4

Real Time Multiagent Decision Making by Simulated Annealing

[1]Dawei Jiang and [2]Jingyu Han
[1]*National University of Singapore,*
[2]*Nanjing University of Posts and Telecommunications,*
[1]*Singapore*
[2]*China*

1. Introduction

In this chapter, the application of Simulated Annealing (SA) algorithm in real time multiagent coordination problem is described. A **M**ulti**a**gent **S**ystem (MAS) consists of a group of agents that interact with each other. Research in MAS aims to provide theories and techniques for agents' behavior management. The focus of this chapter is on fully cooperative MAS, where all the agents share a common long-term goal. Examples include a team of robots who play football against another team or a group of rescue robots that, after an earthquake, must safely rescue the victims as soon as possible. The challenging issue in such systems is ***Coordination***: the policy to insure that the individual action of each agent can generate the optimal joint action as a whole.

Coordination in MAS has been explored from many aspects such as game theory (Osborne & Rubinstein , 1999), communications (Carrier & Gelernter, 1989), social conversions (Boutilier, 1996) and learning(Tan, 1997). Unfortunately these approaches have some flaws. First, in the worst case, these approaches degrade to a naïve solution which searches the whole joint action space whose size grows exponentially with the number of agents (It is called "curse of dimensionality"). Therefore, they do not scale well for large systems. Second, many of the approaches report an answer only when all the possible statuses have been considered. This is not suitable for real time case. In many real time scenarios such as robot football, rescue robots, etc., it is often needed that decision making algorithm returns a well enough answer at any time.

Recently, there is some work on how to decrease the joint action space by *coordination graph* (CG) (Guestrin & Venkataraman , 2002). The insight in CG is that in MAS only a small part of agents need to coordinate their actions while others can still act individually. Thus, the global joint payoff function, the representation of the global joint coordination dependencies among all agents, is approximated as a sum of local payoff functions, each of which represents the local coordination dependencies between a small sub-group of the agents. Then, the agents use a *variable elimination* (VE) algorithm to determine their optimal joint action. Unfortunately, the worst time complexity of VE grows exponentially with the number of agents. Moreover, VE only reports results when the whole algorithm terminates, therefore it is unsuitable for real-time systems. Max-plus (MP) algorithm is proposed as an

Source: Simulated Annealing, Book edited by: Cher Ming Tan, ISBN 978-953-7619-07-7, pp. 420, February 2008, I-Tech Education and Publishing, Vienna, Austria

approximate alternative to VE (Kok & Vlassis, 2005). MP can converge to the optimal solution for tree-structured graphs and also find near optimal solutions in graphs with cycles, but it limits the local payoff functions to contain at most 2 agents.
In this chapter, An **S**imulate **A**nnealing (SA) based algorithms to address aforementioned coordination problem is presented. This approach has two main benefits. First, the time taken by the algorithm grows polynomial with the number of agents. Second, the algorithm can report a near-optimal answer at any time.
The chapter is organized as follows. Section 2 describes the problem setting and representative work on how to solve multiagent decision problem, especially on **V**ariable **E**limination (VE) approach. Section 3 introduces the general steps and key elements of SA algorithms, which is employed in later sections. Section 4 gives how to effectively find a satisfactory answer in any time for multiagent decision problem by SA algorithm. In Section 5, the performance of SA algorithm on multiagent decision problem is evaluated by comparing it with comparable approaches followed by conclusion and future work.

2. Problem setting and variable elimination approach

Multiagent decision making problem can be formally describe as follows.
Given a group of agents $G=\{G_1,G_2,\ldots,G_n\}$, they are interacting with each other together during a long time sequence $\{t_1,t_2,\ldots,t_n\}$ to reach final goal . At each time t_i, each agent G_i selects an individual action a_i from his own action set A_i (Thus the joint policy space is $A=\times A_i$) based on payoff function $v(a)$ and goes into next time t_{i+1}. At each time, the decision making problem is to find the optimal joint action a^* that maximize the global payoff function $v(a)$. That is to say, $a^*=maxarg_a\, v(a)$.
To overcome the curse of dimensionality, the global joint payoff function is decomposed into a linear combination of s set of local payoff functions, each of which is only related to a small number of agents. For example, in RoboCup, only the players that are close to each other have to coordinate their actions to perform a pass or a defend. In some situations, this approach can get a very compact representation for coordination dependencies among agents. Furthermore, such representation can be mapped onto a coordination graph $G=(V,E)$ according to the following rules: each agent is mapped to a node in V, and each coordination dependency is mapped to an edge in E. Then **V**ariable **E**limination (VE) can be used on G to determine the optimal joint actions.
Variable Elimination is also called bucket elimination. It is first used for reasoning in Bayes network. It can also be effectively used to solve the multiagent decision making problem. The technical steps include two passes. In the first pass, by enumerating all the possible combinatorial joint actions of his neighborhood, each agent conditionally computes his own optimal action and sends the result to the entire neighborhood. Then, the agent will be eliminated from the system. This process will continue until only one agent remains in the system. In the second pass, all agents do the entire process in reverse elimination order. In the process every agent can find his own optimal decision based on his neighborhood agent's behavior. An example is taken to illustrate the execution of VE algorithm. Suppose that the system has 4 agents with each one having 4 different actions, then the number of joint actions is $4^4=256$, and global joint payoff function can be decomposed as:

$$V(a)=v_1(a_1,a_2)+v_2(a_2,a_4)+v_3(a_1,a_3) \quad (1)$$

Fig.1 shows the initial corresponding coordination graph. The key idea in VE is that, rather than enumerating all possible joint actions and summing up all functions to do

maximization, each time only one variable is optimized. The example begins with optimization for agent 1. Agent 1 collects all local payoff functions including its own, i.e., v_1 and v_3 then does maximization. Hence, it can be obtained that

$$\max_a v(a) = \max_{a2,a3,a4} \{v_2(a_2,a_4) + \max_{a1} [v_1(a_1,a_2)+v_3(a_1,a_3)]\} \quad (2)$$

After enumeration of possible action combinations of his neighbors, i.e., agent 2 and agent 3, agent 1 conditionally returns his best response and yield a new function $e_1(a_2,a_3) = \max_{a1}[v_1(a_1,a_2)+v_3(a_1,a_3)]$. Its value at the point a_2, a_3 is the value of the internal max expression in equation (2). At this time, agent 1 is eliminated from G. The global joint payoff function is rewritten as:

$$\max_a v(a) = \max_{a2,a3,a4} \{v_2(a_2,a_4)+e_1(a_2,a_3)\} \quad (3)$$

Now fewer agents remain. Next, agent 2 does the same procedure. After collecting $v_2(a_2,a_4)$

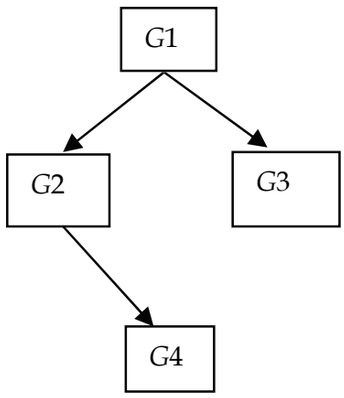


Fig.1. Initial coordination graph

and $e_1(a_2,a_3)$, agent 2 produces a conditional strategy based on the possible actions of agent 3 and agent 4, and returns his choice, i.e., $e_2(a_3,a_4) = \max_{a2} \{v_2(a_2,a_4)+e_1(a_2,a_3)\}$ to the system, then is eliminated. The global payoff function only contains 2 agents now:

$$\max_a v(a) = \max_{a3,a4} \{e_2(a_3,a_4)\} \quad (4)$$

Agent 3 begins to do optimization. Enumerating actions of agent 4, he reports his own choice and gives a conditional payoff $e_3(a_4) = \max_{a3} e_2(a_3,a_4)$. Finally, the only remaining agent 4 can simply choose his optimal action: $a_4^* = \arg\max_{a4} e_3(a_4)$.

In the second pass, all agents do the entire process in reverse elimination order. To fulfill agent 4's optimal action a_4^*, agent 3 must select $a_3^* = \arg\max_{a3} e_3(a_4^*)$. Then agent 2 can make a decision $a_2^* = \arg\max_{a2} e_2(a_3^*,a_4^*)$. Finally, agent 1 does $a_1^* = \arg\max_{a1} e_1(a_2^*,a_3^*)$ to choose his optimal action appropriately. The whole procedure needs only 4×4+4×4+4=36 iterations which is much smaller than 256 iterations of the whole joint action space.

The outcome of VE is independent of the elimination order and always gives the optimal joint action (Guestrin, 2003). However, the running speed of VE is depended on the elimination order and exponential in the induced width of the coordination graph (Guestrin

& Venkataraman, 2002) (Dechter,1999). Finding the optimal elimination order for VE is a well known NP-complete problem (Arnborg et al., 1987). Thus, in some cases and especially in the worse case, the time consumed by VE grows exponentially with the number of agents. Furthermore, VE can not give any useful results until the termination of the complete algorithm. Therefore, it is not suitable for real time multiagent decision making scenario. So in the following graph how to use simulated annealing (SA) approach to circumvent such limitations is addressed in detail.

3. Simulated annealing algorithms

The simulated annealing algorithm (also called as monte carlo annealing or probabilistic hill-climber), inspired by statistical mechanics, is very popular for combinatorial optimization. In this area efficient methods are developed to find minimal or maximal values for a function of a number of independent variables. The simulated annealing process executes by 'melting' the system being optimized at a high effective temperature at first, and then lowering the temperature by slow stages until the system 'freezes' and no further change occurs. In the following subsection the generic procedure to solve combinatorial optimization is introduced first, and then the essential factors in designing SA algorithm are analyzed.

3.1 Generic procedure to solve combinatorial optimization by SA

Given a generic function to be optimized f: $(x_1,x_2,\ldots,x_j,\ldots x_n)\rightarrow R+$, where $x_j \in S$ (here S is the domain) is a component of vector X and $N(x_j)\in$ S is the neighborhood of x_j. To find the maximal or minimal result, SA algorithm executes as the following 4 steps.

1. Initial temperature T_{max} and initial answer X (0) is given.
2. Based on X (i), a new resultant X' which contains a certain newly produced component $x' \in N(x(j))$ is obtained.
3. Whether X' will be accepted as a new answer X (i+1) depends on the probability

$$P(X(i)\rightarrow X')= \begin{cases} 1 & \text{if } f(X') < f(X(i)) \\ e^{-\frac{f(X')-f(X(i))}{T_i}} & \text{otherwise} \end{cases} \quad (5)$$

In other words, If $f(X')$ is less than $f(X(i))$ then $X(i+1)=X'$, otherwise X' will be accepted as X (i+1) with the probability of $e^{-\frac{f(X')-f(X(i))}{T_i}}$. If X' is rejected, the control flow goes to step 2 again until an acceptable X (i+1) is found.

4. Step 2 and 3 is repeated until a final status defined before reached.

It can be seen that the process of SA is a discrete status sequence. At each temperature T_i, its new status X (i+1) only depends on X (i) and has no relevance with X (i-1), X (i-2)…, X (0). Thus it is a Markov process.

3.2 Essential factors for designing simulated annealing algorithm

When a simulated annealing algorithm is designed, six essential factors should be taken into consideration.

3.2.1 Neighbor function (status production function)

A neighbor function is used to generate a new candidate answer based on current status. When a neighbor function is designed, it should ensure that all the candidate answers in the state space can be reachable. In general, designing a neighbor function focuses on two key aspects, which are the rule of producing candidate answers and the distribution of candidate answers. The former determines how to produce a candidate answer based on current answer. The latter determines the probability of newly produced different candidate answers. Usually production rule of neighbor function is devised according to concrete problem and distribution of candidate answers takes uniform distribution, normal distribution, exponential distribution and Cauchy distribution .etc.

3.2.2 Status transition probability (acceptance probability)

Status transition probability is the likelihood that one feasible answer, denoted as x_{old}, transits to another feasible answer, denoted as x_{new} . In other words, it is the chance that a new feasible answer will be accepted as current answer. As a rule, the status transition probability observes the followings.

1. At the same temperature, the chance to accept the candidate answer which will decrease objective function value is larger than that which will increase objective function value.
2. As the temperature declines, the chance to accept the answers that will decrease objective function value should gradually become smaller and smaller.
3. As the temperature is approaching zero, only the answers that make objective function value decrease can be accepted.

In most of the cases, Metropolis rule as equation (5) is used.

3.2.3 Cooling function

Cooling function determines how the simulated annealing proceeds from a high temperature T_{max} to lower temperature by stages. If the temperature decreases slow enough, the objective function value can concentrate on the global minima or maxima with an expensive cost. If the temperature decreases too fast, the global minima or maxima will not be reachable. Let $T(t)$ be the temperature at time t. The classical cooling function usually takes $T(t) = T_{\max}/lg(1+t)$ and the fast cooling function usually takes $T(t) = T_{\max}/(1+t)$. These two types of cooling function can gurantee the algorithm converge to the global minima or maxima.

3.2.4 Initial temperature

Many experiments show that the higher the initial temperature $T(0)$ is, the greater the chance of obtaining high quality final answer is. But the time consumed is also longer. Therefore, to get a better initial temperature $T_{\max}$, both optimization effectiveness and efficiency should be taken into consideration. Usually, the following several approaches can be applied.

1. At first, a group of statuses is obtained by uniform sampling. Then, the initial temperature T_{max} is defined as the variance of all the statuses' objective function values.
2. At first, a group of statuses is random obtained. Then, the biggest difference of objective function values, denoted as $|\triangle_{\max}|$, is obtained by comparing every two statues.

Finally, the initial temperature T_{max} is determined by a function which takes $|\triangle_{\max}|$ as parameter.

3. The initial temperature T_{max} is determined based on engineering experience for some specific problems.

3.2.5 Metropolis sampling rule

This rule is used to determine how many candidate answers will be produced at a certain temperature. The following policies are widely used.

1. Test whether the average of object function values is stable or not. If so, the sampling will continue, otherwise the sampling will stop.
2. Test whether objective function value difference in continuous steps is small enough. If so, the sampling will continue, otherwise, the sampling will stop.
3. The sampling is constrained by fixed number of steps.

3.2.6 Termination rule

It is used to determine when the simulated annealing algorithm ends. It includes the following approaches.

1. An ending temperature threshold is set. If the current temperature is below the threshold, the simulated annealing stops.
2. The number of iteration is set. The simulated annealing process will proceed according to the times of iterations.
3. The simulated annealing will end if the objective values do not change in a series of continuous steps.
4. The termination depends on whether the system entropy is stable or not.

4. Multiagent decision making by simulated annealing algorithm

It is natural to apply SA to multiagent decision making problem since the global payoff function needs to be optimized via a number of independent action variables of each agent. The process works as follows. First, the global payoff function is decomposed into a number of local terms. Then, global payoff function will be rewritten as the linear combination of the local terms to avoid the curse of dimensionality. That is to say, given n agents, its global payoff function can be decomposed as follows:

$$v(\mathrm{a}) = \sum_{i \in G} v_i(a_i) + \sum_{i,j \in G} v_{ij}(a_i, a_j) + \ldots + \sum_{i,j,k} v_{i,j,k}(a_i, a_j, a_k) + \ldots. \quad (6)$$

Here $v_i(a_i)$ represents the payoff that an agent contributes to the system when acting individually, e.g. dribbling with the ball. $v_{ij}(i,j)$ denotes the payoff of a joint action between agent i and j, and $v_{ijk}(a_i,a_j,a_k)$ depicts another coordination action involving three agents, e.g. pass from i to j, then j to k . Coordination among more agents can be added similarly if needed. This decomposition approach is different from MP for the number of agents is not limited while MP does. In MP algorithm, the global joint payoff function can only be decomposed into $\sum_{i \in G} v_i(a_i) + \sum_{i,j \in G} v_{ij}(a_i, a_j)$.

Now SA can be smoothly applied to solve the multiagent decision problem. The goal is to find the optimal joint action, i.e., $a^* = \arg\max_a v(a)$. The pseudo-code of SA is presented in

Alg.1. The SA algorithm is implemented in a centralized version and performed by the agents in parallel, without assuming the availability of communications. The idea behind the algorithm is very straightforward. In each iteration (called an independent try), the algorithm starts with a random choice of joint action for the agents, then loop over all the agents. Each agent optimizes the global payoff function with his own action while all of the other agents stay the same. If the agent's local optimization can yield a better joint action than the initial one, the new solution is accepted. Otherwise, the new solution is accepted with a probability of $\frac{1}{1+e^{-(\Delta/T)}}$. The looping continues until the temperature T decayed from T_{max} to a predefined threshold T_{min}. Then a new random starting position is selected and the whole process is repeated. When an agent should send action to the server, he returns his own action from the optimal joint action found so far.

Basically, what the SA does is to seek the global maximum of the global joint payoff function. As a stochastic algorithm, although SA can not guarantee the convergence to optimal joint action, in a rather short time it can find an approximately optimal solution. It has the following attractive features. First, SA is an anytime algorithm that can report an answer at any time. Secondly, in each independent try, agent i only has to iterate his own actions instead of all combinatorial actions of his neighbors, thus makes the algorithm tractable.

Algorithm 1. Pseudo-code of the simulated annealing algorithm
Define: $G = \{G_1, G_2, \ldots, G_n\}$ the agents who want to coordinate their actions
Define: $v(a)$ the global joint payoff function
Define: a^* the optimal joint action so far
Define: a_i the action of agent i
Define: a_i^* the optimal action of agent i found so far
Define: a_{-i} the actions of all agents but agent i

$g \leftarrow 0$
$t \leftarrow 0$
While $t < MaxTries$ do
a = random joint action
T$\leftarrow T_{max}$
repeat
for each agent i in G do
$a' = \text{argmax}_{a_i}(a_{-i} \cup a_i)$
$\triangle \leftarrow v(a') - v(a)$
if ($\triangle > 0$) then
$a \leftarrow a'$
else
$a \leftarrow a'$ with probability $\frac{1}{1+e^{-(\Delta/T)}}$
end if
if $v(a) > v(a^*)$ then
$a^* \leftarrow a$
$g \leftarrow v(a^*)$

choose a_i* from a*
end if
if should send action to server then
send a_i* to server
end if
end for
$T \leftarrow T.decay$
until $T<T_{min}$
$t \leftarrow t+1$
end while

5. Experiments

In this section, the simulated annealing algorithm is evaluated by comparing it with other algorithms, especially with variable elimination algorithm. The following subsections include three parts. The first subsection describes the test bed of experiment since there is no standard benchmark to use. The remaining two subsections give the details of the experiment. It runs in two stages. In the first stage, the number of agents and the number of different actions per agent are fixed to test the scalability of the two algorithms when the number of neighbors per agent grows. In the second stage, the relative payoff SA returned and the optimal payoff produced by VE is compared to evaluate SA algorithm's performance.

5.1 Test bed setting

Since there is no standard benchmark to evaluate multiagent decision algorithm, a *random generator* (RG) is used to generate all test sets. The inputs of RG include the number of agents $|\boldsymbol{G}|$, the number of different actions per agent $|A|$, maximum number of neighbors per agent nr_s, and the number of payoff functions each agent has nr_ρ. It is believed that these aspects should be sufficient to describe the difficulty of the coordination problem. The output of RG is a set of payoff functions. Each function is a value rule $<\rho : v>$, which is first used by literature (Guestrin & Venkataraman , 2002) and proved suitable for many real-world applications. The global joint payoff function is thus represented by the sum of value rules of all agents. A sample output of RG with $|G| = 4$, $|A| = 4$, $nr_s = 3$, $nr_\rho = 1$ is shown in table 1.

$<\rho : v>$
$< a_1 = 3 \wedge a_3 = 3 \wedge a_4 = 4 : 7.19085 >$
$< a_2 = 4 \wedge a_3 = 4 : 4.67774 >$
$< a_1 = 1 \wedge a_2 = 1 \wedge a_3 = 2 \wedge a_4 = 2 : 4.67774 >$
$< a_1 = 4 \wedge a_3 = 2 \wedge a_4 = 1 : 4.67774 >$

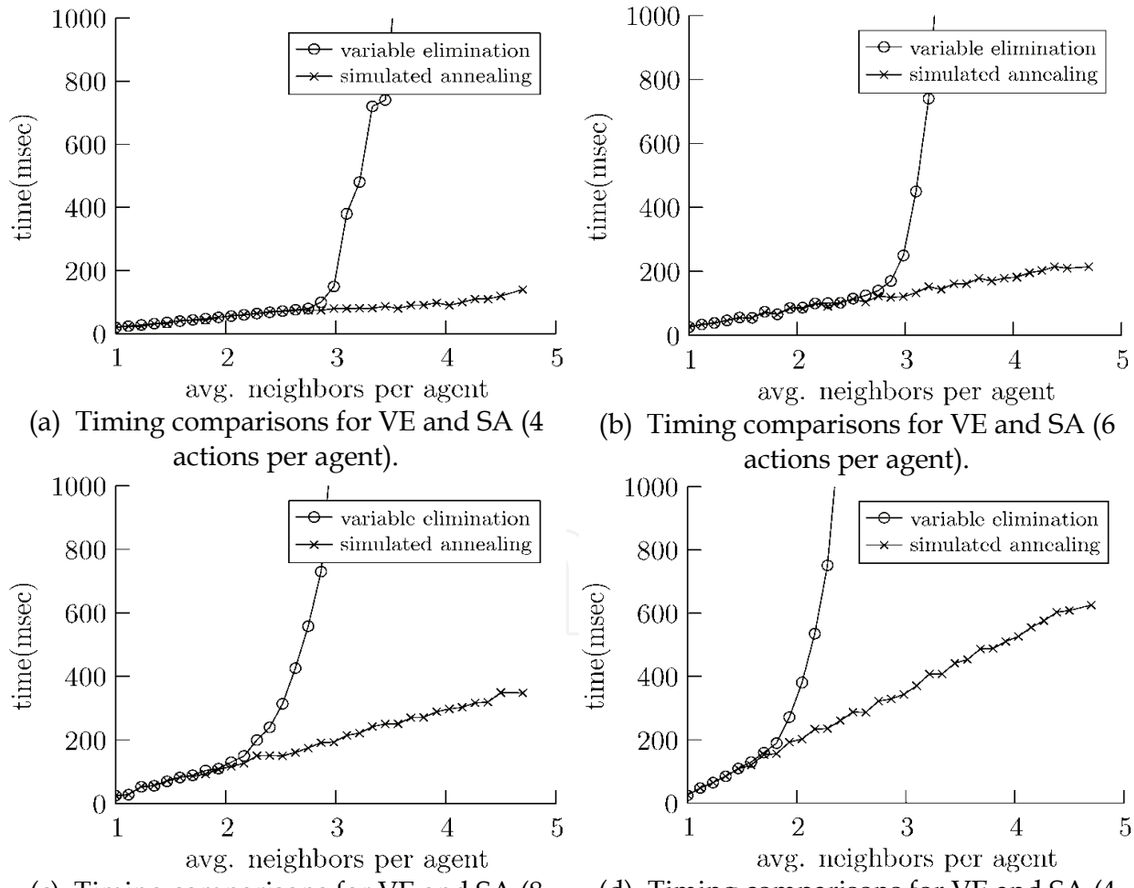
Table 1. Sample output of RG

Here the integer value of a_i is an action index and is mapped to a predefined action in real MAS such as dribbling, pass .etc. in a real RoboCup. The details will not be addressed here for the focus is concentrated on the performance of multiagent decision. The following two subsections give how to evaluate the performance of SA algorithms in details. All the programs are implemented in C++, and the results are generated on a 2.2GHz/512MB IBM notebook computer.

5.2 Scalability of SA algorithm

In this stage, 120 coordination problems are generated and each one is assigned with 4 test sets based on different actions per agent. The aim of this experiment is to evaluate the scalability of SA algorithm. For the problem in each test set, the settings are as follows. The number of the agents is set to $|G| = 15$. Each agent has $nr_\rho = 8$ value rules with different number of neighbors. The payoff in each value rule is generated from a uniform random variable U [1, 10]. The number of neighbors k in each value rule is in the range $k \in [1, nr_s]$. Each value has a chance of $\binom{Nr_{ne}}{k} / 2^{Nr_{ne}}$.

When applying VE, the algorithm is speed up by eliminating the agent with the minimum number of neighbors. When running SA, *MaxTries* is set to 20, the highest temperature T_{max} is 0.3, and lowest temperature T_{min} is 0.05. The temperature *decay* of this algorithm is in proportion to nr_s. Therefore, if certain value rule contains a large number of agents, the SA algorithm will search deeply in an independent try, vice versa. To weaken the side effect of hardware and operating system the experiment is repeated 10 times and the average is adopted as the measure. Fig. 2(a)–2(d) gives a clear picture of the timing results for the four



(a) Timing comparisons for VE and SA (4 actions per agent).

(b) Timing comparisons for VE and SA (6 actions per agent).

(c) Timing comparisons for VE and SA (8 actions per agent).

(d) Timing comparisons for VE and SA (4 actions per agent).

Fig.2. Time consumed comparisons for both VE and SA

test sets. It can be easily seen that the running time of SA algorithm grows linearly as the number of the neighbors per agent increases. In contrast, the time of VE algorithm grows exponentially, since it must enumerate all neighbors' possible combination actions in each iteration.

Furthermore, when the average number of neighbors per agent is more than 3.5, VE can not always compute the optimal joint action so these tests are removed from the test sets. In sum, the SA algorithm outperforms the VE algorithm with respect to scalability and this is especially meaningful in multiagent decision scenario.

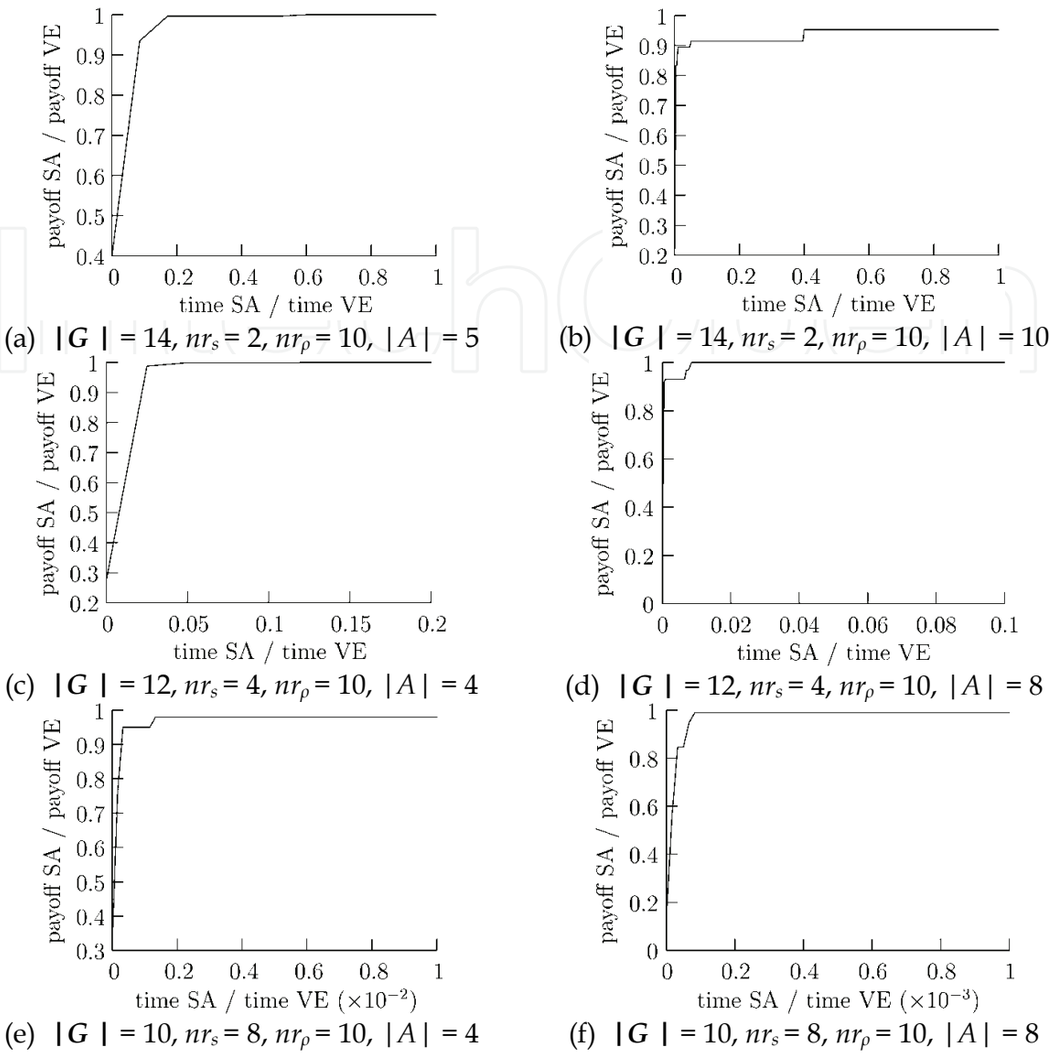
5.3 Relative payoff comparison

In the second stage, 6 coordination problems are generated. Each problem has its own settings such as number of agents, number of neighbors per agent .etc.. VE and SA are both employed to solve them. When SA is applied, instead of starting with a random choice for all agents, in i^{th} independent try, the agent selects action according to the i^{th} highest value rule if he is involved; otherwise the action is selected randomly. The *MaxTries* is set to 200, so that SA has enough time to run. Other settings are the same as the first stage.

To give a clear comparison of VE and SA, the payoff axis is scaled so that the global maximum payoff is 1. The time axis is also scaled so that the whole time taken by VE to terminate is 1. Thus the points in the figure can be seen as the fraction of the payoff and the running time of VE. The results of SA will be scaled to its VE companion. The experiment is also repeated 10 times to weaken hardware and software's side effect.

The relative payoff found by the SA with respect to VE is plotted in Fig. 3(a)–3(f). It can be seen SA performed very well. It is obvious that the near optimal result is found in all tests. In loosely connected coordination problem with few actions, i.e., Fig. 3(a), SA converges to the maximum payoff with only the 60% time that VE takes. However, if the number of actions is big as Fig. 3(b), SA can not reach the optimal result although it can find near optimal solution (96% payoff) quickly. Further experiments show that if the joint action space is huge (more than 15 agents, and each agent has more than 10 actions) the acceptable probability should be increased to speed up the convergence to optimal result. This is because in such situations, a little higher acceptable probability can increase the chance of stochastic movement of SA. This technique help SA jump away from local optimizations and cover the joint action space as possible as it can. But the exact relationship between acceptable probability and the convergence speed is still not very clear. For the medium connected problems (Fig. 3(c)–3(d)), SA can compute the optimal policy with a little fraction of time (2%–6%) that variable elimination needs to solve the same problem. Fig. 3(e) and Fig. 2(f) give us a strong impression that SA can compute more than 98% payoff within the time ranges between 0.015% to 0.2% of the time VE takes in the densely connected problems.

Other unpublished tests are also carried out. For example, SA is compared with max-plus algorithm informally. The experiment shows that when reaching the same relative payoff, the time difference between the two algorithms is at most 5%. Although SA algorithm is not much faster than max-plus, it is still believed that SA approach is more appropriate for complex coordination problems, since in these problems the coordination dependencies in the value rule is often more than two, therefore max-plus can not be applied directly.



(a) $|G| = 14, nr_s = 2, nr_\rho = 10, |A| = 5$ (b) $|G| = 14, nr_s = 2, nr_\rho = 10, |A| = 10$

(c) $|G| = 12, nr_s = 4, nr_\rho = 10, |A| = 4$ (d) $|G| = 12, nr_s = 4, nr_\rho = 10, |A| = 8$

(e) $|G| = 10, nr_s = 8, nr_\rho = 10, |A| = 4$ (f) $|G| = 10, nr_s = 8, nr_\rho = 10, |A| = 8$

Fig. 3. Relative payoff found by SA with respect to VE.

6. Conclusion

In this chapter, SA algorithm is employed to solve real time multiagent decision making problem. Compared with exact method this chapter's empirical evidences show that (1) this method is almost optimal with a small fraction of the time that VE takes to compute the policy of the same coordination problem; (2) the running time of SA grows linearly with the increasing number of neighbors per agent;(3) it is an anytime algorithm which return result at any time. For above reasons, it is believed that SA is a feasible approach for action selection in large complex cooperative autonomous systems.

As future research, an appropriate setting of the acceptable probability will be figured out, especially the decay rate in SA. Some recent work shows that neural network algorithm can produce a good decay rate for larger problems. Such techniques may be employed to solve multiagent decision making problem. Furthermore, whether reinforcement learning algorithms can be applied to automatically learn the payoff in each value rule is to be investigated

7. References

Arnborg,S.; Corneil,D.G. & Proskurowski,A. (1987). Complexity of finding embeddings in a K-tree. *SIAM Journal on Algebraic Discrete Methods*, vol.8, no.2, (1987) page number (277-284), ISSN: 0196-5212

Boutilier,C.(1996). Planning, learning and coordination in multiagent decision processes, *Proceedings of the 6th conference on theoretical aspects of rationality and knowledge*,pp.195-210,ISBN:1-55860-417-9,the Netherlands,1996, Morgan Kaufmann publisher Inc., San Francisco,CA, USA

Carrier,N.; Gelernter,D.(1989). Linda in context. *Communications of the ACM*, vol.32, no.4,(1989)page number(444-458),ISSN: 0001-0782

Dechter,R.(1999). Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, vol.113,No.(1-2),(1999)page number(41-85),ISSN: 0004-3702

Guestrin, C. ,Venkataraman (2002). Context specific multiagent coordination and planning with factored MDPs, *Proceedings of the eighteenth national conference on artificial*, pp.253-259,ISBN: 0-262-51129-0, Edmonton Canada, July 2002, American Association for Artificial Intelligence, Menlo Park, CA, USA

Guestrin,C. (2003).Planning Under Uncertainty in Complex Structured Environments .PhD thesis ,Stanford University

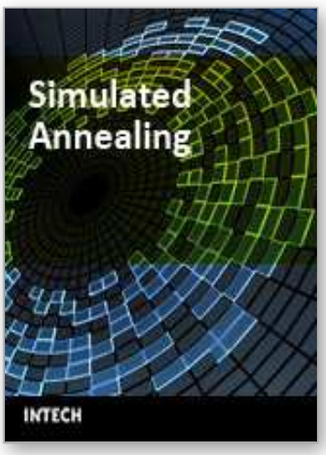
Jiang,D.W;Wang,S.Y.(2007). Using the simulated annealing algorithm for multiagent decision making, proceedings of the 10th annual RoboCup international symposium,pp.109-120,ISBN: 978-3-540-74023-0, Bremen Germany, June 2006,Springer, Berlin Heidelberg Germany

Kok,J.R.; Vlassis,N.(2005).Using the max-plus algorithm for multiagent decision making in coordination graphs, *Proceedings of RoboCup 2005*, pp.1-12,ISBN: 978-3-540-35437-6,Osaka, Japan, 2005, Springer Berlin, Heidelberg

Osborne,M.J.;Rubinstein,A.(1999). *A course in game theory*, MIT Press, ISBN: 978-0262650403, Cambridge of USA

S.Kirkpatrick, C.D.Gelatt, Jr., M.P.Vecchi (1983). Optimization by Simulated Annealing. *Science*, vol.4598,no.220, (May 1983)page number(671-681),ISSN: 220.4598.671

Tan,M.(1997). Multi-agent reinforcement learning: independent vs. cooperative learning. *Proceedings of Readings in agents*,pp.487-494,ISBN:1-55860-495-2,Morgan Kaufmann Inc., San Francisco, USA



Simulated Annealing

Edited by Cher Ming Tan

ISBN 978-953-7619-07-7
Hard cover, 420 pages
Publisher InTech
Published online 01, September, 2008
Published in print edition September, 2008

This book provides the readers with the knowledge of Simulated Annealing and its vast applications in the various branches of engineering. We encourage readers to explore the application of Simulated Annealing in their work for the task of optimization.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Dawei Jiang and Jingyu Han (2008). Real Time Multiagent Decision Making by Simulated Annealing, Simulated Annealing, Cher Ming Tan (Ed.), ISBN: 978-953-7619-07-7, InTech, Available from: http://www.intechopen.com/books/simulated_annealing/real_time_multiagent_decision_making_by_simulated_annealing

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen