Otimização com Colônias de Formigas

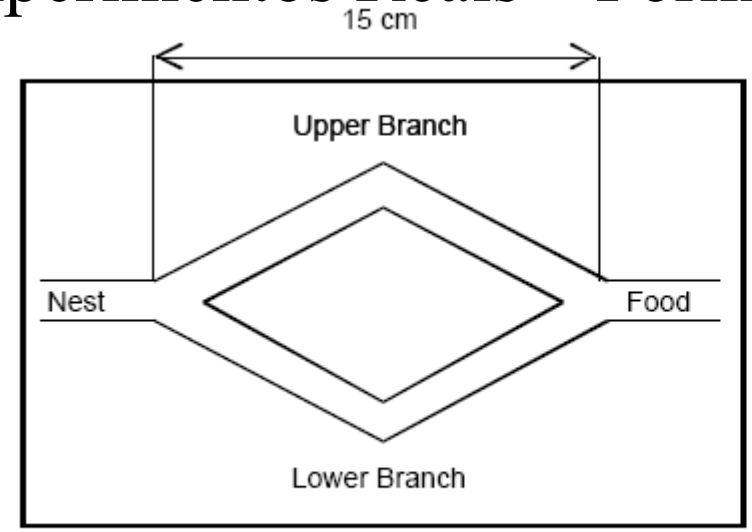
Computação Natural

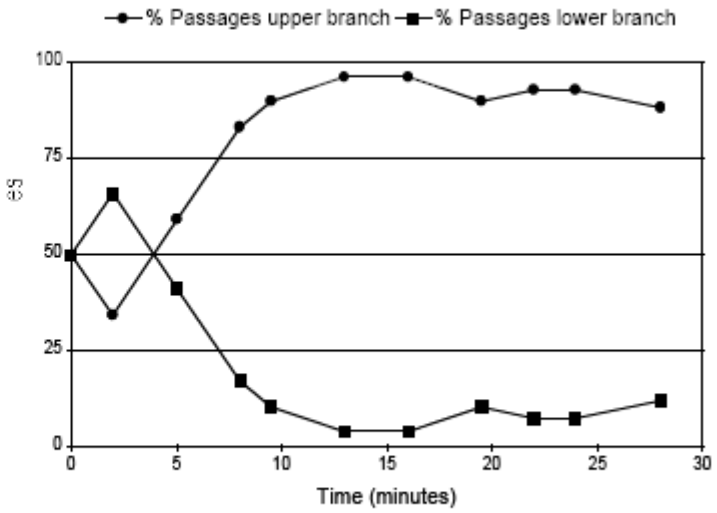
Gisele L. Pappa

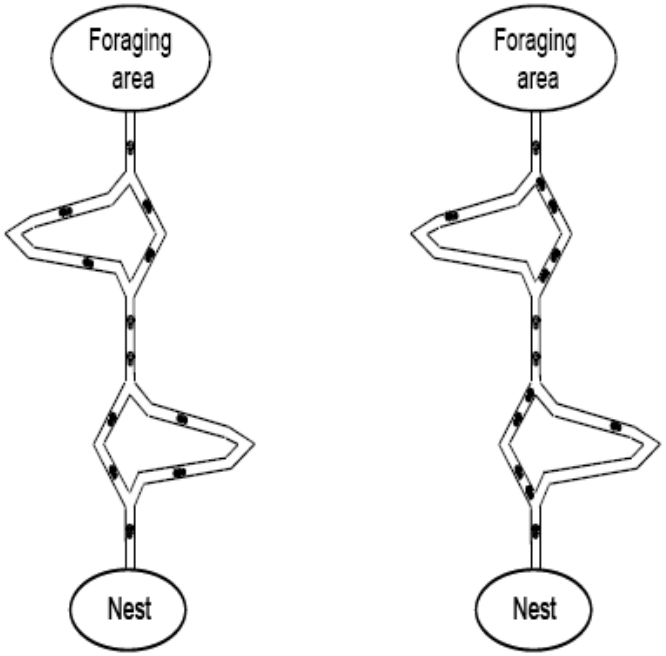
Recapitulando...

- Baseados no comportamento real dos formigueiros
- Principais características:
  - Comportamento emergente
    - Cada um contribui pouco, mas o conjunto leva a uma boa solução
  - Feedback positivo
  - Estigmergia (comunicação por feromônio)

Experimentos Reais – Formigas em Busca de Comida

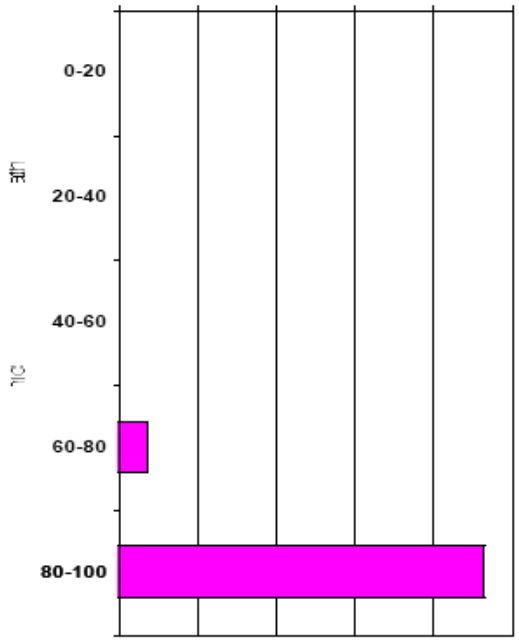






(a)

(b)



(c)

ACO (maxIt, N, τ_o)

Inicializa τ_{ij} (igualmente para cada aresta)

Distribui cada uma das k formigas em um nó selecionado aleatoriamente

t = 1

while (t < maxIt) do //número de iterações

for i = 0 to N do //para cada formiga

1. Constrói uma solução aplicando uma regra de transição probabilística (*e*-1) vezes // *e* é o número de arestas do grafo
2. Avalia o custo de cada solução construída
3. Atualiza melhor solução

end for

Atualiza as trilhas de feromônio

t = t + 1

end while

Aplicações

- Caixeiro viajante
  - Utilizada por empresas de telefonia para rotas de tráfico de ligações telefônicas
- Robótica coletiva
- Mineração de dados

Ant-Miner

[Parpinelli et al 2002]

Problemas

Indução de Regras de Classificação usando ACO

- Dado: um conjunto de treino
- Objetivo: regras de classificação para conjunto de treino
- Saída: lista ordenada de regras de decisão

Conjunto de treinamento

Classe

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Regras de decisão

IF (humidity = high) and
(outlook = sunny) THEN
play = no (3.0/0.0)

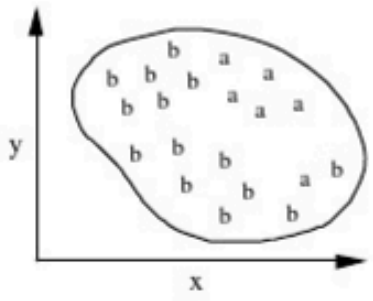
Regras sempre seguem o formato
IF (ATRIBUTO OPERADOR VALOR)
THEN CLASSE

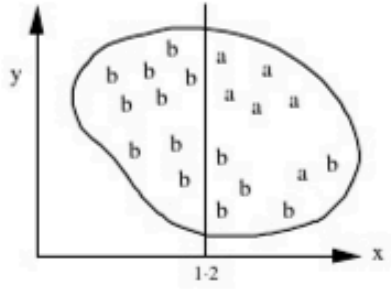
Exemplo de regra

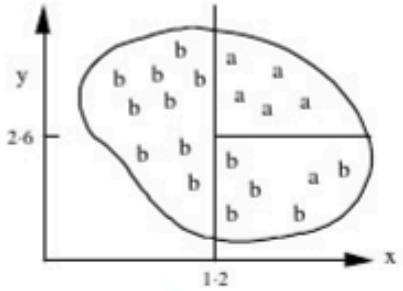
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

IF (humidity = high) and (outlook = sunny) **THEN** play=no (3.0/0.0)

Exemplo de Regras



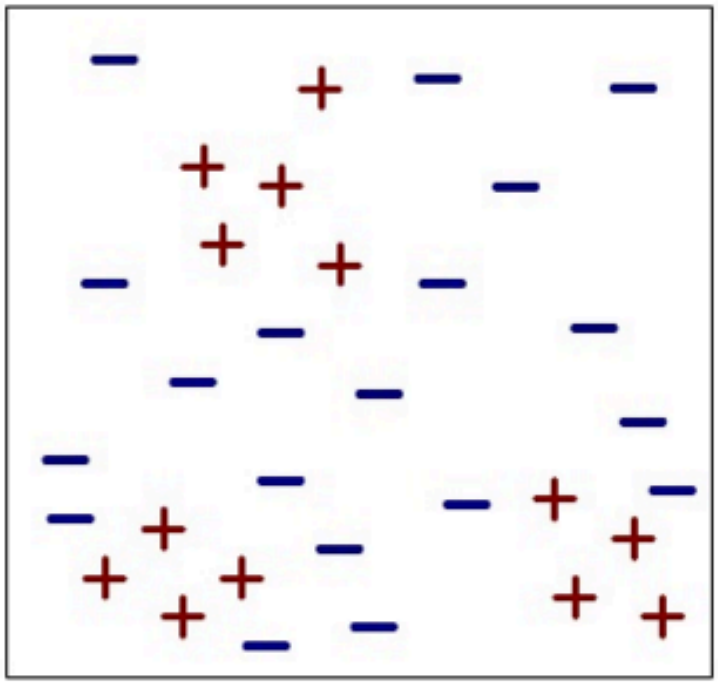




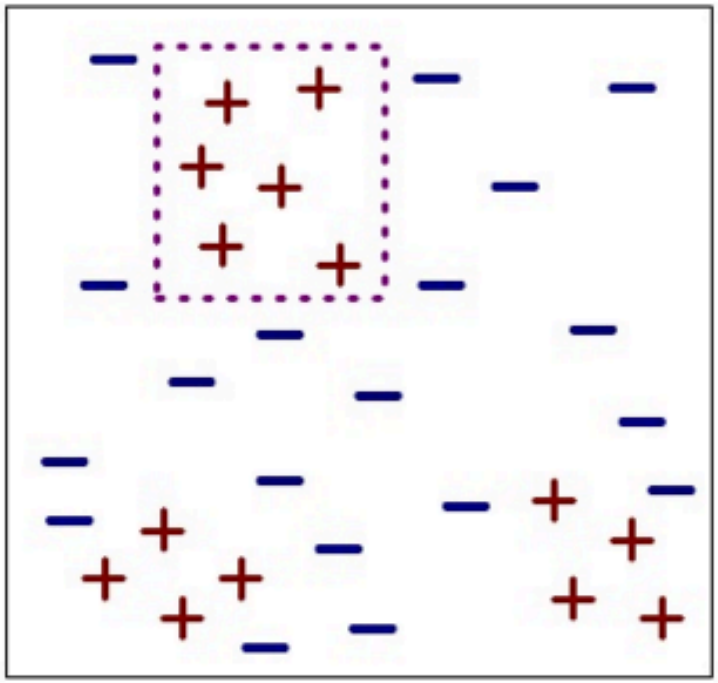
If true then class = a

If x > 1.2 and y > 2.6 then class = a

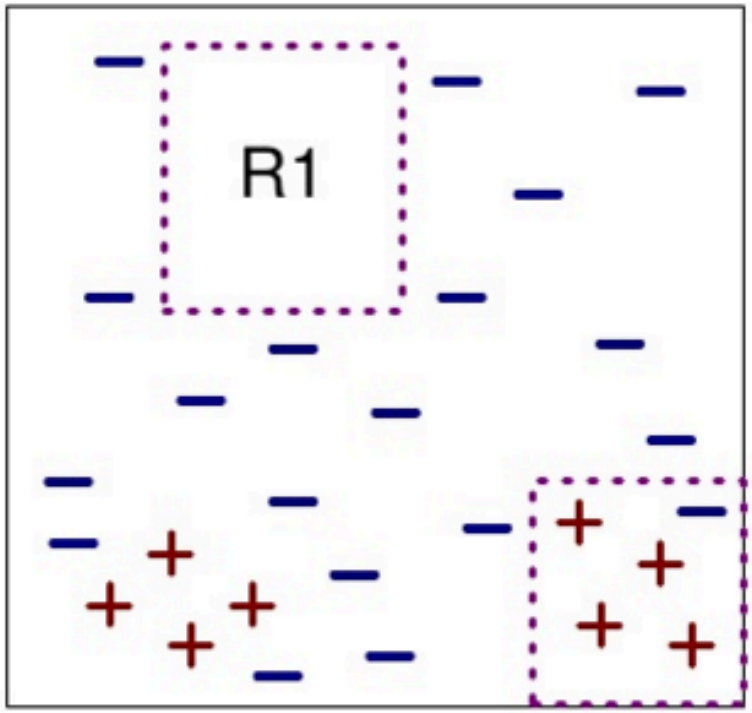
If x > 1.2 then class = a



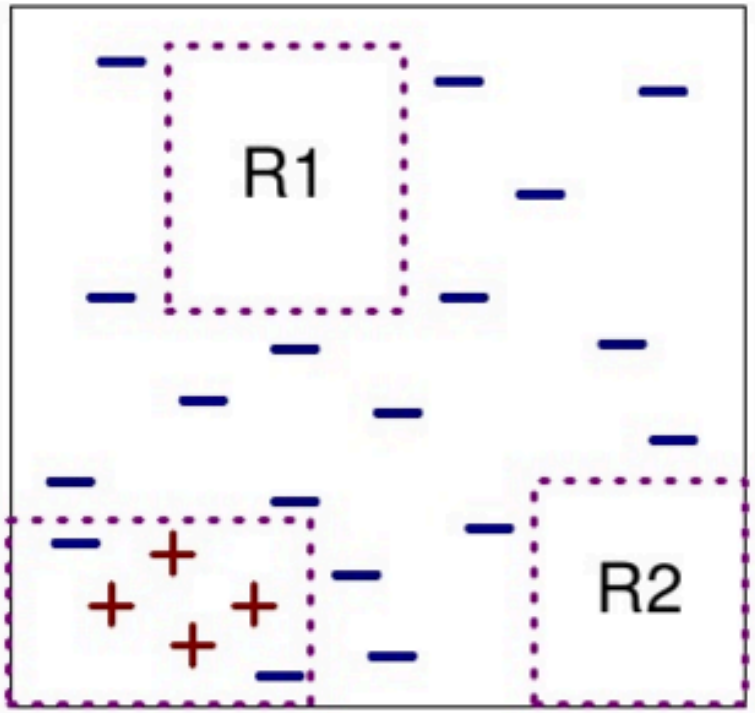
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

Sequential Covering

while existirem elementos no conjunto de treinamento

- Aprenda uma regra que cubra parte dos exemplos de treinamento
- Remova os exemplos cobertos pela regra do conjunto de treinamento

end while

Como um AIR aprende regras?

- Baseado em 4 elementos principais

1. Linguagem de representação de regras
2. Um mecanismo de busca
3. Um método de avaliação das regras encontradas
4. Métodos de *pruning*

Ant-Miner

- Segue a abordagem de cobertura sequencial
  - Formigas trabalham para gerar uma regra de decisão
  - ACO executado várias vezes (a cada execução um número menor de exemplos é considerado)
- Cada formiga inicialmente representa uma “regra vazia”
- Formigas incrementalmente adicionam um termo (condição) a regra (par atributo+valor) de acordo com uma métrica de entropia e a quantidade de feromônio depositada

Ant-Miner

- Cada regra deve ter uma cobertura mínima
  - Serve de critério de parada e controla *overfitting*
- Ao final da iteração, a classe da regra é escolhida de acordo com a classe da maioria dos exemplos cobertos.
- Versão inicial só trabalha com atributos categóricos

Ant-Miner

WHILE (No. of cases in the Training set > Max-uncovered allowed)

i=0;

REPEAT

i=i+1

Ant_i incrementally constructs a classification rule

Prune the just constructed rule

Update the pheromone of the trail followed by Ant_i

UNTIL (i ≥ No-of-Ants) or (Ant_i covers less examples than a predefined threshold)

Select the best rule among all constructed rules

Remove the cases correctly covered by the selected rule from the training set

END WHILE

AntMiner

Inicialização do feromônio

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i}$$

a é o número de atributos,
b_i é o número de valores q a pode assumir

Regra de transição

$$P_{ij} = \frac{\eta_{ij}^{\alpha} \ \tau_{ij}^{\beta}}{\sum_{i=1}^{a} \sum_{j=1}^{b_i} \eta_{ij}^{\alpha} \ \tau_{ij}^{\beta}, \forall i \in I}$$

η_{ij} – quanto maior seu valor, mais importante para o classificação é o valor j do atributo i

Ant-Miner

Avaliação

$$\eta_{ij} = \frac{\log_2 k - H(W \mid A_i = V_{ij})}{\sum_{i=1}^{a} \sum_{j=1}^{b_i} \left(\log_2 k - H(W \mid A_i = V_{ij})\right)}$$

k é o número de classes,
W é a classe

Qualidade da regra é dada de acordo com a entropia normalizada, que varia de 0 a $\log_2 k$

$$H(W \mid A_i = V_{ij}) = -\sum_{w=1}^{k} (P(w \mid A_i = V_{ij}).\log_2(w \mid A_i = V_{ij}))$$

Existe também um método guloso para podar as regras

Ant-Miner- Resultados

Data Set	Ant-Miner's predictive accuracy (%)	CN2's Predictive accuracy (%)
Ljubljana breast cancer	75.28 ± 2.24	67.69 ± 3.59
Wisconsin breast cancer	96.04 ± 0.93	94.88 ± 0.88
tic-tac-toe	73.04 ± 2.53	97.38 ± 0.52
dermatology	94.29 ± 1.20	90.38 ± 1.66
hepatitis	90.00 ± 3.11	90.00 ± 2.50
Cleveland heart disease	59.67 ± 2.50	57.48 ± 1.78

	No. of rules		No. of terms / No. of rules	
Data Set	Ant-Miner	CN2	Ant-Miner	CN2
Ljubljana breast cancer	7.10 ± 0.31	55.40 ± 2.07	1.28	2.21
Wisconsin breast cancer	6.20 ± 0.25	18.60 ± 0.45	1.97	2.39
tic-tac-toe	8.50 ± 0.62	39.70 ± 2.52	1.18	2.90
dermatology	7.30 ± 0.15	18.50 ± 0.47	3.16	2.47
hepatitis	3.40 ± 0.16	7.20 ± 0.25	2.41	1.58
Cleveland heart disease	9.50 ± 0.92	42.40 ± 0.71	1.71	2.79

Recapitulando....

- ACO combina os benefícios de uma heurística local, baseada no problema, com o benefício de uma busca global, baseada em populações

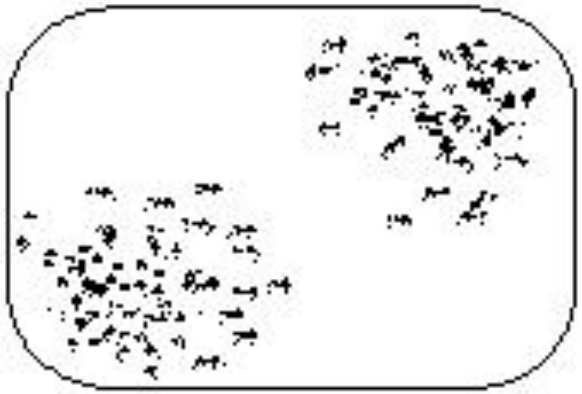
Recapitulando...

- ACO clássico considerava como as formigas buscavam por comida
- Existem outras atividades que podem ser utilizadas como inspiração
  - Organização de larvas
  - Organização dos cemitérios

ACO para Agrupamento

(a)

(b)

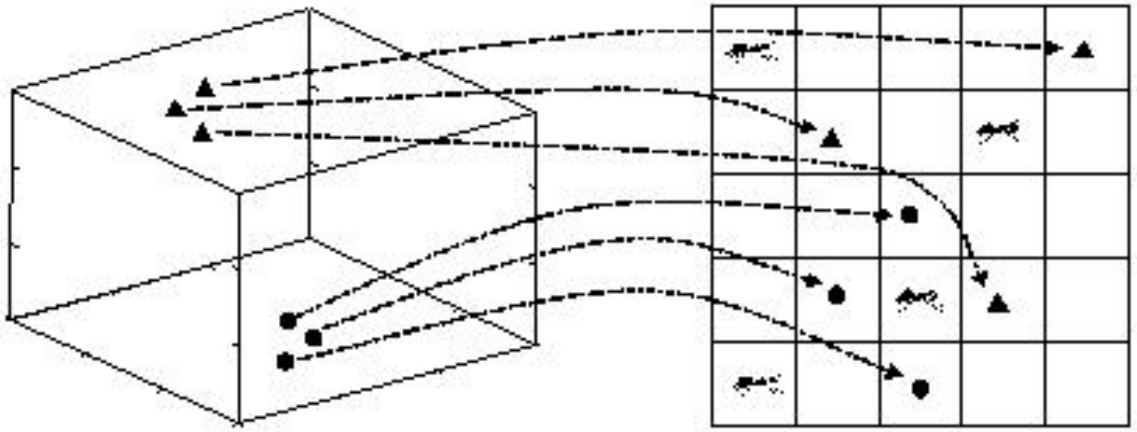


(c)

(a) Inicialmente, corpos aleatoriamente distribuídos.
(b) Depois de algum tempo, corpos começam a ser empilhados.
(c) Agrupamentos de corpos são formados.

OCF para Agrupamento

- Idéias gerais
  - Formigas andam em uma matriz 2D
  - Máximo de uma formiga por célula



  - Formigas movem objetos a fim de agrupá-los

OCF para Agrupamento

- Itens isolados devem ser recolhidos e depositados em um novo local, com probabilidade

$$p_p = \left(\frac{k_1}{k_1 + f} \right)^2$$

onde f é a fração de itens próximos a formiga, e k_1 é uma constante.

– f >> k1, probabilidade de pegar um objeto numa região com muitos outros objetos é pequena, e vice-versa

- A probabilidade de uma formiga depositar um item sendo carregado é

$$p_d = \left(\frac{f}{k_2 + f} \right)^2$$

onde k_2 é uma outra constante.

– f << k2, probabilidade de depositar um objeto numa região com muitos outros objetos é pequena, e vice-versa

Algoritmo de Agrupamento baseado em Formigas

- Algoritmo de Agrupamento baseado em Formigas (ACA)
  - Aplicação: análise de dados.
  - As formigas se movem em uma matriz 2D e consideram uma região de vizinhança de área s^2 , que corresponde a um quadrado Neigh($s \times s$) de $s \times s$ células ao redor da célula atual r

Algoritmo de Agrupamento baseado em Formigas

– Função f que calcula a densidade local de itens na vizinhança do objeto $\mathbf{x}_i$ situado na célula r:

$$f(\mathbf{x}_i) = \begin{cases} \frac{1}{s^2} \sum_{\mathbf{x}_j \in \text{Neigh}_{(s\times s)}(r)} \left[1 - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\alpha}\right] & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $f(\mathbf{x}_i)$ é uma medida da similaridade média de um objeto $\mathbf{x}_i$ com um outro objeto $\mathbf{x}_j$ em sua vizinhança
- α define a escala de dissimilaridade
- $d(\mathbf{x}_i,\mathbf{x}_j)$ é a distância euclidiana entre duas instâncias de dados no seu espaço de dados original (L-dimensional).

Algoritmo de Agrupamento baseado em Formigas

- As probabilidade de pegar e soltar um item são definidas pelas seguintes fórmulas:

$$p_p(\mathbf{x}_i) = \left(\frac{k_1}{k_1 + f(\mathbf{x}_i)} \right)^2$$

$$p_d(\mathbf{x}_i) = \begin{cases} 2f(\mathbf{x}_i) & \text{if } f(\mathbf{x}_i) < k_2 \\ 1 & \text{otherwise} \end{cases}$$

- Aplicação:
  - Análise de dados exploratória, onde existe um conjunto de dados cujos “grupos” (classes) são desconhecidos, e alguma informação deve ser extraída desses dados

Resultados

- Observou-se que o algoritmo apresentado apresenta mais grupos no espaço projetado que no espaço real.
- Seguintes mudanças foram propostas:
  - Uso de formigas se movendo a velocidades diferentes
  - Memória
  - “Distúrbios” de comportamento

Mudanças sugeridas

- Formigas com diferentes velocidades
  - Formigas que se movem mais rápido são menos seletivas, e isso se refletirá no cálculo da densidade local dos itens
  - Velocidade influencia as probabilidades de pegar ou deixar um item
- Memória
  - Formiga guarda os últimos *m* itens deixados e suas posições
  - Cada vez que a formiga pega um novo item, ele é comparado com os itens na memória
  - Levado para a célula mais similar a ele

Mudanças sugeridas

- “Distúrbio de comportamento”
  - Formiga tem a capacidade de destruir grupos que não tenham recebido nenhum item por um determinando período de tempo

MUTE
Compartilhamento de Arquivos Anônimo

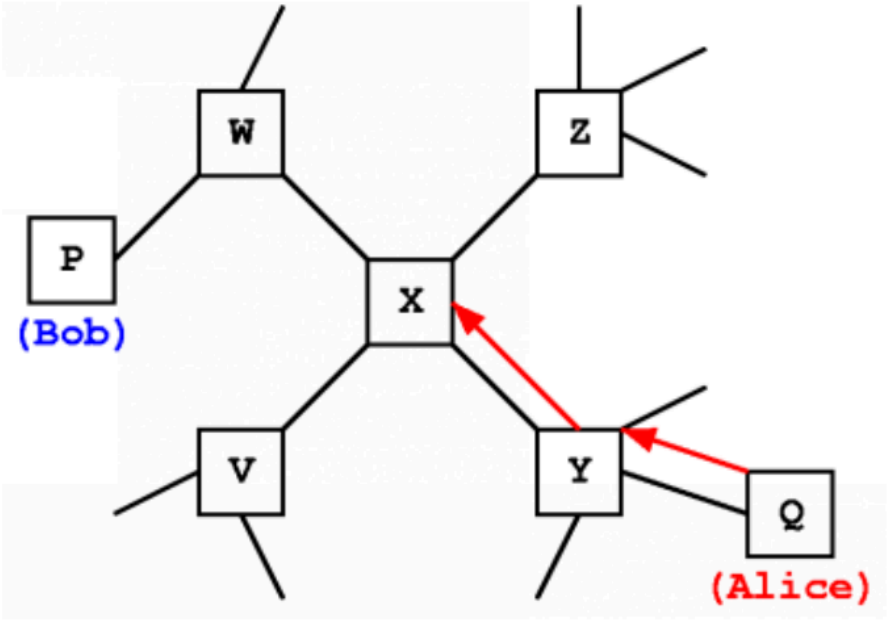
http://mute-net.sourceforge.net/

MUTE

- Problema: permitir compartilhamento de arquivos entre usuários usando P2P mantendo a privacidade do usuário
- Normalmente, quando uma mensagem é enviada, os IPs de origem e destino são visíveis
- MUTE quer garantir que nenhum dos nós saiba qual computador o remetente ou destinatário está usando

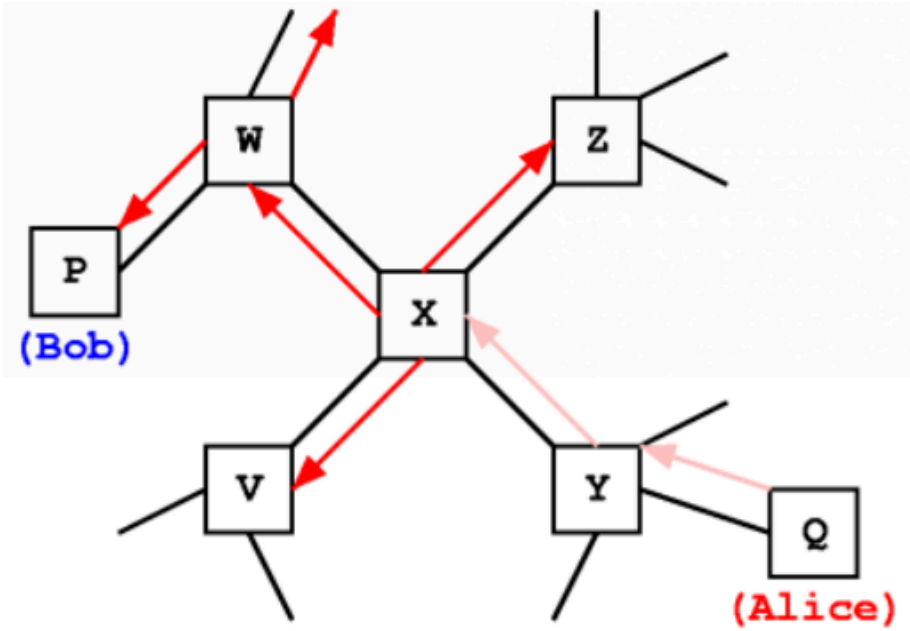
MUTE

- X recebe uma mensagem de Alice para Bob através de Y, um de seus vizinhos
- X pode não saber onde o Bob está na rede
- Porém, quando recebe a mensagem, X aprende algo sobre Alice: que a mensagem de Alice vem através de Y
- Se no futuro X quiser mandar uma mensagem para Alice, ele sabe que deve ser via Y



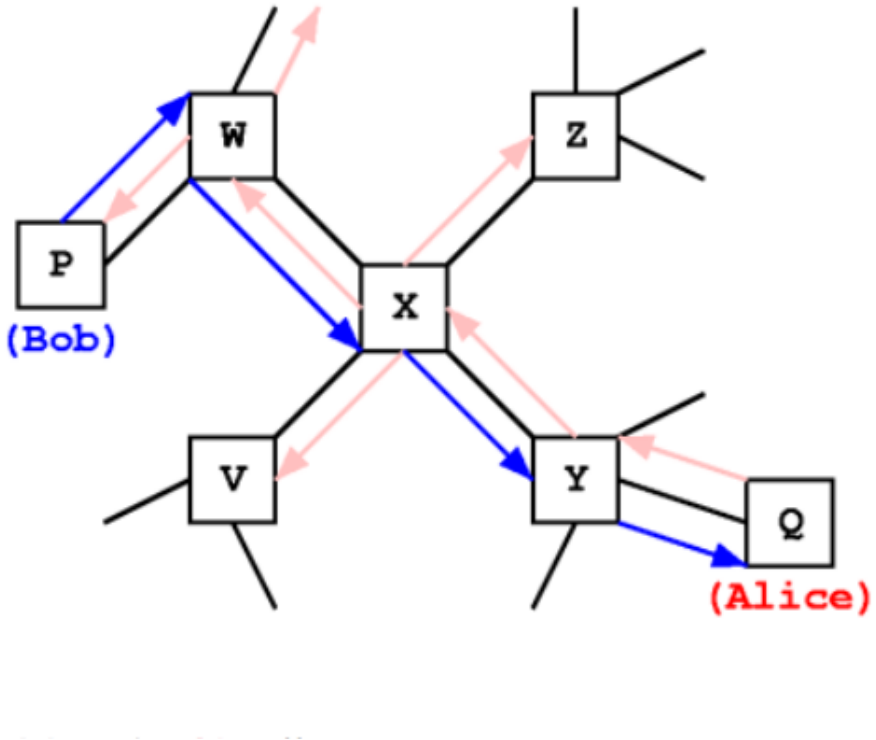
MUTE

- X não sabe nada sobre Bob, então ele envia a mensagem para seus vizinhos
- Um dos vizinhos de X pode ter informação sobre a direção em que Bob está.
- Se o Bob existir, em algum momento a mensagem vai chegar até ele.



MUTE

- Através da busca por Bob, a mensagem deixou pistas sobre Alice.
- Se Bob quiser responder a Alice, a resposta pode seguir o mesmo caminho.
- Isso deixa pistas de como encontrar Bob (sem revelar o IP de Bob)



Um nó nunca pode dizer “minha vizinha é Alice”
Ele só pode dizer “meu vizinho sabe mais sobre Alice do que eu"

Bibliografia

Leitura recomendada

- M. Dorigo, G. Di Caro and L. M. Gambardella. Ant Algorithms for Discrete Optimization. Artificial Life, 5(2):137-172.
- Parpinelli, R.S., Lopes, H.S., Freitas, A.A. "Data mining with an ant colony optimization algorithm". IEEE Transactions on Evolutionary Computation, special issue on Ant Colony Algorithms, v. 6, n. 4, p. 321-332, August, 2002