An Optimal Distributed Constraint Optimisation Algorithm for Efficient Energy Management

Yoseba K. Penya
Department of Information Systems
Vienna University of Economics and
Business Administration, Austria.
yoseba.penya@wu-wien.ac.at

Nicholas R. Jennings
School of Electronics and Computer Science
University of Southampton
Southampton, UK.
nrj@ecs.soton.ac.uk

Gustaf Neumann
Department of Information Systems
Vienna University of Economics and
Business Administration, Austria.
gustaf.neumann@wu-wien.ac.at

Abstract

The deregulation of the electricity industry in Europe has created a number of marketplaces in which producers and consumers can operate in order to more effectively manage and meet their energy needs. To this end, in previous work we introduced a new market design where consumers market the supply of the energy they need in a number of simultaneous reverse combinatorial auctions. To do this efficiently, each consumer needs to coordinate and optimise the use of its various appliances to achieve the best possible consumption schedule according to the received bids. In particular, this paper models each client's constellation of devices as a distributed constraint optimisation problem (DCOP) and we develop a novel optimal algorithm (COBB - Constraint Optimisation By Broadcasting) for solving it. We also evaluate COBB, prove its optimality and show how it outperforms other comparable algorithms.

1. Introduction

The deregulation of energy has changed the face of European electricity markets. From the traditional national monopolies has emerged a marketplace model where customers can choose the providers that best fit their needs. In this context, in [10] we redefined market relationships to help clients and utility companies (UCs) benefit from the new situation. The model we developed is based on a system of simultaneous reverse combinatorial auctions that allow customers to drop the cost of their electricity consumption and allows providers to foresee the demand they have to cope with.

However, there is still an aspect of this solution that demands more attention. In fact, we presented the general architecture of the energy market and an algorithm to find an optimal allocation according to the received bids and hourly needs. But the system lacks a procedure that coordinates these needs to provide an optimal (cheapest) solution. That is, a method that brings a consumer's appliances to work together in order to achieve the most convenient consumption schema (on a per consumer basis). In this way, the algorithm introduced in [10] would become the *fitness* (or utility) function invoked in the system to evaluate each possible global consumption schema.

Moreover, assessing one of these complete solutions would be a trivial task if UCs bid by offering a simple tariff (e.g. x €/Kilowattt from 8 to 9 am, y €/Kilowattt from 10 to 11 am, etc.). Yet in this case, UCs would lose the possibility of predicting and controlling the demand by offering discount for bundles (e.g. if you consume from 6 to 7 am and from 10 to 11 pm, you get a 10% discount). Therefore, all members have to be integrated in the optimisation process, since the individual decisions and acts affect the outcome of the entire system because a single device consuming or not at a certain time may involve getting a discount at another time. In other words, this is (yet) another approach to the so-called scheduling problem, this time applied to energy management. The goal here is to optimise the constraints that each device sets when consuming. The optimal global consumption plan of the system (e.g. flat, home or factory) is obtained when each device uses the energy in the cheapest possible time it cans. These characteristics, together with the fact that the knowledge about each device's alternatives of consumption is local, leads us to set out the problem as a distributed constraint one (DCP). DCPs

have been successfully applied to a wide range of problems, from resource allocation in a communication network [1] to distributed scheduling [13].

Further, our problem domain falls within the category of DCPs, distributed constraint optimisation problems (dCOP), where the constraint(s) can only be *optimised* (in opposition to those problems where they can be *satisfied*, called distributed constraint satisfaction problems, dCSP). Unfortunately, current optimal dCOP algorithms cannot be applied to our problem domain. First, the existence of discounts demands algorithms that work with complete solutions (i.e each energy consuming task has certain time to be executed). Second, supply-function bidding prevents linearity in the solution space and therefore, heuristics cannot be used (as explained in section /refevaluation). We address both shortcomings in the novel dCOP algorithm Constraint Optimisation by Broadcasting (COBB), by combining a greedy tactic with broadcast (so all agents are involved in deciding which is the new most promising solution to be processed).

Against this background, this paper advances the state of the art in two main ways. First, we introduce COBB, the only existing optimal algorithm to solve the distributed constraint optimisation problem described above. Furthermore, we prove COBB complete and optimal, and show how it outperforms other comparable algorithms. Second, COBB allows us to complete the description of a general market-based framework to model and solve the energy management problem in a more efficient way than another current ones. The remainder of the paper is organised as follows: section 2 describes our problem domain: the electricity market. Section 3 presents the COBB algorithm and section 4 its evaluation. Finally, section 5 discusses related work and section 6 draws our conclusions and the avenues of future work.

2. Electricity Market Design

As explained in the introduction, the deregulation of European electricity markets hat increased its competitiveness and dynamism by introducing new players that increase the offer. Solving such a system efficiently demands integrating the search of both optimal consumption schedule and optimal suppliers' allocation (this is, optimal auction clearing: who supplies how much and when). The consumption schedule should be adapted to take advantage of the most convenient tariffs and the suppliers should be chosen taking the consumption plan into account. Thus, both aspects are interdependent. In this way, we dealt with the clearing of the auction in ([10]) and hereby we explain how to achieve an optimal consumption schedule. But first, let us describe the features of the electricity market in which this optimisation must be performed.

A very simple application of the deregulation principle would allow to choose the UC with the cheapest tariff. However, why choose one? There is no technical problem on being supplied by more than one company simultaneously ([10]), so the election becomes a Knapsack (and thus, NP-Hard) Problem ([4]): finding the best allocation of x Kilowatts of electricity among the different providers and their tariffs.

Still, this model is just favourable for the customers, transformed into auctioneers that sell their future consumption to the highest bid (i.e. cheapest tariff). Combinatorial auctions are those where bidding for a group of items is typically valued differently from bidding separately for each of the constituent items. They maximise the revenue for the auctioneer ([11] but also allow UCs to influence clients' consuming behaviour. Bidding for a bundle (for instance, consuming at 9 am but *also* consuming at 11 pm is rewarded with a 10% discount), encourages the client to shift part of its consumption to 11 pm (off-peak and thus, favourable for the UC). This is an extension of the traditional "expensive day - cheap night" energy pricing system. Moreover, giving a *monolithic* tariff (x €/Kilowattt from 6 to 7 am, etc.) does not represent two very important dimensions for UCs: the number of clients consuming simultaneously and the quantity consumed by each one. This is, the price does only approximately depend on how many clients consume simultaneously (by setting peak prices higher). And, in case it staggers the tariff according to the amount of energy consumed, it expresses it in a coarse manner (by setting a threshold above which it becomes more expensive). We try to remove these shortcomings by allowing to bid with linear supply functions ([12, 3, 10]). This policy helps consumption be spread among many energy providers and therefore, furthers away the risk of a blackout.

The disadvantage of this approach is the complexity of clearing a combinatorial auction (i.e. finding the wining bid or the amount allocated to each winner), increased by using supply function bidding. This problem has been tackled in [2] with an algorithm that runs in polynomial time (but is not guaranteed to find the optimal solution), in [3] with an optimal algorithm, and in [10], a less complex optimal clearing algorithm tailored to electricity supply functions. Furthermore, it is not enough to find the optimal allocation of one client's consumption if her energy consumers, appliances or machines, cannot follow it. To this extent, devices can be divided in three categories regarding whether they can control their own consuming behaviour [9]: *Active*, those who can adapt their consumption schedule (e.g. a dishwasher that, switched on in the morning looks for the cheapest time until the afternoon to wash the dishes), *informative*, those who cannot control their consumption but at least may issue a prognosis (e.g. a fridge that knows how much energy will it need in the next hours to keep the same temperature) and *non-informative*, neither able to issue prognoses nor to adapt their consumption schedule (e.g. a vitroceramic hob that must start working on demand). With these premises, the goal of this work is to bring active devices to work together, take into account the prognoses of informative devices, and achieve an optimal consump-

tion plan (meaning cheap for the customer and predictable in beforehand for the UC).

3. The COBB Algorithm for Energy Management

With this market design in mind, the next step is to design an optimal algorithm that finds the cheapest way to consume the energy needed in the system. To this end, we have chosen an architecture with two main parts. First, the energy market involving customers and suppliers. The arrangement of customers' electricity supply from multiple UCs should be achieved by having contracts that specify the provision of an amount of energy for a certain period of time (say one hour). These contracts should not necessarily be exclusive and, thus, customers may achieve agreements with different companies for the same hour if this is the best thing to do. Finally, we assume customers auction, on a daily basis, their next 24 hours consumption divided into 24 items (representing one hour each). They subsequently receive bids from the UCs and make their decision for the next 24 hours. The second part is the multi-agent system corresponding to each private customer. It executes the algorithm that finds the optimal schedule, meaning which devices consume how much and, designed by the fitness function, when and which UCs will provide how much energy and when. For the sake of simplicity, each agent solely represents one consuming task. Moreover, there exist constraints between any pair of agents (since any local decision about consuming or not affects the whole system) and we suppose all agents interconnected. In opposition to the simultaneous auction market, where bidders *compete* to be selected, the agents of the dCOP system *collaborate* to achieve the common goal (the cheapest possible consumption).

3.1. The Algorithm

Since all possible solutions must be analysed, we have chosen a brute-force greedy strategy to develop the COBB algorithm (Constraint Optimisation by Broadcasting), detailed in Figure 1. Basically, it is a distributed constraint optimisation algorithm where agents broadcast their values and choose the answer proposing the best solution to continue with the algorithm. Note that the COBB always starts with a current solution as a basis for the comparison, so each local change automatically produces a new solution (that can be better or worse than the current one).

The greedy strategy allows COBB to always choose the best available solution. This tactic, however, could lead it to eventually get stuck in a local maximum but the fact of broadcasting (i.e. including *again* all the other agents in the next step) helps preventing it. Essentially, COBB is a recursive algorithm that in each recursive call does the following. First, improve the solution received with local changes. Then, for each possible local change (sorted out descending) broadcast the solution and finally, compare each answer (sorted out descending) with the current best solution and keep it in case it is better.

```
COBB(candidate_solution) {
  new_sol_list = improve (candidate_solution)
  if (new_sol_list != null) {
    foreach new_solution in new_sol_list {
      answers = broadcast (new_solution)
    }
  } else {
    answers = broadcast (candidate_solution)
  }
  sort_desc (answers)
  foreach solution in answers {
    if (fitness(solution) < fitness(best_solution)) {
      best_solution = solution
    } COBB(solution)
  }
}
```

Figure 1. The COBB algorithm

3.2. Worked Example

Let us illustrate this process with an example. Suppose an agent A starts the scheduling process: it takes the current solution and calculates possible improvements by selecting a different value for its own variable (e.g. "if the task can be performed at 9 or 10 am, and the current solution includes it's task at 9 am, the agent will asses the whole solution but with it's task at 10 am"). Then, it will broadcast the cheapest of both solutions. The other agents will afterwards carry out exactly the same process that agent A has done so far: calculate whether any local change improves the solution and reply. Further, Agent A chooses the cheapest (say best) solution received and restarts the process until no agent responses; in that case it will come back to the second best solution and continue as usual. The recursive nature of the algorithm assures that, given enough time, all possible solutions are processed and, therefore, the algorithm always finds the optimal solution.

3.3. Properties

The complexity of COBB is $2^n \cdot m$, where n is the number of agents (or tasks) and m the average number of alternatives for a task to be placed. In our system, the minimum allocation unit is the hour and all tasks can be placed at 1 am, 2 am, 3 am ... 12 pm. Thus, in a worst case scenario where tasks can be placed all over the day, m will be 24 and the complexity $2^n \cdot 24$. Finally, it is not possible to skip some candidate solutions heuristically since the discount for combinations and the linear supply function bidding requires that all possible solutions are processed. For

instance, at first sight it can seem economical to place agent A's consumption at 11 instead of at 12, if all bids submitted to 11 are much cheaper than those to 12. However, they may exist discounts rewarding the consumption on 12 that make it, at last, cheaper.

4. Evaluation of the COBB algorithm

As detailed in section 5, there exist other optimal algorithms for distributed constraint optimisation problems but, unfortunately, the dimensions of the application domain are not the same. Both ADOPT [5] or SynchBB [6], for instance, require a framework including variables with discrete values and binary or ternary cost functions and a kind of linearity in the solutions that makes heuristic work (either best-first search in ADOPT or branch and bound in SynchBB): this is, the capacity of abandon some solutions before processing them (i.e. being able to determine whether a solution is better or worse *before* evaluating it). For instance, in a similar problem domain but without supply-function bidding and bundle discounts, an algorithm could use any heuristic to directly choose the cheapest consumption alternative for a certain task before processing the rest. It would know all other variants worse *for sure*. This is not the case of our domain. A task consuming or not at a certain time slot may involve choosing one UC for supplying more or less, or not at all, at another time slot. And this fact cannot be foreseen. Moreover, both algorithms work with partial solutions, whereas our problem demands working with complete solutions (because of the discount combinations and the linear supply bidding). Therefore, in order to illustrate the pros and contras of the COBB algorithm and translate it into a framework where it can be compared to some counterparts, we will use a well-known problem modeled as a DCSP: the n-queens problem.

4.1. The N-Queens Problem as a DCSP

The n-queens problem has been traditionally a paradigm of combinatorial problems. We have chosen it since, as in our problem, there exist constraints between any pair of agents (say queens). The challenge consists in, having an n x n draughtboard, placing n queens so that no two queens threaten each other. Thus, no two queens can be on the same row, column or diagonal. If the problem (modeled as a DCP) is finished when one combination is found, where no queen is threaten, then it is a distributed constraint *satisfaction* one. In case the objective is to list all combinations, where no queen is threaten, then it is a distributed constraint *optimisation* problem. We will use it as a DCSP; therefore, the goal is to find just one solution. Furthermore, we have chosen two classical and simple dCSP Algorithms that could be adapted to our domain (by making them work with whole solutions and not partial ones) and are, therefore, counterparts.

4.2. Algorithm Execution Comparison

Figure 2 shows the execution steps followed by one of the algorithms presented in [17] (an asynchronous weak-commitment search), until no queen is threatened. Having four queens, there are subsequently four agents, each one having a variable with values from 1 to 4 (represented as a row in the draughtboard). The start situation for this example is with all variables having same value 1 (i.e. all queens lined up on the left column).

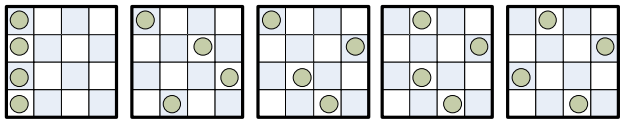

Figure 2. Example of the Asynchronous Backtracking algorithm execution ([17])

Figure 3 presents the execution steps of the COBB algorithm, starting with the same initial situation. In each cycle, COBB first considers local changes and then broadcasts the most convenient of the solutions obtained (with these local changes). In this case, the possible local changes are having the first queen in position 2, 3 or 4 of the first row. Position 2 is still violating a constraint (threatened in the diagonal by queen in row 2) and therefore, it chooses position 3 (4 would be also acceptable) and broadcasts the solution, being the one depicted in the middle draughtboard the best of the answers received (the queen of the third row gets value 4). In the next cycle, it does not issue any local change and broadcasts the solution as it is, obtaining a combination where no queen is threatened.

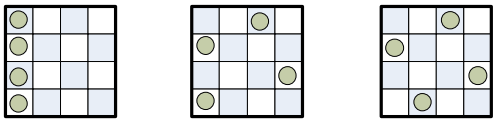

Figure 3. Example of the COBB algorithm execution

Figure 4 illustrates the execution of the asynchronous weak-commitment search ([17]), faster than its counterpart, depicted in figure 2. The start situation is this time slightly different, with three constraints violated: queen of row 1 in the same diagonal as queen of row 4, and this one in the same column as queen of row 2. Finally, figure 5 depicts the execution of COBB starting from the same initial situation as in Figure 4. The COBB algorithm is faster again.

4.3. Performance Comparison

The execution steps shown in Figures 2 to 5 show just some examples about how COBB's tactic helps to faster find the solution. For the evaluation, we have tested COBB with the distributed n-queens problem varying from n 10

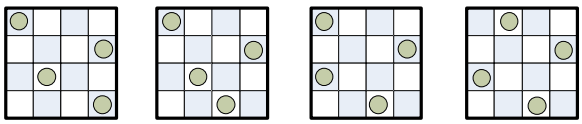

Figure 4. Example of the Asynchronous Weak-Commitment algorithm execution ([17])

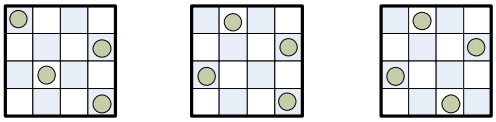

Figure 5. Example of the COBB algorithm execution

to 50 (as in [16]). The results are summarised in table 1. For each n, we have averaged the results of testing 100 randomly generated different start situations. One cycle corresponds to a loop of the COBB algorithms or, in case of the other two, a series of agent actions, in which an agent recognises the state of the world, decides its response to that state and communicates it [16]. We also assume that a message sent at time t arrives at time $t + 1$, so we can analyse the performance in terms of cycles needed to find an optimal solution. Finally, following [16] we have set a time limit for solving the problem (1000 cycles), so $ratio$ expresses the average of runs that successfully completed the problem on time.

Table 1. Performance comparison in the Distributed N-Queens Problem).

	n	10	50
Asynchronous backtraking [16]	ratio	100%	50%
	cycles	105.4	325.4
Asynchronous weak-commitment [16]	ratio	100%	56%
	cycles	41.5	59.1
COBB	ratio	100%	100%
	cycles	15.36	35.5

As it can be seen, COBB clearly outperforms both algorithms. It is not only faster but also most efficient, since it always found the solution without exceeding the time limit. The reason is that, as shown in the previous section, the greedy tactic helps COBB to quickly come to better solutions and broadcasting enables it to escape from local minima, since all agents participate in each decision. This is, in the other algorithms the view of problem space in each loop is restricted to one agent and it's neighbour. In COBB, this restriction disappears with the broadcast, so any agent can contribute to a better solution.

The cost of broadcasting can be, however, prohibitively high in large systems. In case of a DCSP, the network overhead is alleviated by the fact that the algorithm stops as soon as one solution is found (so the number of messages broadcasted is compensated by the messages that other algorithms interchange in the cycles where COBB is already finished). It does become a problem when the algorithm has to analyse all the possible solutions. The next section explains how to lighten this shortcoming.

4.4. Loosely-coupled Neighbour Networks

Using broadcast in the COBB is a double-edged sword. On the one hand, it helps find a very good solution (if not the best) very fast but on the other, it requires a high amount of exchanged messages. There are, however, some of these messages that do not need to be sent. Although all agents are virtually neighbours (i.e. can contact each other) they can be in different *neighbourhoods*. With a monolithic tariff this fact means that agents whose consumption alternatives overlap are neighbours (e.g. one consuming at 9 am and 10 a, another at 10 am and 11 am, share 10 am). In our problem domain, the notion of neighbourhood involves more factors. It not only entails the possibility of consuming energy at the same time-slot but also consuming at a time-slot associated by a discount to any other of the neighbouring agent. For instance, suppose that UC1 offers a 3% discount if consuming at 8 am and 11 am. In case Agent A can place its consumption at 8 am and 9 am, neighbour will be every agent potentially consuming at 8, 9 and 11 am.

There are two reasons for this phenomenon. First, if Agent A consumes at 9 and not at 8 am, Agents consuming at 11 am will be affected by this fact since the discount could not take place, or viceversa, agents consuming at the same time slot will have to take into account A's consumption to plan their's. In this way, non-neighbouring agents do not need to interchange messages or be part of the broadcast since their local changes will not affect each other. Therefore, determining neighbourhoods will help reduce the network overload by converting broadcast into *selective multicast*. Unfortunately, this reduction can only be exactly evaluated with real data because the composition of neighbourhoods totally depends on the number of discounts, the time slots they include and the number of time slots that each task has on average as alternative to consume energy.

5. Related Work

There has been a lot of work both in distributed constraint problems and energy management with multi-agent systems. However, these two strands of work have not been brought together before. The work of Yokoo is seminal in the area of distributed constraint problems. His formalisation of the DCSPs can be found in [16], as well as classical and new algorithms in [17]. Modi provides the general formalisation of DCOPs and a new optimal algorithm, called ADOPT (Asynchronous Distributed Constraint Optimisation with Quality Guarantees) to solve it in [5, 7] outperforming previous DCOP algorithms. ADOPT is unfortunately unable to work within our problem domain since it

does not evaluate all possible solutions (condition needed, as already explained, due to the use of combinations and linear supply function bidding).

Energy management research is usually grouped under the general banner of Demand-Side Management. Its main goals are finding the cheapest consumption plan for the clients and the smoothest one for the suppliers. There have been some original attempts of solving the energy scheduling problem as for instance in [8], through a multi-agent system carrying out a genetical algorithm. The application domain is different, since the multi-agent system can only count with very limited resources: nodes of some hundreds Kilobytes memory interconnected by very slow field area networks. Further, the work of Ygge ([14, 15]) takes the leading role in this area. Specifically, he combines power load management with market-oriented programming. He introduces a hierarchical structure of *HomeBots*, intelligent agents that represent every load in the system and buy the energy in a system of forward non-combinatorial auctions. His approach places all the initiative on the *HomeBots* so the UC cannot express its preferences for having more or less demand at a certain time. We address this shortcoming by allowing combinatorial bidding. Moreover, this system does neither offer the possibility of having more than one supplier simultaneously, as we do, and therefore does not take advantage of current deregulated markets.

6. Conclusions and Future Work

The deregulation of energy markets in the European Union has drawn a new draughtboard where players can extend their traditional roles and thereby, maximise their revenues. While UCs are enabled in recent research to express more complex aims, and thus, increase their influence on customers ([10]), these customers are simultaneously allowed to choose cheaper ways of buying the energy they need. Still, within each client's system the consumption of this energy must be achieved efficiently, in a coordinated manner between all devices and according to the bids received.

Against this background, this paper presents, for the first time, a constellation of energy consumers designed as a multi-agent system carrying out COBB, an optimal distributed constraint optimisation algorithm, to find the best consumption plan. Moreover, we describe how this system fits into a bigger deregulated energy market that assures both customers and suppliers higher revenues than in the current classical market. Furthermore, we prove COBB optimal and show that it performs better than potential counterparts, even in different problem domains as the one, it has been designed to. Finally, we describe effective ways to attenuate the network overhead caused by COBB, in order to make it applicable in real-life scenarios. Future work will focus on developing a simulator to test the energy system with real values. This will lead us to an accurate assessment of the whole system under concrete and realistic conditions.

References

[1] S. Conry, K. Kuwabara, V. Lesser, and R. A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1462–1477, 1991.

[2] V. D. Dang and N. R. Jennings. Polynomial algorithms for clearing multi-unit single item and multi-unit combinatorial reverse auctions. In *Proceedings of ECAI '02 (Lyon France)*, pages 23–27, Lyon France, 2002.

[3] V. D. Dang and N. R. Jennings. Optimal clearing algorithms for multi-unit single item and multi-unit combinatorial auctions with demand/suppy function bidding. In *Proceedings of ICEC'03*, pages 25–30, Pittsburgh PA, 2003.

[4] S. Martello and P. Toth. *Knapsack Problems, Algorithms and Computer Implementations*. John Wiley and Sons Ltd, England, 1990.

[5] P. J. Modi. *Distributed Constraint Optimization for Multiagent Systems*. PhD thesis, University of Southern California, 2003.

[6] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Joint Conference on Agents and Multiagent Systems (AAMAS-03)*.

[7] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Joint Conference on Agents and Multiagent Systems (AAMAS-03)*.

[8] P. Palensky. *Distributed Reactive Energy Management*. PhD thesis, Vienna University of Technology, 2001.

[9] Y. Penya, P. Palensky, and L. Lobashov. Requirements and prospects for consumers of electrical energy regarding demand-side management. In *Proceedings of the IEWT'03*, pages 101–102, Vienna Austria, 2003.

[10] Y. K. Penya and N. R. Jennings. Combinatorial markets for efficient energy management. In *Proceedings of the IEEE/WIC/ACM Intelligent Agent Technology (IAT)*, pages 626–632, Compigne, France, 2005.

[11] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[12] T. Sandholm and S. Suri. Market clearability. In *Proceedings of the IJCAI'01*, pages 1145–1151, Seattle WA, 2001.

[13] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1446–1461, 1991.

[14] F. Ygge. *Market-oriented programming and its application to power load management*. PhD thesis, Department of Computer Science, Lund University, 1998.

[15] F. Ygge, H. Akkermans, A. Andersson, M. Krejic, and E. Bortjes. The homebots system and field tests: A multicommodity market for predictive load management. In *Proceedings of the PAAM'99*, pages 363–382, London UK, 1999.

[16] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, September 1998.

[17] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.