Local Maxima in ADCOPs via Side Payments

Yair Vaknin
Ben Gurion University
Israel

Amnon Meisels
Ben Gurion University
Israel

ABSTRACT

Distributed constraints optimization problems (DCOPs) are composed of multiple constrained agents and their solution is an assignment of values by all agents that has the maximal global social welfare. DCOPs are known to be NP-Hard and therefore need incomplete search algorithms. Local search algorithms for DCOPs are incomplete distributed algorithms that start from some initial state and attempt to find improved solutions ([10, 13]). When the constraints have different values for the constrained agents, the problems are termed *Asymmetric DCOPs, or ADCOPs*. Standard local search algorithms, such as DSA or MGM, are not guaranteed to converge on ADCOPs. When convergence does happen, the solution is not necessarily an extremum of the global social welfare. This is true also for local search algorithms that were designed specifically for asymmetric DCOPs [3].

The present paper proposes an iterative local search algorithm for ADCOPs, that relies on the analogy between ADCOPs and multi-agents games. In each iteration of the algorithm agents can propose side-payments to their neighbors, in return to their choice of an assignment that is preferred by the agent offering the side payment. It is shown that the proposed protocol produces a behavior of the distributed algorithm that is an extension of the best-response dynamics in muti-agent games. These properties of the proposed Bidding Efficient Equilibria Contracts (BEECon) algorithm guarantee convergence towards a local optimum of the global social welfare. Extensive experimental evaluation on randomly generated ADCOPs demonstrates that convergence is fast and that the resulting solutions are of higher social welfare than those of the best former ADCOP local search algorithm.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Distributed Constraints, Side payments, Local search

ACM Reference Format:
Yair Vaknin and Amnon Meisels. 2021. Local Maxima in ADCOPs via Side Payments. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3486622.3493919

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493919

1 INTRODUCTION

In local search algorithms on distributed constraints optimization problems (DCOPs) agents coordinate their changes of assignments in neighborhoods, attempting to find assignments of higher global utilitarian social welfare [13, 14]. In general, these algorithms are not guaranteed to converge, nor do they necessarily find a solution of higher social welfare (hence the need for the anytime property [14]). When one moves on to asymmetric DCOPs [3], where the values of constraints are different among constraining agents, addressing the convergence of local search becomes more complicated [3]. Since agents do not know the constraint values of their neighbors (i.e., their constraining agents), they need to perform various kinds of checks of their combined constraint value depending on the specific algorithm [3, 4]. The asymmetry of the distributed constraints problem hinders also the guarantee of a local maximum of the utilitarian social welfare when (and if) the local search algorithm converges.

The present paper proposes an incomplete decentralized search algorithm for ADCOPs. The proposed algorithm uses side payments to enable ADCOPs agents to perform hill climbing and to find and stabilize a local maximum of the global utilitarian social welfare. Finding a local maximum of the social welfare for Asymmetric DCOPs is unique. As explained above, even when a standard local search algorithm, such as DSA [13], MGM [10], and MGM-MCS [3] converges when searching on ADCOP, the state is not guaranteed to be a local maximum of the global social welfare.

Asymmetric distributed optimization problems (ADCOPs) have constraints among agents that are represented by bi-matrices [3], each of a pair of values for an assignment of two constraining agents representing the cost or gain of one of the constrained agents. These constraint matrices are equivalent to normal form games among the constraining agents, expressing the costs (or gains) to each of the agents for selecting the specific value assignment (i.e., selecting a strategy to play). The game is assumed to have interactions among a limited number of agents, which is equivalent to assuming some underlying graph or social network. This compact general structure for multi-agent games has been termed graphical games in the past [7] and they are known to have a compact representation. For clarity, the term multi-agents games (MAGs) will be used through the rest of the paper interchangeably with ADCOP. Since the distributed search algorithm proposed by the present paper uses strategic reasoning by the agents that compute their transfers of payoffs, transfer functions can be thought of as *side payments* [6]. Side payments are contracted by the agents during the run of the algorithm, to be transferred for its final outcome [5, 6, 8, 9].

The proposed Bidding Efficient Equilibria Contracts (BEECon) algorithm uses a fixed order of all the agents. Each agent in turn may propose several possible changes of strategy. The responding agents (i.e., its neighbors) can sacrifice part of their payoff in order to convince the proposing agent to play a certain strategy. Their

proposed transfer of funds (i.e., side-payments) is observed as a binding contract. The inter-agent transfers of funds are used to ensure the convergence of the algorithm to a stable outcome of enhanced efficiency. The outcome resulting from the run of the BEECon algorithm is at least as efficient as the original outcome. It is guaranteed that the resulting outcome can be transformed into a stable state by the use of the transfers of funds that were contracted by the agents during the run of the algorithm. We compare the proposed algorithm to MGM-MCS, a local search algorithm specially designed for asymmetric DCOPs [3] and demonstrate the superior performance of the BEECon proposed algorithm by extensive experimental evaluation.

The next section presents distributed local search and specifically the MGM-MCS algorithm for ADCOPs. Section **Search with Side Payments** introduces side payments, their use in search for equilibria states and the proposed BEECon algorithm in detail. The correctness proofs for BEECon's main guarantees are presented in section **Correctness**. An extensive experimental evaluation comparing BEECon to the former local search algorithm MGM-MCS by using several performance measures is in section **Experimental evaluation**. In this section, we also justify our choice to take MGM-MCS as a reference point for new ADCOPs local search algorithms. Our conclusions are listed in the last section.

2 LOCAL SEARCH FOR ASYMMETRIC DCOPS

Distributed local search algorithms for DCOPs operate in synchronous rounds or iterations [10, 13]. On each iteration agents either randomly select a change of assignment or propose their best improving change of assignment to their constraining agents (i.e., their neighborhood). Agents respond by selecting the best offer in their neighborhood and if, depending on the specific algorithm, certain conditions are met the selected agent performs the proposed change of its assignment. Note that standard local search DCOP algorithms decide on the best change of assignment, in each neighborhood, to be performed. Algorithms differ from one another in the method of proposing a change of assignment (i.e., random for DSA [13]). Constraining agents in asymmetric DCOPs (ADCOPs) have different values (i.e., costs or gains) for the constraint connecting them. Consequently, when agents need to select a change of assignment during the run of an algorithm they need somehow to get additional information from their constraining agents.

Let us focus on a specific local search algorithm - Minimal Constraint Sharing MGM (MGM-MCS) [3]. It is a version of the classical Max Gain Message (MGM) algorithm [10], that is specifically designed for ADCOPs and is guaranteed to converge on asymmetric DCOPs. The MGM-MCS algorithm is presented below (Algorithm 1). On each iteration, all agents perform the while loop starting in line 2 of Algorithm 1. As in MGM, every agent computes its maximal cost reduction (line 14), publishes it to its neighbors (line 15), and acts on it (lines 16-21) if none of its neighbors can achieve a greater cost reduction (ties are broken using indices). An extra phase (lines 5-13) was added to the original MGM algorithm, to guarantee convergence on asymmetric DCOPs. On this phase, any agent i who suffers a large increase of its cost due to the last action of agent j, communicates its new cost to agent j. Next, both agents update their constraint table accordingly.

Algorithm 1 MGM-MCS

```
1:  value ← ChooseRandomValue()
2:  while no termination condition is met do
3:    send value to neighbors
4:    collect neighbors' values
5:    for neighbor A_n do
6:      Δ ← increase due to A_n's new value
7:      if Δ > A_n's last known LR then
8:        send constraint cost with A_n's new value
9:        change constraint cost with A_n's new value to 0
10:     end if
11:   end for
12:   collect neighbors' constraint updates
13:   update constraint with each of the neighbors
14:   LR ← BestPossibleLocalReduction()
15:   send LR to neighbors
16:   collect neighbors' LRs
17:   if LR > 0 then
18:     if LR > LRs of neighbors then
19:       value ← the value that gives LR
20:     end if
21:   end if
22: end while
```

MGM-MCS suffers from two weaknesses that are addressed by the proposed BEECon algorithm:

- Consider the 2-agents ADCOP (i.e., *game*) presented in Table 1, and let **(a,a)** be the initial assignment profile of the two agents. MGM-MCS would terminate with the final outcome **(a,a)**, because no agent is able to propose a local reduction to its cost. However, a deviation of agent **A** to the assignment **b** would decrease the cost of agent **B** (and the overall cost as well). This deviation which is part of a small local neighborhood, is not explored by a standard local search algorithm like MGM-MCS. In contrast, the proposed BEECon algorithm is guaranteed to terminate on a local extremum of the social welfare (Proposition 4.3).
- Consider now the 2-agents ADCOP in Table 2 and again let **(a,a)** be the initial assignment profile. On iteration 1, agent **A** proposes its best possible local reduction (lines 14-15 of Algorithm 1). The proposal is accepted, and agent **A** changes its assignment to **b**. Agent **B**, who suffers a large loss as a result, reports its constraint cost to agent **A** (lines 7-8) on iteration 2. Agent **A** updates its constraint table accordingly, and returns to assignment **a**. In this example MGM-MCS terminates in a local maximum, but visits a certain state more than once. Consequently, convergence may require larger amounts of processing and of communication. The extensive experimental evaluation presented below compares the number of messages until termination required by both MGM-MCS and BEECon.

3 SEARCH WITH SIDE PAYMENTS

The BEECon search algorithm is iterative, where each iteration goes over all agents in a predefined order. Each agent in its turn

A\B	a	b
a	10,10	10,10
b	10,0	10,10

Table 1

A\B	a	b
a	10,0	10,10
b	9,10	10,10

Table 2

A\B	a	b
a	10,20	0,0
b	20,5	30,0

Table 3

A\C	a	b
a	30,40	0,0
b	30,50	0,0

Table 4

proposes to its neighbors (i.e., its constraining agents) strategy changes that it can perform (e.g., change of its assignment value). The proposing agent then receives from its neighbors response messages indicating the transfer of funds (i.e., side payments) they are willing to offer for each of the proposed changes of strategy. Each of the offered side payments relates to one of the proposed changes. The side payments that are offered by the neighboring agents are the basis for the proposing agent in selecting a better gaining strategy for itself. In this section and in the **Correctness** section, we refer to BEECon as a utility-maximizing algorithm. In our **Experimental Evaluation**, BEECon is adapted to minimize the cost of the environment.

3.1 The BEECon Algorithm

The BEECon algorithm is composed of two consecutive stages.

Starting from an initial outcome v, the main stage iterates over all agents until it converges to an outcome v^*. The efficiency of outcome v^* is proven to be at least as high as the efficiency of the initial outcome v. The term *efficiency* is used here to denote the sum of gains of all agents, that is, the global social welfare. Outcome v^*, although unnecessarily stable per se, is stabilized by the side payments contracted by the participating agents. A set of transfers that can be used as side payments among the agents to enforce a pure Nash equilibrium (PNE) is computed by the agents during the iterations. These side payments form binding contracts among the agents, computed by the neighboring agents during each step. Thus, the proposed BEECon algorithm is an iterative method for searching for both an efficient outcome and the side payments that can guarantee its stability. Each iteration ends with an outcome and side payments that were computed during the iteration. The final outcome is reached when for a complete iteration no agent offers a change of its strategy in order to achieve a greater utility. In the second stage of the proposed BEECon algorithm, the transfers of payoffs among the agents are applied according to the computed contracts so that the outcome v^* is transformed into a stable state (i.e., a PNE).

3.1.1 Main Stage. All agents perform the main stage (Algorithm 2) in a fixed order, and each agent in turn executes the function **onTurn()**. We will term the agent performing function **onTurn()** the *current agent*. During the execution of Algorithm 2 the agents agree on contracts with their neighbors which are the incentives for the agents to select their strategies. In order to decide on the choice of action, the current agent must take into consideration not only its own utility but also the bids (contracts) offered by its neighbors (lines 7-11 of Algorithm 2). Contracts are dissolved only due to the decision of an agent to change its strategy. When an agent updates its neighbors about a change in its strategy, all old contracts (related to its former strategy) concluded between itself and its neighbors are nullified.

In response to the message of the current agent proposing to change its strategy (line 3), each neighbor decides upon the payoff transfers it wishes to sacrifice in order to convince the current agent to select any of its proposed changes of value. To this end it runs procedure **reply()**. This procedure considers the responding agent's current strategy, and matches a bid for each of the strategy proposals made by the *current agent*. Each bid represents the relative gain (lines 5-6 of **reply()**) of the responding agent from the proposed strategy, given its current strategy, in comparison to the least beneficial possible strategy of the *current agent*. The gain of the responding agent from the least beneficial strategy of *current agent* is denoted s_{min} in line 4 of the **reply()** procedure. These offers (bids) are sent back to the current agent and considered as new contracts only in case the agent selects the strategy for which the specific transfer was offered. After the *current agent* receives the responses from all of its neighbors (line 4), it iterates over its proposed assignments (line 7) and computes the sums of its neighbors' offers for each one (line 8). The current agent compares its overall utility from each of the options to the utility of its current strategy (line 9) and deviates to a new strategy (i.e., assignment) only if it gains from the deviation. The selection of a new strategy by the *current agent* is a selfish (i.e., rational) decision.

The run of the algorithm terminates when no agent can offer a new deviation.

Algorithm 2 defines the communication protocol among the agents and the decision-making procedure for changes of strategies. Procedure **reply()** determines the bids each agent is willing to offer in response to the deviations proposed by the *current agent*. Line 2 of Algorithm 2 leaves the exact choice of explored assignments unspecified and enables several versions of the BEECon algorithm. These versions differ by the part of the domain that the *current agent* offers as potential new assignments. Through the rest of the paper, V_i^s is simply taken to be V_i, i.e., the *current agent* explores its entire domain. Another natural choice of V_i^s is $V_i^s = \{v_i' : u_i(v_i', v_{-i}) > u_i(v_i, v_{-i})\}$, under which the *current agent* performs a less extensive search, and only explores deviations that are perceived as beneficial (from the perspective of the *current agent*).

Let us follow the run of the BEECon algorithm on the example in Tables 3,4. The tables represent a MAG of three agents **A, B, C**, where each agent holds a variable with two possible values (**a,b**) in

Algorithm 2 Bidding Enhanced Efficiency Contracts

onTurn()

1: **let** $v_i \leftarrow$ the current strategy of agent i
2: choose $V_i^s \subseteq V_i$
3: **send(choice**, $V_i^s \cup \{v_i\}$) to all $j \in N_i$
4: **receive** $T_{j,i}^s$ **foreach** $s \in V_i^s \cup \{v_i\}$ from all $j \in N_i$
5: **let** $\mathbb{T} \leftarrow \sum_{j \in N_i} T_{j,i}^{v_i}$
6: $v_i' \leftarrow v_i$
7: **for all** $s \in V_i^s$ **do**
8: **let** $T^s \leftarrow \sum_{j \in N_i} T_{j,i}^s$
9: **if** $u_i(s, v_{-i}) + T^s > u_i(v_i, v_{-i}) + \mathbb{T}$ **then**
10: $v_i' \leftarrow s$
11: $\mathbb{T} \leftarrow T^s$
12: **end if**
13: **end for**
14: **send(update**,v_i') to all $j \in N_i$

reply(choice, V_i^s**)**

1: **denote** $v_1^s, \ldots, v_{|V_i^s|}^s$ as the strategies in V_i^s
2: **for all** $v_k^s \in V_i^s$ **do**
3: **let** $s_{v_k} \leftarrow u_j(v_1, \ldots v_k^s, \ldots v_n)$
4: **end for**
5: **let** $s_{min} \leftarrow min\{s_{v_1}, \ldots, s_{v_{|V_i^s|}}\}$
6: **let** bids $\leftarrow [s_{v_1^s} - s_{min}, \ldots, s_{v_{|V_i^s|}} - s_{min}]$
7: **return** bids

the domain. Table 3 represents the constraint table between agents **A,B**, and Table 4 represents the constraint table between agents **A,C**. Agents **B,C** are not constrained:

Example 3.1. Let the initial outcome be v = (**a, a, a**), and the order of agents be lexicographic. On its turn, agent **A** explores a possible deviation from strategy **a** to **b**. It sends both its current strategy and the optional deviation to its neighbors. Agent **B** detects the strategy of agent **A** which is least beneficial for itself (strategy **b**) and computes its contracts $[s_a - s_b = 15, s_b - s_b = 0]$. In words - agent **B** is willing to sacrifice 15 to make agent **A** select strategy **a**, and 0 to make it select strategy **b**. Agent **C** computes its gains and comes up with the contracts $[0, 10]$. Agent **A** sums up the bids and gets $[15, 10]$. The utility of agent **A** from strategy **a** is $15 + u_{\mathbf{A}}$(**a, a, a**) = 55, while its utility from strategy **b** is $10 + u_{\mathbf{A}}$(**b, a, a**) = 60. Consequently, agent **A** deviates to strategy **b** and updates its neighbors.

Next, agent **B** explores a deviation to strategy **b** and receives the contracts $[0, 10]$ from agent **a**. The utility of agent **B** from strategy **a** is $0 + u_{\mathbf{B}}$(**b, a, a**) = 5, while its utility from strategy **b** is $10 + u_{\mathbf{A}}$(**b, b, a**) = 10. Agent **B** deviates to strategy **b** and updates its neighbors. At this point no agent can find a beneficial deviation and the final outcome is v^* = (**b, b, a**).

Regarding the privacy of the agents, observe that BEECon agents never expose their absolute utility for any given strategy profile. They only expose their **relative** utility for some pair of strategies (v_i, v_j) compared to their utility in some other pair $(v_{i'}, v_j)$. This is different than MGM-MCS agents, where agents may reveal the entire information regarding some (v_i, v_j).

4 CORRECTNESS

Proposition 4.1. *The gain of a BEECon agent deviating from v_i to $s \in V_i^s$ equals the change in the global social welfare.*

Proof. The change in the global social welfare is

$$u_i(s, v_{-i}) + \Sigma_{j \in N_i}(u_j(s, v_{-i})) -$$
$$u_i(v_i, v_{-i}) - \Sigma_{j \in N_i}(u_j(v_i, v_{-i}))$$

while the gain of the deviating agent is

$$u_i(s, v_{-i}) + \Sigma_{j \in N_i} T_{j,i}^s -$$
$$u_i(v_i, v_{-i}) - \Sigma_{j \in N_i} T_{j,i}^{v_i} =$$
$$u_i(s, v_{-i}) + \Sigma_{j \in N_i}(u_j(s, v_{-i}) - s_{min}^j) -$$
$$u_i(v_i, v_{-i}) - \Sigma_{j \in N_i}(u_j(v_i, v_{-i}) - s_{min}^j) =$$
$$u_i(s, v_{-i}) + \Sigma_{j \in N_i}(u_j(s, v_{-i})) -$$
$$u_i(v_i, v_{-i}) - \Sigma_{j \in N_i}(u_j(v_i, v_{-i}))$$

□

Corollary 4.2. *A multi agent search for stable solutions that uses the contracts of Algorithm 2 satisfies the condition of an exact potential game.*

Proof. Follows immediately from observation 4.1. □

Proposition 4.3. *Algorithm 2 converges.*

Proof. Corollary 4.2 guarantees the condition of an exact potential game for a search that includes side payments. Additionally, BEECon agents select their strategies in a best-response manner. Best-response search converges on exact potential games. □

Proposition 4.4. *Algorithm 2 converges to a local maximum of the global social welfare.*

Proof. When algorithm 2 converges, no agent can increase its utility. From Proposition 4.1, no agent can deviate from its strategy such that the global social welfare increases. □

Proposition 4.4 indicates the unique guarantee of BEECon algorithm. Generally, local search terminates when no agent can increase **its own utility**. In contrast, BEECon terminates when no agent can increase the **global social welfare**.

5 EXPERIMENTAL EVALUATION

A recent study ([12], 2018) compared several incomplete ADCOPs algorithms on scale-free networks. The best-performing algorithm (among Max_Sum [2], DSA [13], MGM-MCS [3], and more) in terms of utilitarian social welfare was MGM-MCS. Consequently, BEECon is evaluated here against the best performing algorithm to-date, MGM-MCS. Experiments were conducted on both randomly generated ADCOPs and on scale free networks (e.g., Barabasi-Albert [1]) of varied sizes and densities. Barabasi-Albert networks can be categorized by a parameter m - the number of nodes that connect each new node to the already existing network during its construction [11]. The domain size of the agents in all of our experiments is 10, and the utilities in the game matrices were randomly and uniformly selected in the range [0,100]. For each configuration (type

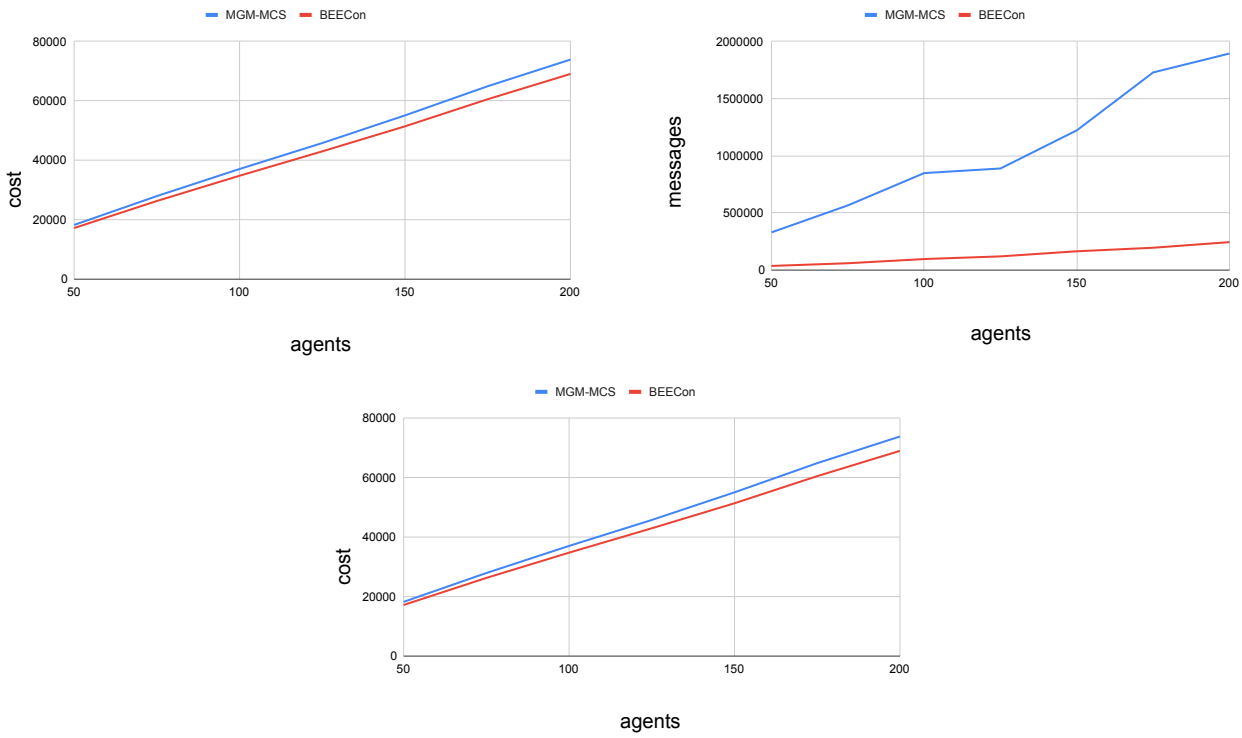

Figure 1: random networks with 10 neighbors per agent

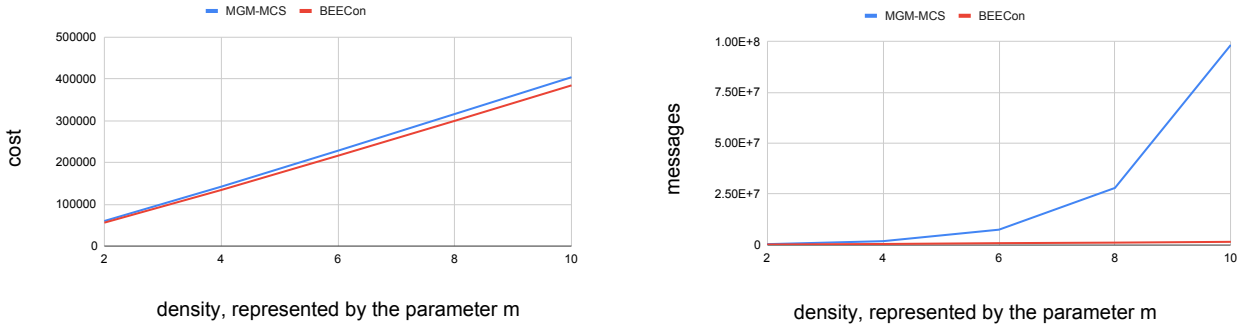

Figure 2: scale-free networks with 500 agents

of network, size, and density) the results are averaged over 30 runs on randomly generated networks.

5.1 Experimental results

Figure 1 demonstrates the much faster convergence of BEECOn towards an outcome of greater utilitarian social welfare on random networks with 10 neighbors on average per agent. On randomly generated networks with 50 agents, it takes 1/9 of the messages (or 1/50 of the iterations) for BEECon to terminate with an outcome whose cost is 5.7% lower. On larger games with 200 agents, BEECon uses 1/8 of the messages and presents an improvement of 6.5% in the overall cost.

Figure 2 emphasises the efficiency of the proposed search over MGM-MCS on scale-free networks. The number of messages exponentially grows with m under MGM-MCS, and linearly under BEECon (this result is reflected also in the iterations graph, which was omitted). The utilitarian social welfare is 6.5% better for BEECon when $m = 2$. On dense networks ($m = 10$), the improvement is about 5%.

6 CONCLUSION

Current ADCOPs local search algorithms do not guarantee convergence to a local extremum of the global social welfare [3]. The present study focuses on this problem and proposes a new class

of local search algorithms. The Bidding Enhanced Efficiency Contracts (BEECon) algorithm uses side-payments among constraining agents as incentives for assignments that are preferred by the agent proposing a side payment. It is a simple protocol that enables agents to efficiently converge to a local maximum of the global social welfare. One can relate to the proposed BEECon algorithm as a protocol for selfish agents that pursue assignments from which they benefit most. Alternatively, one can describe the BEECon algorithm as run by agents who cooperatively share some information about their constraint costs (using bidding), in order to assist the current agent in selecting the strategy that best increases the social welfare. Most importantly, in an extensive experimental evaluation on both random uniform ADCOPs and on ADCOPs with scale-free networks, BEECon produces better results than the best former local search algorithm. BEECon converges faster than MGM-MCS on all of the problems in the experimental evaluation. The natural and interesting next step in this direction, which we currently pursue, is the design and implementation of a new version of the algorithm that is robust for strategic bidding (for side payments).

REFERENCES

[1] Albert-Laszlo Barabasi and Reka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.

[2] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm.. In *AAMAS (2)*, Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons (Eds.). IFAAMAS, 639–646. http://dblp.uni-trier.de/db/conf/atal/aamas2008-2.html#FarinelliRPJ08

[3] Tal Grinshpoun, Alon Grubshtein, Roie Zivan, Arnon Netzer, and Amnon Meisels. 2013. Asymmetric Distributed Constraint Optimization Problems. *JAIR* 47 (2013), 613–647.

[4] Alon Grubshtein, Roie Zivan, Tal Grinshpoun, and Amnon Meisels. 2010. Local search for distributed asymmetric optimization. In *AAMAS*. 1015–1022.

[5] Matthew O. Jackson. 2008. *Social and economic networks*. Vol. 3. Princeton University Press Princeton.

[6] Matthew O Jackson and Simon Wilkie. 2005. Endogenous games and mechanisms: Side payments among players. *The Review of Economic Studies* 72, 2 (2005), 543–566.

[7] Michael J. Kearns. 2007. Graphical games. In *Vazirani, Vijay V.; Nisan, Noam; Roughgarden, Tim; Tardos, Éva (2007). Algorithmic Game Theory*. Cambridge University Press, 159–178.

[8] Vadim Levit, Zohar Komarovsky, Tal Grinshpoun, and Amnon Meisels. 2015. Tradeoffs between Incentive Mechanisms in Boolean Games. In *IJCAI*. 68–74.

[9] Vadim Levit and Amnon Meisels. 2018. Distributed Constrained Search by Selfish Agents for Efficient Equilibria. In *Proc. 24th Intern. Conf. Princ. Pract. Const. Prog. (CP-18)*. Lille, France, 707–24.

[10] R. T. Maheswaran, J. P. Pearce, and M. Tambe. 2004. Distributed Algorithms for DCOP: A Graphical-Game-Based Approach. In *Intern. Conf. Parallel Distrib. Comp. Sys. (PDCS)*. 432–439.

[11] Francisco C. Santos and Jorge M. Pacheco. 2005. Scale-free networks provide a unifying framework for the emergence of cooperation. *Physical Review Letters* 95, 9 (2005), 098104.

[12] Cornelis Jan van Leeuwen and Przemyslaw Pawelczak. 2017. CoCoA: A Non-Iterative Approach to a Local Search (A)DCOP Solver.. In *AAAI*, Satinder P. Singh and Shaul Markovitch (Eds.). AAAI Press, 3944–3950. http://dblp.uni-trier.de/db/conf/aaai/aaai2017.html#LeeuwenP17

[13] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence* 161:1-2 (January 2005), 55–88.

[14] R. Zivan, S. Okamoto, and H. Peled. 2014. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence* 212 (2014), 1–26.