Partial Cooperation in Multi-agent Local Search

Alon Grubshtein [1] and **Roie Zivan** [2] and **Amnon Meisels** [1]

Abstract. Multi-agent systems usually address one of two forms of interaction. One has completely competitive agents that act selfishly, each maximizing its own gain from the interaction. Auctions and voting scenarios usually assume such agents and follow game theoretic results. The other form of interaction has multiple agents that cooperatively search for some global goal, such as an optimal time slot allocation for all landing aircrafts in an airport.

The present paper proposes a paradigm for multiple agents solving a distributed problem using local search algorithms and acting in a partially cooperative manner. That is, agents with different preferences search for a minimal cost solution to an Asymmetric Distributed Constraints Optimization Problem (ADCOP), while keeping a limited form of self interest.

Two approaches for using local search in the partial cooperative paradigm are proposed. The first, modifies the anytime mechanism introduced by Zivan [19] so that agents can eliminate solutions which do not satisfy their cooperation thresholds. The second proposes a new local search algorithm that explores only valid solutions.

The performance of two innovative algorithms implementing these two approaches, are compared with state of the art local search algorithms on three different setups. When personal constraints are strict, the proposed algorithms have a large advantage over existing algorithms. We provide insights to the success of existing algorithms within the anytime framework when constraints are loose.

1 Introduction

Most studies investigating multi agent systems consider either fully cooperative agents which are willing to exchange information and take different roles in the process of achieving a common global goal (cf. [17, 11, 2]), or self interested agents which are considered to be rational when they take actions that will increase their personal gains (cf. [3]). A notable exception to this two class division includes cooperative game theory which considers cases where self interested agents will join or form coalitions in which they will fully cooperate.

When one considers the standard working environment in which employees perform tasks for the benefit of the organization they work for and get a pay check in return, it seems that this most common situation is not covered by any of the two models described above. The agents in any working environment are naturally self interested and often have the option to increase their own benefit within the organization, even when benefits are non monetary. However, the success of the organization, and ultimately of the agents themselves, requires that the agents act loyally to increase the organizational profits (e.g., optimize some global goal).

Consider for example a factory, in which employees work in shifts. Workers are assigned to shifts according to the constraints they submit to their manager. It is natural to assume that some shifts are more desirable than others. Workers may be willing to take the less desired shift for the benefit of the organization as long as some threshold on their own gain is satisfied. This threshold can be related to the amount of decrease in their own utility, to fairness (uniform distribution of the load among workers), or to different forms of compensation.

Another example naturally arises when agents need to schedule meetings in a working environment. The agents are assumed to be working for the same organization, and need to schedule their meetings to fulfill their own tasks. If agents are asked to postpone or cancel meetings for the benefit of the organization they would probably be willing to do so as long as some level of quality on their meeting schedule is maintained.

In real world situations such as those described above, agents need to collaborate in finding the best (or a good) *global* solution to the problem, despite having personal goals which may be in conflict.

Combinatorial optimization problems similar to the examples described above, e.g., generating employee timetables that incorporate gains from all constraints among agents, can naturally be represented as Asymmetric Distributed Constraint Optimization Problems (ADCOPs) [4]. ADCOPs incorporate personal gains (or costs) of agents within distributed combinatorial problems. Like the Distributed Constraint Optimization Problem (DCOP) [11, 10, 14], they incorporate a cost or utility for every combination of assignments (aka constraints). However, ADCOP constraints may include a different cost for every agent involved in a joint assignment, representing the personal valuation of the assignment combination by the agent.

Previous studies of ADCOPs considered full cooperation of the agents, and assumed that the main reason for maintaining the asymmetric structure of the problem relates to privacy [1, 4]. In contrast, the scenarios described above have agents that are cooperative only when some conditions are satisfied. This generic situation of multi-agent complex interactions raises the need to investigate modes of collaboration for self-interested agents solving combinatorial problems.

The present paper focuses on two new and fundamental questions regarding asymmetric multi agent optimization:

- What are the basic modes of collaboration one can define for ADCOP agents?
- What are the relevant local search methods for exploiting such modes of collaboration?

The following contributions to the investigation of the questions above are presented:

1. A model for representing agents' intentions for cooperation is proposed. Levels of cooperation are defined as a function of the personal outcomes that agents can expect of the process with respect to the expected result of a non-cooperative interaction, of the level of trust they have in others and of their willingness to sacrifice for the common good (the quality of the organizational, global goal).
2. A mechanism that allows agents to reject solutions which do not satisfy their threshold for cooperation is proposed. The mechanism is based on the anytime mechanism of [19]. It keeps track of

[1] Dept. of Computer Science, Ben Gurion University of the Negev, Israel. email: {$alongrub, am$}@cs.bgu.ac.il

[2] Dept. of Industrial Engineering and Management, Ben Gurion University of the Negev, Israel. email: $zivanr$@post.bgu.ac.il

the best solution found in terms of global utility, which satisfies all agents' conditions for cooperation. This mechanism allows the agents to use any local search method, even if it traverses states which are not acceptable by some of the agents.

3. Two innovative local search algorithms are proposed and compared with state of the art local search algorithms. The proposed algorithms attempt to traverse high quality states which satisfy the thresholds on personal gain that agents have. That is, we show that these algorithms provide globally good solutions while maintaining a predefined guarantee on the maximal cost incurred on each agent. All algorithms are evaluated on graphs with different structures and the comparison reveals the advantages and trade offs against state of the art local search.

2 Related Work

Local search algorithms for ADCOPs were investigated in [4]. The investigation revealed that standard local search algorithms which perform well on DCOPs do not converge when solving ADCOPs. Algorithms that guarantee convergence were proposed and shown to find solutions with higher quality than standard DCOP algorithms. The algorithms proposed in the present paper are compared to both DCOP and ADCOP algorithms.

Numerous studies in the field of game theory define various types of equilibria states which address the personal goals of agents in a broader scope (cf. [13]). Strong equilibria notions are most appealing to multi agent systems, but most combinatorial problems do not posses such stable states and require a mediator to address this issue [15, 12]. Moreover, these equilbria do not guarantee efficiency and may often be improved when cooperation is introduced.

An attempt to propose a trust based, game theoretic model for decision making was made by [16]. Their model defines risk and trust in normal form games where a Nash equilibrium serving as a non-cooperative baseline exists. Although this paper is not concerned with distributed search and does not address algorithmic challenges, it inspired our work. An additional challenge which we address in our work and was not addressed in [16] is the the level of cooperation that is required in order to consider solutions which do not Pareto improve the baseline state.

Another study which proposed partial cooperation with respect to a non-cooperative baseline was [20]. This study considered resource allocation problems (cake division) and the baseline allocation used was the non-cooperative proportional division. Our work can be considered as a generalization of this preliminary work to complex multi agent interactions.

Finally, a recent study proposes an optimization scheme for solving a specific network based game [5]. Focusing on cooperative agents interacting in this game, authors propose a solution which is proven to secure a minimal game theoretic based gain to each agent. The present study is able to generalize upon this approach, and proposes different modes of (partial) cooperation, regardless of the underlying problem addressed.

3 Asymmetric DCOPs

An $ADCOP$ is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. $\mathcal{A}$ is a finite set of agents $A_1, A_2, ..., A_n$. $\mathcal{X}$ is a finite set of variables $X_1, X_2, ..., X_m$. Each variable is held by a single agent but an agent may hold more than one variable. $\mathcal{D}$ is a set of domains $D_1, D_2, ..., D_m$. Each domain D_i contains a finite set of values which can be assigned to the variable X_i. $\mathcal{R}$ is a set of relations (constraints).

Each constraint $C \in \mathcal{R}$ of an asymmetric DCOP is defined over a subset of the variables X_C, and maps any value assignment combination for these variables to a vector of non-negative *costs*:

$$C : D_{i_1} \times D_{i_2} \times \cdots D_{i_k} \rightarrow \mathbb{R}^k_+ \quad (1)$$

This vector includes the cost for each agent whose variables are included in X_C.

A *binary constraint* refers to a partial assignment with two variables, i.e., $k = 2$. An *optimal solution* is a complete assignment of aggregated *minimal cost*. In maximization problems, each constraint has utilities instead of costs and an optimal solution is a complete assignment of maximal aggregated utility. In the remainder of this paper, the problems we discuss are binary minimization problems in which each agent holds exactly one variable.

4 Partial Cooperation Model

Previous studies of ADCOPs assumed full cooperation by the agents and that the main reason for the asymmetric structure of the problem relates to privacy [1, 4]. The collaboration model for ADCOPs proposed below (λ-cooperation), enables representations of partial intentions for cooperation of agents.

The level of cooperation (which will be represented by λ) affects the possible outcome of an *Interaction Process* among agents in an ADCOP. The interaction process is defined as follows:

Definition 1. *Given an ADCOP, an* Interaction Process IP *of* n *agents is a predefined sequence of events that upon termination has each agent select an assignment for its variables.*

An interaction process defines a sequence of actions to be performed by agents prior to the assignment selection, for example:

1. Agents select an assignment without any interaction among them. This scenario is similar to simultaneous one shot games which are intensively studied by game theoreticians [9].
2. Agents perform k synchronous message exchange iterations in which each agent sends a message to each of its neighbors and receives a message from each of them. This is usually known as a local ADCOP search algorithm. [4, 6]
3. Agents send messages that initiate a bounded sequence of asynchronous message exchanges. In this process an agent reacts to messages it receives. This IP can be thought of as an asynchronous search algorithm [11, 2, 1].

Following the definition of the interaction process let us examine expected outcomes of an IP for different degrees of cooperation.

Non-Cooperative (NC) - In the non-cooperative setting agents interact with one another based on their local knowledge of the system and driven by their own goals and interest. Typically, local knowledge of agents includes their constraining agents (i.e., their *neighbors*) and their view of the gains and costs of their constraints with their neighbors. Agents perform rationally, i.e., they perform actions that would increase their own utility (or minimize their costs). The outcome of the interaction process depends on the details of the interaction. For a multi-step interaction one can expect the end result to be some form of equilibrium if the problem includes such a state and the interaction process allows convergence to it.

Guaranteed Personal Benefit collaboration (GPB) - Collaborating agents can be expected to reach an agreement. Collaboration in the framework of an ADCOP may require the agents to perform a sequence of actions, as part of the interaction process, that are not necessarily rational. However, the outcome of a sequence of actions in the GPB cooperation model must be a state which is weakly superior for each agent, i.e., the outcome state reached Pareto improves the outcome of the NC process. Such an agreement can be reached via some mediation process and requires that each agent is aware of its expected gain in the NC baseline. [3]

[3] For mediating models that transfer non stable states to stable states in normal form games see [18].

It is important to note that in some multi agent interactions there can be more than a single Pareto improvement state with respect to the initial, baseline state. This provides some freedom for the system's designer in selecting the global most desirable GPB state.

The relevance of a GPB agreement depends on the ability to enforce the binding collaboration agreement. This enforcement can be direct or indirect (active or passive). By direct enforcement, one usually refers to punishment methods to the offending agent, e.g., if the agent does not follow the agreement it needs to pay some tax which reduces the attractiveness of the non-cooperative strategy. Alternatively, an agent can follow the agreement because of its *trust* in its peers or in the system's ability to enforce the binding agreement.

λ-cooperation - In order to allow the agents to consider solutions with high global quality, which are not a Pareto improvement of an initial baseline state, we define a parametrized family of high level cooperation schemes. These are characterized by the requirement that the benefit of some of the agents decreases in order to increase the global quality of the outcome.

The λ-cooperation model is based on the amount of risk, in the form of personal losses, that agents are willing to undertake in order to satisfy the global objective of the organization. The following definitions are used in the proposed λ-cooperation model (as before, and to avoid confusion, we assume minimization problems in the following definitions, i.e., constraints specify costs and not utilities).

Definition 2. *Let μ_i be the non-cooperative (NC) cost of agent i (i.e., the cost for agent i in the baseline solution). The* **risk** *level of agent i is a value $\lambda_i \geq 0$ which defines the maximal* increase *in the value of μ_i which is acceptable by agent i.*

Risk bounds can significantly decrease the number of feasible outcomes, as can be seen in the next definition.

Definition 3. *Let μ_i be the NC cost of agent i and let λ_i be its acceptable risk level. A* **Feasible outcome** *is defined to be any outcome (solution) o in the set of all possible outcomes O, that satisfies the following condition.*

$$O^{feasible} = \{o \in O \mid \forall i \in A, \quad c_i(o) \leq \mu_i + \lambda_i\}$$

Where $c_i(o)$ is the cost for agent i in outcome o.

Definition 4. *An interaction is λ-cooperative if $\lambda = \max_{i \in A} \lambda_i$.*

When $\lambda_i \geq \max_{o \in O} c_i(o) - \mu_i$ for every agent i, agents will be fully cooperative and direct their joint efforts towards attaining the designer's global goals. When $\lambda = 0$ we revert to the GPB cooperation case described above. Thus, in the rest of this paper we consider it as a special case of the λ-cooperation model. For ease of presentation and without loss of generality to the algorithms' properties we use the same λ value for all agents in the remainder of this paper.

5 Partial Cooperation Local Search

Two methods towards finding λ-cooperative solutions in large distributed settings are presented below. The first enables unsatisfied agents (agents whose cost exceeds their baseline cost $+\lambda$) to mark undesired solutions. The best, *valid*, solution is then maintained with a modified version of a distributed anytime scheme [19]. With this modified mechanism agents can safely explore parts of the search space which would otherwise be considered invalid (outcomes $o \notin O^{feasible}$) yet still agree on the best valid solution examined at any instant.

The second approach details an algorithm which guarantees that the personal cost of an agent does not exceed the predefined cooperative value, while constantly seeking globally improving solutions. Agents executing this algorithm exploit possible improvements until they converge to some local minima which can't be further improved without breaching the baseline bound of any agent.

5.1 Anytime Mechanism for λ-Cooperation

The Anytime mechanism for DCOP local search algorithms presented in [19], uses the messages passed by the agents performing a local search algorithm, to aggregate the cost of each state traversed by the agents and determine which was the best among them. The mechanism generates a BFS spanning tree of the constraint graph and each agent is responsible to aggregate and calculate the cost of the state in the subtree it roots and pass it to its parent. After a number of iterations equal to the height h of the tree, the root agent can calculate the cost of a state. If this cost is less than the cost of the best solution found so far, the root propagates down the tree that a new solution was found. Agents hold their value assignment in the best solution and the last $2h$ assignments so that if they are informed that a new solution was found they still hold the relevant value assignment. The overhead required by the algorithm is very small in runtime, storage and privacy terms [19].

In the proposed partial cooperation paradigm, only solutions which satisfy the λ requirements of all agents should be stored. To this end we require that each agent, when passing the information up the tree, includes an indication whether it approves the state in the relevant iteration. An agent indicates that it approves the state in some iteration j if:

1. The state in iteration j satisfies its cooperation threshold, i.e., an agent i approves the state in iteration j if $c_i(o_i) < \mu_i + \lambda_i$.
2. The agent received an $approve$ signal for step j from **all** its children in the tree.

The root agent propagates that the state held by the agents in iteration j is a new solution if the accumulated cost of the state j is smaller than the cost of the best solution it currently holds and if it was approved by all agents. In the beginning of the run the base-line solution is held by the root agent as best. Thus, we are guaranteed that only solutions which satisfy all agents' thresholds can be considered.

5.2 Goods-MGM

The anytime mechanism described in the previous sub-section coordinates an agreeable outcome to general-form distributed local interactions. However, this mechanism provides very little feedback to its underlying algorithm. As a result, agents combining a local ADCOP algorithm with the anytime mechanism may end up exploring invalid parts of the search space without reaching a valid outcome (besides the initial state) throughout their entire execution. The outcome of the algorithm as reported by the mechanism, could, in this case, be the original base line assignment – the input.

Algorithm 1 attempts to overcome this problem by directing outcomes towards valid solutions. The algorithm uses "Good" and "NoGood" messages to signal the agents in the neighborhood $N(i)$ of agent a_i which actions, pending the current state, may result in a breach of its baseline. Algorithm 1 does not guarantee that all the states it considers are valid.

The algorithm proceeds in five synchronous steps that together constitute a single cycle. During initialization, each agent initializes its main data structures:

- $localView$ – The last known state of a_i's neighbors.
- NG_store – A list of neighboring agents to whom a $NoGood$! message was sent, and the context in which this message was sent.
- $elim$ – A list of domain values which were eliminated and the set of agents requesting their elimination.

The agent then proceeds to notify its neighbors of its current state and begins its operation (phases 1–5) until a termination condition is met.

Algorithm 1 - **Goods-MGM**
input: baseLineAssignment, baseLineGain and λ

```
Init:
    value ← baseLineAssignment;
    localView/NG_store/elim ← null ;
    send(value) to N(i);
    while stop condition not met do
        execute phases 1-5;
Phase 1
    Collect all value messages and update localView;
    for all a_j ∈ NG_store do
        ng ← NG_store.get(a_j);
        if ng is inconsistent with localView then
            send(Good!) to a_j;
Phase 2
    for all Good! message collected do
        elim.remove(msg.sender());
    while cost > baseLineCost + λ do
        a_j ← first agent in some ordering of localView;
        v_j ← localView.valOf(a_j);
        localView.remove(a_j);
        NG_store.add(a_j, localView);
        send(NoGood!, a_j ≠ v_j) to a_j and update(cost);
Phase 3
    Collect NoGood! messages and update elim;
    if Domain \ elim == ∅ then
        value ← baseLineAssignment;
        send(BaseLine!) to N(i);
Phase 4
    if received BaseLine! message then
        value ← baseLineAssignment;
    else if did not send a BaseLine! message then
        ⟨v_i, gain_i⟩ ← bestAssignment(Domain \ elim);
        send(gain_i) to N(i);
        canImprove ← true;
Phase 5
    Collect all gain_j messages;
    if canImprove && gain_i > max_{j∈N(i)} gain_j then
        value ← v_i;
        send(value) to N(i);
```

Phase 1: The agent collects all new value messages and updates its local view. It then proceeds to examine its NG_store and validates that the contexts in which $NoGood$! messages were sent to different neighbors are still valid. In case of an inconsistency, a $Good$! message is sent to the relevant agent.
Phase 2: After collecting all $Good$ messages the agent attempts to remove certain values from its $elim$ set. If the sender of a $Good$! message is also the last in the set of agents requesting to remove some value v_i, this value is returned to the current domain. At this point a_i examines its current cost. While this cost is greater than its base line cost + λ (i.e. the agent is "unsatisfied"), an agent a_j with minimal ID is selected and removed from the agent's $localView$. A $NoGood$! message is sent to a_j forbidding its current assignment, and the remainder of the $localView$ is then used as an explanation, or the context, stored in the NG_store for this request. Finally, the cost incurred on a_i by a_j's assignment is removed.
Phase 3: If, after collecting $NoGood$! requests from all neighbors, no values remain in the current domain the agent assigns its base line assignment and requests its local surrounding to do the same.
Phase 4: Any agent receiving a $Baseline$! message assign its base line assignment. If the agent did not receive such a message and did not initiate one, it iterates over the values in its domain which were not removed by $NoGood$! messages and finds the assignment yielding minimal cost. The gain from assigning it instead of the current assignment is then sent to all neighboring agents.
Phase 5: If, after receiving all maximal gain messages, a_i's improvement is greater than its neighbors improved gains, the agent assigns its maximal gain assignment, and notifies its neighbors of this assignment change (as in standard MGM [8]).

It is important to note that unlike most local search algorithms, Goods-MGM focuses on the personal state of each agent rather than the standard minimal global cost. To meet these goals the algorithm introduces both "Good!" and "NoGood!" messages but also allows for local restarts (Phases 3 and 4).

Algorithm 2 Asymmetric Gain Coordination (AGC)
input: baseLineAssignment, baseLineGain and λ

```
Init:
    value ← baseLineAssignment;
    localView ← null ;
    send(value) to N(i);
    while stop condition not met do
        execute phases 1-3;
Phase 1
    Collect all value messages and update localView;
    ⟨val_i, gain_i⟩ ← improvingAssignment();
    send(⟨val_i, gain_i⟩) to N(i);
Phase 2
    Collect all ⟨val_j, gain_j⟩ messages;
    a_j ← agent in N(i) ∪ {a_i} with maximal gain s.t.
          costFrom(n, val_n) < baseLineCost + λ;
    send(Neg!) to N(i) \ a_j;
Phase 3
    Collect Neg! messages;
    if did not receive Neg! && can improve then
        value ← val_i;
        send(value) to N(i);
```

5.3 Asymmetric Gain Coordination

An alternative approach for Local Partial Cooperative search is to design an algorithm that explores only valid states. Here, the anytime mechanism is not required since the validity of the states is maintained in the algorithmic level.Asymmetric Gain Coordination (AGC) is an algorithm that preserves this property and is presented next in Algorithm 2.

An AGC cycle includes three synchronous steps. In the initialization phase agents instantiate a $localView$ data structure, assign their baseline assignment and notify their neighbors of this assignment. Agents then proceed to continuously execute phases 1 – 3 until some stop condition occurs.
Phase 1: Each agent collects all new value messages and updates its local view. Next, each agent attempts to find an improving value assignment. Note that this is not necessarily the maximal gain improving assignment (still an improving assignment) and that different heuristics for selecting an improving value can determine the agents' level of exploration. This phase ends when the assignment, and the cost reduction from it are sent to all neighbors.
Phase 2: Receiving proposed value assignment and gain improvement of neighbors, agents find the neighbor a_j whose unilateral assignment change results in a valid outcome (cost does not breach baseline + λ) of maximal improvement. All other assignment changes, including a_i's proposed assignment if not of maximal gain, are then rejected and a Neg! message is sent to the relevant neighbors.
Phase 3: If an agent did not receive a Neg! message and proposed an improving assignment, it commits to the new assignment and notifies all its neighbors.

Proposition 1. *During the entire execution of AGC, the global state o is feasible (i.e. $o \in O^{feasible}$)*

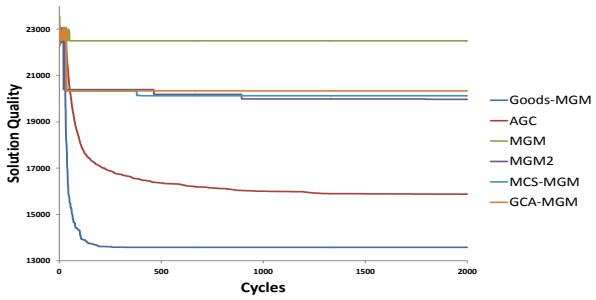

Figure 1. The aggregated solution cost when using the anytime mechanism and $\lambda = 0.4$ on Erdös – Rényi graphs

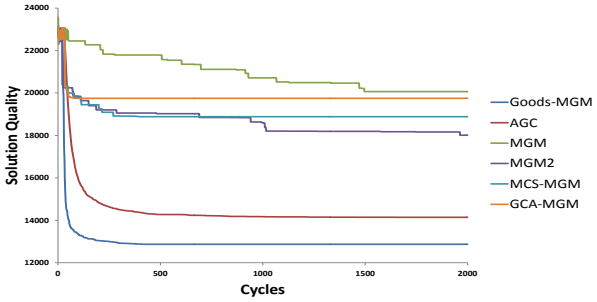

Figure 2. The aggregated solution cost when using the anytime mechanism and $\lambda = 0.8$ on Erdös – Rényi graphs

Proof. First, notice that any proposed assignment which results in a personal cost greater than the baseline $+ \lambda$ cost is rejected in phase 2. This is an insufficient condition – the cost change calculation is made with respect to the current view of the agent. If two of a_i's neighbors simultaneously change their assignment, the cost incurred on a_i may breach its personal bound. Coordinating a single change in each local environment is handled with Neg! messages. Only one neighbor of a_i does not receive such a message from a_i (last line of phase 2), guaranteeing that at most one agent in its neighborhood can commit to a new assignment. □

6 Experimental Evaluation

Three sets of experiments were conducted in order to compare the proposed algorithms with standard DCOP and ADCOP local search algorithms combined with the λ-cooperation anytime mechanism. In all experiments, agents first conduct a short non cooperative interaction to find a baseline solution later used by the local search algorithm.

The initial baseline interaction is composed of 100 synchronous steps in which each agent assigns its best improving value at each step. The local search algorithms executed after this interaction, operate for another 1900 cycles. At each cycle, the aggregated sum of costs is calculated (note that when using the anytime mechanism only feasible outcomes are recorded).

Beside AGC and Goods-MGM, MGM and MGM2 [8], MCS-MGM and GCA-MGM [4] were evaluated – all sharing the same baseline (i.e. 100 steps). Three distinctive problem sets were considered [7]: (1) Uniform random problems where pairs of agents are connected in an asymmetric constraint with a probability value p_1 (Erdös – Rényi graphs). (2) K-regular graphs where all agents share the same degree. (3) Scale Free problems where the underlying network is a scale free network constructed by using the Barabási-Albert model.

Results are averaged over 50 random instances with 100 agents, each holding a single variable with 10 values in its domain and present the aggregated solution quality as a function of cycles[4].

Setup 1 - Erdös-Rényi graphs: in this setup, the p_1 value was set to 0.1. The cost of a joint assignment by two constrained agents was set to 0 with a probability of 0.5 or to a randomly selected (uniform) cost in the range 0..99, otherwise.

Figure 1 and figure 2 present the aggregated costs of the agents when the anytime mechanism is applied to all algorithms and the cooperation parameter λ is set to 0.4 and 0.8 respectively. In both settings the Goods-MGM and AGC algorithms produce significantly better results than all the other local search algorithms. One can see by these results that despite MGM and MGM2's greedy pursuit of improving assignments there are long periods in which no improvement is made. This does not mean that the algorithms are not improving the aggregated total cost but rather that at least one agent breached its baseline $+ \lambda$ gain. Thus, when the λ value of agents increases and the problem becomes more relaxed, these algorithms provide better results (remember that the algorithms are ignorant of the λ value and visit the same parts of the search space).

[4] Recall that the number of phases within each cycle may vary between different algorithms

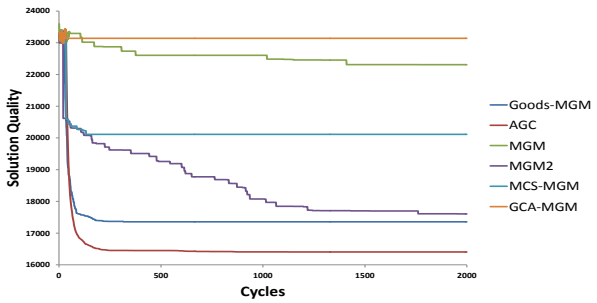

Figure 3. The aggregated solution cost when using the anytime mechanism and $\lambda = 0.4$ on regular graphs with $deg = 5$

The results also demonstrates that the local search procedure proposed for ADCOPs, namely MCS-MGM, GCA-MGM and ACLS are hardly affected by the length of the execution or the λ values used. This can be attributed to their quick convergence that was already reported in [4]. The anytime mechanism which is required for the coordination of feasible solutions is not effective for algorithms which are quick to converge.

Setup 2 - K-reg graphs: this setup included randomly generated graphs in which all agents shared the same degree $deg = 5$. The cost of a joint assignment by two constrained agents was randomly and uniformly selected in the range 0..99. This setup generates looser problems than the previous one in two senses:

- The cost of any joint assignment is uniformly sampled in the range 0..99 (instead of just 50% of the joint assignments), and the mean cost is therefore expected to be higher. This implies that the initial baseline cost of each agent is higher as well and that in general, more states become feasible outcomes. Hence the problem is looser in this sense.
- The degree of agents in this setup is lower than the (expected) degree of agents in setup 1 and hence less coordination is required for finding a feasible outcome.

Figure 3 presents the aggregated cost of agents when $\lambda = 0.4$. Although Goods-MGM and AGC still find the best feasible outcome (this time, it is AGC that finds the best results) one can see that the performance of MGM2 is significantly better. As more states become feasible, the benefits of combining the anytime mechanism with an exploratory algorithm greatly increases. This is also evident in Figure 4, presenting the aggregated cost for even looser problems. In this setup, the combination of an anytime mechanism with MGM2 yields the best results.

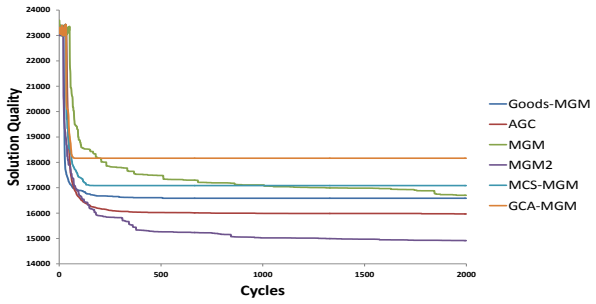

Figure 4. The aggregated solution cost when using the anytime mechanism and $\lambda = 0.8$ on regular graphs with $deg = 5$

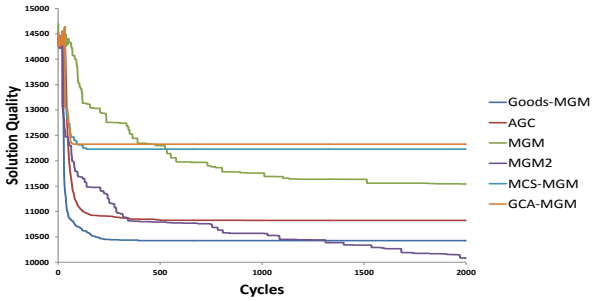

Figure 5. The aggregated solution cost using the anytime mechanism and $\lambda = 0.4$ on Scale Free networks

Setup 3 - Scale Free networks: in this setup a scale free network was built using the Barabási – Albert (BA) model. An initial set of 20 agents was randomly selected and connected. At each iteration of the BA procedure an agent was added and connected to 3 other agents with a probability that is proportional to the number of links that the existing agents already have. The constraint structure used was similar to the one used in the regular graphs setup, where the cost of a joint assignment was randomly selected in the range 0..99.

Although some agents in a scale free network are expected to be of high degree, most are not. That is, some parts of the network have local interactions which are expected to be looser and as a result the advantages of the modified anytime mechanism are more significant, as seen in Figure 5.

Similar to the previous experiments, increasing the λ value further will mostly benefit MGM and MGM2 but not Goods-MGM. This is explained by the different algorithmic approaches taken by these algorithms. While MGM and MGM2 attempt to minimize the total sum of costs and ignore the feasibility of an outcome, Goods-MGM focuses on the feasibility of a solution rather than its quality. As more states become feasible (looser problems) the probability that the low cost states visited by MGM and MGM2 will be marked by the anytime mechanism becomes higher.

7 Conclusion

A paradigm and a model for partial cooperation by agents solving a distributed global combinatorial problem is presented. A broad spectrum of cooperation levels can be represented by the use of a cooperation parameter λ. The model requires that a baseline global solution is first computed (e.g. a non cooperative proportional division or a Bayesian Nash equilibrium). This is followed by a cooperative procedure which is proven to produce Pareto improving global solutions over the non cooperative baseline. The proposed model does not rely on a specific precomputed baseline and uses the parameter λ (risk level) for determining the level of cooperation. The value of λ dictates the maximal personal cost that agents are willing to undertake in order to achieve a better global solution.

Asymmetric DCOPs [4] are used to define the distributed framework of the problem, where asymmetric constraints represent different valuations that agents have for various mutual outcomes. Two different methods for finding minimal cost solutions which also satisfy the personal baseline gain thresholds of agents are presented. The first method modifies the local search anytime mechanism of [19] to select the best global outcome among all considered outcomes which were approved by all agents. Thus, any local search algorithm combined with the mechanism is guaranteed to produce a valid solution. The second method designs an algorithm which only considers valid solutions. In the Asymmetric Gain Coordination (AGC) algorithm presented in Section 5.3 all states held by agents throughout the algorithm's execution are guaranteed to be feasible.

Our empirical evaluation included three different graph classes and different setups. It demonstrates that the combination of the anytime framework with the Goods-MGM algorithm (Section 5.2) on the one hand and the AGC algorithm on the other hand, produce globally high quality solutions when compared to other local search algorithms that use the anytime method. This is especially evident when the thresholds of agents are tighter (i.e. the level of cooperation of agents is lower).

REFERENCES

[1] I. Brito, A. Meisels, P. Meseguer, and R. Zivan, 'Distributed constraint satisfaction with partially known constraints', *Constraints*, **14**(2), 199–234, (2009).
[2] A. Gershman, A. Meisels, and R. Zivan, 'Asynchronous forward bounding', *J. of Artificial Intelligence Research*, **34**, 25–46, (2009).
[3] J. Grant, S. Kraus, M. Wooldridge, and I. Zuckerman, 'Manipulating boolean games through communication', in *IJCAI*, pp. 210–215, (2011).
[4] A. Grubshtein, R. Zivan, T. Grinshpoun, and A. Meisels, 'Local search for distributed asymmetric optimization', in *AAMAS*, pp. 1015–1022, (2010).
[5] Alon Grubshtein and Amnon Meisels, 'Cooperation mechanism for a network game', in *ICAART (2)*, pp. 336–341, (2011).
[6] Sergiu Hart, 'Adaptive heuristics', *Econometrica*, **73**(5), pp. 1401–1430, (2005).
[7] Matthew O. Jackson, *Social and Economic Networks*, Princeton University Press, 2008.
[8] Rajiv T. Maheswaran, Jonathan P. Pearce, and Milind Tambe, 'Distributed algorithms for dcop: A graphical-game-based approach', in *In PDCS*, (2004).
[9] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic theory*, Oxford University Press, 1995.
[10] Amnon Meisels, *Distributed Search by Constrained Agents: Algorithms, Performance, Communication (Advanced Information and Knowledge Processing)*, Springer, 2007.
[11] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, 'Adopt: asynchronous distributed constraints optimizationwith quality guarantees', *Artificial Intelligence*, **161:1-2**, 149–180, (January 2005).
[12] Dov Monderer and Moshe Tennenholtz, 'Strong mediated equilibrium', *Artificial Intelligence*, **173**(1), 180–195, (2009).
[13] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, 2007.
[14] A. Petcu and B. Faltings, 'A scalable method for multiagent constraint optimization.', in *Proc. IJCAI-05*, pp. 266–271, Edinburgh, Scotland, UK, (2005).
[15] Ola Rozenfeld and Moshe Tennenholtz, 'Routing mediators', in *Proc. IJCAI-07*, pp. 1488–1493, Hyderabad, India, (2007).
[16] V. Sankaranarayanan, M. Chandrasekaran, and S. J. Upadhyaya, 'Towards modeling trust based decisions: A game theoretic approach', in *ESORICS*, pp. 485–500, (2007).
[17] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nicholas R. Jennings, 'Decentralised coordination of mobile sensors using the max-sum algorithm', in *IJCAI*, pp. 299–304, (2009).
[18] M. Tennenholtz, 'Game-theoretic recommendations: some progress in an uphill battle', in *AAMAS (1)*, pp. 10–16, (2008).
[19] R. Zivan, 'Anytime local search for distributed constraint optimization', in *AAAI-2008*, Chicago, USA, (2008).
[20] R. Zivan, 'Can trust increase the efficiency of cake cutting algorithms', in *Proc. AAMAS-11 - short paper*, Taipei, Taiwan, (May, 2011).