An Algorithm for Optimizing Web Services Composition by Combining GA and ACO Algorithms

Mahdi Bigham[1] and AbbasAli Rezaee[2],

University of Bojnord, Bojnurd, Iran, Payam-e Noor University, Mashhad, Iran

Summary
Today, service providing for users is one of the main issues and challenges for users for Web development. So, extensive researches are presented about the variety of Web services and how to combine them for giving an optimal response in terms of time and cost for users. Another issue that must be considered is the existence of different scenarios for different Web services composition and designing systems that have a flexible architecture to satisfy users. For this reason, soft wares that are being produced in Web services domain should have an optimal condition for Web services composition. In this article, we will examine parallel scenario for the combination of different Web services for making the best answer in the services search space and will use a Hybrid algorithm of Ant Colony Optimization and Genetic Algorithm (HACOGA) for increasing the chance of gaining the best optimal response. The results of the implementation of this method in MATLAB software show that HACOGA algorithm will work better than the optimal genetic algorithm.

Keywords:
Web Services Composition, Genetic Algorithm, HACOGA, Qualitative Factors.

1. Introduction

Today, service-oriented systems acquired special importance, because of the chance of activity in heterogeneous distributed environments. The user of these systems uses services that are provided by system components. In most cases, the needs of users wouldn't be resolved by the available single services and the desired service should be achieved by combining several services. One of the issues raised in the context of service-oriented systems is how to combine and manage them. Several methods have been provided for the composition operation [1-6]. In this article, we seek a way to manage services composition according to the service-oriented architecture layers. Due to problems of convergence speed and falling into the local optimum in current methods, we were seeking a method that in addition to convergence and adequate speed in responding from provided algorithm, method optimality would be clearly visible in large scale service composition problem. Combining GA and ACO algorithms and synchronized implementation of them, is the proposed solution for this problem. If these algorithms run simultaneously and then share their answers with each other, they can be responsible for optimality and not falling into the local optimum. In this article, the ACO algorithm early answers were used to mutate GA answers from local optimum and GA answers were used to extend the ants search space. The rest of this paper is organized as follows :

In section II, we describe different approaches that have already been proposed and use these descriptions in the next sections. Then in section III, we describe basic definitions of Web services composition problem. In section IV, we introduce and express the problem. Section V presents a way for dynamic optimal Web services composition and shows experimental results after initial evaluation. Section VI analyzes and presents the gained results.

2. Related Works

GA and ACO algorithms have been used in several methods such as TSP, Knapsack, Test Set Optimization, Neural Networks, AI and Signal Transmission problems. But these algorithms are intrinsically incomplete. GA has some disadvantages such as weak local search and slow convergence speed and is vulnerable against local optimum problem. ACO algorithm has some shortages such as slow beginning and depression in gaining the sufficient early pheromones .

Many researchers have developed a lot of algorithms to improve GA and ACO algorithms, because of the existing problems in each algorithm. Zhang in [7] uses multi-dimensional GA to solve the multipath services composition problem, but this algorithm does not support some complex structures. Hui Ma in [8] presents a combined method for services composition that combines genetic programming and random greedy search, but this method is very complicated and impractical. Stutzle in [9] presents a MMAS algorithm that the main idea is to limit pheromones on the path and to update them slightly to overcome depression. Nakamura in [10] mixed dynamic programming and ACO algorithm to solve Web services

Manuscript received August 5, 2026
Manuscript revised August 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.8.12

composition problem. The difference is that only sequential combined trips have been considered.

ZhengDe Zhao in [11] has studied on using the merge of GA and ACO algorithms in Web services composition. In this method, their approved algorithm uses the merge of GA and ACO algorithms for solving the composition time complexity problem. In the first step, the global optimal path can be found by GA and in the next step, the overall optimal solution is obtained by ACO implementation. The convergence speed of this hybrid algorithm reduced and the time of services composition increased. It's time complexity is better than traditional algorithms. However, its implementation time is not acceptable for growing number of services. Finally, this algorithm is not effective in terms of convergence speed and optimality.

3. Web Services Composition basic Definitions

Service-oriented architecture is a style of architecture that supports the services loosely coupling for the sake of the flexibility and interoperability of business, which is independent of technology and composed of the combination of a set of services based on business. These services provide flexibility and dynamic configuration for processes. Services composition is one of the five steps that has been introduced in the service-oriented architecture modeling. These steps are as following: service discovery, service description, service objectifying, services composition and service implementation. In this paper, we focused only on services composition step and its available methods. Services composition is the progress of a composite service development.

To compare services quantitatively, the QoS function makes the quality quantitative. Meeting any need for the user has a privilege by which the overall QoS is evaluated and used in comparisons. Eq. 1 shows how to calculate QoS [12].

(1) [12] $$QoS(s) = \sum w_i * Score(s)$$

In Eq. 1, the user gives score w_i to meet the i-th need and the s service can meet this need with the $Score_i$ quality. When we are looking for implementing a composite service to perform a composite action, initially we plan its implementation program, and then find the considered service for every operation. There may exist multiple services for any action on the system space, which we call them the candidate services. Service implementation on schedule, causes the implementation of the composed service.

The composite service: In the system space, many candidate services are provided by different suppliers for every action, so that selecting the appropriate service among them by the user can be a difficult and almost impossible act [13, 14], so by the automatic service composition, a computer system can choose the appropriate service in the shortest time with the highest accuracy in the midst of services for each action.

For every action in the implementation program, we choose a service that its function feature demonstrated its ability to perform the expected action and benefit of an interface that fits with nearby services. The extent of service-oriented systems and the existence of multiple suppliers lead to creating associate services with different interfaces and we should consider that they match together during the composition operation.

When creating services, we select a state among various states that in addition to the adaptation of function and interface features, as well as the acquired service, has the best quality (maximum QoS) according to the user criteria and will finally have an optimized composite service.

We can consider the optimization both local and global. We select a service with the highest QoS of the total collaboration services for each action in the local optimum. This method acts as greedy, so has a high speed, but does not guarantee the optimization of the entire service. In the global optimality, we look for services for each action that by putting them together, the QoS of the composite service finally have the highest value. It should be noted that the entire QoS of the composite service is the sum of QoS features of constituent services of the composite service.

Choosing the best case can be done at design time and all required services can be collected together (static) or occurs at run time that in this case, we find the appropriate services of each action and then we go to the next action (dynamic). Due to changes in the quality and number of services, dynamic choices are more realistic and produce more practical results [15].

If running a Web service implementation logic was involved with calling another Web services, it is necessary to combine the functions of several Web services and the final result is called composite service. Services composition is the process of creating a composite service. Services composition can be done with the composition of atomic services and composite services in which case the topic of Hierarchical services composition arises. In the composite service case, the answer of customer requests has been implemented with several services. In the following, we describe a dynamic way to find the optimal composition in order to build composite services.

- Genetic Algorithm

Genetic algorithm was presented in 1975 by John Holland. This algorithm is a random search technique that is based on the human evolution theory. In fact, genetic algorithms are stochastic search techniques based on natural mechanisms and composition and mutation genetic rules. Genetic algorithms begin with an initial set of random solutions called initial population. Each member of the population is called a chromosome that shows an answer to the problem.

The chromosomes evolve during frequent periods and each period is called a generation. In every generation the population changes. A new generation is created that is more powerful than previous population in terms of being close to the optimal solution. The evolution of chromosomes will be done in two ways [16, 17].

At first step, a number of chromosomes are randomly selected from the population and are combined together to create new elements. Different and several methods enumerated for genetic algorithms that each of which has its own advantages and uses. Mutation is the second step that one or more chromosomes are randomly selected in each iteration in this step and also one of the genes in the chromosome is randomly selected and will vary according to the specific mechanism, as a result, new chromosomes are generated. The mutation step is very short and the mutation rate is usually a small value. In the final step, the new generation is selected from the best members of the expanded population of the number of the initial population and is considered as a new generation. There are various methods for the selection step such as roulette wheel selection, complete random sampling, local and shear selection. Genetic algorithms differ in the combination, mutation and selection methods or the order of using these stages.

- Ant colony optimization algorithm

In 1996 Dorigo and Gambardella created reforms in the ant algorithm and presented the ant colony optimization algorithm [18]. This algorithm is different from previous algorithm in three factors, these differences are as follows:

1. The position transfer rule that controls the effect of new nodes and previous nodes on the algorithm.
2. The general update rule that is used only in the nodes that belong to the better ant path.
3. When the ants create an answer, the local update rule is used.

The ACO algorithm method is an evolutionary way to search in the answer space. So in this way the purpose is to find a path that has less distance to the food. This shorter path for the ants is determined based on the accumulation of pheromone on the path. In Web services composition problem, there are several paths based on the number of services on the previous or next level in each service level. Of course, there is not any path in the first or last level. When the ant colony algorithm begins to search, at any level examines how much the cost between any compositions of two services is, this means that we use the Eq. 2.

$$F(g)=\frac{W1*Cost(g)+W2*ResponseTime(g)}{W3*Availability(g)+W4*Reliability(g)}+W5*D(g) \quad \textbf{(2)}[19]$$
$$cli(g)\le 0, i=1,...,n$$

The major difference between ACO and GA algorithms during search is that the ACO algorithm only searches randomly at start time and by continuing the search, tries to find the optimal path based on the smart pheromones redundancy method.

When we look good on Eq. 4, we can see that with regard to the Eq. 7 which specifies how to initialize the pheromone, the ACO algorithm can recognize the importance of services composition by the evaporation rate based on the passage of time. This method, recognition by the amount of pheromone, is a local search method at each service level for identifying those services that are mostly combined together. Recognizing the services that are mostly combined together, causes the search in two services composition at each level to be done more smartly and the random search should be avoided. It also causes the results to be more accurate, because the amount of pheromone is obtained based on a function of time and the pheromones evaporation occurs during the time. The stronger compositions are specified in the services composition, gradually. Since the main purpose of this paper is to recognize the optimal services composition and the important parameters of optimization are qualitative parameters; at each level, services composition should happen optimally from the perspective of the qualitative parameters. Services that have a better state in terms of Eq. 2 (i.e. have a lower value), at different times also could have a better value. But if the value of the composition also changes, the algorithm adapt itself to new conditions and finds the best new services composition and thus, for each level, the best services composition are identified and used. The second part of Eq. 2 is not applicable in the ACO algorithm and is used to find the optimal services combination in the genetic equation.

When we are talking about optimizing the compositions, we discuss about the Eq. 2 and try to minimize this equation. That means, QoS is at its maximum state in the service-oriented architecture if F (g) is minimal. Because this function is the function of the services composition cost and it is optimal when it has a minimum value. But the second part of the equation that is $W5*D(g)$, in fact, is the value of solution violation. The reason for considering this violation is for guiding the algorithm to the problem feasible space. GA tries to minimize the cost function (Eq. 2) and when the solution is

violated, it's found out that produced solution is not a perfect solution. But the main point is how to find the best solution. The method of creating solutions in GA is random, that means, creating solutions is based on the use of solutions that are created and prepared randomly using GA operators (Mutation and Crossover), but there is no assurance to provide a better solution than the current solution and this recognition is only through minimizing the Eq. 2. But if there are more efficient solutions which are not found by genetic algorithm by using random operators, it can be stated that the algorithm has fallen into the local optimum. In this case, if we want the genetic algorithm to find a better solution, it will be almost impossible, unless in one of the iterations of the algorithm an answer is produced by the mutation operator which is better than the previous answers and this means that the algorithm browses a part of the search space that is feasible, but until then the algorithm has not been in that space for searching. What we can conclude from this problem is that, if the search space is large, i.e. , if we have 20 levels to select services and there are 50 services at every level (i.e. 1000 different services) and we want to achieve the optimal services composition using an algorithm such as GA, there will be the risk of falling into local optimum by this algorithm.

The ACO algorithm that we used is the algorithm presented by Dorigo in 1996 [18]. In this algorithm at first, m ants that have memory created in the first phase, these ants are placed randomly on n nodes. There is an initial value of pheromones on each node. In the second phase, in order to create the initial answer, the following steps are performed in parallel.

a) The ant that located in the r node, uses the Eq. 3 to select the s service for its next move that is known as the state transition rule in the ACO algorithm. In Eq. 3, q is a random number between [0, 1] and q_0 is value between zero and one.

(3) - $$S = \begin{cases} argmax_{u \in J_k(r)} \{[\tau(r,u)].[\eta(r,u)]^{\beta}\} \Rightarrow if (q \leq q_0 \\ s \Rightarrow other \end{cases}$$

In this equation, τ (r, s) represents the amount of pheromone on the arc (r, s), η (r, s) represents the inverse distance δ (r, s) and J_k (r) is the remaining services set for crossing the k ant that is located on the r-th service. β is a parameter to determine the importance of the relationship between pheromones and distance. Also in the above equation, S is a random variable that follows the probability distribution that is showed in the Eq. 4.

(4) [20] $$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,s)].[\eta(r,s)]^{\beta}} \Rightarrow if\ (s \in J_k(r)) \\ 0 \Rightarrow other \end{cases}$$

In this equation, p_k (r, s) is the probability that the k-th ant selects the r service after the s service.

b) After the ants created their own paths, they returned to their initial service. At this step, the local update occurs and the pheromones are changed by using the Eq. 5.

(5) [21] $$\tau(r,s) \leftarrow (1-p).\tau(r,s) + p.\Delta\tau(r,s)$$

In this equation, ρ is the pheromone evaporation parameter and Δτ (r, s) is obtained from the equation 6.

(6) - $$\Delta\tau(r,s) \begin{cases} (L_{gb})^{-1} \Rightarrow if (r,s) \in global\_best\_tou \\ 0 \Rightarrow other \end{cases}$$

Which L_{gb} is the length of the best path that is found in the current iteration of the algorithm.

c) Finally, if the condition have been established, the algorithm stops, otherwise the above steps will be re peated.

4. Research methodology

4.1 The objectives

As mentioned in the previous section, the services composition arises as a basic requirement for further organizational goals in response to customers' needs with the consideration of new requirement to meet the introduced needs and benefit from multiple services simultaneously. We can mention candidates with equal operational characteristics and different quality characteristics such as response time, cost, accessibility, reliability, security and so on.

Each services composition has different features that are combined in different scenarios to respond to the needs of users. For this reason, the consideration of services composition is under consideration in terms of the scenario and the composition optimality for the scenarios. Hence organizations as well as Web service providers, need to formulate these compositions optimally in accordance with different scenarios to reduce the computing costs and time and increase the quality of service provided to users. This research tries to provide a new algorithm, in order the optimal services composition to be performed optimally, so that services can be provided with a high quality based on services composition parameters as well as pending scenarios.

The optimal services composition problem is more important because if the process is composed of m steps and each work includes n candidate services on average, so

there are n^m different ways to combine Web services at the end and this issue turns the services composition to a NP-Hard problem. Many different methods have been proposed based on various algorithms for solving the problem of the optimal Web services composition, the Web services composition optimality problem is discussed more than any issue in the proposed algorithms; Because in different search algorithms such as GA, also the random search method and the search optimality criteria is based on formulating the problem as the cost function. For this reason, and due to the evolutionary approach of genetic algorithm that is based on randomly creating composition responses, it can be noted that the algorithm may be stuck in a local optimum. Hence, due to random search or define constraints with defect or inconsistency, the problem falling into the local optimum that really is not optimal or is a part of the infeasible response, or is an answer that can be just as a response to find the best answer, but is not the best answer, this issue causes the ways in which search methods that are not only based on random search be more important. In this article, the optimal services composition problem is presented with different scenarios by the hybrid algorithm of ACO and GA algorithms (HACOGA). The results of this article indicates that HACOGA algorithm is appropriate for being used in the field of the optimal Web services composition.

4.2 Modeling Assumptions

The Web Services composition problem consists of n services that there could be one path between its any two services. Each of these paths has a certain distance or cost. The Web services composition will start its own path from one service and then travel all services and passes each service only once, and finally return to the origin service. In this issue, the goal is to find a sequence of services that Web services composition pass them, so that the total traveled distance (travels cost) is minimized by the seller.

It is easy to understand the Web services composition problem, but if the problem size enlarges, solving it gets hard and in large-scale is nearly impossible. If we show the distance between i and j nodes with d_{ij} and if $d_{ij} = d_{ji}$, then Web services composition type is symmetric, otherwise it is asymmetrical Web services composition [22, 23].

4.2.1 Finding the optimum composition

In this article, we present a method in which the services available in the system space are considered as nodes of the graph. We form the edges of this graph to correspond to the actions. After making a graph, we find the best way with a search among existing paths and introduce the set of services that are located on that path as the best service composition. In the following, we will consider how to make graphs and searching in it [22, 23].

4.2.2 The QoS model

The first step to achieve the optimal Web services composition is to create an appropriate model for describing the qualitative characteristics of them. This model should be accepted by customer and service provider. Qualitative characteristics of Web services are divided into two general categories: positive and negative features. The most important features include response time, cost, reliability and accessibility. Another item that must be specified in the QoS model is how to calculate the total amount of qualitative characteristics of composite Web service that is done using Table 1 [24].

Table 1: The formula for calculating the qualitative characteristics of the composite Web services in parallel scenario [24]

Feature	*Response time*	*Cost*	*Accessibility*	*Reliability*
Calculation Formula	Rt(x1) + max{ Rt(x2) … Rt(xn)}	P(x1) +…..+ P(xn)	A(x1) *……* A(xn)	R(x1) *…...* R(xn)

w is a parameter to express the importance of each parameter in services composition. In this equation, the ratio of cost in response time on the amount of accessibility and reliability is displayed that is well represented the service composition state, because all the main parameters that affect the services composition are included in this equation.

4.2.3 Web services composition optimization

All states that are available for service composition are stored in a matrix in multi levels with respect to the ranking. At the same time the ACO algorithm and genetic algorithm start searching for composition states in the matrix as follow to find the optimal services composition.

In the Web services composition problem, finding the optimal Web services composition is same as finding the best path on the ACO algorithm, with the difference that in here the distance have no meaning and we use the Eq. 2 to check the QoS. When the cost of the path is calculating by the Eq. 2, the output number shows the state of composition. As mentioned earlier, when the value of F(g) is lowest, it means that we have reached the best solution for composition in two service levels. The mathematical concept of this problem is displayed in Eq. 7.

$$Optimize_{Route_{i,j}} \prod_{Route_i}^{Route_j} \begin{Bmatrix} Minimum(F(Route_{i,j})) \\ Maximize(\tau(\mathrm{i},\mathrm{j})) \end{Bmatrix} \quad (7)$$

The Eq. 7 describes that in the services composition of two levels, there is the best composition when the calculated cost according to Eq. 2 have the lowest value and therefore, the amount of pheromones in this composition has the maximum value. In fact, the ACO algorithm has a table for each service level that finds out where the value of the cost function 2 is minimum after calculating the Eq. 2 for the number of services at any level. The use of evolutionary algorithms is the reason for using a number of available services and because of we do not want to examine all possible compositions. After a series of available services compositions, the amount of pheromones should be the maximum and the algorithm should converge to that path.

4.2.4 The overall quality of composite service

Since the paths with massive edges have services with better quality, we choose the heaviest path among the running paths to find the best service with the highest quality. So we find the best path in the built graph to find the best composite service. Therefore, we do a search on the graph. Among different methods have ever expressed for quick search on the graph, the dynamic programming method could search for the best path in the graph with a good complexity [18].

4.3 HACOGA algorithm

We use ACO and GA characteristics in the hybrid algorithm that we will explain it in the following. In this algorithm, the initial answers of the ACO algorithm is used to mutate (to pass) the GA answers from the local optimum answers and the GA answers is used to enlarge the ants search domain. The hybrid algorithm has been developed only to improve the answer and therefore the problem-solving time is not been addressed. You can see the general schema of the Hybrid algorithm in Figure 1. As the figure suggests, this algorithm has three basic concepts that include hybrid algorithm, genetic algorithm and the answers exchange part that will be described later.

One of the main challenges in the field of research on the optimal Web services composition is the lack of data collection for investigation and research in this area. For this reason, the only way to analyze the optimal Web services composition to use modeling to create optimal compositions. We also created a model for analyzing the evolutionary algorithms at Matlab software which can create a composition scenario for each of parallel, concurrent, sequential and ring states. To build the model, we have considered 10 levels and each level has 100 services. That is a composition problem with 1000 services. The reason for using these services is to expand the space

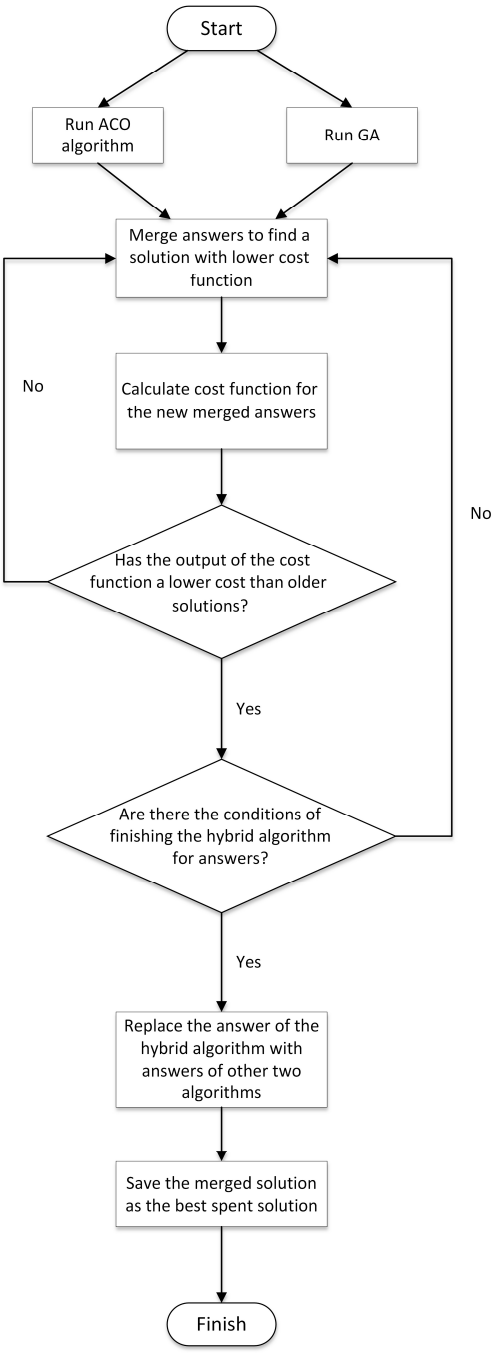

Figure 1: A view of the proposed hybrid algorithm (HACOGA)

of services composition and to specify the hybrid algorithms performance and better.

To ease the services composition analyzing task for all existing scenarios, services composition matrix was created according to the method of calculating the cost function and then the services composition problem was given to the evolutionary algorithms to search and find the

best services composition for different scenarios. Meanwhile, the parallel scenario provides the best answer which we can see that in the following. Both GA and HACOGA were done on the same terms, as 30 times original iteration and 20 times iteration to find random answers and the population size (pop) for GA as well as in same conditions for HACOGA, that is 30 times original iteration with 20 times sub-iteration in each iteration for GA, and 30 ants with 20 times search run for each ant to find optimal answers for different composition scenarios. Table 2 shows the GA parameter settings.

In the utility function (the first part of Eq. 2), cost and time are aggregated and divided on the availability and reliability, that this equation represents cost for services composition. Hence, the lower the value of the output of this equation, the better composition takes place between services and the search algorithms such as GA. HACOGA search the space of optimal answers for this equation. Through the output of this equation, the amount of optimality and quality of combined services is realized.

Table 2 : GA parameter settings

Features	***Answers***
Mutation rate	0.02
Crossover rate	0.7
Selection operator type	Roulette-Wheel selection
Mutation operator type	Random mutation, natural selection and Tournament selection
Crossover operator type	Uniform crossover One-point crossover
Number of considered optimal compositions	100 from 1000 exist states
Number of levels for service selection	10 levels

For the HACOGA algorithm have been set and ready according to Table 3 parameters.

Table 3 : HACOGA parameter settings

Features	***Answers***
Mutation rate	0.02
Crossover rate	0.7
Selection operator type	Roulette-Wheel selection
Mutation operator type	Random mutation, Natural selection and Tournament selection
Crossover operator type	Uniform crossover One-point crossover
Number of the ants that to find 100 optimal compositions from thousands of the existing compositions are available at any execution of the ACO algorithm	m=30
Number of the optimal services compositions that the ACO algorithm should find them.	n=100
A random number in [0,1]	q
A random number in [0,1]	q_0
Number of levels for nodes in the optimal services selection	S=Floor

4.4 Evaluation of of the proposed method

The Pseudo codes 1, 2 and 3 respectively are abstracts of GA, ACO and HACOGA. It is clear from the flowchart of figure 1 that the ACO algorithm is not as a separate algorithm, but as an active algorithm in both GA and HACOGA algorithms. In the Pseudo codes 1, 2 and 3, MaxIt represents the maximum numbers of iterations, Popc represents the selected population for the crossover operation, nc represents the number of parents, Popm represents the selected population for the mutation operation and nm shows the number of mutations. Empty individual is an empty structure to create solution. Costs is the array of costs of every solution. Pop is the total population matrix as well as NFE is the number of assessments performed by the fitness function. The performance of the NFE in this method, is the same as pheromone performance in the ACO algorithm.

```
GA
   Problem Definition & Initialize
   For it = 1 : MaxIt          //main loop for GA
     Execute Crossover operation;
     Execute Mutation operation;
     Pop = [pop   Popc   Popm];
     Costs = [Pop.Cost];
     SortOrder Pop and Costs;
     Trauncate extra members;
     Update Best Solution & NFE
   End
   Extract Results
End
```

Pseudo code 1: Genetic algorithm (GA)

In the following, we analyze and compare the results of the optimal Web services composition using genetic evolutionary and HACOGA algorithms.

```
ACO
   Problem Definition
   For it = MaxIt          //main loop for ACO
      Place ant on random positions
      For i = nNode
        ant(k).P = tau(i,j) ^ alpha * etha(I,j) ^ beta;
        //compute probability of moving from node i to
      node j
        Normalize probabilities
```

```
        Add nodes one-by-one
      End
    End
    For k = 1 : nAnt
      ant(k).L = Tour_Cost(ant(k.Tour))
      Move to the next node
      tau(i,j) = tau(i,j) + 1/ant(k).L;
      tau(j,i) = tau(j,j);
    End
    tau = (1 – rho) * tau;
    Select best tour;
End
```

Pseudo code 2: Ant Colony Optimization (ACO) algorithm

5. Conclusion

In a comparison of the results obtained in GA, MGAACA and HACOGA, as you can see in Figure 2, the performance of these studied algorithms clearly indicates that the performance of HACOGA is better than the other algorithms and this is due to parallel implementation of GA and ACO. At first GA algorithm was implemented and then ACO algorithm is implemented in the MGAACA, but in HACOGA algorithm, GA and ACO algorithms run in parallel, which reduces the running time. The results of HACOGA and MGAACA algorithms are very close. In Figure 3 you can see different run times in above algorithms. Since the HACOGA algorithm basis is parallel, its execution time is less than other algorithms.

```
Hybrid Algorithm
  Problem Definition
  For CostN = 1 : nCost
      Initialize and Evaluate initial solution
      For it = 1 : MaxIt
          For subit = 1 : MaxSubIt
              Create and Evaluate new solution
              If newsol.cost <= Sol.cost
                  Sol = newsol;
              Else
                  Find new solution by heating new
                  solution
              End
          End
          Update best solution and save best cost;
          Update NFE;
          Display iteration information;
          T = alpha * T;       //Update temperature
      End
      Extract results;
  End
  End
```

Pseudo code 3: HACOGA algorithm

By comparing the results in the optimal services composition using HACOGA algorithm we conclude that generating appropriate answers, is based on the cost function of the optimal services composition together, the search strategy of the hybrid algorithm is to use the mechanism of generating initial solutions and then neighbor solutions of the best solution. It means, first we begin searching in the entire search space, for example the services composition space and then we find the best solution that is the candidate solution and then the algorithm tries to examine the neighbor solutions of the candidate solution to see if there is any better solutions (solution at a lower cost) in the neighborhood of the candidate solution. If there is a better solution, it is replaced with the candidate solution.

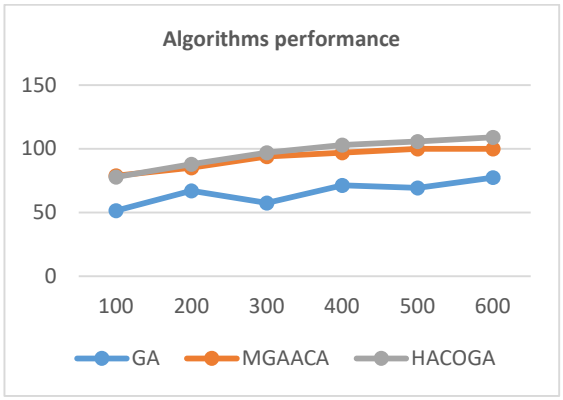

Figure 2: Performance compare of GA, MGAACA and HACOGA

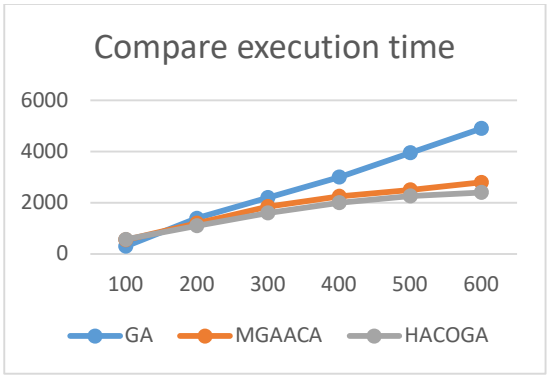

Figure 3 : Execution time compare of GA, MGAACA and HACOGA

It is clear from the results of this article that the hybrid algorithm can offer better compositions for different services composition scenarios and by this method we can obtain good results in the big composition problems.

References

[1] Beardwood, J., J.H. Halton, and J.M. Hammersley. *The shortest path through many points*. in *Mathematical Proceedings of the Cambridge Philosophical Society*. 1959. Cambridge Univ Press.

[2] 2. Clarke, G. and J.W. Wright, *Scheduling of vehicles from a central depot to a number of delivery points.* Operations research, 1964. **12**(4): p. 568-581.

[3] 3. Golden, B. and W. Stewart, *Empirical Analysis of Heuristics in the travelling Salesman Problem, E. Lawer, J. Lenstra, A. Rinnooy and D. Shmoys*. 1985, Wiley-Interscience, New York.

[4] 4. John, H., *H. Adaptation in Natural and Artificial Systems*. 1975, University of Michigan Press, Ann Arbor, Mich.

[5] 5. Lin, S. and B.W. Kernighan, *An effective heuristic algorithm for the traveling-salesman problem.* Operations research, 1973. **21**(2): p. 498-516.

[6] 6. Platzman, L.K. and J.J. Bartholdi III, *Spacefilling curves and the planar travelling salesman problem.* Journal of the ACM (JACM), 1989. **36**(4): p. 719-737.

[7] 7. Zhang, C.-W., S. Su, and J.-L. Chen, *Genetic algorithm on web services selection supporting QoS.* Jisuanji Xuebao(Chinese Journal of Computers), 2006. **29**(7): p. 1029-1037.

[8] 8. Ma, H., A. Wang, and M. Zhang, *A hybrid approach using genetic programming and greedy search for qos-aware web service composition*, in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII*. 2015, Springer. p. 180-205.

[9] 9. Stutzle, T. and H. Hoos. *MAX-MIN ant system and local search for the traveling salesman problem*. in *Evolutionary Computation, 1997., IEEE International Conference on*. 1997. IEEE.

[10] 10. Nakamura, L.H.V., et al. *A comparative analysis of algorithms for dynamic web services composition with quality of service*. in *Proceedings of the 19th Brazilian symposium on Multimedia and the web*. 2013. ACM.

[11] 11. Zhao, Z., X. Hong, and S. Wang, *A Web Service Composition Method Based on Merging Genetic Algorithm and Ant Colony Algorithm.* Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, 2015: p. 1007-1011.

[12] 12. Benatallah, B., Q.Z. Sheng, and M. Dumas, *The self-serv environment for web services composition.* Internet Computing, IEEE, 2003. **7**(1): p. 40-48.

[13] 13. Ambite, J.L., et al. *Getting from here to there: Interactive planning and agent execution for optimizing travel*. in *AAAI/IAAI*. 2002.

[14] 14. Arpinar, I.B., et al., *Ontology-driven web services composition platform.* Information Systems and E-Business Management, 2005. **3**(2): p. 175-199.

[15] 15. Fluegge, M., et al. *Challenges and techniques on the road to dynamically compose web services*. in *Proceedings of the 6th international conference on Web engineering*. 2006. ACM.

[16] 16. Dorigo, M. and L.M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem.* Evolutionary Computation, IEEE Transactions on, 1997. **1**(1): p. 53-66.

[17] 17. Dorigo, M. and L.M. Gambardella, *A study of some properties of Ant-Q*, in *Parallel Problem Solving from Nature—PPSN IV*. 1996, Springer. p. 656-665.

[18] 18. Dorigo, M., G. Di Caro, and L.M. Gambardella, *Ant algorithms for discrete optimization.* Artificial life, 1999. **5**(2): p. 137-172.

[19] 19. Papazoglou, M.P. and W.-J. Van Den Heuvel, *Service oriented architectures: approaches, technologies and research issues.* The VLDB journal, 2007. **16**(3): p. 389-415.

[20] 20. Pei, S., X. Shi, and D. Hu, *Research on the Particle-Ant Colony Algorithm in Web Services Composition Problem.* Journal of Applied Sciences, 2014. **14**(8): p. 805.

[21] 21. Wu, Q. and Q. Zhu, *Transactional and QoS-aware dynamic service composition based on ant colony optimization.* Future Generation Computer Systems, 2013. **29**(5): p. 1112-1119.

[22] 22. Burke, E. and G. Kendall. *Evaluation of two dimensional bin packing problem using the no fit polygon*. in *Proceedings of the 26th International Conference on Computers and Industrial Engineering, Melbourne, Australia*. 1999. Citeseer.

[23] 23. Gao, Y., et al. *Optimal web services selection using dynamic programming*. in *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on*. 2006. IEEE.

[24] 24. Huang, A.F., C.-W. Lan, and S.J. Yang, *An optimal QoS-based Web service selection scheme.* Information Sciences, 2009. **179**(19): p. 3309-3322.

Mahdi Bigham, computer software expert at University of Bojnord. He received his bachelor's degree in Computer Software Engineering from the Islamic Azad University of Mashhad in 2008. He finished also his master's program in software engineering at Islamic Azad University of Neyshabur in 2017. Since February 2010, he started working in the University of Bojnord as a computer expert. His research fields are Web Developing and Web Services.

AbbasAli Rezaee Assistant Professor at Payam-e Noor University in Computer Engineering. He studied computer engineering at the undergraduate degree in computer hardware at Shahid Beheshti University in 1998, a master's degree in architecture at Isfahan University in 2001 and a phd degree in hardware education at Islamic Azad University, Science and Research Branch of Tehran In 2013, it is over. Since the summer of 2016, he has been working as the president of the Payam-e Noor University of Chenaran. His research fields are Networks and WSN.