Received: February 2, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Virtual Machine Placement Using Hypercube Ant Colony Optimization Framework

Boominathan Perumal[1]* **Aramudhan Murugaiyan[2]**

[1]*School of Computing Science and Engineering, VIT University, Vellore, India*
[2] *Perunthalaivar Kamarajar Institute of Engineering and Technology, Puducherry, India*
* Corresponding author's Email: boominathan.p@vit.ac.in

Abstract: This paper presents a novel application of hypercube framework of ant colony optimization to the virtual machine (VM) placement problem, with the objective of minimizing the power consumption and resource wastage. The aim of the VM placement is to develop an optimal placement strategy to allocate the VM's to physical servers such that the usage of physical resource is minimized. The hypercube framework of ACO differs from its usual ACO implementation in the hyperspace for the pheromone values. In other words, in the hypercube framework, the updating of pheromone values is constrained to lie between 0 to1. This constrained pheromone updating has an advantage of automatically handling the scaling of objective function values and further leads to robust version pf ACO procedure. Here, we experimentally investigate the influence of hypercube framework of ACO for virtual machine placement using three ACO variants, namely, Ant system, Max-Min Ant System, and Ant colony system.

Keywords: Virtual Machine, Ant colony Optimization, Hyper cube, Power Consumption, Resource Wastage.

1. Introduction

Ant colony optimization (ACO) [1] is one of the metaheuristic approaches widely used for optimization problems. The collaborative behaviour of original ants that cooperate into a colony for finding best routes from the nest to the food location is the real inspiring source of ACO techniques. On finding the food source, the ants communicate indirectly with each other by depositing a substance called pheromone in the path it travelled. The subsequent ants tend to choose the path having high pheromone deposit. As time progresses, if the pheromone signal is not strengthened by other ants, then the path information is gradually lost and the pheromone signal evaporates. On the basis of these concepts, the movement of the artificial ants is driven by probabilistic decisions taken on the basis of the pheromone values.

The application of ACO is well explored in literature by mapping the functionalities of real ant colonies to artificial ant colonies to solve the several combinatorial optimization problems in various engineering domains. Lopez-Ibanez and Blumb [2] proposed an algorithm named Beam-ACO for the traveling salesman problem (TSP) with time windows. The minimum spanning tree problem was solved by Neumann and Witt [3] using a novel ACO approaches. In paper [4] ACO was proposed for production scheduling problem, a class of Asymmetric TSP. A hybrid algorithm was proposed for resource-constrained project scheduling [5].

Bi-Objective ACO approach was proposed to optimize production and maintenance scheduling [6]. Chen et al. [7] applied ACO to monitor the cash flow monitoring and to control the project cost. A model based on ACO approach was proposed by Lee et al. to search for the optimum gardener manpower requirements and a near-optimal maintenance schedule for the green areas [8]. ACO model was proposed for heterogeneous spatial information processing [9]. Martinez et al. developed an optimized ACO model for Reinforced Concrete (RC) bridge piers with hollow rectangular sections [10]. Rama Rao et al. proposed ACO based novel optimization method for determining kinetic and film thickness model parameters [11]. An Ant colony optimization approach was used for economic dispatch problem with non-smooth cost functions [12]. Xu et al. applied ACO model for MTT in Image processing [13]. Ketabi et al. proposed to use ant colony algorithm to reactive power pricing in an open electricity market [14].

Kwang and Weng [15] conducted a comparative study on all routing algorithms with ACO. Amilkar, et al. analysed the performance of ACO on various case studies in TSP using a two stage approach [16]. In the process of introducing virtual machine (VM) placement in cloud computing as a NP- hard optimization problem, we propose to use the well-developed metaheuristic technique named ant colony optimization technique for mapping a set of virtual machines to a set of physical machines. The problem of optimal mapping of VM to PM is considered as a multi objective optimization problem where we aim to simultaneously minimize power consumption and resource wastage. As it is well known that, there are several variants of ACO techniques available in literature and each has its own characteristics, we propose to apply the Hypercube Ant colony framework [17] which, in contrast to traditional ACO algorithms, brings several benefits like increase in the robustness of the ACO algorithm when implemented in the HCO framework, as it automatically scales the objective function values. That is, the behaviour of the algorithm is independent of whether the objective function values of a problem instance lie in the interval of [0,1] or [0,1000]. The other benefit would be in introducing changes in the pheromone update rule such that the pheromone values are constrained to lie between 0 and 1 for any variant of the ACO algorithm implemented in HCO framework.

In this paper, the hypercube framework is implemented on three major variants of ACO namely Ant System (AS), Max-Min ant system (MMAS) and Ant Colony System (ACS). The results show that the implementation of ACO algorithms in the HCF increases their robustness and makes it easier the handling of the pheromones during its implementation.

The rest of the paper is organized as follows. In Section 2 we formulate the virtual machine placement problem. Section 3 provides the variants of ACO and their implementation strategy for virtual machine placement. Section 4 gives the detailed description of virtual machine placement using hypercube ant colony system optimization framework. In section 5 we present the computational experimental study conducted and the results obtained. Section 6 concludes the paper followed by references.

2. Problem Formulation

The problem of VM placement is formulated as follows. Given, a set of n virtual machines, m physical servers and the physical servers already hosting n_i VMs, the VM placement algorithms generate possible mapping solutions for $n+n_i$ VMs on the m physical servers under a specified set of constraints. The objective function given in Eq. (1) is to minimize the power consumption and the objective function in Eq. (2) focuses on reducing the resource wastage.

Here, we consider the VMP placement problem as a multi-objective combinatorial optimization problem where the optimization strategy focuses on simultaneously minimizing the power consumption and the resource wastage under the given set of constraints. The constraint given in Eq. (3) assigns a VM i to only one of the servers. Constraints (4) and (5) specify the constraint related to the server capacity. Constraint (6) defines the range of the variables related to the VM placement. The notations used in the problem formulation are given in Table 1.

Table 1. Notations and Description

Notation	Description
$[\, R^i_{cpu} \;\; R^i_{mem} \,]$	CPU and Memory resource requirements of the server i
$[TC^j_{cpu}\, TC^j_{mem}]$	Total capacity vector of server j with respect to CPU and memory
$alloc_{ij} \in \{0,1\}$	The binary variable $alloc_{ij}$ is set to 1 if virtual machine i is allocated to the server j
$y_j \in \{0,1\}$	The binary variable y_j is set to 1 if server j is packed with any virtual machine i.
RW_j	Resource wastage of the jth server
P_j	Power consumed by the jth server
U^j_{cpu}	CPU utilization of the jth server
U^j_{mem}	Memory utilization of jth server
L^j_{cpu}	Normalized remaining CPU of the server j
L^j_{mem}	Normalized remaining memory of server j
P_j^{busy} and P_j^{idle}	Average power values when server j is busy and server j is idle.

$$Minimize \sum_{j=1}^{m} P_j = \sum_{j=1}^{m}\left[y_j \times \left(\left(P_j^{busy} - P_j^{idle}\right) \times \sum_{i=1}^{n}\left(alloc_{ij}\, R_{cpu}^{i}\right) + P_j^{idle}\right)\right] \tag{1}$$

$$Minimize \sum_{j=1}^{m} RW_j = \sum_{j=1}^{m}\left[y_j \times \frac{\left|\left(TC_{cpu}^{j} - \sum_{i=1}^{n}\left(alloc_{ij}.\, R_{cpu}^{i}\right)\right) - \left(TC_{mem}^{j} - \sum_{i=1}^{n}\left(alloc_{ij}\, R_{mem}^{i}\right)\right)\right| + \varepsilon}{\sum_{i=1}^{n}\left(alloc_{ij}\, R_{cpu}^{i}\right) + \sum_{i=1}^{n}\left(alloc_{ij} R_{mem}^{i}\right)}\right] \tag{2}$$

Subject to:

$$\sum_{j=1}^{m} alloc_{ij} = 1 \;; \forall\, i \in I \tag{3}$$

$$\sum_{i=1}^{n} R_{cpu}^{i}\, .alloc_{ij} \leq TC_{cpu}^{j} y_j \,;\; \forall\, j \in \mathrm{J} \tag{4}$$

$$\sum_{i=1}^{n} R_{mem}^{i}\; alloc_{ij} \leq TC_{mem}^{j}\, y_j ; \forall_j \in \mathrm{J} \tag{5}$$

$$y_j ,\, alloc_{ij} \in \{0,1\}; \forall\, i \in \mathrm{I}\, , \forall\, j \in \mathrm{J} \tag{6}$$

The potential cost of wasted resources is modelled using the following Eq. (7)

$$RW_j = \left(\left|L_{cpu}^{j} - L_{mem}^{j}\right| + \varepsilon\right) / \left(U_{cpu}^{j} + U_{mem}^{j}\right) \tag{7}$$

ε is a very small positive real number and its value is set to be 0.0001. The power consumption of the j-th server is defined as a function of the CPU utilization as shown below.

$$P_j = \begin{cases} \left(P_j^{busy} - P_j^{idle}\right) \times U_{cpu}^{j} + P_j^{idle}; & U_C^{j} > 0 \\ 0 \;; & \text{otherwise} \end{cases} \tag{8}$$

Where, P_j^{idle} and P_j^{busy} are the average power values when the j^{th} server is idle and when fully utilized, respectively. The values of P_j^{idle} and P_j^{busy} have been fixed to 162 and 215 Watt [18].

3. Variants of ACO for Virtual Machine Placement

The three major variants of ACO namely Ant system, Max-Min Ant System and Ant colony System are used to explore the implementation of hypercube framework.

3.1 Ant Systems (AS)

The two main phases of AS [19] algorithm are the ant's solution construction phase and the pheromone update phase. At each solution construction step, an ant k in AS applies a probabilistic action choice rule, named random proportional state transition rule, to decide the next VM i to place in server j. In particular, the probability with which an ant k, chooses server j for VM i is

$$P_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{u \in \Omega_k(j)} [\tau_{iu}]^{\alpha} \cdot [\eta_{iu}]^{\beta}} & ; \; i \in \Omega_k(j) \\ 0 & ; \;\text{otherwise} \end{cases} \tag{9}$$

Where, τ_{ij} is the pheromone trail deposited when virtual machine i is placed in server j. It defines the favourability of packing VM i into server j. The heuristic information η_{ij} indicates the desirability of assigning *VM i* to server j. α and β parameters weigh the relative importance of the pheromone trail and the heuristic information.

$$\Omega_k(j) = \left\{ \begin{array}{l} i \in \{1,\ldots,n\} \left| \left(\sum_{u=1}^{m} alloc_{iu} = 0 \right) \right. \wedge \\ \left(\left(\sum_{u=1}^{n} \left(alloc_{uj} \times R_{cpu}^{u}\right) + R_{cpu}^{i} \right) \leq TC_{cpu}^{j} \right) \wedge \\ \left(\left(\sum_{u=1}^{n} \left(alloc_{uj} \times R_{mem}^{u}\right) + R_{mem}^{i} \right) \leq TC_{mem}^{j} \right) \end{array} \right\} \tag{10}$$

Ω_k (j) is the set of eligible virtual machines determined for ant k to pack in a particular server j.

In Eq. (10), each ant k stores the virtual machines placed so far, and this memory is exploited to determine Ω_k (j) in each construction step.

The pheromone updating is done once all ants have finished constructing their solutions and is also evaporated in all the solution components.
The update rule is as given below:

$$\tau_{ij}(t) = \left(1-\rho_g\right)\tau_{ij}\ (t-1) + \sum_{k=1}^{NA}\Delta\tau_{ij}^{k} \quad (11)$$

$$\Delta\tau_{ij}^{k} = {}^{1}\!/_{VMP_k} \quad (12)$$

Where, $\rho_g \in \{0,1\}$ is the evaporation rate, NA is the number of ants, t is the iteration number and VMP_k is the quality of the solution determined using the normalized power consumption and resource wastage of the solution S given by ant k.

$$VMP_k = P'(S_k) + RW(S_k) \quad (13)$$

For the initial iteration, the pheromone value τ_0 is determined using the term NA/VMP. NA is the number of ants and VMP is the quality of the initial solution S_0 determined as $P'(S_0)+RW(S_0)$. $P'(S_0)$ is the normalized CPU power consumption of the solution S_0 and is calculated using Eq. (14).

$$P'(S_0) = \sum_{j=1}^{m}\left({}^{P_j}\!/_{P_j^{max}}\right) \quad (14)$$

RW (S_0) is the resource wastage of the solution S_0. The initial solution S_0 is generated by initial random placements. P_j^{max} is the maximum power consumption of the server j.

3.2 Max-Min Ant System (MMAS)

The Max–Min Ant System (MMAS) algorithm introduces few modifications to the AS [20]. The pheromone update function is bound to lie in the interval [$\tau_{\min}$, $\tau_{\max}$]. The initial pheromone value τ_0 is computed as

$$\tau_0 = {}^{1}\!/_{\rho_g} \cdot VMP \quad (15)$$

The pheromone update rule for MMAS is as follows:

$$\tau_{ij} = \left[\left(1-\rho_g\right) \cdot \tau_{ij}(t-1) + \Delta_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}} \quad (16)$$

Where $\tau_{\max}$ and $\tau_{\min}$ are the upper and lower pheromone bounds, respectively.

$$\Delta_{ij}^{best} = {}^{1}\!/_{VMP^{best}} \quad (17)$$

$$\tau_{max} = {}^{1}\!/_{\rho_g \cdot VMP^{best}} \quad (18)$$

$$\tau_{min} = \tau_{max}\left(1-\sqrt[n]{pbest}\right)/\left((avg-1)\sqrt[n]{pbest}\right) \quad (19)$$

$$avg = n/2 \quad (20)$$

VMP^{best} is the best solutions quality determined using normalized power consumption and resource wastage. Chosen a value for $pbest$, the value of $\tau_{\min}$ can be estimated. The different cases would be if $pbest$ is chosen to be 1, then $\tau_{\min}$ is assigned to zero. If $pbest$ is chosen to be too small then it may lead to a situation where $\tau_{\min} > \tau_{\max}$.

In such a case, we set $\tau_{\min} = \tau_{\max}$, where only heuristic information will be used in the process of solution construction. For our experiments we have chosen the $pbest$ value as 0.05.

If we let the update function be x, the upper and lower bounds are imposed in the following way:

$$[x]_{\tau_{min}}^{\tau_{max}} = \begin{cases} a\,; & \text{if } x > \tau_{max} \\ b\,; & \text{if } x < \tau_{min} \\ x\,; & \text{otherwise} \end{cases} \quad (21)$$

When solution gets stagnated, occasionally pheromone trails are reinitialized using the variable $pbest$ and convergence factor (cf).

$$cf = 2\left(\left(\frac{\sum_{\tau_{ij} \in T^{max}}\{\tau_{max}-\tau_{ij},\ \tau_{ij}-\tau_{min}\}}{|T|.(\tau_{max}-\tau_{min})}\right)-\tau_{init}\right) \quad (22)$$

For the purpose of re-initialization and updating of pheromone values, it is required to keep track of the solutions, which are best in the current iteration, the best solution-so-far and the restart-best solution. Eq. (22) is used as a part of re-initialization process. The strategy behind the pheromone update in MMAS is as follows: At the start of the restart phase (i.e., when $pbest$ is TRUE and convergence-factor is nearer to zero), the updating of pheromone values is done using iteration best solution. Then, as the number of iterations increases, very often, the best solution which is found in the current restart phase is considered for pheromone update. Just before convergence occurring (i.e., convergence-factor < 0.9999), the restartbest solution alone is used for pheromone update, and once when convergence is found (i.e., convergence-factor > 0.9999), the control variable *pbest* is set to TRUE and the best-so-far solution is used for updating the pheromone values. If there is no convergence, then in the next iteration, the pheromone values are reset to initial values and the algorithm is restarted [20].

3.3 Ant Colony System (ACS)

In Ant Colony System (ACS) [21], the pseudo random proportional rule is given by

$$i = \begin{cases} argmax_{u\in\Omega_k(j)} \left\{ \begin{matrix} \alpha \times \tau_{uj} \\ + \\ (1-\alpha) \times \eta_{uj} \end{matrix} \right\}; q \leq q_0 \\ s \qquad ; \text{otherwise} \end{cases} \tag{23}$$

Where, q is a random value uniformly distributed in [0, 1]. If q is greater than q_0, the process is called exploration, otherwise it is exploitation. q_0 is a fixed parameter between 0 to 1. The pheromone value τ_{ij} is as given in Eq. (25-26). is a random variable selected according to the following random-proportional rule probability distribution [22], which is the probability that ant k chooses to assign VM i to server j.

$$P_{ij}^{k} = \begin{cases} \dfrac{\alpha \times \tau_{ij} + (1-\alpha) \times \eta_{ij}}{\sum_{u\,\epsilon\,\Omega_k(j)} (\alpha \times \tau_{ij} + (1-\alpha) \times \eta_{ij})}; i\ \epsilon\ \Omega_k(j) \end{cases} \tag{24}$$

The local pheromone update rule is given below in Eq. (25).

$$\tau_{ij} = (1-\rho_l).\ \tau_{ij}(t-1) + \rho_l.\tau_0 \tag{25}$$

Where, $\rho_l\ \epsilon\{0,1\}$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone. The global updating rule is applied after all ants have finished building a solution. The solutions of Pareto set are updated using following rules.

$$\tau_{ij}(t) = \left(1-\rho_g\right) \tau_{ij}(t-1) + \rho_g \Delta\tau_{ij}^{best} \tag{26}$$

$$\Delta\tau_{ij}^{best} = \lambda / VMP^{best} \tag{27}$$

$$\lambda = NA / t - NI_S + 1 \tag{28}$$

For multi objective optimization, in general the Pareto set which contains the global non-dominated solutions are stored in an external set. If a current iteration solution is not dominated by any other solutions in the current iteration or in the external set, then the current iteration solution is added to the external set and the quantity of pheromone in all movements which constructed it is increased.

Further, all the solutions that are dominated by the added one are removed from the external set. In Eq. (28), *NA* is the number of ants and NI_s represents the number of epochs that the solution S has existed in the external set. λ as an adaptive coefficient controls how a solution in the external set contributes to pheromone information over time. The non-dominated solution updates are also considered for AS and MMAS as well.

3.4 Hyper cube framework for Ant System (HC-AS)

In the AS framework described in section 3.1, the ants construct the solution using the pheromone values associated to the solution components. It is known that the pheromone values considered as a vector $\overline{\tau} = (\tau_1, \tau_{2,\ldots,} \tau_n)$ moves in a hyperspace with different limits for different pheromone updating rules. The hyperspace is denoted as T. For AS, the pheromone values τ_i are limited by

$$\lim_{t\to\infty} T_i(t) \leq \left(1/_{1-\rho}\right) . \left(k/_{f(s^{opt})}\right) \tag{29}$$

It is clear that the limits of T can be very different depending on the pheromone updating rule itself and the amount of pheromone added in each step, which is a function of the solution quality. Here, we rewrite the pheromone updating rules for hypercube framework of Ant System [23].

Ant System's new updating rule is obtained via a normalization of Eq. (11-12)

$$\tau_{ij}(t) = \rho_g.\ \tau_{ij}(t-1) + \left(1-\rho_g\right) \sum_{k=1}^{NA} \Delta\tau_{ij}^{k} \tag{30}$$

$$\Delta\tau_{ij}^{k}= \left(1/_{VMP_k}\right)/\left(\sum_{k=1}^{NA} 1/_{VMP_k}\right) \quad (31)$$

The new update rule restricts the pheromone values to lie between 0 and 1.

3.5 Hyper cube framework for Max-Min Ant System (HC-MMAS)

In hypercube framework of MMAS, instead of using only one solution per iteration for updating the pheromone values, a weighted average of the three solutions is used. The modified update rules are as given in Eq. (32) and Eq. (33).

$$\tau_{ij}(t)= \rho_g \tau_{ij}(t\text{-}1)+ \left(1\text{-}\rho_g\right).\Delta\tau_{ij}^{\text{bestsol}} \quad (32)$$

$$\Delta_{ij}^{bestsol}= w_{ibest}.\Delta_{ij}^{ibest}+ w_{rbest}.\Delta_{ij}^{rbest}+ w_{bbest}.\Delta_{ij}^{bbest} \quad (33)$$

In Eq. (33) w_{ibest} w_{rbest} and w_{bbest} are the weights of the iteration best (*ibest*), restart best (*rbest*) and best so far (*bbest*) solutions. The update rules given in Eq. (32) sets the pheromone values that exceed τ_{max}, back to $\tau_{max}..\Delta_{ij}^{bestsol}$ in Eq. (33) allows choosing how to schedule the relative influence of the three solutions used for updating pheromones.

The Setting of weights based on the convergence factor and Boolean variable for updating Eq. (33) is given by Blum and Dorigo [23].

3.6 Hyper cube framework for Ant Colony System (HC-ACS)

The modified update rules and the process of VM placement for HC-ACS is given in next section.

4. Application of Hyper cube Ant colony system framework for Virtual Machine Placement

In this section, we apply the hypercube framework of Ant colony system approach for virtual machine placement. The given procedure works as follows: in an initialization phase, the parameters are initialized and all the pheromone trails associated with every VM to PM mapping are set to τ_0 *as* given in Eq. (34).

Then we generate initial solution by mapping VM-PM in a random manner. In the iterative optimization phase, each ant receives all VM requests, introduces a physical server and starts assigning VMs to servers. Here, we use pseudo random proportional rule which describes the desirability for an ant to choose a particular VM for placing in the current server based on pheromone and heuristic information. A local pheromone update is performed once an artificial ant has built a solution. After all ants have constructed their solutions, a hypercube framework based global update rule is used to update the pheromone values of the current Pareto set. The initial pheromone value is calculated using Eq. (34)

$$\tau_0=1/\left[n.(P^{'}(S_0)+RW(S_0))\right] \quad (34)$$

The heuristic information denoted as η_{ij} indicates the desirability of assigning *VM i* to server *j*. The partial contributions of assigning VM *i* to the server *j* are given in Eq. (35-36).

$$\eta_{ij1}= 1/\left(\varepsilon+ \sum_{v=1}^{j} (P_v/P_v^{max})\right) \quad (35)$$

$$\eta_{ij2}=1/\left(\varepsilon+ \sum_{v=1}^{j} W_v\right) \quad (36)$$

The total desirability of each movement using Eq. (37)

$$\eta_{ij}= \eta_{ij1}+\eta_{ij2} \quad (37)$$

In the process of choosing the next VM to place in current server, the ant *k* selects the VM according to the pseudo-random-proportional rule given in Eq. (23-24). The local updating rule of ACS is given in Eq. (25) and the global update of the pheromone values in HCF framework is computed using the following modified global update rule:

$$\tau_{ij}(t)=\left(1\text{-}\rho_l\right)\tau_{ij}(t\text{-}1)+\rho_g \Delta\tau_{ij}^{best} \quad (38)$$

$$\Delta\tau_{ij}^{best}= \left(\frac{\lambda}{VMP^{best}}\right)/\sum_{pareto=1}^{paretoset\_n} \frac{\lambda}{VMP^{pareto}} \quad (39)$$

The best solutions are the non-dominated solutions (Pareto set) and each of the solution components are updated such that the values lie between 0 and 1.

5. Computational Experiments

In this paper, we have conducted experiments on six ant colony optimization algorithms for virtual machine placement problem.

Received: February 2, 2017

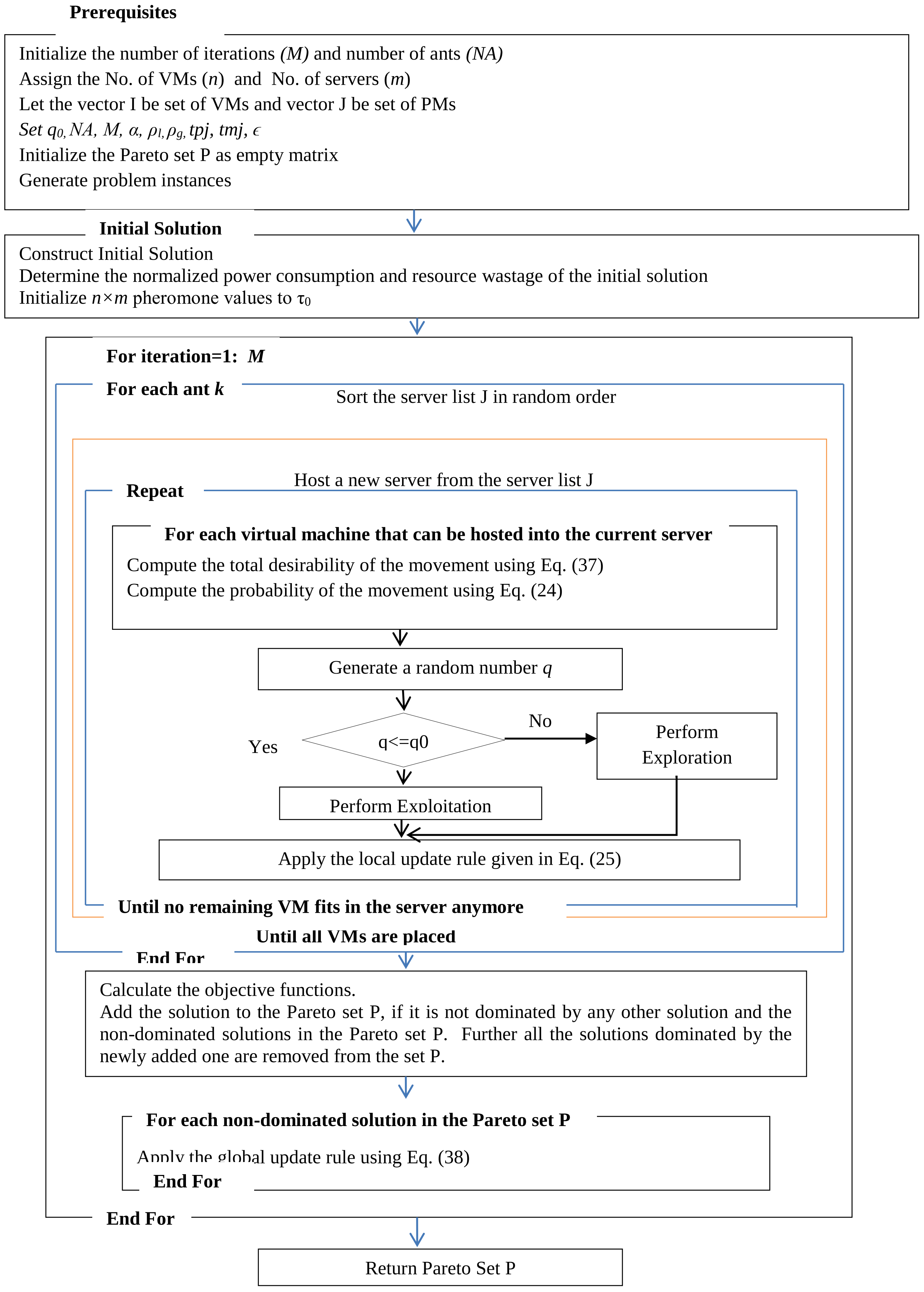
Prerequisites
Initialize the number of iterations *(M)* and number of ants *(NA)*
Assign the No. of VMs (n) and No. of servers (m)
Let the vector I be set of VMs and vector J be set of PMs
Set q_0, NA, M, α, ρ_l, ρ_g, tpj, tmj, ϵ
Initialize the Pareto set P as empty matrix
Generate problem instances
Initial Solution
Construct Initial Solution
Determine the normalized power consumption and resource wastage of the initial solution
Initialize $n\times m$ pheromone values to τ_0
For iteration=1: *M*
For each ant *k*
Sort the server list J in random order
Repeat
Host a new server from the server list J
For each virtual machine that can be hosted into the current server
Compute the total desirability of the movement using Eq. (37)
Compute the probability of the movement using Eq. (24)
Generate a random number *q*
q<=q0
Yes
No
Perform Exploration
Perform Exploitation
Apply the local update rule given in Eq. (25)
Until no remaining VM fits in the server anymore
Until all VMs are placed
End For
Calculate the objective functions.
Add the solution to the Pareto set P, if it is not dominated by any other solution and the non-dominated solutions in the Pareto set P. Further all the solutions dominated by the newly added one are removed from the set P.
For each non-dominated solution in the Pareto set P
Apply the global update rule using Eq. (38)
End For
End For
Return Pareto Set P

Figure.1 Flow-chart of the HC-ACO algorithm applied to virtual machine placement problem

Firstly, we have used the traditional ant system approach for placing virtual machines in optimal number of servers. The initial pheromone values for AS are initialized as *n/ P′*(*S*$_0$)+*RW*(*S*$_0$). The algorithm performs global update of pheromone values once all ants construct the solution.

Secondly, we have used Max-Min ant system where the initial pheromone values are initialized as $1/\rho.P'(S_0)+RW(S_0)$ and the updating of pheromone values during solution construction process are restricted between τ_{min} and τ_{max}..

Thirdly, we use ant colony system where it has additional characteristics of local update done for each solution constructed by the ants and the global update as in other variants. The initial pheromone values for ACS are as given in Eq. (34).

In the next three experiments related to hypercube framework, the Eq. (30-39) is used to update pheromone values. The programs were coded in the MATLAB and ran on an Intel Pentium® Dual-Core processor with 2.50 GHz CPU and 3 GB RAM. The random sequences of problem instances are generated using the procedure given by Gao with reference values set as 35% [18]. The procedure used for generating solutions is given in Fig. 1.

By conducting experiments, we have determined the power consumption and resource wastage of the placement solutions generated by ACO and its variants. The results recorded in Table 4 – Table 8 are average of 20 runs of 100 iterations.

From the results obtained, we observe that as the numbers of virtual machines are increased gradually, among the classical ACO techniques Ant colony System (ACS) variant which performs both exploration and exploitation performed better in obtaining placement solutions which had minimum power consumption and resource wastage. Further, compared to classical ACO techniques like AS, MMAS and ACO, the hyper cube ACO variants generated still better solutions with much lesser power consumption and resource wastage than classical variants.

Table 4. Power consumption and resource wastage of VMP using Variants of ACO for 50 VMs

Algorithm	**Power (W)**	**Resource Wastage**	**CPU time (Sec)**
AS	8621	2.00	0.99
MMAS	8821	1.60	0.97
ACS	8732	1.37	0.88
HC-AS	8950	2.04	1.23
HC-MMAS	8860	1.91	1.81
HC-ACS	8775	1.82	1.36

Table 5. Power consumption and resource wastage of VMP using Variants of ACO for 100 VMs

Algorithm	**Power (W)**	**Resource Wastage**	**CPU time (Sec)**
AS	15757	2.61	1.01
MMAS	15343	3.11	1.84
ACS	15518	4.21	1.66
HC-AS	15586	4.39	1.42
HC-MMAS	16096	2.12	2.24
HC-ACS	15757	2.61	1.01

Table 6. Power consumption and resource wastage of VMP using Variants of ACO for 150 VMs

Algorithm	**Power (W)**	**Resource Wastage**	**CPU time (Sec)**
AS	19628	3.96	2.05
MMAS	19621	4.14	3.45
ACS	18617	3.61	2.06
HC-AS	19078	4.00	2.57
HC-MMAS	19728	3.96	3.76
HC-ACS	18538	3.24	2.96

Table 7. Power consumption and resource wastage of VMP using Variants of ACO for 200 VMs

Algorithm	**Power (W)**	**Resource Wastage**	**CPU time (Sec)**
AS	25029	7.79	4.90
MMAS	25763	6.55	3.91
ACS	25159	4.92	3.46
HC-AS	25297	4.85	4.97
HC-MMAS	25143	4.85	5.54
HC-ACS	25053	3.58	4.53

Table 8. Power consumption and resource wastage of VMP using Variants of ACO for 250 VMs

Algorithm	**Power (W)**	**Resource Wastage**	**CPU time (Sec)**
AS	31219	8. 46	5.39
MMAS	31330	8.12	5.73
ACS	31298	8.31	5.49
HC-AS	31198	8.41	6.45
HC-MMAS	29576	6.64	6.08
HC-ACS	31192	7.61	6.22

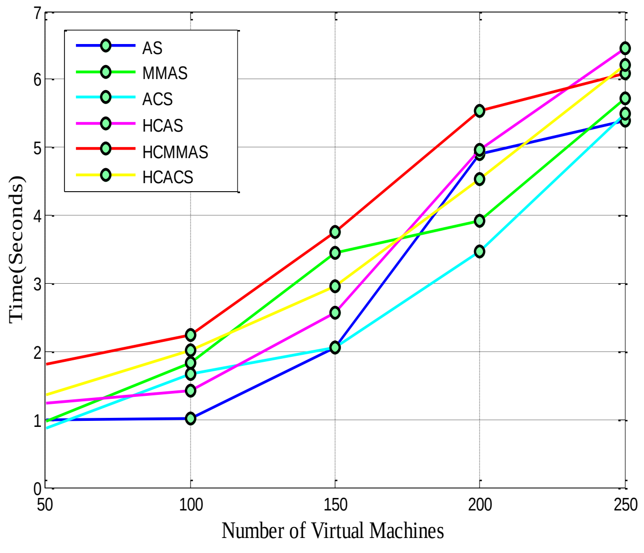

Figure. 2 Running time of ACO variants relative to number of virtual machines

From Fig. 2, we find that among hypercube algorithms, HC-ACS takes less time to get optimal results compared to other variants when numbers of VMs are increased gradually.

6. Conclusion

In this paper, we have applied hypercube ant colony optimization framework for optimal and efficient placement of static multi objective virtual machine placement problem. In using traditional ACO techniques, generally the upper limit of the pheromone values is unknown and it majorly depends on the values of an optimal solution obtained.

However, if the ACO algorithm is implemented in the Hyper cube framework (HCF), especially the Max- Min ant system gets benefited as the upper and lower limit for pheromone values are known to be respectively 1 and 0. The HCF framework also automatically handles the scaling of the objective function values. The HCF solution performance is compared to that of an existing ACO variant, without hypercube framework and within hypercube framework. The results demonstrate that HCF algorithm gives on average a more robust behaviour of ant colony optimization algorithms for virtual machine placement. Finally the scalability of the proposed algorithm is also considered and verified by conducting several experiments varying the number of virtual machines.

The possible future research directions related to virtual machine placement is to make an attempt on parallel implementation of ant colony optimization techniques for the execution of placement algorithms. The other direction would be to investigate on dynamic placement strategies with load prediction and live migration techniques as a part of server consolidation and placement problem.

References

[1] M. Dorigo, *Optimization, learning and natural algorithms, PhD thesis,* Politecnico di Milano, Italy, [in Italian], 1992.

[2] M. Lopez-Ibanez, and C. Blum, “Beam-ACO for the travelling salesman problem with time windows”, *Computers & Operations Research,* Vol. 37, pp.1570–1583, 2010.

[3] F. Neumann, and C. Witt, “Ant Colony Optimization and the minimum spanning tree problem”, *Theoretical Computer Science,* Vol. 411, pp. 2406–2413, 2010.

[4] A. Andziulis, D. Dzemydiene, and C. Steponavicius, “Comparison of two heuristic approaches for solving the production scheduling problem”, *Information Technology and Control,* Vol. 40, No.2, pp. 118–122, 2011.

[5] W. Chen, Shi, H.-F. Teng, X.-P. Lan, and L.-C. Hu, “An efficient hybrid algorithm for resource-constrained project scheduling”, *Information Sciences*, Vol. 180, pp. 1031–1039, 2010.

[6] A. Berrichi, F. Yalaoui, L. Amodeo, and M. Mezghiche, “ Computers Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling”, *Operations Research*, Vol. 37, pp. 1584–1596, 2010.

[7] W.-N.Chen, J. Zhang, H. S. - H.Chung, R.-Z. Huang, and O. Liu, “Optimizing Discounted Cash Flows in Project Scheduling—An Ant Colony Optimization Approach”, *IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews,* Vol .40, No. 1, pp. 64-77, 2010.

[8] H.-Y. Lee, H.-H .Tseng, M.-C. Zheng, and P.-Y. Li, “Decision support for the maintenance management of green areas”, *Expert Systems with Applications*, Vol. 37, pp. 4479–4487, 2010.

[9] S. Schockaert, P.D. Smart, and F.A. Twaroch, "Generating approximate region boundaries from heterogeneous spatial information: An evolutionary approach", *Information Sciences*, Vol.181, pp. 257–283, 2011.

[10] F.J. Martinez, F. Gonzalez-Vidosa, A. Hospitaler, and V. Yepes, "Heuristic optimization of RC bridge piers with rectangular hollow sections", *Computers and Structures,* Vol. 88, pp. 375–386, 2010.

[11] T. Rama Rao, C. Srinivasan, and C. Venkateswarlu, "Mathematical and kinetic modeling of biofilm reactor based on Ant Colony Optimization", *Process* Biochemistry, Vol. 45, pp. 961–972, 2010.

[12] S. Pothiya, I. Ngamroo, and W. Kongprawechnon, "Ant colony optimisation for economic dispatch problem with non-smooth cost functions", *Electrical Power and Energy Systems*, Vol. 32, pp. 478–487, 2010.

[13] B. Xu, Q. Chen, J. Zhu, and Z. Wang, "Ant estimator with application to target tracking", *Signal Processing*, Vol. 90, pp. 1496–1509, 2010.

[14] A. Ketabi, A. Alibabaee, and R. Feuillet, "Application of the ant colony search algorithm to reactive power pricing in an open electricity market", *Electrical Power and Energy Systems*, Vol. 32, pp. 622–628, 2010.

[15] M. S. Kwang, and H. S. Weng, "Ant Colony Optimization for routing and load balancing: Survey and new directions", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 33, No.5, pp. 560–572, 2003.

[16] P. Amilkar, B. Rafael, and H. Francisco, "Analysis of the efficacy of a two-stage methodology for Ant Colony Optimization: Case of study with TSP and QAP", *Expert Systems with Applications (Elsevier)*, Vol. 37, pp. 5443–5453, 2010.

[17] C. Blum, A. Roli , and M. Dorigo, " HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization", In: *Proc. 4th Metaheuristics International Conference*, pp. 399-403, 2001.

[18] Y. Gao , H. Guan , Z. Qi , Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", *Journal of Computer and System Sciences,* Vol. 79, pp. 1230–1242, 2013.

[19] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics—Part B,* Vol. 26, pp. 29-41, 1996.

[20] T. Stützle, and H.H. Hoos, "MAX–MIN ant system", *Future generation computer systems,* Vol. 16, No.8, pp. 889-914,2000.

[21] M. Dorigo, and L. M. Gambardella, " Ant Colony System: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53–66, 1997.

[22] V.Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem" , *INFORMS J. Comput*, Vol. 11, No.4, pp. 358–369, 1999..

[23] C. Blum, and M. Dorigo, "The Hyper-Cube Framework for Ant Colony Optimization", *IEEE Transactions on Systems, Man and Cybernetics, PART B: Cybernetics*, Vol. 34, No. 2, 2004.