HYBRID MULTI-AGENT ARCHITECTURE (HMAA) FOR MEETING SCHEDULING

Serein Al-Ratrout, Francois Siewe**, Omar Al-Dabbas***, Mai Al-Fawair**

*Department of Software Engineering, Alzytoonah University, Jordan
**Software Technology Research Laboratory, Demontfort University, UK
***Faculty of Engineering, Al-Balqa Applied University

ABSTRACT

Abstract- This paper presents a novel multi-agent architecture for meeting scheduling. The proposed architecture is a new Hybrid Multi-Agent Architecture (HMAA) that generates new heuristics for solving NP-hard problems. Moreover, the paper investigates the feasibility of running computationally intensive algorithms on multi-agent architectures while preserving the ability of small agents to run on small devices, including mobile devices. Three experimental groups are conducted in order to test the feasibility of the proposed architecture. The results show that the performance of the proposed architecture is better than those of many existing meeting scheduling frameworks. Moreover, it has been proved that HMAA preserves small agents' mobility (i.e. the ability to run on small devices) while implementing evolutionary algorithms

Index Terms— multiagent, meeting scheduling, heuristic, genetic programming.

1. INTRODUCTION

This research proposes a "Hybrid Multi-Agent Architecture" for solving many NP-hard problems. The researcher believes that a method for computing solutions for NP-hard problems using the algorithms and computational power available nowadays within a reasonable time frame remains undiscovered. Unfortunately, many practical problems such as route planning, scheduling, calendar management/meeting scheduling and creation of timetables fall into this class. It is essential that these problems are solved, and the only possibility of doing so is to use approximation techniques, since no straightforward solution technique is known.

Researchers tend to use *Heuristic* techniques [7, 8] because they are generally powerful. However, heuristics are not absolutely guaranteed to provide the best solutions, or even to find a solution at all. This demands adopting some optimisation techniques such as Evolutionary Algorithms (EA) or Evolutionary Computation (EC) [17, 9].

The present work proposes a *new Hybrid Multi-Agent Architecture (HMAA)* for solving NP-hard problems. This architecture is hybrid because it is a *semi-distributed/semi-centralized* architecture. In the proposed HMAA, variables and constraints are distributed among small agents exactly as in distributed architectures. But when these small agents become stuck, a centralized control becomes active where the variables are transferred to a super agent that has a central view of the whole system and possesses much more computational power and intensive algorithms such as EAs to find an optimal solution.

This can be done by defining different classes of agents [6] including super agents and small agents. Heuristics of small agents that are fixed and limited can be updated by the super agent that generates new skills/heuristics using evolutionary
approaches. The Meeting Scheduling Problem (MSP) has been adopted and investigated in order to examine and validate the idea.

In an MSP [1, 2, 4] each scheduling agent manages the calendar for its user, while the basic objective is to find a common free time slot for all participants in a particular meeting. Where MSP is a NP-hard problem, it is not possible to optimally solve every instance of MSP in an acceptable time using the algorithms and computing power available nowadays [1]. It has been solved within the MAS environment using heuristics, a solution that holds the promise of finding feasible solutions within a reasonable time [7]. There are several heuristics for MSP [3, 4, 5, 10, 11, 13, 14], but most of them are of limited success because they use predefined deterministic heuristics which are domain specific, meaning that they work well in some environments and not in others.

MSPs and others have been solved within MAS and suffer from this limitation. Mobile agents within distributed MAS have limited capabilities and cannot perform complicated computations that would generate new solutions. "Overcoming this limitation of having

restricted capabilities/heuristics that work well in some environments and not in others; by implementing computationally intensive algorithms on super/central agents, propose computationally intensive algorithms in order to generate new heuristics to be executed by the small agents, while preserving their simplicity and their ability to run on small devices" is the motivating factor for this study.
Section 5.2 discusses the formalization of the Meeting Scheduling Problem (MSP) adopted in this research; Section 5.3 illustrates the proposed solution approach for MSP with HMAA. Scenarios of HMAA negotiations and MSP within HMAA are illuminated in Section 5.5. Finally Results and analysis are discussed in Section 5.6

2. MEETING SCHEDULING FRAMEWORKS

Each meeting x_i has a set of attendees which are one or more of the agents. This formal definition contains the following elements:

1. Agents: each agent represents its user.
2. n variables xi (i=1…n), each representing a meeting,
3. n domain sets Di (i=1…n), where each $D_i=\{t_1,t_2,\dots,t_m\}$ is a set of timeslots which are the possible values of the corresponding variable x_i.
4. Constraints: define which domain values are valid assignments.

The HMAA implements the following two formalisations for the MSP:

1) Distributed Constraint Satisfaction Problem (DisCSP):

DisCSP [12] where MSP is considered as a search problem consisting of a set of distributed *agents*, each one having a set of variables represent *meetings* $\{x_1, x_2..., x_n,\}$, and each domain is a set of timeslots, and each agent has a *hard constraint* C_H, which stipulates that no two meetings are scheduled at the same time. The goal is to search for the value assignment that satisfies the agents' constraints.

2) Distributed Constraint Optimization Problem (DCOP):

DCOP [15, 16] where MSP is considered is an optimization problem consisting of a set of distributed *agents*, each one with a set of variables represent *meetings* $\{x_1, x_2..., x_n,\}$, and each domain is a set of timeslots, and each agent has *a hard constraint* C_H which stipulates that no two meetings are scheduled at the same time. The goal of the agents is to choose time slots from the domains for the meetings to *optimize* (i.e. minimize) the violation of constraints.

Two formalizations have been adopted in order to generalize HMAA more and enable it to encompass more specifications and needs. This is because in some cases meetings must be scheduled, leaving the choice of which meeting to attend to individual participants, while trying to minimize the overlapping meetings as much as possible (the optimization problem). In other situations, not scheduling meetings leaves to the initiator the opportunity to enter new options or domains for these unscheduled meetings (the search problem). Hence, HMAA implements these two options, leaving the users the choice of which framework is more suitable to its situation.

3. SOLUTION APPROACH FOR MEETING SCHEDULING WITHIN HMAA

SUper Agents and SMall Agents in meeting scheduling architecture HMAA cooperate in such a way that the SUper Agent is the centre of the whole system. It decides the moves that the SMall Agent should follow (the heuristic), in order to overcome a failure, or to optimise the current solution. SMall Agents obey the commands of this central agent. This central agent is called the super agent, and the other agents dominated by it are called small agents.
The small agents are responsible for autonomously managing the scheduling process on behalf the individuals they represent, through negotiation between each other. Agents negotiate by having one agent propose a meeting which the other agents accept or reject, based on whether or not the proposal fits their own schedules. Each agent knows its user's calendar availability, and the meetings to be scheduled with the attendees' ranks in order to act on behalf of its user.
As mentioned in the previous section, the scheduling activity is regarded as an optimisation problem for which the best feasible schedule is sought, or a search problem for which a feasible schedule is sought.
Within the context of this research project the researcher has proposed a solution technique for meeting scheduling and repair strategy for the attained solution. The heuristic considers the different parameters (the available time domains for the meetings, and meeting and attendee rankings) in the scheduling process. If there is a violation, the repair strategy then starts from the initial violated solution and enters a loop that navigates the search space, stepping from one solution to one of its neighbours. The neighbourhood is composed of the solutions that can be obtained by a local change from the current one.
However, the capabilities of the small agents are restricted, since they adopt fixed negotiation skills that would sometimes fail to attain a feasible solution, so that it reaches a dead-end - i.e. there is no possible value for the current variable. To overcome these limitations, the small agents would pass on their situation to a super agent which has more sophisticated algorithms, based on more

powerful functions that would generate new better solutions. This super agent is capable of generating new heuristics/negotiation skills using evolutionary approaches like Genetic Programming (GP).

4. HYBRID MULTI-AGENT ARCHITECTURE PROPOSED

The proposed architecture is hierarchical as depicted in Fig. 1. Which are, from top to bottom: the Super Agent (SUA) layer, the Facilitator Agent (FA) layer, the SMall Agent (SMA) layer and the Interface Agent (IA) layer. SUper Agent (SUA) flanked on one side by the Facilitator Agent (FA) and FA on the other side flanked by the SMall Agent (SMA). The SMA adjoining the (FA) layer and the Interface Agent (IA); the latter is only connected to the SMA. The arrows are interactions between the architecture components. Each pair of adjacent layers can communicate with each other by exchanging data and messages.

The interface agent (IA) is reactive agent: it responds to changes in the environment; receives input from users then update the database with the input data, and exchanges data with the SMA (scheduling request). Each IA correlated on one SMA is the window of this SMA to the external environment, where SMA can perceive the environment, receive input from the environment, and present the attained results to the external environment through this IA.

The SMA is a Believe Desired Intention (BDI) small agent, cannot perform complex computations, but rather receives data from the IA or FA, incorporates small algorithms to accomplish specific tasks, and sends the results back to them. SMA can receive data from the meeting database and input updated data -the assignment field- to the same database. SMAs are interacting by sending and receiving messages through FA.

The FA is the central agent, like a server agent; any two or more agents who want to talk or exchange messages or data do so through the FA. The FA knows the agents' names, IDs and addresses or locations, so it forwards the messages it receives to the corresponding agents. Any new agents added by the administrator or environment should be registered in the FA with their names, IDs and locations, which is why the FA is considered to be a reactive agent, FA receives data from environment (administrator), exchange data with SMAs (messages) and with SUA (heuristics).

Finally, the SUA is a BDI agent, performing the same tasks as the SMA, but it can have very large computational power and can implement computationally intensive algorithms like EAs that find more solutions and perform better for NP-hard problems. Meetings are variables to be assigned; they are the problems that the system HMAA would find a solution or value for. SMAs are the only agents responsible for assigning values or finding solutions for the problem; they can read data and update values. SUAs can read stored data from meetings in order to be able execute their algorithms, but cannot update values; they pass the result (heuristic) to the FA who forwards it to the corresponding SMA, which updates the values for the meetings according to advice received from the SUA in order to overcome the failure or to reduce the violation.

One SUA can be defined and large number of SMAs and IAs can be initialised, where each SMA correlated to one IA.

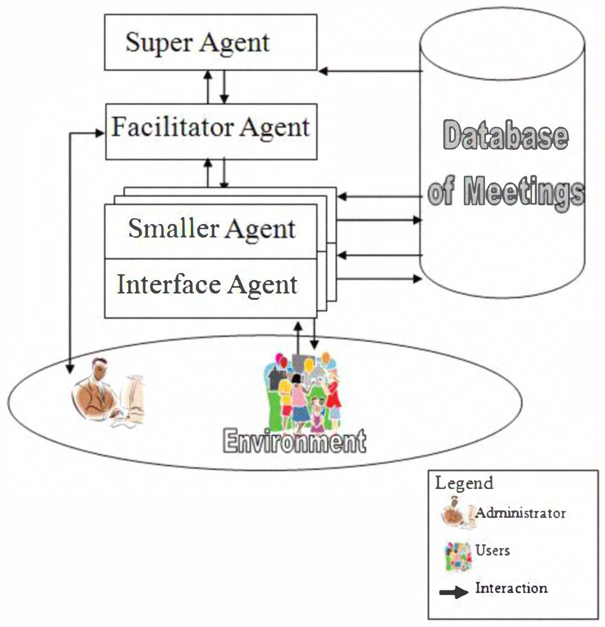

Fig. 1: HMAA Architecture

5. SCENARIO: MSP WITHIN HMAA

The process starts when an employee decides to hold a meeting. As shown in Fig. 2; he registers in the HMAA by defining his name and location to the FA which stores this information, assigns an ID, create an SMA and an IA for him and send him a list of logged users. The user enters a meeting request, defines a list of time domains for this meeting and a list of attendees from the list of logged users with their ranks to attend the corresponding meeting through the IA.

The IA accepts the data, creates a meeting as a variable to be assigned a value, and sends the request to the SMA. The SMA initiator proposes one time slot and sends the proposal to the FA for forwarding to the other SMA attendees who would check their calendars and send replies as to the proposal's acceptability or otherwise to the FA to send back to the SMA initiator. When the latter has received all the replies, it calculates the violation (i.e. how many conflicts replies there are). If the violation values greater than 0, it tries to find a better proposal, by putting forward the next time slot from the domain. If there is no better solution, he sends the best one available to the FA to forward to SMA attendees for confirmation (optimisation problem), and updates the meeting variable with the best gaining value.

If a violation remains after all the meetings have been scheduled, then the SMA contacts the SUA through the FA to try to find a better solution by performing extensive algorithms for the NP-hard problem. The SUA

interrogates the database to obtain the relevant data and executes its evolutionary algorithms.
SUA uses Linear Genetic Programming (LGP)[18] to accomplish its task in searching for better algorithm. In FMAA two types of LGP SUper Agents have been implemented (SUA_LGP and SUA_LGP_LO). SUA_LGP_LO exits searching when it reaches the local optimum (i.e. where the next generations' violations are greater than the current parents' violations), unlike SUA_LGP which continues searching until it reaches to the optimal solution.
Once the SUA finds a heuristic with better results it passes it to the corresponding SMA through FA, who will follow the recommendations of SUA, and update the meetings' values accordingly.

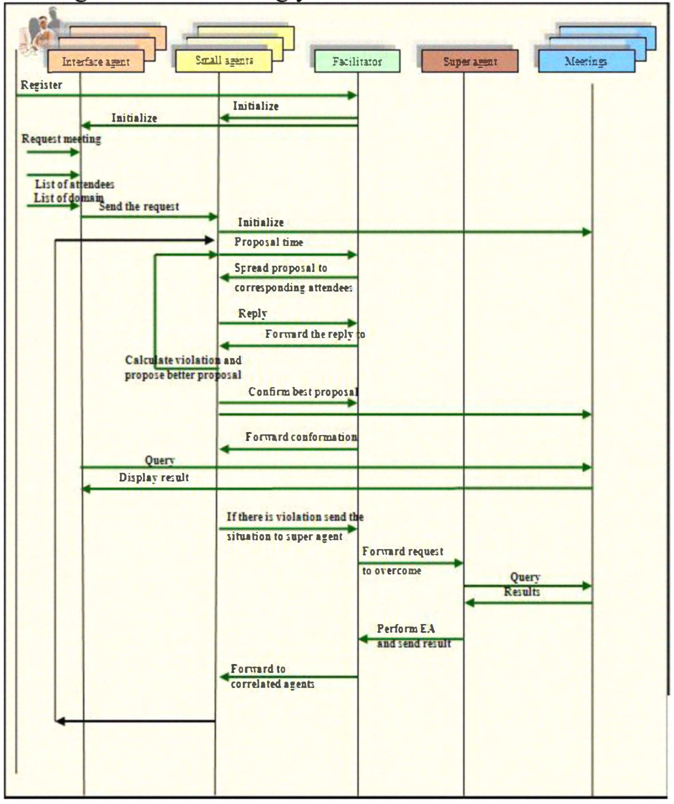

Fig 2: Sequence diagram for Meeting Scheduling Problem within HMAA

6. RESULTS AND ANALYSIS

In experiments; three groups of experiments are conducted. The first presents simple cases with differing numbers of attendees and a variety of situations and combinations of meetings. The aim of these experiments is to examine the feasibility of running the HMAA for large number of attendees.

The second group contains more complicated cases and susceptible situations with one user, who has chosen their data carefully, and a large number of meetings. This group was constructed in order to measure the feasibility of the system in very complicated cases when the domain range data is limited. The cases measure the feasibility of the local optima in the searching process. This means that instead of continuing until the solution is found, or completing a specified number of rounds, the search ends when the number of violations of the new generations are more than that of the parents' violations (for Linear Genetic Programming),
The final group consists of randomly selected cases that measure the feasibility of the proposed architecture in different situations. It must be said that each meeting is determined by the specific domain of possible dates on which the meeting can be held, and the potential attendees with their weight/significance. This is because the HMAA aims partly to optimise scheduling, which means reducing the number of violations as much as possible so that if there is no feasible schedule without overlaps, the system schedules the meeting by overlapping the meetings of the least significant attendees

6.1 Analysis for Experiment Group 1

The following observation could be made:

- It is feasible for the proposed framework to run any number of users without affecting its performance. HMAA has been run on up to ten users, and it could quite conceivably run hundreds.

6.2 Analysis of Experiment Group 2

From this group of experiments the following observations could be made:

- HMAA in this group has been examined on very complicated situations; where the meetings' domains are very constrained. We have documented one case of these situations:

"Having (16) meetings and (30) domain timeslots; (%43.34) of this domain is unavailable (constrained); hence (%56.66) of this domain is available. On the other hand the percentage of the domain needed to schedule these 16 meetings is (16/30=%53.34). Then the meetings would be scheduled in %53.34 of the domain where %56.66 is available."
The results of this group of experiments show that the percentage of the scheduled meetings' violation using SMA default heuristic is (%36), and this percentage is considered small since these cases are very restricted and constrained situations. On the other hand, this violation has been reduced more in the cases of using the help of one of the SUAs. Using SUA that deploys Linear Genetic Programming (SUA_LGP) reduced the violation up to 3%, while using SUA deploys Linear Genetic Programming that breaks in local optima (SUA_LGP_LO) reduced the violation up to 6%. Hence it is perfectly feasible for the four SUAs in HMAA to solve very susceptible and complicated situations; they reduced the violations by a huge margin of the SMA's violation (33 per cent, and 30 per cent respectively). And this level of reduction is a great achievement.

- The feasibility of the system is measured by the number of violations as well as the time taken or the number of rounds needed. Sometimes local optimum is more feasible than global optima, in some cases SUA_LGP reached the same violation of SUA_LGP_LO in a very small number of rounds. In other cases SUA_LGP reached to “zero” violations but in large number of rounds, while SUA_LGP_LO reached to small violation (i.e. 2) in small number of rounds, Hence, in some cases in is more feasible to reach acceptable violation in acceptable time; than “zero” violation in very long time.

6.3 Analysis of experiments Group 3

In these experiments, three agents are initialised and thirty meetings with different domain distributed between them. Some of these domains overlap.
The experiments have been carried out in three stages:

- *Stage 1* — the SMA stage. The SMA attempts to resolve problems by using its heuristic to scheduling meetings with a minimum of violations.
- *Stage 2* — the SUA_LGP. The SMA asks the SUA_LGP to generate new heuristics to solve the violations. The SUA_LGP uses its LGP algorithm to generate new heuristics for the SMAs.
- *Stage 3* — SUA_LGP_LO. The SMA asks the SUA_LGP_LO to generate new heuristics to solve the violation. The SUA_LGP_LO uses its linear genetic programming algorithm that breaks in local optima, to generate a new heuristic for the SMAs.

These experiments can measure the feasibility of the system by comparing the results of Experimental Stage 1 with Stage 2 and Stage 3. As it can be seen in Fig. 3, the SUAs in the implemented HMAA are responsible for a big reduction in the number of violations. The violation of the SMAs using its heuristic was 94, which was reduced to between 18 and 20 using one of the SUA-generated heuristics. This means that the new HMAA can reduce violations by nearly 75 per cent.

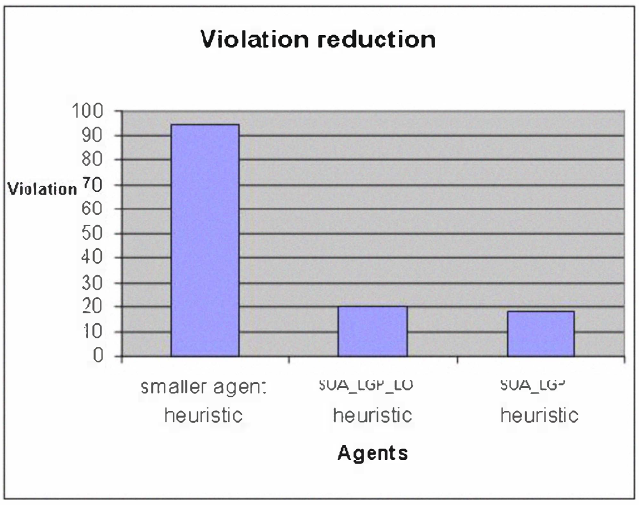

Fig. 3: Violation reduction

By comparing results of Stage 2 with those of Stage 3, the SUA_LGP_LO produces algorithms that produced violations totalling 20, while the SUA_LGP equivalent was 18; this reveals that SUA_LGP is better that SUA_LGP_LO.

Fig. 4 shows that the SUA_LGP loops 60 rounds while SUA_LGP_LO loops 54: SUA_LGP loops 10 per cent more rounds. This is the reason why SUA_LGP violations are fewer in number than those of SUA_LGP_LO. But both are good enough to solve many complicated and sensitive situations

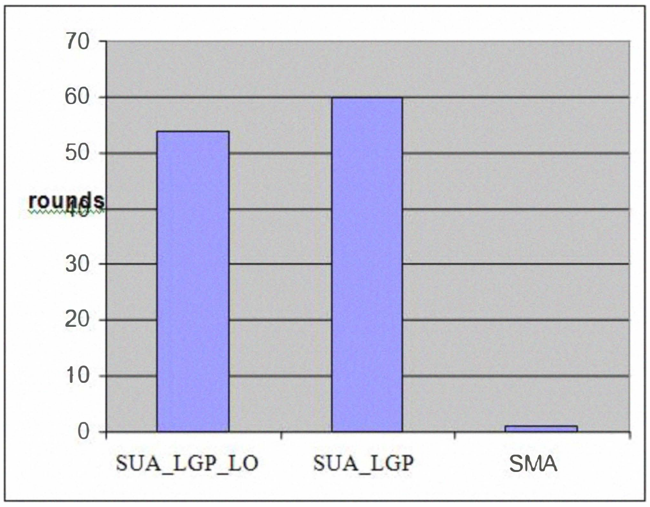

Fig 4: Loops and iteration

7 CONCLUSION

The research proposed HMAA; a clarification for the motivations to this new architecture has been stated. The research shows the architecture of HMAA which is hierarchical architecture composed of four adjacent layers: SUA, FA SMA, and IA. And the relationship between these layers has been clarified. The Meeting Scheduling Problem (MSP) has been adopted and investigated in order to examine and validate the idea.
Some experiments performed on HMAA have been discussed. The experiments have been done on both frameworks search and optimisation problem solving frameworks.
The results reveal that this architecture is applicable to many different application domains because of its simplicity and efficiency. Its performance is better than those of many existing meeting scheduling frameworks. It preserves small agents’ mobility (i.e. the ability to run on small devices) while implementing evolutionary algorithms. HMAA is very robust in that it can implement more than one optimisation technique without affecting mobility.

8 REFERENCES

[1] A. Hassine, T. B. Ho, and T. Ito. Meetings Scheduling Solver Enhancement with Local Consistency Reinforcement. Applied Intelligence, vol. 24, issue 2, pp. 143–154, 2006.

[2] A. Hassine, T. Bao Ho. An Agent-based Approach to Solve Dynamic Meeting Scheduling Problems with Preferences. Engineering Applications of Artificial Intelligence, vol. 20 , issue 6, pp. 857-873, 2007

[3] A. Hassine, T. Ito, and T. B. Ho. A new Distributed Approach to Solve Meeting Scheduling Problems. In the proceedings of IEEE/WIC Int. Conf. IAT, 2003.

[4] A. Hassine, X. Defago, and T. B. Ho. Agent-based Approach to Dynamic Meeting Scheduling Problems. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, vol. 3, 2004.

[5] A. Meisels and O. Lavee. Using Additional Information in Discsps Search. Distributed Constraint Reasoning Workshop (DCR), 2004.

[6] C. Guilfoyle and E. Warner. Intelligent Agents: the New Revolution in Software. Ovum, 1994.

[7] C. Voudouris, D. Lesaint, and D. Pothos. Solving Large Industrial Problems using Heuristic Search and Constraint Programming. Intelligent System Research, BT, 1998.

[8] D. Partridge. A new Guide to Artificial Intelligence. Ablex Pub. Corp. (Norwood, N.J), 1991.

[9] E. Mezura-montes, A. Carlos, and C. Coello. An Improved Diversity Mechanism for Solving Constrained Optimisation Problems using a Multimembered Evolution Strategy. In the proceedings of GECCO, 2004.

[10] E. Shakshuki, H. Koo, D. Benoit, and D. Silver. A Distributed Multi-agent Meeting Scheduler. Journal of Computer and System Sciences archive, vol. 74, pp. 279-296, 2008.

[11] I. Demirel and N. Erdogan. Meeting Scheduling with Multi-agent Systems: Design and Implementation. In the proceedings of the 6th WSEAS International Conference on Software Engineering, pp. 92-97, 2007.

[12] M. Yokoo, E. Durfee, and K. Kuwabara. The Distributed Constraint Satisfaction Problem: Formalisation and Algorithms. IEEE Transactions on knowledge and data engineering, vol. 10, no. 5, 1998.

[13] P. Modi and M. Veloso. Multiagent Meeting Scheduling with Rescheduling. In the proceedings of the fifth Workshop on Distributed Constraint Reasoning (DCR), 2004.

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

[15] P. Modi, W. Shen, M. Tambe, and M. Yokoo. An Asynchronous Complete Method for Distributed Constraint Optimisation. In the proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS, pp. 161–168, 2003.

[16] R. Mailler and V. Lesser. Solving Distributed Constraint Optimisation Problems using Cooperative Mediation. In the proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS, pp. 438–445, 2004.

[17] T. Back. Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, 1996.

[18] W. Langdon and W. Banzhaf. Repeated Sequences in Linear Genetic Programming Genomes. Complex Systems publications, 15 (2005) 285–306; Inc. 2005.