Computing and Informatics, Vol. 28, 2009, 673–693

THE IMPACT OF THE CONFLICT ON SOLVING DISTRIBUTED CONSTRAINT SATISFACTION PROBLEMS

Samaneh Hosseini, Kamran Zamanifar

Department of Computer Science
University of Isfahan, Isfahan
Iran
e-mail: samaneh_hosseiny@yahoo.com, Zamanifar@eng.ui.ac.ir

Manuscript received 19 November 2007
Communicated by Jozef Kelemen

Abstract. Distributed Constraint Satisfaction Problems (DCSPs) involve a vast number of AI and Multi-Agent problems. Many important efforts have been recently accomplished for solving these kinds of problems using both backtracking-based and mediation-based methods. One of the most successful mediation based algorithms in this field is Asynchronous Partial Overlay (APO) algorithm. By choosing some agents as mediators, APO tries to centralize portions of the distributed problem, and then each mediator tries to solve its centralized sub-problem. This work continues until the whole problem is solved. This paper presents a new strategy to select mediators. The main idea behind this strategy is that the number of mediators conflicts (violated constraints) impacts directly on its performance. Experimental results show that choosing the mediators with the most number of conflicts not only leads to considerable decrease in APO complexity, but also it can decrease the complexity of the other extensions of the APO such as IAPO algorithm. MaxCAPO and MaxCIAPO are two new expansions of APO which introduce this idea and are presented in this article. The results of using this mediator selection strategy show a rapid and desirable improvement over various parameters in comparison with APO and IAPO.

Keywords: Distributed Constraint Satisfaction, APO, cooperative mediation, multi-agent

1 INTRODUCTION

A Distributed Constraint Satisfaction problem (DCSP) is a problem of finding suitable values to assign to distributed variables. This distributed environment involves multiple autonomous agents, each one holding one or more variables. An autonomous agent can suggest solutions to the other agents by exchanging some messages, and the receiving agents can accept or refuse the solution according to their preferences. This model can represent a vast number of real-world and multi-agent problems, such as distributed meeting scheduling [14], distributed resource allocation problems [3] and multi-agent truth maintenance [7]. Due to the variety of problems in this domain, several algorithms have been proposed since 1991 to solve them. These kinds of algorithms can be divided into two categories. Some of them, such as Asynchronous Backtracking (ABT) [17] and Asynchronous Weak-Commitment (AWC) [16] are completely distributed, while others use a hybrid of distributed and centralized methods. One of the best known algorithms of the second group is Asynchronous Partial Overlay (APO) [12] which is represented by Mailler and Lesser. On the first category algorithms agents cannot reveal information that breaks the privacy; it means that the agents dont have sufficient information about the global effects of making their local decisions. Although the first category algorithms nearly satisfy privacy, the second group of algorithms outperforms them by revealing necessary information. The second group of the algorithms and specially APO are inheriting the speed of centralization while using the advantages of parallelism. This new methodology, which is called cooperative mediation, is a method between centralized and distributed problem solving methods.

The main part of the APO algorithm is the coordinator selection, in which the participating agents select the highest priority agent as mediator when they recognize some conflicts about themselves. The mediator completes its information about the space of the problem by exchanging some messages with the other agents, and then, according to the received information, tries to solve the sub-problem locally by changing its local variable or by starting a mediation session. All the related agents participate in this session, receive the mediator messages and also send suitable messages to the coordinator. Then the coordinator computes a solution for this part of the problem and recommends the new value to the other agents. As they are autonomous, they can accept or refuse these values. In this way each coordinator solves a part of the whole problem and at last by putting these parts next to each other, the whole problem will be solved, just like putting the pieces of a puzzle next to each other. As it can be seen each coordinator solves its part of the problem in a centralized manner. Although APO outperforms its previous algorithms, new researches have been done recently that outperform APO. One of these researches is proposed by Benicsh and Sadeh [1]. They investigated different heuristics for mediator selection and found that if you place higher priority on the agents with the smallest good-list, then you will see a substantial speedup from the solver.

As it can be seen, one of the most important parts of the APO algorithm is the mediator selection part. So it seems to be a key part of the algorithm and several effective strategies can be proposed in this part. Making a small change in coordinator selection strategy can lead to a completely different solution. These all show the importance of choosing the best mediator selection strategy.

In this paper a new and effective strategy is proposed. The chief idea of the method is developing a heuristic method based on the number of a mediators conflicts to select more effective agents as mediators. In fact the agents that have the most complete information about the conflicts can compute the best solutions. The use of the agents with the largest number of conflicts will increase the speed of the algorithm and decrease the number of messages exchanged.

In the rest of this article, first, the DCSP definition is presented formally and an overview of previous works performed on DCSP is given. Section 3 introduces the MaxCAPO (Max Conflict APO) algorithm which is an extension of APO and uses the new strategy described for choosing mediators along with an overview of APO and an example of execution APO and MaxCAPO. Next, Section 4 introduces the MaxCIAPO (Max Conflict Inverse APO) algorithm that is an extension of IAPO, which itself is an extension of APO and is also presented in this paper briefly. Section 5 presents the experimental setup and results of applying this mediator selection strategy to the APO and IAPO in addition to comparing APO, MaxCAPO, IAPO and MaxCIAPO results. Finally, in Section 6, the conclusion of this work is presented and some possible future tends in this area are introduced.

2 BACKGROUND

In this section, first DCSP is introduced formally, and then the woks which have been performed to solve these problems until now are presented briefly.

2.1 Distributed Constraint Satisfaction Definition

A DCSP is a distributed form of CSP. This distributed environment involves multiple autonomous agents each one holding one or more variables. It was first discussed by Sycaro et al. and Yokoo et al. [15, 17]. The CSP which is the basis of DCSP is formally defined as follows:

- A set of n variables: $V = \{x_1, \ldots, x_n\}$
- A set of finite, discrete domains for each variable: $D = \{D_1, \ldots, D_n\}$
- A set of constraint: $R = \{R_1, \ldots, R_m\}$ where each $R_i(d_{i1}, \ldots, d_{ij})$ is a predicate that is defined on the Cartesian product $D_{i1} \times \ldots \times D_{ij}$. If the value assignment of these variables satisfies this constraint, the predicate returns true and otherwise false.

The final goal of solving DCSP is finding an assignment of values to all variables which satisfy all the constraints in R. Each agent tries to reach this goal not only

by satisfying its local constraints but also by communicating with other agents to solve external conflicts. As can be seen, agents should have strong communications with each other because their goals are interrelated. For example, in order to solve its sub-problem, each agent may create new conflicts for other agents by changing its or other agents' value.

In this paper, it is assumed that agents can communicate with each other by exchanging various messages and that the receiver agent receives messages exactly in the order they were sent, of course, after a finite delay. And it is also assumed that just one variable is under the control of each agent for simplicity. So the name of the agent can be the same with the name of the variable that it holds and manages. Each agent has the complete information about the constraints on its variable. The next assumption is that the constraints are defined only between two variables which are called binary constraints. It is clear that these restrictions are easily removable. Itemized and numbered lists can be created by help of standard LATEXenvironments "itemize" and "enumerate".

2.2 Related Work

Several algorithms with their advantages and disadvantages have been proposed, since formulizing DCSPs. The most powerful ones have been Asynchronous Backtracking (ABT) [17], Asynchronous Weak Commitment (AWC) [16], and the latest and the most successful one which is called Asynchronous Partial Overlay (APO) [10]. ABT algorithm is the distributed form of backtracking algorithm. As the backtracking algorithm tries to solve CSPs, the ABT tries to solve DCSPs. In the ABT algorithm, each agent assigns a random value of its domain to its variable. The agents communicate with each other by sending "ok?" and "nogood" messages. By receiving a message, the receiver agent will save the messages information on its agent-view, which contains the state of the other agents from its viewpoint. In this algorithm, each agent has a priority number which is determined according to the alphabetical order of the agents variables. Since each agent has a priority number, if an agents current value is not consistent with the value of the higher priority agents, it will change its assignments. In other words, in such a condition, the agent revises its assigned values and if no consistent value remains it will backtrack. In the latest condition, the agent generates a new "nogood" message, and sends it to the higher priority agent. The higher priority agent, by receiving this message, changes its value. In this algorithm the "nogood" list is a subset of agent-view, which shows the list of the agents that cannot find any consistent value with the subset.

Another successful algorithm is AWC which extends ABT but presents a new min-conflict heuristic. By using this heuristic, the risk of choosing a bad solution is reduced. The performance of these two algorithms is improved rapidly by presenting several heuristics.

The next successful algorithm, APO, uses a hybrid of distributed and centralized methods. The main idea of this algorithm is selecting special agents as mediators to solve their local sub-problems in a centralized manner. One of the most important

parts of this algorithm is mediator selection part, in which all the agents that recognize the existence of one or more than one conflicts select the most suitable agent as the mediator. The mediator agent is responsible for solving the sub-problem by exchanging various messages. To do this, the mediator tries to gather necessary information from other agents by starting a mediation session. Although APO outperforms its previous works in some problem domains, several heuristics have been proposed recently which try to remove its weaknesses. For example, Roger Mailler had done some work on expanding APO to operate in dynamic environments [8], to solve optimization problems [11] and to provide privacy [9]. Also, Benicsh and Sadeh improved APO by changing the mediator selection strategy. They confirmed that choosing agents with smallest good-list size can result in substantial speedup from the solver [1]. Benicsh and Sadeh also presented a different decentralized hybrid strategy which works based on ABT [2]. As can be seen, several researches have been done in this field, but none of them have considered the impact of the number of conflicts in mediator selection procedure.

3 MAX CONFLICT APO (MAXCAPO)

In this section, in addition to introducing APO algorithm, MaxCAPO, a new expansion of APO, which uses a new strategy to select mediator agents, is proposed, and finally an example of executing both of the algorithms is presented to exactly show the differences between them.

3.1 APO Overview

Since APO is the basic form of the MaxCAPO, it is presented it in a summarized form in Figure 1. You can find the latest and the most accurate version of this algorithm in [10]. As can be seen in Figure 1, APO starts by sending out an "Init!" message from each agent to its neighbors. This message contains the primary information about the sender, such as its name, priority, value, etc. While receiving an "Init!" message by each agent, it records the received information in its agent-view. Agent-view is a list that contains the primary information about linked agents. In fact, it is the agents view of the problem. Then each agent checks its agent-view to find any possible conflict with its neighbors. If one or more conflicts are found and no higher priority agent wants to mediate, the agent itself accepts the mediator role.

The priority of the agents in APO is determined according to their number of neighbors, and if two agents have the same size, it is determined according to the lexicographical ordering of their names. Of course, considering the priority for agents has several advantages. The most important one is that it guarantees that the agent which has the most knowledge about the sub-problem (which is the agent that has the largest number of neighbors) gets the role of the mediator and makes decisions.

The mediator first tries to correct the conflict by changing its local variable. But, if it is impossible, the mediator will start a mediation session by sending out an "evaluate?" message to the agents in its good-list. Good-list is a list that contains the information of any agent that connects to the owner directly by a link or indirectly by several links. The agent-view and the good-list are two main data structures that each agent defines. The main difference between these two lists is that agent-view holds just the information of directly linked agents but good-list holds the information of any agent that is connected to the owner by a direct link or by a path in the graph.

The receiving agent will reply with a "Wait!" message, if it is participating in another session or is expecting a request from a higher priority agent; otherwise, it replies with an "Evaluate!" message. While receiving the response message, the mediator starts a Branch and Bound search [5] among the agents which responded by "Evaluate!" message. If the mediator succeeds in finding the solution, it will inform other agents about their new values by sending "Accept!" messages. It also sends an "Ok!" message to the agents that were not participating on the session to update their agent-view.

Sometimes the solution that is found by mediator causes some violations for the agents outside the session. In such a condition the mediator adds these agents to its good-list and links with them. Therefore the mediator supposes that it is somehow related to these agents. This work prevents the mediator from repeating this mistake in future and in other sessions. This step of the algorithm is called "linking" step.

APO is a sound and complete algorithm. The soundness and completeness of this algorithm is shown by Mailler et al. [10].

3.2 MaxCAPO Algorithm

As mentioned in the previous section, APO generates a priority number for each agent. This number is generated according to the agents good-list size. The primary idea behind this mediator selection strategy is that the agents which have the highest amount of information about the problem can choose better solutions by making better decisions. Therefore the priority ordering in APO is a key point and is very important because it guarantees that the decisions are made by the agents which have the highest knowledge about the sub-problem. The second advantage of considering the priority ordering for agents is that it helps the effectiveness of the mediation process by choosing the higher priority agents as mediators and lets free the lower priority agents to be available for future mediation requests.

But according to this mediator selection strategy, it frequently occurs that several agents have the same good-list size. In this situation, APO determines the priority according to lexicographical ordering of the agents names. For example the priority of ND5 is more than the priority of ND2 if they have the same good-list size. In fact it is a random way for breaking the tie that had occurred. In

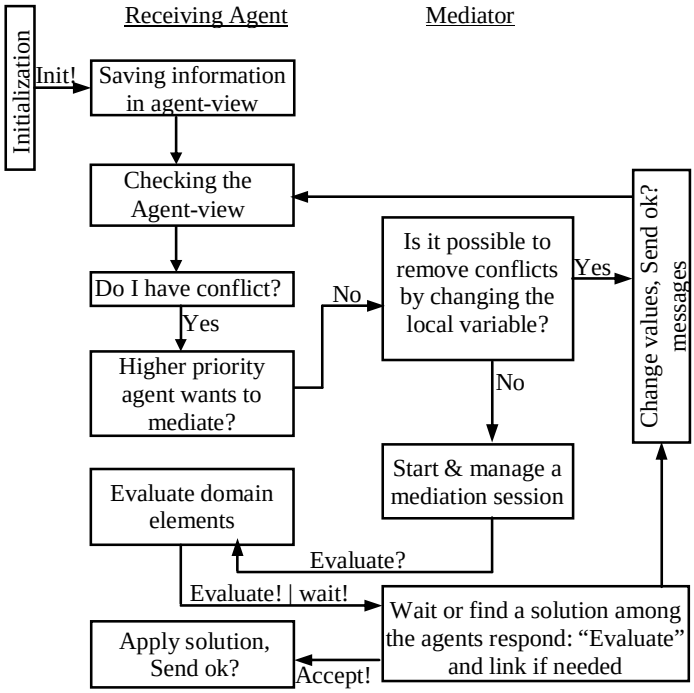


Fig. 1. APO algorithm

MaxCAPO this random method is replaced with a heuristic one. We believe that if several agents have the same number of neighbors, the agent that has the highest number of conflicts with its neighbors can make better decisions with exchanging fewer messages than the others. This happens because this particular agent exists exactly in the middle of the problem and, of course, is the most related agent to the conflicts. On the other hand, it has the highest information about the sub-problem restrictions in comparison with the other agents. So, in MaxCAPO, the priority is determined at the first step according to the agents good-list size, and at the second step according to the number of conflicts of each agent. Exactly the same reason that we mentioned before for the importance of considering the priority, is correct now for the importance of using this heuristic method. In fact, by choosing the agent with the highest number of conflicts MaxCAPO ensures that the agent with the highest knowledge gets the role of the mediator, and so makes the fundamental decisions; and of course it improves the effectiveness of the mediation process because it inherits APO.

The name of this algorithm, "MaxCAPO", comes from "Maximum Conflicts APO", which means that this algorithm is an expansion of APO that prefers to choose agents with the maximum number of conflicts instead of choosing them according to lexicographical order. It is obvious that the algorithm uses this rule just

when the good-list sizes of two or more agents are the same. The other parts of the algorithm are similar to APO.

3.3 An Example

In this section, an example of a solvable Distributed 3-color Graph Coloring (D3GC) problem is presented to compare the performance of APO and MaxCAPO. In a general definition, Distributed Graph Coloring consists of a set of n distributed variables $V = \{x_1, \ldots, x_n\}$ in which each element is associated with a set of possible colors $D = \{D_1, \ldots, D_n\}$. The number of elements of all the domains is equal to k which is the number of possible colors, and, at last, a set of constraints $R = \{R_1, \ldots, R_m\}$ in which $R_i(d_i, d_j)$ is true if the value of x_i is "not equals" to the value of x_j. The goal is assigning special colors to the variables in which all constraints in R return true. If we set k to 3, the problem will be D3GC. Obviously D3GC is a kind of DCSP.

Consider the problem in Figure 2 a). There are 8 agents (nodes) in this problem, each one with exactly one variable, and there are 12 edges between them that show all satisfied and violated constraints. The domain of each variable is Red, Blue, Black, because it is a 3-coloring problem. As can be seen there are 5 violated constraints at the beginning: (ND0, ND5), (ND2, ND3), (ND2, ND6), (ND3, ND4) and (ND5, ND7). The purpose of APO and MaxCAPO is to find a suitable assignment to the variable in which none of the connected nodes have the same color.

When starting in both of the algorithms, each agent sends an "init!" message to its neighbors and also adds itself to its good-list. By receiving an "init!" message, receivers add the sender to their good-list. The next step is checking the agent view in which all the agents except ND1 find one or two conflicts in their agent-views. In both of the algorithms ND0 (priority = 4), ND3 (priority = 4), ND6 (priority = 3), ND1 (priority = 3) and ND4 (priority = 4) wait for ND5 (priority = 7) to mediate.

In APO, ND2 (priority = 4) and ND3 (priority = 4) have the same priority but, according to lexicographical ordering, ND2 waits for ND3. Also in MaxCAPO, as ND2 and ND3 have the same priority and the same number of conflicts (number of conflicts = 2), according to lexicographical ordering ND2 waits for ND3. So, in both of the algorithms, the first mediator is ND5. ND5 first tries to remove the conflict by simply changing its local variable. Fortunately, the sub-problem is locally solvable, so ND5 changes its color to Red and sends an "ok?" message to its neighbors. The result graph is shown in Figure 2 b).

From this point, APO and MaxCAPO choose different ways. In APO, ND3 (priority = 4) waits for ND4 (priority = 4) because of lexicographical ordering; but in MaxCAPO, ND4 (priority = 4) waits for ND3 (priority = 4), because ND4 has just one conflict and ND3 has two conflicts. Again, in both algorithms, ND6 (priority = 3) waits for ND2 (priority = 4). In APO, ND2 (priority = 4) waits for ND3 (priority = 4) according to lexicographical ordering and in MaxCAPO, as they have the same number of conflicts (2 conflicts) again according to lexicographical

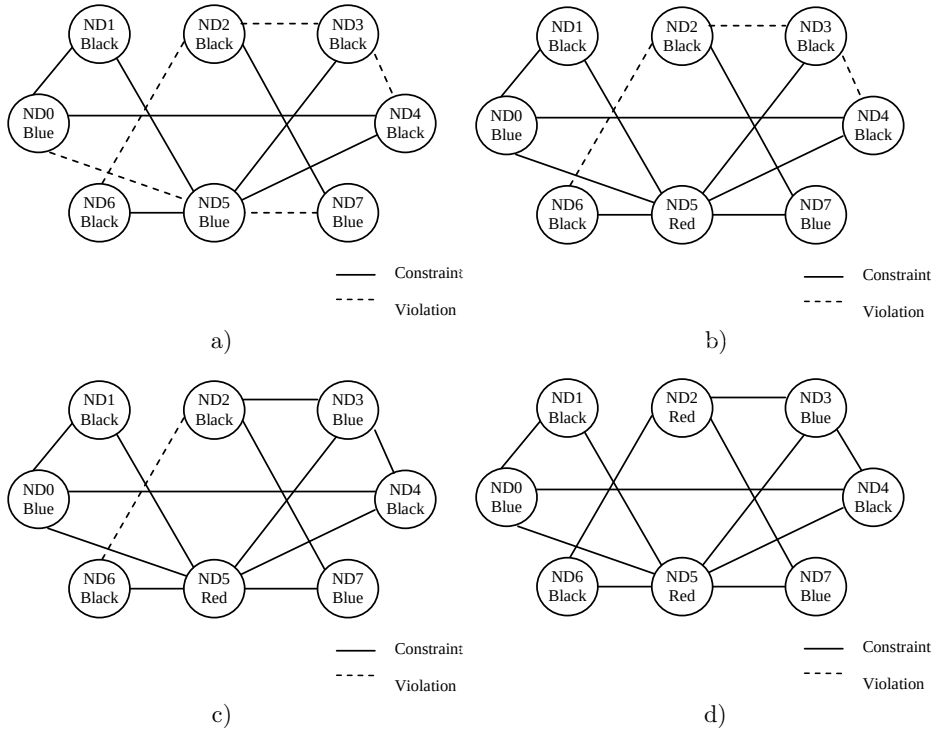


Fig. 2. An Example of a 3-coloring Problem with 8 Nodes and 12 Edges; a) Start, b) After ND5 mediates, c) After ND4 mediates in APO and ND3 mediates in MaxCAPO, d) Solved problem

ordering ND2 waits for ND3. So, in APO, ND4 will be the mediator, but in MaxCAPO ND3 will be the mediator. In MaxCAPO, ND3 finds that it can solve the sub-problem by changing its local value to Blue. Therefore it changes its value and sends an "ok?" message to its neighbors. On the other hand, in APO, ND4 can not solve the problem by changing its local variable, so it starts a mediation session by sending an "evaluate?" message to each of its neighbors. When each agent receives this message, it evaluates its domain elements and sends an "evaluate!" message to ND4 in response.

- ND0 sends: Red conflicts with ND5, Black conflicts with ND4 and ND1, Blue makes no conflict.
- ND3 sends: Black conflicts with ND4 and ND2, Red conflicts with ND5, Blue makes no conflict.
- ND5 sends: Black conflicts with ND4, ND3, ND1 and ND6, Blue conflicts with ND0 and ND7, Red makes no conflict.

While receiving all the "evaluate!" messages, ND4 chooses a solution that solves the sub-problem and also minimizes the number of outside conflicts with a branch and bound search. ND4 finds that the solution is to change ND0s color to Blue, not to change ND5s color, change ND3s color to Blue, and not to change its own color. ND4 informs its neighbors about their colors by sending an "accept!" message to them. By making these changes, the problem state will be what is shown in Figure 2 c).

Now, again, both APO and MaxCAPO algorithms are in the same step and follow solving the problem the same way. ND2 and ND6 find one conflict by checking their agent-views. ND6 (priority = 3) waits for ND2 (priority = 4), so ND2 takes the mediator role and solves the problem by changing its local value to Red, then it sends an "ok?" message to its neighbors. Now, all of the agents check their agent-view and find no conflict, so the problem is solved (Figure 2 d)).

Obviously MaxCAPO reduced the Runtime and the number of exchanged messages by choosing the best agent as mediator in step 2.

This heuristic mediator selection leads to reducing 15 messages and 3 cycles of time. This improvement is the result of selecting agents with the highest number of conflicts as mediators instead of choosing them in a random way.

4 MAX CONFLICT INVERSE APO (MAXCIAPO)

In this section, in addition to introducing IAPO algorithm, MaxCIAPO, a new expansion of IAPO, which uses a new strategy to select mediator agents, is proposed.

4.1 IAPO Overview

In APO, the mediator selection rule biased toward selecting larger mediation sessions instead of small ones; but according to the mathematical analyses reported in [1], choosing smaller mediation sessions can improve the performance significantly. Michael Benicsh et al. believe that the two most important sources of complexity that impact directly on APO complexity are the complexity of the mediation process and the complexity of the overlay process.

Mediation complexity refers to the complexity of a branch and bound search, which is accomplished to solve the centralized version of the sub-problem that involves all the participating agents in the mediation session. The overlay complexity refers to the complexity of putting the partial solutions next to each other and fitting them to each other.

By larger mediation sessions the complexity of the branch and bound search grow rapidly; by selecting smaller mediation sessions the mediation complexity will decrease. On the other hand, higher number of mediation sessions increases the overlay complexity. Benicsh et al. proved that according to mediation and overlay complexities, selecting smaller mediation sessions improves the performance.

IAPO is an extension of APO that follows this theory. Unlike the APO that assigns priorities proportional to the agents number of constraints, the IAPO assigns

them inversely proportional to the agents number of constraints. All the parts of the IAPO algorithm except the mediator selection part are the same as the APO algorithm.

4.2 MaxCIAPO Algorithm

Both the APO and IAPO algorithms assign the agents priorities according to the number of constraints, but none of them considers the impact of number of conflicts in mediator selection strategy. In the previous section the MaxCAPO algorithm proposed this algorithm changed the mediator selection method in conditions that agents had the same number of neighbors. In this section we want to use the same strategy on IAPO algorithm instead of APO. Although IAPO replaced the mediator selection strategy with a new one, also in this new strategy it frequently occurs that several agents have the same priority, because it frequently occurs that several agents have the same number of neighbors and the agents priority is inverse of the number of its neighbors. Like APO, in such a condition IAPO assigns priorities according to the lexicographical ordering of the agents names. In fact this is just a random way. MaxCIAPO replaces this random method with a heuristic one. Just like the method which was used in MaxCAPO, in MaxCIAPO in the conditions that agents have the same number of neighbors and of course the same priority numbers, the agents that have the highest number of conflicts with its neighbors will be selected. So, in MaxCIAPO, the priority is determined at first step inversely proportional to the agents number of constraints and at the second step proportional to the agents number of conflicts. By using this method, if several agents have the same number of constraints, the agent that has most conflicts and as a result the highest information of the sub-problem, is selected as the mediator and as the main decision maker.

The same reason that we mentioned in previous section about the advantages of MaxCAPO is also true about MaxCIAPO in addition that MaxCIAPO inherits the goodness of IAPO as well. So MaxCIAPO outperforms all the previously mentioned algorithms containing APO, MaxCAPO and IAPO.

The name of this algorithm, "MaxCIAPO", comes from "Maximum Conflicts Inverse APO (IAPO)", which means that this algorithm is an extension of IAPO (which itself is an extension of APO) that prefers to choose agents with the maximum number of conflicts instead of choosing them according to lexicographical order. It is obvious that the algorithm uses this rule just when the good-list sizes of two or more agents are the same. The other parts of the algorithm are similar to IAPO.

Figure 3 summarizes four different mediator selection strategies, which involve APO and IAPO strategies in addition to two new strategies which were introduced in this paper.

APO	P_i = | neighborhood (i) | If several agents have the same P_i then Mediator = the agent that have the highest lexicographical order

Max Conflict APO (MaxCAPO)	P_i = | neighborhood (i) | C_i = | conflict's number (i) |
	If several agents have the same P_i then
	Mediator = the agent with the most C_i
	If several agents have the same C_i then
	Mediator = the agent that have the highest lexicographical order

Inverse APO (IAPO)	$P_i = \frac{1}{\text{\| neighborhood (i) \|}}$ If several agents have the same P_i then Mediator = the agent that have the highest lexicographical order

Max Conflict Inverse APO (MaxCIAPO)	$P_i = \frac{1}{\text{\| neighborhood (i) \|}}$ C_i = | conflict's number (i) |
	If several agents have the same P_i then
	Mediator = the agent with the most C_i
	If several agents have the same C_i then
	Mediator = the agent that have the highest lexicographical order

Fig. 3. Summary of four different mediator selection strategies

5 EXPERIMENTAL EVALUATION

5.1 APO and MaxCAPO

5.1.1 Experimental Environment and Setup

The distributed 3-color graph coloring (D3GC) problem is selected as the test case of our experiments. The purpose of doing these experiments is comparing MaxCAPO and APO algorithms under different parameters, such as the number of "cycles", the number of "messages" and the number of "links". The last parameter shows the number of links generated in the "linking" step of the algorithm. Of course the least number of generated links is preferred. These parameters are selected because they can present computational and communicational complexities clearly.

For this purpose, 600 random graphs were generated. These graphs were created in various sizes: $n = 15, 30, 45, 60, 75$ and 90. All of these graphs are produced with medium density ($m = 2.3$), which means that the average number of constraints per variable is 2.3. Then, 10 different graphs with 10 different initial variable assignments

were created for each size. These graphs were generated by random seeds which were saved in special files and used for both MaxCAPO and APO algorithms. As a result, all the conditions of running these two algorithms were generated in the same way.

Farm simulator [6], which is a Java simulator, was used for running the algorithms to keep all conditions similar to the previous work conditions for fairness.

5.1.2 Experimental Results

Since exactly the same implementation environment and the same simulator are used, the results that are achieved of APO are almost the same as those reported in [10] by Mailler. This confirms that the improvement in MaxCAPO is only caused by entering the number of conflicts in mediator selection strategy. As can be seen in Figure 4, MaxCAPO shows considerable improvement in comparison with APO algorithm, and this improvement covers all numbers of "cycles", "messages" and "links" parameters. Another important point is that the results show that by increasing the problem size, the difference between APO and MaxCAPO is increased. In other words, as the problem becomes larger and harder, the effect of conflicts becomes more important.

Figure 5 shows the percentage of improvement of MaxCAPO in comparison with APO in various problem sizes. As can be seen in Fig. 5 a), MaxCAPO has an improvement of 6 % to 47 % in decreasing the number of massages sent. The percentage of Runtime improvement is shown in Figure 5 b), which is from 4 % to 34 % improvement in decreasing the number of cycles needed to solve the problem, and, finally, improvement in decreasing the number of generated links is presented in Figure 5 c), which is from 1 % to 11 %.

5.2 IAPO and MaxCIAPO

5.2.1 Experimental Environment and Setup

Again the distributed 3-color graph coloring (D3GC) problem is selected as the test case of our experiments. The purpose of doing these experiments is comparing MaxCIAPO and IAPO algorithms under different parameters, such as the number of "cycles", the number of "messages" and the number of "links". For this purpose, 400 random graphs were generated.

These graphs were created in various sizes: $n = 15, 30, 45$ and 60. All of these graphs are produced with medium density ($m = 2.7$), which means that the average number of constraints per variable is 2.7. The density of 2.7 is selected to adapt with the reported results about IAPO in previous articles.

5.2.2 Experimental Results

To show the improvement of MaxCIAPO over the APO, it is first necessary to know the improvement of IAPO over the APO, and next the improvement of MaxCIAPO

over the IAPO. In fact we should portion the problem into two separate parts. For the fist step the results which were reported by Benisch and Sadeh [1] are used. This is shown in Figure 6 [1]. In this research they generated 10 solvable D3GC problems for each pair of n and m, $n = 15, 30, 36, 45, 51, 60$ and $m = 2.0$ (low density) and $m = 2.7$ (high density) then for each of these problems they generated 10 different random strategy assignments.

For the second step, which is the comparison of IAPO and MaxCIAPO, 400 random graphs were generated with $m = 17$ and $n = 15, 30, 45, 60$. By choosing these values for $m = 17$ and $n = 15, 30, 45, 60$, the results would be comparable with the previously reported IAPO results.

The results show that IAPO outperforms APO over runtime parameter and also over the number of messages parameter. It shows favoring smaller mediation sessions instead of large ones, as IAPO does, helping mediators decrease the Branch and Bound search complexity by avoiding solving unnecessarily large problems.

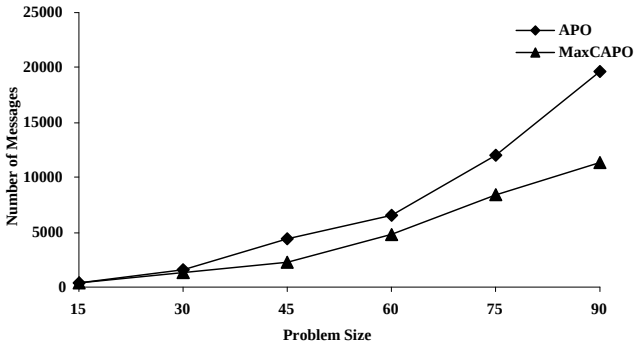
Now is the time to show the improvement of MaxCIAPO over IAPO. Figure 7 shows this reasonable improvement. This improvement covers all numbers of "Cycles", "Messages" and "Links" parameters. As this figure shows the percentage of improvement of MaxCIAPO over IAPO in decreasing the number of messages is from 5.5 % to 18.4 %. The percentage of runtime improvement is shown in Figure 7 b), which is from 7.1 % to 13.9 % improvement in decreasing the number of cycles needed to solve the problem; and, finally, improvement in decreasing the number of generated links is presented in Figure 7 c), which is from 1 % to 4.6 %.

Putting the results of these two parts which was shown in Figures 6 and 7 next to the each other reveals the great improvement of MaxCIAPO over APO. According to the results that were extracted from [1] and also to the experimental results that are proposed in this research Figure 8 can be generated.

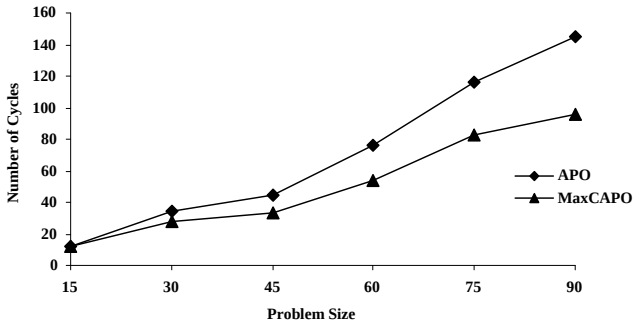
This figure shows a great decrease in the number of messages and also in the runtime of the MaxCIAPO. This shows that, by selecting mediators from among the agents that have the least number of constraints and the most number of conflicts, the APO performance improves reasonably.

6 CONCLUSIONS AND FUTURE WORK

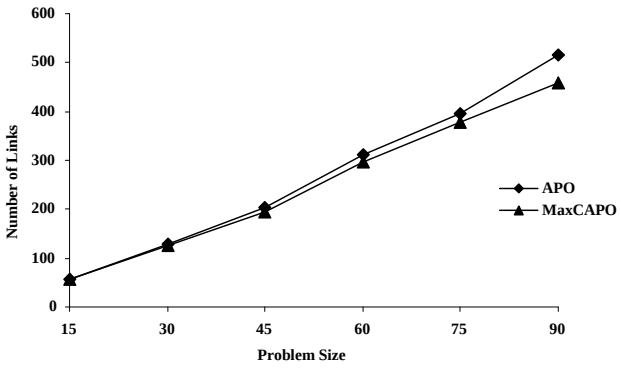
In this paper, two new extension algorithms of APO called MaxCAPO and MaxCIAPO have been proposed. The role of conflicts in solving DCSPs is shown by presenting these algorithms. The experimental results confirm that the mediators that have the highest number of conflicts can solve the problem faster and by a smaller number of messages and links. To show the completeness of this idea, hundreds of graphs were generated in a D3GC problem in various problem sizes. The averages of the results derived from various problem sizes were computed. These results showed that in all terms MaxCAPO and MaxCIAPO outperform APO, the previously most successful algorithm in DCSPs.



a)



b)



c)

Fig. 4. Messages, cycles, links needed to solve random solvable D3GC instances in APO and MaxCAPO; a) Number of message exchanged, b) Number of cycles, c) Number of links

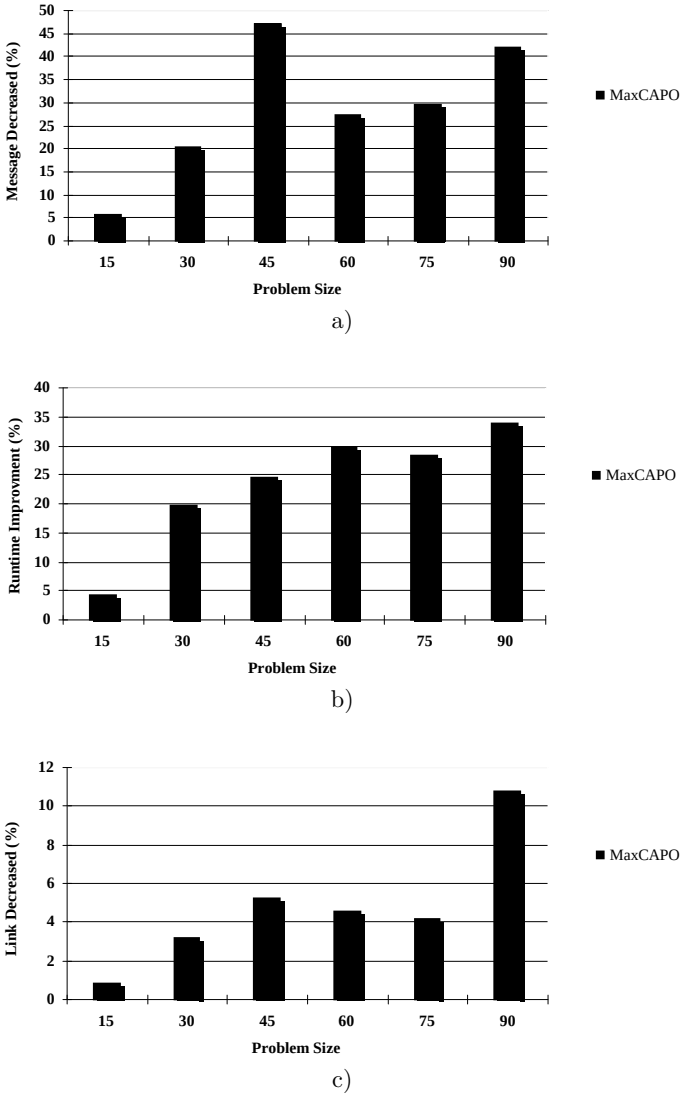
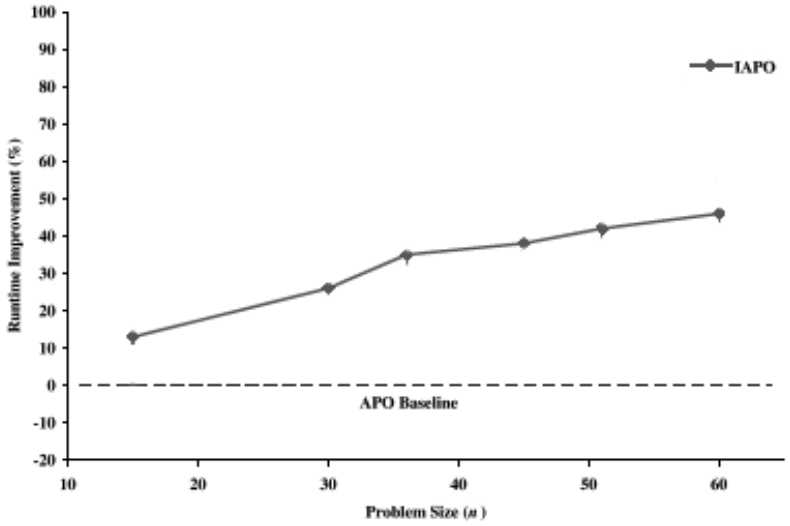
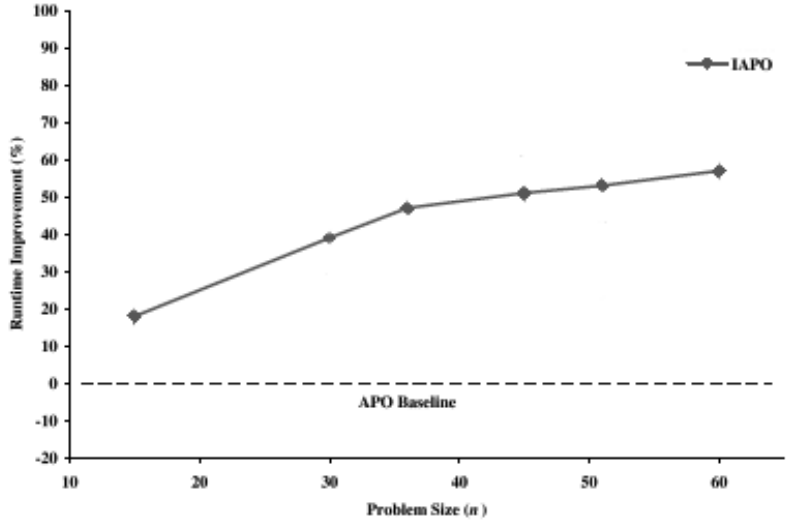


Fig. 5. Mean percentage of improvement of MaxCAPO in comparison with APO in solvable random instances; a) Number of message decreased, in percent, b) RunTime improvement, in percent, c) Number of links decreased, in percent



(a) Low Density Problems ($m = 2.0$)



(b) High Density Problems ($m = 2.7$)

Fig. 6. Running time needed to solve D3GC instances as mean percentage improvement over APO; a) Low Density Problems ($m = 2.0$), b) High Density Problems ($m = 2.7$)

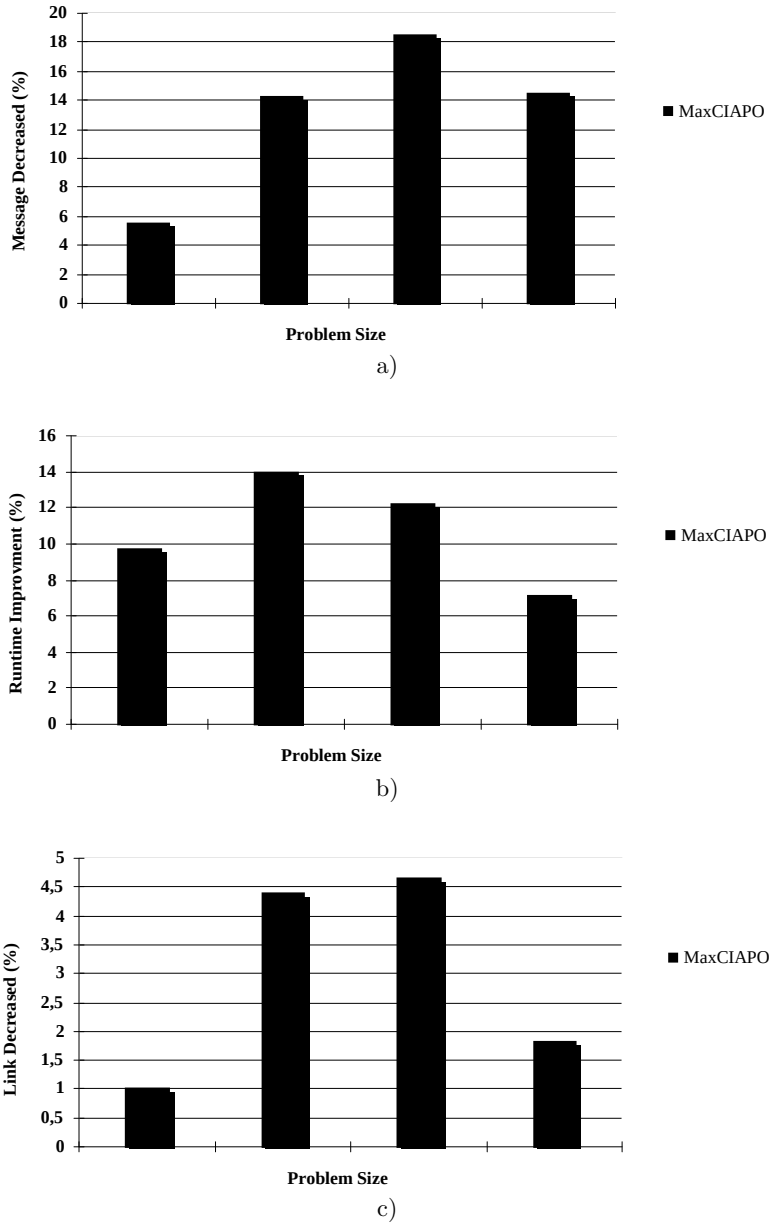


Fig. 7. Mean percentage of improvement of MaxCIAPO in comparison with IAPO in solvable random instances; a) Number of messages decreased, in percent, b) RunTime improvement, in percent, c) Number of links decreased, in percent

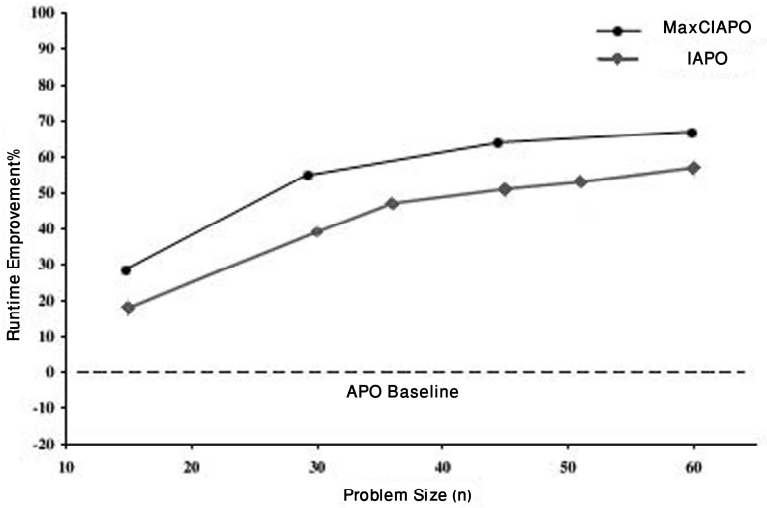


Fig. 8. Running time needed to solve D3GC instances as mean percentage improvement over APO and IAPO

Since DCSP covers a vast domain of problems and also APO has been the best known algorithm in this domain until now, MaxCAPO and MaxCIAPO can help solve many problems in the large domain of mediated cooperative problems by making considerable improvement in APO.

Employing other measurement units for measuring the computational complexity of MaxCAPO, such as non-concurrent consistent checks, which has recently been proposed by Meisels et al. [13], may be recommended for future works.

Using the strategy of choosing mediators with the highest number of conflicts in other examples such as random binary DCSPs and other domains such as SensorDCSP [4] would expand the domain of the strategy proposed.

REFERENCES

[1] Benicsh, M—Sadeh, N.: Effect of Mediator Selection Strategy for Distributed Constraint Satisfaction. In Workshop on Distributed Constraint Reasoning (DCR), 2005.

[2] Benicsh, M.—Sadeh, N.: Examining Distributed Constraint Satisfaction Problem (DCSP) Coordination Tradeoffs. In Proceedings of International Conference on Automated Agents and Multi-Agent Systems (AAMAS), 2006.

[3] Conry, S. E.—Kuwabara, K.—Lesser, V. R.—Meyer, R. A.: Multistage Negotiation for Distributed Constraint Satisfaction. International Journal of IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, 1991, No. 6, pp. 1462–1477.

[4] Fernandez, C.—Bejar, R.—Krishnamachari, B.—Gomes, C.—Selman, B.: Distributed Sensor Networks: A Multiagent Perspective. Chapter "Communication and Computation in Distributed CSP Algorithms", pp. 299–317, Kluwer Academic Publishers 2003.

[5] Freuder, E. C.—Wallace, R. J.: Partial Constraint Satisfaction. Artificial Intelligence, Vol. 58, 1992, No. 1–3, pp. 21–70.

[6] Horling, B.—Mailler, R.—Lesser, V.: Farm: A Scalable Environment for Multi-Agent Development and Evaluation. In Second International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2003).

[7] Huhns, M. N.—Bridgeland, D. M.: Multi-Agent Truth Maintenance. In Proceedings of International Journal of IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, 1991, No. 6, pp. 1437–1445.

[8] Mailler, R.: Comparing Two Approaches to Dynamic, Distributed Constraint Satisfaction. In Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), pp. 1049–1056.

[9] Mailler, R.: Solving Distributed CSPs Using Dynamic Partial Centralization Without Explicit Constraint Passing. In Second Workshop on the Challenges in the Coordination of Large Scale Multi-Agent Systems (LSMAS 2005).

[10] Mailler, R.—Lesser, V.: Asynchronous Partial Overlay: A New Algorithm for Solving Distributed Constraint Satisfaction Problems. Journal of Artificial Intelligence Research (JAIR), Vol. 25, 2006, pp. 529–576.

[11] Mailler, R.—Lesser, V.: Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), pp. 438–445.

[12] Mailler, R.—Lesser, V.: Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. In Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York, pp. 446–453.

[13] Meisels, A.—Kaplansky, E.—Razgon, I.—Zivan, R.: Comparing Performance of Distributed Constraints Processing Algorithms. In DCR Workshop at AAMAS '02.

[14] Modi, A. P. J.—Veloso, M.—Smith, S.—Cmradar, J. Oh.: A Personal Assistant Agent for Calendar Management. Agent Oriented Information Systems (AOIS) 2004.

[15] Sycara, K.—Roth, S. F.—Sadeh, N.—Fox, M. S.: Distributed Constrained Heuristic Search. International Journal of IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, 6, pp. 1446–1461.

[16] Yokoo, M.: Asynchronous Weak-Commitment Search for Solving Distributed Constraint Satisfaction Problems. In Proceedings of The First International Conference on Principles and Practice of Constraint Programming 1995, pp. 88–102.

[17] Yokoo, M.—Durfee, E. H.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. In Proceedings of 12th IEEE International Conference on Distributed Computing Systems, Vol. 1, 1992, pp. 614–621.

Samaneh Semnani Hosseini is a Ph. D. Student at faculty of Engineering, University of Isfahan since 2007 and now she is a visiting scholar in the University of the Waterloo (Fall 2009). In 2007 she received her M. Sc. degree in computer engineering with honors. Her Ph. D. thesis is focused on constraint satisfaction problem in the distributed form. Her other interests include constraint optimization problem, multi-agent systems, artificial intelligence and distributed and parallel computing. She has published some journal and conference papers and also has some teaching experiences.

Kamran Zamanifar is Assistant Professor at the University of Isfahan since 1996. He received his M. Sc. in electrical and electronic engineering from the Faculty of Engineering, University of Tehran, Iran. He also received his Ph. D. in computer science (parallel and distributed systems) from the School of Computer Studies, University of Leeds, England (1992–1996). He is a member of various scientific committees such as Management Board of Computer Society of Iran, Iranian Association of Electrical and Electronic Engineers and Annual International CSI Computer Conferences. He is also reviewer of many scientific journals. His main interests include parallel and distributed systems, concurrent systems and high performance computing. He has published two books and some journal and conferences papers.