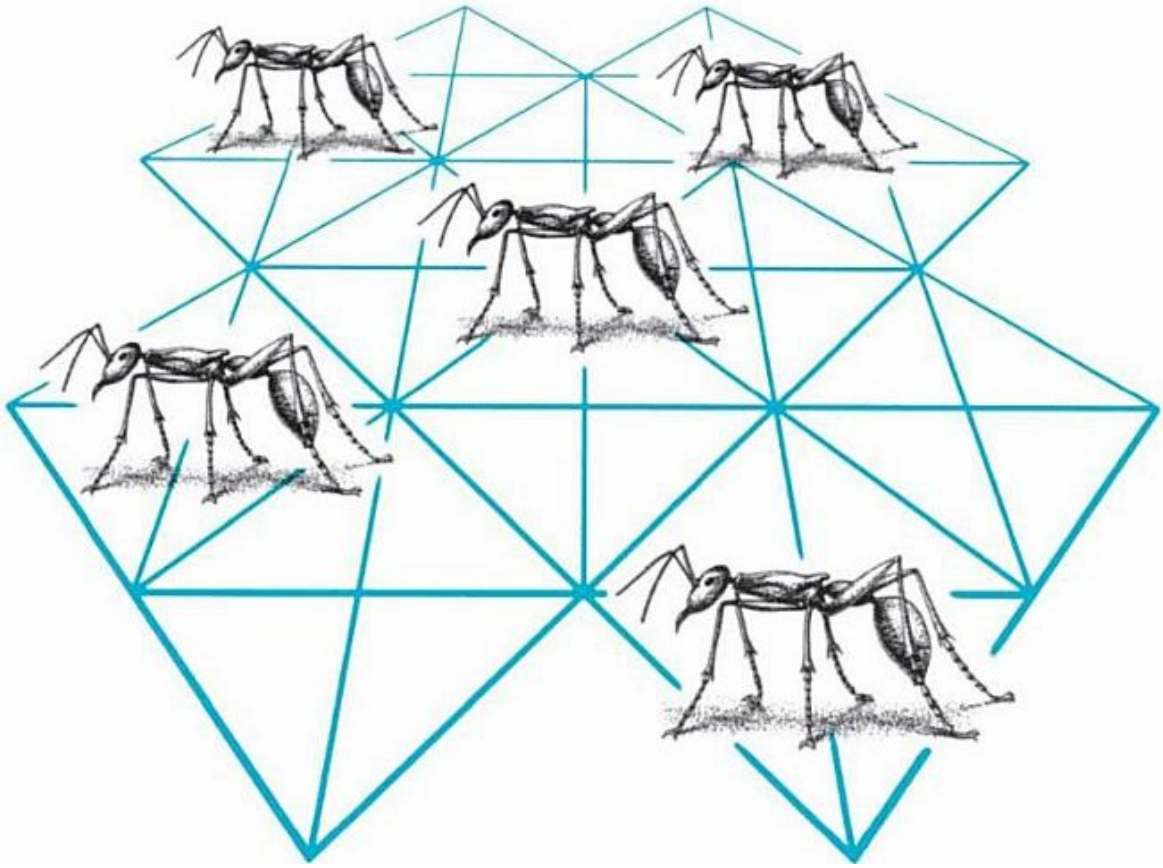
Ant Colony Optimization



A Seminar Report on

“Ant Colony Optimization”

A Seminar submitted in partial fulfilment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE ENGINEERING

Presented By

Ranjith Kumar A (06J11A0534)

Department of computer science engineering

HITECH COLLEGE OF ENGG & TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Himayathnagar, C.B.Post, Moinabad, Hyderabad-5000

075.

CERTIFICATE

This is to certify that the Seminar Report on ***"Ant Colony Optimization"****, is a bonafide Seminar work done by* ***Ranjith Kumar A (06J11A0534),*** *in partial fulfillment for the award of the degree Bachelor of Technology in "Computer Science engineering" J.N.T.U Hyderabad during the year 2010.*

Y.V.S Pragathi

M.Tech

Head of CSE Department

Abstract

Ant Colony Optimization (ACO) has been successfully applied to those combinatorial optimization problems which can be translated into a graph exploration. Artificial ants build solutions step by step adding solution components that are represented by graph nodes. The existing ACO algorithms are suitable when the graph is not very large (thousands of nodes) but is not useful when the graph size can be a challenge for the computer memory and cannot be completely generated or stored in it. In this paper we study a new ACO model that overcomes the difficulties found when working with a huge construction graph. In addition to the description of the model, we analyze in the experimental section one technique used for dealing with this huge graph exploration. The results of the analysis can help to understand the meaning of the new parameters introduced and to decide which parameterization is more suitable for a given problem. For the experiments we use one real problem with capital importance in Software Engineering: refutation of safety properties in concurrent systems. This way, we foster an innovative research line related to the application of ACO to formal methods in Software Engineering.

CONTENTS

PAGENO

1 Introduction:

1.1 Swarm Intelligence:

Swarm intelligence (SI) describes the collective behaviour of decentralized, self-Organized systems, natural or artificial. The concept is employed in work on artificial intelligence. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems.

Swarm intelligence is the discipline that deals with natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization. In particular, the discipline focuses on the collective behaviours that result from the local interactions of the individuals with each other and with their environment. Examples of systems studied by swarm intelligence are colonies of ants and termites, schools of fish, flocks of birds, herds of land animals. Some human artefacts also fall into the domain of swarm intelligence, notably some multi-robot systems, and also certain computer programs that are written to tackle optimization and data analysis problems.

Emphasis is given to such topics as the modelling and analysis of collective biological systems; application of biological swarm intelligence models to real-world problems; and theoretical and empirical research in ant colony optimization, particle swarm optimization, swarm robotics, and other swarm intelligence algorithms. Articles often combine experimental and theoretical work.

1.2 Ant Colony:

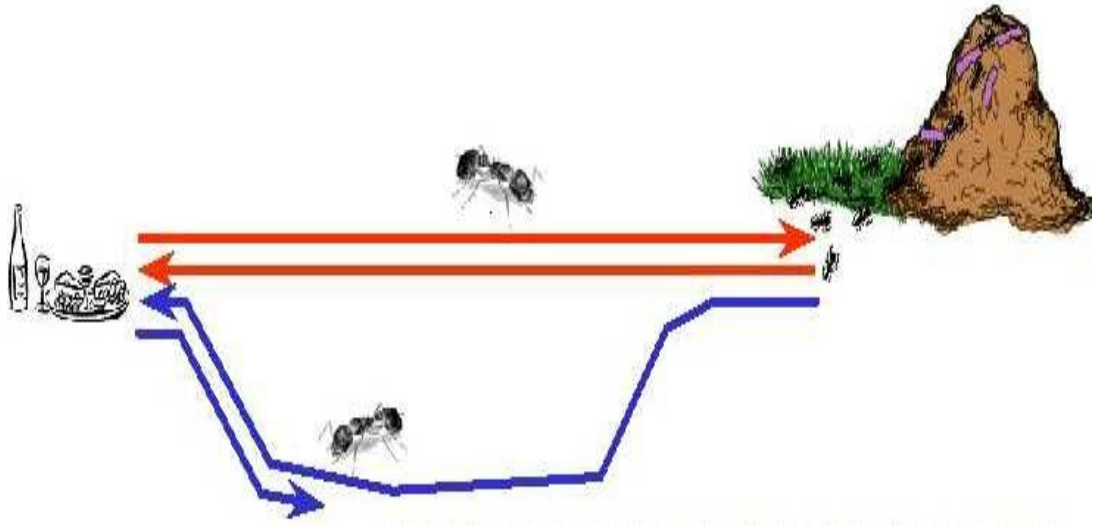
The complex social behaviours of ants have been much studied by science, and computer scientists are now finding that these behaviour patterns can provide models for solving difficult combinatorial optimization problems. The attempt to develop algorithms inspired by one aspect of ant behaviour, the ability to find what computer scientists would call shortest paths, has become the field of ant colony optimization (ACO), the most successful and widely recognized algorithmic technique based on ant behaviour. This book presents an overview of this rapidly growing field, from its theoretical inception to practical applications, including descriptions of many available ACO algorithms and their uses.

The book first describes the translation of observed ant behaviour into working optimization algorithms. The ant colony metaheuristic is then introduced and viewed in the general context of combinatorial optimization. This is followed by a detailed description and guide to all major ACO algorithms and a report on current theoretical findings. The book surveys ACO applications now in use, including routing, assignment, scheduling, subset, machine learning, and bioinformatics problems. AntNet, an ACO algorithm designed for the network routing problem, is described in detail. The authors conclude by summarizing the progress in the field and outlining future research directions. Each chapter ends with bibliographic material, bullet points setting out important ideas covered in the chapter, and exercises. *Ant Colony Optimization* will be of interest to academic and industry researchers, graduate students, and practitioners who wish to learn how to implement ACO algorithms.

1.3 Real Ant Behavior :

Natural behaviour of ants have inspired scientists to mimic insect operational methods to solve real-life complex problems such as Travelling sales man problem, Quadratic assignment problem, Network model, Vehicle routing. By observing ant behaviour, scientists have begun to understand their means of communication

Ants communicate with each other through tapping with the antennae and smell. They are considered, together with the bees, as one of the most socialized animals. They have a perfect social organization, and each type of individual specializes in a specific activity within the colony. They are thought by many as having a collective intelligence, and each ant is considered then as an individual cell of a bigger organism. Ants wander randomly & on finding food return to their colony while laying "PHEROMONE TRIALS".

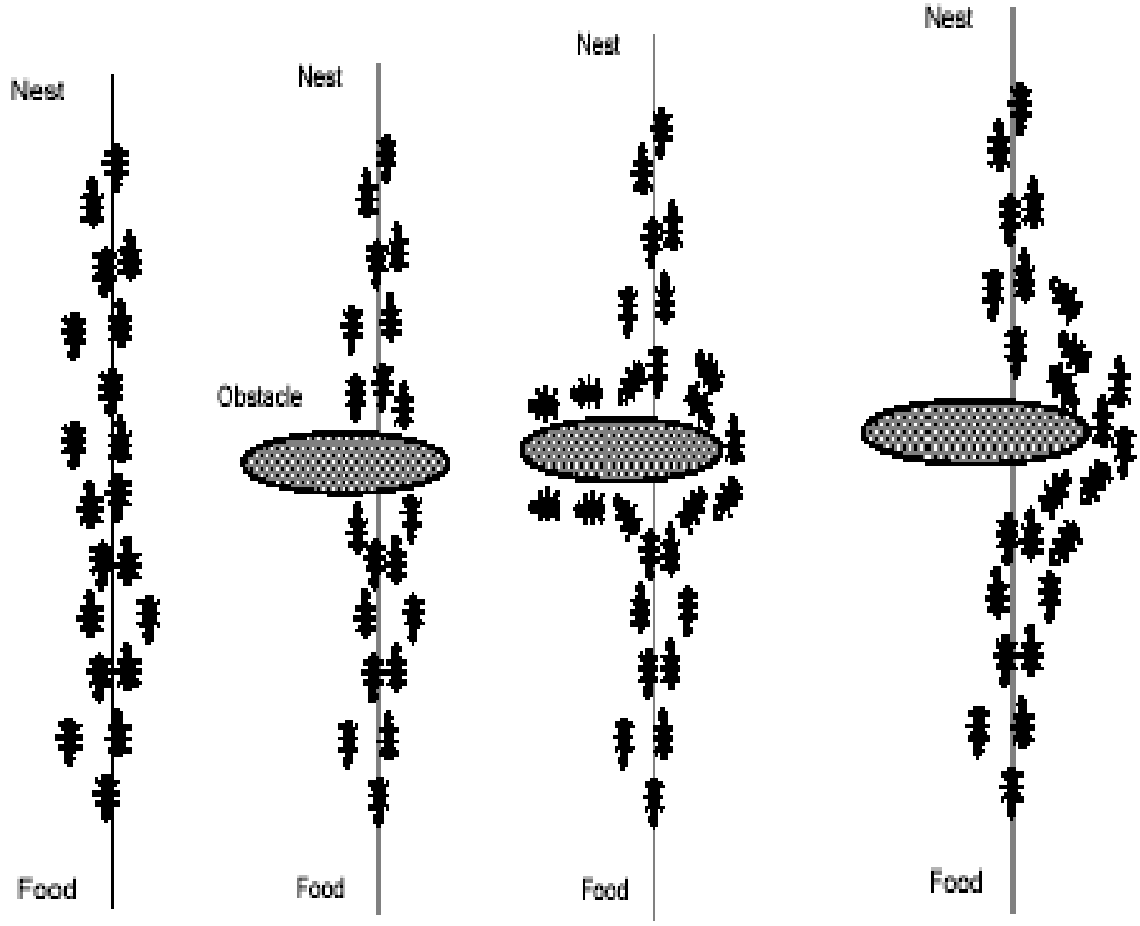


If other Ants find such paths they do not travel randomly but follow the Pheromone trail. Ants secrete pheromone while travelling from the nest to food and vice versa in order to communicate with each other to find shortest path.

Pheromone is a highly volatile substance which starts to evaporate, more the time taken by the ant to travel to and fro more time the pheromone have to evaporate. A shortest path gets marched over faster and thus the pheromone density remains high. If one of the ant finds the shortest path from colony to food source other ants are more likely to follow the same path.

Behaviour of ant in presence of an obstacle:

Ants are forced to decide whether they should go left or right. The choice that is made is a random one. Pheromone accumulation is Faster on shortest path.



Ants:

One of the first researchers to investigate the social behaviour of insects was the French entomologist Pierre-Paul Grasse. In the forties and fifties of the 20-th century, he was observing the behaviour of termites { in particular, the *Bellicositermes natalensis* and *Cubitermes* species. He discovered [26] that these insects are capable to react to what he called \significant stimuli", signals that activate a genetically encoded reaction. He observed that the effects of these reactions can act as new significant stimuli for both the insect that produced them and for the other insects in the colony. Grasse used the term *stigmergy* to describe this particular type of indirect communication in which the \workers are stimulated by the performance they have achieved". The two main characteristics of stigmergy that di®erentiate it from other means of communication are:

- the physical, non-symbolic nature of the information released by the communicating insects, which corresponds to a modification of physical environmental states visited by the insects and
- the local nature of the released information, which can only be accessed by those insects that visit the place where it was released (or its immediate neighbourhood).

Examples of stigmergy can be observed in colonies of ants. In many ant species, ants walking to, and from, a food source deposit on the ground a substance called *pheromone*. Other ants are able to smell this pheromone, and its presence influences the choice of their path i.e., they tend to follow strong pheromone.

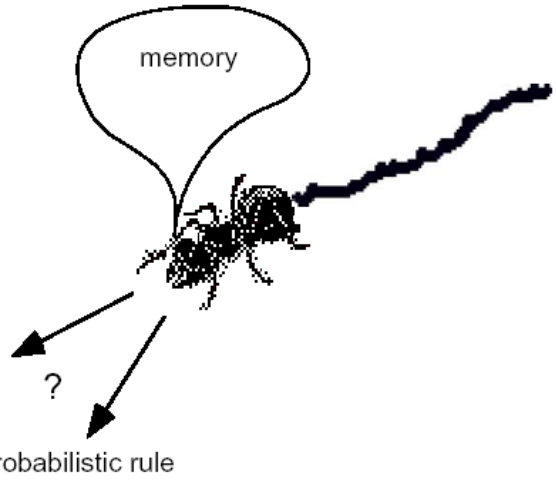
2 Ant Colony Optimization(ACO):

Ant Colony Optimization (ACO) is a paradigm for designing metaheuristic algorithms for combinatorial optimization problems. The first algorithm which can be classified within this framework was presented in 1991 and, since then, many diverse variants of the basic principle have been reported in the literature.

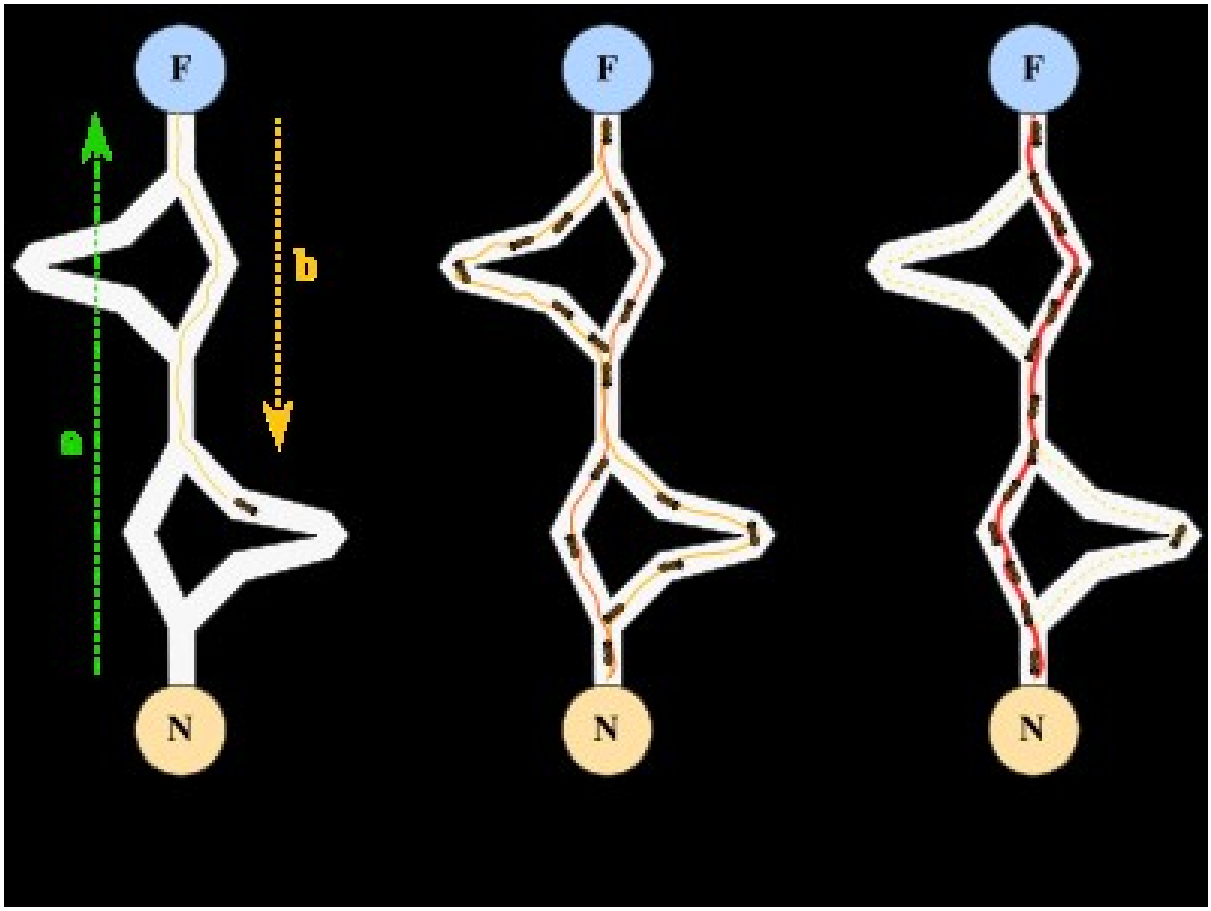
The essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with posterior information about the structure of previously obtained good solutions. Metaheuristic algorithms are algorithms which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one.

The metaheuristic part permits the low-level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbour solutions to consider in local search (as is the case of simulated annealing [or tabu), or by combining elements taken by different solutions (as is the case of evolution strategies and genetic or bionomic algorithms).

The characteristic of ACO algorithms is their explicit use of elements of previous solutions. In fact, they drive a constructive low-level solution, as GRASP does, but including it in a population framework and randomizing the construction in a Monte Carlo way. A Monte Carlo combination of different solution elements is suggested also by Genetic Algorithms , but in the case of ACO the probability distribution is explicitly defined by previously obtained solution components. The particular way of defining components and associated probabilities is problem- specific, and can be designed in different ways, facing a trade-off between the specificity of the information used for the conditioning and the number of solutions which need to be constructed before effectively biasing the probability dis- tribution to favour the emergence



of good solutions. Different applications have favoured either the use of conditioning at the level of decision variables, thus requiring a huge number of iterations before getting a precise distribution, or the computational efficiency, thus using very coarse conditioning information. The chapter is structured as follows. Section 2 describes the common elements of the heuristics following the ACO paradigm and outlines some of the variants proposed. Section 3 presents the application of ACO algorithms to a number of different combinatorial optimization problems and it ends with a wider overview of the problem attacked by means of ACO up to now. Section 4 outlines the most significant theoretical results so far published about convergence properties of ACO variants.



ACO is a class of algorithms, whose first member, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo . The main underlying idea, loosely inspired by the behaviour of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. The collective behaviour emerging

from the interaction of the different search threads has proved effective in solving combinatorial optimization (CO) problems.

we use the following notation. A combinatorial optimization problem is a problem defined over a set **C** = c*1*, ... , c*n* of basic *components*. A subset **S** of components represents a *solution* of the problem; **F** ⊆ 2***C*** is the subset of *feasible solutions*, thus a solution **S** is feasible if and only if **S** ∈ **F**. A *cost function z* is defined over the solution domain, z : 2***C*** ▪ **R**, the objective being to find a minimum cost feasible solution S*, i.e., to find S*: S* ∈ **F** and z(S*) ≤ z(**S**),

∀**S**∈**F**. Given this, the functioning of an ACO algorithm can be summarized as follows (see also [27]). A set of computational concurrent and asynchronous agents (a colony of ants) moves through states of the problem corresponding to partial solutions of the problem to solve. They move by applying a stochastic local decision policy based on two parameters, called *trails* and *attractiveness*. By moving, each ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. This pheromone information will direct the search of the future ants.

Furthermore, an ACO algorithm includes two more mechanisms*: trail evaporation* and, optionally, *daemon actions*. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective .More specifically, an *ant* is a simple computational agent, which iteratively constructs a solution for the instance to solve. Partial problem solutions are seen as *states*. At the core of the ACO algorithm lies a loop, where at each iteration, each ant *moves* (performs a *step*) from a state ι to another one ψ, corresponding to a more complete partial solution. That is, at each step σ, each ant *k* computes a set **A***k* σ(ι) of feasible expansions to its current state, and moves to one of these in probability. The probability distribution is specified as follows. For ant *k*, the probability pιψ*k* of moving from state ι to state ψ depends on the combination of two values:

• the *attractiveness* ηιψ of the move, as computed by some heuristic indicating the *a priori* desirability of that move.

• the *trail level* τιψ of the move, indicating how proficient it has been in the past to make that particular move: it represents therefore an *a posterior* indication of the desirability of that move.

Trails are *updated* usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively. The general framework just presented has been specified in different ways by the authors working on the ACO approach. The remainder of Section 2 will outline some of these contributions.

The ant system simply iterates a main loop where *m* ants construct in parallel their solutions, thereafter updating the trail levels. The performance of the algorithm depends on the correct tuning of several parameters, namely: α, β, relative importance of trail and attractiveness, ρ, trail persistence, $\tau_{ij}(0)$, initial trail level, *m*, number of ants, and Q, used for defining to be of high quality solutions with low cost. The algorithm is the following.

1. {Initialization}

Initialize $\tau_{\iota\psi}$ **and** $\eta_{\iota\psi}$**,** $\forall(\iota\psi)$**.**

2. {Construction}

For each ant *k* (currently in state ι**) do**

repeat

choose in probability the state to move into.

append the chosen move to the *k*-th ant's set tabu*k*.

until ant *k* has completed its solution.

end for

3. {Trail update}

For each ant move ($\iota\psi$**) do**

compute $\Delta\tau_{\iota\psi}$

update the trail matrix.

end for

4. {Terminating condition}

If not(end test) go to step 2

3 Applications OF ACO:

3.1 Travelling sales man:

The TSP is a very important problem in the context of Ant Colony Optimization because it is the problem to which the original AS was first applied, and it has later often been used as a benchmark to test a new idea and algorithmic variants.

OBJECTIVE:

Given a set of *n* cities, the Traveling Salesman Problem requires a salesman to find the shortest route between the given cities and return to the starting city, while keeping in mind that each city can be visited only once

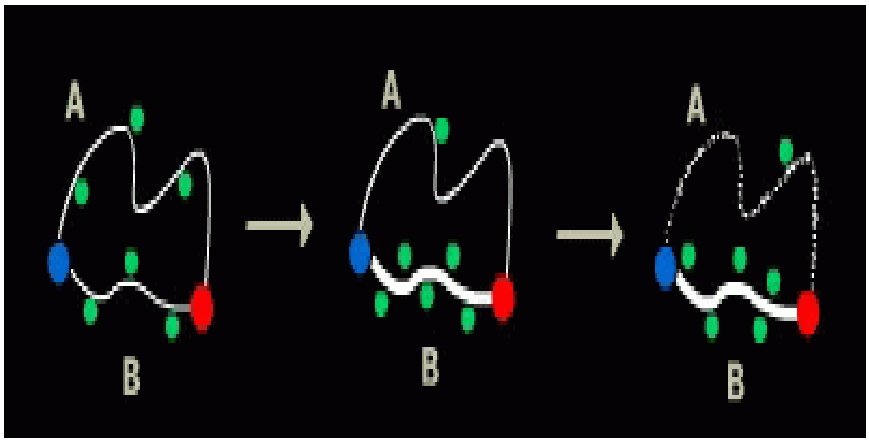
The TSP was chosen for many reasons:

- It is a problem to which the ant colony metaphor
- It is one of the most studied NP-hard problems in the combinatorial optimization
- it is very easily to explain. So that the algorithm behavior is not obscured by too many technicalities.

Since the route B is shorter, the ants on this path will complete the travel more times and thereby lay more pheromone over it.

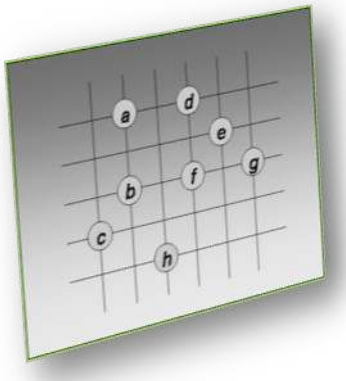
The pheromone concentration on trail B will increase at a higher rate than on A, and soon the ants on route A will choose to follow route B

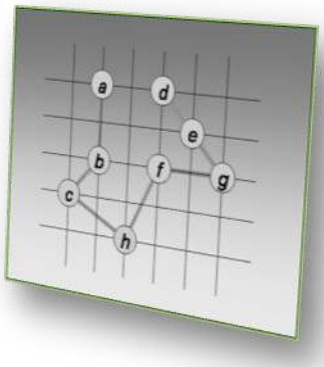
Since most ants will no longer travel on route A, and since the pheromone is volatile, trail A will start evaporating Only the shortest route will remain



WHY TSP IS DIFFICULT TO SOLVE

Finding best solution may entail an exhaustive search for all combination of cities, this can be prohibitive as "N" gets large. Heuristic like greedy methods doesn't guarantee optimal solutions.

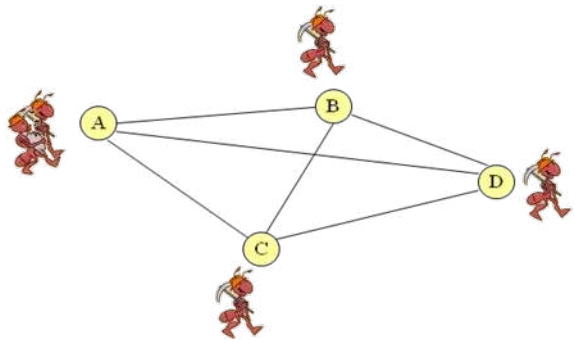




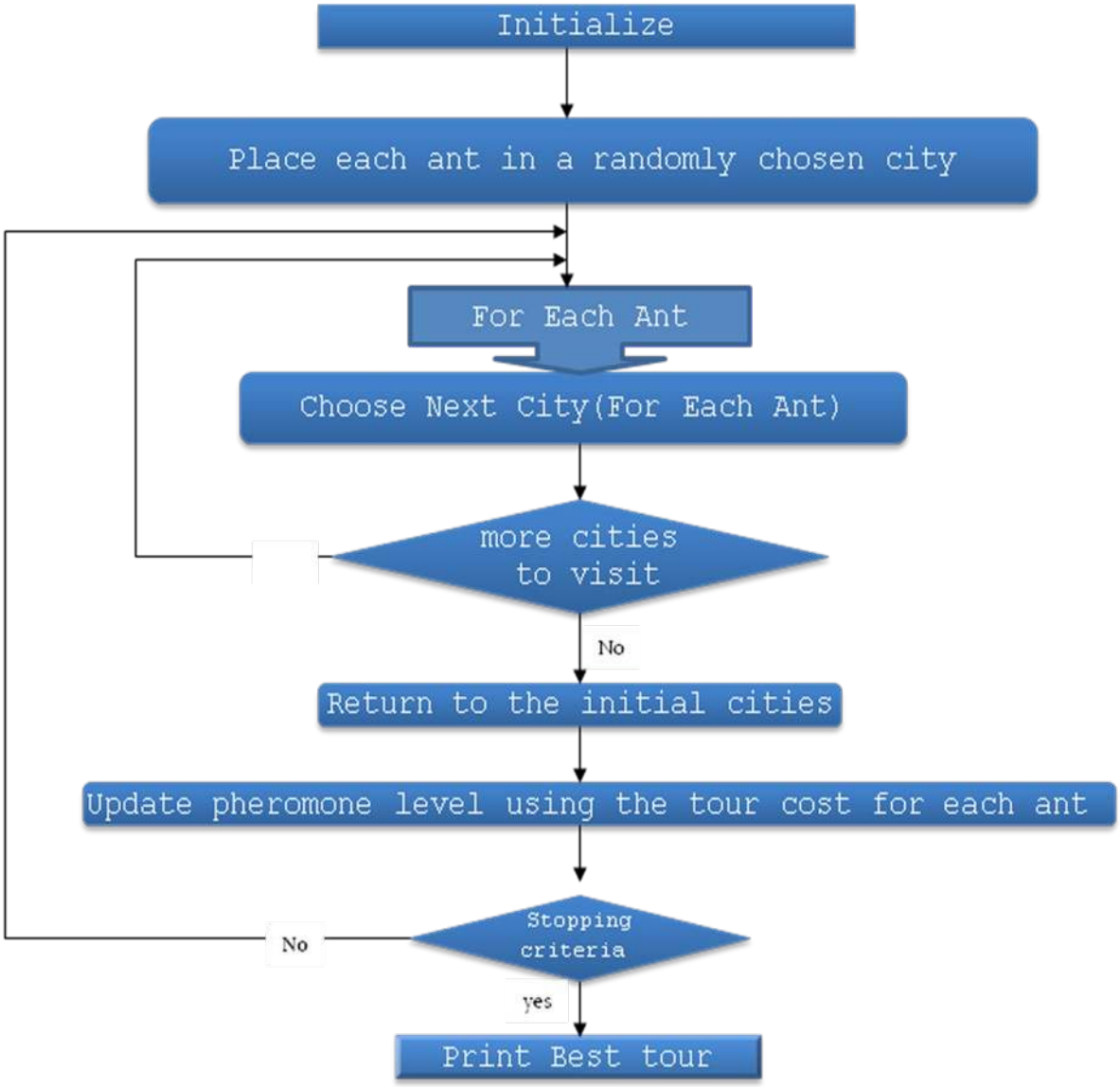
Ant colony optimization in TSP:

The meta heuristic Ant Colony Optimization (ACO) is an optimization algorithm successfully used to solve many NP hard optimization problems introduced in ACO . ACO algorithms are a very interesting approach to find minimum cost paths in graphs especially when the connection costs in the graphs can change over time, i.e. when the problems are dynamic. The artificial ants have been successfully used to solve the (conventional) Traveling Salesman Problem (TSP) , as well as other NP hard optimization problems, including applications in quadratic assignment or vehicle routing.

The algorithm's based on the fact that ants are always able to find the shortest path between the nest and the food sources, using information of the pheromones previously laid on the ground by other ants in the colony. When an ant is searching for the nearest food source and arrives at several possible trails, it tends to choose the trail with the largest concentration of pheromones, with a certain probability p. After choosing the trail, it deposits another pheromone, increasing the concentration of pheromones in this trail. The ants return to the nest using always the same path, depositing another portion of pheromone in the way back. Imagine then, that two ants at the same location choose two different trails at the same time. The pheromone concentration on the shortest way will increase faster than the other: the ant that chooses this way, will deposit more pheromone in a smaller period of time, because it returns earlier. If a whole colony of thousands of ants follows this behaviour, soon the concentration of pheromone on the shortest path will be much higher than the concentration in other paths. Then the probability of choosing any other way will be very small and only very few ants among the colony will fail to follow the shortest path. There is another phenomenon related with the pheromone concentration since it is a chemical substance, it tends to evaporate, so the concentration of pheromones vanishes along the time. In this way, the concentration of the less used paths will be much lower than that of the most used ones, not only because the concentration increases on the other paths, but also because its own concentration decreases.



Flow chart for TSP using ACO



Ant Colonies Optimization Algorithm:

procedure Ant colony algorithm
Set for every pair (i, j): Tij = Tmax
Place the g ants
    For i = 1 to N:
    Build a complete tour
        For j = 1 to m
            For k = 1 to g
            Choose the next node using pk ij in (2)
            Update the **tabu list** T
        End
    End
Analyze solutions
    For k = 1 to g
        Compute performance index fk
        Update globally Tij(t + m × g) using (3)
    End
End

Euclidean distance between two locations dij is used as heuristic. However, within a city, the traveling time tij between two machines is more relevant than distance, due to traffic reasons. Therefore, the heuristic function is given by _ = (tij − tmin)/(tmax − tmin), where tij is the estimated traveling time between location i and location j and tmin = min tij and tmax = max tij are the minimum and maximum travelling times considered. In this way, the heuristic matrix _ entries are always restricted to the interval [0, 1]. The objective function to minimize, fk(t), is simply the sum of travelling time between all the visited locations:

Rules for Transition Probability

1. Whether or not a city has been visited

Use of a **memory**(tabu list): J_i^k : set of all cities that are to be visited

2. $\eta_{ij} = 1/d_{ij}$ **visibility**: Heuristic desirability of choosing city j when in city i.

3.Pheromone trail: $\tau_{ij}(t)$ This is a global type of information

Transition probability for ant k to go from city i to city j while building its route.

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

a =

Trial visibility is $\eta_{ij} = 1/d_{ij}$

The intensity in the probabilistic transition is α

The visibility of the trial segment is β

The trail persistence or evaporation rate is given as ρ

Trail intensity is given by value of τ_{ij}

TSP Applications

• Lots of practical applications

• Routing such as in trucking, delivery, UAVs

• Manufacturing routing such as movement of parts along

manufacturing floor or the amount of solder on circuit board

• Network design such as determining the amount of cabling required

• Two main types

– Symmetric

– Asymmetric

TSP Heuristics

• Variety of heuristics used to solve the TSP

• The TSP is not only theoretically difficult it is also difficult in practical

application since the tour breaking constraints get quite numerous

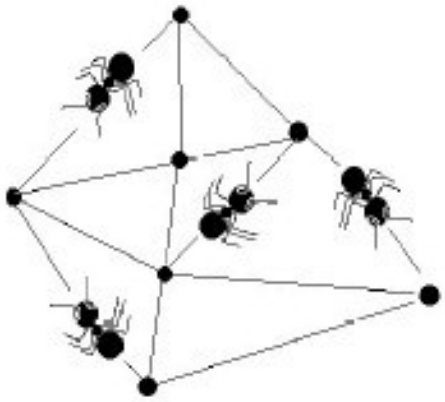
• As a result there have been a variety of methods proposed for the TSP

• Nearest Neighbour is a typical greedy approach

Advantages:

• Positive Feedback accounts for rapid discovery
of good solutions
• Distributed computation avoids premature
convergence
• The greedy heuristic helps find acceptable
solution in the early solution in the early stages
of the search process.
• The collective interaction of a population of
agents.

3.2 Quadratic Assignment Problem(QAP):



The **quadratic assignment problem** (**QAP**) is one of fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics, from the category of the facilities location problems.

There are a set of *n* facilities and a set of *n* locations. For each pair of locations, a *distance* is specified and for each pair of facilities a*weigh* or *flow* is specified (e.g., the amount of supplies transported between the two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows

The formal definition of the quadratic assignment problem is

> Given two sets, P ("facilities") and L ("locations"), of equal size, together with a weight function $w : P \times P \to \mathrm{R}$ and a distance function $d : L \times L \to \mathrm{R}$. Find the bijection $f : P \to L$ ("assignment") such that the cost function:
>
> $$\sum_{a,b \in P} w(a,b) \cdot d(f(a), f(b))$$
>
> is minimized.
>
> > Usually weight and distance functions are viewed as square real-valued matrices, so that the cost function is written down as:
>
> $$\sum_{a,b \in A} w_{a,b} d_{f(a),f(b)}$$

Problem is:

• Assign n activities to n locations (campus and mall layout).

• D= $d_{i,j}$, $\lfloor d_{i,j} \rfloor_{n,n}$, distance from location i to location j

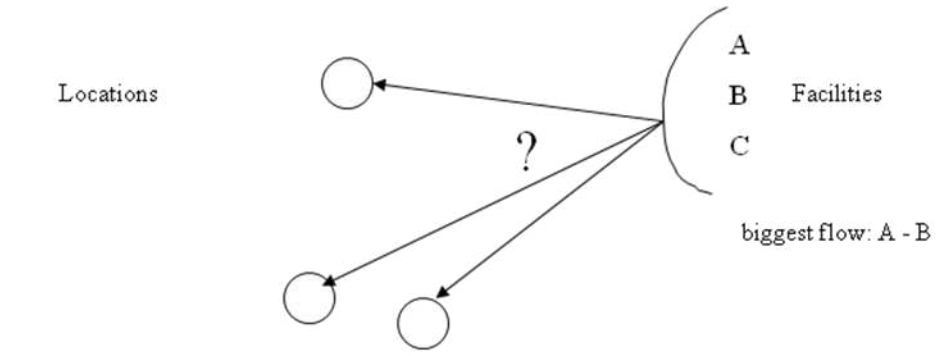
• F= $\lfloor f_{h,k} \rfloor_{n,n} f_{i,j}$ flow from activity h to activity k

• Assignment is permutation Π

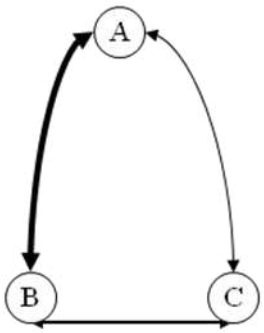
• Minimize: $C(\pi) = \sum_{i,j=1}^{n} d_{ij} f_{\pi(i)\pi(j)}$

• It's NP hard

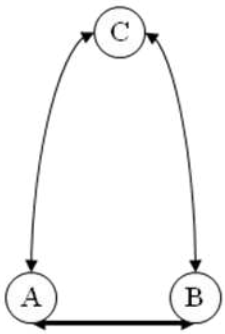
QAP Example



How to assign facilities to locations ?



Higher cost



Lower cost

Algorithms

1) Ant System for the QAP: Ant System (AS) uses ants in order to construct a solution from scratch. The algorithm uses a heuristic information on the potential quality of a local assignment that is determined as follows: two vectors d and f whose components are the sum of the distances, resp. flows from location, resp. facility i to all other locations, resp. facilities are computed. This leads to a coupling matrix E = f · dT where eij = fi · dj . Thus, _ = 1/eij denotes the heuristic desirability of assigning facility i to location j. A solution is constructed by using both this heuristic information and information provided by previous ants using pheromones. At each step an ant k assigns the next still unassigned facility i to a location j belonging to the feasible neighbourhood of the node i, i.e. the locations that are still free, with a probability pk ij given by Equation 2.

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k \qquad (2)$$

After their run, the ants update the pheromone information Tij :

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k \qquad (3)$$

This algorithm uses an evaporation rate of (1 − p) in order to forget previous bad choices at the cost of loosing useful information too. $\Delta\tau_{ij}^k$ is the amount of pheromone ant k deposits on the edge (i, j):

$$\Delta\tau_{ij}^k = \begin{cases} Q/J_\psi^k & \text{if } i \text{ is assigned to } j \text{ in } k \\ 0 & \text{otherwise} \end{cases}$$

The algorithm parameters are the number of ants n, the weight given to either heuristic or pheromone information's _, _ and the maximal amount of laid pheromone Q. *2) The* MAX − MIN *Ant System:* MAX − MIN Ant System (MMAS) is an improvement over AS that allows only one ant to add pheromone. The pheromone trails are initialized to the upper trail limit, which cause a higher exploration at the start of algorithm. Finally, methods to increase the diversification of the search can be used, for example by reinitialising the pheromone trails to _max if the algorithm makes no progress. In MMAS, the ant k assigns the facility i to the location j with a probability pk ij given by formula 4. MMAS does not use any heuristic information but is coupled with a local search for every ant.

$$p_{ij}^k = \frac{\tau_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \tau_{il}(t)} \qquad \text{if } j \in \mathcal{N}_i^k \qquad (4)$$

SIMPLIFIED CRAFT (QAP)

Simplification Assume all departments have equal size

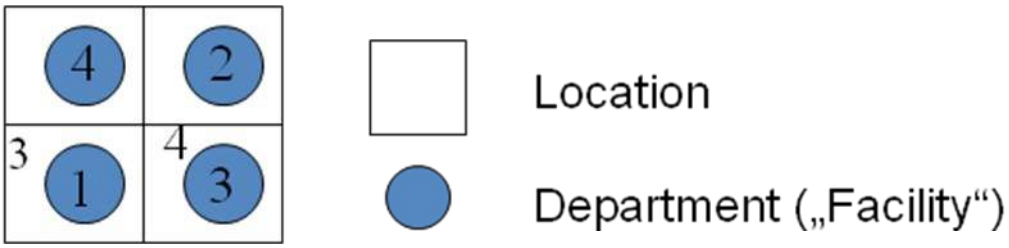
Notation distance between **locations** i and j

travel frequency between **departments** k and h

1 if department k is assigned to location i

0 otherwise

Example



Distance* $d_{i,j}$

	1	2	3	4
1	.	1	1	2
2	1	.	2	1
3	1	2	.	1
4	2	1	1	.

Frequency* $f_{k,h}$

	1	2	3	4
1	.	1	3	2
2	2	.	0	1
3	1	4	.	0
4	3	1	1	.

Ant System (AS-QAP)

Constructive method:

step 1: choose a facility j

step 2: assign it to a location i

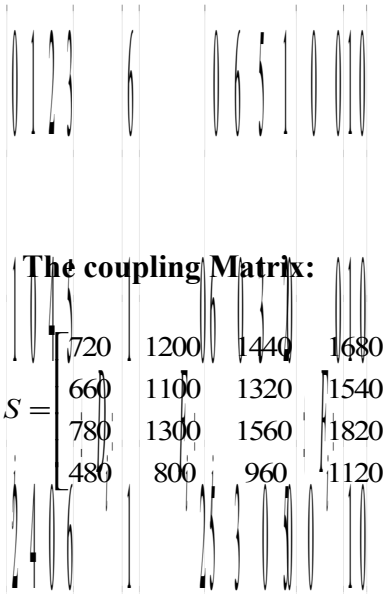
Characteristics:

– each ant leaves trace (pheromone) on the chosen couplings (i,j)

– assignment depends on the probability (function of pheromone trail and a heuristic information)

– already coupled locations and facilities are inhibited (Tabu list)

Heuristic information



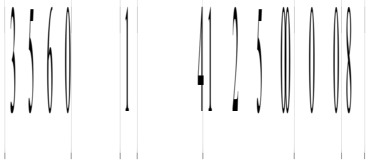
The coupling Matrix:

$$S = \begin{bmatrix} 720 & 1200 & 1440 & 1680 \\ 660 & 1100 & 1320 & 1540 \\ 780 & 1300 & 1560 & 1820 \\ 480 & 800 & 960 & 1120 \end{bmatrix}$$

$s_{11} = f_1 \bullet d_1 = 720$

$s_{34} = f_3 \bullet d_4 = 960$

Ants choose the location according to the heuristic desirability “Potential goodness”



$$\zeta_{ij} = \frac{1}{s_{ij}}$$

AS-QAP Constructing the Solution:

AS-QAP Constructing the Solution facilities are ranked in decreasing order of the flow potentials, Ant k assigns the facility i to location j with the probability given by

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & if \quad j\in N_i^k\end{cases}$$

where N_i^k is the feasible Neighborhood of node i

- When Ant k choose to assign facility j to location i it leave a substance, called trace "pheromone" on the coupling (i,j)
- Repeated until the entire assignment is found

AS-QAP Pheromone Update:

Pheromone trail update to all couplings:

$$\tau_{ij}(t+1)=\rho.\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

$\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k puts on the coupling (i,j)

$$\Delta_{ij}^{k}=\begin{cases}\dfrac{Q}{J_{\psi}^{k}} & if \text{ facility i is assigned to location j in the solution of ant k}\\ 0 & \text{otherwise}\end{cases}$$

J_{ψ}^{k} ...the objective function value

Q...the amount of pheromone deposited by ant k

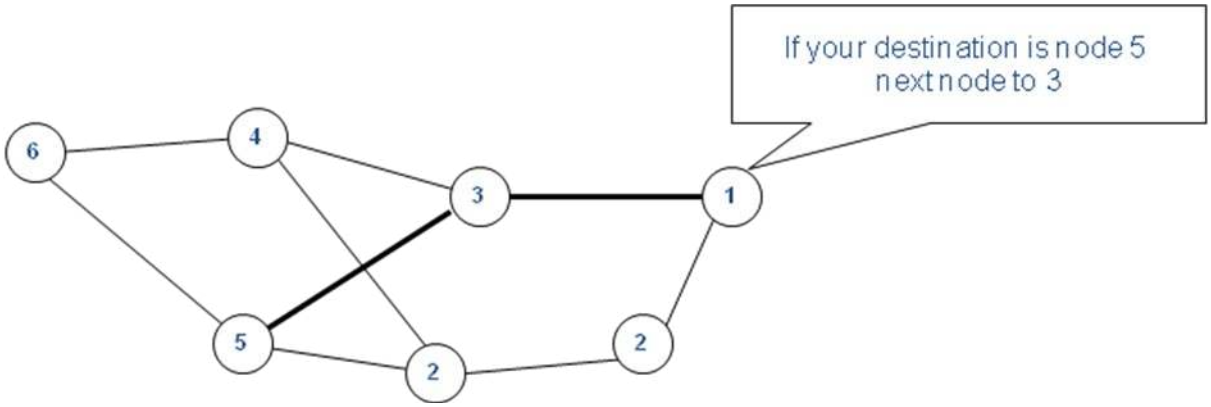
3.3 Network Model:

Routing (or routeing) is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network, electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes; typically hardware devices called routers, bridges, gateways, firewalls, or switches. General-purpose computers with multiple network cards can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the routers' memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

Routing task is performed by Routers.

Routers use “Routing Tables” to direct the data.



Problem statement:

- Dynamic Routing

At any moment the pathway of a message must be as small as possible. (Traffic conditions and the structure of the network are constantly changing)

- Load balancing

Distribute the changing load over the system and minimize lost calls

Algorithm:

Increase the probability of the visited link by:

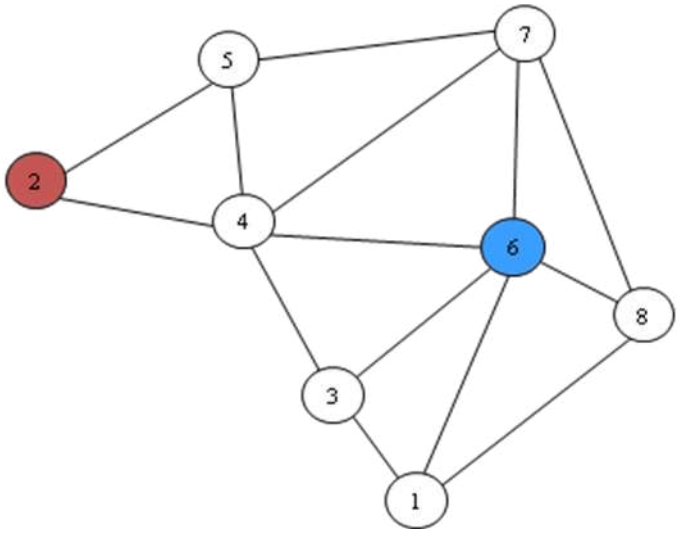
$$\rho = \frac{\rho_{old} + \Delta\rho}{1 + \Delta\rho}$$

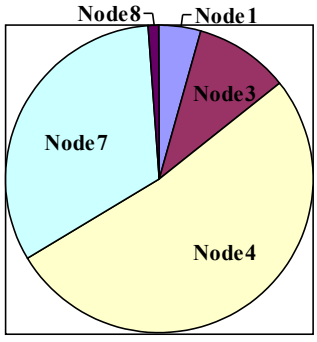
Decrease the probability of the others by :

$$\rho = \frac{\rho_{old}}{1 + \Delta\rho}$$

Where $\Delta\rho = f\left(\frac{1}{age}\right)$

Example:



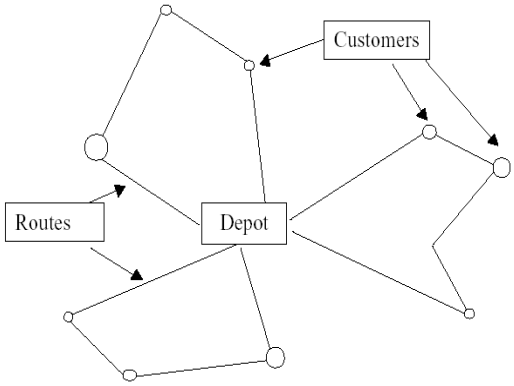


3.4 Vehicle Routing Problem with Time Windows (VRPTW):

The vehicle routing problem (VRP) is a combinatorial optimization and integer programming problem seeking to service a number of customers with a fleet of vehicles. Proposed by Dantzig and Ramser in 1959, VRP is an important problem in the fields of transportation, distribution and logistics.[1] Often the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. Implicit is the goal of minimizing the cost of distributing the goods. Many methods have been developed for searching for good solutions to the problem, but for all but the smallest problems, finding global minimum for the cost function is computationally complex.

Objective Functions to Minimize

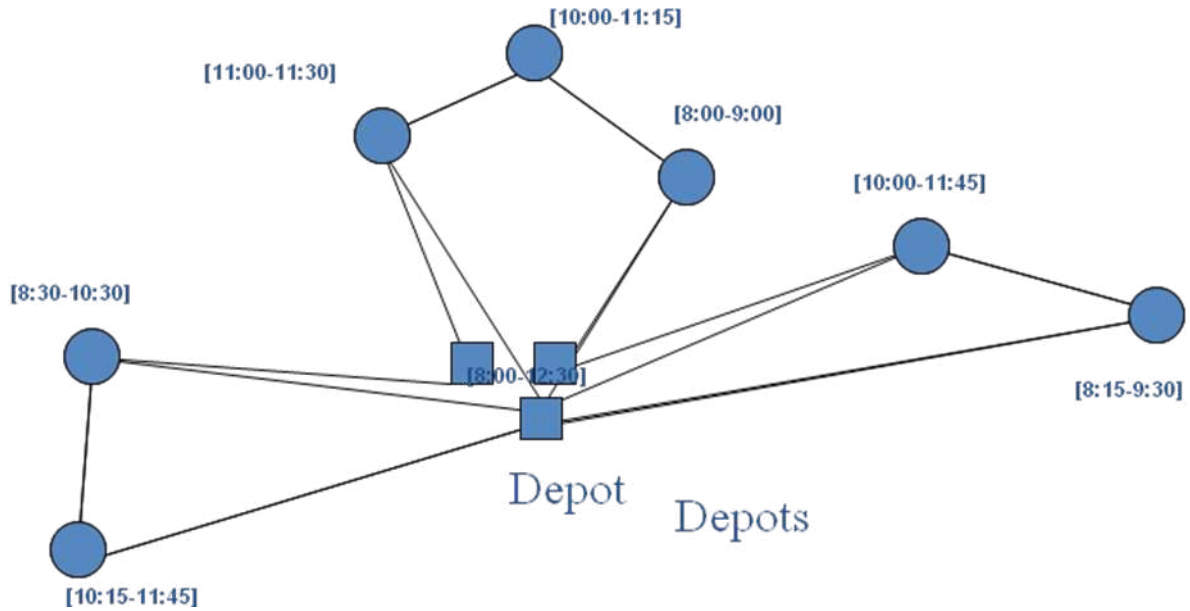
- Total travel distance
- Total travel time
- Number of vehicles



Subject to:

- Vehicles (# ,Capacity, time on road, trip length)
- Depots (Numbers)
- Customers (Demands, time windows)
-

Vehicle Routing Problem with Time Windows (VRPTW):



Simple Algorithm:

- Place ants on depots (Depots # = Vehicle #).
- Probabilistic choice

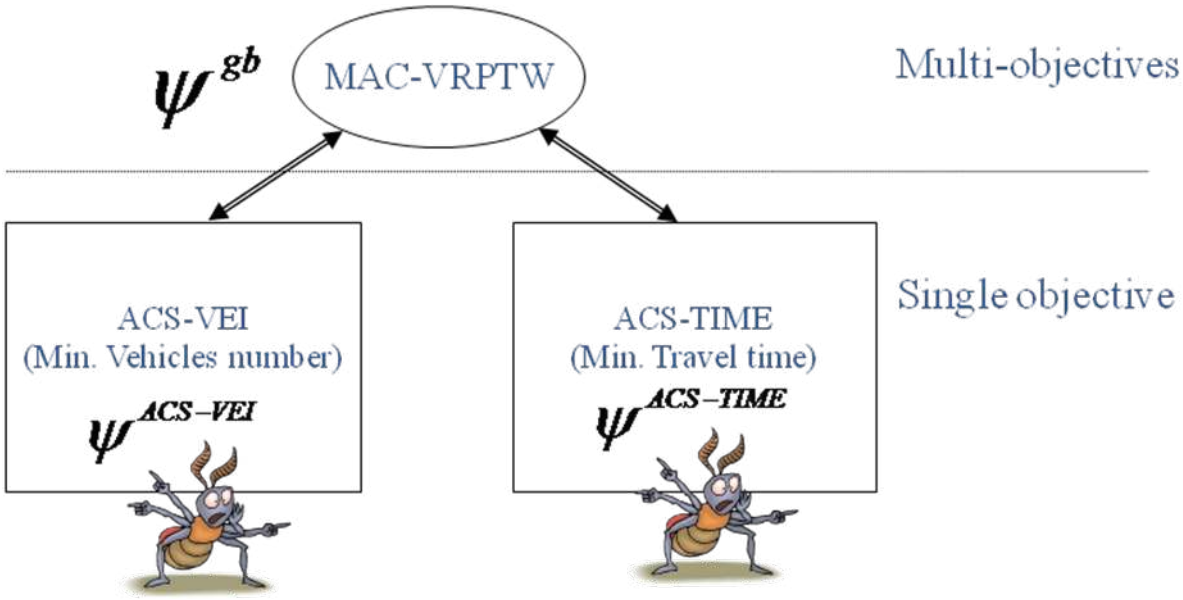
    ~ (1/distance, d_i, Q)

    ~ amount of pheromone

- If all unvisited customer lead to a unfeasible solution:

    Select depot as your next customer.
- Improve by local search.
- Only best ants update pheromone trial.

Multiple ACS For VRPTW:



ADVANTAGES OF ACO:

1. Inherent parallelism.
   - Ants works in parallel to find a solution.

2. Parallelism at the level of data.

   – Ants working for sub-problems
3. Efficient for several problems like TSP, QAP.
4. Positive feedback which accounts for rapid discovery of good solution
5. Can be used in dynamic application(Adapts to change such as new distances etc)

DISADVANTAGES OF ACO:

1. Theoretical analysis is difficult.
2. Probabilistic distribution changes by iteration.
3. Research is experimental rather than theoretical.
4. Time to converge is uncertain.

-

Conclusions:

• ACO is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems.

• Artificial ants implement a randomized construction heuristic which makes probabilistic decisions.

• The a cumulated search experience is taken into account by the adaptation of the pheromone trail.

• ACO Shows great performance with the “illstructured” problems like network routing.

• In ACO Local search is extremely important to obtain good results.

BIBLIOGRAPHY:

• Dorigo M. and G. Di Caro (1999). **The Ant Colony Optimization Meta-Heuristic.** In D. Corne, M. Dorigo and F. Glover,
editors, *New Ideas in Optimization,* McGraw-Hill, 11-32.

• M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Bio Systems*, 43:73–81, 1997.

• M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the travelling salesman problem.
IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

• G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In H. El-Rewini, editor, *Proceedings of the 31st International*
Conference on System Sciences (HICSS-31), pages 74–83. IEEE Computer Society Press, Los Alamitos, CA, 1998.

• M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: An autocatalytic optimizing process. Technical Report 91-016
Revised, Dipartimento di Elettronica,Politecnico di Milano, Italy, 1991.

• L. M. Gambardella, ` E. D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing
problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76.
McGraw Hill, London, UK, 1999.

• L. M. Gambardella, ` E. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the*
Operational Research Society,50(2):167–176, 1999.

• V. Maniezzo and A. Colorni. The Ant System applied to the quadratic assignment problem. *IEEE Transactions on Data and*
Knowledge Engineering, 11(5):769–778, 1999.

• Gambardella L. M., E. Taillard and M. Dorigo (1999). **Ant Colonies for the Quadratic Assignment Problem.** *Journal of*
the Operational Research Society, 50:167-176.