (4)$$

influenciando a partícula a permanecer numa mesma direção.

  - Note, no entanto, que o fato da velocidade tornar-se excessivamente grande depende de $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$. Portanto, W deve ser escolhido de forma a contornar a influência destes.
  - Como no algoritmo de recozimento simulado, pode-se começar privilegiando a exploração do espaço de estados e, conforme aumenta-se t, o refinamento das soluções é preferível. Assim, W pode depender de t:

$$W(t+1) = w_{max} - \left(\frac{w_{max} - w_{min}}{t_{max}} \right) t, \quad (5)$$

sendo $w_{max} = 0.9$ e $w_{min} = 0.4$ valores típicos destes parâmetros.

Convergência prematura

- Como já discutido previamente, em funções com múltiplos mínimos locais é possível que algoritmos de otimizações fiquem presos em soluções localmente ótimas.
- Algoritmos naturalmente inspirados são justificados por argumentos geralmente exógenos à matemática ou à análise de algoritmos. Assim, a convergência dos mesmos para uma solução global (ou mesmo local) não pode ser assegurada.
- Outros problemas de convergência podem levar a comportamentos ainda mais inesperados, como, por exemplo
  - A função objetivo carrega uma dependência temporal $f = f(\boldsymbol{x}|t)$, que altera o espaço de busca durante este processo.
  - Existem múltiplas soluções que resultam no mesmo valor de f e, portanto, não podem ser diferenciadas pelo valor da função.

Vamos discutir algumas modificações de forma a aliviar esse tipo de problema.

Modelo predador-presa

- Partículas como aquelas descritas anteriormente são **presas** para um segundo grupo de partículas, chamadas de **predadores**.
- Os predatores movem-se em direção às presas de acordo com a regra

$$\begin{cases} \boldsymbol{v}_{predador}(t+1) = \alpha[\boldsymbol{x}_{opt}(t) - \boldsymbol{x}_{predador}(t)] \\ \boldsymbol{x}_{predador}(t+1) = \boldsymbol{x}_{predador}(t) + \boldsymbol{v}_{predador}(t+1) \end{cases} \tag{6}$$

- Já as presas fogem dos predadores conforme estes se aproximam das mesmas. Isto equivale a inserir um termo repulsivo na velocidade descrita na equação (2):

$$\boldsymbol{v}_{presa}(t+1) = \boldsymbol{v}_i(t+1) + \boldsymbol{A}\exp(-\lambda||\boldsymbol{x}_{presa}(t) - \boldsymbol{x}_{predador}(t+1)||), \tag{7}$$

onde A denota a influência do predador sobre a presa e λ é um fator de escala para a distância predador-presa. Pode-se escolher um predador para atualização ou mesmo a posição média do grupo de predadores.

Partículas carregadas

- Outra forma de produzir diversidade entre as partículas é adicionar um termo repulsivo às velocidades, da mesma forma que a conhecida força de Coulomb da teoria do eletromagnetismo.

$$\boldsymbol{v}_i^{rep}(t+1) = \boldsymbol{v}_i(t+1) + \sum_{j \neq i} \boldsymbol{\tau}_{i,j}(t), \tag{8}$$

onde

$$\boldsymbol{\tau}_{i,j}(t) = \frac{Q_i Q_j [\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)]}{||\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)||^3} \tag{9}$$

caso $D_{nuc} < ||\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)|| < D_{rep}$ ou 0 em caso contrário.

- D_{nuc} é uma descrição do tamanho da partícula. Quando uma segunda está a uma distância menor que D_{nuc} da primeira, o efeito repulsivo cessa.
- D_rep descreve o raio de atuação da força repulsiva. Isto é, para partículas suficientemente distantes, a força não atua.

Enxames em espaços discretos

- Para espaços discretos de busca, o vetor de velocidades pode ainda ser atualizado como na expressão (2).
- Uma vez realizado o cálculo de $\boldsymbol{v}_i(t+1)$, este é discretizado. Para isso, atribui-se uma probabilidade à $k-$ésima entrada do vetor de velocidades de acordo com uma sigmóide:

$$p_{i,k} = \frac{1}{1+\exp(-v_i^k(t+1))} \tag{10}$$

e então gera-se um número aleatório u uniformemente distribuído no intervalo $(0,1)$. E então $\boldsymbol{v}_i$ é discretizado de acordo com a regra

$$\begin{cases} v_i^k(t+1) = 1, \text{ se } u > p_{i,k} \\ v_i^k(t+1) = 0, \text{ caso contrário.} \end{cases} \tag{11}$$

- A atualização de $\boldsymbol{x}_i$ por $\boldsymbol{v}_t$ é feita então de acordo com a regra de adição para representações binárias.

Algoritmos de formigas

- O comportamento social de determinados insetos permite que um agrupamento destes realize tarefas que seriam impossíveis para indivíduos
- Estima-se que apenas 2% das espécies de insetos sejam sociais, porém correspondem a mais de 50% da biomassa de insetos no planeta, indicando que este tipo de comportamento pode ser a chave para sua prevalência
- Algoritmos de formigas constituem uma família de métodos para os quais a comunicação entre pares de indivíduos leva à emergência de um comportamento complexo e organizado

Formigas em grafos I

- Uma forma na qual formigas interagem entre si é através da deposição de feromônios no chão: a presença elevada desta substância indica um caminho que é percorrido por uma quantidade igualmente elevada de indivíduos, o que é um indicativo de optimalidade.
- Suponha um problema de otimização discreta que pode ser representado por um grafo $G = (V, E)$, sendo V o conjunto de vértices do grafo e E o conjunto de arestas do mesmo.
- Pode-se pensar no grafo como um ambiente no qual as formigas estão situadas e, à medida que trafegam entre vértices passando por arestas, feromônios são depositados de forma a reforçar caminhos optimais
- Esse tipo de estratégia é, portanto, utilizada para encontrar caminhos ótimos em grafos de acordo com uma função objetivo que atribui custos aos caminhos.

Formigas em grafos II

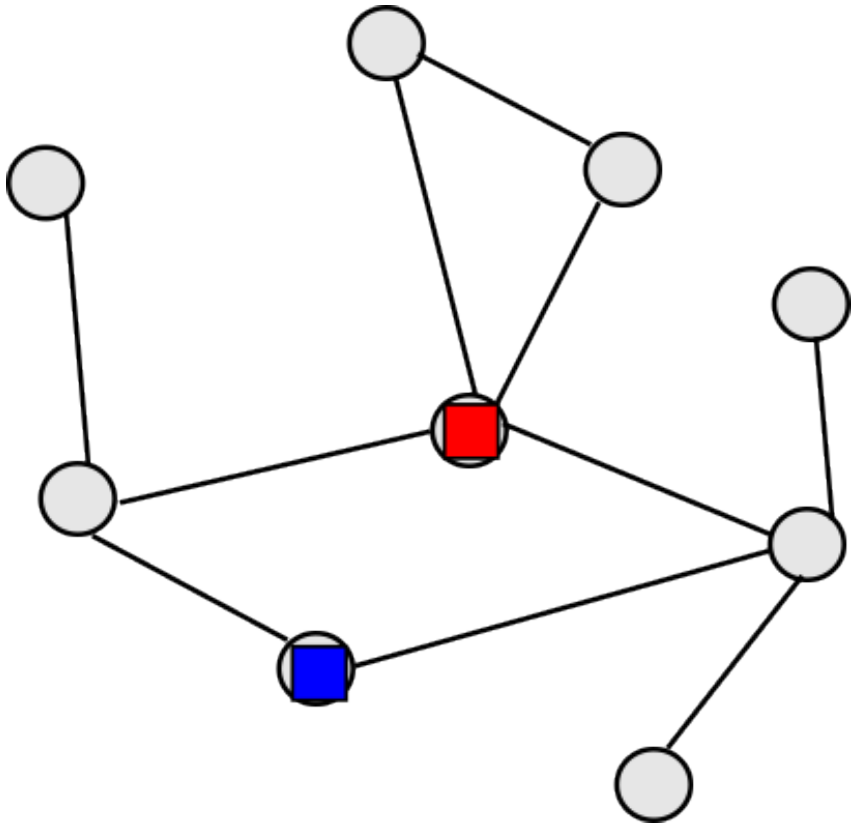

Figura: Exemplo das formigas em um grafo.

Formigas em grafos III

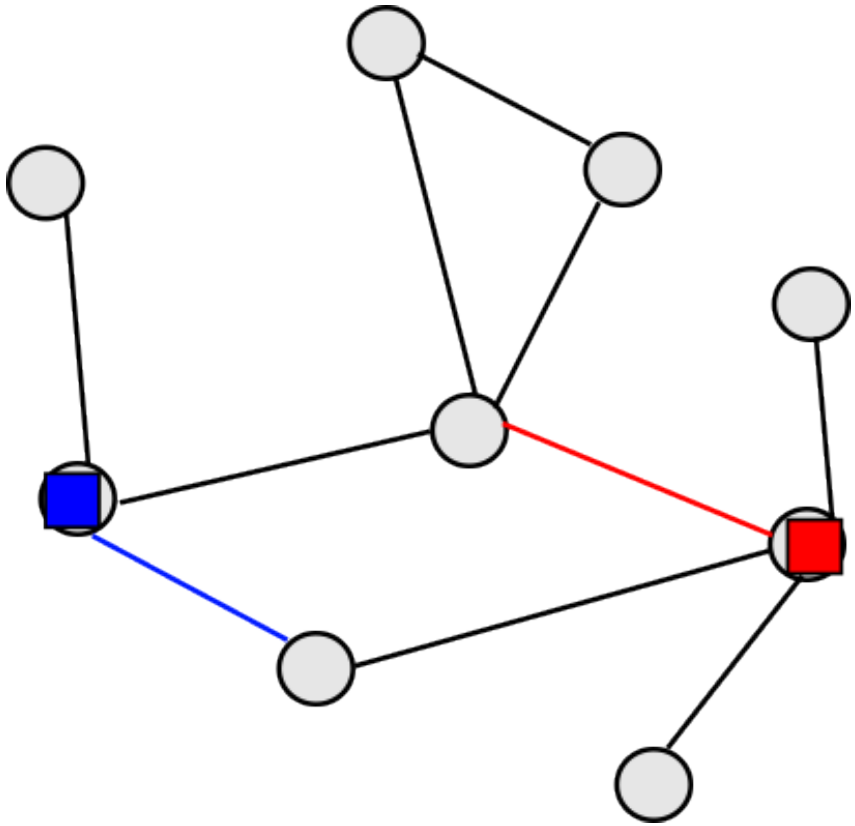

Figura: Ao se deslocarem, feromônios são depositados nas arestas.

Formigas em grafos IV

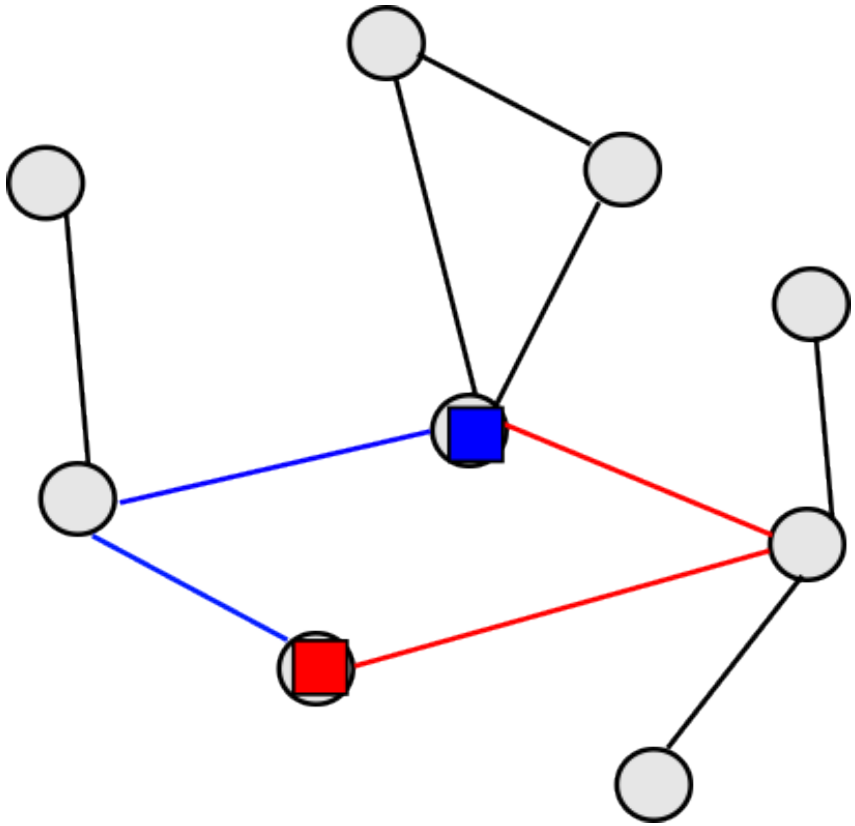

Figura: Ao se deslocarem, feromônios são depositados nas arestas.

Formigas em grafos V

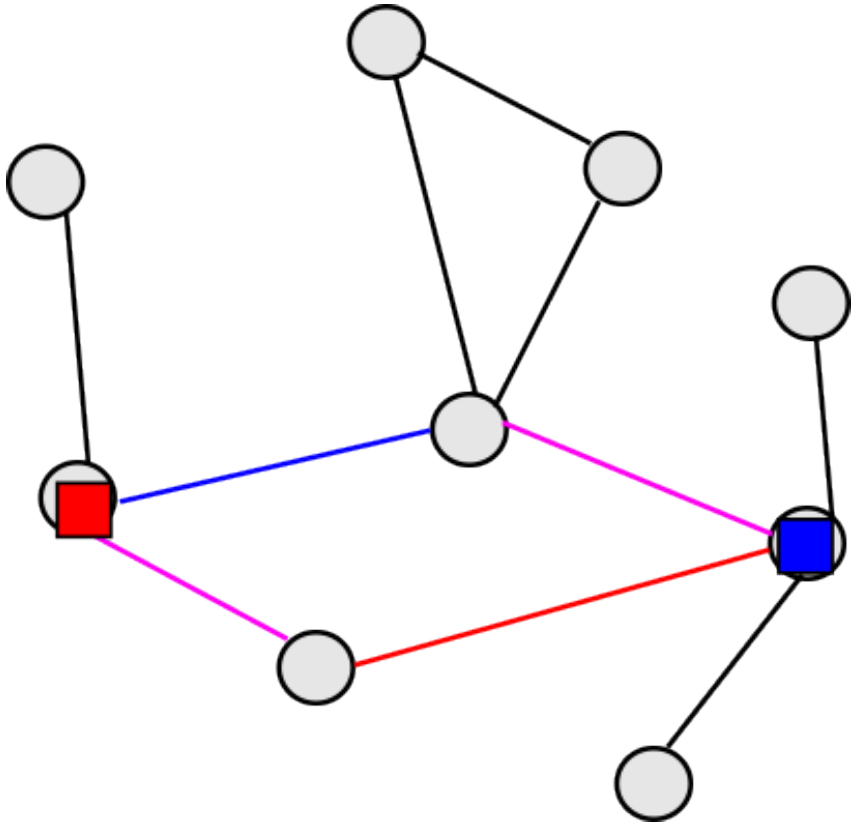

Figura: Caminhos com maior quantidade de feromônio são preferidos pelas formigas.

A matriz de feromônios

- Para um grafo de N vértices, a matriz de adjacência $A_{N\times N}$ é da forma

$$A_{i,j} = \begin{cases} 1, & \text{se } (i,j) \in E \\ 0, & \text{caso contrário.} \end{cases} \tag{12}$$

- Cria-se então uma segunda matriz, $F_{N\times N}$ que é igual à A, mas na qual as entradas não-nulas são ponderadas pela quantidade de feromônio depositada pelas formigas em cada aresta.
- A cada iteração do algoritmo, cada formiga movimenta-se do vértice i para o vértice j com probabilidade

$$p_{i,j} = \frac{F_{i,j}}{\sum_l F_{i,l}}. \tag{13}$$

Mais problemas de exploração I

- A regra de movimento descrita anteriormente pode ser simplista no sentido de que as formigas rapidamente prendem-se a um mínimo local, deixando de explorar o espaço de soluções.
- Em problemas como o do caixeiro viajante, por exemplo, o objetivo é construir um caminho optimal sem passar duas vezes pela mesma cidade. Logo, é útil incluir este tipo de informação no algoritmo na forma de uma heurística.
- Logo, uma primeira modificação é excluir as cidades já visitadas pela $k-$ésima formiga. Sendo $C_i^{(k)}$ o conjunto de vértices atingíveis a partir de i e ainda não visitados pela $k-$ésima formiga, uma primeira modificação de (13) é

$$p_{i,j}^{(k)} = \frac{F_{i,j}}{\sum_{l \in C_i^{(k)}} F_{i,l}}. \tag{14}$$

Mais problemas de exploração II

- Em problemas como o do caixeiro viajante, há ainda um custo para cada aresta, corresponde à distância entre cidades e que é representada na forma de uma matriz $D_{N\times N}$.
- Assim, o custo de deslocamento de uma cidade a outra também serve como heurística para o algoritmo e pode ser utilizado para refinar ainda mais a expressão (14):

$$p_{i,j}^{(k)} = \frac{F_{i,j}^{\nu} D_{i,j}^{-\mu}}{\sum_{l \in C_i^{(k)}} F_{i,l}^{\nu} D_{i,j}^{-\mu}}, \tag{15}$$

onde ν e μ correspondem aos pesos dados à informação de feromônio e à informação heurística, respectivamente.

Deposição e evaporação de feromônios

- Uma vez que uma formiga completa seu percurso, a matriz de feromônios é atualizada de acordo com uma regra

$$F_{i,j}(t+1) = F_{i,j}(t)(1-e) + \Delta_{i,j}. \tag{16}$$

- O parâmetro $e \in (0,1)$ é dito a taxa de evaporação do feromônio e controla o quanto caminhos já percorridos influenciam na busca.
- $\Delta_{i,j}$ controla o quanto de feromônio será depositado em um caminho. Para o problema do caixeiro viajante, uma forma é um incremento inversamente proporcional ao comprimento $L^{(k)}$ do caminho percorrido pela $k-$ésima formiga.

Competição e coperação para detecção de comunidades I

- Um problema comum em grafos é a detecção de estruturas de comunidades nos mesmo. Isto é, a existência de grupos nos quais nós em um mesmo grupo são densamente conectados entre si, mas fracamente conectados com nós externos ao grupo.
- No caso em que não há informação prévia sobre o número de grupos ou sobre o pertencimento dos nós a qualquer comunidade, diz-se que o problema é **não-supervisionado**.
  - Esse problema é muito similar ao problema de **agrupamento** para dados contínuos. Uma abordagem muito utilizada é a de mapear um conjunto contínuo em um grafo e então efetuar um algoritmo de detecção de comunidades para obter um agrupamento.
- Em alguns casos, é possível saber o número de grupos existentes e também o pertencimento de alguns vértices às comunidades, mas não de todos. Nesse caso, o problema é dito **semi-supervisionado**.

Competição e coperação para detecção de comunidades II

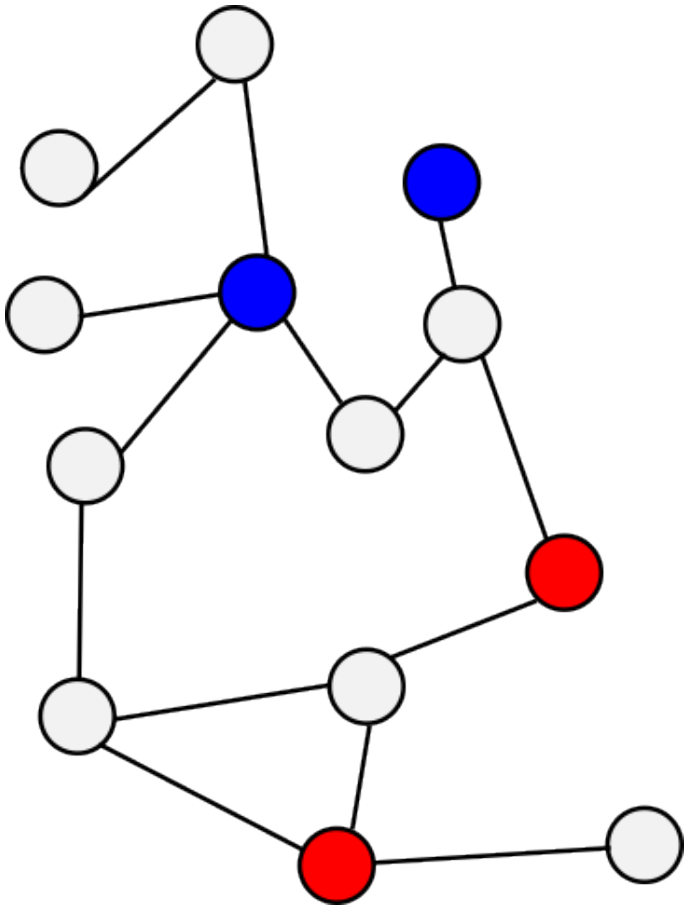

Figura: Exemplo de uma situação do problema de detecção semi-supervisionada de comunidades.

Competição e coperação para detecção de comunidades III

- A dinâmica das formigas também pode ser utilizada neste caso, mas depositando feromônios sobre os nós ao invés das arestas.
- Para cada grupo existente, é inicializado um grupo de formigas que se encarrega de depositar feromônios de forma a marcar território pertencente a um grupo.
- Os diferentes grupos de formigas competem entre si pela dominação do grafo e, ao final do processo, haverá uma divisão do mesmo entre estes grupos que, idealmente, refletirá a verdadeira estrutura de comunidade do grafo.
- Uma discussão mais elaborada desse tipo de algoritmo pode ser vista aqui.

Competição e coperação para detecção de comunidades IV

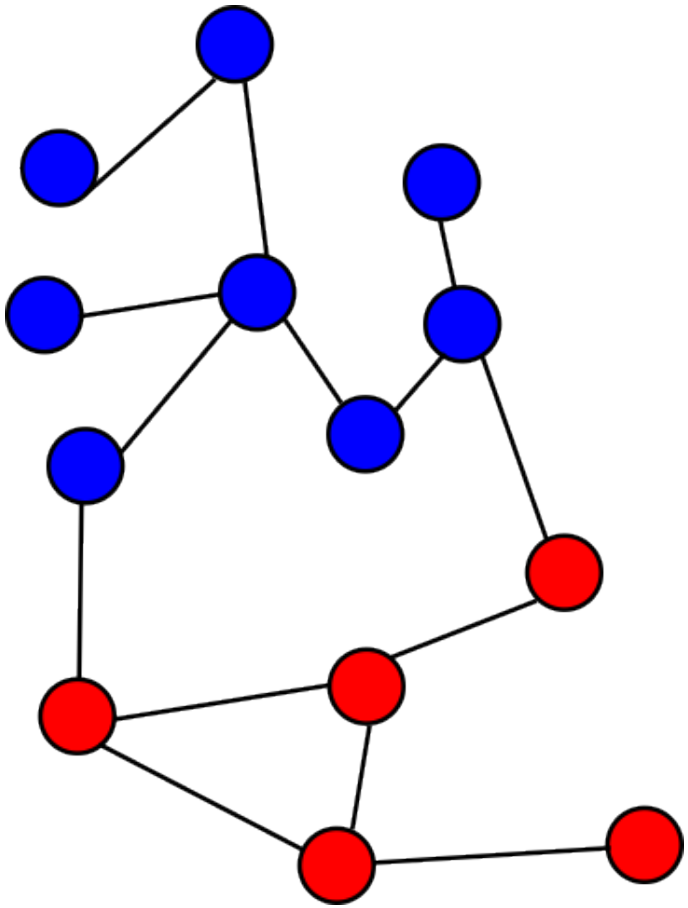

Figura: Exemplo da estrutura de comunidade detectada pelo algoritmo discutido.

Formigas em domínios contínuos

- Em domínios contínuos, a noção de nós e arestas desaparece e a deposição de feromônio se dá sobre pontos no espaço de busca.
- Essa deposição é na forma de uma distribuição gaussiana multimensional

$$F(\boldsymbol{x}) \propto \exp(-||\beta \odot (\boldsymbol{x} - \boldsymbol{x_{min}})||), \tag{17}$$

onde $\boldsymbol{x}_{min}$ é a melhor solução encontrada até agora. Assim, cada formiga sorteia um ponto desta distribuição e verifica se esta melhora a solução.

- Posteriormente, o parâmetro β, que descreve a variância do comportamento das formigas, é atualizado. Sendo m o número de formigas e f a função objetivo, uma regra de atualização é

$$\beta = \frac{1}{\sum_{k=1}^{m}[f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{min})]^{-1}} \sum_{k=1}^{m} \frac{\boldsymbol{x}_k - \boldsymbol{x}_{min}}{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{min})}. \tag{18}$$

Uma discussão conceitual

Como fica a noção de agente racional, tanto no caso dos enxames de partículas como no algoritmo das formigas?

- Um agente controla partículas e formigas?
- Partículas e formigas são agentes?
- Como fica a noção de ambiente nestes casos?

Simulação e solução

- No caso em que os algoritmos são utilizados no âmbito de otimização, estes são executados por um agente de forma a encontrar uma solução para o problema.
- No entanto, formigas e partículas podem ser vistos como agentes em um ambiente descrito pela função objetivo f. A partir daí, características comportamentais dos mesmos podem ser estudadas enquanto resolvem o problema de otimização.

Exercícios sugeridos

Revise os algoritmos do enxame de partículas e da colônia de formigas e como contornar o problema de parada prematura nos mesmos, que é o caso onde estes ficam presos em mínimos locais em função de uma baixa exploração do espaço de busca. Escreva pseudocódigos para as versões básicas dos mesmos (isto é, sem as modificações para evitar parada prematura).

Atividade

Implemente os dois métodos de controle de velocidade para o algoritmo do enxame de partículas discutidos nesses slides.

Trabalho

- Implemente o modelo predador-presa com controle de velocidade para as presas. Utilize ambos os métodos de controle de velocidade.
- Implemente o modelo de partículas carregas com controle de velocidade. Utilize ambos os métodos de controle de velocidade.
- Compare sua implementação com o algoritmo original visto em aula e com as modificações que você fez na atividade.
- Use como custos a função de Rastrigin 2d e a função de Rosenbrock

$$R(x, y) = 100(y - x^2)^2 + (1 - x)^2 \tag{19}$$

  - Para cada método, avalie pelo menos três quantidades distintas de partículas.
  - Os métodos possuem diferentes hiperparâmetros que devem ser escolhidos por quem os implementa. Faça testes de forma a encontrar valores que produzam melhores resultados.
  - Utilize diversas condições iniciais (pelo menos 10) e tome a média entre estas, alternando o número de partículas. Calcule média e desvio padrão das quantias de interesse.