ARTICLE

QOS-AWARE CLOUD SERVICE SELECTION IN DYNAMIC PERVASIVE ENVIRONMENTS

Deepak Gupta[1*], Rajeshkannan R[2]

[1]*Student, School of Computer Science And Engineering VIT University, Vellore, INDIA*

[2]*Professor, School of Computer Science And Engineering, VIT University, Vellore, INDIA*

ISSN 0976-3104

ABSTRACT

Most of the web service selection methods don't account for conflicting objectives. These work by combining the objective function into a single objective function by scalarization. When the value of weights is, not given and fixed weights are used, this can lead to problems. In this paper, modified reinforcement learning algorithms are proposed which are used to derive Single policy and multiple policy solutions respectively and applies the concept of multi-agent Reinforcement learning and value iteration respectively in the original algorithm which uses fixed weights, our algorithm modifies the weights in order to achieve output. Our proposal algorithm dynamically updates the weight for the objectives functions in the value function vector to achieve accuracy especially for large no. of states (value function approximation). Our modified algorithm uses the combination of multi objectives reinforcement learning with value function approximation to achieve better results. We are formulating the problem as a multi objectives partially observable Markov Decision Model problem description. In pervasive environment, such technique can prove to be useful for selecting web services. Examples of such services are E- Governance applications.

KEY WORDS
cloud computing, web service selection,reinforcement learning

Received: 11 March 2017
Accepted: 20 May 2017
Published: 17 July 2016

***Corresponding Author**
Email:
deepak.gupta2013@vit.ac.in

INTRODUCTION

A large number of services are being offered on mobile today like E Governance application. A large amount of study has been undertaken to examine web service composition in dynamic pervasive environments I.e. which are aware of the environment and change their behavior accordingly (content aware).Web service composition engine takes in data which mostly consists of content sensitive data in the form of ontology of the domain.

Selecting web services out of a set of web services in order to create a composite web services is a complex task. When selecting web services from the set, certain QoS parameters have to be considered, this is known as QoS aware service composition.

QoS-aware web service selection has been widely researched in areas of service oriented Architecture (SOA)and service oriented computing (SOC)[1].All the existing approaches deals with evaluating objective functions of QoS parameters by combining them into a single objective function.

More-over, they give solutions which are not optimal. We propose two hybrid algorithms which formulate the web service selection as a multi-objective optimization problem .This involves simultaneous optimization of multiple objectives [1].

We have used 5 parameters which are mentioned in the problem description some of them are conflicting objectives like availability and response which are both positive when one has to be maximum and one has to be minimum in response to wrong inputs.

Availability= Q availability [29]

Response time = [100*(exp(-r^2)*exp(-r+0.1r))/(1+exp(-r+0.1r)] [29]

In single policy algorithm, weight is assigned by us whose purpose is to create a bias between preference of services and it is updated to maximize the LMS (Least mean square error) between them.

The problem of selecting web services has been studied in great detail. Various Techniques have been applied to selection of web services such as mixed integer Programming which formulates the problem as a multi-objective mixed integer programming problem [1]. In mixed integer linear programming, all variables do not belong to integer group but only a few.

Another formulation of the web service composition problem is the knapsack problem which is multiple choice and multi-dimensional (MMKP). In MMKP,

Weight= (w1,w2,w3,...,wd) Capacity = (c1,c2,c3,...,cd)[26] The aim is to maximize the ratio of sum of value of items in the knapsack to the sum of weights in such dimension of the knapsack. [26]. Another formulation uses an immune binary PSO (IDIPSO) is proposed.

IIOAB JOURNAL
ISSN: 0976-3104

IDIPSO uses immune concepts and methods with PSO to solve the web service composition problem. [5] The web service composition problem is formulated as multi-objective ant colony optimization and is solved using multi objective ant colony optimization. [6]

Petri net based algebra is also used to model web service composition process related work. [27]This algebra is expressive enough to capture the semantics of complex web service composition. [27] Another formulation of web service composition uses the LIAC algorithm to construct the service path [14]. Web service composition has applications in pervasive computing.

Related Work

Wang et al [1] proposed a method to select a web service from a set of web service using mixed integer programming. Confora et al [1] proposed a method to select a web service from a set of web service using GA (genetic algorithm). Kiu and Lin [1] proposed a method to select a web service from a set of web service by formulating it as a knapsack problem which is both multiple choice and multi- dimensional.

The problem of selecting a web service from a set of web service can be solved using non-linear techniques as proposed by Grossmann [3]. In (Ran 2003) the main idea is to incorporate the QoS awareness into the UDDI registries as to facilitate QoS aware service discovery.

IDIPSO uses immune concepts and methods with PSO to solve the web service composition problem .[5] The problem of selecting web service from a set of web service can be solved by using multi-objective ant colony optimization.[5] The problem of selecting web service from a set of web services can be solved using elitist MOEA ,Zitzler's and Thielem's SPFA [6], Knowles and corne's Pareto-archived PAES [7] and Rudolph elitist GA [8]. The problem of selecting a web service from a set of web service can be solved by using Petri nets as proposed by Hamadi and Benatallah [9].

The problem can also be solved using Maxmilien and Singh's [10] approach which is based on learning based on historical information but without explanation.[11] and PICO as proposed by kalasapur et al [11] which solves the problem using graph theory and the LIAC algorithm as proposed by Raman and Katz [12].

Various middleware has been proposed such as spider net by Gu et al to solve the above problem. [15] the problem of selecting web services from a set of web services is a pervasive involvement has been studied by Sonia, Nicolas, Valerie [13].Application of the above problem to business process has been proposed by F. Koshkina And M. Von Bruegel [16].In software agent oriented approach solve the above problem has been proposed by Za aria maonnar, sorayukocicedic and hamdii vahyavi [17].

Objective function for QoS parameters for web services

We are using Response time, availability, throughput, successability, reliability as our objectives.

Response Time- Response time is the overall time required to complete a server request. [29]

Q (Response time) = [100*(exp(-r^2)*exp(-r+0.1r))/(1+exp(-r+0.1r)] [29]

Availability - It is a factor which measures whether the server is up after selection [29]

Throughput – It is referred as total number of completed transactions by a web service over a time period

Successability – It is defined as the extent to which web services yield successful results over request messages.

Reliability – It is ability of a web service to perform well over a given time span [29].

Most of the web service selection methods don't account for conflicting objectives. These work by combining the objective function into a single objective function by scalarization. Eg- when there are conflicting inputs to a web service i.e. parameters conflicts, the availability is high which is to be maximized and the response time is also high which is to be minimized.

Markov decision process

We formulate the problem as an MOMDP-WSC (Multi Objective Markov Decision Process Web Service Composition). This is applied to multi objective optimization in partially observable environment. MDPs has applications in optimization problems and reinforcement learning and dynamic programming. MDPs is a random process .An MDPs consists of transition diagram between states and a reward function R(s, s') after moving from one state to another. [28]

An MOMDP-WSC is defined as a 6-tuple MOMDP-WSC-[(s), Si, St, A, P, R]

S- Set of states agents can be in

ISSN: 0976-3104

Si- state from which agents starts
St – set of terminals
A-Set of action (web services) that can be executed by the agent (web services composition engine)
P is the probability of going from one state to another by choosing an action.

Output of MOMDP-WSC is an optimal decision policy

π: S → A

Each policy of MOMDP-WSC can define a single workflow, and therefore, the task of our service composition model is to identify the set of Pareto optimal policies that gives the best trade-offs among multiple QoS criteria. [1]

Multi objective reinforcement learning for service composition [1]

The goal of MORL is to require the set of Pareto- Optimal Policies in MOMDP model [1]

Moreover, [V (s) = (V_1^{π} (s), V_2^{π}(s), ..., V_m^{π} (s)]

Is the value vector of scale s under policy π and it is defined by

E_{π} - Mean
S_{π} - State
r_{π} - Reward vector
r – Discount rate parameter

We also define the Q – learning vector

Q_{π} (s,a) = E_{π} {$\sum^{inf}_{k}$=0gamma$^{k}r_{t+k+1|}s_t$=s, a_t=a} [1]

The MORL consists of two classes of algorithms, single policy algorithm and multiple policy algorithm. Single policy finds the single state to action mapping of an MDP formulation using multi agent reinforcement learning. The multiple policy algorithms produce many solutions in form of policies of the MDP formulation. [1]

Value function approximation [2]

Value Function Approximation deals with approximation of value function using another function, we approximate the weight of value function using value function approximation which helps us to obtain an accurate solution for large number of actions.

The aim is to estimate the value function with function approximation

$\hat{v}(s,w) \approx v_{\pi}(s)$

or $\hat{q}(s,a,w) \approx q_{\pi}(s,a)$

And to generalise from seen states to unseen states [2]

Update the weights of the value vectors dynamically (with the following formula)

$V(S, W) = X(S)^{T}W = E\sum_{nJ=1}X_J(S)w_J$[2]

Objective function for W:-
$J(w) = E_{\pi}[(v_{\pi}(s) - x(S)^{T}w)^2]$ [2]
update rule is particularly simple-
DERIVATIVE
$\hat{V}(s,w) = x(S)$
$\Delta w = \alpha(v_{\pi}(S) - \hat{v}(s,w))(x(S))$ [2]
update = step size x prediction error x feature value.
We apply the above value approximation to the single policy and multiple policy algorithm.

Architecture diagram

IIOAB JOURNAL
ISSN: 0976-3104

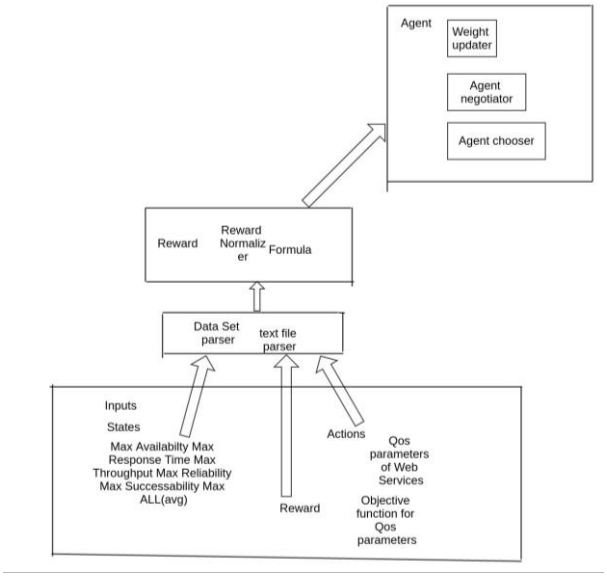

Fig. 1: Modified single policy reinforcement learning algorithm.

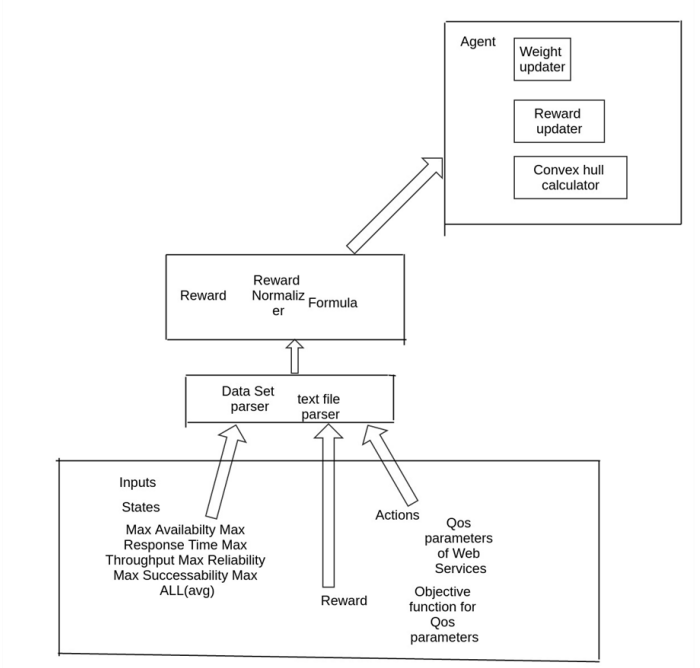

Fig. 2: Modified multiple policy reinforcement learning algorithm.

METHODOLOGY

Modified single policy multi objective service composition

In the first algorithm, each QoS objective is implemented as a separate Q learning agent [1] at every state & each agents selects its own such service (agent's performance) that optimizes its own QoS objectives. The relative gain is calculated obtained by agents in choosing its own web service to the reward obtained by choosing the service recommended by other agent. The agents negotiates among themselves to reach on optimum decision policy.

Our modified algorithm states that we assign a performance to the objective function by multiplying with its weights. The optimum decision policy is calculated by choosing its action with maximum state action value (reward). The weights are updated dynamically by using value function approximation in order to minimize its least square error between its optimum decision value function and the current value function (current

IIOAB JOURNAL
ISSN: 0976-3104

iteration) which helps to iteratively achieve the optimal decision policy in a shorter time and provide approximately accurate to a large no. of actions.(Fig 1)

First, we randomly choose an agent which selects an optimal action based on epsilon –greedy policy. The remaining agents optimizes the remaining objective function assigned to them by calculating the relative reward gained by choosing their choice to optimize the objective function assigned to them or following the choice of the randomly chosen agent (single policy algorithm).

Modified single policy algorithm

The algorithm consists of the following steps

```
1. Agent negotiation (discussed above)
2. Weight updation
Function single_policy_algorithm() [1]
{
Choose a starting states from set of states
k=random integer between 1 to the no of actions
Re_k=0//maximum reward so far
Re_i=0 // maximum reward at each iteration
deltaW=0 //change in weight
gamma=0.01 //discount factor for reinforcement learning equation
epsilon=0.01 //error limit
ak=index of service with maximum  re ward of agent k ‘s objective function
repeat
for all agents I except k do
{
t=maximum of agent_i ‘s objective function
 t2=W[agent_i]*(max-R_[s,ak,agent_i])   #agent   negotiation relative reward //obtained when agent_i choose his choice rather than following the choice of randomly //chosen agent
deltaW= alpha * (Re_k-t2)*(t2/W[agent_i]) //change in weight
W[agent_i]=W[agent_i]+(deltaW) #weight updation
if t2 > max3: //choosing max Re_i
{
   max3=t2
Re_i=t2
}
ifRe_i>Re_k : #in general case we keep Re_i for all agents except k
{ // if obtained  reward is greater than maximum  reward
            Re_k=Re_i
 k=agent_i #argmax(R_agent_i([s,a,agent_i]))
}
if delta < abs(Re_i-t):          //convergence condition
delta= abs(Re_i-t)
if (delta < epsilon * (1 - gamma) / (gamma)) :
break
maxiter=maxiter+1
print ("Iteration No",maxiter)
if maxiter == 15:
break
}
If maxiter == 15:
break
}
```

Multiple policy multi-objective service composition

The multiple policy algorithm uses an exact method to solve the MDP formulation of the problem known as value iteration is use to solve the problem, this is combined with dynamic programming which updates the state value action vector by calculating the convex hull in 2 dimensions.

In multiple policy algorithm, we use value iteration and dynamic programming to calculate the convex hull in 2 dimensions. We then choose the action which is optimal i.e., which has maximum value when we dot product of the reward with weight .we update the state action value function according to value iteration and weights according to value approximation. We have used availability and response time as parameters as they are conflicting objective because when we supply wrong inputs to web service the availability is positive and response time is negative.(Fig 2)

We compute the convex hull in 2 dimensions. There two objectives using the following equations→

1. Convex Hull(Union(Q(s’,a’))={q belongs to Q(s’,a’) | max {w.q}}
2. Q(s,a)=(1-alpha)Q(s,a) + alpha[r(s,a) + gamma* (Convex Hull(Union(Q(s’,a’)))]

IIOAB JOURNAL
ISSN: 0976-3104

3. W=w+delta(w)

Multiple policy multi-objective service composition algorithm

The algorithm consists of the following steps

1. Convex hull calculation (discussed above)
2. Value iteration
3. Weight updation

```
Function multiple_policy_algorithm()  [2]
{
Initialize state action value function Q
epsilon=0.5 //error condition
max=0 //for convex hull
maxiter=0 //max iterations
Re_k=0 // maximum reward so far
total=asarray([0.5,0.5])  //weights (value function approximation)
for maxiter in range(0,T)
{
Initialize state variables s= random integer between 1 to the no of states
Calculate maximum reward from actions in next states
Max'= Convex Hull (Union(Q(s',a'))={q belongs to Q(s',a') | max {w.q}}
Update q using the following formula:
Q(s,a)=(1-alpha)Q(s,a) + alpha[r(s,a) + gamma* (Convex Hull(Union(Q(s',a'))))]
        Deltaw=w.dot(max(Union(Q[s',a'])))
        W=w+deltaw
        S=s'
`       if(s==send)
        {
        Break;}}}
```

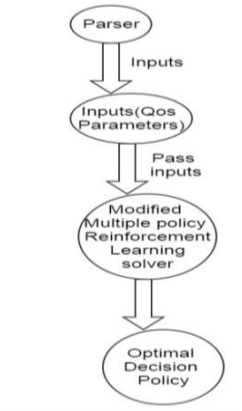

Fig. 3: Flow diagrams of multiple policy reinforcement learning algorithm.

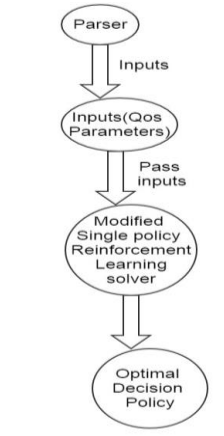

Fig. 4: Flow diagrams of single policy reinforcement learning algorithm.

IIOAB JOURNAL
ISSN: 0976-3104

Data Set (Table 1) is parsed using Data Set Parser. Inputs in form of vectors are passed to the modified single policy reinforcement learning solver and modified multiple policy reinforcement learning solver. From both the algorithms, we get the optimal decision policy (Fig. 3 and 4)

EXPERIMENT RESULTS AND ANALYSIS

We have considered the following dataset

Table 1: Sample QWS dataset with QOS parameters

Name of web service	Availability	Response Time	Throughput	Successability	Reliability
MAPP Matching	302.75	89	7	90	73
MSSOAP INTEROP	482	85	16	95	73
STRIKE IRON	3321	89	1.39	96	73
HOLIDAY WEB SERVICE	126	98	12	100	67
GALEX	107	87	1.89	95	73
INTEROP	107	80	1.7	81	67
EMBOSS4	255	98	1.29	99	67
EMBL-EBI	136	76	2.79	76	60
SERVICEOBJECTS.COM	102	91	15.300	97	67
LEAD TOOLS	93	96	13.5	99	67

Single policy algorithm

Algorithm trace out

Agent negotiation and weight updation and accumulated reward

1. Agent negotiation relative reward obtained when agent_i choose his choice rather than following the choice of randomly chosen agent
2. Reward updation
3. Weight updation
4. Choosing action with max reward

Index of web services with Maximum value of objective function: 2
delta W -3.50398068269
Re_i: 3.94629541198
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Index of web services with Maximum value of objective function: 2
delta W 0.0
Re_i: 0.0
Re_k: 3.94629541198
Iteration No 1
Accumulated Reward: 3.94629541198
and so on till convergence condition or till max iterations.

Multiple policy algorithm

The algorithm consists of the following steps:

1. Epsilon-greedy strategy:
2. Convex hull calculation
3. Value iteration
4. Weight updation

IIOAB JOURNAL
ISSN: 0976-3104

Algorithm trace out

Modified multiple policy algorithm with value function approximation
Maximum value of objective function: [[0.010160994923391649, 12.239028778368278, 0.72789115734524901, 1.0, 0.53846153846153844]]
Index of web services with Maximum value of objective function: [3]
Accumulated Reward: 162.072528122

Iteration No 1

Maximum value of objective function: [[0.010160994923391649, 12.239028778368278, 0.72789115734524901, 1.0, 0.53846153846153844], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0]]
Index of web services with Maximum value of objective function: [3, 5, 7]
Accumulated Reward: 541.358115854

Iteration No 2

Maximum value of objective function: [[0.010160994923391649, 12.239028778368278, 0.72789115734524901, 1.0, 0.53846153846153844], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0]]
Index of web services with Maximum value of objective function: [3, 5, 7, 5, 7]
Accumulated Reward: 1200.85210952

Iteration No 3

Maximum value of objective function: [[0.010160994923391649, 12.239028778368278, 0.72789115734524901, 1.0, 0.53846153846153844], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0]]
Index of web services with Maximum value of objective function: [3, 5, 7, 5, 7, 5, 7]
Accumulated Reward: 1807.09099934

Iteration No 4

Maximum value of objective function: [[0.010160994923391649, 12.239028778368278, 0.72789115734524901, 1.0, 0.53846153846153844], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.004398966982931054, 38.41560588468586, 0.02721089075305546, 0.20833333333333334, 0.5384615384615384], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0], [0.013426147004615304, 50.0, 0.10204081599553161, 0.0, 0.0]]

Index of web services with Maximum value of objective function: [3, 5, 7, 5, 7, 5, 7, 7]

Accumulated Reward: 2468.55487898

Parameter	Meaning	Value
Alpha	learning rate	0.1
Gamma	discount factor	0.01

The experiments results were to examine the ability of the hybrid algorithm. results are as follows:

1-Singly policy reinforcement learning algorithm
2-improved algorithm

IIOAB JOURNAL
ISSN: 0976-3104

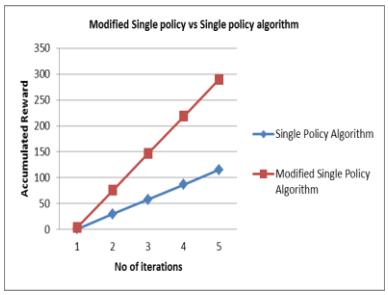

Fig. 5: Modified single policy vs single policy algorithm.

As you can see in the above chart, the accumulated reward for modified single policy algorithm is greater than the accumulated reward for original algorithm. This is true for each iteration, thus the quality of solutions obtained by Modified single policy algorithm is greater than single policy algorithm.(Fig 5)

Thus the modified algorithm yield better results than original algorithm

1-Multiple policy reinforcement learning algorithm
2-improved algorithm

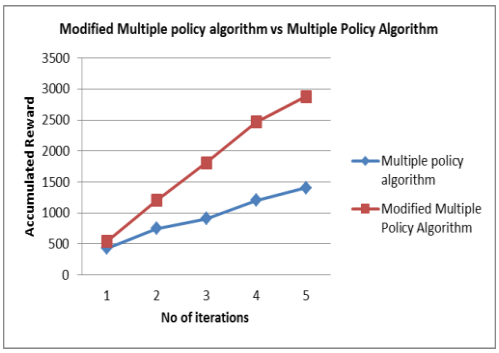

Fig. 6: Multiple policy algorithm vs modified multiple policy algorithms.

As you see in the above chart the accumulated reward for modified multiple policy algorithm is greater than the accumulated reward algorithm for original multiple policy algorithm. This is true for each iteration, thus the quality of solutions obtained by Modified multiple policy algorithm is greater than multiple policy algorithm.(Fig 6)

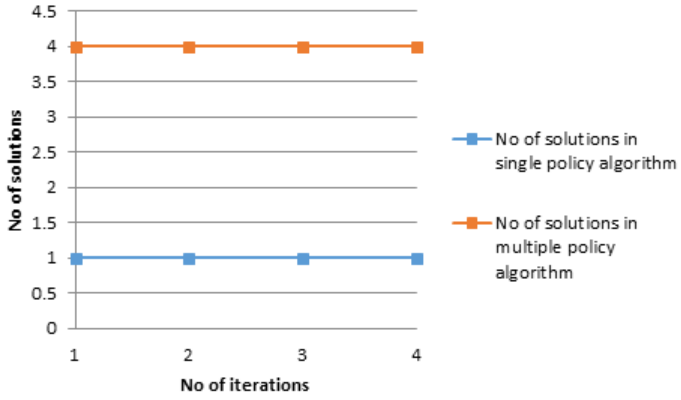

Fig. 7: No of solutions in single policy algorithm an multiple policy algorithms.

As you can see from the above chart, the no of solutions obtained in multiple policy algorithms is greater than no of solutions obtained in single policy algorithm. Thus, multiple policy algorithms is better than single policy algorithm. Thus, we can obtain multiple solutions to the same problem in multiple policy algorithms as compared to single policy algorithm.(Fig 7)

IIOAB JOURNAL
ISSN: 0976-3104

CONCLUSION

We propose modified reinforcement learning algorithm, the result have shown that the modified reinforcement learning algorithm is more effective than the original algorithm for selecting a web services in a pervasive environments. The future work is to further modify learning algorithm to achieve better results. The effectiveness of the proposed reinforcement learning algorithm is shown in the results.

The scenario addressed by single policy algorithm is that we have partially observable environment and we want to find the optimal state of action mapping of the MDP formulation of the problem .This is single in number in single policy algorithm .The scenario addressed by multiple policy algorithm is that we want to find multiple solutions to the MDP formulation of the problem. The future work involves optimization of the value function approximation technique to yield better results.

CONFLICT OF INTEREST
The authors have no competing conflicts of interest.

ACKNOWLEDGEMENTS
I am thankful to Prof. Rajeshkannan R. and VIT University for supporting me in this project.

FINANCIAL DISCLOSURE
There has been no financial investment in the project.

AUTHOR CONTRIBUTION
This research was done by Deepak Gupta under the guidance of Prof. R. Rajeshkannan.

REFERENCES

[1] Moustafa A, Zhang M. [2013] Multi-objective service composition using reinforcement learning. International Conference on Service-Oriented Computing 2013 Dec 2. Springer Berlin Heidelberg. 298-312
[2] Geist, Matthieu, Olivier Pietquin. [2010] A brief survey of parametric value function approximation. Rapport interne, Supélec.
[3] Grossmann I. [2002] Review of nonlinear mixed-integer and disjunctive programming techniques. Optimization and Engineering. 3(3):227–252.
[4] Sreenath RM, Singh MP. [2004] Agent-based service selection. In Web Semantics: Science, Service and Agents on the World Wide Web. 261-279.
[5] Parejo JA, Segura S, Fern´andez P, Ruiz-Cort´es A. [2014] QoS-aware 940 web services composition using GRASP with Path Relinking. Expert Systems with Applications, 41:4211–4223. URL: http://dx.doi.org/10.1016/j.eswa.2013.12.036. doi:10.1016/j.eswa.2013.12.036.
[6] Zitzler E. [1999] Evolutionary algorithms for multi objective optimization: Methods and applications. Doctoral thesis ETH NO. 13398, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany: Shaker Verlag
[7] Knowles J, Corne D. [1999] The Pareto archived evolution strategy: A new baseline algorithm for multi objective optimisation. Proceedings of the 1999 Congress on Evolutionary Computation, Piscatway: New Jersey: IEEE Service Center. 98–105.
[8] Rudolph G. [1999] Evolutionary search under partially ordered sets. Technical Report No. CI-67/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany
[9] Hamadi R, Benatallah B. [2003] A Petri net-based model for web service composition. ADC '03: Proceedings of the 14th Australasian database conference. Darling hurst, Australia, Australian Computer Society, Inc.191—200.
[10] Fu Y, Zhijiang D, Xudong H. [2006] Modeling, validating and automating composition of web services. ICWE '06: Proceedings of the 6th international conference on Web engineering, New York, NY, USA, 2006. ACM. 217—224.
[11] Maximilien EM, Singh MP. [2005] Multi agent System for Dynamic Web Services Selection. Proc. Int. Conf. Auton. Agents and Multi-Agent Syst.
[12] Kalasapur S, Kumar M, Shirazi BA. [2007] Dynamic service composition in pervasive computing. In Proceedings of the IEEE Transactions on Parallel and Distributed Systems (TPDS). 907–918.
[13] Hashemain SV. Mavaddat F. [2005] A graph-based approach to web services composition. In Proceedings of the 2005 IEEE/IPSJ International Symposium on Applications and the Internet (SAINT). 183–189.
[14] Raman B, Katz RH. [2003] An architecture for highly available wide-area service composition. Computer Communications. 26(15):1727–1740.
[15] Gu X, Nahrstedt K, Yu B. [2004] Spider Net: An integrated peer-to-peer service composition framework. In Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing. 110–119.
[16] Sonia Ben Mokhtar, Nikolaos Georgantas, Valerie Issarny. [2005] Ad hoc composition of user tasks in pervasive computing environments.In Proceedings of the 4th Workshop on Software Composition (SC 2005). Edinburgh, UK. LNCS 3628.
[17] Sonia Ben Mokhtar, Jinshan Liu, Nikolaos Georgantas, Valerie Issarny. QoS-aware dynamic service composition in ambient intelligence environments. Submitted for publication.
[18] Brogi A, Corfini S, Popescu R. [2005] Composition-oriented service discovery. In Proceedings of the 4th Workshop on Software Composition. Edinburgh. UK. LNCS 3628
[19] Harry Chen, FilipPerich, Timothy W, Finin, Anupam Joshi. [2004] Soupa: Standard ontology for ubiquitous and pervasive applications. In MobiQuitous. IEEE Computer Society .258–267.
[20] Dey A, Salber D, Abowd G. [2001] A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In Human-Computer Interaction. 16:97–166.
[21] Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer. [2003] Model-based verification of web service compositions. In IEEE International Conference on Automated Software Engineering.
[22] Koshkina M, van Breugel F. [2003] Verification of business processes for web services. Technical report, York University.
[23] Seng Wai Loke. [2004] Logic programming for context-aware pervasive computing: Language support, characterizing situations, and integration with the web. In Web Intelligence, IEEE Computer Society. 44–50.

ISSN: 0976-3104

[24] Zakaria Maamar, Soraya Kouadri, Hamdi Yahyaoui. [2004] A web services composition approach based on software agents and context. In SAC '04: Proceedings of the 2004 ACM symposium on Applied computing, New York, NY, USA. ACM Press. 1619–1623.
[25] Shalil Majithia, David W, Walker, Gray WA. [2004] A framework for automated service composition in service-oriented architecture. In 1st European Semantic Web Symposium.
[26] Bazgan C, Hugot H, Vanderpooten D. [2009] Solving efficiently the 0–1 multi-objective knapsack problem. Computers & Operations Research. 31:36(1):260-279.
[27] A Petri Net-based Model for Web Service Composition Rachid Hamadi Boualem Benatallah
[28] Howard RA. DYNAMIC PROGRAMMING AND MARKOV PROCESSES
[29] QoS AWARE FORMALIZED MODEL FOR SEMANTIC WEB SERVICE ELECTION
[30] Divya Sachan1, Saurabh Kumar Dixit1, and Sandeep Kumar1 Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, India.