Journal of Theoretical and Applied Information Technology





www.jatit.org

# USING A CONFLICT-BASED METHOD TO SOLVE DISTRIBUTED CONSTRAINT SATISFACTION PROBLEMS

**[1] SAMANEH HOSEINI SEMNANI, [2]KAMRAN ZAMANIFAR**

[1]Ph.D. Student, Dept. of Computer Engineering,, Faculty of Engineering, University of Isfahan, Isfahan Iran

[2]Assoc. Prof., Dept. of Computer Engineering,, Faculty of Engineering, University of Isfahan, Isfahan Iran

E-mail: s_hosseini@ eng.ui.ac.ir , zamanifar@eng.ui.ac.ir

**ABSTRACT**

A broad range of AI and multi-Agent problems fall in to the Distributed Constraint Satisfaction Problems category. Many of the problems in this domain are real-world problems. This fact makes DisCSPs an effective area of research. Considering all of the efforts that have recently been accomplished for solving these kinds of the problems, the most successful algorithm proposed is Asynchronous Partial Overlay (APO), which is a mediation-based algorithm. APO tries to solve the problem first by dividing the whole problem in to smaller portions and then solving these sub-problems by choosing some agents as mediators. This paper presents a new and effective strategy to select these mediators; moreover, it introduces two new expansion algorithms of APO that use this new strategy. These algorithms are called MaxCAPO and MaxCIAPO. The chief idea behind this strategy is that the number of mediators' conflicts (violated constraints) impacts directly on their performance. Experimental results show that choosing the agents which have the most number of conflicts as mediators not only leads to a considerable decrease in APO complexity, but also can decrease the complexity of the other extensions of the APO, such as IAPO algorithm. The results of using this conflict-based mediator selection strategy show a rapid and desirable improvement, in comparison with APO and IAPO, over various parameters such as the message and runtime complexities.

**Keywords:** *Distributed Constraint Satisfaction, APO, cooperative mediation, multi-agent, autonomous, Artificial Intelligence*

## 1. INTRODUCTION

All the problems in which the goal is finding suitable values to assign to distributed variables can be included in Distributed Constraint Satisfaction Problems (DisCSPs). Each one of these distributed variables is assigned to an autonomous agent. Each autonomous agent tries to assign suitable values to its variables and also to other agents' variables by exchanging some messages. A vast number of real-world and multi-agent problems can be classified, Such as distributed meeting scheduling [14], distributed resource allocation problems [3] and multi-agent truth maintenance [7] under this model. Due to the variety of problems in this domain, several algorithms have been proposed since 1991. 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 privacy; it means that the agents don't have sufficient information about the global effects of their local decisions. Although the first category algorithms satisfy privacy completely, the second group of algorithms outperforms them by revealing necessary information. The second group of the algorithms and specially APO inherit the speed of centralization while using the advantages of parallelism too. This new methodology, which is called cooperative mediation, is a method somewhere between centralized and distributed problem solving methods.

One of the most important parts of the APO algorithm is the coordinator selection part in which

the agents that detect some conflicts among themselves select the highest priority agent of the group as mediator. The mediator first tries to complete its information about the problem space by exchanging various messages with the other agents and then to solve the sub-problem by changing its local variables or starting a mediation session. After solving the sub-problem, the mediator recommends the new values to relevant agents. As they are autonomous, they can accept or refuse these values. After all the sub-problems are solved by mediators, the whole problem will be solved by juxtaposing these parts, just like putting the pieces of a puzzle next together. As it can be seen, the distributed problem is solved by solving various smaller sub-problems in a centralized manner. Although APO outperforms its previous algorithms, new research has been done recently that outperforms APO. Benish and Sadeh, who have done some studies in this domain, investigated different heuristics for mediator selection and found that by assigning higher priority to the agents with the smallest good-list, a substantial speedup from the solver will be seen.

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, it is very important how to choose the mediator to reach the best solution. Making a small change in mediator selection strategy can lead to a different solution. These all show the importance of choosing the best mediator selection strategy.

This paper presents a new and effective conflict-based strategy. In this strategy, more effective and powerful agents will be selected as mediators by developing a heuristic method based on the number of mediator's conflicts. The main idea behind this method is that the agents that have the most complete information about the sub-problem 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, after defining the DisCSP, a brief overview of the previous works preformed on DisCSP, specially the latest ones, is given; section 3, describes the new conflict-based mediator selection strategy clearly and also shows its usefulness by presenting MaxCAPO ( Max Conflict APO) algorithm which is an extension of APO and uses the conflict-based strategy. An overview of APO is also presented in this section. Next, section 4 presents an example of execution of APO and MaxCAPO to show the practical differences between them exactly. Section 5 first overviews IAPO algorithm, which is an extension of APO, briefly and then introduces the MaxCIAPO (Max Conflict Inverse APO) algorithm as an expansion of IAPO that uses the new mediator selection strategy. Section 6 presents the experimental setup and results of applying this mediator selection strategy to the APO and IAPO in addition to comparing APO and MaxCAPO results, IAPO and MaxCAPO results and finally APO, IAPO and MaxCIAPO results. At last, 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 DisCSP is introduced formally, and then the woks which have been performed to solve these problems, so far are presented briefly.

### 2.1 Distributed Constraint Satisfaction Definition

A DisCSP 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 DisCSP 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 DisCSP 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 communication 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 as 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.

### 2.2 Related Work

As CSP is the basis of the DisCSP, the proposed algorithms for solving CSP are the basis of the proposed algorithms for solving DisCSP. CSP's algorithm can be divided in to two groups called search algorithms and consistency algorithms. Also, search algorithms can be divided in to two important categories called backtracking algorithms and iterative algorithms.

Since the formulation of DisCSP, several algorithms with their advantages and disadvantages have been proposed. A number of them, such as Asynchronous Backtracking (ABT) [17] and Asynchronous Weak Commitment (AWC) [16], completely inherit their characteristics from centralized versions, but others, such as Asynchronous Partial Overlay (APO) [10], don't have any previous version to solve CSPs. Table 1 summarizes the most important proposed algorithms in DisCSP domains. Among these algorithms, APO is the latest and the most successful. Of course, after presenting APO, several extensions of it were also presented, such as IAPO that outperform it, but they cannot be considered as independent algorithms. ABT algorithm is the distributed form of the backtracking algorithm. As the backtracking algorithm tries to solve CSPs, the ABT tries to solve DisCSPs. 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 message's 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 agent's variables. Since each agent has a priority number, if an agent's 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 latter condition, the agent generates a new nogood message and sends it to the higher priority agent. By receiving this message the higher priority agent, 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. APO starts its work by sending the information of each agent to its neighbors and continues it by sending the highest priority agents as mediators to solve their local sub-problems in a centralized manner. One of the most important parts of the APO is the 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 mediator. The mediator is responsible to solve 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, 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, Benish and Sadeh improved APO by changing the mediator selection strategy. They confirmed that choosing agents with the smallest good-list size can result in substantial speedup from the solver [1]. Benish 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. CONFLICT-BASED MEDIATION SELECTION STRATEGY

In this section, in addition to introducing APO algorithm generally, the new conflict-based mediation selection strategy, which is the basis of the MaxCAPO is presented. This section continues

with proposing MaxCAPO, the new expansion of APO.

**Table 1 : Algorithms for solving DisCSP**

| | Single | multiple | partial |
|---|---|---|---|
| Backtracking | Asynchronous BT | Secure DisCSP | Asynchronous IR |
| Iterative Improvement | Distributed Breakout | | 1. Iterative DB<br>2. APO |
| Hybrid | Asynchronous WC (AWC) | 1. variable ordering AWC<br>2. agent-ordering AWC | |

### 3.1 APO Overview

Since APO is the basic form of MaxCAPO, it is presented it in a summarized form in Fig. 1. You can find the latest and the most accurate version of this algorithm in [10]. As can be seen in Fig. 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 agent's 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) assumes 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 through a link or indirectly through 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 through a direct link or through 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 all the appropriate response messages, 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 a 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 up with them. Therefore, the mediator supposes that it is somehow related to these agents. This work prevents the mediator from repeating this mistake in the future and in other sessions. This step of the algorithm is called the "linking" step. The number of generated links in the "linking" step is a parameter for measuring the advantages of the algorithm, and is used for this purpose in section 6.

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

The APO strategy for selecting mediators from among the agents that have various properties consisting in assigning a special number, called priority number, to each agent and then selecting the mediator according to the agent's priorities. These priority numbers are generated first,

according to the number of each agent's neighbors and then by continuing the algorithm and creating the good-list according to each agent's 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 leaving free the lower priority agents to be available for future mediation requests.

Although this strategy has several advantages, it has some weaknesses also. For example, by using this mediator selection strategy, it frequently occurs that several agents have the same good-list size. In such a condition, 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.

Obviously, it would be wonderful if we replaced this random way with a heuristic one. The heuristic method that is presented in this article, called the conflict-based method, pays more attention to the number of conflicts. We believe that if several agents have the same number of neighbors, the agent that has the most 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 to the conflicts. On the other hand, it has the highest information about the sub-problem restrictions in comparison with other agents. Now, the reason of calling this method the "Conflict-Based method" should be completely clear.

MaxCAPO is an expansion of APO that inherits the whole structure of the APO, but uses the conflict-based mediator selection strategy. So, in MaxCAPO, the priority is determined, at first step, according to the agent's good-list size, and secondly, 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 ensure that the agent with the highest knowledge acts as 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.

## 4. AN EXAMPLE OF EXECUTION OF APO AND MAXCAPO

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 = { x1 , . . . . . , xn } in which each element is associated with a set of possible colors D = { D1 , . . . , Dn}. Respectively, 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 = { R1 , . . . , Rm} in which Ri ( di , dj ) is true if the value of xi is "not equals" to the value of xj. 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 DisCSP.

Consider the problem in Fig. 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),

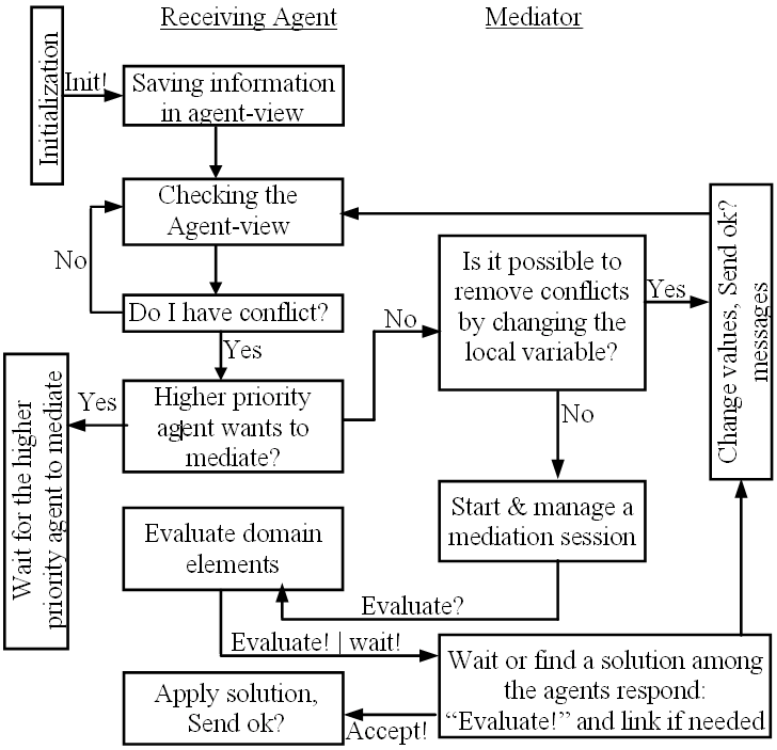



**Fig 1: APO algorithm**

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 Fig. 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 ordering, ND2 waits for ND3. As a result, 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,

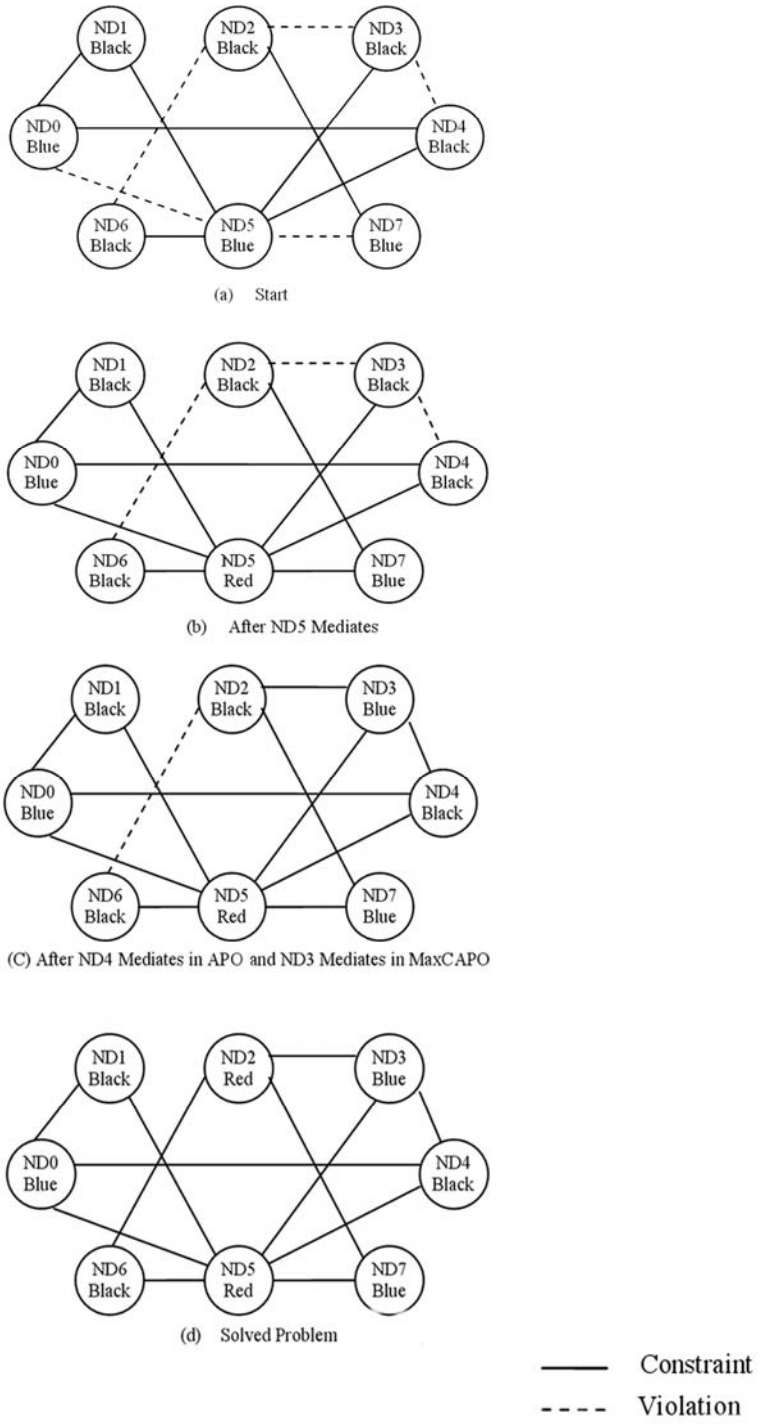



**Fig 2: Example of a 3-coloring Problem with 8 Nodes and 12 Edges.**

with a Branch and Bound search. ND4 finds that the solution is to change ND0's color to Blue, not to change ND5's color, change ND3's 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 if Fig. 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 (Fig. 2(d)).

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

The results of executing this example in Farm simulator [6] is presented in Fig. 3 and Fig. 4. Fig. 3 shows the results of the MaxCAPO execution and Fig. 4 is related to APO algorithm. In this picture the most important parameters including, "the number of messages", "the number of execution cycles (runtime)" and "the number of generated links" are shown.

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.

## 5. 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.

### 5.1 IAPO Overview

In APO, the mediator selection rule was biased toward selecting larger mediation sessions instead of smaller ones. But, according to the mathematical analyses reported in [1], choosing smaller mediation sessions can improve the performance significantly. Michael Benish 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 that of the overlay process.

Mediation complexity has to do with the complexity of a Branch and Bound search which is accomplished to solve the centralized version of the sub-problems that involve all the participating agents in the mediation session. And the overlay complexity is that of putting the partial solutions together and fitting them to each other.

By larger mediation sessions, the complexity of the Branch and Bound search grows rapidly. So, by selecting smaller mediation sessions the mediation complexity will decrease. On the other hand, the more the number of mediation sessions, the more will increase the overlay complexity. In fact, finding a tradeoff point between mediation and overlay complexities is the most important point in this domain. Benish 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 agent's number of constraints, the IAPO assigns them inversely proportional to the agent's number of constraints. All the parts of the IAPO algorithm except the mediator selection part are the same as the APO algorithm.

### 5.2 MaxCIAPO Algorithm

Both of the APO and IAPO algorithms assign the agent's priorities according to the number of constraints, but none of them consider 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. This section shows another practical usage of the conflict-based mediator selection strategy. As it can be seen, conflict-based strategy is a general strategy that can be employed in various algorithms in DisCSP domain. Of course, it can be more or less effective in different domains and also in different algorithms proposed in the domain of DisCSPs. For example, by choosing this method on the expansions of APO that outperform it, it would be possible to outperform APO and other expansions more powerfully.

As mentioned in previous section, IAPO is one of the best expansions of APO and, as a result, it is an appropriate choice for using the conflict-based strategy. Therefore, in this section the same strategy that was used over APO and converted it to MaxCAPO is used for improving the performance of IAPO. Although IAPO replaced the mediator selection strategy with a new one, in this new strategy, it frequently occurs that several agents

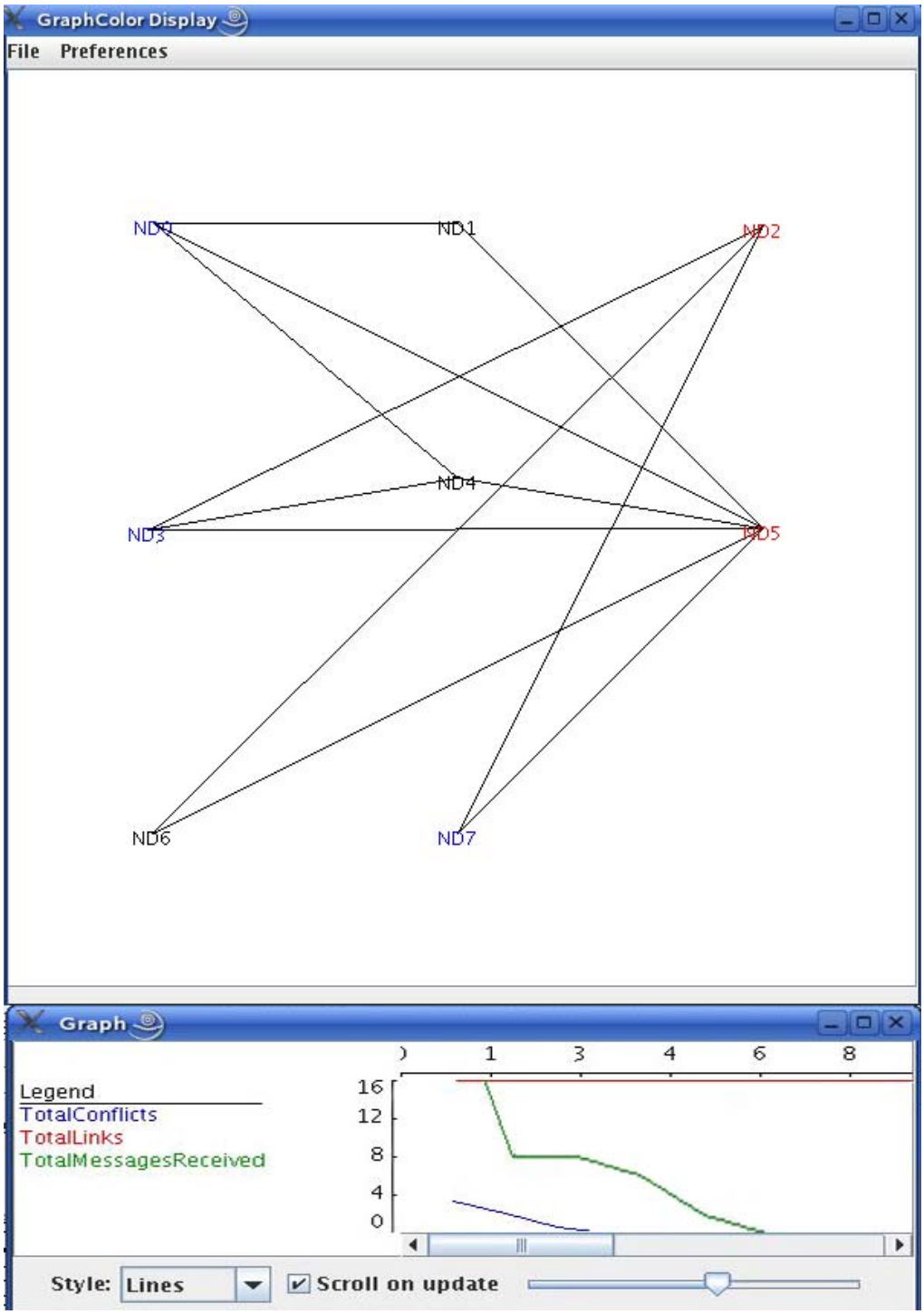



**Fig 3. The display of the execution of the example in Farm simulator for the MaxCAPO algorithm**

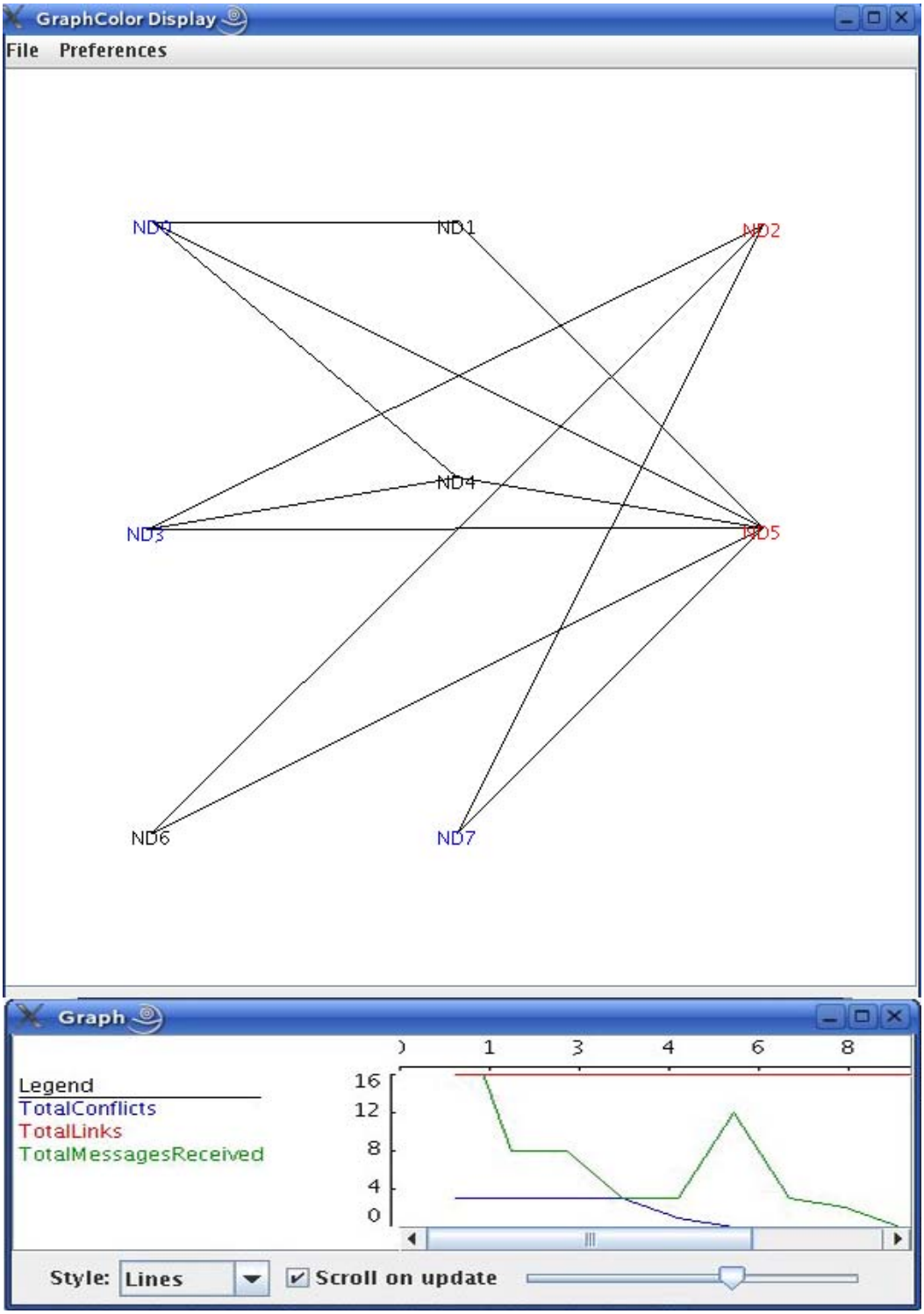



**Fig. 4. The display of the execution of the example in Farm simulator for the APO algorithm**

have the same priority, because it frequently occurs that several agents have the same number of neighbors and the agent's priority is in inverse proportion to 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 strategy. MaxCIAPO replaces this random strategy 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 agent that have the highest number of conflicts with its neighbors will be selected. Accordingly, in MaxCIAPO, the priority is determined, at first step, inversely proportional to the agent's number of constraints and, at the second step, proportional to the agent's number of conflicts. By using this method, if several agents have the same number of constraints, the agent that has the 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 advantages 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.

Fig. 5 summarizes four different mediator selection strategies, which involve APO and IAPO strategies in addition to the two new strategies which were introduced in this paper.

## 6. EXPERIMENTAL EVALUATION

### 6.1 APO and MaxCAPO

#### 6.1.1 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 that are 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 clearly present computational and communicational complexities.

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 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.

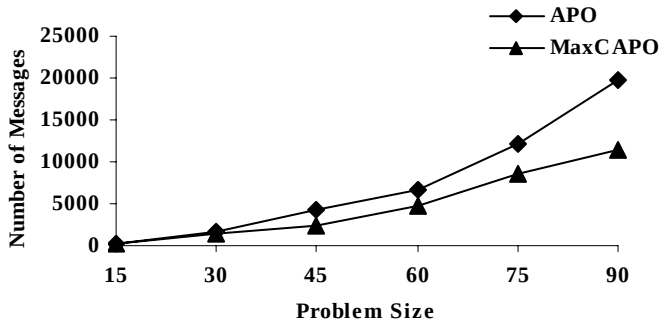
#### 6.1.2 Experimental Results

Since exactly the same implementation environment and the same simulator are used, the results that are achieved from 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 it can be seen in Fig. 6, MaxCAPO shows considerable improvement in comparison with APO algorithm, and this improvement covers all numbers of "cycles", "messages" and "links" parameters. As another important point, the results show that by increasing the size of the problem, 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.

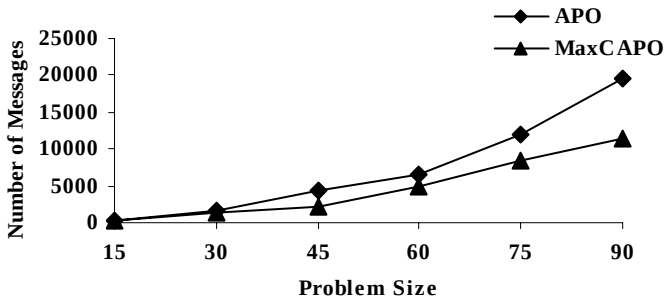
Fig. 7 and Table 2 show the percentage of improvement of MaxCAPO in comparison with APO in various problem sizes. As it can be seen in Fig. 7(a), MaxCAPO has an improvement of 5.7% to 47% in decreasing the number of massages sent. The percentage of Runtime improvement is shown in Fig. 7(b), which is an improvement from 4.2% to 34% in decreasing the number of cycles needed to solve the problem. And, finally, improvement in decreasing the number of generated links is presented in Fig. 7(c), which is from 1% to 11%.

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

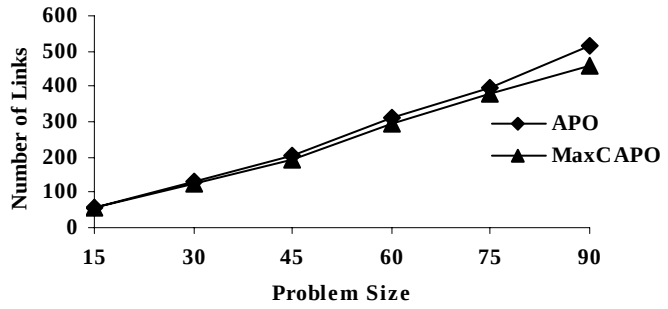
**Fig. 5. Summery of four different mediator selection strategies**




**(a) Number of Message Exchanged**




**(a) Number of Message Exchanged**




**(c) Number of Links**

**Fig. 6. Messages, Cylices, Links needed to solve random solvable D3GC instances in APO and MaxCAPO**

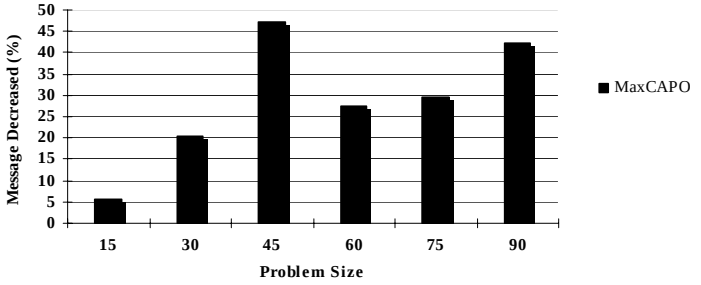
### 6.2 IAPO and MaxCIAPO

#### 6.2.1 Experimental Environment and Setup

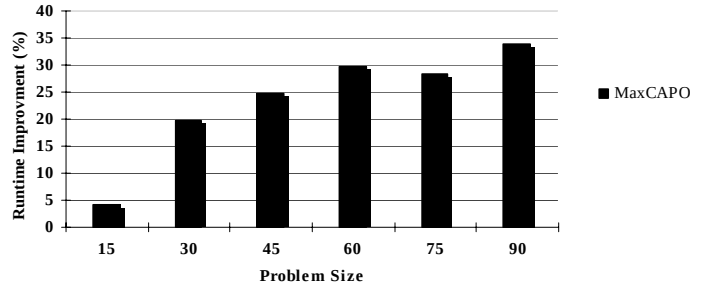
Again the distributed 3-color graph coloring (D3GC) problem is selected as the test case of our experiments. The purpose of doing these experiments is comparing MaxCIAPO and IAPO algorithms under different parameters, such as the number of "cycles", the number of "messages" and the number of "links".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

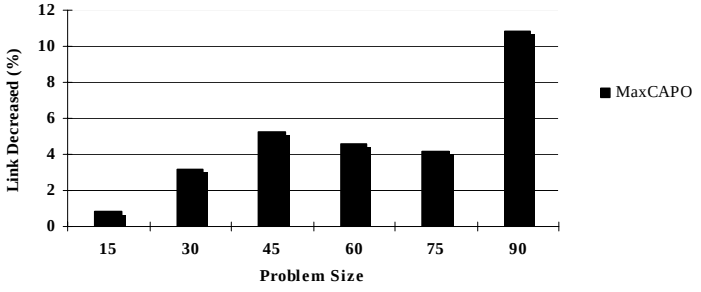
with medium density (m = 2.7), which means that the average number of constraints per variable is 2.7. A density of 2.7 is selected to adapt to the reported results about IAPO in previous articles.




(a) Number of Message Decreased, in percent




(b) RunTime Iprovment, in percent

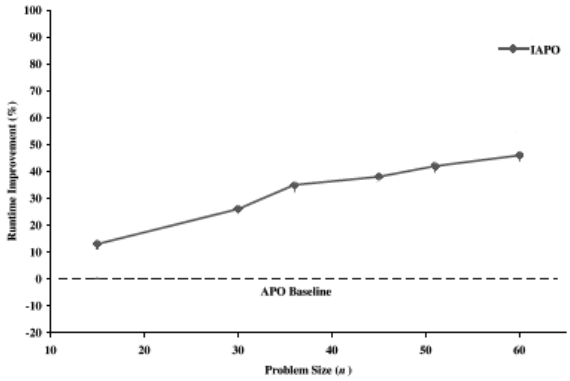



(c) Number of Links Decreased, in percent

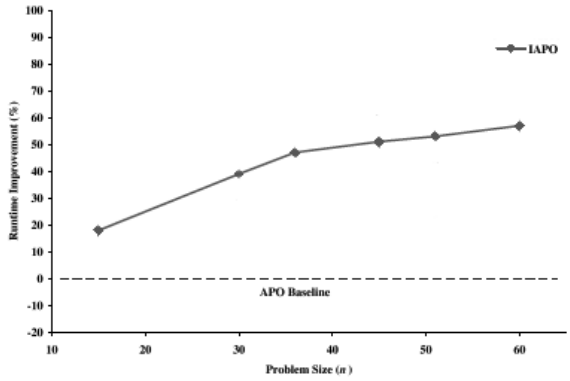
**Fig. 7. Mean Percentage of Improvement of MaxCAPO in Comparison With APO in solvable random Instances**

**Table II: the percentage of improvement of MaxCAPO over APO in the number of messages, cycles and links needed to solve satisfiable 3-coloring problems of various sizes.**

| Problem density | Problem size | Number of messages | Number of cycles | Number of links |
|---|---|---|---|---|
| M=2.3 | 15 | 5.705435 | 4.18679 | 0.84583 |
| | 30 | 20.2579 | 19.86022 | 3.179994 |
| | 45 | 47.01683 | 24.58352 | 5.213825 |
| | 60 | 27.50309 | 29.83555 | 4.584084 |
| | 75 | 29.70646 | 28.23123 | 4.143269 |
| | 90 | 42.09607 | 33.80529 | 10.81579 |




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




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

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

### 6.2.2 Experimental Results

To show the improvement of MaxCIAPO over the APO, first, it is 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 Fig. 8 [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 and n, the results would be comparable with the previous reported IAPO results.

The results show that IAPO outperforms APO over runtime parameter and also over the number of messages parameter. It shows that favoring smaller mediation sessions instead of large ones, as IAPO

does, helps 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. Fig. 9 and Table. 3 show 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 Fig. 9(b), which is an improvement from 7.1% to 13.9% in decreasing the number of cycles needed to solve the problem. And, finally, improvement in decreasing the number of generated links is presented in Fig. 9(c), which is from 1% to 4.6%.

Putting the results of these two parts which were shown in Fig. 8 and Fig. 9 next to each other reveals the great improvement of MaxCIAPO over APO. According to the results that were extracted from [1] and also the experimental results that are proposed in this research, Fig.10 can be generated. This figure shows a great decrease in the number of messages and 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.

## 7. 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 DisCSPs is shown by presenting these algorithms. The experimental results confirm that the mediators having the highest number of conflicts can solve the problem faster and using 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 DisCSPs.

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

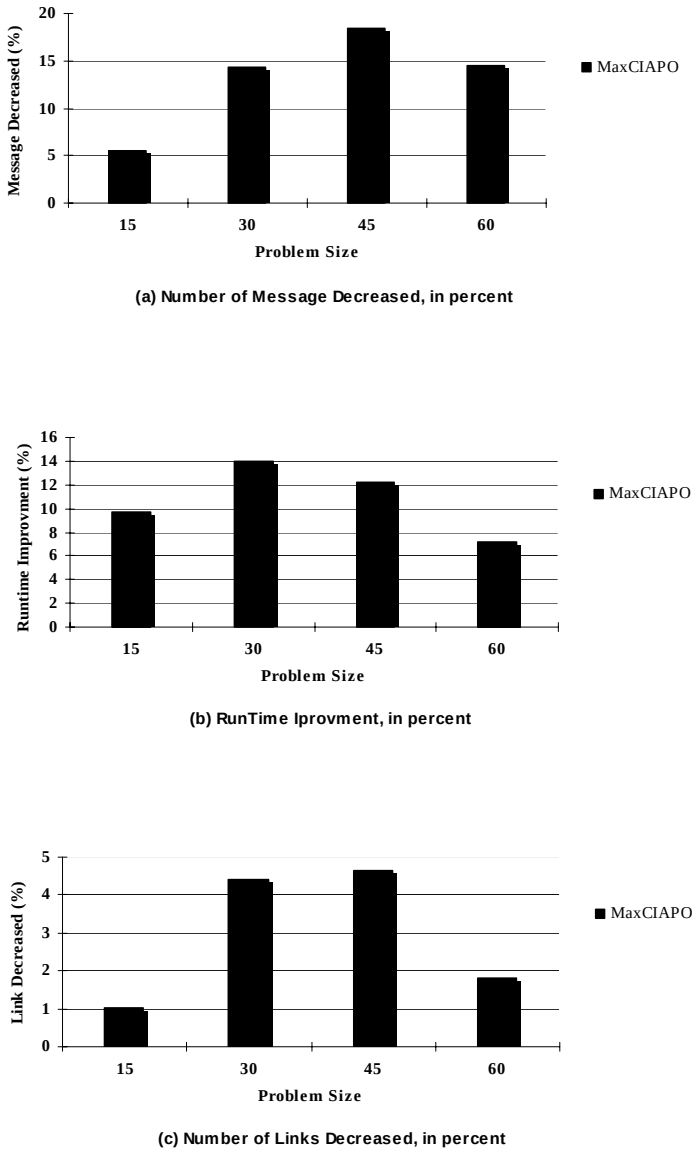



**Fig. 9. Mean Percentage of Improvement of MaxCIAPO in Comparison With IAPO in solvable random Instances**

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 DisCSPs and other domains, such as SensorDCSP [4] would expand the domain of the strategy proposed.

**Table III: the percentage of improvement of MaxCIAPO over IAPO in the number of messages, cycles and links needed to solve satisfiable 3-coloring problems of various sizes.**

| Problem density | Problem size | Number of messages | Number of cycles | Number of links |
|---|---|---|---|---|
| M=2.3 | 15 | 5.509183 | 9.66662 | 1.025635 |
| | 30 | 14.30943 | 13.95454 | 4.400992 |
| | 45 | 18.47375 | 12.24738 | 4.656425 |
| | 60 | 14.44901 | 7.12881 | 1.811559 |
| | 75 | 5.509183 | 9.66662 | 1.025635 |
| | 90 | 14.30943 | 13.95454 | 4.400992 |

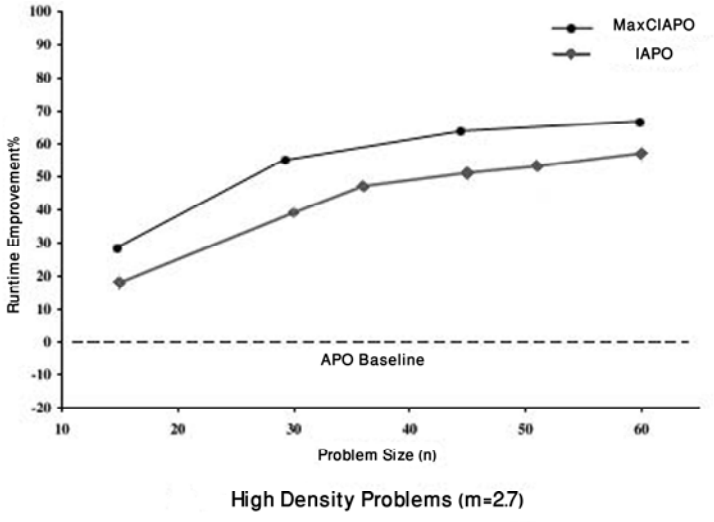



**Fig. 10. Running time needed to solve D3GC instances as mean percentage improvement over APO and IAPO**

**REFRENCES:**

[1] M. Benish & N. Sadeh (2005). Effect of mediator selection strategy for distributed constraint satisfaction. In Workshop on Distributed Constraint Reasoning (DCR).

[2] M. Benish & N. Sadeh. (2006). Examining distributed constraint satisfaction problem (DCSP) coordination tradeoffs. In Proceedings of International Conference on Automated Agents and Multi-Agent Systems (AAMAS).

[3] S. E. Conry, K. Kuwabara, V. R. Lesser & R. A. Meyer (1991). Multistage negotiation for distributed constraint satisfaction. International Journal of IEEE Transactions on Systems, Man and Cybernetics 21(6),1462–1477.

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

[5] E. C. Freuder & R. J. Wallace, (1992). Partial constraint satisfaction. Artificial Intelligence,58 (1–3), 21–70.

[6] B. Horling, R. Mailler & V. Lesser (2003). 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] M. N. Huhns & D. M. Bridgeland (1991). Multi-agent truth maintenance. In Proceedings of

International Journal of IEEE Transactions on Systems, Man and Cybernetics 21(6),1437–1445.

[8] R. Mailler, (2005). Comparing two approaches to dynamic, distributed constraint satisfaction. In Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), 1049–1056.

[9] R. Mailler, (2005). 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] R. Mailler & V. Lesser, (2006). Asynchronous partial overlay: a new algorithm for solving distributed constraint satisfaction problems. Journal of Artificial Intelligence Research (JAIR) 25, 529–576.

[11] R. Mailler & V. Lesser, (2004). Solving distributed constraint optimization problems using cooperative mediation. In Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), 438–445.

[12] R. Mailler & V. Lesser (2004). 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, 446–453.

[13] A. Meisels, E. Kaplansky, I.. Razgon, & R. Zivan, (2002). Comparing performance of distributed constraints processing algorithms. In DCR Workshop at AAMAS'02.

[14] A P. J. Modi, M. Veloso, S. Smith, & J. Oh. Cmradar, (2004). A personal assistant agent for calendar management. Agent Oriented Information Systems (AOIS).

[15] K. Sycara, S. F. Roth, N. Sadeh & Fox, M. S. (1991). Distributed constrained heuristic search. International Journal of IEEE Transactions on Systems, Man and Cybernetics, 21(6), 1446–1461.

[16] M. Yokoo, (1995). Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In Proceedings of The First International Conference on Principles and Practice of Constraint Programming, 88–102.

[17] M. Yokoo & E. H. Durfee, (1992). Distributed constraint Satisfaction for formalizing distributed problem solving. In Proceedings of 12th IEEE International Conference on Distributed Computing Systems, 1, 614–621.