Optimizing Large Search Space using DE Based Q-learning Algorithm

Jaya Sil,[a] Zenefa Rahaman[b]

[a]Indian Institute of Engineering Science and Technology, Shibpur, Howrah, 711103
India
js@cs.iiests.ac.in
[b]University of Tulsa, 800 South Tucker Drive, Tulsa, Oklahoma 74104
USA
zenefa-rahaman@utulsa.edu

Abstract. Finding global optimum solution in minimum time from large search space is challenging due to involvement of large no. of variables and their varied degree of participation in problem solving process. Complexity of a problem increases with the dimensionality, which must be learnt efficiently to improve performance of the method. Q-learning, a reinforcement learning algorithm is used widely to learn the environment dynamically. However, the conventional Q-learning is not fast and becomes inefficient while solving large scale problem. In the proposed approach by hybridizing Differential Evolution (DE) algorithm and Q-learning (QL) method (QL-DE) optimal partitioning of the search space is obtained involving multiple agents with an objective to achieve maximum classification accuracy. Performance of the proposed algorithm has been compared with state of the art optimization algorithms.

Keywords: Multi Agents, Differential Evolution algorithm, Q-Learning algorithm, Dimensionality Reduction

1 INTRODUCTION

Real world optimization problems become complex due to presence of multiple conflicting objectives, non-linearity and multi-modal non-convex search space. Moreover, search space often prevents convergence at global optimum in a reasonable time when the problem may get stuck at local optimum. The existing stochastic search methods like evolutionary algorithms (EA) as found in Ref. 9 and 18 are able to handle the complexities and combined with existing local search methods to achieve global optimal solution. However, large search space optimization problem needs devising efficient learning algorithm to handle dimensionality of the problem.Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence. Reinforcement learningas found in Ref. 5 is goal directed learning where an agent interacts with an unfamiliar, dynamic and stochastic environment. However, the main drawback of reinforcement learning is that it learns nothing from an episode until it is over. So the learning procedure is very slow and impractical for large space applications.

Distributed genetic algorithm based behavior learning was proposed as found in Ref. 7, where they have used internal reinforcement signal for learning, generated by fuzzy inference. InRef. 7, dynamic recurrent neural networks are used as action generation module where effectiveness of the learning algorithm is verified to cooperative research experiment. Three techniques are proposed in Ref. 2 to improve the performance of the algorithms for large scale continuous global function optimization namely, random grouping, adaptive weighting and self-adaptation of the sub-component sizes in Cooperative Co-evolution approach. However, the algorithm does not give guarantee of optimum results for problem evolving large search space. A Genetic algorithm based reinforcement learning algorithm is developed as found in Ref. 19 to solve large space application problem. This algorithm suffers from slow convergence property of genetic algorithm. A new searching method, CLARANS for clustering large scale applications based upon randomized search using DE is proposed as in Ref.11. A mobile DE agent has been created along the tangent curve to perform DE operation and calculate the cost difference of sub-population in different clusters. But it does not give any idea whether this algorithm is able to solve overlapping sub problems.

In this paper, the large search space has been tackled by dividing the problem into sub-problems among the agents where the overlapping space is learnt through co-ordination using the proposed DE

based QL algorithm. To learn the search space dynamically, QL algorithm has been hybridized with DE algorithm in order to achieve maximum accuracy in minimum time. The difficulty of applying QL algorithm in problems involving large action space has been handled by imposing regulated randomness in the learning method. The mutation operator of DE algorithm resembles the learning of Q value that introduces diversity in the population and fast convergence property of DE algorithm.

2 BACKGROUND OF THE WORK

2.1 Differential Evolution Algorithm

Differential Evolution algorithm (DE) is a Stochastic, heuristic, population-based optimization algorithmas in Ref. 16, developed to optimize real valued functions. The DE algorithm maintains fixed size of population from which the potential solutions are selected using a fitness function. The steps of DE algorithm are described below.

Initialization of the parameter vectors: DE searches global optimum point in a D-dimensional real parameter space R^D. It begins with a randomly initiated population of size NP with D dimensional real-valued parameter vectors. The following notation is given for representing *i*-th vector of the population at generation G:

$$X_{iG} = [x_{1iG}, x_{2iG}, x_{3iG}, \ldots . x_{DiG}]$$

Mutation: DE generates new individuals or Donor vectors in the population by adding weighted difference between two population vectors to a third vector, selected randomly. The Donor Vector is generated using equation (1) and described in Fig. 1.

$$V_{iG+1} = X_{ri1G} + F * (X_{ri2G} - X_{ri3G}) \qquad (1)$$

V_{iG+1} denote the Donor vector in *i*-th population at generation G+1. *F* is the scaling factor typically lies in the interval [0, 1]. X_{ri1G}, X_{ri2G} and X_{ri3G} are randomly chosen vectors from parent population with *r*1, *r*2 and *r*3 three randomly chosen numbers uniformly distributed between 1 and NP and different from each other.

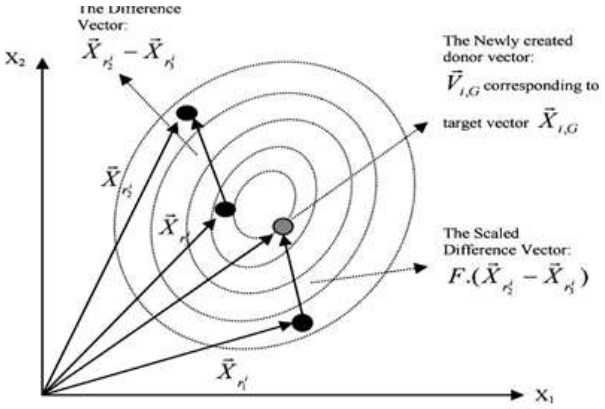

Fig. 1. Mutation operation of DE algorithm

Crossover: The goal is to increase the diversity of the perturbed population vectors. To achieve this goal, the mutated vector parameters are mixed with the parameters of another pre-determined vector, i.e. target vector to yield a trial vector. The trial vector is computed in the following way.

The trial vector for generation G+1 is represented as: $U_{iG+1} = (u_{1iG+1}, u_{2iG+1}, \ldots\ldots, u_{DiG+1})$

Each component of the trial vector at generation G+1 isformed using equation (2).

$$U_{i.j.G+1} = \begin{cases} v_{j,i,G} & \left(if\ rand_{i,j}[0,1] \leq Cr\ or\ j = j_{rand} \right) \\ x_{j,i,G} & otherwise \end{cases} \quad (2)$$

Here $rand_j[0, 1]$ is the j-th evaluation of a uniform random number generator, Cr is the crossover constant $\in[0,1]$ given by the user and j_{rand} is a randomly chosen index $\in\{1, 2, ...D\}$ which ensures that $U_{i,G+1}$ gets at leastone parameter from V_{iG+1}. The DE family of algorithms can use two kinds of crossover methods, exponential or two-point modulo and Binomial or uniform.

Selection: In selection, if the trial vector yields a lower fitness value than the target vector, the trial vector replaces the target vector inthe next generation, as given in equation (3). Each population vector must serve once as the target vector so that NP competitions take place in one generation. The initial population is chosen randomly andshould cover the entire parameter space.

$$X_{i,G+1} = \begin{matrix} \{U_{i,G}\ if\ f(U_{i,G} \leq f(X_{i,G}\} \\ X_{i,G}\ otherwise \end{matrix} \quad (3)$$

$f(.)$ is the objective function to be minimized

2.2 Q-Learning

Q-learning is a reinforcement learning procedure used to find an optimal action-selection policy by learning an action-value function that gives the expected utility of taking an action in a given state. When such an action-value functionis learned, the policy is framed by selecting the action with optimal value in each state. Q-learning is defined mathematically by equation (4).

$$Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a(Q(s_{t+1}, a) - Q(s_t, a_t))] \quad (4)$$

Where S is the set of states such thats$\in$ S and A is the set of actions i.e $a\in$ A. $Q(s_t, a_t)$ is the Q-value at time instant t after performing action a_t in state s_t. Rest of the terms in R.H.S. of equation (4) represents the learned Q-value at time t+1. The learning rate α determines to what extent the newly acquired information override the old reward by r_{t+1}, the reward at time (t + 1). Discount factor γ trades off the importance of different reward signals.

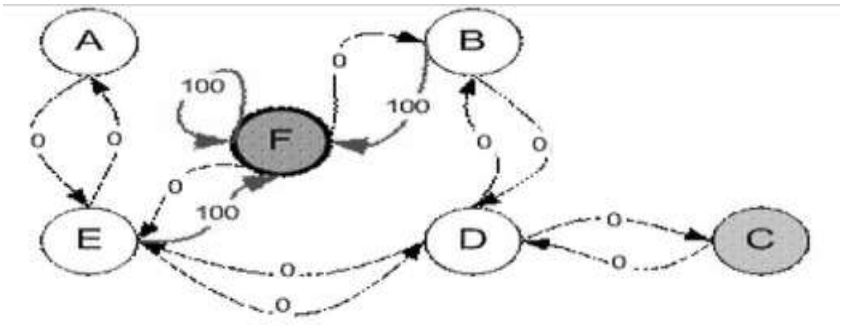

Fig. 2: Illustration of Q-learning algorithm

In Fig. 2 an example is described to understand Q-learning method. An agent can pass fromone state to another without any knowledge of the environment. Suppose an agent in state C and we want the agent to learn the sequence to reach to the final state F. Say, (A,B,C,D,E,F) are states of the set S depicted using nodes in the state diagram of Fig. 2 while action is represented by the arrow. When agent is in state C can go directly to state D but not directly to state B. From state D, the agent can go either to state B or state E or back to state C. If the agent is in state E, three possible actions can move

agent to state A, or state F or state D and soon. Given a state diagram the objective of the algorithm is to find minimum path from any initial state to the goal state.

3 PROPOSED APPROACH

We proposed an algorithm for solving large dimensional problems by integrating DE and QL algorithm as in Ref. 16, namely QL-DE algorithm. In this paper we modeled the QL-DE algorithm for both single agent and multi-agent environment.A multi-agent systemis designed to develop methods that allow building complex systems composed of autonomous agents, capable of enacting the desired global behaviors, based on local knowledge and possessing limited abilities. DE algorithm is a quick convergent algorithm while it has too many parameters to handle. On the other hand QL is very slow algorithm and not very proficient to handle problem with large search space. To overcome the drawbacks of these algorithms, a synergy of DE and QL is proposed and applied on multi-agent environment. To overcome slow convergence of QL algorithm, guided randomness has been introduced in the learning process using different operators of DE algorithm. In QL procedure the Q-value at a particular state is updated by mutation operation of DE. The mutation operation enforces diversity as random changes occurto vector space. The donor vector of DE representing the difference between two population vectors are randomly selected and mapped to update the Q value and therefore, the approach enforces a new learning paradigm.

3.1 DE based QL algorithm

DE algorithm begins with a randomly initiated population of size NP represented by D dimensional real-valued parameter vectors. Each vector of the population forms a candidate solution to the multidimensional optimization problem. Different operators of DE are used to compute Q values at different operations as explained below.

(i) In the context of DE algorithm, mutation is observed as change or perturbation with a random element. In the proposed QL-DE algorithm, the Q value is generated randomly using mutation operation and explores the large search space. The Q vector of QL-DE is $Q_{iG} = [q_{1iG}, q_{2iG} \ldots\ldots, q_{DiG}]$ is evaluated using equation (5).

$$Q_{iG+1}^{inter} = Q_{iG} + F * \left(Reward + (X_{r1G} - X_{r2G})\right) \quad (5)$$

Where, Q_{iG+1}^{inter} is the intermediate Q value generated using mutation and Q_{iG} is the old Q value while random indices $r_1, r_2 \in [1, 2, \ldots\ldots, NP]$. The two indices are mutually independent and different from index i. F is a scalar typically lays in the interval [0.4,1] and acts as the learning rate which controls the amplification of the differential variation (X_{r1G} - X_{r2G}).
Reward is based on the fitness function, defined as $exp(-w)$ such that($w = w_1 - w_2$),where w_1 is the fitness function of the parent vector at generation G and $w_2$is thefitness function of the child vector at generation G+1. The positive value of $w$indicates the fitness function is decreasing (in case of minimization) in nature when agent changes states and the reward is added otherwise, the reward is deducted as punishment. Therefore, the proposed algorithm tries to converge to minimum value and Q-value increases to justify the proper action.

(ii) The goal of crossover operation of QL-DE algorithm is to enforce diversity in the learning process. To achieve this, the Q vector of one state exchanges itscomponents with the Q vectors of other states to form the trial vector $Q_{jiG+1}^{trial} = [q_{1iG}, q_{2iG} \ldots\ldots, q_{DiG}]$ for the i-th target vector Q_{iG}. Here, exchange of elements in different states is performed to impose diversification in Q value that facilitates exploitation of large search spaceusing equation(6).

$$Q_{jiG}^{trial} = \begin{cases} Q_{jiG}^{inter} & if\ (\ rand_{ij}[0,1] \leq Cr\ \ or\ j = j_{rand}) \\ Q_{jig} & otherwise \end{cases} \quad (6)$$

Here, Cr is the crossover constant within [0, 1] and j_{rand} is randomly chosen index such that $j_{rand} \in [1,2,\ldots,D]$ to ensure that Q_{jiG+1}^{trial} gets at least one parameter from Q_{jiG} and act as momentum for the learning procedure.

(iii) Selection process is invoked to decide whether the target or trial Q vector survives to its next generation or it should become a member of generation *G*+1. Qvalues are updated using equation (7).

$$Q_{iG+1} = \begin{cases} Q_{iG}^{trail}, & if\ f\left(Q_{i,G}^{trail}\right) \le f(Q_{i,G}) \\ Q_{iG} & otherwise \end{cases} \quad (7)$$

Here *f* is the fitness function, equivalent to *Reward* of Q-learning algorithm. Updation, exchange and selection operations are performed for each individual andrepeated until any of the stopping criteria are reached.
Stopping Criteria can be defined in a few ways like:

I. By a fixed number of iterations, say G_{max}, suitably large depending on the complexity of the fitness function.
II. When best fitness of the population does not change over successiveiterations.
III. Attaining a pre-specified fitness value, (say,20% of the initial)

3.2 QL-DE algorithm

Algorithm 1:

Step1: *F*, *Cr* and population size *NP*
Step2: Generation no. *G*=0 and randomly initialize population of size *NP*, individual *PG* = $[X_{1G},X_{2G},....,X_{NPG}]$ and $Q_{iG}= [q_{1iG},q_{1iG}\ldots\ldots, q_{DiG}]$
Step 3: Function QL-DE (*F*, *Cr*, *NP*, *PG*)
WHILE (*until stopping criteria is reached*)
FOR i = 1 *to NP*
Step 3.1 Updation Step: Generate a donor vector Q_{iG+1}^{inter} corresponding to target vector Q_{iG}

$$Q_{iG+1}^{inter} = Q_{iG} + F * (Reward + (X_{r1G} - X_{r2G})$$

Step 3.2 Exchange step: Q_{jiG+1}^{trial} for the *i*-th target vector Q_{iG}

$$Q_{jiG+1}^{trial} = \begin{cases} Q_{jiG}^{inter}, & if\ (rand_{i,j}[0,1] \le Cr\ \ or\ j = j_{rand}) \\ Q_{jiG} & otherwise \end{cases}$$

Step 3.3 Selection Step: Evaluate the trial vector Q_{iG+1}^{trail}

$$Q_{iG+1} = \begin{cases} Q_{iG}^{trail}, & if\ f\left(Q_{iG}^{trail}\right) \le f(Q_{iG}) \\ Q_{iG} & otherwise \end{cases}$$

Step 3.4: G = G + 1
END OF FOR
END WHILE
END of FunctionQL-DE(*F*,*Cr*,*NP*,*PG*)

3.3 Role of Agents

A complex problem is solved by dividing the problem into sub-problems and multiple agents are involved to solve each such sub-problem. Here each vector is comprised of multiple variables and each variable is mapped as a state transition valueor action for a pair of states. Here the states of one agent are hidden from another agent and there is no interaction between two independent agents. However, when there is overlapping of sub-problems (common variables), agents interact for learning behavior of respective agents. The agents communicate through sharing of the common variables and producing impact on the respective sub-problems by updating thevalues of the common variables. But at a time stamp only one agent is allowed to make changes to the overlapped elements. Each agent executes the proposed QL-DE algorithm and communicates through common variable but how an agent solves a sub-problem that remains hidden from others.

Say, the initial population or the problem space represented by variables/attributes is divided among two agents (Agent1 and Agent2) on a random basis. Assignmentof variables between two agents using QL-DE algorithm is shown in Fig. 3. At each iteration variable assignment continues until consecutive result remains same.

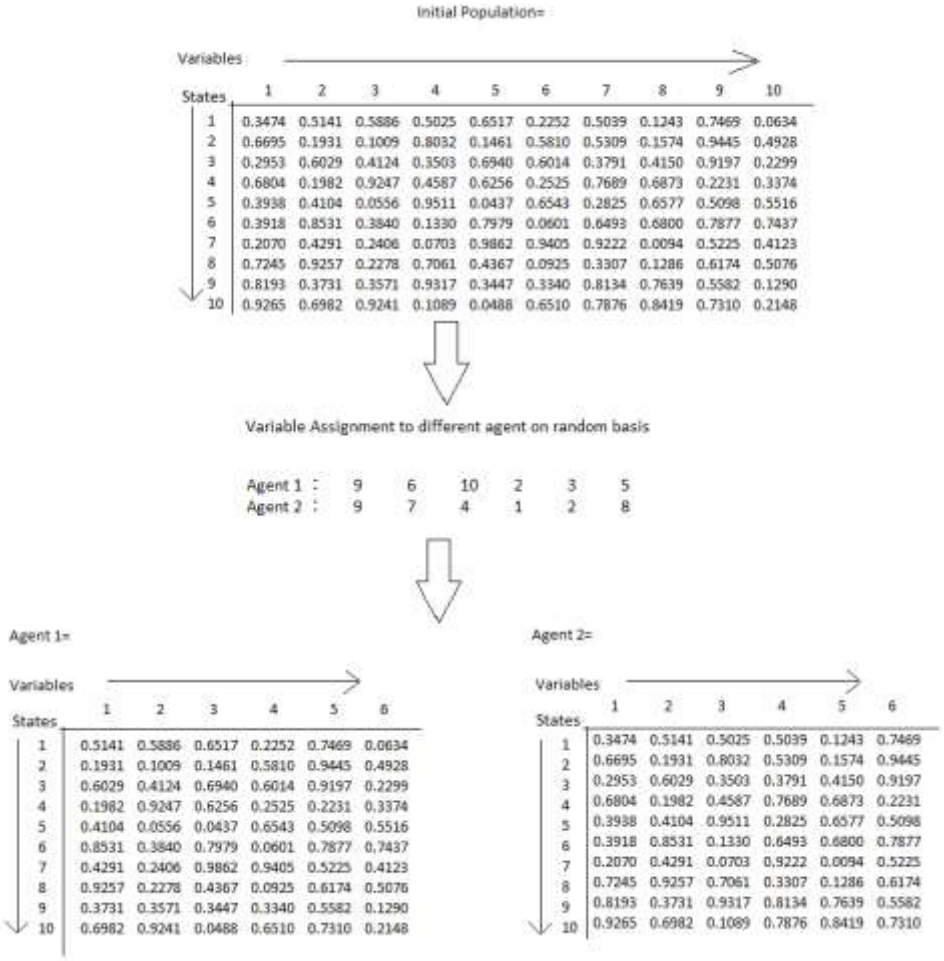

Fig. 3: Variable assignment to agents randomly

The flow of the proposed QL-DE algorithm for multi-agent based system isdescribed in Fig.4.

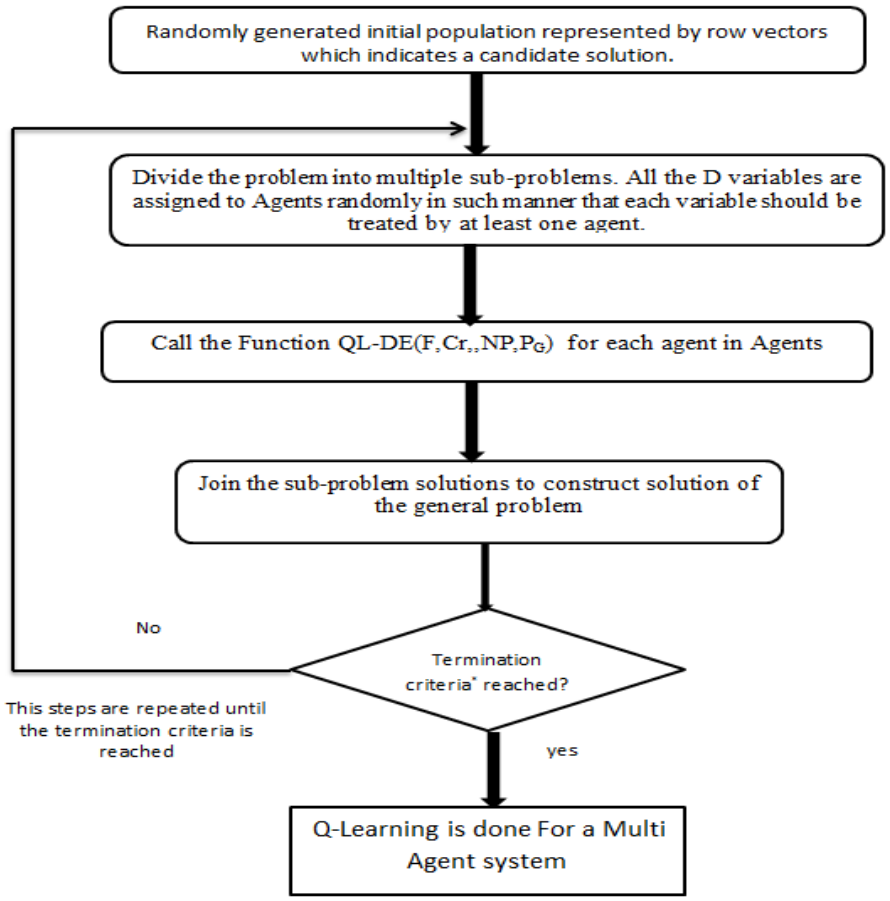

Fig. 4: Flow diagram of the QL-DE algorithm for Multi Agent system

3.4 QL-DE algorithm for Multi-agent based system (QL-DE-MAS)

Algorithm 2: QL-DE algorithm for Multi-agent based system

Step 1: Initialize F,Cr and population size NP
Step2: Set the generation number $G = 0$ and randomly initialize a population of NP individuals $PG = [X_{1G}, X_{2G}, ..., X_{NPG}]$ and $Q_{iG} = [q_{1iG}, q_{1iG} ..... q_{DiG}]$
$Agents = [A_1, A_2 ..... A_n]$ where, n is the number of agents.
Step 4: *QL-DE-MAS* (*Agents*, *n*, *PG*, *NP*, *F*, *Cr*)
Step 4.1: Divide the problem into multiple sub-problems so that D no. of variables are assigned randomly as elements of each Agent with a constraint that all variables should be treated by at least one agent.
Step 4.2: Call the Function *QL-DE* (*F*, *Cr*, *NP*, *PG*) for each element in the Agents
Step 4.3: Join the sub-problem solutions to construct solution of the problem
Step 5: Repeat the above step3 and step4 until the termination criteria is reached.
End

In multi-agent based approach, the attributes or variables of large problem space are distributed among multiple agents with an aim to attain maximumoutput using minimum number of agents. Proper allocation of attributes to eachagent using QL-DE algorithm solve large space problem efficiently.

4 RESULTS

4.1 Comparison between Q-Learning and QL-DE algorithm

Result of algorithm 1 applied for a single agent are presented using Benchmark functions to find the converging state and compared with Q-learning algorithm, as shown in figure 5. Result shows faster rate of convergenceof QL-DE algorithm compare to Q-Learning algorithm.

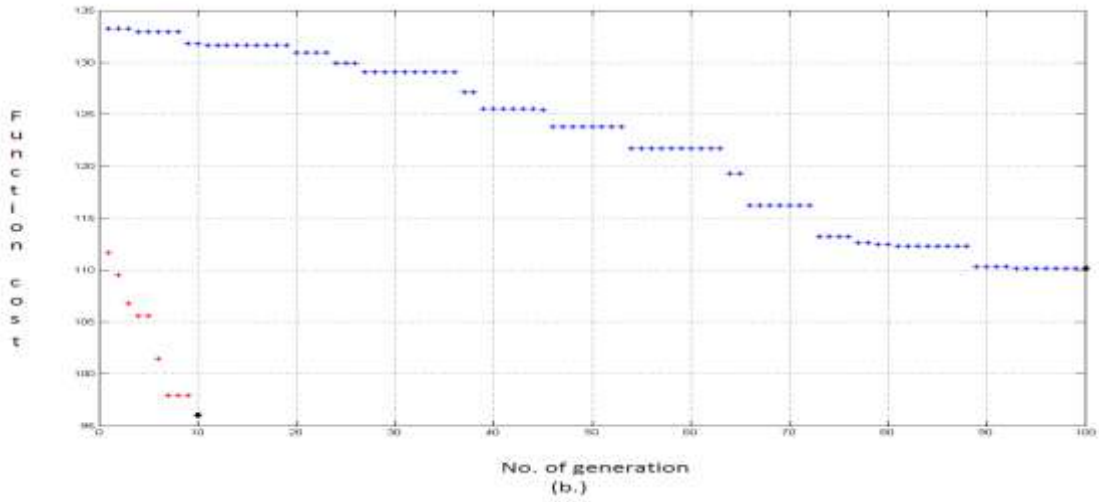

Fig. 5: Comparison between Q-Learning (blue asterisk) and QL-DE algorithm (red asterisk)

Two curves in Fig.5 show rate of convergence of Q-Learning and QL-DE algorithms. It has been observed that the cost function of Q-learning is almost 1.5 times higher than the QL-DE algorithm at the time of convergence.

4.2 Result of Multi-agent based QL-DE algorithm

The QL-DE algorithm has been applied to solve large space problem by dividing the problem into multiple sub-problems and assigned to multiple agents. Say, 1000variables in population set are initially randomly distributed among large numberof agents, say 37. After applying QL-DE-MAS algorithm, number of agents is reduced from 37 to 15, which is almost half of initial number of agents.

Comprehensive result and comparison of Q-learning and QL-DE algorithm applied in multi-agent system are shown in Fig.6.

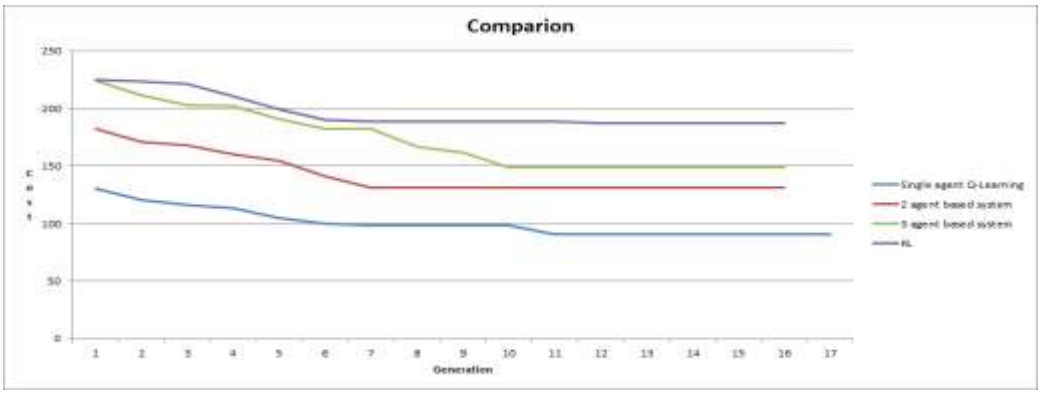

Fig. 6: Comparison between Q-Learning and QL-DE algorithm

5 CONCLUSIONS

The work provides an insight in building a simple yet strong algorithm for single and multi-agent based system to tackle large search space problem. Each agent individually optimizes the subspace

using the proposed DE based Q-learning algorithm (QL-DE). Through numerical experiments, it is observed thatperformance of Q-Learning could be improved by the proposed methods. It proves to be faster than Q-Learning method. The quick convergence property of DE enhances the convergence of QL-DE algorithm. It provides a satisfactory result incase of large space application the algorithm is invariant to initial distributionof variables. The variable resource management and the inter-variable relations (co-evolution and correlation) can be considered forfurther improvement of the result.

References

AKargar, H. K., Aghmasheh, R., Safari, A., &Govar, G. Z. (2008, December).*Multi-Agent-Based particle swarm optimization approach for PSS designing in multi-machine power systems*. In Power and Energy Conference, 2008.PECon 2008. IEEE 2nd International (pp. 73-78). IEEE.

Anbo, M., Xiangang, P., &Hao, Y. (2012, January).*Multi-agent based distributed genetic algorithm applied to the optimization of self-adaptive PID parameters of hydro-turbine*. In Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on (pp. 359-363). IEEE.

Bolívar Baron, H., Rojas, M. M., Crespo, R. G., & Martinez, O. S. (2012, September).*A multi-agent matchmaker based on hidden markov model for decentralized grid scheduling.* Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on.IEEE, 2012.

Cohen, I., Sebe, N., Gozman, F. G., Cirelo, M. C., & Huang, T. S. (2003, June).*Learning Bayesian network classifiers for facial expression recognition both labeled and unlabeled data*. In Computer Vision and Pattern Recognition, 2003.Proceedings.2003 IEEE Computer Society Conference on (Vol. 1, pp. I-595).IEEE.

G. Rudolph M. Laumanns and H.-P.Schwefel (2001).*Mutation control and convergence inevolutionary multi-objective optimization*.In Proceedings of the 7th International MendelConference on Soft Computing (MENDEL).

Goel, T., & Deb, K. (2002*).Hybrid methods for multi-objective evolutionary algorithms*. In Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL, 02).(Singapore) (pp. 188-192). Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL, 02).(Singapore).

Intille, S. S., &Bobick, A. F. (1999).*A framework for recognizing multi-agent action from visualevidence*. AAAI/IAAI, 99, 518-525.8.

Jun, H. B., &Sim, K. B. (1997).*Behavior learning and evolution of collective autonomous mobile robots based on reinforcement learning and distributed genetic algorithms*. In Robot and Human Communication, 1997.RO-MAN'97.Proceedings., 6th IEEE International Workshop on (pp. 248-253). IEEE.

Knowles, J. D., &Corne, D. W. (2000). M-PAES: *A memetic algorithm for multiobjective optimization*. In Evolutionary Computation, 2000.Proceedings of the 2000 Congress on (Vol. 1, pp. 325-332).IEEE.

Liu, Y. and S. Li,, 2011. *A new differential evolutionary algorithm with neighborhood search*.In Information Technology Journal 10, no. 3 (2011): 573-578.

Liu, Xiyu, Yinghong Ma, and Liandi Jiang.*Mobile Clustering Agents based on Differential Evolution*.InPervasive Computing and Applications, 2008.ICPCA 2008.Third International Conference on.Vol. 1.IEEE, 2008.

Maja J. Mataric. (1993, August).*Designing emergent behaviors: from local interactions to collective intelligence*. In Proceedings of the second international conference on From animals to animats 2.MIT Press, Cambridge, MA, USA, (pp.432-441).

Milano, M., &Roli, A. (2004).*MAGMA: a multiagent architecture for metaheuristics. Systems, Man, and Cybernetics*, Part B: Cybernetics, IEEE Transactions on, 34(2), (pp.925-941).

Oh, C. H., Nakashima, T., &Ishibuchi, H. (1998, May). *Initialization of Q-values by fuzzy rules for accelerating Q-learning.In Neural Networks Proceedings*, 1998.IEEE World Congress on Computational Intelligence.The 1998 IEEE International Joint Conference on (Vol. 3, pp. 2051-2056).IEEE.

Omidvar, M. N., Li, X., Yang, Z., & Yao, X. (2010, July*).Cooperative co-evolution for large scale optimization through more frequent random grouping*.In Evolutionary Computation (CEC), 2010 IEEE Congress on (pp. 1-8).IEEE.

Rahaman, Z., &Sil, J. (2014, January).*DE Based Q-Learning Algorithm to Improve Speed of Convergence in Large Search Space Applications.*InElectronicSystems, Signal Processing and Computing Technologies (ICESC), 2014 International Conference on (pp. 408-412). IEEE.

Stacy Marsella and JafarAdibi and Yaser Al-onaizan and Gal A. Kaminka and Ion Muslea and MilindTambe. (1999, April).*On being ateammate: Experiences acquired in the design of robocup teams,*Proceedings of the Third Annual Conference on Autonomous Agents.1999.(pp.221—227).ACM Press.

Wu, J., Xu, X., Zhang, P., & Liu, C. (2011).*A novel multi-agent reinforcement learning approach for job scheduling in Grid computing.* Future Generation Computer Systems, 27(5), (pp.430-439).

Zhao, Long, and Zemin Liu. (1996, June).*A genetic algorithm for reinforcement learning*.IEEE International Conference on.Neural Networks Vol. 2.(pp. 1056-1060). IEEE.