Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
REVIEW ARTICLE
Available Online at www.ijarcs.info

Evolution and Growth of Metaheuristics for Solving Variety of Problems

Gurdeep Kaur
Department of Computer Engineering andTechnology
Guru Nanak Dev University
Amritsar, Punjab

Amit Chhabra
Department of Computer Engineering andTechnology
Guru Nanak Dev University
Amritsar, Punjab

Abstract:In this paper we have discussed the growth and evolution of various metaheuristics used for solving variety of problems in this world. Additionally this paper also discusses few metaheuristics that are used for web service composition in Cloud. Comparison between various metaheuristics is done on the basis of main idea, advantages and disadvantages. At last, conclusion is drawn on the basis of information collected from the comparison of metaheuristic algorithms.

Keywords:Metaheuristics, scheduling, web service composition, Cloud computing.

1. INTRODUCTION

The optimization problems of real world are very effortful to solve. Many metaheuristic algorithms(nature-inspired algorithms) or optimization tools are introduced to solve such type of problems in science and engineering[6].

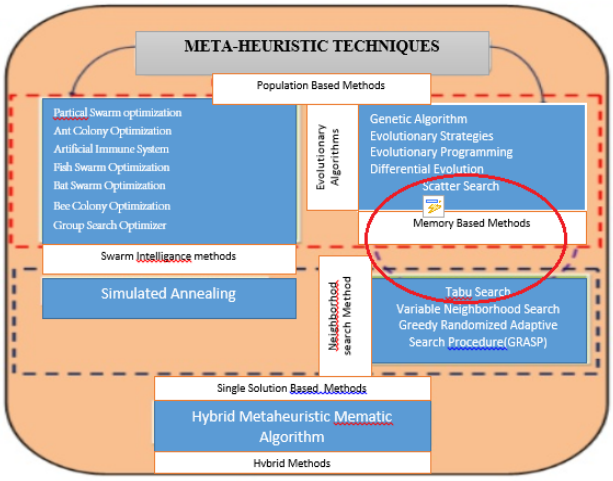

Figure 1: Meta-Heuristic Techniques[3]

Ideas to developing such algorithms come from nature (biological, chemical and physical processes). In the past decades algorithms implementation is based on some methods like dynamic programming, greedy method, branch and bound, backtracking etc. But those methods are not suitable for the large scale combinational problems as well as there are not any optimization technique for some problems like NP-Hard. Such problems are solved by trial and errormethod [1]. Cloud Computing involves On-demand self-service, Broad Network Access, Resource Pooling, Rapid elasticity, Measured service, Performance, Reduced Costs, Outsource management, Reliability and Multi-tenancy. It offers high level of data availability to external customer(on behalf of pay per use) all the time. Resources provided by cloud are: software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS)[9]. However, this paper not only focus on cloud computing environment,it provides bridge between ancient planning with metaheuristic planning to supply a directions to researchers for that specialize in ancient planning to use metaheuristic planning on cloud computing systems[6]. Section2 of this paper contains Meta-heuristic algorithms. Section 3 includes an overview of meta-heuristic methods for service composition and Section 4discusses theoretical results and comparison. Last section of the paper includes conclusion and future work.

2. META-HEURISTIC ALGORITHMS

Heuristic and Randomization tends to metaheuristic. Approximate algorithms don't guarantee to find the optimal solution in bounded time. Meta-heuristic algorithms are used when problems have large size and the main goal is to find the near optimal solution quickly for problem. Main objective of metaheuristic is efficiently and effectively explore the search space [8]. In the remaining section we will discuss various metaheuristics with the help of their pseudo-code.

Table I: Meta-heuristic Algorithms from 1975 to 2015

Sr.No.	*YEAR*	*ALGORITHM*
1	1975	Genetic algorithm by Holland
2	1977	Scatter Search by Glover.
3	1983	Simulated annealing by Kirkpatrick et al.
4	1986	Tabu search by Glover.
5	1989	Memetic algorithm.
6	1992	The ant colony algorithm
7	1995	particle swarm optimization
8	2000	Harmony search
9	2001	Bootstrap Algorithm
10	2002	Multi-objective optimization
11	2005	Glow worm swarm optimization
12	2005	Artificial Bee Colony Algorithm (ABC)
13	2006	Honey-bee mating optimization.
14	2007	Intelligent Water Drops.
15	2008	Yang introduces firefly algorithm.
16	2008	The Monkey Search
17	2009	The League Championship Algorithm (LCA)

18	2009	Cuckoo search.
19	2009	Gravitational Search Algorithm
20	2009	Virus Optimization Algorithm by Josue Cuevas et al.
21	2010	Yangintroduce bat algorithm
22	2011	Galaxy-based Search Algorithm.
23	2011	Spiral optimization
24	2012	Differential Search Algorithm
25	2013	Artificial Cooperative Search Algorithm (ACS)
26	2015	Election Algorithm

2.1. Genetic Algorithm(1975)

Genetic algorithm invented by Holland is a meta-heuristic based on the concepts of natural selection and genetics widely used in different areas such as grid scheduling [1].Firstly we have a tendency to generate AN initial populationcomprises chromosomes (individual resolution to the problem), then choose parent from this population.Then apply crossover and mutation operators on the selective parents to make new off-springs. A replacement generation is created selectively, in line with the fitness values, a number of the oldsters and offspring. At last, these off-springs replace the current individuals in the population and the process repeats again [2].

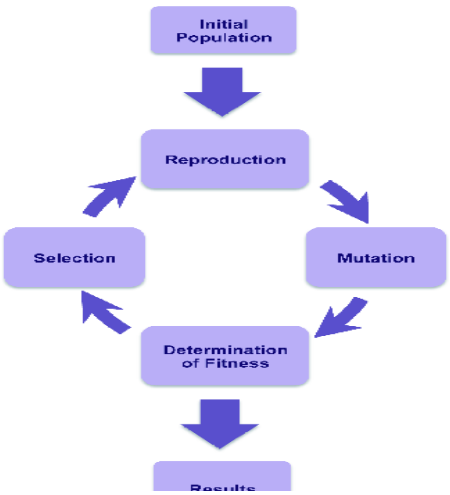

Figure 2: Basicsteps of genetic algorithm [12]

In this manner genetic algorithms try to mimic the human evolution to some extent.Genetic algorithmhasn't guaranteed the optimum results but always give the solution that close to optimum or optimum solution. The time consumed by the optimization algorithm is also high since it involves so many parameters.

1. Initialize population;
2. Evaluate population;
3. **While** (!*stopCondition*) **do**
4. Select the best-fit individuals for reproduction;
5. Breed new individuals through crossover and mutation operations;
6. Evaluate the individual fitness of new individuals;
7. Replace least-fit population with new individuals;

Figure 3: Generic pseudocode of GA procedure [19]

2.2 Scatter Search Global Optimization (1977)

Population-based Scatter Search Metaheuristic is presented by Fred Glover that makes solution by using diversification (variegation) and intensification methods (It combines others named as reference set in a set of solution) [21].

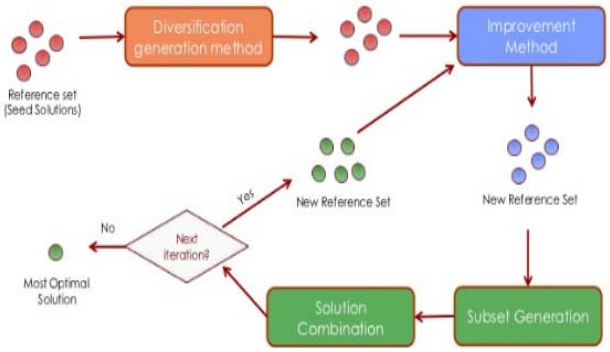

Figure 4: Working Steps of Scatter Search[22]

Procedure Sequential Scatter Search
begin
repeat
CreatePopulation(*Pop*,*PopSize*);
repeat
Generate ReferenceSet(*RefSet*,*RefSetSize*);
repeat
SelectSubset(*SubSet*,*SubSetSize*);
CombineSolutions(*SubSet*, *CurSol*);
ImproveSolution(*CurSol*, *ImpSol*);
until (*StoppingCriterion*1);
UpdateReferenceSet(*RefSet*);
until (*StoppingCriterion*2);
until (*StoppingCriterion*3);

Figure 5: Pseudocode of Scatter Search Metaheuristic [21]

2.3. Simulating Annealing (1992)

Input: ProblemSize, $iterations_{max}$, $temp_{max}$
Output: S_{best}
$S_{current} \leftarrow$ CreateInitialSolution(ProblemSize)
$S_{best} \leftarrow S_{current}$
For ($i = 1$ To $iterations_{max}$)
  $S_i \leftarrow$ CreateNeighborSolution($S_{current}$)
  $temp_{curr} \leftarrow$ CalculateTemperature(i, $temp_{max}$)
  If (Cost(S_i) $\leq$ Cost($S_{current}$))
    $S_{current} \leftarrow S_i$
    If (Cost(S_i) $\leq$ Cost(S_{best}))
      $S_{best} \leftarrow S_i$
    End
  ElseIf (Exp($\frac{CostS_{current} - CostS_i}{temp_{curr}}$) > Rand())
    $S_{current} \leftarrow S_i$
  End
End
Return (S_{best})

Figure 6:Pseudocode of PSO [23]

Simulating Annealing which belongs to local search algorithms was proposed by Kirkpatrick to position a best approximation to the global optimum of large search space (in discrete search space) for any particular problem or we can say that for Global Optimization Problem [1].It is very popular with mathematicians.

2.4. Tabu Search (1986)

Tabu search metaheuristic is design to solve optimization problems having finite solution set. It uses adaptive memory to store some of the attribute solution instead of whole solution, which makes it more flexible [23].

```
Input: TabuList_size
Output: S_best
S_best ← ConstructInitialSolution()
TabuList ← ∅
While (¬ StopCondition())
  CandidateList ← ∅
  For (S_candidate ∈ Sbest_neighborhood)
    If (¬ ContainsAnyFeatures(S_candidate, TabuList))
      CandidateList ← S_candidate
    End
  End
  S_candidate ← LocateBestCandidate(CandidateList)
  If (Cost(S_candidate) ≤ Cost(S_best))
    S_best ← S_candidate
    TabuList ← FeatureDifferences(S_candidate, S_best)
    While (TabuList > TabuList_size)
      DeleteFeature(TabuList)
    End
  End
End
Return (S_best)
```

Figure 7: Pseudocode of Tabu Search[23]

Forbidding (control what enters in the list), Freeing (control what and when exist from the list) and short-term (intermediate between forbidding and freeing to select trial solution) are three main strategies on which this algorithm works.

2.5. Memetic Algorithm (1989)

It was introduced by Moscato. It is combination of Evolutionary Algorithms and local search .It is close to Genetic Algorithms by having capability of performing local refinements [24].

```
Input: ProblemSize, Pop_size, MemePop_size
Output: S_best
Population ← InitializePopulation(ProblemSize, Pop_size)
While (¬StopCondition())
  For (S_i ∈ Population)
    Si_cost ← Cost(S_i)
  End
  S_best ← GetBestSolution(Population)
  Population ← StochasticGlobalSearch(Population)
  MemeticPopulation ← SelectMemeticPopulation(Population, MemePop_size)
  For (S_i ∈ MemeticPopulation)
    S_i ← LocalSearch(S_i)
  End
End
Return (S_best)
```

Figure *8:* Pseudocode of Memetic Algorithm [23]

2.6. Ant Colony Optimization (1992)

Ant colony optimization is a metaheuristic proposed by Dorigo and Birattari (1992) for solving hard optimization problems. It is a probabilistic technique to find solutions of problems like good paths through graph. Firstly one ant find a shortest path from source (from its current position) to destination (food). When it move back to colony, it leaves markers (pheromones), these markers show that the path has food, so others ants follow that path and get food [4].

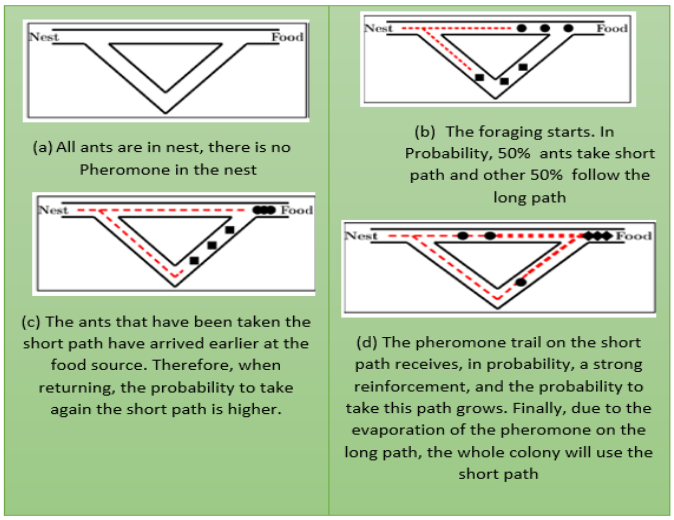

Figure 9: Behaviour of real ants [4]

```
Begin;
  Initialize the pheromone trails and parameters;
    Generate population of m solutions (ants);
    For each individual ant k∈m: calculate fitness (k);
    For each ant determine its best position;
    Determine the best global ant;
    Update the pheromone trail;
  Check if termination=true;
End;
```

Figure 10:Pseudocode of ACO [19]

2.7.Particle Swarm Optimization (1995)

Particle swarm algorithm is an intelligent optimization algorithm invented by Kennedy, Eberhart and Shi. It mimics human or insects social behavior. PSO applied in many areas like Artificial Neural Network, Fuzzy System Control, Function Optimization and many other areas where Genetic-Algorithm can be applied [5]. PSO based upon communication, measurements and learning parameters. In PSO, for some problem and function, the particles are placed in the search space, and do evaluation. Its working is as the movement of organisms in a bird flocking. Initially system searches for optima from the population of random solutions. PSO has no crossover and mutation operators as GA. In particle swarm optimization, by following the current optimum solution particles fly through this solution. Moreover, PSO has been used for variety of applications [7].

```
Begin;
   Generate random population of N solutions
   (particles);
   For each individual i∈N: calculate fitness (i);
   Initialize the value of the weight factor, ω;
      For each particle;
         Set pBest as the best position of particle i;
         If fitness (i) is better than pBest;
         pBest(i) = fitness (i);
      End;
      Set gBest as the best fitness of all particles;
      For each particle;
         Calculate particle velocity according to Eq. (3);
         Update particle position according to Eq. (4);
      End;
      Update the value of the weight factor, ω;
   Check if termination = true;
   End;
```

Figure 11: Pseudocode of PSO[19]

2.8.Harmony Search (HA) (2000)

In 2000, it was firstly proposed by Geemto solve the improvement drawback (of water distribution networks). This metaheuristic belongs to the class of Computational Intelligence[25].

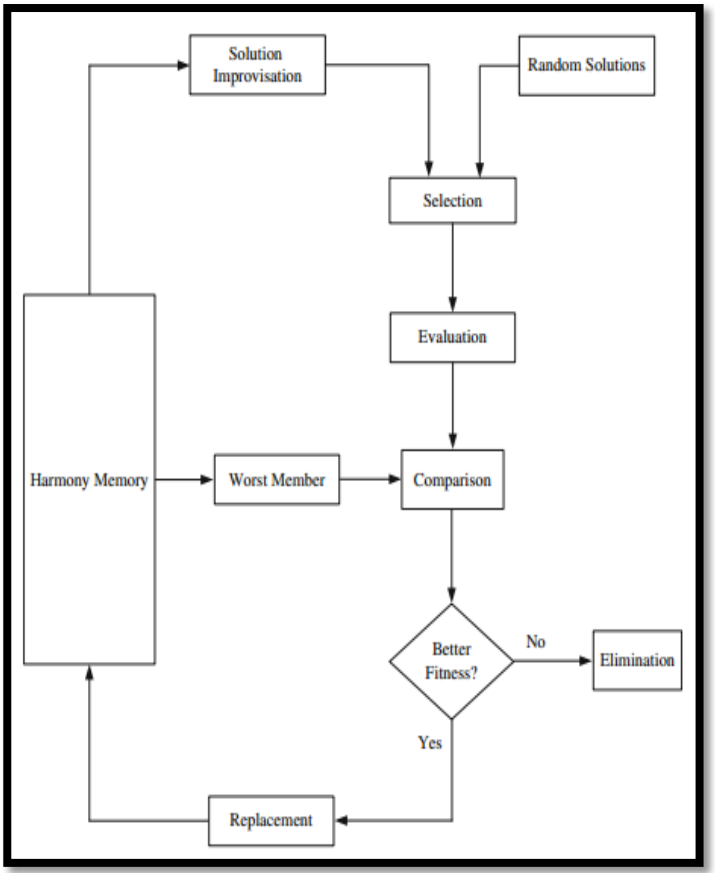

Figure 12: HA Method [25]

The adjustment of a pitch selected from the harmony memory is typically linear, for example for continuous function optimization:

$$x' \leftarrow x + range \times \epsilon$$

Where vary could be a user parameter (pitch bandwidth) to manage the scale of the changes, and ∈ could be a uniformly random variety ∈ [-1, 1][23].

```
Building the Harmony Memory;
Begin
  for (i=1 to HMS) do
     x_i=∅;
    while the coverage demand of night shift not met do
      allocate two patterns of one-week from the pool of solutions that include night shifts cyclically to x_i
    End while
    while not reaching the end of x_i do
      allocate two patterns of one-week from the pool of solutions of morning and evening shifts
    End while
    calculate the PV of x_i
    add x_i to HM
  end for
    sort the solutions based on its associated penalty values in HM.
End
```

Figure 13.1: The pseudo-code of building the HM (Harmony Memory)[23]

```
Step 3 Improvise a new harmony (generate new solution)
begin
   x^new=∅
  for i=(1 to NI) do
            for j=(1 to n) do /* n is the number of nurses */
              for k=(1 to sp) do /* sp is the number of shift patterns */
              if (rand(0,1) =HMCR) then
                  choose p_i randomly from the HM:
                     if (rand(0,1)=PAR) then
                   pitch adjusted x^new [p_i] by:
                   x^new [p_i] = x^new [p_i]+ rand BW
                  else
                     x^new [p_i]=x[p_i]
                  endif
                  change the rate of PAR; /* According to equation (2) */
             else choose sp_i randomly from the pool of solutions
                x^new [p_i]= x[p_i]
            endif
            change the rate of HMCR; /* According to equation (1) */
           endfor
       endfor
  endfor
end
```

Figure 13.2: Improvisation step Harmony Search[23]

2.9. Bootstrap Algorithm (2001)

Bootstrapping introduced by Hanseth and Aanestad is based on statistical sampling, because it evaluates a statistic of the value of a parameter of a population. This evaluated statistic use to refine the solution of that population. This approach is used in computational intelligence [26].

$B \leftarrow$ number of bootstrap replications;
$Vboot \leftarrow$ vector of length B;
$Vsample \leftarrow$ vector of length n_j;
for $b = 1 : B$
  for $i = 1 : n_j$
    $Vsample \leftarrow$ sample $f_{T_{ij}}(t_i)$ with replacement;
  $Vboot \leftarrow g(Vsample)$;
  clear $Vsample$;
$Nsamples \leftarrow$ number of samples in $Vboot \in [\beta_{min}, \beta_{max}]$;
$\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}] \approx Nsamples/B$.

Figure 14: Pseudocode of Bootstrap Algorithm [27]

2.10. Multi-objective Optimization (2002)

It has enormous practical importance because al real world problems are suited to be simulated using multiple conflicting objectives. In any problem Multi-objective Optimization helps to optimize more than one functions simultaneously [28]. Deb et al. propose NSGA-II for multi-objective.

2.11. Glow worm swarm optimization (2005)

It is new intake in swarm intelligence proposed by Krishnanand and Ghose. ACO and PSO are previous swarm intelligence methods have been applied on many problems.

This is very simple model with fast overlapping rate and parameters (which are less adjustable), in routing, pattern recognition, combinatorial optimization and many more problems [29].

(a) Initialization: $n, l_0, r_0, s, n_t, \rho, \gamma, \beta, r_s, T_{\max}, p$;
(b) While $(t \le T_{\max})$ then do:
  {
    for $i = 1$ to n; do:
      $l_i(t) = (1-\rho)\, l_i(t-1) + \gamma J(x_i(t))$;
      for each i do:
        $\{N_i(t) = \{j : \|x_j(t) - x_i(t)\| \le r_d^i(t); l_i(t) \le l_j(t)\}$;
      for each $j \in N_i(t)$; do:
        $\{p_{ij}(t) = \dfrac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}$;
        $j = select(p_{ij})$;
        $st = s/(1 - \exp(-20 \times (t/T_{\max})^p))$;
        $x_i(t+1) = x_i(t) + st\left(\dfrac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right)$;
        $r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t), \beta(n_t - |N_i(t)|)\}\}$;}
    $t = t + 1$;}

Figure 15: Pseudocode of Glow worm swarm optimization [29]

2.12. Artificial Bee Colony Algorithm (ABC) (2005)

This optimisation algorithmic program was declared by Karaboga depends on the intelligent search behaviour of honey bee swarm. After finding the solution, we say that the performance of ABC is better than others.

(1) Generate the initial population x_i $(i = 1, 2, \ldots, SN)$
(2) Evaluate the fitness (fit(x_i)) of the population
(3) Set cycle to 1
(4) Repeat
(5) For each employed bee {
Produce new solution v_i by using (2)
Calculate its fitness value fit(v_i)
Apply greedy selection process}
(6) Calculate the probability values P_i for the solution (x_i) by (3)
(7) For each onlooker bee {
Select a solution x_i depending on P_i
Produce new solution v_j
Calculate its fitness value fit(v_j)
Apply greedy selection process}
(8) **If** there is an abandoned solution for the scout,
then replace it with a new solution which will be randomly produced by (4)
(9) Memorize the best solution so far
(10) Cycle = cycle +1
(11) **Until** cycle = MEN

Figure 16: Pseudocode of ABC [30]

2.13. Honey-bee mating optimization(2006)

It is typical swarm based optimization technique introduced by Haddad et al. in which search algorithm is based on the behavior or process of mating inrealhoney-bees[30].

Initialize (High parameters for half of drones and low parameters for rest of them)
While not done **do**
The algorithm starts with the mating–flight
Creation of new broods
Use of workers
Adaptation of workers' fitness
Replacement of weaker queens by fitter broods.
For each drone$_i$ in hive **do**
Generate a new string genome
Evaluate the new string genome
If *f*(new string genome)>*f*(old string genome) **then**
Accept the new string genome
End if
Select *Se* queens from hive
Generate the reinforcement signal vector
Update LAs of drone $_i$
End for
In a Specified repetition change the string genome of drones (with high and low parameters) with each other.
End while

Figure 17: Pseudocode of Honey-bee mating[30]

2.14. Intelligent Water Drops (IWD) (2007)

This optimization algorithm introduced by Hamed Shah-Hosseini is inspired by the natural water drops and their actions and reactions that occur between water drops that flows [31].

Static parameter initialization
a) Problem representation in the form of a graph
b) Setting values for static parameters
Dynamic parameter initialization: soil and velocity of IWDs
Distribution of IWDs on the problem's graph
Solution construction by IWDs along with soil and velocity updating
a) Local soil updating on the graph
b) Soil and velocity updating on the IWDs
Local search over each IWD's solution (optional)
Global soil updating
Total-best solution updating
Go to step 2 unless termination condition is satisfied

Figure 18: Pseudocode of IWD [18]

2.15. Firefly algorithm(2008)

It is a nature impressed rule impressed by the flashing behaviour of fireflies.

```
Parameters: n, β0, γ
Initialise the fireflies population x_i randomly
Compute f(x)
while stop condition not met do
    i^min = arg min_i {f(x_i)}
    x_i^min = arg min_{x_i} {f(x_i)}
    for i = 1 to n do
        for j = 1 to n do
            if f(x_j) < f(x_i) then {Move firefly i towards j}
                Calculate distance r_j
                Obtain attractiveness: β ← β0 e^(−γ r_j)
                Generate a random solution u_i
                for k = 1 to d do
                    x_{i,k} = (1 − β) x_{i,k} + β x_{j,k} + u_{i,k}
                end for
            end if
        end for
    end for
    Generate a random solution u
    for k = 1 to d do {Best firefly moves randomly}
        x_{i^min,k} = x_{i^min,k} + u_k
    end for
    Compute f(x)
    Find the current best
end while
Postprocess results and visualisation
```

Figure 19: Pseudocode of Firefly Algorithm [30]

2.16. The Monkey Search (2008)

Proposed by Mucherino and Seref and this nature galvanized improvement algorithmic rule galvanized by the food finding behaviour of monkeys [32].

```
Part I: Set parameters and initialize
    Randomly generate initial population of NP positions of the monkeys (X_ij)
    i=1,2,.....NP, and j=1,2,..NV
    Define low and high boundaries of X_i
    Assign values to parameters: a, b, c and d
Part II: Algorithm loops
    For I = 1: I_max (max. iteration)
     Climb process loop
      For counter l= 1: Nc (number of cycles)
          Randomly generate vector ΔX(l)
          Generate the pseudo-gradient of the objective function f'(X_i)
          Generate new monkeys' position Y_i using a and f'
          Update X_i with Y_i if feasible
      End For N_c
     Watch-jump process
      Generate new Y_i(I) from (X_i-b, X_i+b)
      If -f(Y_i) ≥ -f(X_i)
                Update X_i(I) with Y_i(I) if feasible (i.e., within limits)
      End If
     Apply climb process loop again
      Rank the positions and find current best solution so far
     Somersault process
      Estimate Pivot P(I) from X_ij, c and d
      Calculate Y_i(I) around the pivot
      Update X_i(I) with Y_i(I) if feasible and repeat somersault until feasible
      Rank the positions and find current best solution
      If (the new monkey's position violates its boundary limits)
                Use Search-based function to handle constraints
      End If
    End For I
    Return the highest mountaintop (-f(X)) and its position X
```

Figure 20: Pseudocode of Monkey Search [32]

2.17. The League Championship Algorithm (LCA) (2009)

LCA Optimization rule was given by Ali HusseinzadehKashan. In search space it tries to improve the population by achievable solution [33].

1. *Initialize the league size (L) and the number of seasons (S); t=1;*
2. *Generate a league schedule;*
3. *Initialize team formations (generate a population of L solutions) and determine the playing strengths (function or fitness value) along with them. Let the initialization be also the teams' current best formation;*
4. *While t≤S.(L-1)*
5. *Based on the league schedule at week t, determine the winner/ loser among every pair of teams using a playing strength based criterion;*
6. *t=t+1;*
7. *For i=1 to L*
8. *Devise a new formation for team i for the forthcoming match, while take into account the team's current best formation and previous week events. Evaluate the playing strength of the resulting arrangement;*
9. *If the new formation is the fittest one (that is, the new solution is the best solution achieved so far for the ith member of the population), hereafter consider the new formation as the team's current best formation;*
10. *End for*
11. *If mod(t,L-1)=0*
12. *Generate a league schedule;*
13. *End if*
14. *End while.*

Figure 21: Pseudocode of LCA [33]

2.18. Cuckoo search. (2009)

It was galvanized by obligate brood mutuality some cuckoo species by giving birth their eggs within the nests of alternative host birds (of alternative species).In nest, each egg represent as a solution and a cuckoo egg represent a new solution [34].

begin
Objective function $f(x)$
Generate initial population of n host nest
Evaluate fitness and rank eggs
while (t > MaxGeneration) or Stop criterion
$t = t + 1$
Get a cuckoo randomly/generate new solution by Lévy flights
Evaluate quality/fitness, F_i
Choose a random nest j
if ($F_i > F_j$)
Replace j by the new solution
end if
Worst nest is abandoned with probability P_a and new nest is built
Evaluate fitness and Rank the solutions and find current best
end while
Post process results and visualization
end

Figure 22: Pseudocode of Cuckoo Search [34]

2.19. Gravitational Search Algorithm (2009)

Initialize bodies;
Repeat
Foreach (body in bodies) {
body$_i$.CalculateFitnessValue;
body$_i$.CalculateAcceleration (using the equation 9)
body$_i$.UpdateVelocity (using the equation 10)
body$_i$.UpdatePosition (using the equation 11)
}
Until (terminated condition is met or certain number of iterations are performed)

Figure 23: Pseudocode of Gravitational Search Algorithm [35]

Rashedi planned the gravitative search algorithmic rule that relies on the law of gravity and mass interactions.

2.20. Virus Optimization Algorithm (2009)

This optimization algorithm introduced for the continuous optimizationproblems. It simulates the behavior of the viruses when they attack a host cell, so its name is virus Optimization Algorithm [37].

2.21. Bat Algorithm(2010)

Bat algorithm was inspired and galvanized by the behaviour of crackers, results area unit higher than the Genetic algorithmic rule and PSO algorithmic rule[30].

Objective function $f(x), x = [x_1, x_2, \ldots, x_d]^T$
Initialize the bat population $x_i (i = 1, 2, \ldots, n)$ and v_i
Define pulse frequency f_i at x_i Initialize pulse rates r_i and the loudness A_i
While (t < *Max number of iterations*)
Generate new solutions by adjusting frequency,
and updating velocities and locations/solutions [(1)]
if (rand > r_i)
Select a solution among the best solutions
Generate a local solution around the selected best solution
end if
Generate a new solution by flying randomly
if (rand < A_i & $f(x_i) < f(x_*)$)
Accept the new solutions
Increase r_i and reduce A_i
end if
Rank the bats and find the current best x_*
end while
Postprocess results and visualization.

Figure 24: Pseudocode of Bat Algorithm [36]

2.22. Galaxy Based Search Algorithm (GbSA) (2011)

This is new novel metaheuristic algorithm proposed by Hamed Shah-Hosseini (2011) for continuous optimization. Here we give a brief introduction on GbSA-PCA (Principal components analysis) and GbSA-MLT (Multi Level Thresholding). From the initial solution, The GbSA itself moves spirally, which looks same as core of the galaxy. Spiral like arm moves to search solution space to find the best optimal solution. After finding the new solution, a native search algorithmic program is activated to form the most effective local resolution. For GbSA-PCA, here PCA is developed as an improvement downside, and therefore the GbSA can notice the simplest optimum answer [13]. For GbSA-MLT, a hill-climbing algorithmic rule could also be select by the native search algorithmic rule or anybody of its changed versions [11]. This method perennial once more and once more till a termination condition(s) is happy.

Procedure GbSA

$S \leftarrow GenerateInitialSolution$

$S \leftarrow LocalSearch(S)$

While (termination condition is not met) do

- $Flag \leftarrow False$
- $SpiralChaoticMove(S, Flag)$
- If ($Flag$) then
  - $S \leftarrow LocalSearch(S)$

End while

Return S

End procedure

Figure 25: Pseudocode of GbSA [13]

2.23. Spiral Optimization(2011)

Spiral optimisation could be a new metaheuristic projected by Tamura and Yasuda to Combined Economic and Emission Dispatch.

Procedure SpiralSearch(*maxIteration, maxRetry, maxRW*)

```
1  //H and P are hydrophobic & polar amino acids.
2  //AA-amino acid array of the protein
3  initialise()
4  initTabuList()
5  for (i=1 to maxIteration) do
6      mv ⟵ selectMoveForH()
7      if (mv !=null) then
8          applyMove(mv)
9          updateTabuList(i)
10     else
11         mv ⟵ selectMoveForP()
12         if (mv!=null) then
13             applyMove(mv)
14     evalute(AA)
15     if (not improved) then
16         retry++
17     else
18         improvedList ⟵addSolAtTop()
19         retry=0
20         rw=0
21     if retry ≥ maxRetry then
22         randomWalk(maxPull)
23         resetTabuList()
24         rw++;
25     if rw ≥ maxRW then
26         relayRestart(improvedList)
27         resetTabuList()
```

Figure 26: Pseudocode of Spiral Optimization [38]

2.24. Differential Search Algorithm(2012)

This algorithm is based on stochastic searching to find the optimal solutions[39].

Generate the initial population of individuals
Do
  For each individual j in the population
  Choose three numbers n_1, n_2, *and* n_3 *that is,* $1 \le n_1, n_2, n_3 \le N$ *with* $n_1 \ne n_2 \ne n_3 \ne j$
  Generate a random integer $i_{rand} \in (1, N)$
  For each parameter i

$$y^{i,g} = x^{n_1,g} + F(x^{n_2,g} - x^{n_3,g})$$

$$z_j^{i,g} = \begin{cases} y_j^{i,g} if\ \text{rand}() \le \text{CR}\ or\ j = j_{\text{rand}} \\ x_j^{i,g} \text{otherwise} \end{cases}$$

  End For
  Replace $x^{i,g}$ *with the child* $z^{i,g}$ *if* $z^{i,g}$ *is better*
End For
Until the termination condition is achieved

Figure 27: Pseudocode of Differential Search Algorithm [39]

2.25.Artificial Cooperative Search Algorithm (ACS) (2013)

As we know that there are three type of algorithms complete (exact), approximate (heuristic) and hybrid (combination of complete and approximate algorithms).Results found by hybrid are better than complete and approximate algorithms. When the size of problem becomes large we need a parallel computing and cooperative search is a category of parallel search [40].

2.26. Election Algorithm (2015)

There are various algorithms used in distributed systems. The role of each and every process in distributed system is same. Election algorithms are used to select the coordinator among various processes by using some methods.

Election may be needed at initialization or at the stage when selected coordinator may get crashed or finished. Every process have unique ID number, the selection of new coordinator is based on its highest ID number [14].

"The Bully algorithm" and **"A Ring algorithm",** these are two traditional election algorithms.

Election Algorithm

BEGIN
  Generate initial population;
  Compute eligibility of each individual;
  Create parties: initial candidates and their supporters
  REPEAT //* advertising campaign (ad days) *//
    For candidate size do
      // positive advertisement
      Candidates advertise their plans and improve their stance by learning new ideas;
      // negative advertisement
      Candidates try to attract the supporters from other parties toward themselves;
      // coalition
      Candidates collate if they have same ideas;
      // revision the condition of parties
      Reevaluate the eligibility of candidates;
    END FOR
  UNTIL population has converged //* (Election Day) *//
END

Figure 28: Pseudocode of Election algorithm [20]

a. The Bully algorithm [18]

(Process P calls an election when it notices that the coordinator is no longer responding. Process P sends an election message to all higher-numbered processes in the system. If no process responds, then P becomes the coordinator)

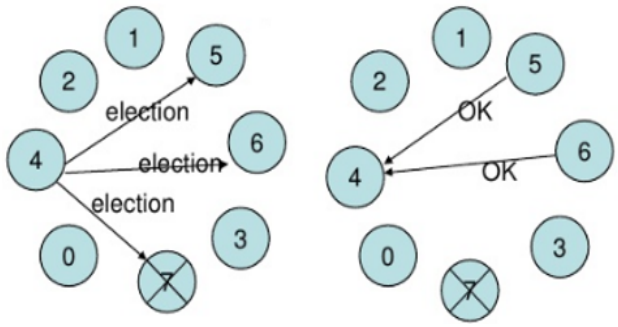

Figure 29: Bully Algorithm

b. A Ring algorithm [18]

Processes are arranged in a logical ring and each process knows the order of the ring of processes.

P thinks the coordinator has crashed; builds an ELECTION message which contains its own ID number. Send to first live successor. Each process adds its own number and forwards to next. OK to have two elections at once

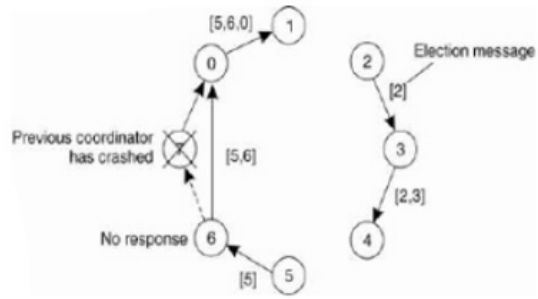

Figure 30: Ring Algorithm

3. SOME EXISTING METAHEURISTICSFOR SERVICE COMPOSIION IN CLOUD COMPUTING

Afterdiscussing evolution of various metaheuristics methods for service composition in cloud computing, we are now going to compare them on the basis of their main objective, advantages and disadvantages.

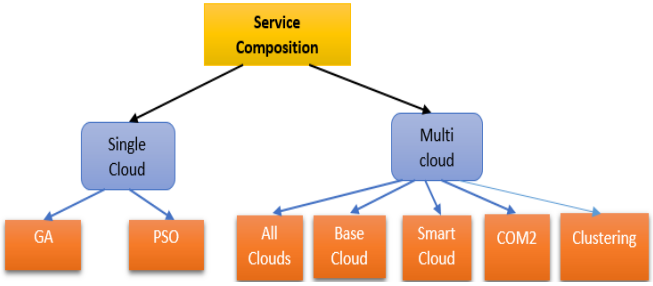

Figure 31:Service composition methods in cloud [10]

We start with genetic algorithm:

3.1. Genetic Algorithm

Web Service composition approach of genetic algorithm proposed by Wang. et al [1]. This composition is for the service providers and geo-distributed cloud. Process is based on the genetic algorithm; it is for those service providers who want minimized SLA violation.

Same as the basic working of GA, It generates initial population, then operators of GA (crossover and mutation) are applied on the current population and near optimum or optimum solution is generated. This work is done in many iterations depends on the results and population. On every iteration, fitness value of resulted solution is evaluated and then new population is generated on the behalf of **choice**, crossover and mutation operations. The whole method is continual once more and once more till we have a tendency to don't get the specified optimum results.But this method is only for a single cloud environment not for the multiple cloud and increase the user's experience as well [2].

3.2. Ant Colony Optimization

ACO-WSC (ACO for web service composition) proposed by Yu, et al., in multiple cloud environment [1]. To make cloud combinations, ants (artificial ants) travel on the logical graph G=(V,E), then those ants which are feasible to select cloud combinations use the minimum number of clouds. In graph G(V,E) each vertices represents a cloud server, which are connected by each other by edges E. According to the markers (pheromone) produced by ant, all other ants choose that optimum path. Disadvantage of this method is that load balancing among cloud servers is inefficient, but this method is good to find the optimal cloud composition in different cases [16].

3.3. PSO (Particle Swarm Optimization)

PSO that's with a combinatorial improvement projected by Ludwig is forthe single cloud, which is firstly appertained to the Hungarian algorithm, then referred to Munkres [1].After every PSO iteration, Munkres algorithm can be applied.

We choose some percent of the particle on which we can apply Munkres Algorithm. After completing the process when we achieve result or any kind of improvement then those selected particles will be updated with optimized workflow, same for the next iteration and so on. Runtime of using Munkres algorithm in PSO is shorter as compare to PSO algorithm. Munkresundergoes from cubic time complexity and PSO undergoes from premature convergence [15].

3.4. Election Algorithm

It may help in resource scheduling in cloud environment. It may help to resolve the Load Balancing issue in cloud.

4. COMPARISON

Now we have a tendency to do comparison of existing meta-heuristic techniques for service composition in cloud computing.GA and PSO can find optimal cloud composition in single cloud environment whereas ACO is in all multiple cloud environment (MCE).

In the **comparison of GA and ACO**, ACO finding optimal cloud composition have more effective in running time and cost, better performance than GA.**GA and PSO**for single cloud environment for web service composition set benefits getting from other clouds. In PSO, after every iteration, other algorithm i.e. Hungarian or Munkres is used, so it hasshort runtime and fast execution. Whereas, GA is used individually for web service composition.

PSO and GA are common in some functions likeinitially, both are have set of population. Unlike GA, PSO has no evolution operators like mutation, crossover and selection and PSO use other algorithm after each iteration,so it is more efficient(less runtime, use less number of functions, having memory) than the GA.

Table II: Comparison of methods with advantages and disadvantage

NAME OF THE ARTICLE	*MAIN IDEA*	*ADVANTAGES*	*DISADVANTAGES*
A genetic-based approach to web service composition in geo-distributed cloud environment [17]	Finds optimal or near optimal solution for cloud composition by generating initial population which is feasible to select a cloud combinations.	Minimizes the SLA violations. Reduces complexity. Maximizes user experience. Considers QOS of network.	Limits benefits received from other clouds. Consumed time is high.
A survey on bio-inspired algorithms for web service Composition [15] (PSO method)	PSO algorithm that combined with a Munkres algorithm that applied after every PSO iteration for fast execution and minimized a run time.	Performs very well inservice-oriented environments. Considers several services simultaneously. Considers scalability. Achieves reasonable runtime. Good balance between execution time and fitness value.	Limits benefits received from other clouds. Cubic time complexity. Premature convergence.
QoS-based Dynamic Web Service Composition with Ant Colony Optimization [16] (ACO method)	Vertices are represented as cloud, connected with each other by edges E and ants satisfy user requirement in clouds.	In this, nodes are represented as cloud and ants satisfy user requirement in clouds.	Load balancing is inefficient. Start node is selected randomly.
Elections in a Distributed Computing System.	Used to select the coordinator among several processes in distributed systems.	As we know, this algorithm is for the distributed systems, so it may help to resolve the Load Balancing issue of cloud computing.	

5. CONCLUSION

In this paper, we have discussed the growth and evolution of twenty six different metaheuristic algorithms which are used in various application areas such as scheduling, mathematical optimization, software engineering etc. Some of the metaheuristics for service composition in cloud computing along with their advantages and disadvantages of are also discussed and compared with each other in single and multi-cloud environment. Theoretical results and analysis showed that no single metaheuristics is efficient in all kinds of real life problems.

6. REFERENCES

[1] Review and Comparison of Meta-Heuristic Algorithms for Service Composition in Cloud Computing by Saied Asghari, NimaJafariNavimipour in Majlesi Journal of Multimedia Processing Vol. 4, No. 4, December 2015

[2] Genetic algorithm learning and the cobweb model by JasminaArifovic inJournal of Economic Dynamics and Control 18 (1994) 3-28. North-Holland

[3] A Review of Population-based Meta-Heuristic Algorithm by Zahra Beheshti, SitiMariyamHj. Shamsuddin in Int. J. Advance. Soft Comput. Appl., Vol. 5, No. 1, March 2013 ISSN 2074-8523; Copyright © ICSRS Publication, 2013 www.i-csrs.org

[4] Ant colony optimization: Introduction and recent trends By Christian Blum ALBCOM, LSI, UniversitatPolitècnica de Catalunya, JordiGirona 1-3, Campus Nord, 08034 Barcelona, Spain Accepted 11 October 2005 www.elsevier.com/locate/plrev

[5] Particle Swarm Optimization: Developments, Applications and Resources by Russell C. Eberhart in 0-7803-6657-3/01/$10.0©2001 IEEE

[6] Metaheuristic Scheduling for Cloud: A Survey by Chun-Wei Tsai and Joel J. P. C. in IEEE SYSTEMS JOURNAL, VOL. 8, NO. 1, MARCH 2014.

[7] Particle swarm optimization, an overview by Riccardo Poli · James Kennedy · Tim Blackwell in Received: 19 December 2006 / Accepted: 10 May 2007© Springer Science + Business Media, LLC 2007

[8] Handbook of Metaheuristic second edition, edited by Michel Gendreau, Jean-Yves Potvin, INTERNATIONAL SERIES IN OPERATIONS RESEARCH AND MANAGEMENT SCIENCE.

[9] Mastering Cloud Computing (Foundation and Application Programming by RajkumarBuyya, Christian Vecchiola and S.ThamaraiSelvi.

[10] Service composition mechanisms in the multi-cloud environments: a survey by Saied Asghari and NimaJafariNavimipour in International Journal of New Computer Architectures and Their Applications, vol. 6, no. 2, 2016, p. 40+. Academic OneFile, Accessed 23 Apr. 2017.

[11] E.H.L. Aarts and J.K. Lenstra, "Local search in combinatorial optimization," in Discrete Mathematics and Optimization, Eds.Wiley, Chichester, UK, 1997.

[12] Fitting of dust spectra with genetic algorithms, Intersteller and circumstellar matter, Issue A&A Volume 516,June-July 2010, 24 june 2010.

[13] Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization by Hamed Shah-Hosseini in Int. J. Computational Science and Engineering, Vol. 6, Nos. 1/2, 2011

[14] Elections in a Distributed Computing System by HECTOR GARCIA-MOLINA, MEMBER, IEEE in IEEE TRANSACTIONS ON COMPUTERS, VOL. C-31, NO. 1, JANUARY 1982.

[15] A survey on bio-inspired algorithms for web service Composition by Lijuan Wang, L., Shen, J. & Yong, J. in Proceeding of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (pp. 569-574). USA: IEEE, research-pubs@uow.edu.au.

[16] QoS-based Dynamic Web Service Composition with Ant Colony Optimization by Wei Zhang, Carl K. Chang in 2010 IEEE 34th Annual Computer Software and Applications Conference.

[17] A genetic-based approach to web service composition in geo-distributed cloud environment by Dandan Wang, Yang Yang in Computers & Electrical Engineering, Volume 43, April 2015, Pages 129–141

[18] Taking screenshot from ppts.

[19] Comparison among five evolutionary-based optimization algorithms by EmadElbeltagi, TarekHegazy, Donald Grierson, 19 January 2005 in Advanced Engineering Informatics 19 (2005) 43–53.

[20] Election algorithm: A new socio-politically inspired strategy by Emami, Hojjat, Derakhshan in Computer Engineering Department, Miandoab Branch, Islamic Azad University, Miandoab, Iran. Email: hojjatemami@yahoo.com.

[21] Parallelization of the scatter search for the p-median problem by F_elixGarc_ıa-L_opez, Bel_enMeli_an-Batista, Jos_e A. Moreno-P_erez, J. Marcos Moreno-Vega, Received 7 June 2002; accepted 19 November 2002 in Parallel Computing29 (2003) 575–589.

[22] Scatter Search by AstriPuspitasari, LuthfanPramono in Media Digital and Game Technology, InstitutTeknologi Bandung, 2012.

[23] Book- Clever Algorithms: Nature-Inspired Programming Recipes by Jason Brownlee phD.

[24] Krasnogor, N. and Smith, J. (2005), A tutorial for competent memetic algorithms: Model, taxonomy and design issues. IEEE Transactions on Evolutionary Computation, 9 (5). pp. 474-488. ISSN 1089-778X Available from: http://eprints.uwe.ac.uk/11069

[25] The Author(s) 2015 X. Wang et al., An Introduction to Harmony Search Optimization Method, Springer Briefs in Computational Intelligence, DOI 10.1007/978-3-319-08356-8_2

[26] Bootstrap - Inspired Techniques in Computation Intelligence; Bootstrap - Inspired Techniques in Computation Intelligence by RobiPolikar, Rowan University,Article·August2007,DOI:10.1109/MSP.2007.4286565;10.1109/MSP.2007.4286565 · Source: IEEE Xplore.

[27] Stochastic robustness metric and its use for static resource allocations by Jay Smith, Anthony A. Maciejewski, Howard Jay Siegel in Aug 2008 · Journal of Parallel and Distributed Computing.

[28] Multidisciplinary System Multidisciplinary System Design Optimization (MSDO) Design Optimization (MSDO) by Dr. Il Yong Kim in Massachusetts Institute of Technology - Prof. de Weck and Prof. Willcox Engineering Systems Division and Dept. of Aeronautics and Astronautics.

[29]Full Glowworm Swarm Optimization Algorithm for Whole Set Orders Scheduling in Single Machine by Zhang Yu and Xiaomei Yang in The Scientific World Journal Volume 2013 (2013), Article ID 652061, 6 pages http://dx.doi.org/10.1155/2013/652061.

[30] New inspirations in swarm intelligence: a survey by R.S. Parpinelli, H.S. Lopes in Int. J. Bio-Inspired Computation, Vol. 3, No. 1, 2011.

[31] The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm by Hamed Shah-Hosseini in Int. J. Bio-Inspired Computation, Vol. 1, Nos. 1/2, 2009.

[32] MAKHA-A New hybrid Swarm Intelligence Global Optimization Algorithm by Ahned M. E. Khalil, Seif-Eddeen K Fateen in Algorithms 8(2):336-366. June 2015, DOI:10.3390/a8020336.

[33] League Championship Algorithm: A new algorithm for numerical function optimization by Ali HusseinzadehKashan in 2009 International Conference of Soft Computing and Pattern Recognition.

[34] PCB Drill Path Optimization by Combinatorial Cuckoo Search Algorithm by Wei Chen Esmonde Lim, G Kanagaraj, S G Ponnambalam inThe Scientific World Journal, Feb 2014.

[35] A Meta-Heuristic Solution for Automated Refutation of Complex Software Systems Specified through Graph Transformations By VahidRafe, Maryam Moradi, Rosa Yousefian, Amin Nikanjam in Applied Soft Computing 2015 · April 2015.

[36] Cloud Model Bat Algorithm by JianXie, Liangliang Li, Mingzhi Ma in The Scientific World Journal, May 2014.

[37] Virus Optimization Algorithm (VOA): A Novel Metaheuristic for Solving Continuous Optimization Problems by Rodolfo Josue, Cuevas Juarez, Hung-Jie Wang, Yun-Chia Liang in, Oct 06, 2015.

[38]Spiral search: A hydrophobic-core directed local search for simplified PSP on 3D FCC lattice by Mahmood A Rashid, MA Hakim Newton, MdTamjidulHoque in BMC Bioinformatics · Article · Jan 2013 .

[39] Correction of Faulty Sensors in Phased Array Radars Using Symmetrical Sensor Failure Technique and Cultural Algorithm with Differential Evolution by S U Khan, I M Qureshi, F Zaman in The Scientific World Journal · Article · Jan 2014.

[40] Artificial Cooperative Search Algorithm for Parameter Identification of Chaotic Systems byOguzEmrahTurgut1 ,MertSinanTurgut 2* , Mustafa Turhan Coban1 in Turgut et al. / BitlisErenUniv J Sci&Technol / 5 (1), 11 – 17, 201