Power Load Management as a Computational Market

Fredrik Ygge [‡] **Hans Akkermans** [§]

[‡]Department of Computer Science (IDE)
University of Karlskrona/Ronneby
372 25 Ronneby, Sweden
Fredrik.Ygge@ide.hk-r.se, http://www.rby.hk-r.se/~fredriky

[§]AKMC and University of Twente
Department of Information Systems (INF/IS)
P.O. Box 217, NL-7500 AE Enschede, The Netherlands
akkermans@ecn.nl, J.M.Akkermans@cs.utwente.nl

To appear in the Proceedings of the
2nd International Conference on Multi-Agent Systems ICMAS'96
Kyoto, Japan, December 1996

Abstract

Power load management enables energy utilities to reduce peak loads and thereby save money. Due to the large number of different loads, power load management is a complicated optimization problem. We present a new decentralized approach to this problem by modeling direct load management as a computational market. Our simulation results demonstrate that our approach is very efficient with a superlinear rate of convergence to equilibrium and an excellent scalability, requiring few iterations even when the number of agents is in the order of one thousand. A framework for analysis of this and similar problems is given which shows how nonlinear optimization and numerical mathematics can be exploited to characterize, compare, and tailor problem-solving strategies in market-oriented programming.

1 Introduction

Computational markets have been suggested as a solution to resource allocation, scheduling, and optimization problems (e.g. [2][5][7][8]). In this paper we present how computational markets can be used to handle the application of power load management. A load, in this context, is any device that consumes electric energy, such as a water heater or an electric motor. With load management we mean the concept of controlling the loads at the demand side to achieve a better use of energy, better for the utility, the customer or both. Load management can be used for many different purposes, like avoiding overloads in bottlenecks in the grid, reduce losses caused by reactive power, reduce overtones and stabilize the network. One normally distinguishes between two different categories of load management [4]: direct and indirect. Direct load management

implies that the utility determines what loads are to be connected, reduced, or disconnected at specific occasions. Indirect load management is the case where the utility sends some signal to the customer, such as price information, and relies on his/her ability to adjust to this signal. The topics discussed in this paper are relevant for both the direct and the indirect approach, although we focus on direct load management.

There is a number of reasons why it is interesting to study decentralized algorithms such as computational markets in the area of power load management. First, the information required at each step in a load management process is distributed throughout the system. There is a potential gain in reduced communication by not transmitting this information to a central point, as in a purely centralized approach. Second, the load management problem itself is a computationally complex task, since there is a large number of loads to deal with. By using a decentralized approach the inherent computational power in the network is utilized. Last, but not least, the software engineering aspect: by using a number of distributed manageable models, such as user and process models, instead of a giant, centralized one, software modifications are kept local, simplifying the whole life-cycle of the software, making it possible to add and delete loads without having to modify the existing system.

In this paper we show how a computational market design can be utilized to approach the highly complex and distributed control problem of power load management. In a broader, application-independent, analysis of the problem solving task, we will further show that Adam Smith's 'invisible hand' [11] allows for a plethora of different problem-solving strategies that fundamentally impact the rate of convergence, efficiency, information and communication requirements, and needed agent competencies within market designs. We sketch a general framework for such a problem-solving method analysis, whereby advances in knowledge system analysis [12][10] and, in particular, nonlinear optimization and numerical mathematics [1][3][6] prove to be helpful. In section 2 we discuss design issues for an automated power load management system and introduce our computational market design. Simulation results are presented in section 3. A framework for analysis of problem-solving strategies, illustrated by some published market approaches, is outlined in section 4, and conclusions are given in section 5.

2 HOMEBOTS: Agents for Automated Load Management

From the energy utility point of view it is desirable to have a system that hides most details of the different loads while still providing enough information for energy optimization. The system should be able respond to high level control commands for, e.g., reduction of the current load in the distribution system by a certain amount.

2.1 Computational Market Design

Every controllable load in our system is represented by a load agent, called a HOMEBOT, whose needs and preferences are modeled by a utility function, describing how important it is for the agent to have a share of the resource (i.e. the electrical power). From these utility functions we introduce market mechanisms which settle the distribution of energy according to the market

model described below. The agents are grouped in correspondence with the inherent topology of electrical grid and the communication system. This can for example be as in Figure 1.

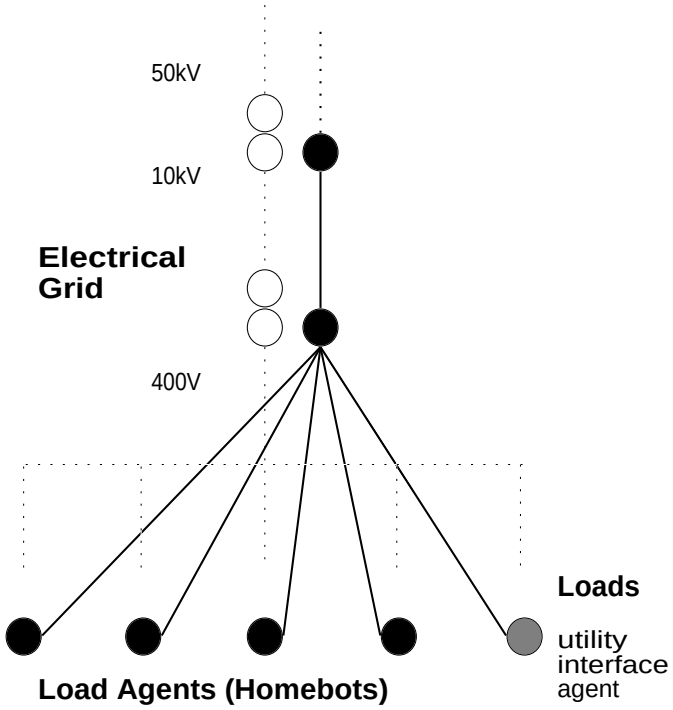

Figure 1: Agent society structure.

The role of the interface agents, as shown in Figure 1, is to provide the utility with an interface to the load management system. The interface agents can be thought of as the utility's local salesmen. An interface agent can thus be directly manipulated by the utility in order to make the system behave in a certain way, so they embody the decentralized control strategies of the energy utility. Interface agents might or might not at the same time serve as load agents. Two examples of the use of the interface agents are given in section 3.

Note that even though we have introduced a computational market, the relation between the utility and the customers is regulated through contracts, and the purpose of the computational market is to make good use of these contracts.

2.2 Utility Functions

The utility function is determined by a number of factors, e.g. load model, current state of the load, user model, utility/customer contract and expected future prices.

From our experience it is, for a number of loads and contracts, very normal to have a concave utility function, so the marginal utility (i.e. the first derivative of the utility function), which serves as the bidding price, decreases with increased resource (i.e. the second derivative is negative). This property of the utility function is very common in other application areas as well [7][13].

Figure 2 shows two typical utility functions, one representing a water heater and one representing an electrically heated house. In the case of the water heater the utility increases until the heater is fully heated. For a heated house, on the other hand, the utility typically raises to a certain point, when comfort is maximized for the user, and then gradually decreases.

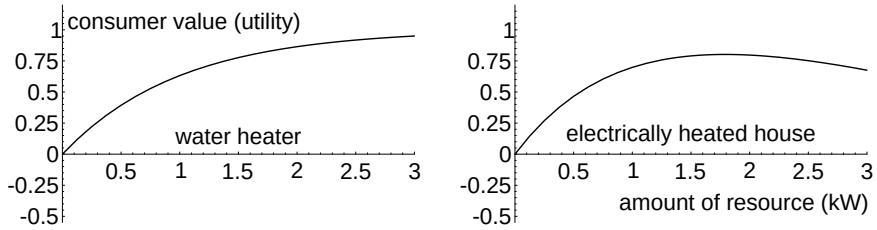

Figure 2: Typical utility curves.

In this paper we have chosen a utility function, denoted by $u(r)$, where r is the allocated resource (electric power) for a certain agent, of the form[1]:

$$u(r) = a - b\exp(-cr) - dr. \tag{1}$$

By adjusting the parameters, a, b, c, and d, we can describe a large number of useful utility functions. For each load, r is constrained, in that it has a lower and an upper bound.

2.3 Bids and Auction Schemes

Reallocations in computational markets are often realized through auctions (e.g. [2][5][7]) and this mechanism is also adopted here. In an auction every agent in a group sends a bid, to a designated auctioneer or to the other agents, and then a reallocation of the resource is computed. When designing a computational market, one has to decide how bids should be constructed and, from these bids, determine a proper reallocation scheme. Thus, the auctioneer should maximize the total utility of the system from the current bids of the agents. Formally this leads to the following optimization problem:

$$\begin{aligned} &\text{Maximize} \qquad u_{tot}(r_1, r_2, \ldots, r_n) = \sum_{j=1}^{n} u_j(r_j), \\ &\text{such that} \qquad \sum_{j=1}^{n} r_j = R. \end{aligned} \tag{2}$$

[1]This utility function is chosen as a *reasonable* utility function. The *exact* form will be established from further investigations of load models etc.

Since the total utility is the sum of the local utilities for every agent, the problem is said to be separable [6]. This property gives rise to a significant simplification of the reallocation task.

From nonlinear optimization theory we know that when the Kuhn-Tucker conditions are fulfilled, i.e. when all first derivatives of the utility functions, $u_i'(r_i)$, are equal to a shared value λ, for all agents having an amount of resource different from their boundary values, then we have achieved an optimal global solution for this constrained nonlinear separable concave problem. (For details refer to for example Ibaraki and Katoh [6] or Fletcher [1].) Loosely, this means that no agent which is not at its boundary, has a higher marginal utility than any other agent.

It follows that an optimal solution can be obtained by solving the nonlinear equation system $u_i'(r_i) - u_n'(r_n) = 0$, for $i = 1, 2, \ldots, n-1$. To this end, we introduce a column vector $\mathbf{f} = [f_1, f_2, \ldots, f_{n-1}]^T$, where $f_i = u_i'(r_i) - u_n'(r_n)$. Then we apply the standard multi-variable Newton method, (3), for solving the vector equation $\mathbf{f} = \mathbf{0}$:

$$\begin{aligned} \Delta\mathbf{r}^k &= (\text{grad}\,\mathbf{f}^k)^{-1} \times \mathbf{f}^k, \\ \mathbf{r}^{k+1} &= \mathbf{r}^k + \alpha\Delta\mathbf{r}^k, \end{aligned} \tag{3}$$

where $\mathbf{r}$ is the column vector, $\mathbf{r} = [r_1, r_2, \ldots, r_{n-1}]^T$, grad $\mathbf{f}$ is the gradient matrix of $\mathbf{f}$ (also called Hessian matrix, containing the second-order partial derivatives of the utility function), α is a step size, and k is the index of the iteration. With utility functions of the above kind we can use a step size equal to unity. The key advantage of a Newton scheme is its very fast quadratic convergence; a property clearly borne out in our experiments, and of central importance to real-life load management.

For solving the constrained problem we rewrite r_n as $R - r_1 - r_2 - \cdots - r_{n-1}$. Then grad $\mathbf{f}$ is:

$$\text{grad}\,\mathbf{f}_{ij} = \begin{cases} u_i''(r_i) + u_n''(r_n) & i = j, \\ u_n''(r_n) & i \neq j. \end{cases} \tag{4}$$

After the elimination-by-substitution of the resource constraint, we have an unconstrained optimization left, which we solve by the standard Newton scheme of equation (3). Due to the separability, inversion of the Hessian matrix in equation (3) is easily done in closed form by use of the Sherman-Morrison formula [3].

In order to manage the local boundaries of the resource we use a scheme based on the RELAX algorithm with lower and upper bounds on variables [6] with the above Newton method as relaxed allocation algorithm. Loosely, the idea of the RELAX algorithm with lower and upper bounds on variables is to first solve the relaxed allocation problem, i.e. without bounds on variables, and for each of the variables that exceed their limit value assign the corresponding limit value and allocate the remaining resource among the unlimited agents by using the RELAX algorithm recursively. For details refer to Ibaraki and Katoh [6]. It is not difficult to show that this way of dealing with the local bounds theoretically scales at most linearly with the number of agents. (In our simulation experiments of section 3, it even appears to be rather independent of the number of agents.)

The information needed from every agent in each auction, the agents' bids, is the current value and the lower and upper bounds of the resource and the first and second derivatives of the utility

function. Then for every consecutive iteration of the auction the first and second derivatives are needed.

With the simple utility functions specifically used in equation (4) we could also use an alternative price directed auction scheme, see section 4. However, the Newton algorithm will work for a very wide class of utility functions, and is thus more general.

3 Simulation Results

In this section we present simulation results demonstrating how interface agents can be used to control the loads in the distribution system, and see that the above algorithm rapidly reallocates the resource properly in the system. Throughout the simulation, one iteration is performed per time unit. As we want to take advantage of the benefits of really decentralized control, the energy utility behaves like any other party on a competitive market, rather than wielding monopolistic power as in a purely centralized approach. We will see that this indeed turns out to be possible, and very efficiently so. Looking at the formal problem statement of equation (2), it follows that basically two different strategies are open to the utility within the confines of distributedness:

1. Influencing a single interface agent's resource in the global resource constraint, so as to effectively reduce the total R.
2. Influencing the utility optimum, by changing a single interface agent's utility function and, thereby, bidding price.

Simulation results for both strategies are described in sections 3.1 and 3.2 respectively. For reasons of clarity, results are given for a system of ten agents. Scaling issues for the field application are discussed in section 3.3.

3.1 Reducing by a Certain Amount of Resource

In the first simulation we demonstrate a situation where the utility needs to reduce the total load, R, in the system, due to shortage in production, overload in a bottleneck in the grid, or similar. In this case the interface agent is designated to a load, say a water heater with a minimum load of 0 kW and maximum load of 3 kW. The utility can directly remove or add resource to this agent. Here we demonstrate how a request for a reduction of 5.3 kW is managed by the system.

In Figure 3 we see that from a given initial allocation, measured in kW of electric power, the system rapidly relaxes to equilibrium (in two time units). Then at time six the utility requests the reduction and simply sets the going resource value for the interface agent from approximately 1.3 kW to -4 kW. This results in a non-feasible allocation, since the lower limit of the interface agent (0 kW) is violated. First, the system regains a feasible allocation; after one time unit the utility controlled load is above its lower bound (0 kW), due to reduced allocation to other agents, and thus the 5.3 kW (1.3 + 4) reduction is performed. After another three time units the system is back in equilibrium. If the price of the resource directly corresponds to the real price of power,

we immediately can tell how much this load reduction costs. This is of course very useful when evaluating the load management system.

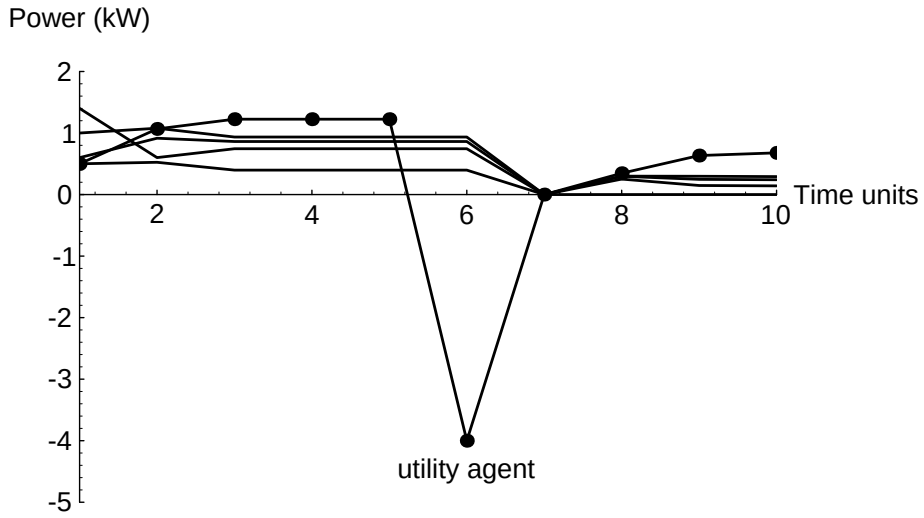

Figure 3: A direct reduction of resource in a load management system.

3.2 Changing the Price

As a second control strategy we show a similar system, but one where the interface agent is not a load agent. Here the utility attempts to influence the total energy consumption not by controlling a specific resource, but by bidding a specific price through changing its utility function. It then trusts the load management system to achieve a new equilibrium; here it is open how much the customer loads will be reduced. For achieving this we have used a linear utility function for the interface agent. This means that the interface agent will buy back power, thus reducing the consumption of the loads and decreasing production with the same amount, from all agents having a lower marginal utility than the marginal utility of the interface agent, i.e. the slope, or derivative, of the straight line.

In Figure 4 a simulation of a price change is shown. Again the agents are assigned a given initial amounts of resource. The interface agent (the top curve in Figure 4) starts with a marginal utility (its bid price) of say 0.3. From the initial situation, equilibrium is reached in three time units. Then the utility attempts to reduce the consumption of the loads, by entering a new bid price for its interface agent equal to, say, 0.4, at time seven. Two time units later the system is seen to be back in equilibrium. From the amount of resource owned by the utility agent we see that by changing the price, the amount of used resource in this part of the system was decreased by approximately 2.6 kW ($r_{ia}(10) - r_{ia}(4) = 4.1 - 1.5 = 2.6$).

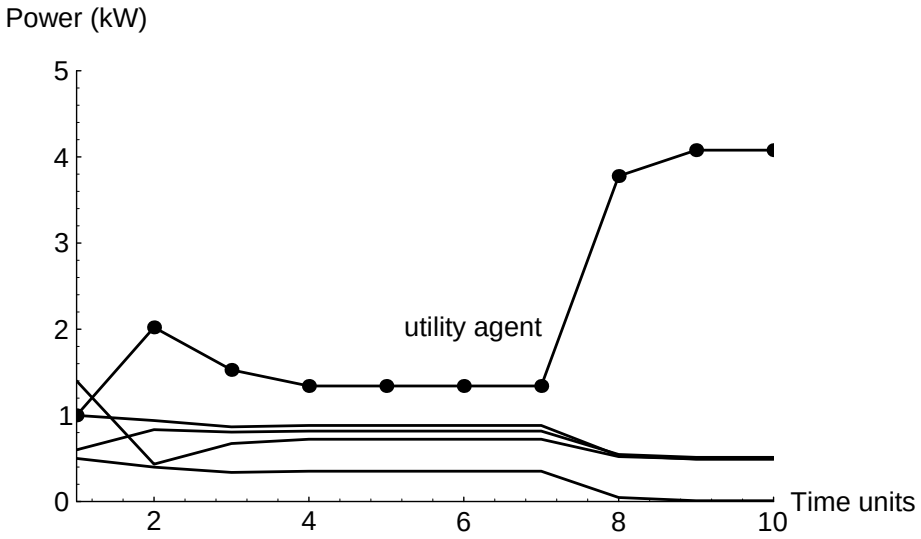

Figure 4: Changing the price of the resource.

3.3 Scaling Up

The two simulations shown in sections 3.1 and 3.2 were run with 10, 20, 50, 100, 500 and 1000 agents and all results are as good, with respect to convergence, as the ones shown above. We note that in a real-life application, the number of loads will typically be in the order of a few hundreds in a secondary substation area. For getting back to equilibrium when the disturbance is introduced, three to four and two iterations are needed in simulation one and two respectively. The result is much more favorable than the worst-case theoretical result which indicates a linear scaling with the number of agents. In the simulations we have only referred to time in terms of number of iterations, but in field applications we have to deal with real time constraints. However, a realistic value for reaching a new equilibrium, which depends on communication speed, computational power in the network, and agent configuration, is estimated to be below 0.5 s. This is an extremely satisfactory figure, because it means that our market approach to power load management in real-life situations is just a matter of seconds.

4 Analysis

Above, we have shown the practical value of the computational market in solving a specific, but important, real-life application problem. As the number of successful applications quickly increases, there is a growing need for a common framework for analysis, the more so because the tackled problems as well as the problem-solving strategies appear highly diverse. Application-independent, knowledge-level [9][10][12] characterizations of problems and strategies are needed, in order to gain an improved understanding of, and formulate better guidelines for market-oriented problem-solving. For computational markets, the following knowledge level aspects of analysis

seem particularly useful to us:

(i) Problem or overall task characteristics: goals, givens, working conditions and features set by the task environment.

(ii) Task organization: decomposition, I/O flow, control, responsibilities due to the distribution over agents.

(iii) Information requirements: what is needed to know for successful task execution, and where is this information located?

(iv) Communication requirements: what information must be communicated to other agents, and when?

(v) Agent competencies: what must be/is an agent capable of?

(vi) Performance and quality metrics for task execution.

This enables us to 'position' different approaches and strategies in a multidimensional 'space' of analysis.

Wellman [15] emphasizes a similar point, and specifically proposes to build on concepts and theory of microeconomics as a formal foundation, and Kurose and Simha [7] offer a practical example. These are steps in turning the computational market idea from an attractive but informal metaphor for multi-agent system architectures (the famous 'invisible hand'[11]) into a full scientific theory. We concur with such a position, but note that there is an extensive additional body of knowledge in modern mathematics, which in our view has not yet been exploited to the full benefit of formal analysis of market-oriented programming. This concerns in particular nonlinear optimization and numerical mathematics (for recent overviews, see e.g. [1][6]). In this section we sketch some insights ensuing from such an analysis, illustrated by various computational market problems published in the literature.

4.1 Resource- vs. Price-Oriented Strategies

Our task organization —which we see as the technical layout of what in fact is a new business process in energy utilities— is based on what is sometimes called a resource-directed strategy, in contrast to a price-directed strategy. Essentially, this distinction refers to how different subtask responsibilities are distributed between the trading agents on the one hand, and the auctioneer on the other hand. The dynamic variables to be dealt with in market problems are given by the vector pair (p, r) of bid prices and demands for resources. In a price-directed strategy, the trading agents handle the r-part by indicating their resource demands at the going price (by inversion of the first derivative of the utility function), and the auctioneer then deals with the p-part, repeatingly sending out market update information in terms of adjusted price signals until the resource constraints are met. This leads to a market dynamics whereby the prices function as the independent variables. In resource-directed strategies, it is the other way around. Here the resource equality demand is first dealt with (through elimination by substitution), and then the utility optimization task is undertaken (by an iteration procedure).

It can be verified that the computational complexity in both cases is the same under appropriate conditions. A capability needed, however, in the price-directed strategy is to carry out the function inversion (existence is guaranteed by concavity). This is only possible in a limited number of cases, but if it works this is also highly efficient. In resource-directed allocation the competence to calculate derivatives is needed instead (to achieve comparable efficiency), but this will work for a wider class of problems.

Wellman's Walras system [14][15], and 'tâtonnement' in general, represent prototypical examples of a price-based strategy. Mixed strategies, in which agents bid both for a resource and offer a price are also possible, cf. Huberman and Clearwater [5]. Kurose and Simha [7] and the present work both employ resource-directed strategies.

4.2 Bid Information and Convergence

As pointed out in section 2.3 we have made the strategic choice that load agents communicate two information point items (p, p') consisting of their marginal utility at the current local resource and its derivative. In principle, one item is sufficient for a working solution, but adding the second item has the advantage that it leads to a quadratic rather than linear convergence rate. If this competency of obtaining derivative values is not available, in our application we could switch to variable-metric (quasi-Newton) or conjugate-gradient strategies which will need more iterations but are still superlinear.

We note that most current market schemes, including the binary search in Walras as well as 'tâtonnement' in general [14], the mixed scheme for energy management in [5], and the first-order algorithm of [7], are all linear schemes, because they are all in some way proportional to the current 'error'. Communication requirements are also very different: Walras [14] communicates full curves rather than point items, the information content in [5] is different but the related communication burden is higher (since additional global information is required for the local bids) compared to our case, whereas in the first-order algorithm of [7] only a single point item is communicated. It follows that given the amount and nature of the communicated information, in all mentioned systems except the first-order algorithm of [7], superlinear rather than linear convergence speed could be achieved. More generally, many strategic choices are at our disposal that can improve problem-solving and computational properties. The tools of nonlinear optimization and numerical mathematics are helpful in a formal analysis of this, and we will show this by example in 4.3 and 4.4 in some technical detail.

Another point worth mentioning here is that in Walras, since the entire utility function is sent to the auctioneer, the convergence is a task solely resting with the auctioneer, while the other strategies presented here require inter-agent communication at each iteration.

4.3 Analysing Kurose-Simha

Kurose and Simha [7] consider a distributed file allocation problem. The major difference with our problem is in the absence of upper bounds and in the shape of the utility functions. In Kurose and Simha's application it is taken to be:

$$u_i(r_i) = c_i r_i + \frac{r_i}{\mu_i - \lambda_i r_i}, \tag{5}$$

Their simulation results demonstrate clearly how computational properties favorably change by adding a single communicated information item (p') in going from their first-order to their second-order algorithm. These results become theoretically obvious by our following proposition.

Proposition: The second-derivative algorithm of [7] is equivalent to a Newton algorithm (as outlined in our equation (3)), specialized to an optimization problem that (i) is separable and (ii) has a linear equality constraint.

Proof: Separability leads to a simple diagonal Hessian matrix in unconstrained problems, and to the form of our equation (4) in the constrained case. Due to this special form, equation (3) can be analytically solved by using the matrix inversion formula named after Sherman-Morrison [3]. Carrying out these calculations precisely gives Kurose and Simha's second-derivative algorithm.

Corollary: Being a Newton scheme, this is provably a quadratically convergent algorithm, in the sense that quadratic test functions are optimized in $\mathcal{O}(1)$ iterations. In contrast, linear schemes as mentioned above, need $\mathcal{O}(n)$ iterations for solving the Kuhn-Tucker gradient conditions, n representing the number of dimensions, i.e. agents, computer nodes or commodities. The utility functions of both equations (1) and (5) are sufficiently similar to a quadratic function over a wide range so that a Newton scheme converges and practically achieves this superlinear convergence. This also theoretically explains why our HOMEBOTS system attains equilibrium in so few iterations.

Thus, a proper choice of information and communication resources and competencies leads to very efficient problem-solving. This has been shown above within a resource-directed strategy, but also holds for price-directed strategies. Interestingly, Kurose and Simha emphasize the drawbacks of the price-directed strategy. However, we now prove that their file allocation problem might also be efficiently solved by a price-directed strategy which is equivalent to Walras [14].

Proposition: The (unconstrained) resource demands corresponding to the (first derivative of the) utility function (5) are given by:

$$r_i = \frac{\mu_i}{\lambda_i} - \frac{1}{\lambda_i}\sqrt{\frac{\mu_i}{p - c_i}}. \tag{6}$$

Proof: The corresponding Kuhn-Tucker gradient condition (all dimensions/agents share the same market equilibrium price p for their marginal utilities) results, due to the separability, in a set of equations with a single resource variable and p as a free parameter. These can be easily solved by function inversion. Basically, the resource scales with the inverse square root of the market price, as seen in equation (6).

Corollary: The task of the auctioneer is to find a value for the equilibrium price p such that the resources satisfy the global equality constraint (telling that (6) sums up to unity). This is a one-dimensional root search problem. We further note that our application might also be handled in the same fashion. Analytical results, such as these, are very useful for testing and validating.

4.4 Analysing Walras

Along the same lines we can show that Wellman's Walras system [14][15] might solve the same problems through alternative resource-directed strategies. In his price-based strategy he employs binary search for the global resource constraint, which has a linear rate of convergence. This may be improved to a quadratic scheme, especially if second-order derivative calculations can be made. In his application to configuration design [15], the constant elasticity utility function is used, and in this case this is not much of a problem.

Walras has the special feature that it solves a multi-commodity problem, while all other cited work as well as our own application involves a single commodity. This poses the question [14] how the complexity of resource allocation problems scales with both the number of agents and commodities. A partial answer was given in the previous sections, where we pointed out that worst-case theory gives a linear scaling with the number of agents, for concave separable problems, and for each commodity. In practice, we found that it is even more favorable. This fits, by the way, with Wellman's remarks. He conjectures that the scaling with the number of commodities is much more badly. Nevertheless, we have that mutatis mutandis, interchanging agents with commodities, a worst-case statement holds analogous to the one above. In Walras, each agent has to solve an optimization problem, but this is very efficiently done by inverse function evaluation. For each commodity, we have a global resource constraint. In the Walras formulation [15], however, these are all coupled through the prices. This leads to a root search in k dimensions simultaneously, with k denoting the number of commodities, rather than k times a one-dimensional search as in a separable case. Hence, the Walras approach is separable in terms of agents, but non-separable in terms of commodities. This explains the difference in computational complexity. But other strategies with different scaling properties are conceivable.

5 Conclusions

In this paper we introduced a computational market design for power load management. Power load management aims at managing the energy consumption of customer equipment such as boilers and heaters. Every device acts as an agent in a computational market buying and selling energy. This is a decentralized way to reduce unwanted peak loads and hereby save money. We describe an operational simulation environment, called the HOMEBOTS system. Simulation experiments carried out with the system show that:

- The optimal allocation is achieved within a few bidding iterations.
- The system scales up nicely since the number of needed iterations is rather independent of the number of agents, even if it is as high as one thousand.
- The utility can act as being just a party among many on the market, and still rely on very simple tactics to successfully manage many loads: it suffices to either remove a certain amount of resource or to change the bidding price, impersonated by a single utility-controlled agent.

In conclusion, market-oriented programming is well suited for this kind of applications, because the information required for optimization is inherently distributed, problem-solving turns out to be simple and efficient and it enables loads to be added and deleted without modifications of the existing system.

Furthermore, we have given a brief, broader analysis of this and other computational market problems from the literature. Nonlinear optimization and numerical mathematics provide an integrated framework to characterize different market strategies and to compare their advantages and disadvantages. From our analysis we conclude that:

- This branch of mathematics offers solutions to problems much more complicated than currently investigated in market-oriented programming, opening up new application areas.
- It is possible to find schemes with a superlinear rate of convergence for market-oriented programming problems, better than the often used binary search schemes.
- It is possible to derive analytical insights into computational markets, important for validating and testing computational market programs for real settings.

Hence this approach helps to better understand, but moreover, considerably extends available problem-solving strategies in market-oriented programming.

Acknowledgments. We would like to thank professor Rune Gustavsson at the University of Karlskrona/Ronneby and president Hans Ottosson at EnerSearch AB for their large support and encouragement, Hans Olsson at Lund University for enlightening us on some numerical analysis issues, the SoC team at the University of Karlskrona/Ronneby and Lund University, the load management team at Lund University, and the student team Flux at SIKT for interesting discussions and comments on draft material.

References

[1] Fletcher, R., *Practical Methods of Optimization*, Second Edition, John Wiley & Sons, 1987.

[2] Gagliano, R.A., M.D. Fraser, and M.E. Schaefer, *Auction Allocation of Computing Resources*, Communications of the ACM, Vol. 38, No. 6, June 1995, pp. 88-102.

[3] Golub, G.H. and C.F. Van Loan, *Matrix Computations*, Second Edition, John Hopkins University Press, 1991.

[4] Hägg, S. and F. Ygge, *Agent-Oriented Programming in Power Distribution Automation - An Architecture, a Language, and their Applicability*, Licentiate thesis, Department of Computer Science, Lund University, LUNFD6/(NFCS-3094)/1-193/(1995), LUTEDX/(TE-CS-3056)/1-193/(1995), 1995.

[5] Huberman, B. and S. Clearwater, *A Multi-Agent System for Controlling Building Environments*, in Proceedings of the First International Conference on Multi-Agent Systems ICMAS'95, San Francisco, CA, June 1995, pp. 171-176.

[6] Ibaraki, T. and N. Katoh, *Resource Allocation Problems — Algorithmic Approaches*, The MIT Press, Boston, 1988, ISBN 0-262-09027-9.

[7] Kurose, J.F. and R. Simha, *A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems*, IEEE Transactions on Computers, Vol. 38, No. 5, May 1989, pp. 705-717.

[8] Lenting, J.H.J. and P.J. Braspenning, *Delegated Negotiation for Resource Re-allocation*, Lecture Notes in Artificial Intelligence, Vol. 671, pp. 299-312, Springer Verlag, Berlin, 1993.

[9] Newell, A., *The Knowledge Level*, Artificial Intelligence, Vol 18, 1982, pp. 87-127.

[10] Schreiber, G., B. Wielinga, H. Akkermans, W. Van de Velde, and R. de Hoog, *CommonKADS - A Comprehensive Methodology for KBS Development*, IEEE Expert 9(6), December 1994, pp. 28-37.

[11] Smith, A., *An Inquiry into the Nature and Causes of the Wealth of Nations*, University of Chicago Press, Chicago, 1976. Reprint of the 1776 edition.

[12] Steels, L., *Components of Expertise*, AI Magazine 11(2), Summer 1990, pp. 28-49.

[13] Steiglitz, K., M.L. Honig and L.M. Cohen, *A Computational Market Model Based on Individual Action*, in Market-Based Control: A Paradigm for Distributed Resource Allocation, Scott Clearwater (ed.), World Scientific, Hong Kong, 1995.

[14] Wellman, M., *A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems*, Journal of Artificial Intelligence Research, Vol. 1, No. 1, 1993, pp. 1-23.

[15] Wellman, M. *A Computational Market Model for Distributed Configuration Design*, Proceedings of AAAI '94, Morgan-Kaufman, CA, 1994, pp. 401-407.