ALGORITMO DE COLÔNIAS DE FORMIGAS

Disciplina: Sistemas Evolutivos

Professor: Romuere Veloso

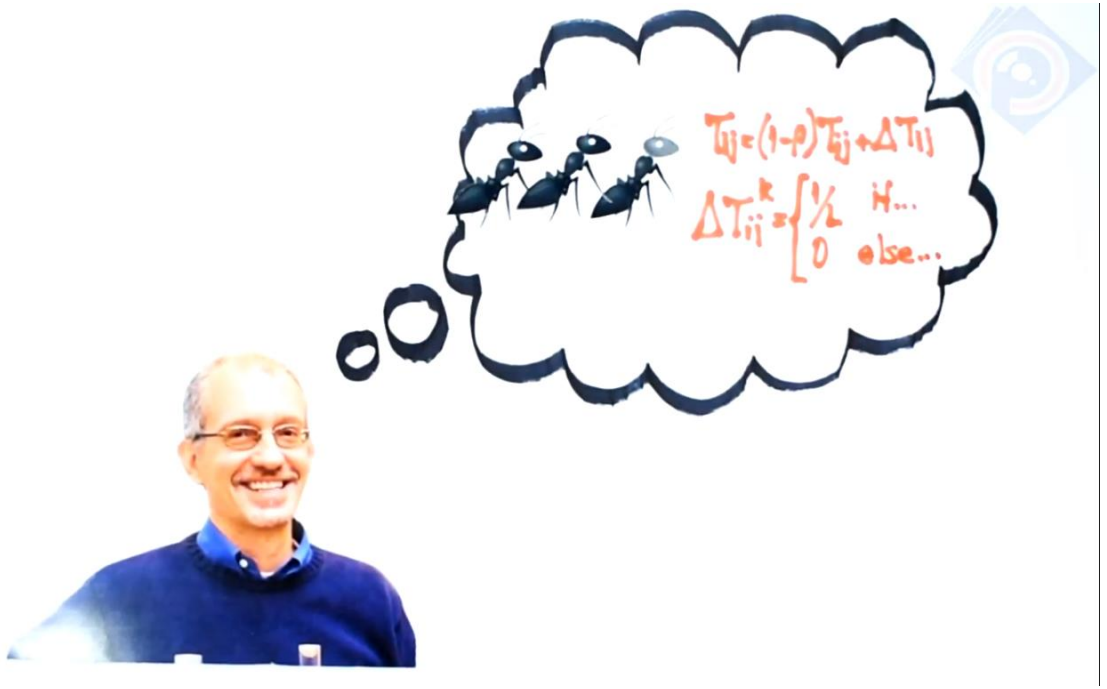
Discentes: Clésio, Lucas e Rodrigo

OBJETIVO

- ENTENDER OS PRECEITOS BÁSICOS DO ALGORITIMO DE COLÔNIA DE FORMIGAS
- COMPREENDER A IMPLEMENTAÇÃO MATEMÁTICA DO ALGORITMO DE COLÔNIA FORMIGAS
- APRESENTAR PROBLEMAS APLICADOS A ESTE ALGORITMO
- APRESENTAR IMPLEMENTAÇÃO DESSE ALGORITMO EM CÓDIGO PYTHON

INTRODUÇÃO

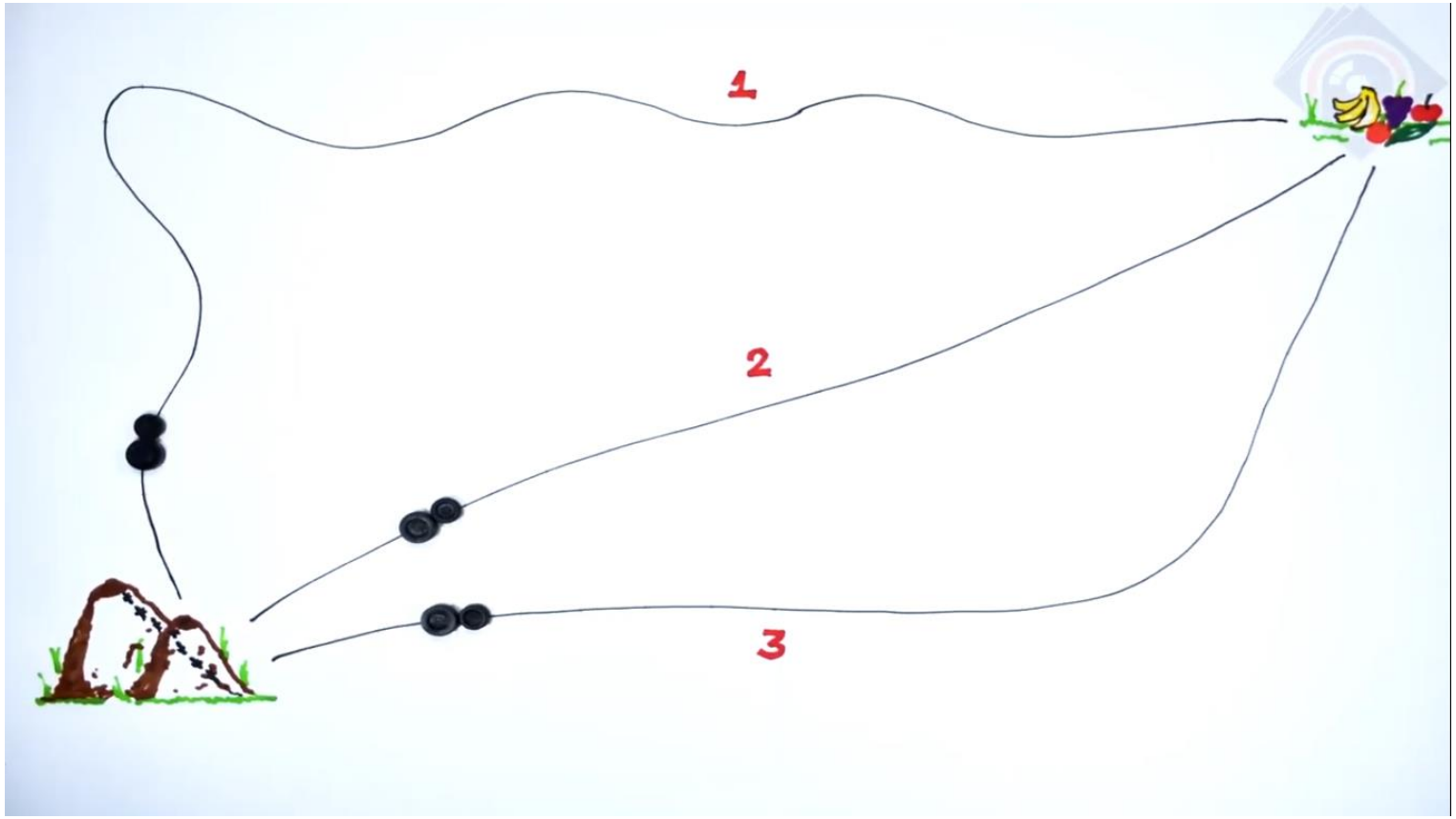
- O algoritmo da otimização da colônia de formigas (ACO, do inglês ant colony optimization algorithm), introduzido por Marco Dorigo em sua tese de PhD é uma heurística baseada em probabilidade, criada para solução de problemas computacionais que envolvem procura de caminhos em grafos. Este algoritmo foi inspirado na observação do comportamento das formigas ao saírem de sua colônia para encontrar comida.



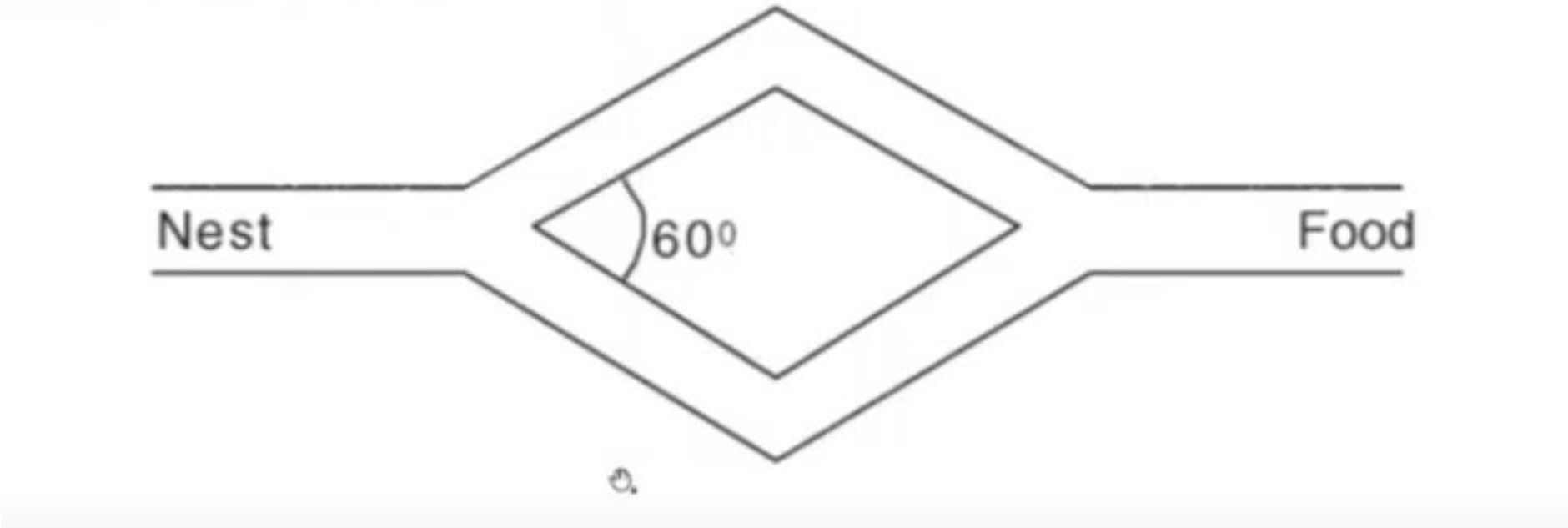
ENTENDO A NATUREZA - COLÔNIA DE FORMIGAS

ESTIGMERGIA

EXPERIMENTO BIOLÓGICO



EXPERIMENTOS COM FORMIGAS

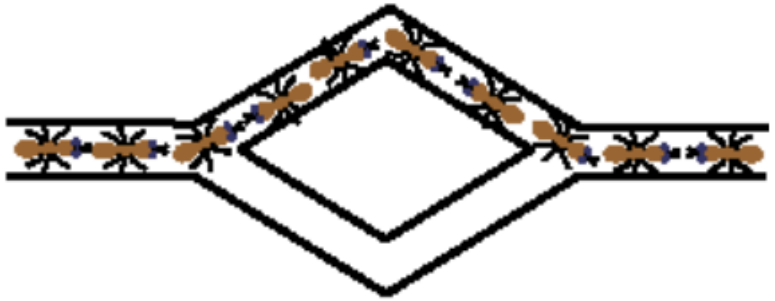


EXPERIMENTOS COM FORMIGAS

- As formigas convergem para um dos caminhos com igual probabilidade.

- Devido a flutuações aleatórias, uma das pontes terá mais feromônio e atrairá as formigas com maior probabilidade.

No Fim:

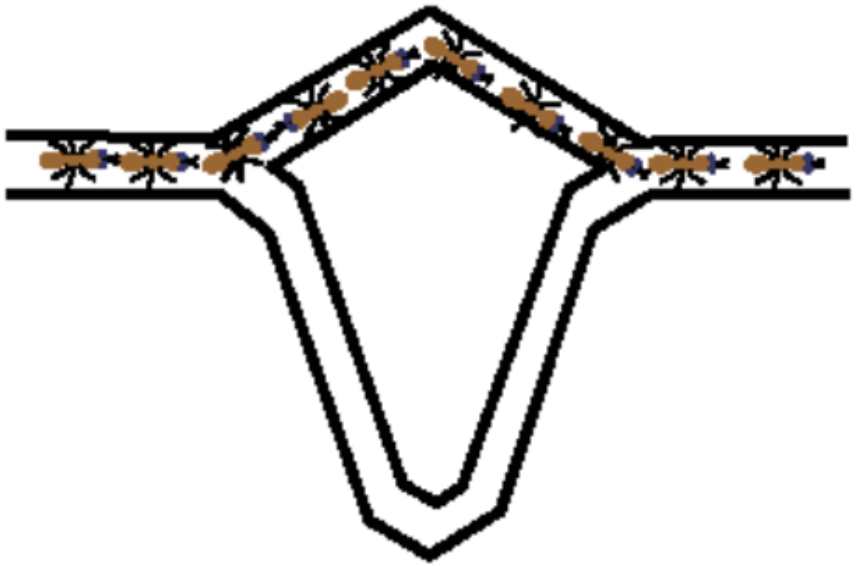


ou

EXPERIMENTOS COM FORMIGAS

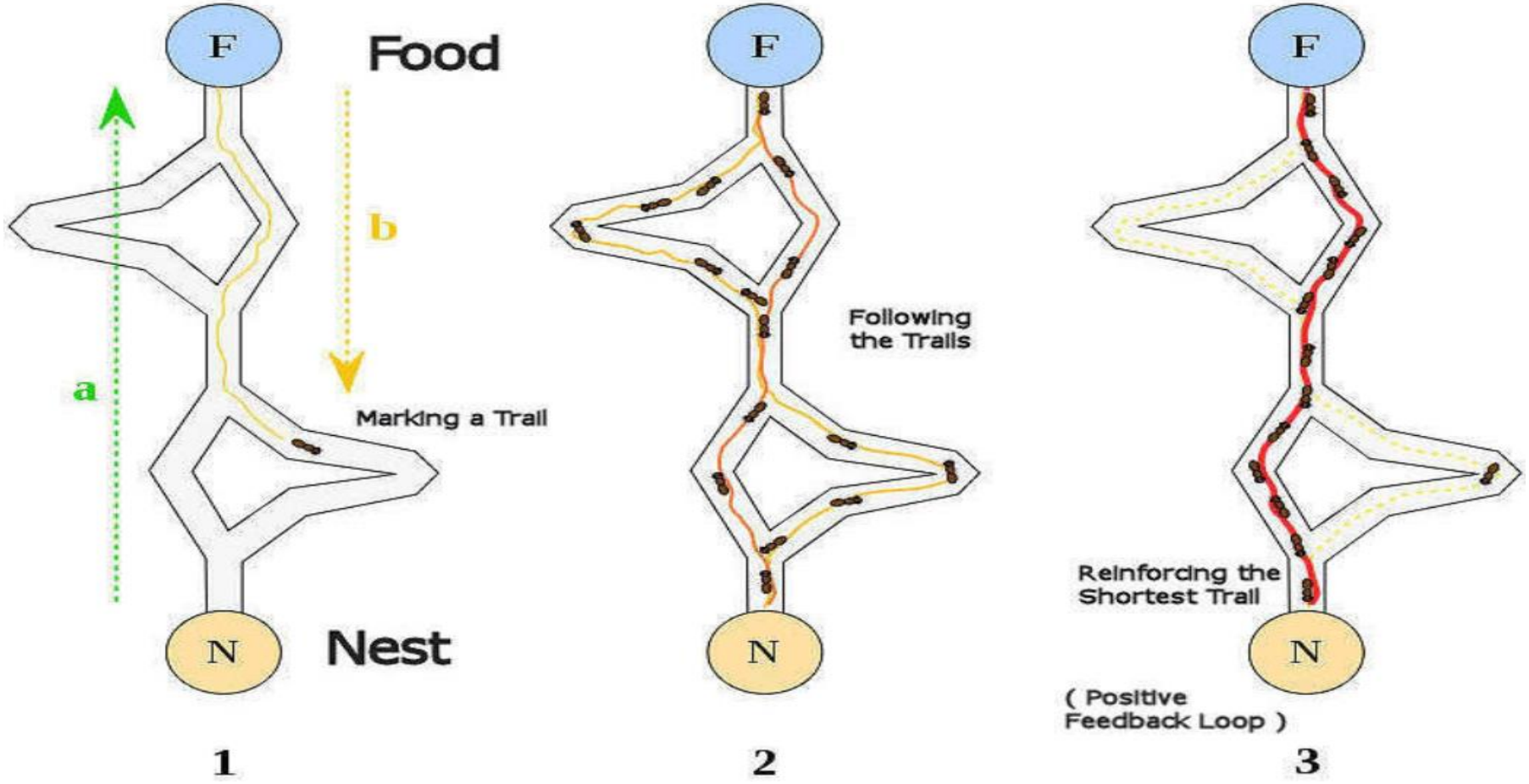
- Usando pontes de tamanhos diferentes, as formigas convergem para a ponte mais curta:

- A ponte curta é percorrida em menos tempo, fazendo com que mais formigas atravessem ela. Logo, mais feromônio é depositado.

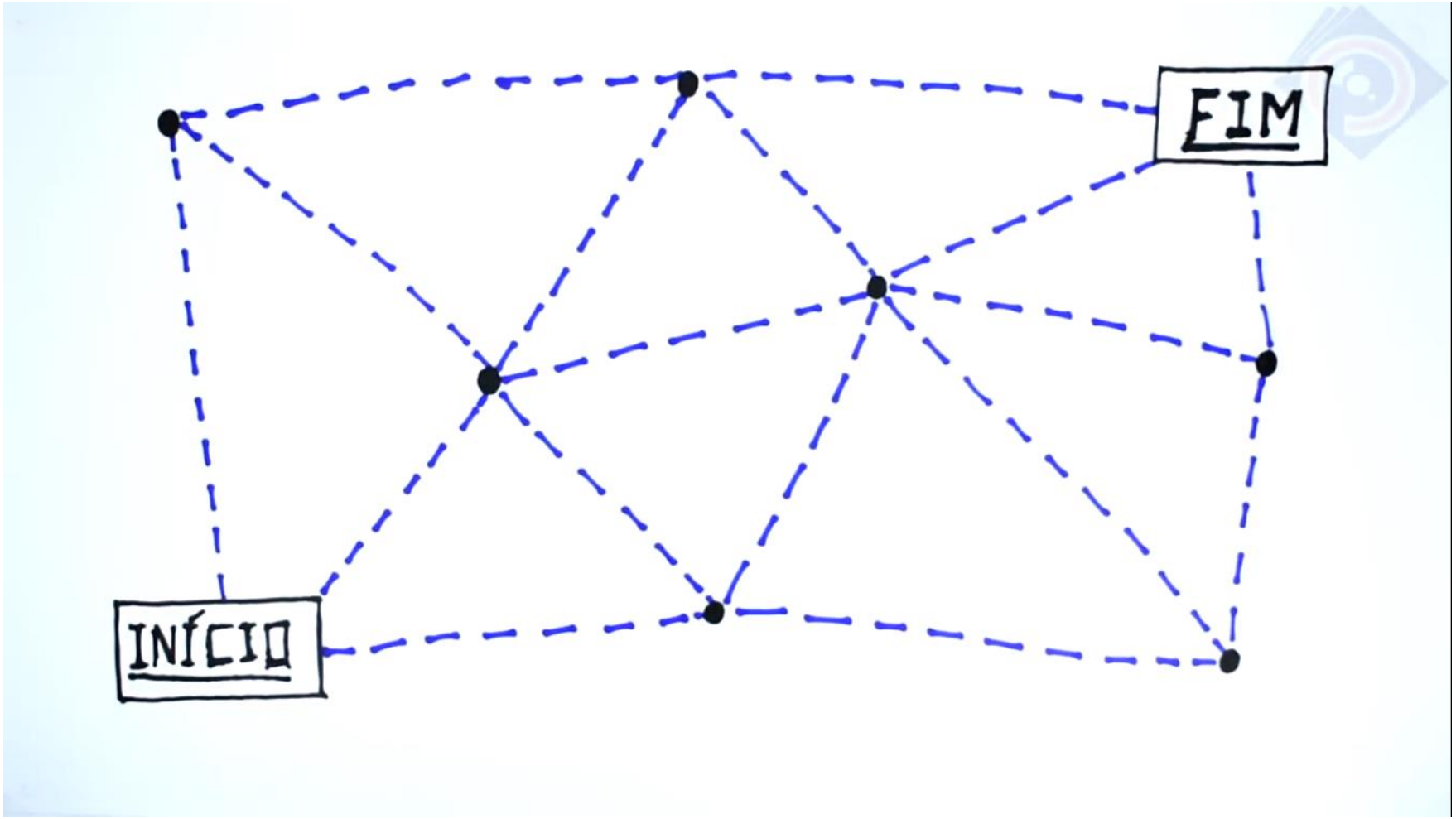


- As formigam escolhem, com maior probabilidade, a ponte curta (com mais feromônio).

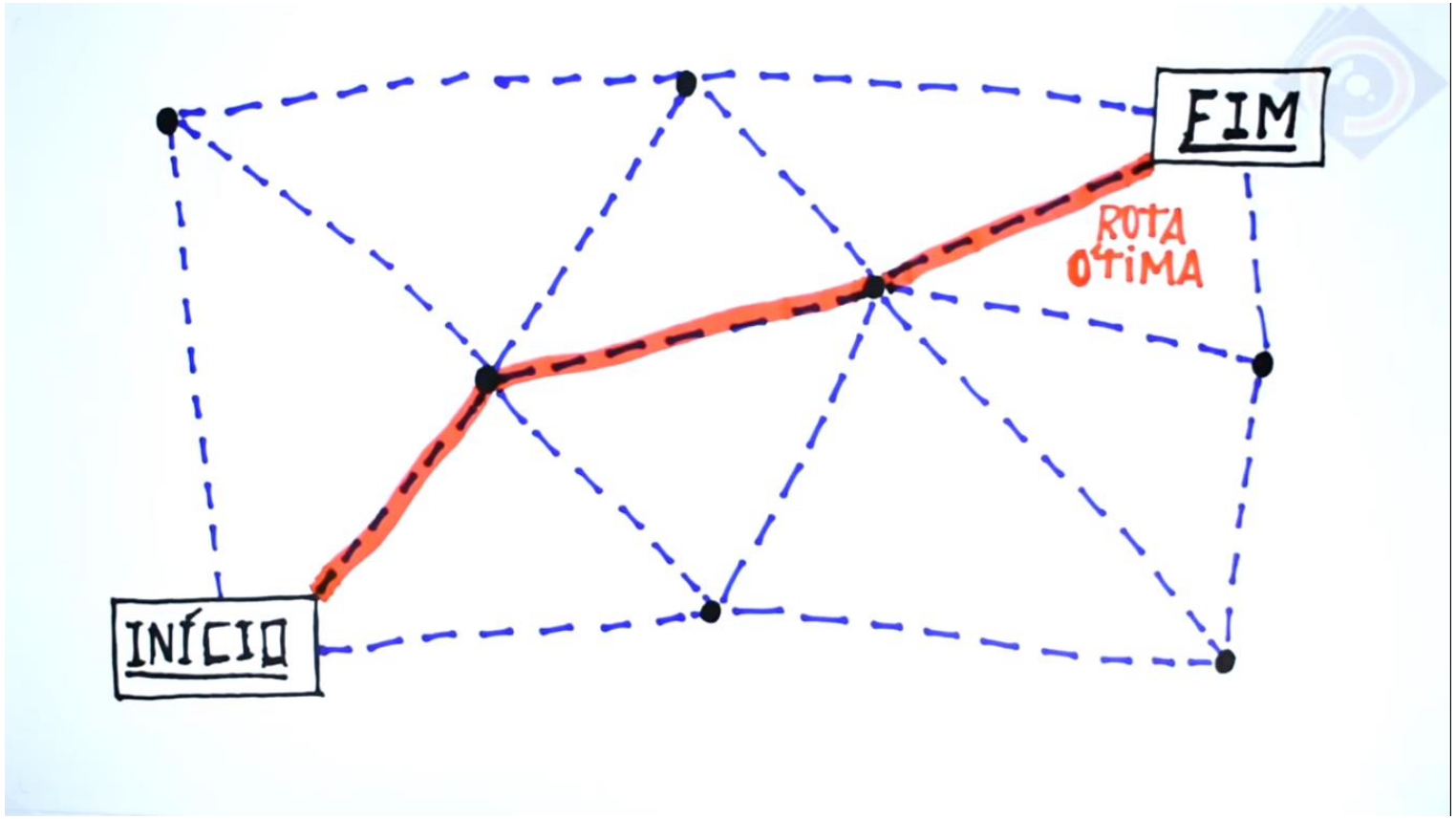
EXPERIMENTOS COM FORMIGAS



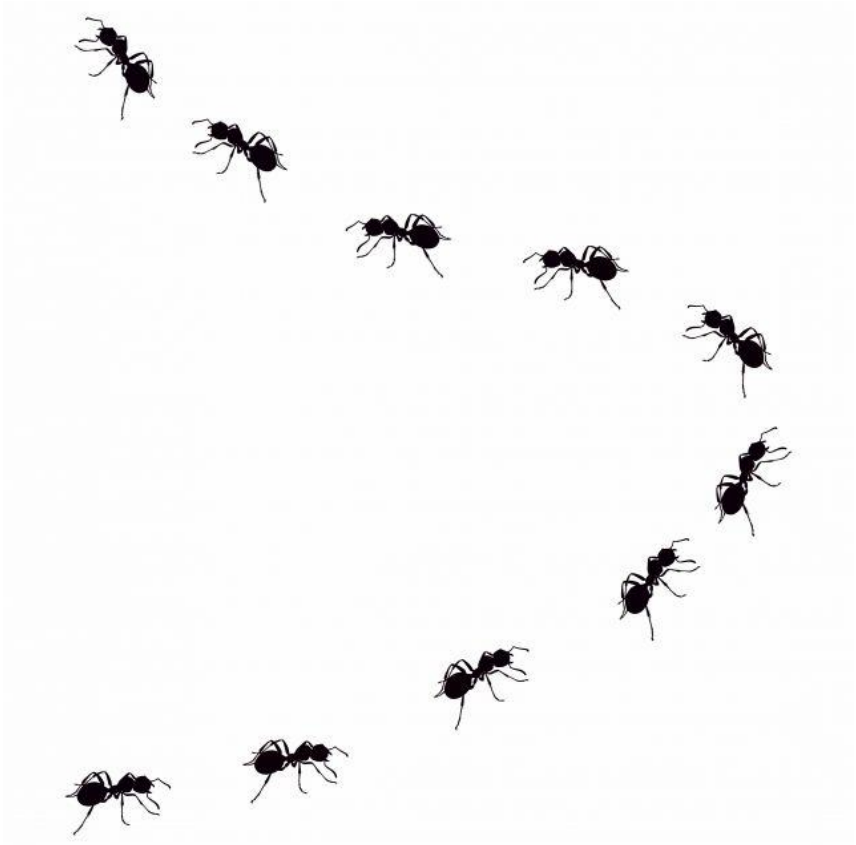
TEORIA DOS GRAFOS

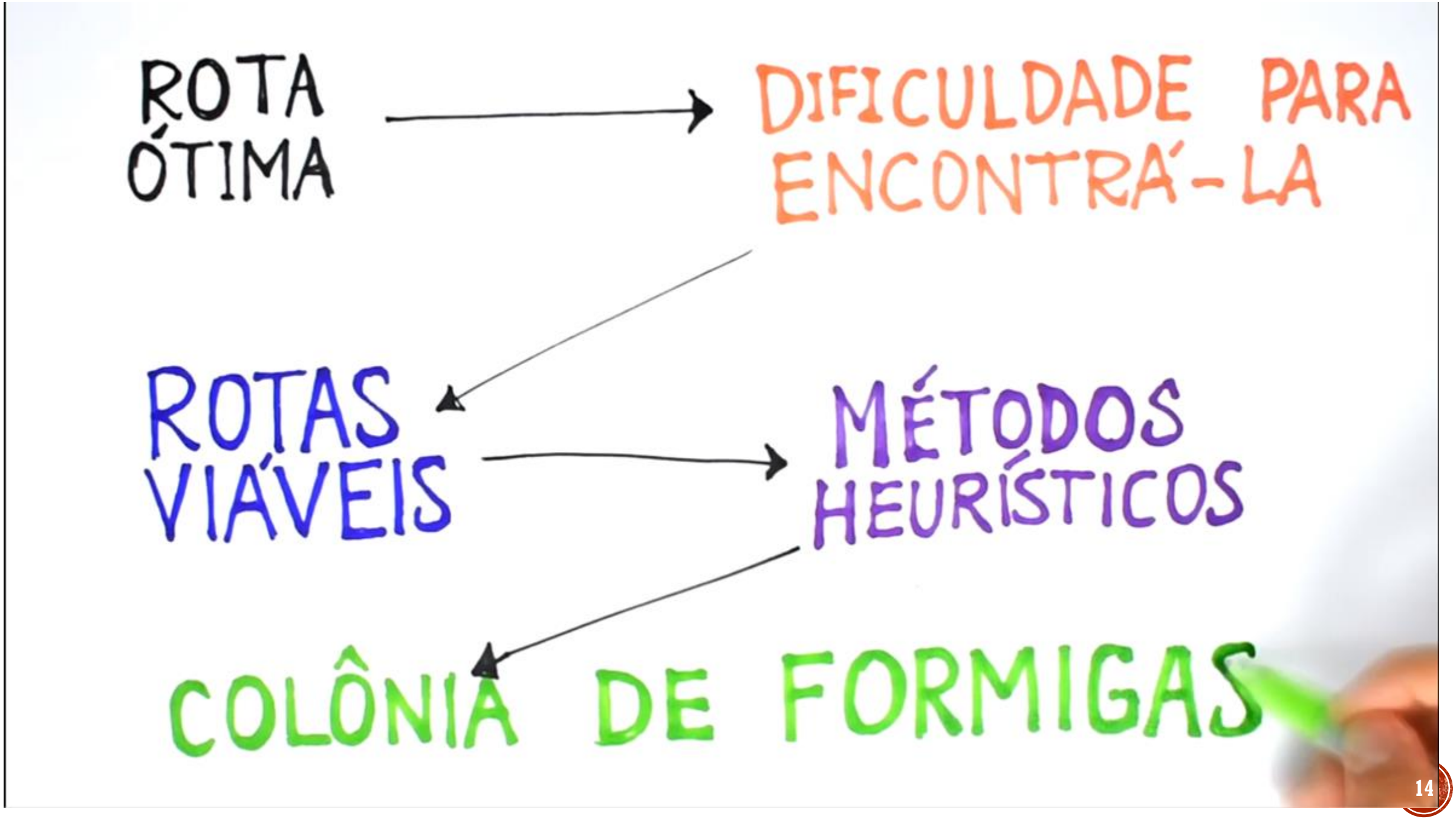


OTIMIZAÇÃO

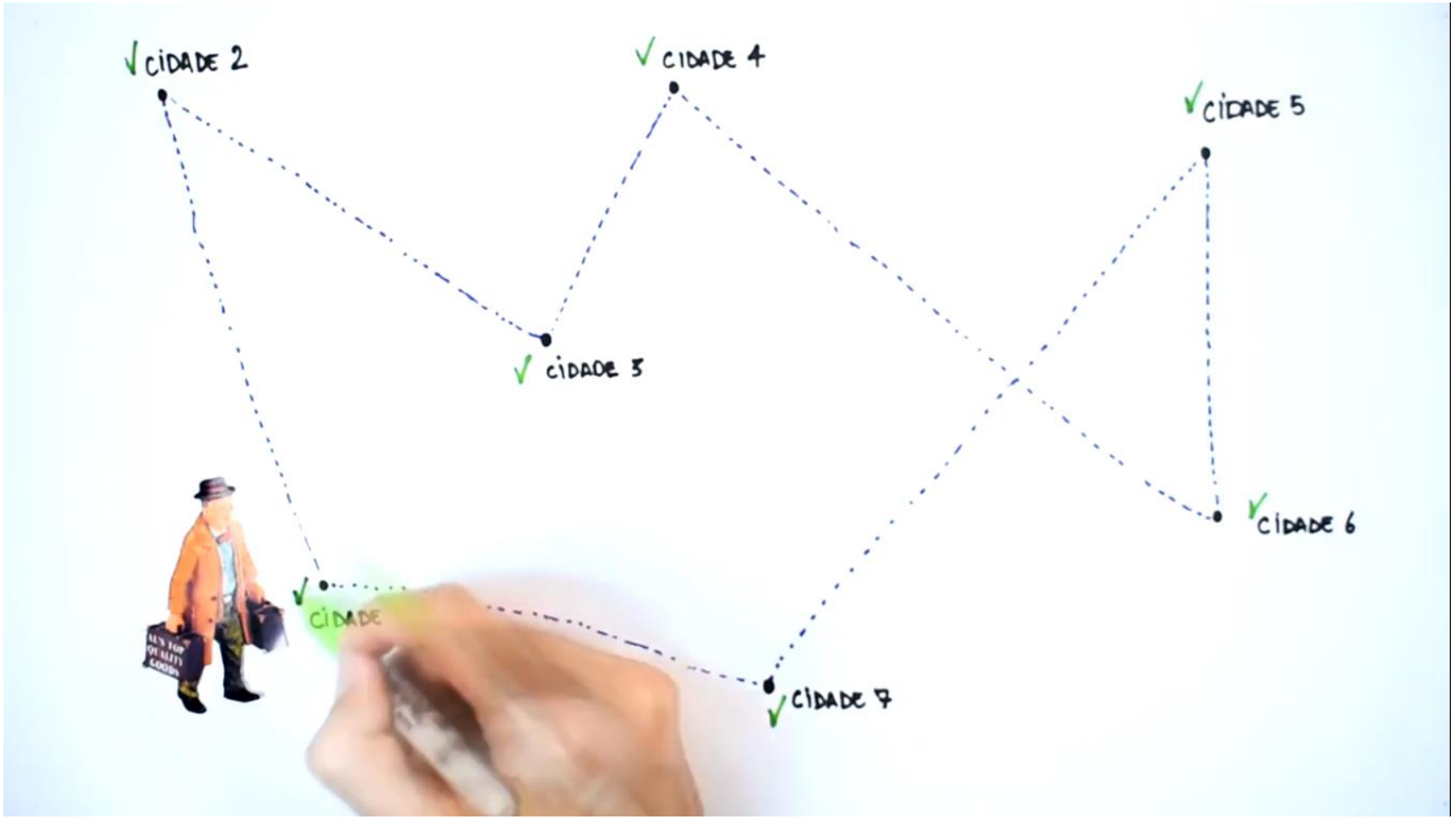


ALGORITMO DE COLÔNIA DE FORMIGAS

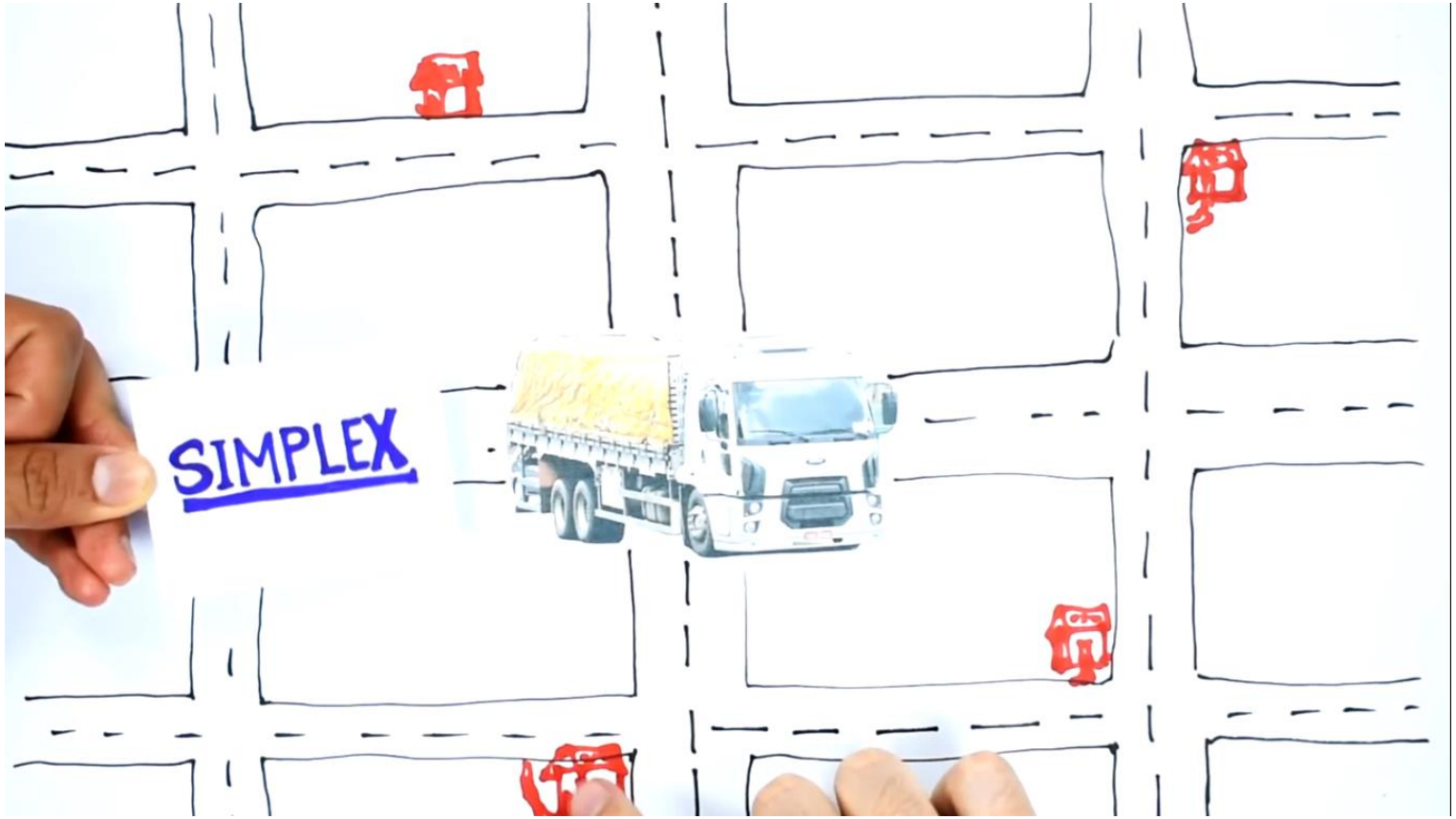


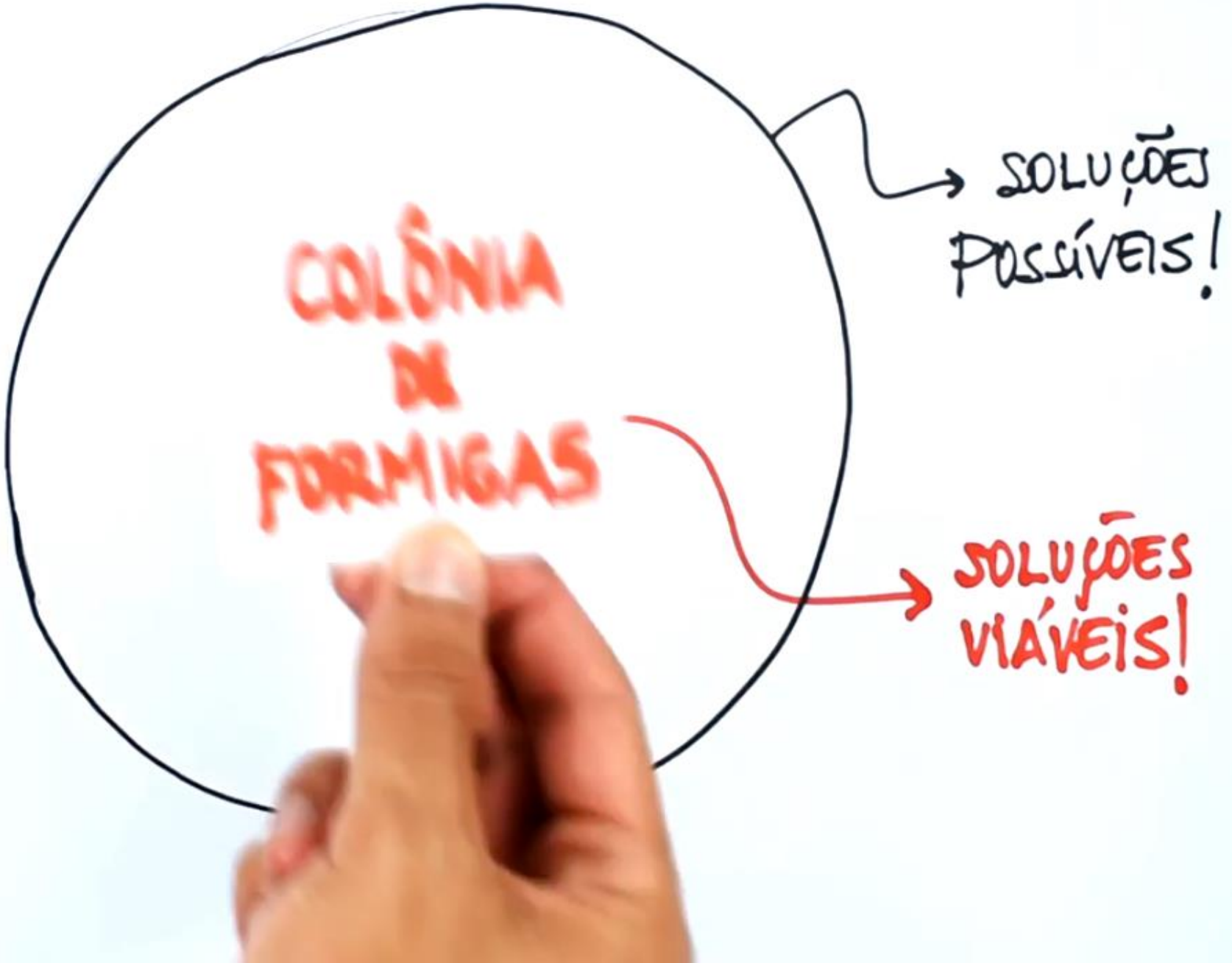

ROTA ÓTIMA
DIFICULDADE PARA ENCONTRÁ-LA
ROTAS VIÁVEIS
MÉTODOS HEURÍSTICOS
COLÔNIA DE FORMIGAS

PROBLEMA DO CAIXEIRO VIAJANTE

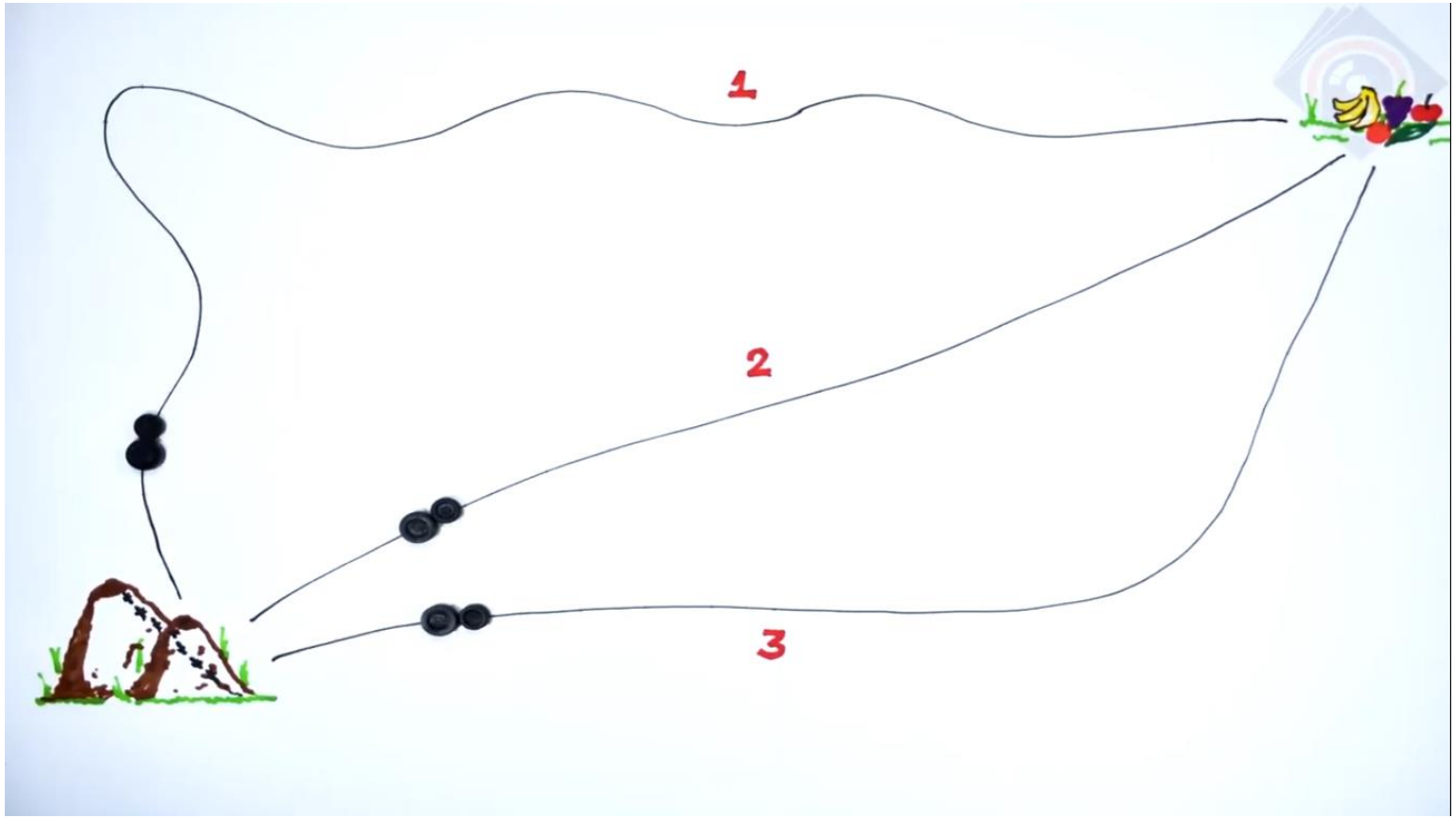


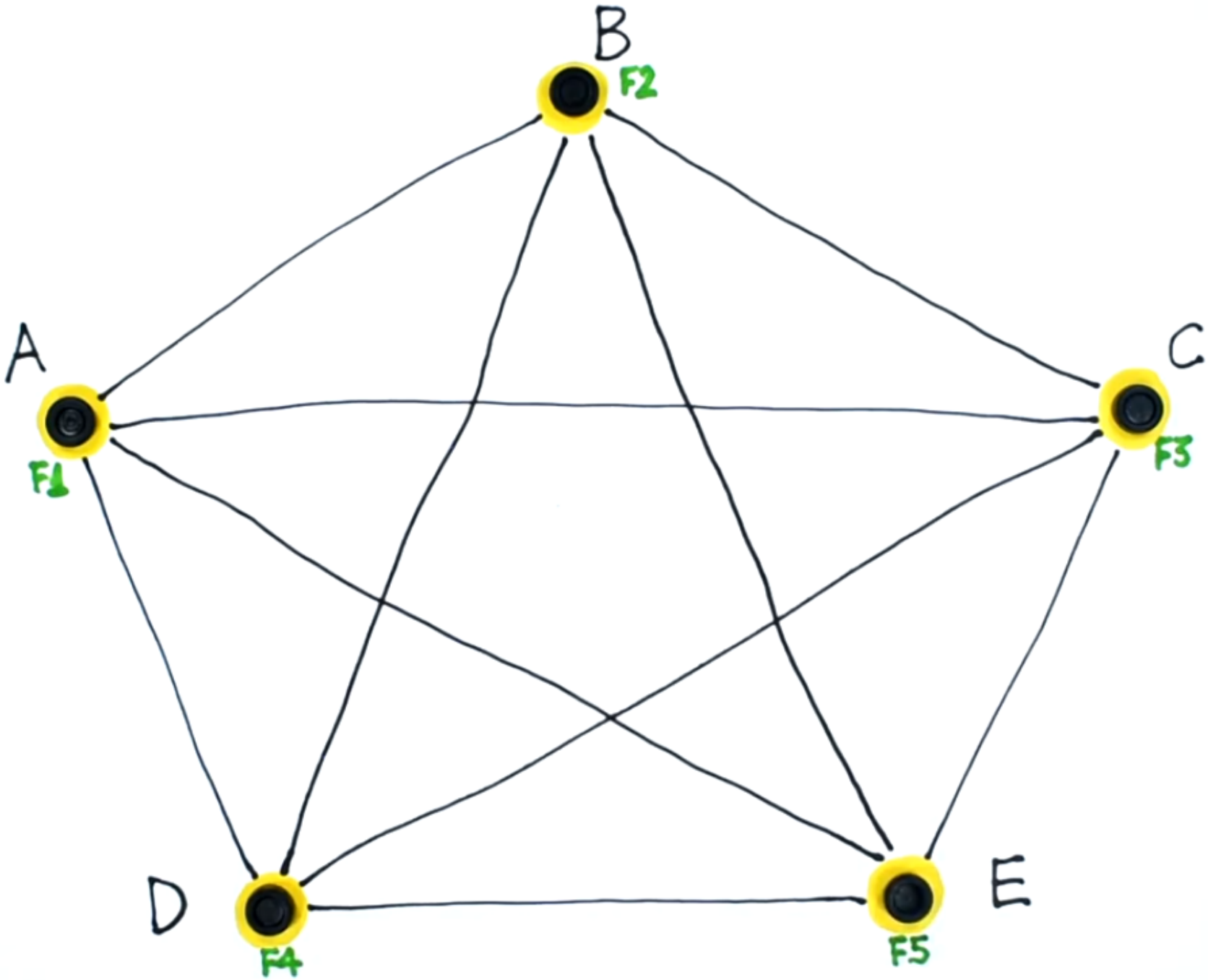
CAIXEIRO VIAJANTE NOS DIAS ATUAIS

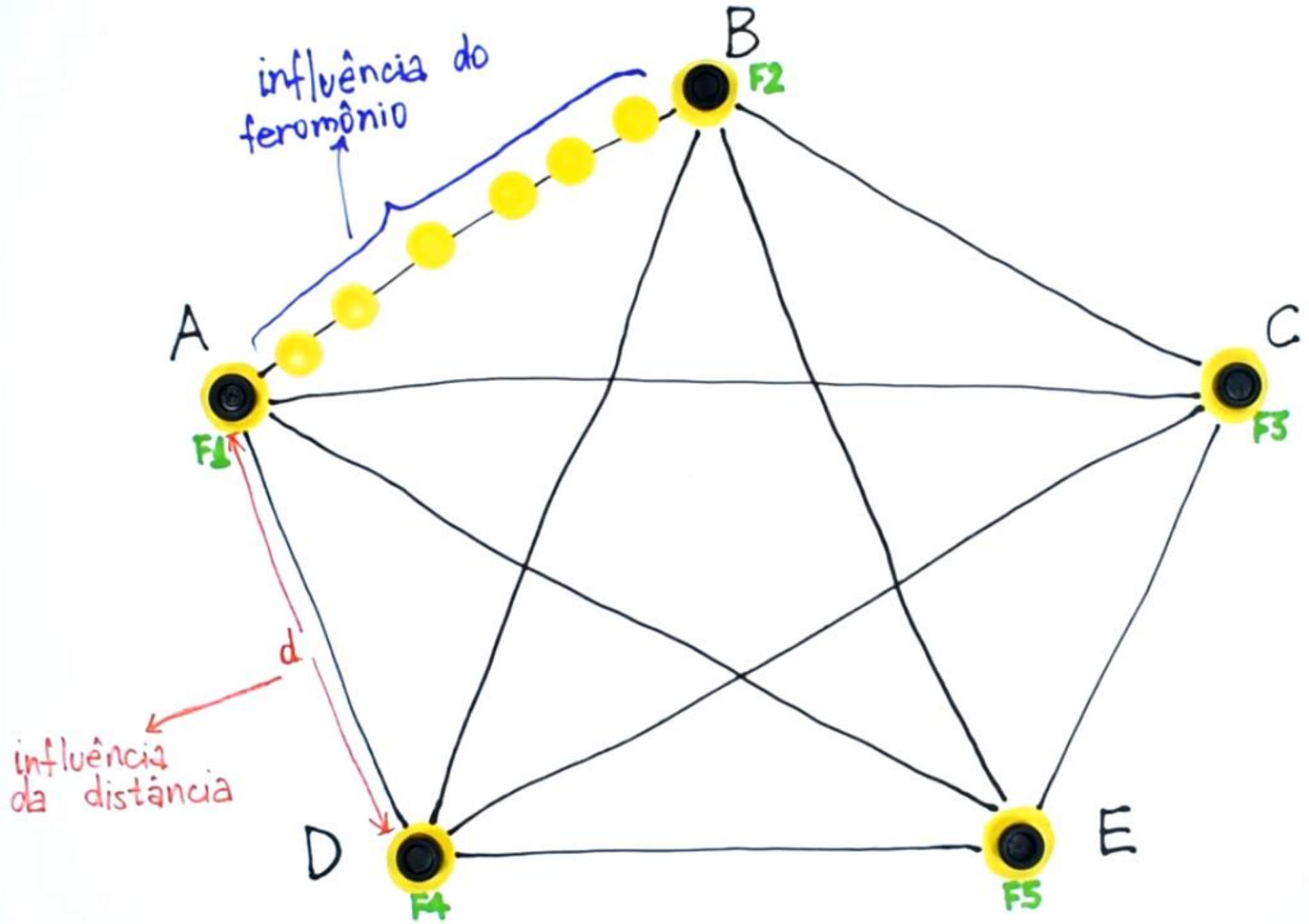


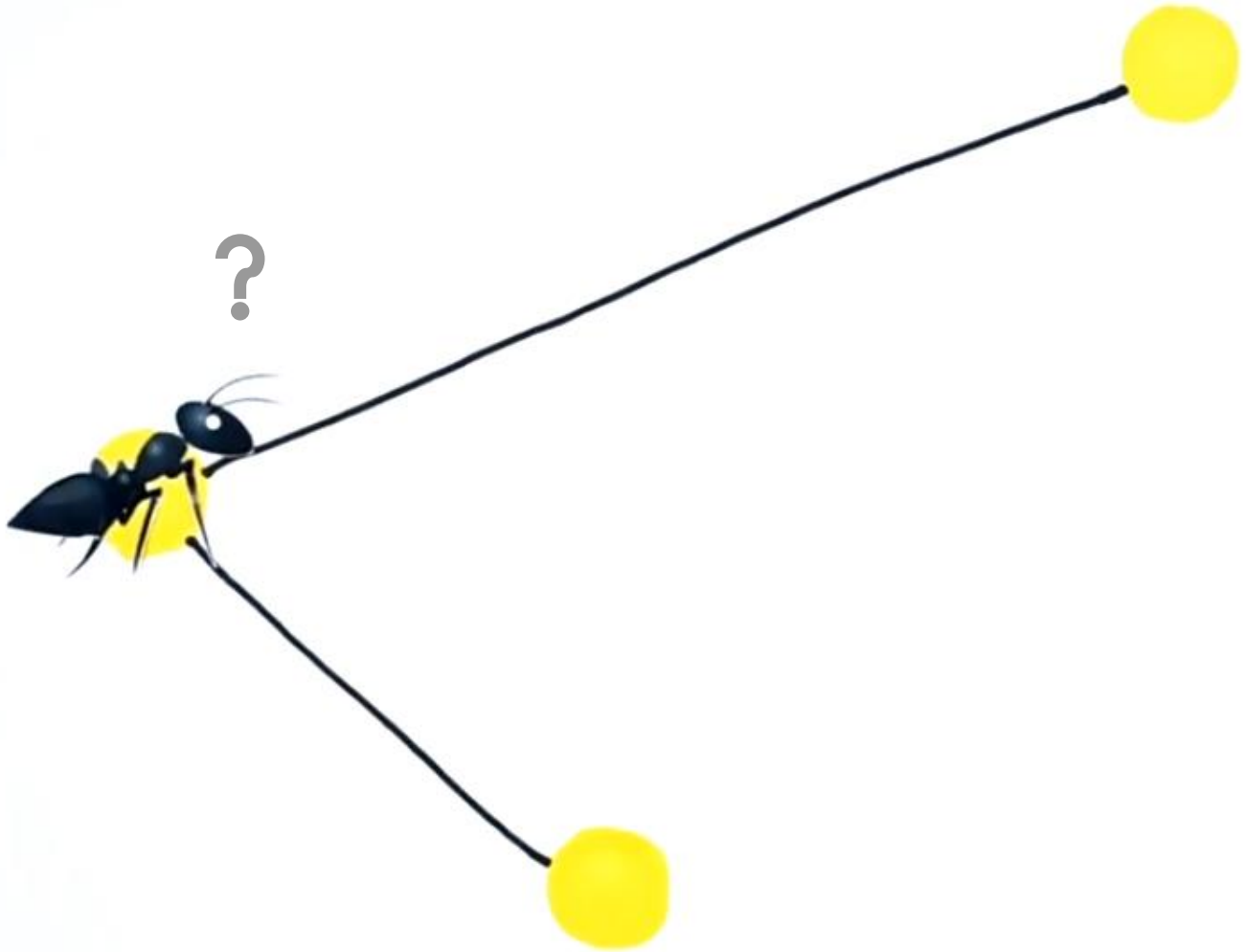

COLÔNIA
DE
FORMIGAS
SOLUÇÕES
POSSÍVEIS!
SOLUÇÕES
VIÁVEIS!

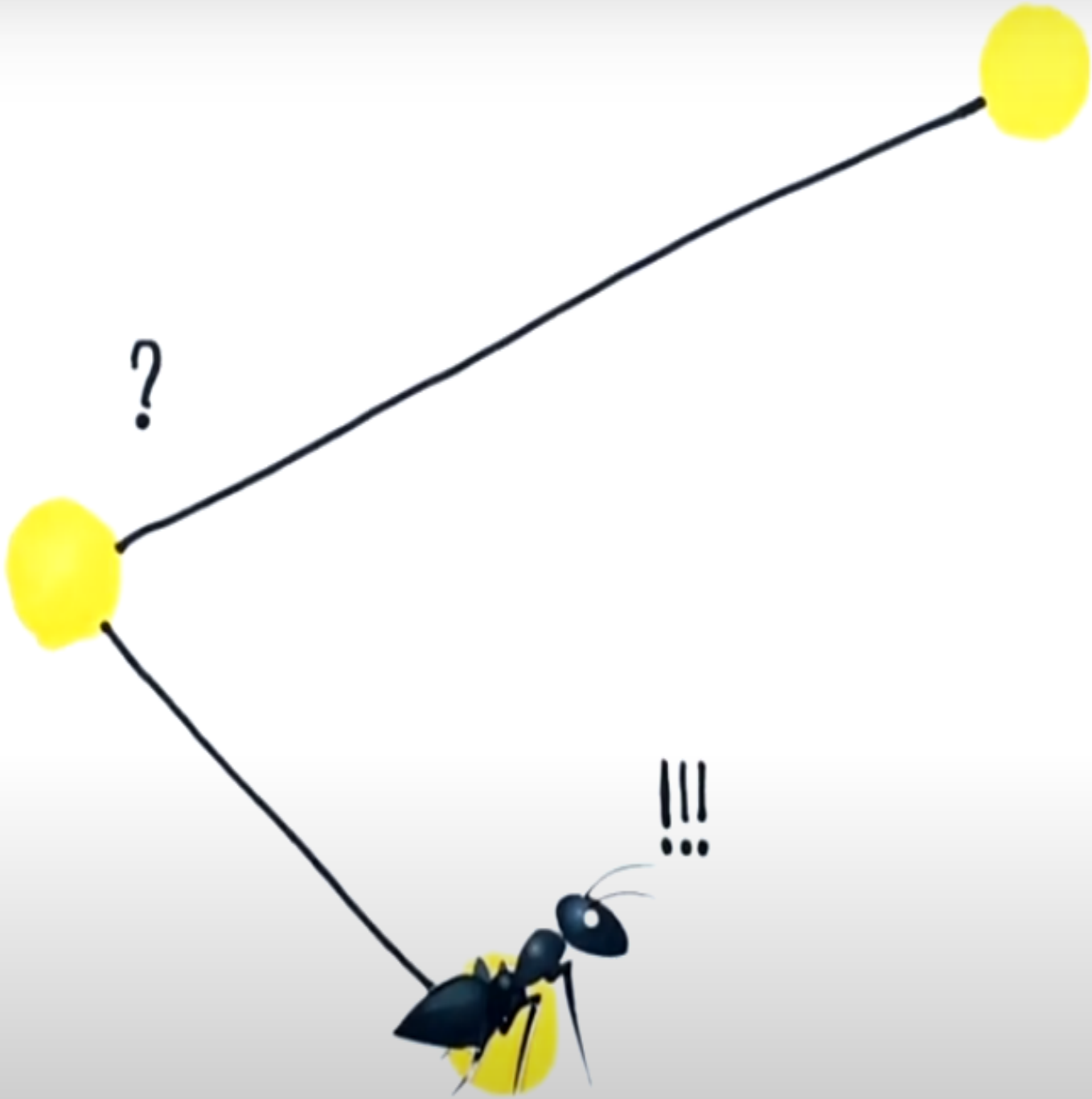
APLICAÇÃO EM PROBLEMA DE OTIMIZAÇÃO

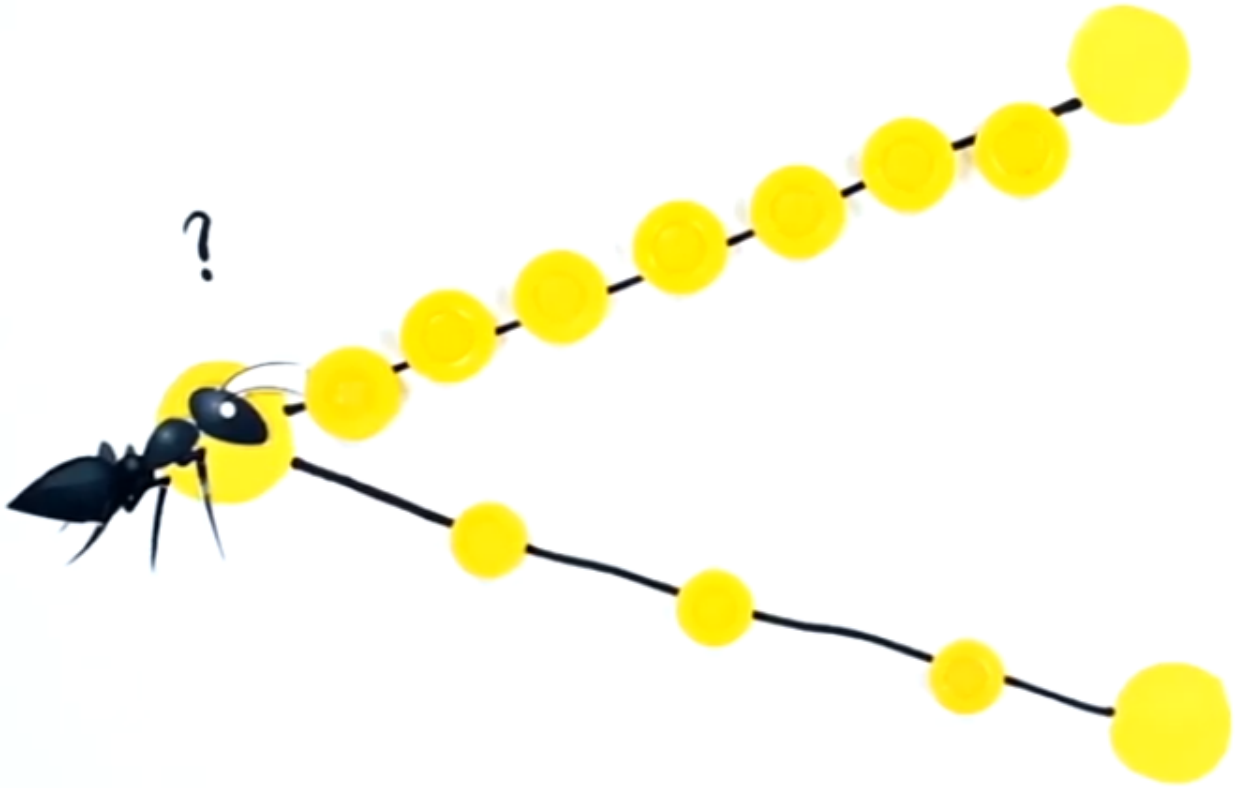


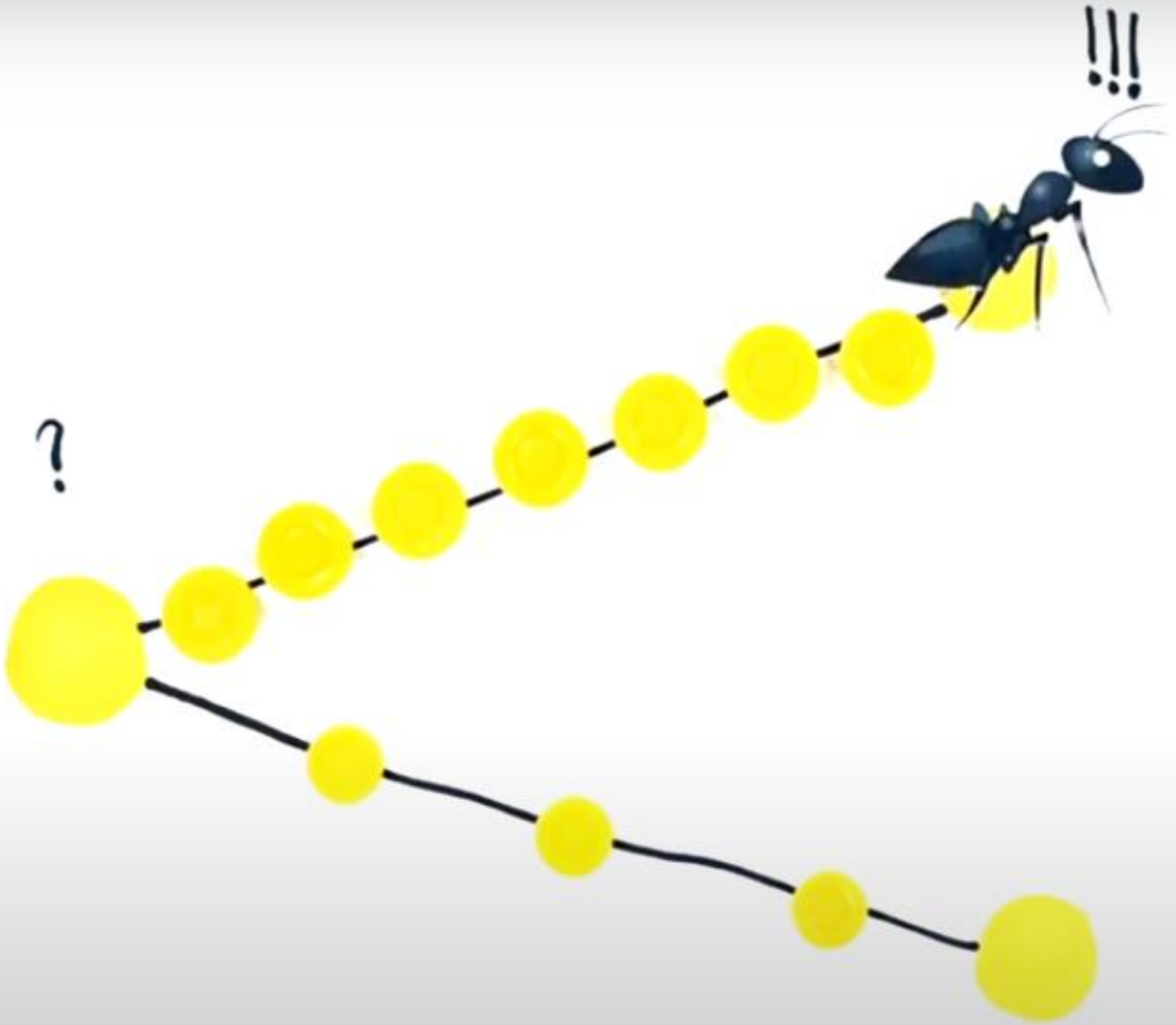

B
F2
A
C
F1
F3
D
E
F4
F5


influência do feromônio
B
F2
A
C
F1
F3
d
influência da distância
D
F4
E
F5









$$p_{xy}^{k} = \frac{[\tau_{xy}(t)]^{\alpha} \cdot [\eta_{xy}(t)]^{\beta}}{\sum_{l \in N_x^k} [\tau_{xy}(t)]^{\alpha} \cdot [\eta_{xy}(t)]^{\beta}}$$

t: iteração

p_{xy}^{k}: probabilidade formiga k: sai de x → chega em y

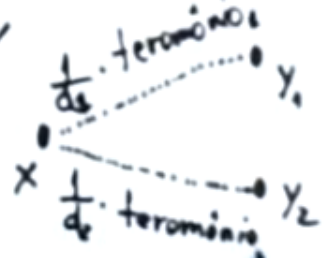
α: parâmetro → feromônio

β: parâmetro → distância

$\tau_{xy}(t)$: x — valor do feromônio — y

$\eta_{xy}(t)$: x — d — y $\Rightarrow \frac{1}{d}$

$\sum_{l \in N_x^k} [\tau_{xy}(t)]^{\alpha} \cdot [\eta_{xy}(t)]^{\beta}$:



$$\Delta\tau_{xy}^{k} = \frac{Q}{d_k}$$

$\Delta\tau_{xy}^{k}$: x — feromônio depositado — y

Q : constante (10)

d_k :  distância total

$$\tau_{xy}^{k}(t) = (1-\sigma)\cdot\tau_{xy}^{k}(t-1) + \sum_{k=1}^{m}\Delta\tau_{xy}^{k}(t)$$

$\tau_{xy}^{k}(t)$: x — feromônio atualizado — y

σ : evaporação do feromônio

$\tau_{xy}^{k}(t-1)$: x — feromônio anterior — y

$\sum_{k=1}^{m}\Delta\tau_{xy}^{k}(t)$: x — $\sum$ feromônio$_1$ feromônio$_2$ feromônio$_3$ — y

Método Heurístico

N - INTERAÇÕES

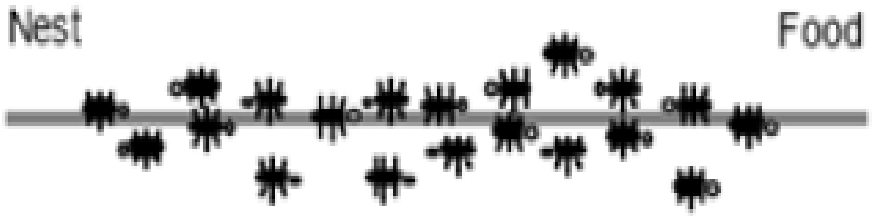
APROXIMAÇÃO DA SOLUÇÃO

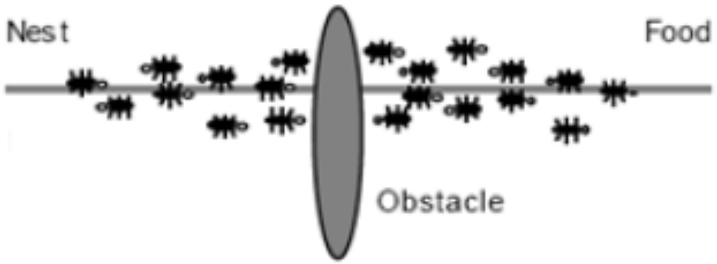
ARTIGO RELACIONADO :MODIFIED ANT COLONY OPTIMIZATION ALGORITHM WITH UNIFORM MUTATION USING SELF ADAPTIVE APPROACH FOR TRAVELLING SALESMAN PROBLEM

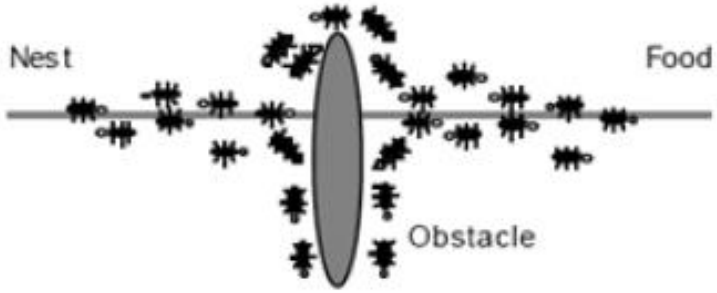
IEEE - 31661

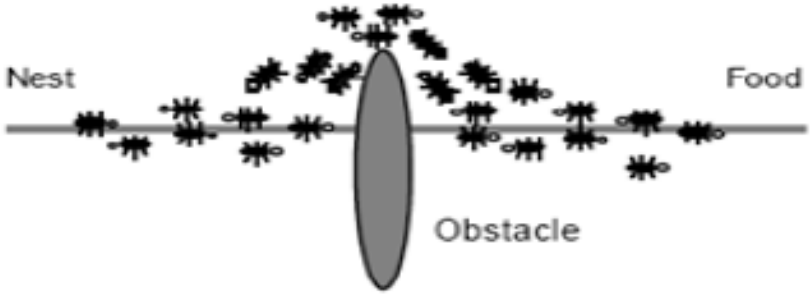
Ramlakhan Singh Jadon, Unmukh Datta,Maharana Pratap College of Technology, Gwalior (M.P.)

- O algoritmo Ant Colony Optimization (ACO) é um novo algoritmo meta-heurístico que tem sido amplamente utilizado para diferentes problemas de otimização combinacional e inspirado no comportamento de forrageamento de colônias de formigas reais.
- Neste trabalho, um algoritmo de otimização de colônia de formigas modificado eficiente com mutação uniforme usando abordagem auto-adaptativa para o problema do caixeiro viajante (TSP) foi proposto. Aqui, o operador de mutação é usado para aumentar o escape do algoritmo dos ótimos locais.


Nest
Food


Nest
Food
Obstacle


Nest
Food
Obstacle


Nest
Food
Obstacle

- O modelo matemático de otimização de colônias de formigas foi aplicado pela primeira vez ao TSP. TSP é descobrir o custo total mínimo dado um conjunto de totalmente conectado gráfico ponderado G (V, E). Aqui, a probabilidade de transição da cidade i para a cidade j para o k-ésima formiga da seguinte forma:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[n_{ij}]^{\beta}}{\sum_{k \in allowed_k}[\tau_{ij}(t)]^{\alpha}[n_{ij}]^{\beta}} & \text{if } j \in \text{allowed} \\ 0 & \text{otherwise} \end{cases}$$

- Na Abordagem Auto adaptativa, os parâmetros são codificados em feromônios e sofrem mutação e recombinação. A ideia é que melhores parâmetros levam a melhores feromônios para encontrar o caminho mais curto.
- O problema do caixeiro viajante (TSP) é um dos problemas de otimização combinacional bem conhecidos e amplamente utilizados. O TSP deve encontrar uma rota de vendedor que comece da localização inicial e visite um conjunto prescrito de cidades e retorne ao local original de forma que a distância total percorrida seja mínima e cada cidade seja visitada exatamente uma vez.

- Neste método proposto, o algoritmo de otimização de colônia de formigas com mutação usando abordagem auto adaptativa é usado. Aqui, a mutação é usada para aumentar o escape do algoritmo dos ótimos locais.

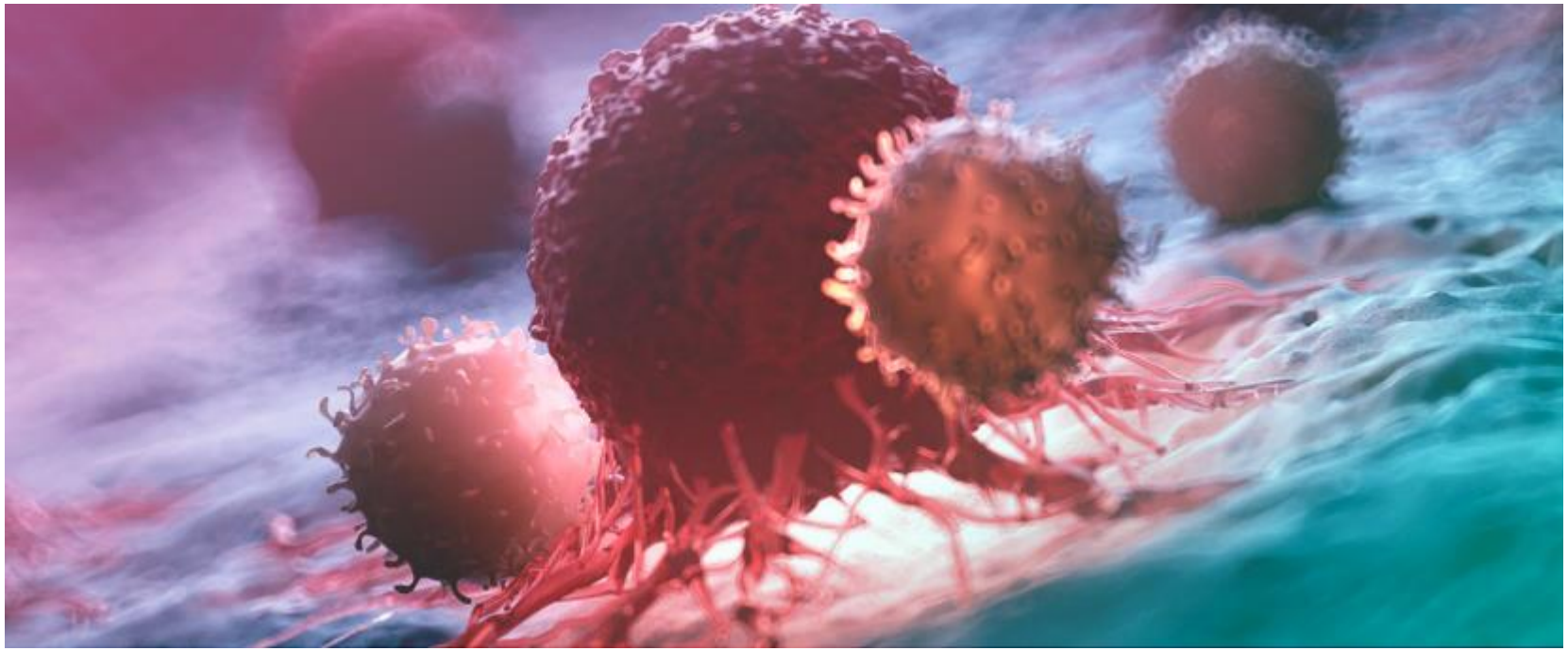
Table 1. Control parameter values

Parameters	Value
ρ	0.1
q_0	0.7
α	1
β	2
Max number of iterations	20 times

TSP	Best length of proposed algorithm	Best length of reference [9]	Best length of reference [10]
eil51	425.23	-	429.98
eil76	534	548.2376	-
bBerlin52	7539	7544.3659	-
st70	672.45	677.1076	677.1096

Algoritmo de Colônia de Formigas e Sistema Imunológico Artificial aplicado ao Problema de Designação Generalizada

Almeida, T.A.; Costa, J.C.; Dias, C.H.; Ferreira, H.M.; Usberti, F.L.

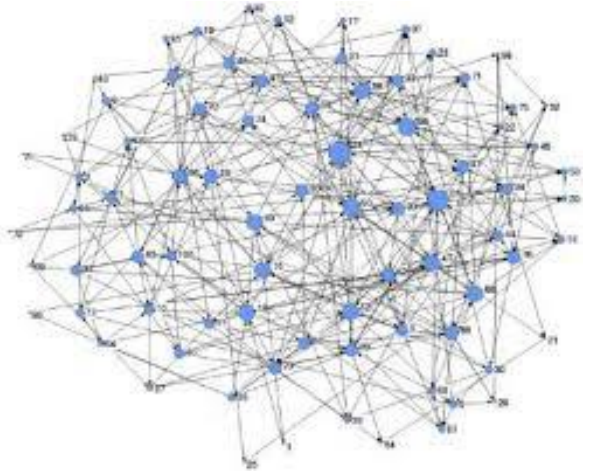


1. INTRODUÇÃO

Esse trabalho tem por objetivo o estudo de técnicas computacionais inspiradas na natureza que tratam de problemas de otimização discreta. Os algoritmos de colônias de formigas (ACO) e sistemas imunológicos artificiais (SIA) são duas dessas técnicas, possuindo uma excelente capacidade de exploração do espaço de soluções de problemas combinatoriais. Muitos desses problemas são de difícil tratamento até a otimalidade, de modo que um acréscimo linear do número de variáveis faz crescer exponencialmente a memória e/ou tempo de processamento necessários para resolver o problema.

O GAP, problema de designação generalizada, foi o problema utilizado nesse trabalho. Trata-se de designar um conjunto de agentes para atender um conjunto de tarefas a um custo mínimo, sem sobrecarregar os agentes além da sua capacidade.

Nesse trabalho foram implementados algoritmos ACO e SIA específicos para o GAP, a partir de onde foram efetuados testes com instâncias da literatura. A análise dos resultados sugere que ambas as técnicas foram excelentes na obtenção de soluções de boa qualidade. Verificou-se que o ACO obteve soluções de boa qualidade em frações de segundo para algumas instâncias. O SIA, por sua vez, conseguiu explorar o espaço de busca, encontrando soluções de boa qualidade e mantendo a diversidade do repertório de soluções obtidas.



SIA – SISTEMA IMUNOLÓGICO ARTIFICIAL

Otimização por colônia de formigas
Ant Colony Optimization – ACO -
Algoritmo de classificação

2. PROBLEMA DE DESIGNAÇÃO GENERALIZADA (GAP)

2.1. Definição

O problema de designação generalizada (generalised assignment problem - GAP) e um problema de otimização combinatorial NP-Dificil muito conhecido na literatura e que envolve encontrar o mínimo custo (ou máximo benefício) de alocar *n* tarefas para *m* agentes de modo que cada trabalho seja designado exatamente a um agente, sujeito à capacidade desse último.

A característica NP-Difícil, inerente ao GAP, torna-o de difícil tratamento até a otimalidade, pois um crescimento linear do número de variáveis do problema (agentes e trabalhos) provoca um acréscimo exponencial nos recursos da máquina utilizada para resolvê-lo, ou seja, tempo de processamento e/ou memória. Problemas dessa natureza ainda são considerados intratáveis pela literatura, ou seja, para instâncias suficientemente grande, não há uma metodologia comprovadamente eficaz para resolvê-los até a solução ótima global. Nesse sentido, os ramos da pesquisa operacional e da inteligência artificial desenvolveram metodologias, denominadas meta-heurísticas, que procuram tratar esses problemas difíceis de natureza combinatorial a partir da exploração "inteligente" do espaço de busca, procurando soluções, quando não ótimas, ao menos de boa qualidade.

2.2. Representação em Grafo

O problema GAP pode ser representado facilmente com uma estrutura de grafo, ou seja, um conjunto de nós (agentes e tarefas) e um conjunto de arestas (as possíveis designações agentes-tarefas). A Figura 1 ilustra essa idéia, onde se pode observar que o conjunto de nós foi particionado em dois subconjuntos, os nós agentes e os nós tarefas, de modo que nós pertencentes ao mesmo subconjunto não apresentam adjacência. Essa característica é a que define um grafo bipartido, muito apropriado para representar um GAP. Para cada aresta do grafo se associam dois pesos distintos: custo (ou benefício) e recurso. Enquanto o custo é parâmetro importante para a função objetivo do problema, o recurso é fundamental para a restrição de capacidade do problema.

Uma solução GAP é considerada infactível quando as tarefas designadas a um (ou mais) agente(s) estão consumindo recursos aquém da capacidade do(s) mesmo(s). Normalmente, o número de soluções infactíveis de um GAP é de ordem muito superior ao número de soluções factíveis, de modo que os algoritmos que procuram tratar esse problema dispensam grande parte do tempo computacional para determinar uma solução factível para posterior melhoria de sua qualidade (redução do custo).

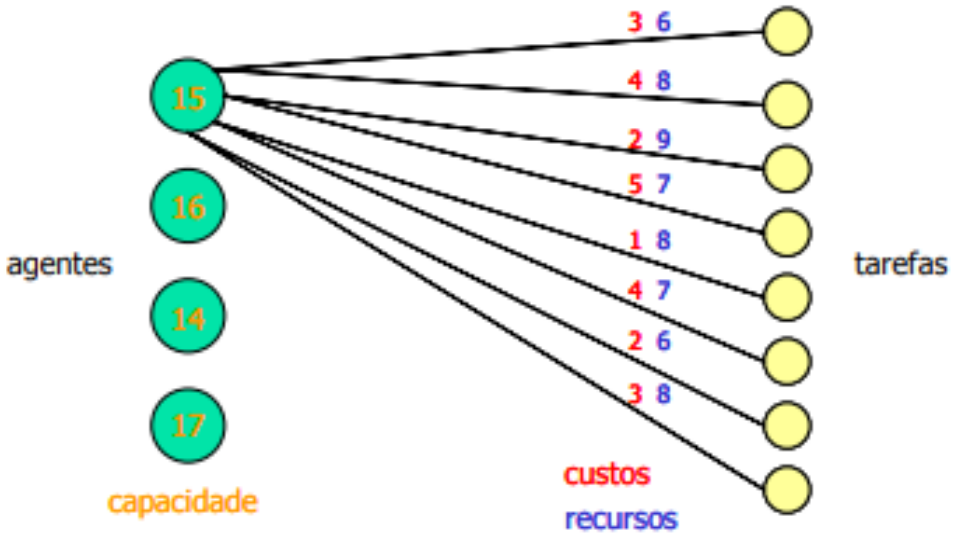
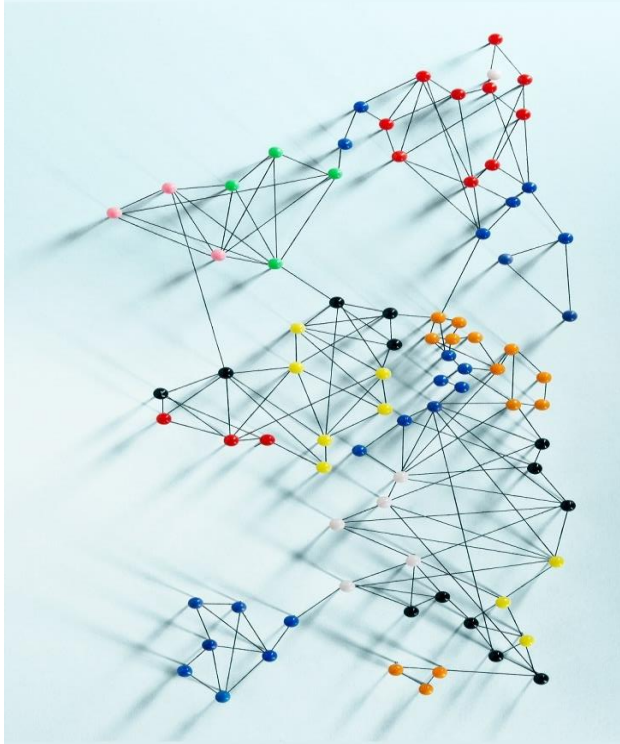


Figura 1. Representação do GAP a partir da estrutura de um grafo.



2.3. Modelo matemático

Apesar do GAP ser um problema NP-completo, é possível encontrar soluções ótimas para instâncias pequenas. Para isso, formula-se o GAP como um problema de programação linear inteira e utiliza-se um pacote computacional de otimização disponível no mercado (CPLEX, XPRESS, LINDO, dentre outros solvers). O modelo mais usual do GAP é apresentado a seguir.

$$Min \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}x_{ij} \qquad (1)$$

s.a.

$$\sum_{i=1}^{N} a_{ij}x_{ij} \le b_j \qquad \forall j \quad 1 \le j \le M \qquad (2)$$

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad \forall i \quad 1 \le i \le N \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad \begin{cases} \forall i & 1 \le i \le N \\ \forall j & 1 \le j \le M \end{cases} \qquad (4)$$

Onde:

x_{ij}: 1 se o trabalho i foi designado ao agente j, 0 caso contrário.

c_{ij}: custo associado à designação do agente j à tarefa i.

a_{ij}: recurso associado à designação do agente j à tarefa i.

b_j: capacidade do agente j.

m: número de agentes.

n: número de tarefas.

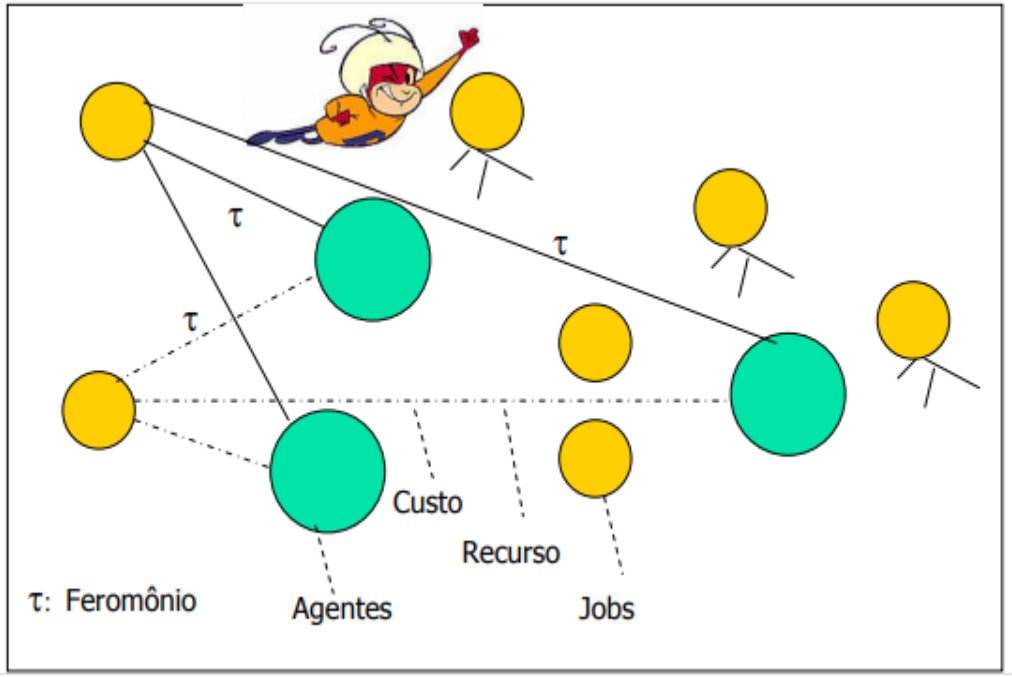
No modelo matemático apresentado tem-se a função objetivo (1) que procura minimizar a somatória dos custos das arestas pertencentes à solução (ou seja, quando $x_{ij} = 1$). Se um modelo de maximização for desejável, basta inverter o sinal dos custos. A restrição de capacidade (2) limita o montante de recursos que cada agente pode dispensar. A restrição de designação (3) garante que cada tarefa terá um único agente para atendê-la. Finalmente, a restrição binária (4), responsável por transformar o GAP em um problema combinatorial, limita dois valores para a variável x_{ij}, um ou zero.

3.4. Colônia de Formigas aplicado ao GAP

Um procedimento de otimização baseado em colônia de formigas é uma meta-heurística baseada em uma população de agentes (formigas) que faz uso de mecanismos de adaptação, cooperação e paralelismo visando a obtenção de um procedimento para resolução de problemas de otimização combinatória. O GAP possui uma estrutura propícia para ser resolvido por meio do algoritmo de colônia de formigas.

Heurísticas baseadas em colônias de formigas para o GAP foram utilizadas por Lourenço e Serra(1998) e mais recentemente por Randall(2004). Lourenço e Serra(1998) propõe uma heurística usando Max-Min. Randall(2004), faz uma análise desta abordagem levando em conta a estrutura especial do problema.

A heurísticas baseadas em colônias de formigas se mostraram promissoras e competitivas com outras apresentadas na literatura para resolução, porém esta abordagem não é a melhor para a resolução do problema. Além disso, a aplicação a este problema é um relevante instrumento de estudo.



Na Figura 3 todos os jobs estão conectados a todos os agentes, porém nem os jobs nem os agentes estão conectados entre si. Os feromônios estão nas arestas que ligam os jobs aos agentes. A própria definição do problema GAP faz com que cada job tenha um custo e consuma um recurso ao ser alocado a um determinado agente, sendo que o recurso total de cada agente é limitado.

Pseudo-código:

```
Para cada iteração
    Para cada formiga
        job = escolheTrabalho();                                    (p1)
        agente = escolheAgente(job);                                (p2)

        Se solução tornou-se infactível
            Reinicializa a formiga;                                 (p3)
        Senão
            Decai feromônio da aresta job-agente escolhida;         (p4)
            Se todos os jobs foram alocados
                Busca Local;                                                (p5)
                Se a solução corrente é a melhor até agora
                    Decai a qtd de feromônio de todas as arestas;           (p6)
                        Aumenta a qtd de feromônio das
                                arestas da solução corrente;                (p7)
                        Reinicializa a formiga;                             (p8)
```

Figura 4. Pseudo-código do ACO aplicado ao GAP.

3.9. Resultados Experimentais

A seguir na Tabela 1, são apresentados os resultados da aplicação do ACO para as instâncias gap1 ao gap12 (60 instâncias ao todo). A tabela apresenta as soluções ótimas das instâncias, informação extraída da literatura e obtida por técnicas de otimização linear inteira ("branch and bound"). Também são apresentadas as melhores soluções (maior custo) encontradas em cada uma das dez repetições efetuadas por instância. Por fim, são mostradas as médias das melhores soluções e os desvios percentuais dessas médias com relação à solução ótima. Uma célula da tabela preenchida com a letra "o" implica que o algoritmo obteve a solução de valor máximo (ótimo) para essa instância.

Pode-se afirmar que, em geral, o algoritmo obteve um desempenho muito bom. De uma amostra de 600 repetições o algoritmo atingiu a solução ótima do problema em 384, o que representa 64%. O maior desvio percentual com relação à solução ótima ocorreu no gap8-1, com 0,432%, um valor relativamente pequeno. O número de instâncias onde o algoritmo atingiu a solução ótima em todas as repetições foi 32 de 60, ou seja, aproximadamente 53,3%.

Considerando as instâncias com até 40 tarefas, o desempenho do algoritmo ACO foi ainda mais surpreendente, onde foi possível contabilizar 326 soluções ótimas de um total de 350, representando 93,1% de ótimos.

Já para as instâncias com mais de 40 tarefas, o desempenho já foi um pouco inferior, porém apresentando soluções de qualidade aceitável. Foram encontradas 58 soluções ótimas de 250 repetições, o que representa 23,2%. Esse número não é desprezível, pois deve-se levar em conta a grandeza do espaço de busca, que é gigantesco para essas instâncias.

A questão da aleatoriedade dos procedimentos de escolha dos agentes ou do termo heurístico utilizado não exerceu forte influência na qualidade das soluções, pois percebe-se que as dez repetições possuem um mesmo padrão característico.

Tabela 1. Resultados da aplicação do ACO ao GAP.

Dados			Melhores Soluções Encontradas em 10 Testes										Estatísticas	
Instância	Classe	Sol Otima[1]	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Média	Desvio (%)
gap1-1	c515	336	o	o	o	o	o	o	o	o	o	o	336.0	0.000
gap1-2	c515	327	o	o	o	o	o	o	o	o	o	o	327.0	0.000
gap1-3	c515	339	o	o	o	o	o	o	o	o	o	o	339.0	0.000
gap1-4	c515	341	o	o	o	o	o	o	o	o	o	o	341.0	0.000
gap1-5	c515	326	o	o	o	o	o	o	o	o	o	o	326.0	0.000
gap2-1	c520	434	o	o	o	o	o	o	o	o	o	o	434.0	0.000
gap2-2	c520	436	o	o	o	o	o	o	o	o	o	o	436.0	0.000
gap2-3	c520	420	o	o	o	o	o	o	o	o	o	o	420.0	0.000
gap2-4	c520	419	o	o	o	o	o	o	o	o	o	o	419.0	0.000
gap2-5	c520	428	o	o	o	o	o	o	o	o	o	o	428.0	0.000
gap3-1	c525	580	o	o	o	o	o	o	o	o	o	o	580.0	0.000
gap3-2	c525	564	o	o	o	o	o	o	o	o	o	o	564.0	0.000
gap3-3	c525	573	o	o	o	o	o	o	o	o	o	o	573.0	0.000
gap3-4	c525	570	o	o	o	o	o	o	o	o	o	o	570.0	0.000
gap3-5	c525	564	o	o	o	o	o	o	o	o	o	o	564.0	0.000
gap4-1	c530	656	o	o	o	o	o	o	o	o	o	o	656.0	0.000
gap4-2	c530	644	o	o	o	o	o	o	o	o	o	o	644.0	0.000
gap4-3	c530	673	o	o	o	o	o	671	o	o	o	o	672.8	0.030
gap4-4	c530	647	o	o	o	o	o	o	o	o	o	o	647.0	0.000
gap4-5	c530	664	o	o	o	o	o	o	o	o	o	o	664.0	0.000
gap5-1	c824	563	o	o	o	o	o	o	o	o	o	o	563.0	0.000
gap5-2	c824	558	o	o	o	o	o	o	o	o	o	o	558.0	0.000
gap5-3	c824	564	o	o	o	o	o	o	o	o	o	o	564.0	0.000
gap5-4	c824	568	o	o	o	o	o	o	o	o	o	o	568.0	0.000
gap5-5	c824	559	o	o	o	o	o	o	o	o	o	o	559.0	0.000
gap6-1	c832	761	o	o	o	o	o	o	o	o	o	o	761.0	0.000
gap6-2	c832	759	o	o	o	o	o	o	o	o	o	o	759.0	0.000
gap6-3	c832	758	757	o	o	757	o	o	o	757	o	757	757.6	0.053
gap6-4	c832	752	o	o	o	o	o	o	o	o	o	o	752.0	0.000
gap6-5	c832	747	746	o	o	o	o	o	o	746	o	o	746.8	0.027
gap7-1	c840	942	941	941	941	941	o	941	o	o	941	941	941.3	0.074
gap7-2	c840	949	o	o	o	o	o	o	o	o	o	o	949.0	0.000

4. SISTEMA IMUNOLÓGICO ARTIFICIAL

4.1. Introdução

A simulação de processos evolutivos naturais objetivando resolver problemas de explosão combinatória e multimodais demonstrou ser uma estratégia eficaz e robusta. A otimização do comportamento de um sistema, obtida através da simulação de processos evolutivos, representa uma abordagem poderosa para aprendizagem de máquina e estudo de fenômenos auto-organizados. A implementação computacional desses métodos de simulação da evolução, chamada computação evolutiva, possibilita a determinação de soluções para várias classes de problemas (Bäck *et al.*, 2000a,b). Além dessas, outras linhas de pesquisa ainda mais recentes têm surgido como novos paradigmas de computação.

Os sistemas imunológicos artificiais (SIA) (Dasgupta, 1998a), que surgiram a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de novas ferramentas computacionais, já vêm sendo utilizados em diversas áreas, como reconhecimento de padrões, detecção de falhas e anomalias, segurança computacional, otimização, controle, robótica, *scheduling*, análise de dados, aprendizagem de máquina, dentre outras, como pode ser encontrado em Dasgupta, (1998a,b), Bäck *et al.*, (2000a,b), Timmis (2000) e em De Castro (2001).

Nas áreas de engenharia e computação, tem surgido um forte interesse pelo estudo dos sistemas imunológicos devido, principalmente, à sua capacidade de processamento de informação. Sob uma perspectiva de engenharia, existem diversas características do sistema imunológico (SI) que podem ser destacadas:

4.5.2. Pseudocódigo

A implementação computacional do SIA pode ser feita seguindo o pseudocódigo descrito no Figura 11.

```
Ab   ← inicializar o repertório de anticorpos com tamanho p;
f    ← afinidade(Ab);
d    ← desafinidade(Ab);
geração ← 1;
enquanto geração ≤ gen faça
    para cada anticorpo i de Ab faça
        se d_i = 0 então
            C   ← clonar(Ab_i,nclones);
          senão
            C   ← criar_anticorpos(nclones);
        fim-se;
            f   ← afinidade(C);
            C*  ← maturar(C,f);
            f   ← afinidade(C*);
            d   ← desafinidade(C*);
          Ab_{n} ← selecionar(C*,f,d);
          Ab_{n} ← busca_local(Ab_{n});
            Ab_i ← selecionar (Ab_{n},Ab_i);
    fim-para;
    Atualizar f e d;
    se diversidade(Ab) < l então
        n       ← Supressão(Ab) //aplica supressão e retorna o
número de soluções extraídas;
        Ab_{d}  ← criar_anticorpos(n);
        Ab      ← Ab ∪ Ab_{d}
    fim_se;
    geração ← geração + 1;
fim_enquanto;
```

5. CONCLUSÕES

O algoritmo ACO mostrou-se eficaz no tratamento do GAP, problema de natureza combinatorial explosiva, de difícil tratamento até a otimalidade. Foram obtidas soluções ótimas em grande parte dos testes efetuados e mesmo quando a solução ótima não é atingida, verifica-se um baixo desvio percentual com relação ao ótimo. A abordagem adaptativa para a utilização de diferentes termos heurísticos ao longo das iterações do ACO mostrou-se em uma técnica eficaz, que ajusta o algoritmo de acordo com a dificuldade ou não de encontrar uma solução factível. O tempo computacional do algoritmo foi relativamente pequeno, considerando um PC de 2 GHz, sendo inferior a 30 segundos para as maiores instâncias e praticamente instantâneo para as instância de menor porte.

Em relação ao desempenho do SIA, concluímos que apesar dos resultados obtidos para algumas instâncias serem ligeiramente inferior aos encontrados pelo método proposto por Chu & Beasley (1997), o fato do SIA conseguir preservar um conjunto diversificado de boas soluções (ótimos locais) permite que um plano de contingência seja proposto, possibilitando uma rápida tomada de decisão em caso de adição de novas restrições. Esta característica, de uma forma geral, faz com que o sistema torne-se mais robusto e menos sensível a falhas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

(1) Bäck, T., Fogel, D.B. & Michalewicz, Z. (2000a), *Evolutionary Computation 1 Basic Algorithms and Operators*, Institute of Physiscs Publishing, Bristol, UK.

(2) Bäck, T., Fogel, D.B. & Michalewicz, Z. (2000b), *Evolutionary Computation 2 Advanced Algorithms and Operators*, Institute of Physiscs Publishing, Bristol, UK.

(3) Chu, P.C. & Beasley, J.E. (1997), *A Genetic Algorithm for the Generalised Assignment Problem*, Computers Operacional Research, Elsevier Science, UK.

(4) Dasgupta, D. (1998a). *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, Germany.

(5) Dasgupta, D. (1998b). An Overview of Artificial Immune Systems and Their Applications, *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, Germany.

(6) De Castro, L.N. (2001). *Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais*. Tese de Doutorado, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e Computação.

(7) Dorigo, M.; Maniezzo, V. & Colorni, A. (1991) - *Positive feedback as a search strategy*, Technical Report 91-016, Dipartim ento di Elettronica e Informazione, Politecnico di Milano, Italy.

(8) Janeway, C.A., Travers P., Walport, M. & Capra, J.D. (2000). *Imunobiologia: O Sistema Imunológico na Saúde e na Doença*. 4 ed, Artmet, Porto Alegre, RS, Brasil.

(9) Kubo, M. & Kakazu, Y. (1993). *Simulating a competition for foods between ant colonies as a coordinated model of autonomous agents*. Proceedings of International Conference on Systems, Man, and Cybernetics, La Touquet, France, Vol. 5, p. 142-148.

(10) Lourenço, H. R., & Serra, D. (1998) – *Adaptive Approach Heuristics for The Generalized Assignment Problem*.

(11) Randall, M.(2004) – *Heuristics for Ant Colony Optimisation using the Generalised Assignment Problem*. IEEE.

(12) Stützle, T. & Dorigo, M. (1999) - *New ideas in optimization*, D. Corne, M. Dorigo & F. Glover (editors), McGraw-Hill.

(13) Timmis, J. (2000). Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory. Tese de Doutorado, University of Whales, Department of Computer Science, Aberystwyth, Ceredigion, Wales.