Unidad 4

Inteligencia artificial:
Inteligencia Colectiva

Dra. Soledad Espezua. Ll.
sespezua@pucp.edu.pe

ia.inf.pucp.edu.pe

Agenda

- Inteligencia Colectiva (IC)
  - Introducción
  - Principales Algoritmos
    - PSO
    - ABC
    - ACO
  - Bibliografía

Introducción

A look back, a glance ahead

"Nature is concerned with that which works. Nature propagates that which survives. She has little time for erudite contemplation, and we have joined her in her expedient pursuit of betterment"[1]

David E. Goldberg

1. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley. Cap. 8, p 309.

Introducción

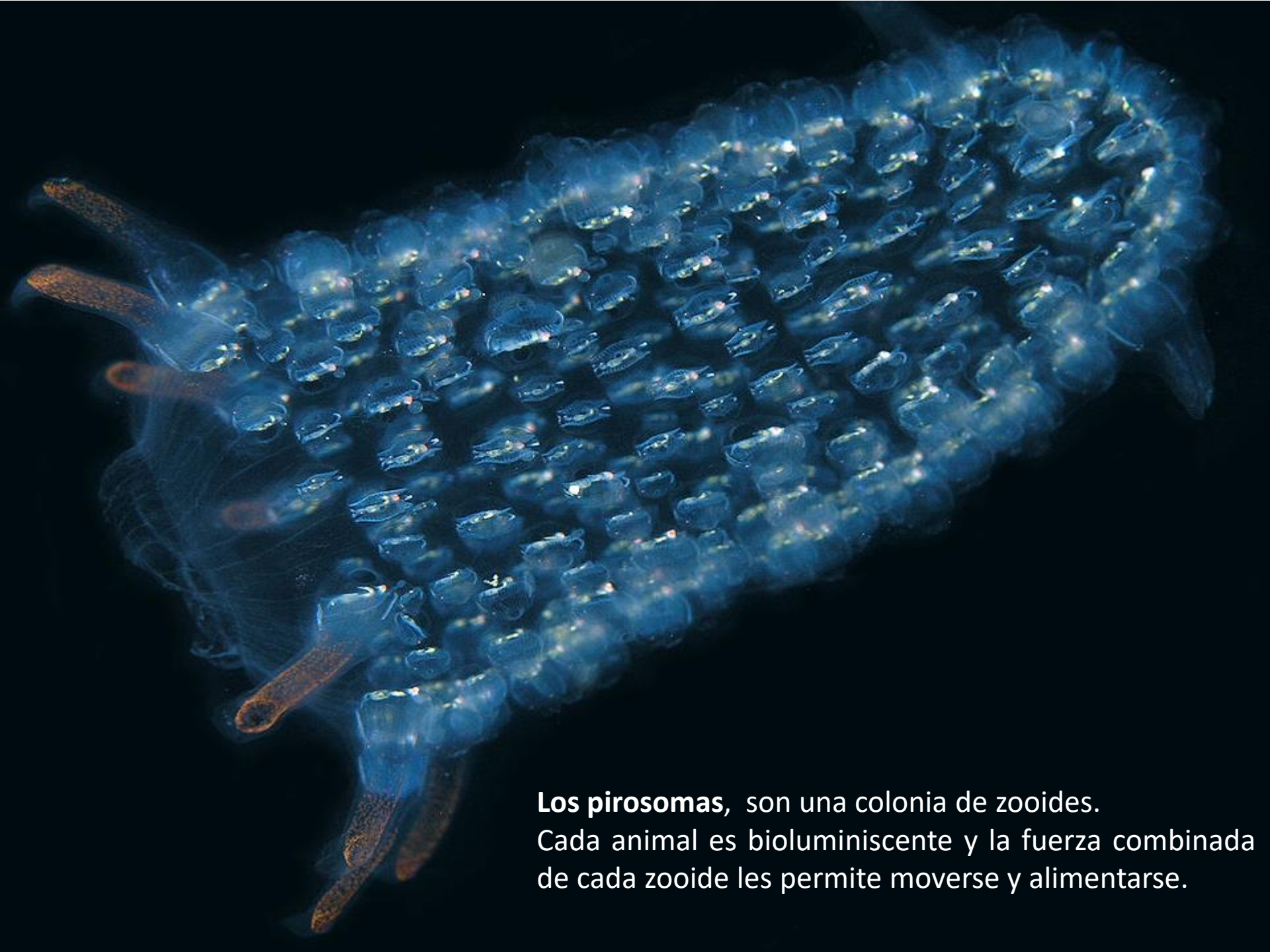
Extremófilos

Pepinos de mar

Carabelas portuguesas

Son una colonia de zooides que al nacer se pegan unos a otros y siguen juntos de por vida.

Cada zooide se especializa en una función de supervivencia: defensa, alimentación o reproducción.

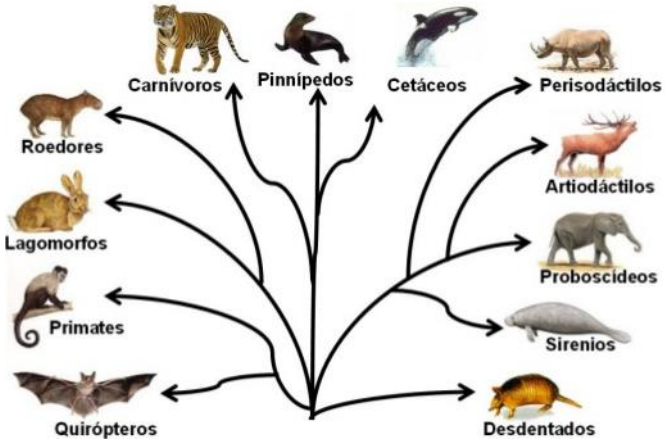


Los pirosomas, son una colonia de zooides.
Cada animal es bioluminiscente y la fuerza combinada de cada zooide les permite moverse y alimentarse.

Introducción

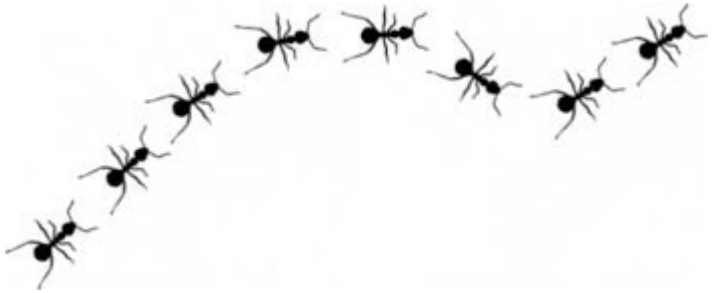
Inteligencia

- Filogenética: a nivel de especies (genoma de las especies).



- Ontogenética: a nivel de individuo (memoria neuronal).

- Socio genética: a nivel de grupo (experiencia compartida).



Inteligencia colectiva (*Swarm Intelligence*)

manadas de mamíferos

La inteligencia colectiva (IC) es una rama de la Computación Bioinspirada, que estudia meta heurísticas inspiradas en conductas colaborativas (generalmente de la misma especie) observadas en la naturaleza.

bandadas de aves o cardúmenes de peces

colonias de abejas, avispas, hormigas, termitas, arañas

sociedades de insectos:

Inteligencia Colectiva (IC)

- El termino IC[1], fue introducida por Gerardo Beni y Jing Wang en 1989, cuando investigaban las propiedades de agentes auto-organizados para sistemas robóticos celulares. Se inspiraron en el comportamiento social de colonias de hormigas.

Gerardo Beni

Jing Wang

[1] E. Bonabeau, et. al , 1999. Swarm Intelligence. From Nature to Artificial Systems. Oxford University Press.

Inteligencia Colectiva

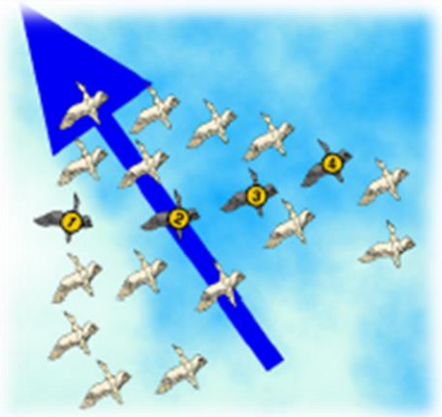
Craig Reynolds

Craig Reynolds, un experto en vida artificial y computación gráfica, observo que el movimiento de bandas de pájaros se sincronizaba sin que existiese un control central.

Homepage: http://www.red3d.com/cwr/

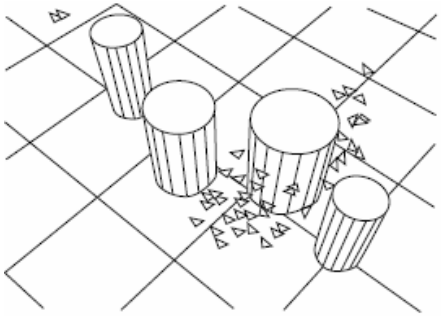
IC - *Artificial life*

C. Reynolds, observo que las aves seguían 4 reglas:



1) **Separación**: para evitar colisiones con sus vecinos
2) **Alineación**: cada ave debe seguir la misma dirección y velocidad de sus vecinos
3) **Cohesión**: mantenerse cerca de sus vecinos
4) **Desviación**: desviar obstáculos al frente

Reynolds propuso un **modelo computacional de movimiento de** "*Boids*" [2]:



Craig Reynolds - Original (1986) Boids simulation

1. Reynolds, Craig W. (1987). "Flocks, herds, and schools: A distributed behavioral model". Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87). ACM. **21** (4): 25–34.

1. *Particle Swarm Optimization (PSO)*

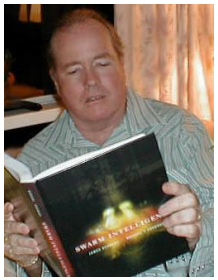
Cúmulos /enjambres/nubes de partículas



Conjunto de métodos inspirados en el comportamiento social de bandadas de aves , cardumen de peces y otros.

Particle Swarm Optimization (PSO)

- En 1995, PSO fue desarrollado por James Kennedy y Russell Eberhart[2], para resolver problemas de optimización continua.



James Kennedy

Russell Eberhart

- PSO se inspiro en el comportamiento grupal de pájaros en la naturaleza.
- Para crear PSO usaron las reglas de cohesión y alineamiento del modelo de C. Reynolds.
- PSO, al igual que AG es un método basado en poblaciones, pero diferentemente de los AG, la metáfora por detrás es la cooperación en lugar de la rivalidad. En PSO, los miembros del grupo nunca mueren.

[2] Kennedy, J.; Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks Vol. IV: 1942–1948. .

Características de PSO

- **<u>Una población en PSO se compone de un conjunto de partículas</u>** que representan las soluciones a un problema.
  - Cada partícula representa un vector de posición

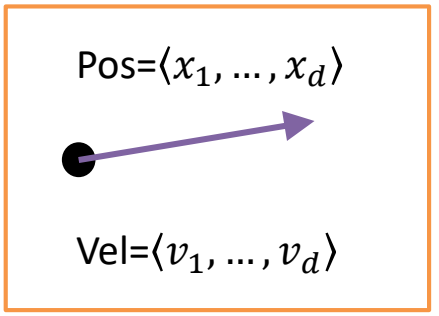
$$x_i = inf + rand(0,1).(sup - inf)$$

    - inf y sup son límites de acotamiento, la longitud de x_i es igual al número de variables del problema $[1, \dots, d\,]$.

- Una partícula en PSO tiene:
  - Un vector de su posición actual.
  - Un vector de velocidad que dirige su movimiento. La velocidad máxima que puede alcanzar la partícula es:

$$v_{max} = rand(0,1).(sup - inf)/2$$

  - La capacidad de intercambiar información con sus vecinos y memorizar una posición anterior.



PSO - Movimiento de partículas

- Actualización de la **posición**:

Una partícula **i** se mueve a una nueva posición x_i^{k+1} , dependiendo de su posición actual (x_i^k) y su nueva velocidad (v_i^{k+1}).

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad (1)$$

- Actualización de la **velocidad**:

Aprendizaje Cognitivo — Aprendizaje Social

$$v_i^{k+1} = \omega . v_i^k + \varphi_1 . rnd_1 . (pBest_i - x_i^k) + \varphi_2 . rnd_2 . (g - x_i^k) \qquad (2)$$

- $\omega . v_i^k$: Fuerza inercial
- $\varphi_1 . rnd_1 . (pBest_i - x_i^k)$: Fuerza hacia la mejor posición local $pBest_i$
- $\varphi_2 . rnd_2 . (g - x_i^k)$: Fuerza hacia la mejor posición global g del enjambre.

La velocidad de desplazamiento de cada partícula **i** se ajusta basándose en la mejor posición que ha obtenido ella misma (aprendizaje cognitivo) y la mejor posición global (aprendizaje social) encontrada hasta el momento por el grupo.

PSO - Movimiento de partículas

Nueva Posición: $x_i^{k+1} = x_i^k + v_i^{k+1}$

Inercia Memoria Cooperación

$$v_i^{k+1} = \omega . v_i^k + \varphi_1 . rnd_1 . (pBest_i - x_i^k) + \varphi_2 . rnd_2 . (g - x_i^k)$$

donde:

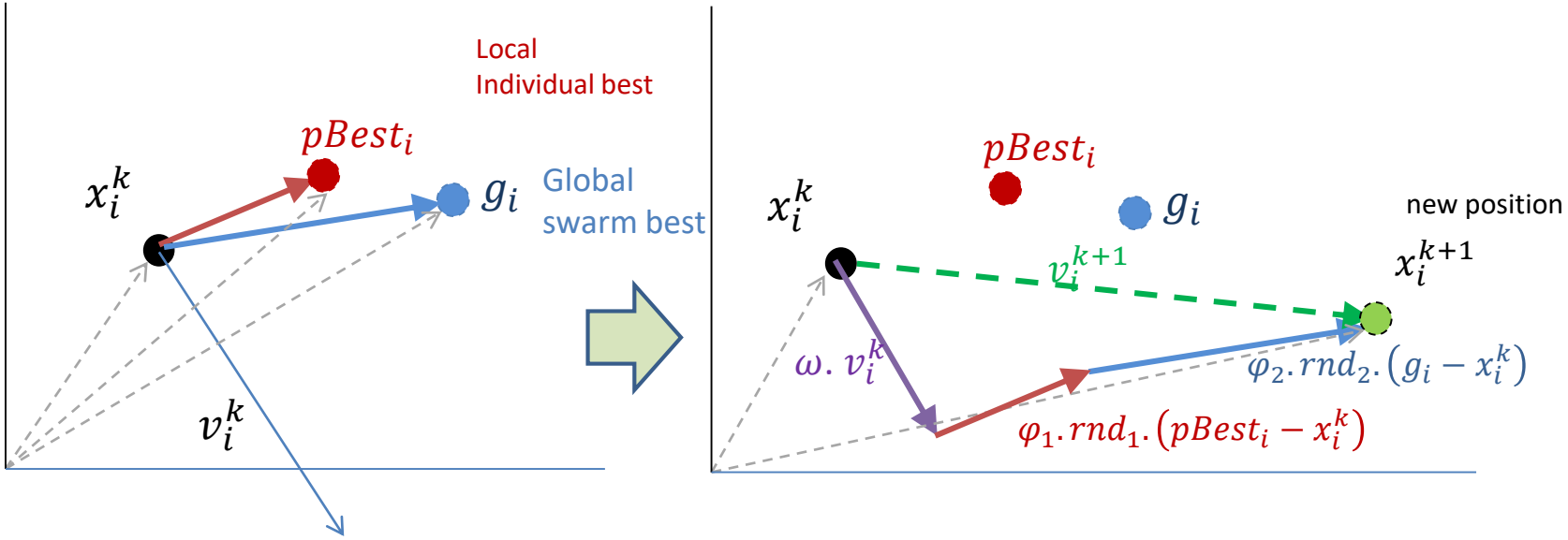
- k, iteración
- i, partícula
- v_i^k, velocidad
- ω, inercia de la partícula
- $pBest_i$, mejor posición encontrada por partícula *i*
- g , mejor posición global encontrada por el enjambre (*swarm*)
- φ_1 y φ_2, pesos de tendencia individual o social respectivamente.
- rnd_1 y rnd_2 , valores aleatorios entre $\{0,1\}$.

PSO - Elección de parámetros ($\omega, \varphi_1, \varphi_2$)

$$v_i^{k+1} = \omega . v_i^k + \varphi_1 . rnd_1 . (pBest_i - x_i^k) + \varphi_2 . rnd_2 . (g - x_i^k)$$

- Algunas recomendaciones de valores para:
  - Inercia de la partícula ω =0.5, porque debe influir menos.
  - Para (φ_1 , φ_2) = 2 (tendencia local φ_1 y tendencia global del enjambre φ_2).
- Una mejora del método consiste en empezar con una inercia relativamente alta (ω = 1.4) que se va reduciendo en cada iteración, por ejemplo, multiplicándola por un factor r < 1 para estabilizar el sistema.

PSO - Movimiento de una partícula



Aprendizaje Cognitivo — Aprendizaje Social

$$v_i^{k+1} = \omega . v_i^k + \varphi_1 . rnd_1 . (pBest_i - x_i^k) + \varphi_2 . rnd_2 . (g_i - x_i^k) \quad (2)$$

Inercia — Memoria — Cooperación

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (1)$$

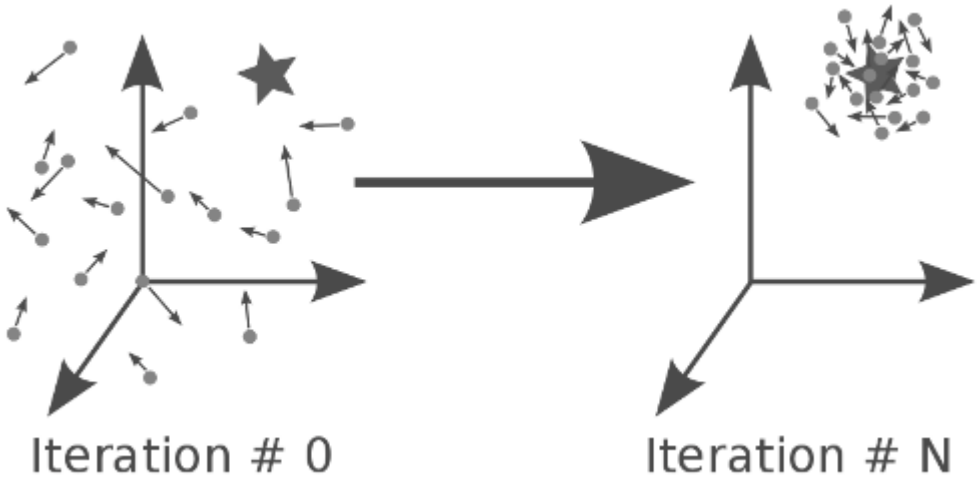
Pseudocódigo PSO

Algorithm Pseudocodigo del algoritmo PSO Canónico

1: $S \leftarrow SwarmInitialization()$
2: **while** not stop condition **do**
3: **for** each particle x_i of the swarm S **do**
4: evaluate(x_i)
5: **if** $fitness(x_i)$ is better than $fitness(pBest_i)$ **then**
6: $pBest_i \leftarrow x_i$
7: **end if**
8: **if** $fitness(pBest_i)$ is better than $fitness(g_i)$ **then**
9: $g_i \leftarrow pBest_i$
10: **end if**
11: **end for**
12: **for** each particle x_i of the swarm S **do**
13: $v_i^{k+1} = \omega \cdot v_i^k + \varphi_1 \cdot rnd_1 \cdot (pBest_i - x_i^k) + \varphi_2 \cdot rnd_2 \cdot (g - x_i^k)$
14: $x_i^{k+1} = x_i^k + v_i^{k+1}$
15: **end for**
16: **end while**
17: **Output:** best solution found

Consideraciones del Procedimiento

- Iteraciones:
  - Después de actualizar todas las partículas, se comparan sus mejores posiciones $pBest$ con la mejor posición global g, si algún $pBest$ es mejor que g, entonces: $g \leftarrow pBest_i$.
  - Al acabar las iteraciones g contiene la mejor solución encontrada por el algoritmo



2. *Artificial Bee Colony (ABC)*

Conjunto de métodos inspirados en las actividades de una colonia de abejas.

Artificial Bee Colony (ABC)

- El algoritmo de colonias de abejas o *Artificial Bee Colony* (ABC) [6], fue presentado por Dervis Karaboga en 2005.

- ABC es inspirado en el comportamiento observado en las abejas domésticas.
- ABC es un algoritmo propuesto para resolver problemas de optimización combinatoria. Está basado **en poblaciones** en la cual **los individuos, son las posiciones de comida**, y se modifican por acción de las abejas artificiales.
  - El objetivo de las abejas es encontrar un lugar con la mayor cantidad de néctar (optimo global).

[6] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

ABC – Modelo artificial

- En un modelo ABC, las abejas se mueven por el espacio de búsqueda eligiendo fuentes de néctar (individuos de la población).
- Cada fuente de néctar/ alimento es una solución alternativa al problema de optimización, y la cantidad de néctar de una fuente es el *fitness* de la solución.

Algunas abejas se mueven aleatoriamente sin influencia externa (exploración).

Cuando encuentran una fuente de néctar mayor, memorizan su posición y olvidan la anterior (explotación) .

- De este modo, ABC combina métodos de búsqueda local y búsqueda global, intentando equilibrar el balance entre exploración y explotación del algoritmo.

ABC y el Modelo biológico

- ABC es un modelo artificial para optimización, que consta de los siguientes elementos:

1) **Fuente de néctar/alimento:** Es un vector (posición) de valores reales.

2) **Abejas Empleadas (*employed*):** Visitan nuevas fuentes de alimento y guardan la ubicación. Seleccionan el alimento según la riqueza de la fuente (*fitness*).

   Luego regresan a la colmena para informar a sus compañeras Observadoras, sobre la fuente de alimento que están explotando, por medio de una danza. La danza es la metáfora del fitness.

3) **Abejas Observadoras (*Onlooker*):** seleccionan alguna de las fuentes de alimento según la riqueza de la fuente. Visitan esas fuentes de alimento y guardan la ubicación.

4) **Abejas Exploradoras (*Scout*):** se encargan de buscar nuevas fuentes de alimento.

ABC - Algoritmo básico

- Población inicial: $X_{SN\times d} \leftarrow \cup\,(inf, sup)$ es iniciada con un número SN (*Source Number*) de fuentes de alimento.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{SN} \end{bmatrix}, x_i \in R$$

- Cada fuente es un vector de valores reales generados de la siguiente manera:

$$x_i = x_{min} + rand(0,1).(x_{max} - x_{min}) \qquad (1)$$

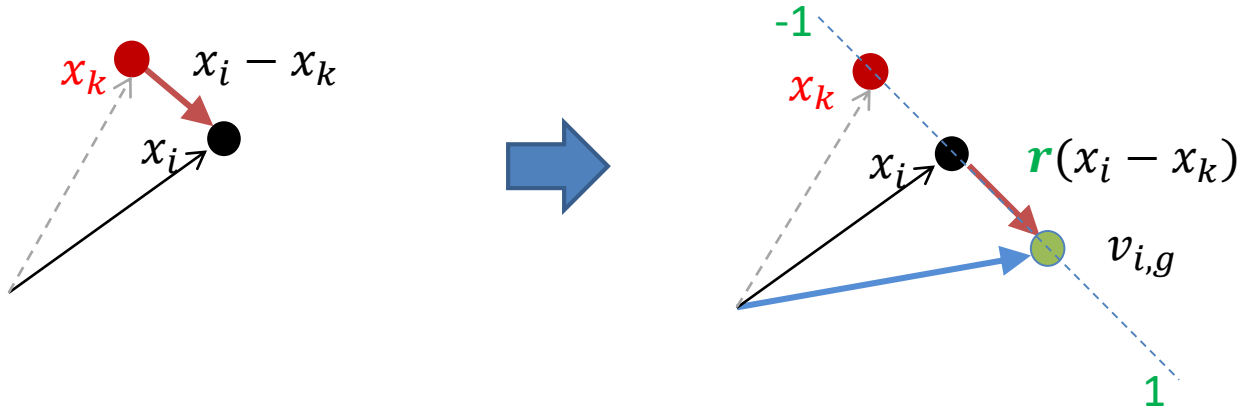
donde cada fuente x_i tiene dimensión [1,…,d], (x_{max}, x_{min}) son límites de acotamiento.

ABC - Algoritmo básico

- Fase de búsqueda por las abejas empleadas (*employee bee*):

Las abejas empleadas (representan operadores de variación), visitan las fuentes de alimento x_i y asignan una nueva posición $v_{i,g}$ a la fuente:

$$v_{i,g} = x_i + r(x_i - x_k) \qquad (2)$$



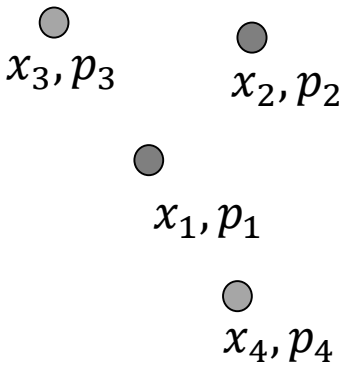
donde:

- x_i, fuente de alimento donde se encuentra la abeja en ese momento.
- x_k, fuente de alimento seleccionada aleatoriamente y diferente de x_i.
- g, iteración actual
- r , número real aleatorio entre [−1, 1].

ABC - Algoritmo básico

- Una vez que se obtiene $v_{i,g}$, se evalúa y se compara con x_i, luego se selecciona la mejor dependiendo de los valores de aptitud (*fitness*) que representan.
  - Si $fitness(v_{i,g}) > fitness(x_i)$

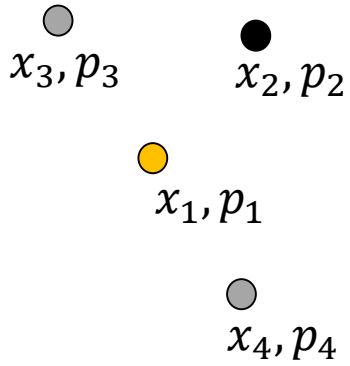
    $\boldsymbol{x_i \leftarrow v_{i,g}}$ se convertirá en un nuevo miembro de la población; de lo contrario x_i se conserva.
- Asignan a las fuentes de alimento una probabilidad p_i, proporcional al valor de su *fitness* (calidad de néctar que tiene la solución).



	fitness		*probability*
x_1	3	p_1	3/10
x_2	4	p_2	4/10
x_3	2	p_3	2/10
x_4	1	p_4	1/10

ABC - Algoritmo básico

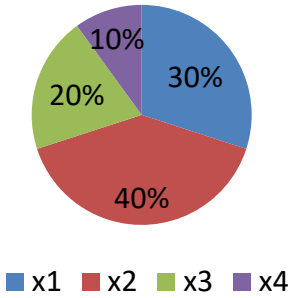
- Fase de selección por las abejas observadoras (*onlooker bee*)



	fitness		*probability*
x_1	3	p_1	3/10
x_2	4	p_2	4/10
x_3	2	p_3	2/10
x_4	1	p_4	1/10

Selección:

ruleta, torneo, etc



En esta fase, las abejas observadoras seleccionan las fuentes de alimento de acuerdo a una probabilidad p_i asociada a la fuente de alimento. Esta es la propiedad de retroalimentación positiva del algoritmo ABC. Se calcula de la siguiente manera:

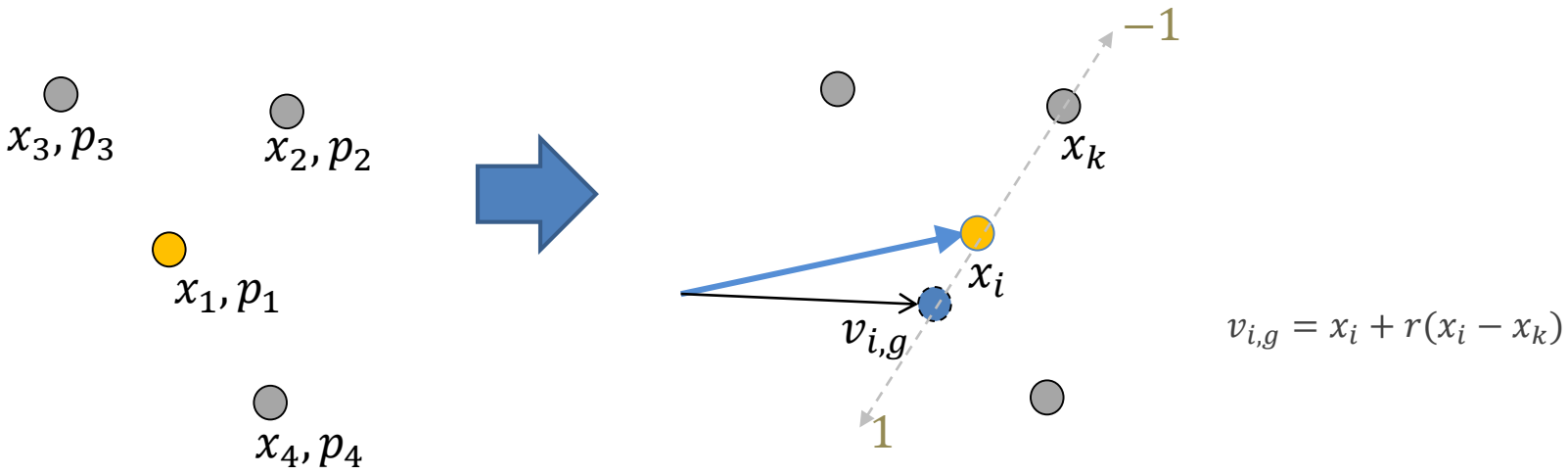
$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \qquad (3)$$

donde: $fit_i = fitness$ (valor de la solución i); SN= número de fuentes de alimento.

ABC - Algoritmo básico

- Fase de selección por las abejas observadoras (*onlooker bee*)

  Luego de seleccionar la fuente de alimento, las abejas observadoras crearán una nueva posición candidata en el vecindario de la fuente de alimento utilizando [2] y luego escoge para reemplazo, solo si tiene mejor fitness que la solución actual.



Si $fitness(v_{i,g}) > fitness(x_i)$

$$\boldsymbol{x_i \leftarrow v_{i,g}}$$

ABC - Algoritmo básico

- Cuando todas las abejas empleadas y observadoras completan sus búsquedas, se verifica si hay alguna fuente que no ha sido mejorada a través de un número predeterminado de ciclos, para abandonarla. Las abejas exploradoras eligen una fuente nueva.

- Fase de abejas exploradoras (*scout*):

  Las abejas exploradoras ingresan para generar nuevas fuentes de alimento de manera aleatoria y para substituir fuentes que no han sido mejoradas. Esta operación puede definirse de la siguiente manera:

$$x_i = x_{min} + rand(0,1).(x_{max} - x_{min}) \qquad (4)$$

  Este proceso ayuda a evitar soluciones sub-óptimas.

ABC - Parámetros del Algoritmo básico

Los parámetros del algoritmo son los siguientes:

1. SN: número de fuentes de alimento (tamaño de la población)
2. max_iter: número total de iteraciones que ejecutará ABC
3. Limit: número de ciclos que será conservada fuente de alimento (una solución sin mejorar) antes de ser reemplazada por una nueva solución generada por una abeja exploradora.

ABC - Pseudocódigo

(1) Generate the initial population x_i (i=1,2,...,SN), using (1)

(2) Evaluate the fitness(fit(x_i))of the population

(3) g = 0

(4) **While** (g < max_iter)

(5) **For** each **employed bee**{

Produce new solutions $v_{i,g}$ in the neighbourhood of x_i, using (2)

Calculate its fitness value fit($v_{i,g}$)

Apply the selection process between x_i and $v_{i,g}$

(6) **For** each solution (x_i) { Calculate the probability values p_i, using (3) }

(7) **For** each **onlooker bee**

Select a solution x_i depending on p_i

Produce new solutions $v_{i,g}$ from the solutions x_i, using (2)

Calculate its fitness value fit($v_{i,g}$)

Apply the selection process

(8) **If Limit==true**

there is an abandoned solution x_i replace it with a new randomly produced solution x_i for the **scout bee**, using (4)

(9) Memorize the best solution achived so far

(10) g= g+1

ABC - Consideraciones

- Abejas empleadas: su número es proporcional al número de fuentes de alimento, y su función es evaluar y modificar las soluciones actuales para mejorarlas. Si la nueva solución no es mejor entonces mantiene la solución actual.

  # Abejas empleadas = # fuentes de alimento

- Abejas observadoras: su número es proporcional al número de fuentes de alimento. Estas abejas escogerán una fuente de alimento, con base en la información que comparten las abejas empleadas mediante la danza. Esta danza se puede simular mediante el método de la ruleta o torneo de tamaño "t", donde la fuente de alimento con mejor *fitness* es seleccionada.

  # Abejas observadoras = # Abejas empleadas = # fuentes de alimento

ABC - Consideraciones

- ABC es un algoritmo de naturaleza estocástica que combina la búsqueda local (realizada por las abejas empleadas y observadoras), y también la búsqueda global (por las abejas exploradoras) para equilibrar el proceso de exploración y explotación.
  - Empleadas y Observadoras (operadores de variación)
  - Exploradoras (operadores de reemplazo)

- Ventajas del algoritmo ABC sobre otros métodos de optimización:
  - ABC es simple de implementar
  - Tiene pocos parámetros de control
  - Es robusto y altamente flexible
  - Fácil de combinar con otros métodos
  - Rápida convergencia al combinar procesos de exploración y explotación.

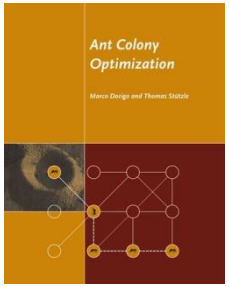
ABC - Consideraciones

- ABC tiene algunas debilidades cuando se pone en práctica.
  - Este método requiere evaluar el *fitness* en cada nueva fase de abejas del algoritmo para mejorar su rendimiento. Necesita una gran cantidad de evaluaciones de la función objetivo.
  - Se ralentiza cuando la población de soluciones aumenta (costo computacional alto).
  - Tiene muchas iteraciones y, por lo tanto, requiere una gran capacidad de memoria.

3. Ant Colony Optimization (ACO)

3. *Ant Colony Optimization* (ACO)

- Marco Dorigo[4] en 1992 presento *Ant Colony Optimization* (ACO) como parte de su tesis de doctorado.

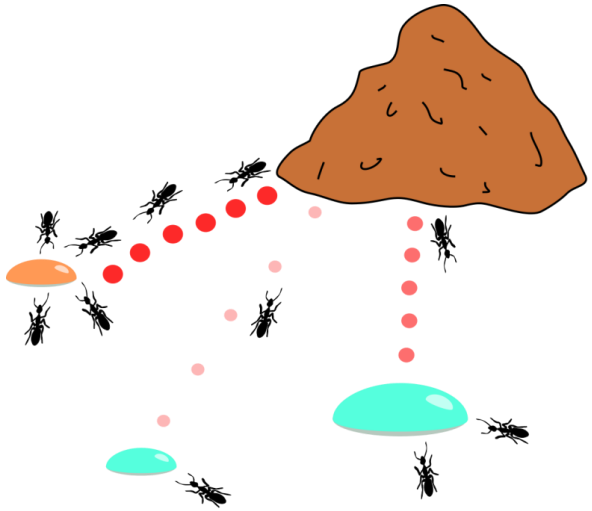


- ACO es una meta heurística de optimización inspirado en el comportamiento de colonias de hormigas. Está basado en el hecho de que las hormigas siempre encuentran el camino más corto a una fuente de alimento.
- ACO fue planteado para resolver problemas computacionalmente complejos que pueden ser reducidos a la búsqueda de caminos cortos en grafos.
- En ACO la colonia de hormigas artificiales, son unos agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales.

[4] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.

Colonias de hormigas naturales

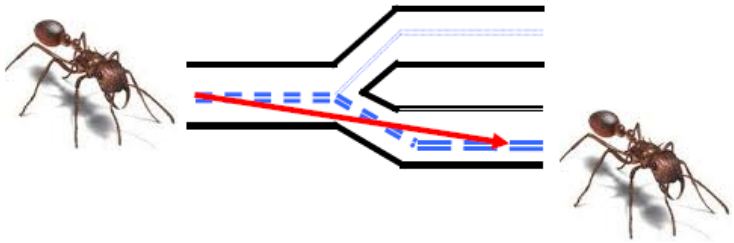
- Cuando las hormigas en sus recorridos encuentran una fuente de alimento, depositan en su trayectoria, una sustancia llamada feromona que todas pueden oler (es una forma de comunicación indirecta).



- La acción continua de la colonia dejando un rastro prolongado de feromonas, permite a las hormigas elegir, siempre el camino más corto y luego regresar a su hormiguero con el alimento.
- Cuando el alimento termina, los rastros no son remarcados por la hormigas que regresan y el olor se desvanece.

Colonias de hormigas naturales

- Si un obstáculo bloquea un camino hacia una fuente de alimento, las hormigas exploran otras rutas. Si tienen éxito, marcan un nuevo camino.
  - Los caminos exitosos son seguidos por más hormigas, y cada una lo refuerza con más feromona.
- Cada vez que una hormiga llega a una bifurcación, decide el camino a seguir de un modo aleatorio.

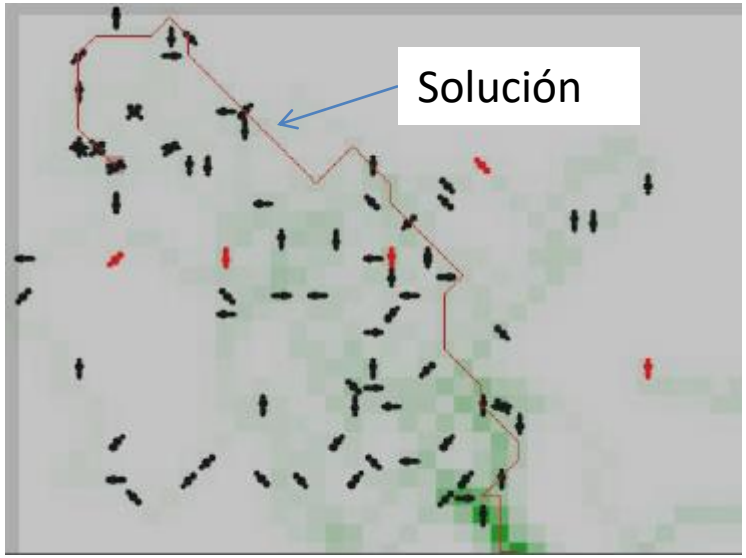


  - Eligen con mayor probabilidad los caminos con un alto rastro de feromona.
  - Las bifurcaciones más prometedoras van acumulando feromonas. Las menos prometedoras pierden feromona por evaporación.

ACO - Características

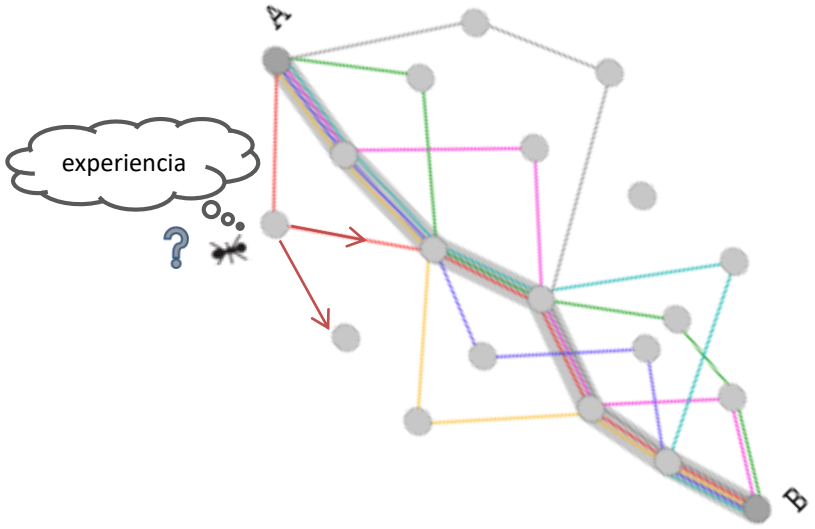
1. Los algoritmos de ACO son esencialmente constructivos.

   En cada iteración, utilizan agentes (hormigas artificiales) para construir soluciones en forma incremental. Cada una de las hormigas construye de forma independiente una solución al problema incorporando soluciones parciales en su recorrido por el grafo.



ACO - Características

2. La elección de incorporación de soluciones parciales se realiza mediante una regla probabilística que toma en cuenta la experiencia adquirida en etapas anteriores de la búsqueda y la información heurística del problema considerado.

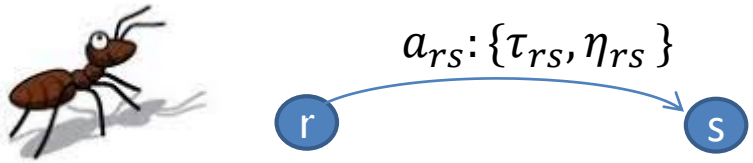


3. Para incorporar la experiencia adquirida en la construcción de soluciones se utiliza una matriz de feromona, a modo de memoria que almacena el rastro depositado por las hormigas.

ACO – Recorrido entre aristas

- ACO inicia con *k* hormigas artificiales ubicadas en el nodo inicial de un grafo.

  Una hormiga se mueve de un estado ***r*** a un estado ***s*** cuando recorre una arista a_{rs}. Una arista del grafo (posibles camino) tiene asociada dos tipos de información que guían su movimiento:

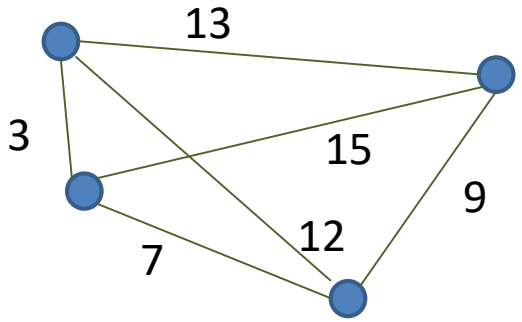


1. ***Información heurística*** (η_{rs}): mide la conveniencia de moverse desde el nodo r hasta el nodo s. Generalmente las hormigas no modifican esta información durante la ejecución del algoritmo. Esta información es calculada por alguna heurística que indica a *priori* la conveniencia de dicho movimiento.
2. ***Información del rastro de feromonas*** (τ_{rs}): imita a la feromona que depositan las hormigas naturales. Está asociado a la "memoria o recuerdo" en el movimiento de a_{rs}. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas.

ACO – Recorrido entre aristas

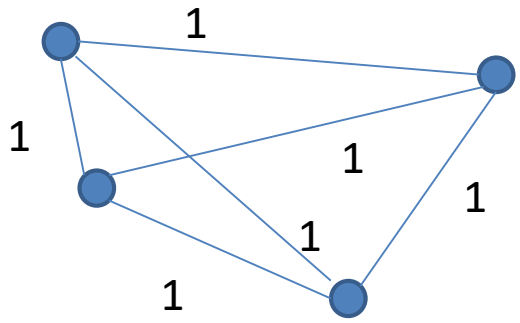
Información heurística (η_{rs})

Grafo de Costos



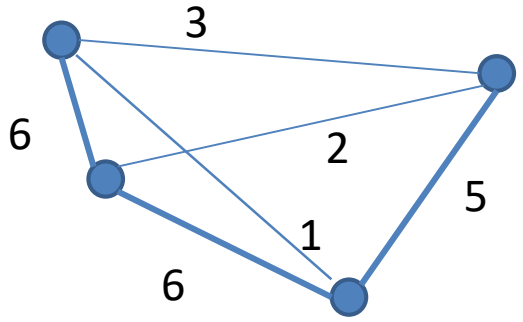
Información del rastro de feromonas (τ_{rs})

Grafo de Feromonas



Iteración 1

. . .



Iteración *n*

ACO – Tabla de transiciones

- Una hormiga toma decisiones basada en la información heurística (η_{rs}) y una tabla de transiciones de la información del rastro de feromonas (τ_{rs}). Esta tabla contiene la información del nodo en el que se encuentra la hormiga y de las aristas adyacentes. Se actualiza cada vez que una hormiga visita una arista. Registra el rastro de la feromona que deja la hormiga. También se actualiza cada vez que se evapora el rastro de feromona.

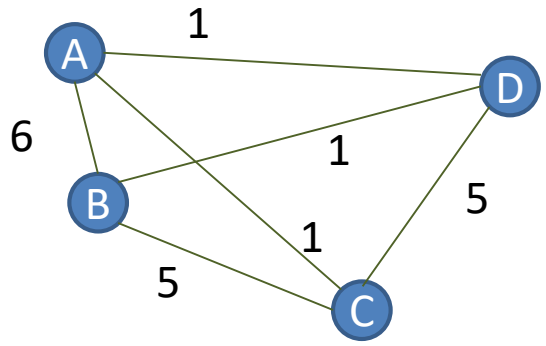
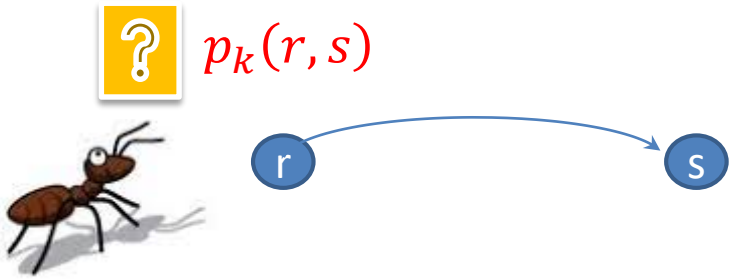


Tabla de transiciones

	B	C	D
A	6	1	1
B	∞	5	1
C	∞	∞	1

ACO - La hormiga artificial

- La hormiga artificial es un agente que:
  - Recuerda los nodos que ha recorrido a través de una lista de nodos visitados. Al finalizar, esta lista contiene la solución construida por la hormiga.
  - En cada iteración, la probabilidad de que una hormiga *k* se mueva de un nodo r hacia un nodo s, es definida por una probabilidad de transición: $p_k(r, s)$.



ACO – Probabilidad de transición

La probabilidad de transición de una hormiga, de moverse de un nodo *r* al *s*:

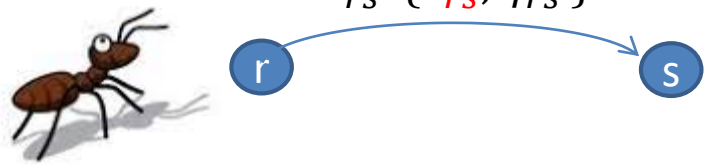
Cantidad de feromonas — Facilidad de transición

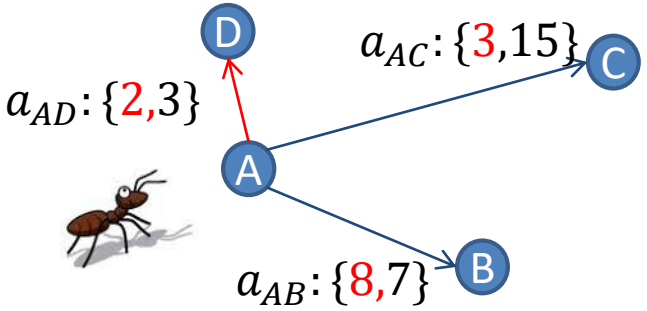
$$p_k(r,s) = \begin{cases} \dfrac{(\tau_{rs})^{\alpha}.(\eta_{rs})^{\beta}}{\sum_{u \in J_k(r)}[(\tau_{ru})^{\alpha}.(\eta_{ru})^{\beta}]} , & \text{si } s \in J_k(r) \\ 0 , & \text{si } s \notin J_k(r) \end{cases} \qquad \textbf{(1)}$$

donde:

- $s \in J_k(r)$ Si s es pertenece al conjunto de nodos alcanzables desde r (y no visitados aún) por la hormiga k.
- $s \notin J_k(r)$ Si s no es alcanzable desde *r,* entonces $p_k(r,s)$= 0
- α y β: son pesos que establecen el equilibrio entre la influencia de la información del rastro de feromonas y la información heurística.
  - $\alpha \geq 0$ controla la influencia del rastro de feromonas τ_{rs}
  - $0 \leq \beta \leq 1$ controla la influencia de la facilidad de transición η_{rs}
- $\sum_{u \in J_k(r)}[(\tau_{ru})^{\alpha}.(\eta_{ru})^{\beta}]$: sumatorio de la vecindad factible de la hormiga *k*
- $\eta_{rs} = \frac{1}{d_{rs}}$, donde d_{rs} es la distancia o costo de ***r*** a ***s***

ACO – Probabilidad de transición





Ejemplo: la $p(A, D)$ viene dada por:

$$p_k(r,s) = \frac{(\tau_{rs})^{\alpha}.(\eta_{rs})^{\beta}}{\sum_{u \in J_k(r)}[(\tau_{ru})^{\alpha}.(\eta_{ru})^{\beta}]}$$

$$\eta_{rs} = \frac{1}{d_{rs}}$$

- α=1 y $\beta = 1$

$$p(A,D) = \frac{2*\frac{1}{3}}{\left(2*\frac{1}{3}\right)+\left(3*\frac{1}{15}\right)+\left(8*\frac{1}{7}\right)}$$

$p(A, D)$= 0.33

- α=0 y $\beta = 1$

$p(A, D)$= 0.61

- α=1 y $\beta = 0$

$p(A, D)$= 0.15

ACO: Fase de feromona

Evaporación de la feromona

- Cada vez que una hormiga k transiciona una arista r -> s, el rastro de feromona (τ_{rs}) se evapora, actualizando la tabla de transiciones de τ_{rs}:

Evaporación

$$\tau_{rs}^{k} = (1 - \rho).\tau_{rs} \qquad \textbf{(2)}$$

$(1 - \rho)$ es una forma artificial de evaporación de feromona

- $0 \leq \rho \leq 1$: coeficiente de evaporación de las feromonas.
- τ_{rs}: cantidad de feromona dejada por la hormiga en una transición anterior.

ACO: Depósito de feromona

- Después de la evaporación de feromonas, las hormigas depositan nuevas feromonas obedeciendo la siguiente regla (*k* representa una hormiga cualquiera)

Depósito

$$\tau_{rs}^{k} = \tau_{rs} + \sum_{k=1}^{m} \Delta\tau_{rs}^{k} \qquad \textbf{(3)}$$

$\sum_{k=1}^{m} \Delta\tau_{rs}^{k}$: depósito de rastro de feromonas dejado por otras k hormigas en la transición de r -> s

- $\Delta\tau_{rs}^{k}$: Es cero cuando la hormiga no ha utilizado esa transición y es un valor constante cuando la transición sí forma parte de la solución.

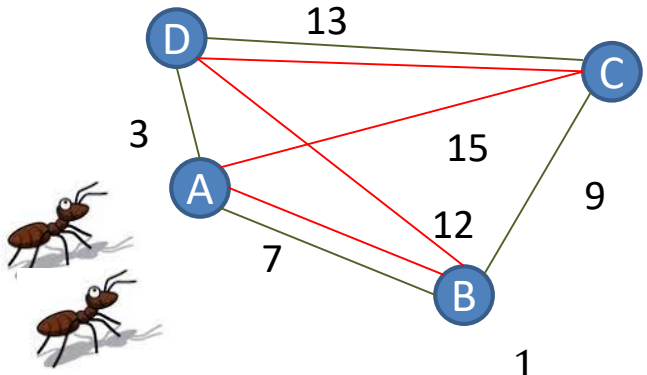
$$\Delta\tau_{rs}^{k} = \frac{1}{L_k}$$

- $L_k$es el largo del camino construido por la *k ésima* hormiga artificial y es calculado como la suma de los valores en los arcos pertenecientes al camino.
- m: número total de hormigas.

Matriz de transición

Ejemplo 1: Sin considerar evaporación $\tau_{rs}^{k} = \sum_{k=1}^{n} \Delta\tau_{rs}^{k}$

η_{rs}^{k} ➡ Grafo de Costos

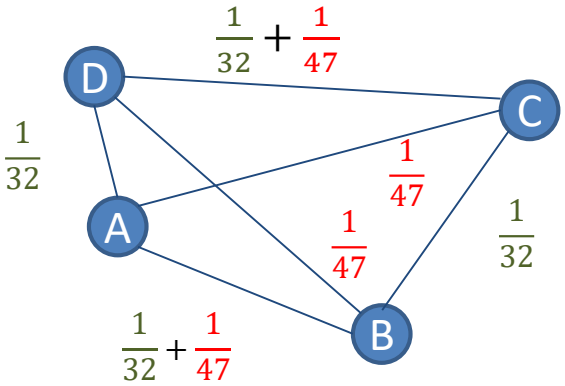


$$\Delta\tau_{rs}^{k} = \frac{1}{L_k}$$

$L_1 = 32 \longrightarrow \Delta\tau_{rs}^{L_1} = \frac{1}{32}$

$L_2 = 47 \longrightarrow \Delta\tau_{rs}^{L_2} = \frac{1}{47}$

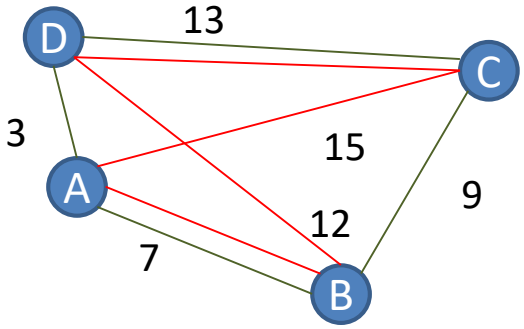
τ_{rs}^{k} ➡ Grafo de Feromonas



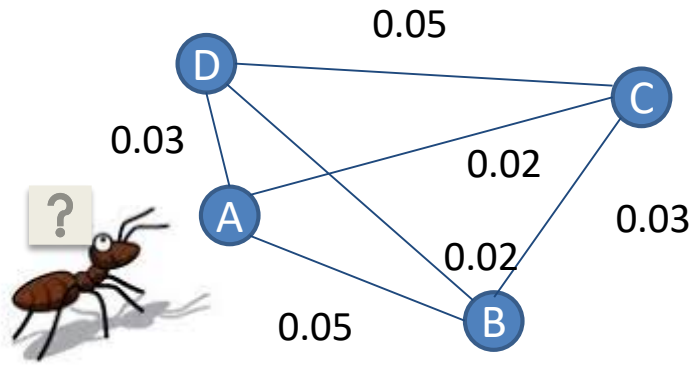
Matriz de transición

Ejemplo 1: Sin considerar evaporación $\tau_{rs}^{k} = \sum_{k=1}^{n} \Delta\tau_{rs}^{k}$

η_{rs}^{k} ➡ Grafo de Costos



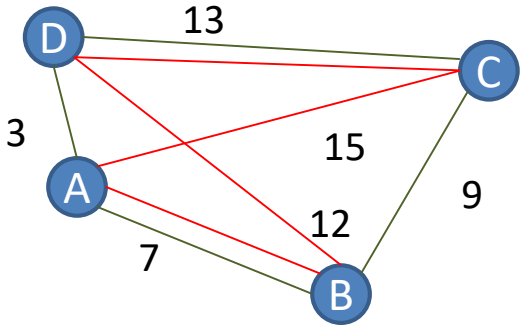
τ_{rs}^{k} ➡ Grafo de Feromonas



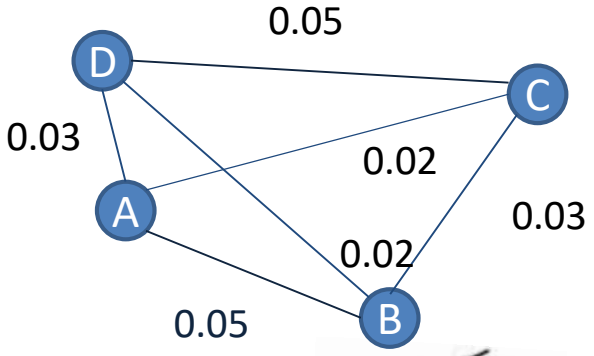
Matriz de transición

Ejemplo 1: Sin considerar evaporación $\tau_{rs}^{k} = \sum_{k=1}^{n} \Delta\tau_{rs}^{k}$

η_{rs}^{k} ➡ Grafo de Costos



τ_{rs}^{k} ➡ Grafo de Feromonas



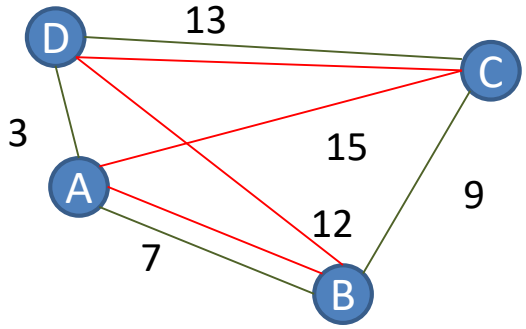
Mayor concentración de feromona

Matriz de transición

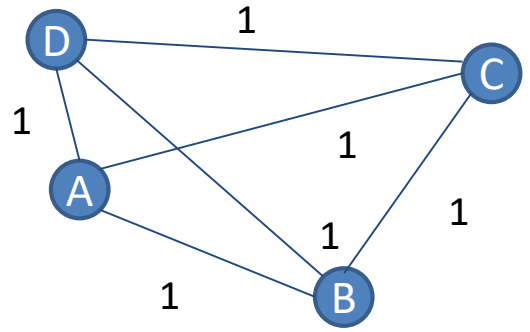
Ejemplo 2: Considerando evaporación

$$\tau_{rs}^{k} = (1-\rho).\tau_{rs} + \sum_{k=1}^{n} \Delta\tau_{rs}^{k} \qquad \rho = 0.5$$

η_{rs}^{k} ➡ Grafo de Costos



τ_{rs}^{k} ➡ Grafo de Feromonas

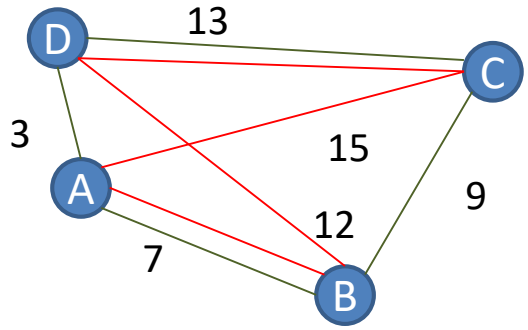


Matriz de transición

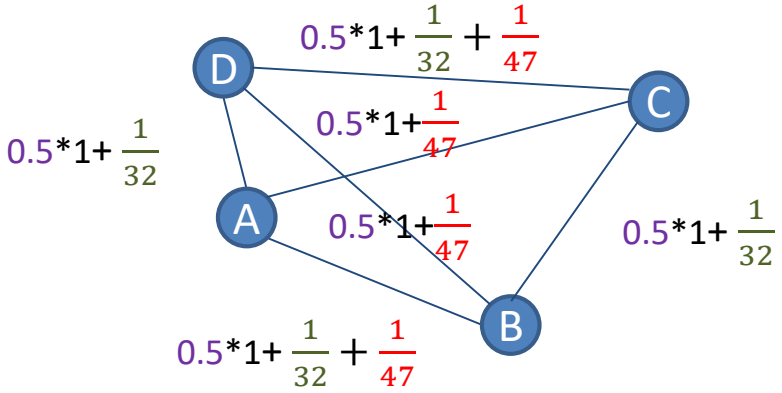
Ejemplo 2: Considerando evaporación

$$\tau_{rs}^{k} = (1-\rho).\tau_{rs} + \sum_{k=1}^{n} \Delta\tau_{rs}^{k} \qquad \rho = 0.5$$

η_{rs}^{k} ➡ Grafo de Costos



τ_{rs}^{k} ➡ Grafo de Feromonas



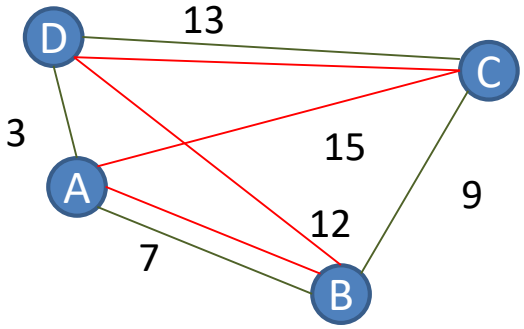
$L_1 = 32 \longrightarrow \Delta\tau_{rs}^{L_1} = \frac{1}{32}$

$L_2 = 47 \longrightarrow \Delta\tau_{rs}^{L_2} = \frac{1}{47}$

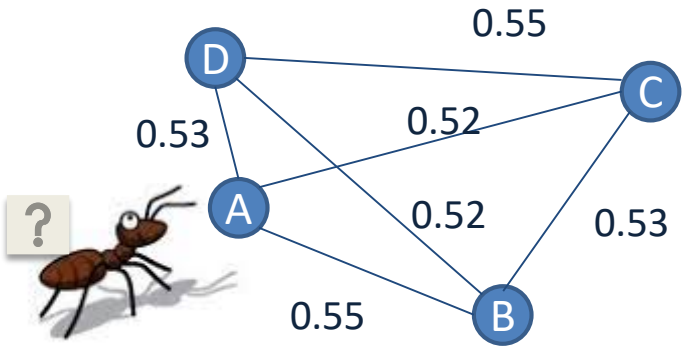
Matriz de transición

Ejemplo 2: Considerando evaporación

η_{rs}^{k} ➡ Grafo de Costos



τ_{rs}^{k} ➡ Grafo de Feromonas



$L_1 = 32 \longrightarrow \Delta\tau_{rs}^{L_1} = \frac{1}{32}$

$L_2 = 47 \longrightarrow \Delta\tau_{rs}^{L_2} = \frac{1}{47}$

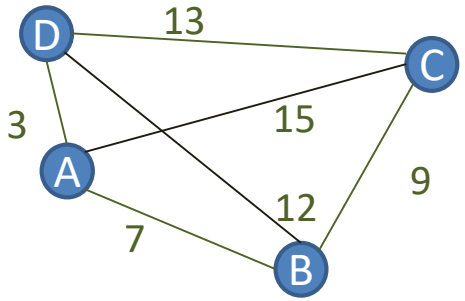
Calculando las probabilidades

Ejemplo 3:

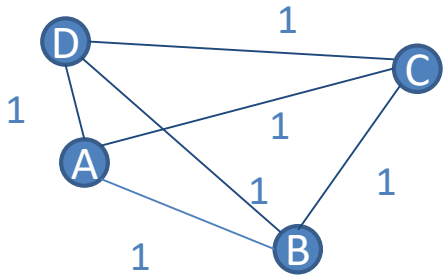
$$p_k(r,s) = \frac{(\tau_{rs})^{\alpha}.(\eta_{rs})^{\beta}}{\sum_{u \in J_k(r)}[(\tau_{ru})^{\alpha}.(\eta_{ru})^{\beta}]}$$

$$\eta_{rs} = \frac{1}{d_{rs}}, \alpha=1, \beta=1$$

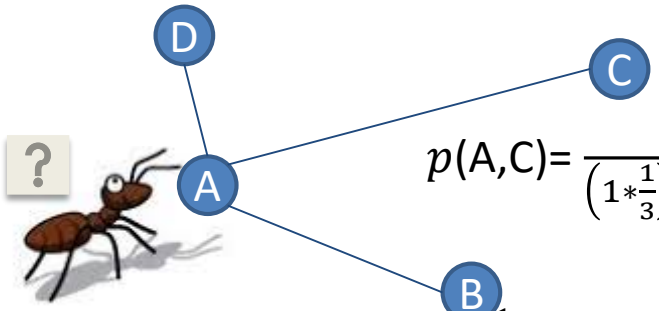
η_{rs}^{k} Grafo de Costos



τ_{rs}^{k} Grafo de Feromonas



$$p(A,D) = \frac{1*\frac{1}{3}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(1*\frac{1}{7}\right)} = 0.6140$$



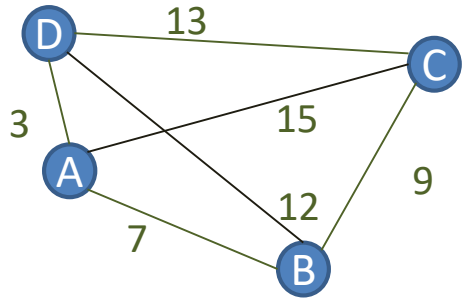
$$p(A,C) = \frac{1*\frac{1}{15}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(1*\frac{1}{7}\right)} = 0.1228$$

$$p(A,B) = \frac{1*\frac{1}{7}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(1*\frac{1}{7}\right)} = 0.2632$$

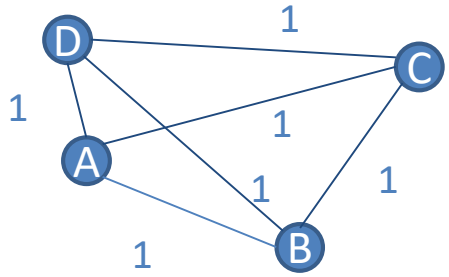
Calculando las probabilidades

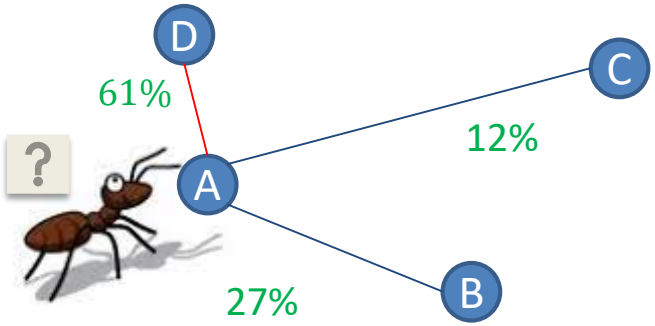
Ejemplo 3:

η_{rs}^{k} Grafo de Costos



τ_{rs}^{k} Grafo de Feromonas

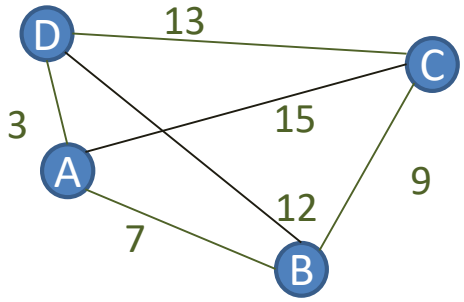




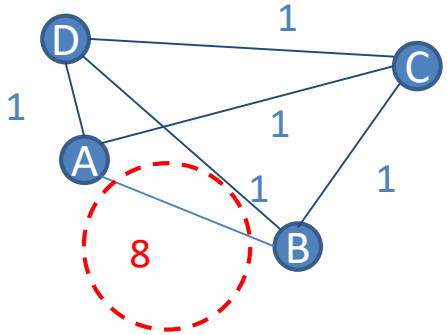
Calculando las probabilidades

Ejemplo 4:

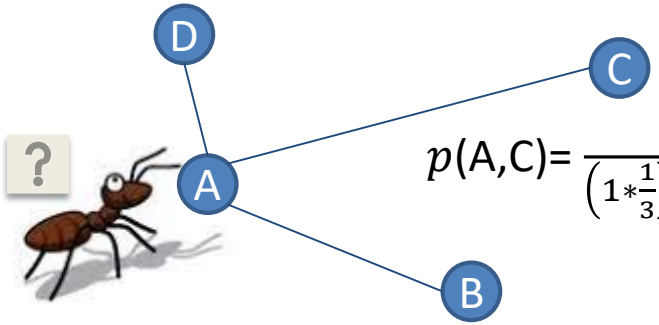
η_{rs}^{k} Grafo de Costos



τ_{rs}^{k} Grafo de Feromonas



$$p(A,D)=\frac{1*\frac{1}{3}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(8*\frac{1}{7}\right)}=0.2160$$

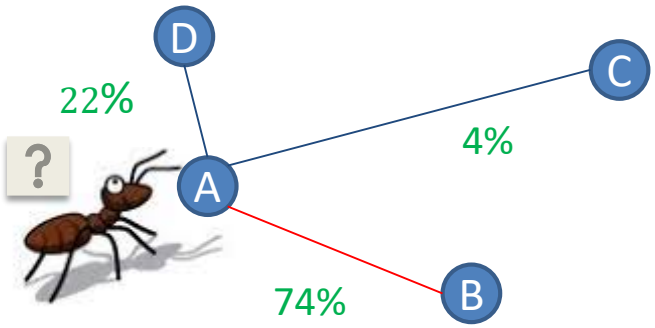


$$p(A,C)=\frac{1*\frac{1}{15}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(8*\frac{1}{7}\right)}=0.0432$$

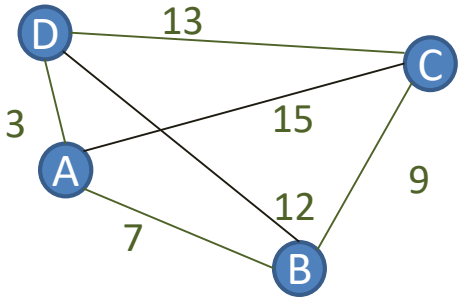
$$p(A,B)=\frac{8*\frac{1}{7}}{\left(1*\frac{1}{3}\right)+\left(1*\frac{1}{15}\right)+\left(8*\frac{1}{7}\right)}=0.7407$$

Calculando las probabilidades

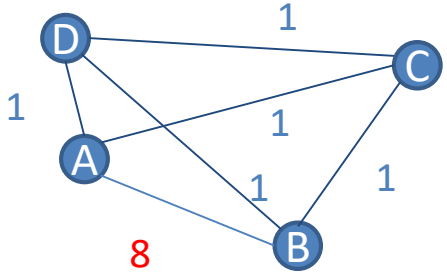
Ejemplo 4:



η_{rs}^{k} Grafo de Costos



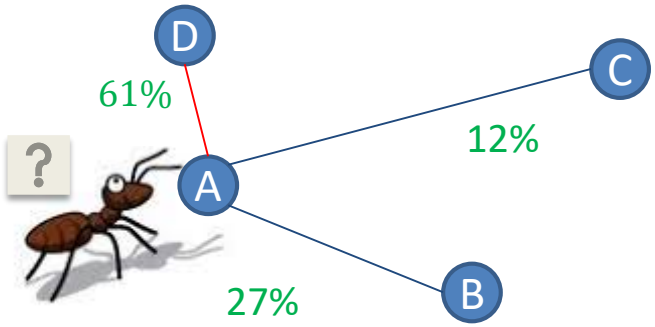
τ_{rs}^{k} Grafo de Feromonas

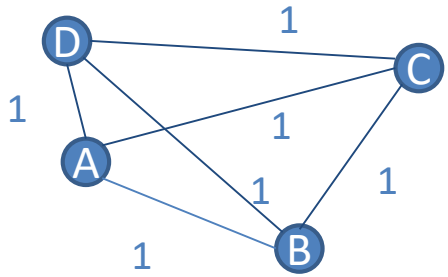


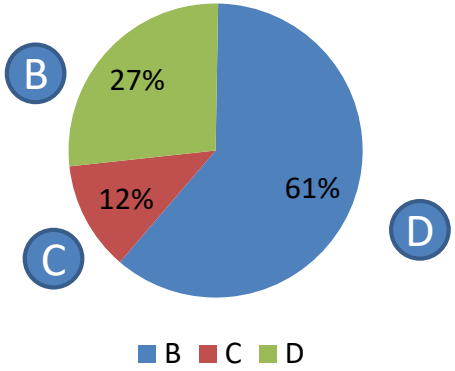
Calculando las probabilidades

Selección Probabilística

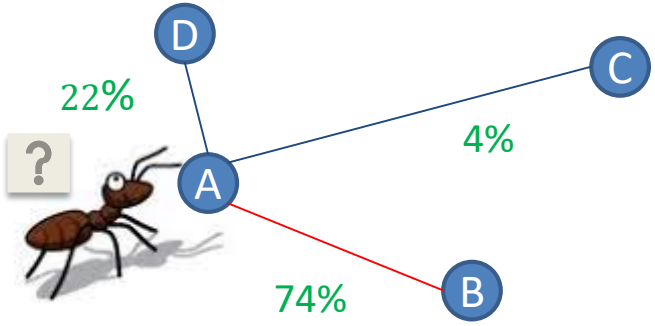
Ejemplo 3:

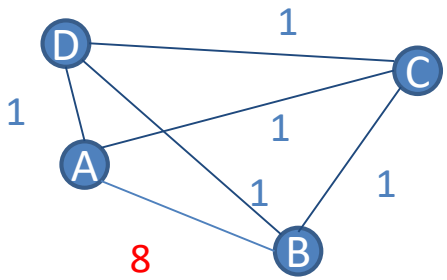


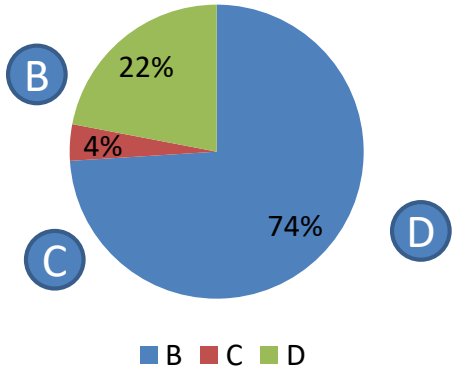




Ejemplo 4:







Pseudocódigo ACO

Procedimiento ACO

Inicializar todos los parámetros: $\alpha, \beta, \tau_0, \eta_k$

Mientras criterio de parada no sea satisfecho **hacer**

Posicionar cada hormiga en el nodo inicial

Repetir

Para cada hormiga $k = 1, \ldots, m$ **hacer**

Escoger el siguiente nodo aplicando la regla de transición de estados

$$p_k(r,s) = \frac{(\tau_{rs})^{\alpha} . (\eta_{rs})^{\beta}}{\sum_{u \in J_k(r)} [(\tau_{ru})^{\alpha} . (\eta_{ru})^{\beta}]}$$

Fin

Hasta que cada hormiga haya construido un $path$ (solución)

Aplicar evaporación de feromonas:

$$\tau_{rs} = (1 - \rho) . \tau_{rs}$$

Para cada hormiga $k = 1, \ldots, m$ **hacer**

Por cada arco $a_{rs} \epsilon pathk$

Calcular $\Delta\tau_{rs} = \frac{1}{L_k}$

Actualizar el rastro de feromonas: $\tau_{rs} = \tau_{rs} + \Delta\tau_{rs}^{k}$

Fin

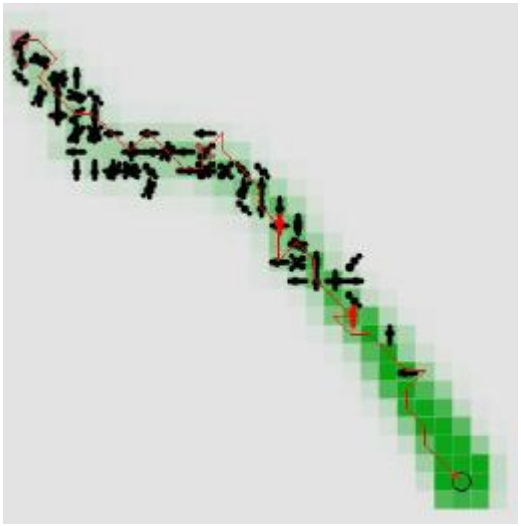
Fin

Actualizar el mejor $path$

Fin

ACO - Consideraciones

- En un sistema natural, a lo largo del tiempo, el rastro de feromonas se va evaporando, reduciendo así la atracción de las hormigas por un camino.
  - Mientras un camino sea más largo, más tiempo tardaran las hormigas que tomen ese camino y más rastro de feromona se evaporara. Así, el camino más corto es el más reforzado con el olor.
  - Sin embargo, en un modelo ACO, cuando un camino tiene mucho tráfico se suele incrementar el coeficiente de evaporación (ρ) para evitar caer en sub-óptimos locales (estagnación).



ACO - Consideraciones

- Debido a su naturaleza poblacional son fáciles de paralelizar.
- El principal problema de este algoritmo radica en la gran cantidad de parámetros que hay que elegir: pesos α y β en el cálculo de probabilidades y coeficiente de evaporación ρ.
- ACO tiene una ventaja principal sobre otros algoritmos como los AG y es que cuando el problema cambia dinámicamente, el algoritmo ACO puede cambiar dinámicamente también durante una ejecución en tiempo real.
  - Esto es muy interesante para aplicaciones en búsqueda de rutas en grandes sistemas de telecomunicaciones y redes de ordenadores o control del trafico urbano.

ACO - Consideraciones

- ACO puede ser usado en cualquier problema que pueda traducirse a búsqueda de un camino óptimo en un grafo, como por ejemplo:
  - Ordenación secuencial
  - Problemas de asignación cuadrática:
    - Distribución de recursos en línea de acopio.
    - Disposición de tiendas en un mall.
    - Distribución de Terminales de Aeropuertos
    - Edificación de Universidades/ Colegios, etc.
    - Asignación de horarios
    - Planificación de tareas
    - Cableado de Placas Electrónicas
- Sin embargo, en algunos casos esa traducción puede ser difícil o muy costosa.

Bibliografía

- Ivan Zelinka; Guarong Chen. **Evolutionary Algorithms, Swarm Dynamics and Complex Networks. Methodology Perpectives and Implementation**. Springer-Verlag, 2018
- Raúl Benítez; Gerard Escudero;Samir Kanaan. **Inteligencia artificial avanzada**. Universitat Oberta de Catalunya (UOC), 2016
- Silja Meyer-Nieberg;Nadiia Leopold;Tobias Uhlig. **Natural Computing for Simulation-Based Optimization and Beyond**. Springer, 2019
- M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.
- Marco Dorigo, Mauro Birattari, and Thomas Stutzle, Universite Libre de Bruxelles, BELGIUM.