# Agrupamento de Itens Utilizando Colônia de Formigas

**Ediana da Silva de Souza**[1], **Êrica Peters do Carmo**[1]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brasil

{edianadasilvadesouza,ericapetersc}@gmail.com

***Resumo.*** *Análise de agrupamento de dados tem sido amplamente utilizada em diferentes áreas, visto que através de assimilações de informações torna-se possível gerar diferentes análises e conhecimentos. Para isso, é necessária a utilização de métodos eficientes para realizar este agrupamento. Nesse sentido, este artigo tem como objetivo relatar o desenvolvimento de um algoritmo de agrupamento baseado em colônias de formigas e os experimentos realizados. O estudo da literatura realizado indica que este método de agrupamento tem se tornado uma alternativa interessante aos métodos tradicionais e tem sido aplicado em diferentes áreas e contextos. Como resultado da implementação, conclui-se a eficácia do algoritmo desenvolvido para determinados casos e as diferenças dos resultados obtidos em relação às propriedades dos agentes envolvidos.*

## 1. Introdução

O processo de agrupamento de objetos físicos ou abstratos em classes ou grupos de objetos similares é chamado de clusterização [Espenchitt et al. 2008]. Uma das formas de realizar este processo é através de algoritmos inspirados em comportamentos biológicos, um desses comportamentos é simulado pelo algoritmo baseado em colônias de formigas, proposto inicialmente em 1991 [Espenchitt et al. 2008]. Este é uma metaheurística na qual uma colônia de formigas artificiais coopera para encontrar boas soluções para problemas de otimização [Dorigo and Stutzle 204]. O modelo tratado neste trabalho é inspirado em Deneubourg et al [apud Jafar and Sivakumar 2010], que descreve um comportamento de colônias de formigas chamado de organização de cemitério, de forma a compor um sistema onde a influência aleatória e probabilística atua sobre seus componentes.

### 1.1. Justificativa e Motivação

Os dados armazenados em um banco de dados, por exemplo, contém um conhecimento oculto valioso, visto que se pode fazer entendimentos e assimilações das informações através de agrupamento de dados [Jafar and Sivakumar 2010]. Esse agrupamento pode ser obtido por algoritmos baseados em enxames que são vistos como uma alternativa à métodos de análise manual e de armazenamento em clusters mais convencionais, como armazenamento hierárquico, visto que possuem uma maior escalabilidade [Handl and Meyer 2007]. Este trabalho justifica-se, então, pela implementação de um método de clusterização capaz de agrupar um conjunto de dados que apresentam semelhanças entre si.

## 1.2. Objetivo

O objetivo deste trabalho foi implementar um algoritmo de clusterização que fosse capaz de simular o comportamento de uma colônia de formigas para agrupamento de dados homogêneos. O modelo conceitual do algoritmo é dado por uma grade bidimensional e um conjunto de agentes - formigas vivas - responsáveis por agrupar um conjunto de itens - formigas mortas. Tanto os agentes quanto os itens que deveriam ser agrupados foram inicializados aleatoriamente no ambiente, e ao final da execução do algoritmo esperava-se encontrar os itens dispostos em clusters.

## 1.3. Agrupamento Baseado em Colônias de Formigas

Com o intuito de definir o estado da arte em relação ao agrupamento baseado em colônias de formigas, uma breve revisão da literatura foi realizada. Abaixo, são descritas as principais informações encontradas.

Algoritmos baseados em inteligência de enxame têm emergido como uma alternativa às técnicas de clusterização mais convencionais [Jafar and Sivakumar 2010]. Um deles é inspirado no comportamento de colônias de formigas, e para entendê-lo, é preciso observar as características desses insetos.

Formigas são seres eusociais, ou seja, organizam-se em sociedade e são incapazes de sobreviver por si só. Assim, as atividades dentro de uma colônia de formigas são coordenadas de acordo com todos os indivíduos do grupo. Porém, diferem-se de outros seres que organizam-se em sociedade devido ao fato de não haver a presença de uma formiga que represente o centro de controle do grupo. Outra característica importante desses insetos é que a comunicação dentro de uma colônia não acontece de maneira direta, e sim conforme cada indivíduo modifica o ambiente ao seu redor. Ainda de acordo com Jafar e Sivakumar [2010], uma colônia de formigas apresenta três características essenciais para os sistemas computacionais: grau de paralelismo, auto-organização e tolerância a falhas.

Destaca-se ainda um fenômeno observável dentro de uma colônia de formigas chamado organização de cemitério e caracterizado pelo comportamento das formigas de criar pilhas com os corpos das formigas mortas - chamadas de cemitérios - de forma a ”limpar”os formigueiros. A Figura 1. apresenta esse fenômeno observado em um experimento realizado com formigas reais descrito em [Bonabeau et al. 1999].

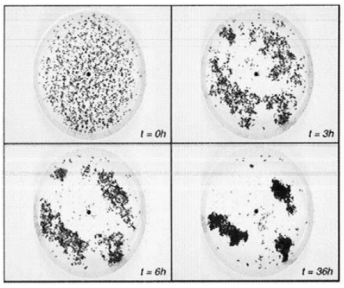



**Figura 1. Organização de cemitério, comportamento observado em experimento com formigas reais. Fonte: [Bonabeau et al. 1999]**

A partir desse comportamento, foi criado o método de agrupamento baseado em colônia de formigas, modelado pela primeira vez por Deneubourg et al. [1990] com o

objetivo de realizar tarefas na área de Robótica. Posteriormente, Lumer e Faieta [1994] modificaram o algoritmo para que se tornasse aplicável na análise numérica de dados.

O comportamento do algoritmo dá-se da seguinte forma: os agentes - formigas vivas - e diversos itens (dados homogêneos ou heterogêneos) são distribuídos randomicamente em um espaço denotado por uma matriz; Os agentes podem ter dois estados, carregados ou vazios, e movimentam-se aleatoriamente para células adjacentes no ambiente; Se o agente chegar em uma célula que contém um item e estiver vazio, ele pega esse item com probabilidade $P_c$; Se o agente chegar em uma célula vazia e estiver carregado, ele solta o item que está carregando com probabilidade $P_d$; Caso contrário, o agente movimenta-se para outra célula aleatoriamente.

De acordo com Jafar e Sivakumar [2010], para dados homogêneos, as fórmulas que calculam $P_c$ e $P_d$ podem ser definidas da seguinte forma:

$$P_c = \left(\frac{k1}{k1+f}\right)^2 \qquad P_d = \left(\frac{f}{k2+f}\right)^2$$

$P_c$ = probabilidade da formiga pegar um item;

$P_d$ = probabilidade da formiga largar um item;

f = fração entre os itens percebidos ao redor e as células dentro do raio de visão da formiga

k1, k2 = constantes de limite

Nos trabalhos relacionados, essas fórmulas foram adaptadas de acordo com as otimizações feitas no algoritmo e a sua finalidade.

Assim, conforme o comportamento descrito, ao final da execução do algoritmo (denotado por um número de iterações ou tempo máximo), percebe-se que os itens são agrupados pelos agentes em diferentes clusters. Nesse sentido, diversos trabalhos buscam utilizar esse método de clusterização, não só em mineração de dados como também em diversas áreas, por exemplo: mineração de texto [Hoe et al. 2002], partição de grafos [Kuntz and Snyers 1999], computação forense [Wang and Hao 2009] e análise do tráfego [Ekola et al. 2004].

### 1.4. Organização do Texto

Este trabalho está estruturado da da seguinte forma: a seção 2 apresenta os métodos e justificativas de desenvolvimento do algoritmo implementado. A seção 3 fornece descrições sobre os experimentos realizados no ambiente e os resultados alcançados. A seção 4 estuda os resultados obtidos. Finalmente, conclusões e direções de pesquisas futuras são apresentadas na seção 5.

## 2. Metodologia de Desenvolvimento ERICA

Conforme o objetivo do presente trabalho, foi implementado um algoritmo para simular o comportamento de agrupamento de corpos observado em colônias de formigas. As características desse algoritmo são descritas nessa seção.

O programa (disponível em: `https://drive.google.com/open?id=19me18bjjiWPdQ9vo9Tefc9UzcmlCJKXS`) foi feito com a linguagem Python de-

vido à familiarização das autoras com a mesma e a biblioteca PyGame utilizada na criação da interface gráfica. Em sua execução, dois parâmetros devem ser passados como entrada:o tempo máximo de execução em minutos e o raio de visão da formiga (mínimo 1 e máximo 5). Este raio diz respeito ao número máximo de células que cada agente consegue enxergar em cada direção, pois eles têm uma visão parcial do ambiente. A Figura 2. exemplifica três possíveis raios de visão.

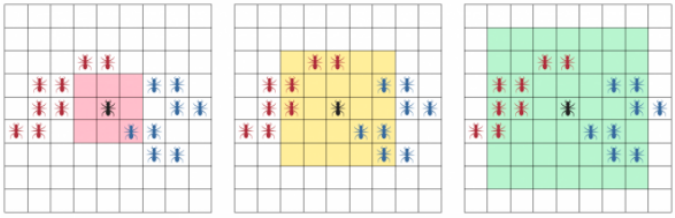


**Figura 2. Raios de visão: uma célula (esquerda), duas células (meio) e três células (direita). Fonte: [TuringFinance 2015]**

O ambiente gerado pelo programa é uma matriz 60x60 não circular, onde inicialmente são dispostos 125 agentes (formigas vivas) e 1000 itens (formigas mortas) de maneira aleatória com uma distribuição uniforme. A proporção de preenchimento do ambiente por itens é de 27,78%. Nota-se que o número de elementos no ambiente foi definido de acordo com testes de desempenho do algoritmo. Sobre os elementos dispostos no ambiente, os itens são homogêneos e os agentes são formados por uma estrutura que guarda a sua posição no ambiente e seu estado (carregando um item ou vazio).

Cada célula do ambiente pode ser ocupada por no máximo um item, porém, mais de um agente pode ocupar a mesma célula. Assim um agente não enxerga o outro como obstáculo. Devido à redução da complexidade, o processamento do programa é sequencial, logo, não existe a possibilidade de duas formigas pegarem ou largaram um item ao mesmo tempo.

Após a inicialização do ambiente, de agentes e itens, começa a execução da função principal *ants_behaviour(anthill)* que recebe como parâmetro o vetor *anthill* formado por todos os agentes dispostos no ambiente. O pseudocódigo dessa função é estruturado da seguinte forma:

Para cada formiga no *anthill*:
    Se a célula do ambiente estiver vazia e o agente estiver carregado:
        Larga item com probabilidade $P_d$
    Senão se a célula do ambiente estiver cheia e o agente estiver vazio:
        Pega item com probabilidade $P_c$
    Agente se move randomicamente para uma célula adjacente

As funções *probability_drop()* e *probability_catch()* definem a probabilidade de largar ($P_d$) e pegar ($P_c$) um item, respectivamente, e são uma simplificação das funções propostas nos trabalhos relacionados. Elas são calculadas conforme as fórmulas abaixo:

$$P_d = \frac{itens\_vizinhos}{celulas\_vizinhas} \qquad P_c = 1 - \frac{itens\_vizinhos}{celulas\_vizinhas}$$

As células vizinhas são contadas de acordo com o raio de visão do agente e os itens vizinhos dizem respeito à soma de células vizinhas com algum item disposto nelas e são calculados na função *count_neighbors()*. Para impedir um comportamento estocástico no caso de não haver itens vizinhos, em ambas as fórmulas foi adicionado um erro $\epsilon$ de $\pm 0.01$.

A função *ants_behaviour(anthill)* é executada até o tempo máximo passado por parâmetro ser atingido. Nesse momento, realizam-se quantas interações forem necessárias até que os agentes que ainda estejam carregando algum item cheguem em alguma célula vazia, fazendo com que eles larguem esse item nessas células.

As medidas de desempenho implementadas no código são: o número de passos (a soma de vezes que cada agente movimenta-se para outra célula) e o número de itens isolados (a soma de todos os itens dispostos no ambiente que ao final da execução não possuem nenhum item vizinho com raio 1). A primeira medida tem seu total incrementado cada vez que a função *move()* é realizada, já a última medida é contada ao final da execução do programa, na função *count_isolated_dead_ants()*. Nota-se que esses fatores foram escolhidos como medidas de desempenho pois permitem a visualização do quão agrupados os itens estão e qual é o esforço necessário para chegar em cada cenário.

## 3. Descrição das Simulações e Resultados Obtidos

Como forma de analisar a solução desenvolvida, foram realizadas simulações com diferentes valores nos parâmetros de tempo e raio, bem como testado o desempenho do programa para geração de imagens. Foram realizadas três execuções para os diferentes valores dos parâmetros (tempo dado por um e cinco minutos, raio dado por um e cinco valores de unidade), bem como examinado a formação de clusters visualmente, o número de passos e o número de corpos isolados.

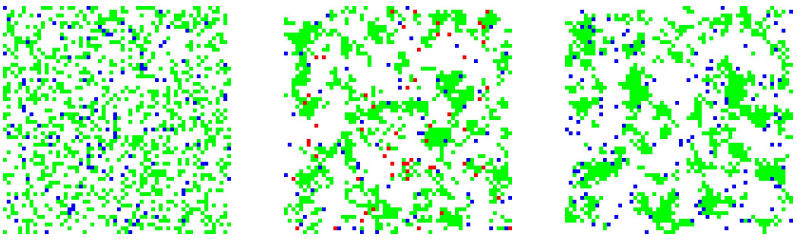
Em relação à formação de imagens, o algoritmo foi testado com geração contínua de imagens e geração momentânea (estado inicial do ambiente, ambiente na metade do tempo de execução, e fim do tempo de execução), onde, devido à alta necessidade de processamento, a geração contínua de imagens obteve um desempenho inferior. Isto é, quando testados com os mesmos parâmetros, os resultados apresentados pelo programa com geração momentânea de imagens era composto pela formação de mais clusters. Desta forma, todos os testes realizados utilizaram a formação momentânea de imagens, onde cada imagem possui a seguinte estrutura:

- Quadrado bidimensional verde: corpos de formigas mortas;
- Quadrado bidimensional azul: formigas vivas que não estão carregando um item;
- Quadrado bidimensional vermelho: formigas vivas que estão carregando um item;

| Execução | Tempo (min) | Raio | Passos | Formigas Mortas Isoladas |
|---|---|---|---|---|
| 1ª | 1 | 1 | 466.043 | 24 |
| 2ª | 1 | 1 | 466.297 | 20 |
| 3ª | 1 | 1 | 466.156 | 30 |
| 1ª | 1 | 5 | 364.524 | 116 |
| 2ª | 1 | 5 | 374.028 | 98 |
| 3ª | 1 | 5 | 386.163 | 88 |

**Tabela 1. Resultados das execuções para o tempo de um minuto. Fonte: as autoras, 2019.**

Um dos testes realizados foi adotando o valor de um minuto para o tempo e os valores um e cinco para o raio, conforme mostrado na Tabela 1. O resultado do agrupamento é mostrado na Figura 3 para raio igual à um e na Figura 4 para raio igual à cinco.

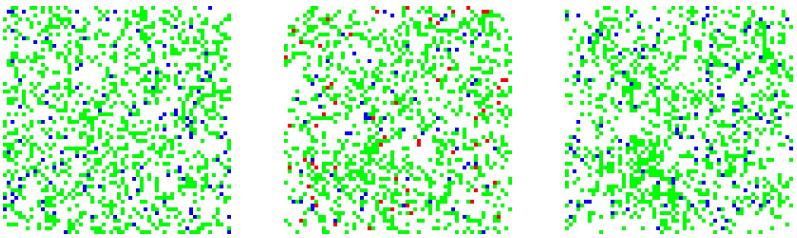


(a) Estado inicial do ambiente

(b) Ambiente na metade do tempo de execução

(c) Fim do tempo de execução

**Figura 3. Execução com tempo de um minuto e raio com um valor de unidade. Fonte: as autoras, 2019.**



(a) Estado inicial do ambiente

(b) Ambiente na metade do tempo de execução

(c) Fim do tempo de execução

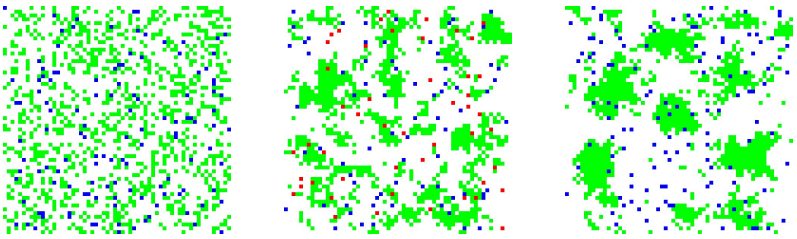
**Figura 4. Execução com tempo de um minuto e raio com cinco valor de unidade. Fonte: as autoras, 2019.**

| Execução | Tempo (min) | Raio | Passos | Formigas Mortas Isoladas |
|---|---|---|---|---|
| 1ª | 5 | 1 | 2.331.958 | 28 |
| 2ª | 5 | 1 | 2.306.570 | 22 |
| 3ª | 5 | 1 | 2.315.413 | 33 |
| 1ª | 5 | 5 | 1.797.268 | 90 |
| 2ª | 5 | 5 | 1.706.528 | 83 |
| 3ª | 5 | 5 | 1.752.903 | 81 |

**Tabela 2. Resultados das execuções para o tempo de cinco minutos. Fonte: as autoras, 2019.**

Igualmente, o experimento feito com o tempo igual à cinco minutos e o raio variando entre os valores um e cinco unidades de medida, são mostrados na Tabela 2. As imagens momentâneas geradas para a execução são mostradas na Figura 5 e Figura 6 para os raios um e cinco, respectivamente.

Como forma de testar a suficiência dos tempos anteriores para o agrupamento de itens, também foi realizado um teste de oito minutos com raio igual à um. Esta execução é mostrada na Figura 7. O número de passos foi igual à 3.718.443 e o número de formigas mortas isoladas foi igual à 20.

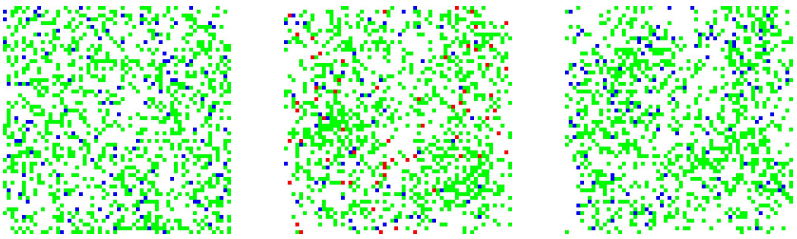


(a) Estado inicial do ambiente

(b) Ambiente na metade do tempo de execução

(c) Fim do tempo de execução

**Figura 5. Execução com tempo de cinto minutos e raio com um valor de unidade. Fonte: as autoras, 2019.**

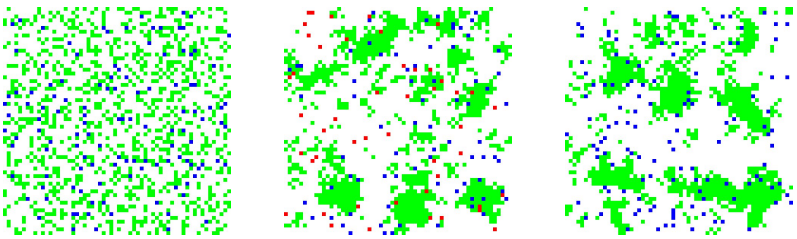


(a) Estado inicial do ambiente

(b) Ambiente na metade do tempo de execução

(c) Fim do tempo de execução

**Figura 6. Execução com tempo de cinco minutos e raio com cinco valores de unidade. Fonte: as autoras, 2019.**



(a) Estado inicial do ambiente

(b) Ambiente na metade do tempo de execução

(c) Fim do tempo de execução

**Figura 7. Execução com tempo de oito minutos e raio com um valor de unidade. Fonte: as autoras, 2019.**

## 4. Análise dos Resultados Obtidos

Analisando os resultados gerados, pode-se entender que as simulações feitas com tempo igual à um minuto foram suficientes apenas para a formação de pequenos e dispersos clusters. Isto está atrelado a linguagem de programação, o desempenho do computador utilizado e as formas como os cálculos estão sendo realizados no algoritmo. Nota-se também que a execução com tempo de oito minutos não sobressaiu a execução de cinco minutos. Desta forma, infere-se que o tempo pertinente para a execução com formação de clusters mais definida seja de cinco minutos.

Considerando o cenário dos cálculos dados por tempos iguais e raios diferentes, compreende-se que o agrupamento fica mais definido com raio igual à um e menos definido raio igual à cinco. Isto acontece porque, quanto maior o raio, maior é o tempo demandado para o cálculo da quantidade de vizinhos, visto que precisa percorrer mais células da matriz. Com isto, têm-se também que o número de passos é menor, gerando menos agrupamento de itens e um maior número de formigas mortas isoladas, como mostrado no gráfico da Figura 8. Os valores de formigas mortas isoladas foi feito a partir da média das três execuções. Para que o agrupamento com raios diferentes fossem semelhantes, seria necessário colocar um tempo maior para a execução com o maior raio e um tempo menor para a execução com menor raio.

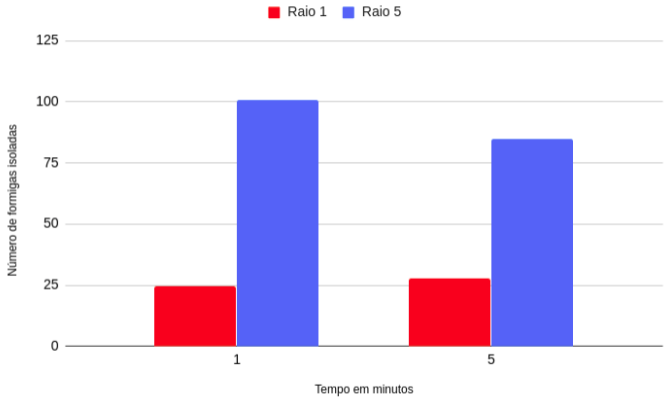



**Figura 8. Gráfico relacionando o número de formigas isoladas pelos diferentes valores de tempos e raios. Fonte: as autoras, 2019.**

Em relação à formação dos clusters, entende-se que um dos motivos para que os itens não tenham sido agrupado em um único conjunto é dado pela forma como o ambiente foi feito, visto que não utilizou-se matriz circular, gerando bordas. Em compensação, não existem buracos no meio dos clusters, isto é, não há células no meio dos clusters sem corpos de formigas mortas. O último item a ser analisado é que a quantidade de clusters formados e a posição em que são formados é diferente em cada execução pois o ambiente é estocástico, sequencial e dinâmico.

## 5. Conclusões e Trabalhos Futuros

O presente trabalho teve como objetivo o estudo e a implementação de um algoritmo de agrupamento de dados baseado em colônias de formigas. A partir do seu desenvolvimento, foi possível realizar uma análise do método implementado e de seu comportamento diante de mudanças no raio de visão dos agentes do sistema e do tempo de execução.

Conclui-se a eficácia do algoritmo implementado quando o raio de visão dos agentes é de uma unidade. Contudo, ao aumentar esse raio, o programa não se comporta de forma a agrupar esses itens em tempo hábil. Um dos fatores para que isso ocorra é o fato dos cálculos necessários para determinar a probabilidades de pegar e largar itens aumentar conforme o aumento do raio de visão dos agentes. Nesse sentido, identificam-se como trabalhos futuros a otimização do código para que se tenha uma eficiência maior em relação a diferentes raios de visão dos agentes e para que se possa trabalhar com itens que possuam mais de uma dimensão.

## Referências


Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm intelligence: From natural to artificial systems. Oxford University Press.

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chrétien, L. (1990). The dynamics of collective sorting robot-like ants and ant-like robots. In *International Conference on Simulation of Adaptive Behavior*, pages 356–363.

Dorigo, M. and Stutzle, T. (204). *Ant Colony Optimization*. A Bradford book, 1ª edition.

Ekola, T., Laurikkala, M., Lehto, T., and Koivisto, H. (2004). Network traffic analysis using clustering ants. In *World Automation Congress (WAC 2004)*.

Espenchitt, D. G., Gomes, J. C., and Ebecken, N. (2008). Agrupamento de dados com algoritmo de colônia de formigas. In *Simpósio de pesquisa operacional e logística da marinha*.

Handl, J. and Meyer, B. (2007). Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95 – 113.

Hoe, K., Lai, W., and Tai, T. (2002). Homogenous ants for web document similarity modeling and categorization. In *International Workshop on Ant Algorithms (ANTS 2002)*, page 256–261.

Jafar, O. M. and Sivakumar, R. (2010). Ant-based clustering algorithms: A brief survey. *International Journal of Computer Theory and Engineering*, 2(5):787 – 796.

Kuntz, P. and Snyers, D. (1999). New results on an ant-based heuristic for highlighting the organization of large graphs. In *Congress on Evolutionary Computation*, page 1451–1458.

Lumer, E. and Faieta, B. (1994). Diversity and adaption in populations of clustering ants. In *International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3*, page 501–508.

TuringFinance (2015). Clustering using Ant Colony Optimization. http://www.turingfinance.com/ant-colony-optimization-finance/. Accessed: 2019-09-01.

Wang, D. and Hao, L. (2009). Application of ant colony clustering in computer forensics. In *International Conference on Information and Computing Science*, pages 87–90.