Distributed Meeting Schedule through Cooperative Mediation: analysing OptAPO' s performance in a complex scenario

Paulo R. Ferreira Jr. and Ana L. C. Bazzan

Instituto de Informática, UFRGS
Caixa Postal 15064
91.501-970 Porto Alegre, RS, Brazil
phone +55 51 3316 6823
{prferreiraj, bazzan}@inf.ufrgs.br

Abstract. In multiagent systems, Distributed Constraint Optimization Problem (DCOP) has been used as a formalism to model a wide range of agents' coordination issues. An approach based on cooperative mediation was recently proposed as a new way to find the optimal solution to DCOP. An important question about any solution to DCOP is whether it is fast enough to be applied in real-world applications, such as Distributed Meeting Scheduling problems. Here, we map the DMS as a DCOP, use cooperative mediation, and compare the performance of this approach with the results achieved by another algorithms for DCOP. Given the time complexity of the complete solutions, we propose a modified approach for the cooperative mediation, in which the idea is to trade the completeness of the search mechanism for the performance of a heuristic search, which yields a good solution in a feasible time.

1 Introduction

The multiagent paradigm has coordination as a central issue. Coordination is a process in which agents engage to ensure that a community of individual agents acts in a coherent manner; when distributed agents work towards a common goal they should act as an unit coordinating interdependent actions, minimizing redundant efforts, sharing resources, etc. Research on coordination aspects has motivated many approaches and methods. Despite these many possibilities, no one is universally accepted as a complete solution. Thus, the literature presents some classical problems in agents' coordination such as distributed resource allocation problems, distributed scheduling problems, distributed planning problems, and so forth.

Distributed Constraint Optimization Problem (DCOP) is a formalism to model a wide range of the classical problems mentioned above. It is a distributed version of constraint optimization problems, which derived from constraint satisfaction problems. Differently from COP, in DCOP collaborative

agents must find solutions over a distributed set of constraints. A DCOP consists of n variables $V = \{x_1, x_2, ..., x_n\}$ that can assume values from a discrete domain $D_1, D_2, ..., D_n$ respectively. Each variable is assigned to one agent that has the control over its value. The goal of the agents is to choose values for the variables to optimize a global objective function. This function is described as the sum over a set of valued constraints related to pairs of variables. Thus, for a pair of variables x_i, x_j, there is a cost function defined as $f_{ij} : D_i \times D_j \rightarrow N$. DCOP generalizes the Distributed Constrains Satisfaction Problem (DisCSP) [8], which has a limited power of representation since every constraint is required to be boolean. This requirement is inadequate to represent many real-world problems where several degrees of quality or cost are necessary. Besides, real problems are often over-constrained, meaning that is impossible to satisfy all constraints.

In [6] an asynchronous complete method for distributed constraint optimization (called Adopt) is proposed to find the optimal solution for problems formalized as DCOP. Adopt provides quality/optimality guarantees on system performance and asynchrony on communication among agents. Besides, Adopt is known as the most efficient algorithm for DCOP [4]. Modi et al. discuss the main differences between Adopt and previous approaches, showing its achievements and limitations.

Another approach, this time based on cooperative mediation, was recently proposed by Mailler and Lesser [5] as a new way to find the optimal solution to DCOP. Cooperative mediation is a partial centralization technique and it is the basis of the *optimal asynchronous partial overlay* (OptAPO) algorithm. OptAPO is also a complete, distributed algorithm to solve DCOP. Furthermore, the authors show that OptAPO performs better than Adopt in an abstract problem of graph coloring that is an instance of the MaxSAT problem.

An important question about all solutions to DCOP is whether the proposed algorithms are fast enough to be applied in real-world applications. An important question here is whether the number and size of exchanged messages makes the approach feasible. In distributed approaches, the communication among agents usually poses demands that can cause network overload. Is the total time consumed acceptable in these situations? Real-world problems usually mean that the planning (for coordination) and action should be treated as quickly as possible. Most of the proposed approaches yields good results in simple scenarios, but there is a lack of analysis in complex real-world ones.

One of these scenarios is the Distributed Meeting Schedule problem (DMS) [7]. In a DMS, a group of persons wish to attend several meetings. The attendees try to optimize their calendars according to personal preferences maintaining the privacy of their information. Each meeting is subject to many constraints. The negotiation proxies to produce a schedule can incur in high communication costs and unacceptable time.

In [4], a DMS is mapped to a DCOP using a systematic reusable framework called Distribute Multi-Event Scheduling (DiMES). Agents' goal is to generate a coordinate schedule for the execution of joint activities or resource usage in a multiple-events scenario. Besides presenting DiMES, the authors have also

tested the efficiency of Adopt in some real-world problems mapped as DCOP using DiMES, and presented two heuristics to improve Adopt's performance. They show that the convergence time of Adopt in the tested scenarios was one hundred times higher than expected, illustrating the existence of a significant difference in Adopt's performance between simple abstract and complex real-world problems. The presented heuristics reduce the convergence time to the expected values and could potentially make Adopt able to deal with complex problems.

In the present paper we discuss the difficulties of applying the cooperative mediation (OptAPO) algorithm in a real-world problem. To illustrate this, we use the Distributed Meeting Scheduling problem mapped as a DCOP using the DiMES framework. The goal here is twofold. First, to check how OptAPO handles a real-world DMS problem, given that the evaluation of OptAPO was initially based on an abstract scenario. We compare its performance with the results achieved by the Adopt, with and without heuristics. Second, we propose to change OptAPO's centralized search mechanism. The idea is to trade the completeness of search (which is achieved via branch-and-bound) for the performance of a heuristic search (using a genetic algorithm). The motivation behind this trade-off is that in real-world applications such as meeting scheduling, to have a good solution in a short time is better than achieving the optimal solution in a long time.

In order to show these points, in Section 2 we summarize the cooperative mediation approach and explain how the OptAPO works. In Section 3 we describe the DMS problem and its mapping to a DCOP using DiMES. In section 4 we discuss the OptAPO's performance and show the results, comparing them to those achieved by Adopt and a centralized approach based on branch-and-bound (B&B). Given the performances, also in Section 4 we propose to replace the B&B by a genetic algorithm (GA). Finally, in Section 5 we conclude giving further directions for this work.

2 Cooperative Mediation and the OptAPO

The optimal asynchronous partial overlay (OptAPO) [5] is a cooperative mediation based DCOP protocol. As said, OptAPO (as well as Adopt) is a complete method, meaning that its final solution is the optimal one. The algorithm allows the agents to extend the context they use for local decision-making to a *relationship graph*. Within this graph, one of the agents has to act as a mediator, computing a solution for this extended context and recommending values for the variables associated with the agents involved in the mediation session. This is possible because agents construct a *good_list* – which holds the names of agents that have direct or indirect relationship to the list owner – and an *agent_view* – to hold the names, values, domains, and functional relationships of related agents.

During the problem-solving process, each agent tries to improve the value of its subproblem (the one it can solve within its relationship graph). The priority

to take the mediation role is be given to the agent with more information about the problem. A connected graph models the DCOP where each node is an agent (plus its values), and the links are the problem constraints. Each constraint or functional relationship has an associated cost. The algorithm has three stages: initialization, checking the agent view, and mediation. Details of these stages can be found in [5]; we give here a brief description. During the initialization, the agent sets its variables: current value (d_i), variable's name (x_i), priority (p_i), domain (D_i), functional relationships (C_i), good_list and agent_view.

The agent's goal is to improve the solution for its subproblem (represented by F_i). Thus, during the second stage, the agent_view is used to calculate the current cost F_i within the relationship subgraph given by i's good_list. If $F_i > F_i^*$ (F_i^* being the optimal value of the subsystem), than agent i conducts either a passive or an active mediation session, after which the value of F_i^* is recomputed. Agent i sets a passive mediation if its priority to mediate is lower than the priority of another agent in the subsystem; otherwise it sets a temporary mediation flag ($m_i^{'}$) to active. If an active mediation flag is on, the agent can actually mediate only if there is no other agent with a higher priority to mediate and with an active mediation flag. The agent tests if a change in its local value would cause a local cost to reach the optimal cost. If it does, then the agent changes its value and does not start the mediation process. If the agent has a passive mediation flag, it starts a passive mediation process.

In the mediation stage, agents receiving a mediation request either evaluate it or send a wait message. Evaluation means looking at each of the domain elements, labelling them with the names of the agents which share *functional relationships* with cost $f_i > f_i^*$, and returning these in a message. The mediator conducts a branch-and-bound (B&B) search to minimize the cost of the subproblem in its good_list, as well as the costs for agents outside the mediation session.

In [5] the OptAPO algorithm was applied to the MaxSAT 3-coloring problem with assignments for different number of variables (agents): 8, 12, 16, etc. Two series of tests were run with under- and over-constrained instances of the problem. These experiments compute the total number of messages, cycles, and time consumed to achieve the solution (measured in seconds using a standard PC configuration). Two conclusions of this evaluation should be pointed out here:

- OptAPO outperforms Adopt in terms of cycles (a round of message changing among agents), messages and runtime;
- and the OptAPO runtime is not a byproduct of the centralized search (B&B).

In Section 4 we show that in a more complex scenario (DMS problem) these conclusions could not be observed.

3 Distributed Meeting Scheduling as a DCOP

In Distributed Meeting Scheduling (DMS) a group of persons wishes to attend several meetings that should be scheduled in a distributed fashion. The schedule

is built interactively by a cooperative network of decision-makers. When doing this within a multiagent system, each agent normally has knowledge only about the meetings it participates and its preferences concerning the schedules. The agents must negotiate to build a consistent global schedule that fits the attendees agenda and respects individual preferences. Additionally, in real-world scenarios of DMS, the attendees try to optimize their agendas according to personal preferences maintaining the privacy of their information. As said, all this negotiation process may produce high communication cost and take an unacceptable time.

In DiMES, a DMS problem is formalized as follows:

- the group of attendees (agents) are represented as a resource set $\mathcal{R} = \{\mathcal{R}_\infty, ..., \mathcal{R}_\backslash\}$ of cardinality N where R_n refers to the n-th resource (attendee);
- the agenda is represented by a fractionated time interval. Let $T \in \mathbf{N}$ be a natural number and Δ be a length such that $T \cdot \Delta = T_{latest} - T_{earliest}$. The possible time intervals to schedule the meetings (time domain) are represented by the set $\mathcal{T} = \{\infty, ..., \mathcal{T}\}$ of cardinality T where $t \in \mathcal{T}$ refers to the time interval $[T_{earliest} + (t-1)\Delta, T_{earliest} + t\Delta]$;
- the meetings are represented as an event set $\mathcal{E} = \{\mathcal{E}^\infty, ..., \mathcal{E}^\|\}$ of cardinality K where E^k refers to the k-th event. The k-th event E^k is characterized as the triple $(A^k; L^k; V^k)$ where $A^k \subset \mathcal{R}$ is the subset of resources that are required by the event, $L^k \in \mathcal{T}$ is the number of contiguous time slots necessary to schedule the event, and V^k is the vector of preferences of each resource to each event as described next;
- the preferences of an attendee is represented by a vector V^k whose length is the cardinality of A^k. If $R_n \in A^k$, then V_n^k is an element of V^k denoting the value per time slot of the n-th resource to schedule event (meeting) k. The n-th resource also has a value $V_n^0(t) : \mathcal{T} \rightarrow \mathbf{R}^+$ to keep the time slot t free of meetings.

In summary, the DMS problem is about how to distribute the meetings through the agendas' time slots in order to maximize the attendees preferences. Let us define a schedule S as a mapping from the event set to the time domain where $S(E^k) \subset \mathcal{T}$ denotes the time slots committed for event k. All resources in A^k must agree to allocate time slots $S(E^k)$ to event E^k. So, formally the problem, which is *NP*-hard, is: $max_s(\sum_{k=1}^{K} \sum_{n \in A^k} \sum_{t \in S(E^k)} (V_n^k - V_n^0(t)))$ such that $S(E^{k_1}) \cup S(E^{k_2}) = \emptyset \ \forall k_1, k_2 \in \{1, ..., K\}$, $k_1 \neq k_2$, $A^{k_1} \cap A^{k_2} = \emptyset$.

In [4], it is shown how to convert a DMS problem to a DCOP in which the goal is to optimize a global objective function. This function is described as the sum over a set of valued constraints related to pairs of variables. Usually, a DCOP is represented by a graph, where nodes represent the set of variables and the edges represent the utility function determined by the values of the adjacent nodes. For each edge $e(i,j) \in E$ there is a function $f_{ij}(x_i, x_j) : D_i \times D_j \rightarrow \mathbf{R}$. To solve the problem, one assignment $a^* \in A = D_1 \times ... \times D_N$ must be chosen, such that $a^* = argmax_{a \in A} \sum_{(i,j) \in E} f_{ij}(x_i = a_i, x_j = a_j)$.

According to Maheswaran et al. there are three possible approaches for the DMS–DCOP mapping:

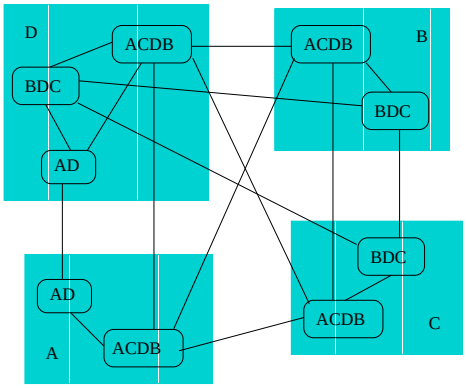

Fig. 1. DCOP graph using PEAV approach

- time slots as variables (TSAV): in the TSAV each variable $x_n(t)$ represents a time slot (t) of a resource (n), which means $T \times N$ variables. The domain of this variables are the set of events, and all variables having the same resource belongs to the same agent. This approach leads to a very dense graph;
- events as variables (EAV): in the EAV each variable represents the start time x^k of an event E^k. The domain of these variables is the set of time slots which is early enough so that the event can be accommodated. Each variable is assigned to one of the agents that attends the event. This approach results in a simple graph. Furthermore, the agents make decisions about multiple resources. To make this decision, the information about preferences must be shared among agents. In real-world scenarios, where an agent represents a resource (attendee), this approach is not realistic due to the issue of lack of privacy;
- private events as variables (PEAV): the PEAV is an alternative to tackle the privacy problem of EAV. The main idea is the same of EAV (variables representing events). However, the agents make decisions only about the events they participate. Let us consider a set of variables $X^k = \{x_n^k : n \in A^k\}$ where $X^k \in \{0, 1, ..., T - L^k + 1\}$ denotes starting time for event E^k where the resource R_n takes part. The DCOP is constructed with the set of variables $X = \cup_{k=1}^{K} X^k$. Each variable $X_n \in X$ that represents the n-th resource belongs to the same agent. These variables are fully connected (intra-agent links). The constraint utility function was designed to compute the values of internal links so as to maintain the privacy. There are inter-agents links connecting all participants of each event.

We are interested in representing DMS problems as close as possible to reality. For this reason we prefer the PEAV approach. Next we show the constraint utility function of PEAV and a conversion example. For details of each approach, including the variable sets, the utility functions, and the proofs of congruency, please refer to [4].

The resource constraint utility function used in the PEAV approach between the variables $x_{n_1}^{k_1}$ and $x_{n_2}^{k_2}$, when $x_{n_1}^{k_1} = t_1$ and $x_{n_2}^{k_2} = t_2$, is given by $f(n_1, k_1, t_1; n_2, k_2, t_2)$, where:

$$\begin{aligned} f(n_1, k_1, t_1; n_2, k_2, t_2) = & -MI_{\{n_1 \neq n_2\}} I_{\{k_1 = k_2\}} I_{\{t_1 \neq t_2\}} \\ & + I_{\{n_1 = n_2\}} I_{\{k_1 \neq k_2\}} f_{intra}(n_1, k_1, t_1; k_2, t_2) \end{aligned} \quad (1)$$

where

$$f_{intra}(n_1, k_1, t_1; k_2, t_2) = \begin{cases} -M & t_1 \neq 0, t_2 \neq 0, t_1 \leq t_2 \leq t_1 + L^k - 1, \\ -M & t_1 \neq 0, t_2 \neq 0, t_2 \leq t_1 \leq t_2 + L^k - 1, \\ g(n, k_1, t_1; k_2, t_2) & \textit{otherwise} \end{cases}$$

and

$$g(n, k_1, t_1; k_2, t_2) = \frac{1}{|X_n| - 1}(Z_n^{k_1}(t_1) + Z_n^{k_2}(t_2))$$

where

$$Z_n^{k_i}(t_i) = \sum_{l=1}^{L^{k_i}} (V_n^{k_i} - V_n^0(t_i + l + 1)) I_{\{t_i \neq 0\}}$$

with $M > NTV_{max}$ where N is the number of participants, T is the number of time slots and $v_{max} = max_{k,n} V_n^k$.

As an illustrative example, assume that four attendees $\{A, B, C, D\}$ should schedule three meetings $\{E^1, E^2, E^3\}$ in a 3-time-slot agenda. Each meeting takes one time-slot and has the following configuration: E^1 involves A and D, E^2 A,C,D, and B, and E^3 involves B,D, and C. Figure 1 shows the DCOP graph for this example.

4 DMS through Cooperative Mediation

To apply OptAPO in the DMS problem we implemented a simple simulator using Java and conducted experiments with randomly generated instances of the DMS problem. We have used the same scenarios used in [4] to test the Adopt algorithm in real-world problems, mapping them as DCOPs using DiMES, following the PEAV approach. We ran the experiments in a standard PC under Linux.

Let us start discussing the performances of both OptAPO and Adopt in a simple scenario (MaxSAT 3-coloring problem, i.e. the domain size is 3). Although both were evaluated also with a large number of agents (more than 20), the density of the graph and the size of the domains were restricted. They worked with a number of constraints equal to $2 \times K$ and $3 \times K$, where K is the number of nodes. Despite this, the evaluation of Adopt has shown that more than 10,000 cycles were necessary to reach the optimal solution. They have also shown that

approximately 50 messages were exchanged among the agents in each cycle. Although the total time (in seconds) has not been analyzed, it is reasonable to expect this to be high. We can see in these results that to find an optimal solution is expensive even in simple scenarios. The total number of exchanged messages exceeds 500,000, which may represent a significant overload in a communication network. It is important to point out that the performance of Adopt was compared to a centralized search based on B&B, i.e. the same centralized search used in OptAPO. Adopt outperforms B&B in this analysis.

The evaluation of OptAPO in the same scenario has shown an immense reduction in the number of messages and cycles. The total number of exchanged messages does not exceed 15,000 and the number of cycles does not exceed 120. OptAPO outperforms Adopt in this analysis, proving that, in simple scenarios, the OptAPO runtime is not a byproduct of the centralized search (B&B).

Let us consider a more complex, real-world scenario: a DMS problem. In this scenario 9 attendees, with a 8-time-slot agenda, must schedule 8 meetings. This scenario was randomly chosen the data set in [4]. Converting this DMS problem to DCOP (using PEAV) generates 23 variables, with a 8-element domain, and 16 constraints. When Adopt is applied in this scenario, its performance decreases dramatically.

For the class of DMS problems, the authors have proposed two heuristics to speedup the basic Adopt (which has not performed as good as in the graph coloring scenario discussed above). These heuristics have shown good results. One heuristic converts the constraint graph into a deep-first search (DFS) tree which is used as a hierarchy to communicate the values and costs. The authors also suggested to replace this structure with a MLSP (Minimum Depth Spanning) tree. They have shown that the communication structure affect the time of convergence to the optimal solution. Besides, the pre-computation of a best case bound in a distributed fashion was proposed. It was shown that the initial accuracy of this bound affects the convergence in complex scenarios. These heuristics were evaluated in the DMS scenario described above and the total number of cycles felt to less than 15,000. Despite this decrease, in a real application, to exchange 750,000 messages (15,000 cycles times 50 messages per cycle) is still a problem.

Since OptAPO has shown a better performance than Adopt in the graph coloring scenario, we expect a similar performance in the DMS one. Initially we intended to compare the original results of Adopt with the results archived in the scenario proposed above using OptAPO. However, it was not possible due the time complexity associated with the scenario.

In a recent paper, Davin and Modi [1] analyze the OptAPO and Adopt under a evaluation metric different of the one used here. In this new analises the authors shown that Adopt in complex scenarios requires less computation of OptAPO despite been outperformed by OptAPO in number of cicles, that was the metric used before (and here also). Their conclusions highlight our experimental results. Next we analyze what happens.

The OptAPO algorithm uses a partial centralization technique (cooperative mediation) based on a centralized search mechanism (B&B) to achieve the optimal result. The performance of OptAPO is closely related to the B&B. Each time an agent decides to mediate in OptAPO (active or passive) it conducts a B&B search in their good_list. As the size of the good_list grows, so does the size of the B&B search space. If we have N variables in the good_list, and M is the domain size, then the size of the B&B search space is M^N in the worst case. Our scenario has 23 variables with an 8-elements domain, which means to search within a 6×10^{20} possibilities. The number of variables involved in the B&B increases dramatically the search space in scenarios with large domains. During the OptAPO execution, the agents' good_list tends to increase as additional links are created due to external conflicts. Through this process, there is a tendency of at least one agent achieving complete centralization and its good_list will have all variables. So, there are B&B searches during the OptAPO execution that involve all variables, which means the worst case.

Therefore, since OptAPO would not run for the above scenario, we first reduced the complexity of it. Later we will return to it, via an heuristic approach. Basically, the reduction is achieved by abandoning the PEAV and using an EAV (see Section 3) instead, in which there are less variables (8). Adopt was executed for the same scenario, and the computer configuration was the same. Table 1 shows the runtime (in seconds), the number of exchanged messages, and the number of cycles of OptAPO, a centralized search based on B&B, Adopt, and Adopt with the speedup heuristics (here after we call it Adopt+H).

	OptAPO	**B&B**	**Adopt**	**Adopt+H**
runtime (s)	61133	56641	12529	89
messages	768	-	3370700	15798
cycles	19	-	146566	701

Table 1. Evaluation of DCOP algorithms in the DMS/EAV scenario

Interestingly, OptAPO is worse than the B&B. Let us see why there was a degradation in the OptAPO performance from the graph-coloring scenario to this one. Two heuristics were used to speedup the B&B inside the OptAPO algorithm: the first branch of the search was the current solution; and the search terminates early when the bound is equal to the current optimal local cost and the cost for the agents outside the mediation is 0. These heuristics are important because the quality of the B&B for a specific problem depends on how the branching takes place and which bounding scheme is used. Let us focus on the initial steps of OptAPO execution, where the performance of B&B should, theoretically, be the best due to the good_list size. In an over-constraint scenario the agents' good_list start with many variables (the good_list is first composed by variables' that shares a relation with the agent by a constraint). The primary values of the variables are randomly setup and the current optimal local cost is equal to 0. Due to these characteristics, the speedup proposed for the B&B

does not affect the initial phase of the OptAPO algorithm. There is no optimal solution computed yet - the current solution is randomly determined, there are many agents in the good_list and no early termination in the search is possible (the bound could not be equal to zero).

For example, when simulating our reduced DMS scenario (EAV), the first search involves all 8 variables, which means a search space of 16 million possibilities. The agent who decides to mediate first (the agent with the highest priority, i.e. the one with more neighbors) has constraints with all other 7 variables. In this case, the initial solution is very poor and better solutions are found only gradually. In summary, when OptAPO is used in this scenario, it has to explore the entire solution space degrading the performance already in the first steps. As the size of the good_list grows, so does the search space of the B&B, increasing the time consumed by OptAPO. Moreover, OptAPO has to run other centralized searches during the further steps of its execution, potentially with more than one including all variables.

The Adopt+H shows the best results due to the speedup heuristics, outperforming all other algorithms. In particular, Adopt without heuristics outperforms OptAPO in terms of runtime. The same arguments used above to justify OptAPO's low performance also apply here. Since Adopt outperforms the centralized B&B already for simple scenarios, and OptAPO was outperformed by B&B here, we expected this to happen. Also, this could be forecasted given the graphs presented in [5], when the authors discuss the performance of Adopt and OptAPO regarding more dense graph (higher number of constraints). Although OptAPO starts with the best performance (in terms of runtime), there is a trend of this not continuing for more dense graphs because OptAPO's performance decreases exponentially while Adopts decreases linearly.

However, and this is very important in real-world scenarios, the number of messages exchanged by OptAPO (as well as the number of cycles necessary to achieve the optimum) is lower than both Adopt and Adopt+H, showing the value of cooperative mediation. Even if Adopt+H outperforms the other algorithms by a large difference in terms of runtime, the number of exchanged messages and cycles taken in this reduced scenario is significantly large.

Assuming that in real-world applications it is normally enough to have a solution close to the optimal, in order to take advantage of the value of the cooperative mediation, we propose to trade the completeness of the B&B search mechanism for the performance of a heuristic search. Our aim is to have the best of the two worlds: OptAPO's performance in terms of number of messages and cycles, and performance in terms of runtime. Therefore, instead of using the B&B algorithm within OptAPO, we use a GA for the search. Of course, this mechanism does not guarantee optimality. However, GAs works well in very large search spaces. Besides, the GA mapping fits this problem well, as demonstrated by the pure-GA based approaches proposed for scheduling and optimization problems [2].

In GA, each string is a possible solution having as many genes as variables appearing in the DCOP. We represent each element of the domain as an integer.

In order to fit the gene and string mapping, integers were converted to binaries. For example, in our reduced (EAV) DMS scenario there are 8 elements in the domain and 8 variables. Each element in the domain represents a possible meeting's start time x^k. As the agenda has 8 time slots and each meeting consumes one of them, the domain values are 0 to 7. Each element of the domain is represented by 3 binary digits. Thus, the string has 8 binary elements, one for each variable. Considering these, each one representing a meeting E^k, one of the optimal solutions for this scenario (obtained from an Adopt+H run) is: $E^0 = 3$, $E^1 = 0$, $E^2 = 2$, $E^3 = 5$, $E^4 = 7$, $E^5 = 2$, $E^6 = 7$, and $E^7 = 1$. This solution is represented as 011000010101111010111001 in the population.

Each generation has 200 strings. The fitness function is the same DCOP global objective function discussed in Section 1. The cost of a solution is computed as a sum of the costs of each constraint. Strings are ordered according to the cost and only a set with 40% of the lowest cost solutions are chosen for reproduction. The best 5% of this set stay unchanged in the next generation; the other 35% reproduce by crossover. From this able-to-reproduce population, a small number of genes are mutated with a rate of $\frac{1}{gene\_size \times num\_of\_genes}$. In our example, only 0.02% mutates. The solution of the GA is the string with the lowest cost after 200 generations.

Table 2 shows the performance of OptAPO using the GA search mechanism (HeuAPO), Adopt+H, and, for sake of comparison with a centralized search, one based only on GA. The runtime, the total number of cycles, and the number of exchanged messages were measured. Here we can afford to use the more complex DMS scenario (PEAV with 23 variables) described in the beginning of this section. The HeuAPO outperforms Adopt+H in all measured parameters. Besides, HeuAPO reaches the optimal solution in this scenario just as Adopt does.

	HeuAPO	**Adopt+H**	**GA**
runtime (s)	978	8268	83
messages	5934	906758	-
cycles	192	13991	-

Table 2. Evaluation of the heuristic algorithm in a DMS scenario, compared to Adopt, and to a centralized search based on GA

5 Conclusions and Further Work

In this paper we analyze the performance of several approaches to solve real-world distributed meeting scheduling problems modelled as distributed constraint satisfaction problems, two well-known scenarios in AI. We compare the performance of the OptAPO and Adopt algorithms. Adopt needs a large number of messages and cycles to converge to a solution, while OptAPO performs

worse in terms of runtime, both critical issues in real-world applications. For instance, it was shown that the main assumptions about OptAPO in simple scenarios are not necessarily valid in more complex ones. In our experiment, in terms of runtime, OptAPO was worse than the B&B used to do the internal centralized search, and worse than Adopt, due to the reasons explained in the previous section. However, OptAPO outperforms Adopt in the number of exchanged messages and number of cycles.

Given the performance of Adopt and OptAPO, we have proposed the use of an heuristic search mechanism to replace the B&B in the cooperative mediation. The results obtained are very promising: the heuristic version of OptAPO achieves the best solution with a significant better performance, outperforming Adopt even with the speedup heuristics.

In the future we intend to pursue three directions. First, to run our approach with several different scenarios to check the quality of the solutions. Second, to compare the results of our proposal with other incomplete, heuristic DCOP algorithms, including a version of Adopt which uses an error threshold thus permitting the algorithm to stop earlier (when the solution is within the threshold). Finally, it would be interesting to investigate which classes of problems are adequate for which type of DCOP algorithm. Liu and Sycara [3] discuss the use of partial overlap among subproblems solutions. In their algorithm, *Anchor&Ascend*, an anchor agent is dynamically chosen and conducts a local optimal search in its subproblem. According to the authors, their approach is really effective when the structure of the problem exhibits extreme disparity among the subproblems. Maybe it is the case that OptAPO is also sensible to the structure of the problem, so that not only the complexity of the DMS problem could be affecting OptAPOs performance, but also its structure. Therefore, a similar analysis (algorithm performance *vs.* problem structure) should be done regarding OptAPo.

References

1. John Davin and Pragnesh Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *Proceedings of Autonomous Agents and Multi-Agent Systems - AAMAS*, 2005. To appear.
2. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.
3. JyiShane Liu and Katia P. Sycara. Exploiting problem structure for distributed constraint optimization. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 246–254, San Francisco, CA, 1995. MIT Press.
4. R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 310–317, Washington, DC, USA, July 2004. IEEE Computer Society.

5. R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445. IEEE Computer Society, 2004.
6. Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2003. ACM Press.
7. Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling. In *Conference on Organizational Computing Systems*, pages 310–317, september 1991.
8. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.