On Resource-Oriented Multi-Commodity Market Computations

by

Fredrik Ygge and Hans Akkermans

On Resource-Oriented Multi-Commodity Market Computations

by Fredrik Ygge and Hans Akkermans

ISSN 1103-1581

ISRN HK/R-RES—98/7—SE

Copyright © 1998 by Fredrik Ygge and Hans Akkermans

All rights reserved

Printed by Psilander Grafiska, Karlskrona 1998

On Resource-Oriented Multi-Commodity Market Computations

Fredrik Ygge

EnerSearch AB and
University of Karlskrona/Ronneby
Department of Computer Science (IDE)
372 25 Ronneby, Sweden
fredrik.ygge@enersearch.se
<http://www.enersearch.se/~ygge>

Hans Akkermans

AKMC Knowledge Management and
Free University Amsterdam
Computer Science Department
De Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands
HansAkkermans{ @compuserve.com, @cs.vu.nl }

Extended version of paper for ICMAS 98

Abstract

In search for general equilibrium in multi-commodity markets, *price-oriented* schemes are normally used. That is, a set of prices (one price for each commodity) is updated until supply meets demand for each commodity. In some cases such an approach is very inefficient, and a *resource-oriented* scheme can be highly competitive. In a resource-oriented scheme the *allocations* are updated until the market equilibrium is found. It is well known that in a two-commodity market resource-oriented schemes are possible. In this paper we show that resource-oriented algorithms can be used for the general multi-commodity case as well, and present and analyze a specific algorithm. The algorithm has been implemented and some performance properties, for a specific example, are presented.

1 Introduction

In virtually all multi-agent systems there are some scarce resources. Thus, the issue of resource allocation in multi-agent systems is of fundamental importance. Different market concepts have been suggested for, and successfully applied to, a number of applications (e.g., [Wellman, 1993, Wellman, 1994, Clearwater and Huberman, 1994, Lenting and Braspenning, 1994, White, 1994, Huberman and Clearwater, 1995, Wellman, 1996, Hu and Wellman, 1996, Miller and Drexler, 1996, Yamaki *et al.*, 1996, Ygge and Akkermans, 1996, Ygge and Akkermans, 1997]). One promising approach is to apply general equilibrium theory to actual resource allocation (e.g., [Wellman, 1993, Wellman, 1994, Wellman, 1996, Hu and Wellman, 1996, Yamaki *et al.*, 1996, Ygge and Akkermans, 1996, Ygge and Akkermans, 1997]). For this approach to be successful, both a deep understanding of foundational micro-economics as well as efficient algorithms are required. The most common approach is to search over a set of prices (one price per commodity) until supply meets demand. Typically each agent is participating in an iterative auction where it faces a set of prices and replies with a demand/supply message (and optionally some derivatives of these). From the total demand/supply of the market a new set of prices is computed. This is iterated until supply is equal to demand for each commodity, see e.g. [Cheng and Wellman, 1998]. This is commonly referred to as a *price-oriented* approach. However, with this approach, the demand for each agent must be computed through solving an optimization problem based on its *utility function*¹ or its *technology*², either by each agent or by an auctioneer. If the demand can not be analytically derived for an agent, it has to be searched for numerically. Thus, a search for general equilibrium prices can result in an iterative search over prices *plus*, for each step of the algorithm, iterative searches for the demands of the involved agents. This is of course very inefficient. A way around this is to use a *resource-oriented* approach.

For a wide class of utility functions (further described below) an alternative way to express the general equilibrium is to define it as an allocation such that, for each commodity, each agent is willing to pay the same price for an additional small amount of resource. The price that an agent is willing to pay is (as will be described in more detail below) simply the relation between two partial derivatives of the utility functions. These derivatives are typically easy to derive analytically and can always, inexpensively, be obtained numerically. The approach of finding an allocation such that the price is equal for each agent is referred to as a *resource-oriented* algorithm. In this paper we introduce a resource-oriented algorithm for the multi-commodity case and demonstrate it on an example.

Throughout the paper we assume the resources to be infinitely divisible. We also assume that there are some upper and lower bounds on how much resources each agent can take. We do not discuss existence and uniqueness of the general equilibrium, but refer to other work on those issues (e.g. [Cheng and Wellman, 1998, Mas-Colell *et al.*, 1995, Takayama, 1985, Shoven and Whalley, 1992]). The work presented in this paper is of importance to industrial applications such as power load management, whereby large numbers of electric industrial equipment and/or household appliances are coordinated in a distributed fashion in order to optimize the total energy use over time [Ygge and Akkermans, 1996, Ygge and Akkermans, 1997, Ygge, 1998]. Here, the energy in different time slots (say, the different hours of the day) acts as the bundle of

¹A utility function is a preference ordering of a consumer's different consumption bundles. It can, e.g., describe whether or not two bananas and one apple are preferred over one banana and two apples for a specific customer.

²The technology describes a producer's possibilities to transform inputs to outputs.

commodities exchanged between electric devices.

2 Multi-Commodity Search

In this section we describe the two main approaches for multi-commodity search and discuss important properties of these approaches.

2.1 Price-Oriented Approaches

The market equilibrium is given by

$$\mathbf{z}(\mathbf{p}) = \mathbf{0}, \quad (1)$$

where $\mathbf{z}(\mathbf{p}) = [z_1(\mathbf{p}), z_2(\mathbf{p}), \dots, z_k(\mathbf{p})]^T$, $z_i(\mathbf{p})$ being the aggregate excess demand³ for commodity i , $\mathbf{p} = [p_1, p_2, \dots, p_k]$ where p_i is the price for commodity i , and k is the number of commodities. In a price-oriented scheme the price vector is updated until Eq. (1) is fulfilled. Since prices are only relative, we set $p_k = 1$, and we need only to search over $k - 1$ elements in the price vector. Inputs to the scheme are the respective net demands of each agent, $\mathbf{z}_\alpha(\mathbf{p})$, where α denotes an agent.

One example of such an algorithm is WALRAS [Cheng and Wellman, 1998]. Another is a multi-variable Newton search as described by

$$\mathbf{p}^{i+1} = \mathbf{p}^i - s \cdot \nabla \mathbf{z}^{-1}(\mathbf{p}^i) \cdot \mathbf{z}(\mathbf{p}^i), \quad (2)$$

where $i + 1$ and i denote iterations, s is a step size, and $\nabla \mathbf{z}(\mathbf{p})$ is the gradient matrix defined by $\nabla z_{ij}(\mathbf{p}) = \frac{\partial z_{ij}(\mathbf{p})}{\partial p_j}$. The proper value of s can be determined at run-time by a suitable algorithm [Press *et al.*, 1994]. The allocation of the k th commodity is obtained from the budget constraint, i.e. $z_{\alpha k} = -\sum_{j=1}^{k-1} p_j z_{\alpha j}$.

2.2 Resource-Oriented Approaches

Under the assumption that there is a one-to-one relationship between prices and excess demands⁴, an alternative way to express Eq. (1) is to reallocate resources between the agents until

$$\begin{cases} p_{\alpha i}(\mathbf{z}_\alpha) \leq p_i, & z_{\alpha i} = z_{\alpha i}^l \\ p_{\alpha i}(\mathbf{z}_\alpha) = p_i, & z_{\alpha i}^l < z_{\alpha i} < z_{\alpha i}^u \\ p_{\alpha i}(\mathbf{z}_\alpha) \geq p_i, & z_{\alpha i} = z_{\alpha i}^u \end{cases} \quad (3)$$

holds. Here, $p_{\alpha i}(\mathbf{z}_\alpha)$ is agent α 's price for commodity i at a change in allocation $\mathbf{z}_\alpha$, n is the number of agents, k is the number of commodities, $z_{\alpha i}^l$ and $z_{\alpha i}^u$ are the lower and upper limit of the change in allocation. Thus, each agent α is holding a price vector $\mathbf{p}_\alpha = [p_{\alpha 1}, p_{\alpha 2}, \dots, p_{\alpha k}]^T$.

³The aggregate excess demand is the sum of the supply and demand of all agents, i.e. $z_i(\mathbf{p}) = \sum_{\alpha=1}^n z_{\alpha i}(\mathbf{p})$, where $z_{\alpha i}(\mathbf{p})$ is the demand of agent α . The demand of an agent describes how much it is willing to buy (or sell – a negative demand) at a specific price level.

⁴Clearly the assumption that there is a one-to-one relationship between prices and excess demands is a restriction. At the same time we note that this is a standard assumption also when price-oriented algorithms are used in order to guarantee the existence of general equilibrium.

For the case with only two commodities, it is well known that resource-oriented schemes are possible [Ygge and Akkermans, 1996, Andersson and Ygge, 1998], but resource-oriented approaches for multi-commodity markets have not yet been constructed.⁵ As in the price-oriented case we only need to search over $k - 1$ commodities, $p_k = 1$, and $z_{\alpha k} = -\sum_{j=1}^{k-1} p_j z_{\alpha j}$. The inputs to a resource-oriented scheme are, for each agent, the price function, and the boundaries ($\mathbf{p}_\alpha(\mathbf{z}_\alpha)$, $\mathbf{e}_\alpha$, $\mathbf{z}_\alpha^l$ and $\mathbf{z}_\alpha^u$).

The competitive behavior of an agent holding a quasi-concave utility function, $u_\alpha(\mathbf{z}_\alpha)$, and endowment, $\mathbf{e}_\alpha$, is the solution to

$$\begin{aligned} \max_{\mathbf{z}} \quad & u_\alpha(\mathbf{e}_\alpha + \mathbf{z}_\alpha) \\ \text{s.t.} \quad & \mathbf{p}^T \cdot \mathbf{z}_\alpha = 0. \end{aligned} \quad (4)$$

where $\mathbf{p}$ is considered as exogenous. We solve this by setting up the Lagrangian:

$$\mathcal{L}_\alpha = u_\alpha(\mathbf{e}_\alpha + \mathbf{z}_\alpha) + \lambda \mathbf{p}_\alpha^T \cdot \mathbf{z}_\alpha, \quad (5)$$

and solve the equation system $\frac{\partial \mathcal{L}_\alpha}{\partial z_i} = 0$ and $\frac{\partial \mathcal{L}_\alpha}{\partial \lambda} = 0$.⁶ Accordingly, we get $\frac{\partial u_\alpha}{\partial z_i} = \lambda p_{\alpha i}$, for all commodities i . It immediately follows that $\frac{\frac{\partial u_\alpha}{\partial z_i}}{\frac{\partial u_\alpha}{\partial z_k}} = p_{\alpha i}$, $1 \leq i \leq k - 1$, where we have used the fact that $p_{\alpha k} = 1$.

Hence, we have obtained a way to compute the prices from the (derivatives of the) utility functions. This computation can be performed by each agent or by the auctioneer. The derivatives can in some cases be computed analytically and in other cases inexpensive numerical approximations are used [Press *et al.*, 1994].

Now we introduce a vector $\mathbf{f}$, defined by $\mathbf{f} = [\mathbf{p}_1(\mathbf{z}_1) - \mathbf{p}_n(\mathbf{z}_n), \mathbf{p}_2(\mathbf{z}_2) - \mathbf{p}_n(\mathbf{z}_n), \dots, \mathbf{p}_{n-1}(\mathbf{z}_{n-1}) - \mathbf{p}_n(\mathbf{z}_n)]^T$. Then $\mathbf{f} = \mathbf{0}$ is equivalent to Eq. (3). A standard multi-variable Newton method for solving this equation is then given by

$$\mathbf{z}^{i+1} = \mathbf{z}^i - s \cdot \left(\nabla \mathbf{f}(\mathbf{z}^i) \right)^{-1} \cdot \mathbf{f}(\mathbf{z}^i), \quad (6)$$

where s is a step size, and $z_{\alpha j}$ denotes the change in agent α 's allocation of commodity j .

From the feasibility constraint we get that $\mathbf{z}_n = -\sum_{\alpha=1}^{n-1} \mathbf{z}_\alpha$ and thus

$$\nabla f_{ij} = \begin{cases} \nabla \mathbf{p}_i(\mathbf{z}_i) + \nabla \mathbf{p}_n(\mathbf{z}_n), & i = j \\ \nabla \mathbf{p}_n(\mathbf{z}_n), & i \neq j \end{cases}. \quad (7)$$

Due to the symmetric appearance of $\nabla \mathbf{f}$, it can be inverted analytically by hand and Eq. (6) can be reduced to the following expression for the update of each agent (proof see Appendix A)

$$\mathbf{z}_\alpha^{i+1} = \mathbf{z}_\alpha^i - s \cdot \diamond \mathbf{p}_\alpha^i \left(\mathbf{p}_\alpha^i - \langle \mathbf{p} \rangle^i \right), \quad (8)$$

⁵In fact, in [Ygge and Akkermans, 1996] we only discussed one commodity and the other was implicit. For a discussion on the relation between a utility maximization problem with one commodity and a market with two commodities see [Ygge and Akkermans, 1997] and [Ygge, 1998].

⁶Note that submitting utility functions and endowments, and computing equilibrium from competitive behavior, does not prevent agents from speculating [Hurwicz, 1986]. Agents can still speculate by giving false utility functions, similar to how agents can speculate when submitting demand functions in the price-oriented case. In small markets, agents that have both detailed and accurate knowledge about other agents can gain by such speculation [Sandholm and Ygge, 1997].

where $\mathbf{p}_\alpha^i$ and $\diamond \mathbf{p}_\alpha^i$, are abbreviations for $\mathbf{p}_\alpha(\mathbf{z}_\alpha^i)$, and $(\nabla \mathbf{p}_\alpha(\mathbf{z}_\alpha^i))^{-1}$, respectively. The term $\langle \mathbf{p} \rangle^i$, defined by

$$\langle \mathbf{p} \rangle^i = \mathbf{p}_n^i + \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \times \diamond \mathbf{p}_n^i \sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i (\mathbf{p}_\beta^i - \mathbf{p}_n^i) \quad (9)$$

can be interpreted as the *expected price*. ($\nabla \mathbf{p}_\alpha^i$ is an abbreviation for $\nabla \mathbf{p}_\alpha(\mathbf{z}_\alpha^i)$.) This would have been the equilibrium price if the current value of $\nabla \mathbf{p}$ would hold everywhere.

From the above we see that even though the termination condition is the same, the intermediate allocations, “the path to the solution”, will depend on the ordering of agents.

It is not hard to see that with only two commodities Eq. (8) reduces to

$$z_{\alpha 1}^{i+1} = z_{\alpha 1}^i - s \frac{p(z_{\alpha 1}^i) - \frac{\sum_{\beta=0}^n \frac{p(z_{\beta 1}^i)}{p'(z_{\beta 1}^i)}}{\sum_{\beta=0}^n \frac{1}{p'(z_{\beta 1}^i)}}}{p'(z_{\alpha 1}^i)}, \quad (10)$$

which is what was used in [Kurose and Simha, 1989] and [Ygge and Akkermans, 1996], except for the fact that in those papers only one commodity was explicit and therefore the marginal utility was used instead of the price, and the total allocation, x , was used instead of the change in allocation, z .

The update of $\mathbf{z}$ includes matrix inversion, and therefore the complexity of each iteration scales with the number of commodities as $\mathcal{O}(k^{2.496})$ [Pan, 1984]⁷. As the computation is made separately for each agent the complexity of each iteration scales with the number of agents, n , as $\mathcal{O}(n)$, and since it is a quadratic scheme⁸ the complexity is $\mathcal{O}(\log(-\log \epsilon))$, where ϵ is the error. The gradient matrixes are typically computed locally (analytically or numerically) by the participating agents and communicated to the auctioneer.

A delicate issue in resource-oriented schemes is the management of the boundaries.⁹ We will look at two different cases — the cases where

1. at least one commodity is unbounded, and
2. all commodities are bounded.

We believe the case where at least one commodity is unbounded to be a very realistic one. For example, it is possible to buy a car for more money than you possess through taking a loan. So within a reasonably wide range, we can assume money to be unconstrained. Some people might be very reluctant to take a loan – they have a big decrease in total utility for a negative value of money – but there is still no hard constraint. For example, if they had the opportunity, most people probably would accept taking a loan of \$100,000 if they had the opportunity to buy a ten million dollar house for that amount of money now or never. Convinced that it can be sold the day after with a considerable profit, taking the loan is acceptable.

⁷ $k^{2.496}$ is the current known record. There are, however, unconfirmed rumors about lower complexity.

⁸ When a quadratic scheme is used the error decreases as $\epsilon_i \leq c\epsilon_{i-1}^2$, for some constant c when the current allocation is sufficiently close to the solution [Press *et al.*, 1994]. Hence, the required number of iterations required for a given error scales as $\mathcal{O}(\log(-\log \epsilon))$. Under some standard assumptions the Newton scheme has quadratic convergence (see [Fletcher, 1987, p. 46, Theorem 3.1.1]).

⁹ When a price-oriented scheme is used this is managed locally by each agent — when giving a demand at a specific price level, the agent ensures that the demand does not violate its boundaries — and then this is no problem.

In the case where at least one commodity is not bounded, the procedure is to order the commodities such that an unbounded commodity becomes the k th commodity, i.e. the commodity which price can be set equal to one. For all other commodities, a RELAX scheme [Ibaraki and Katoh, 1988] is used. That is, first solve as if there were no boundaries, and then, for all commodities that happened to end up outside their boundaries, set to the boundary value and distribute the remaining resource among the other agents. One suggestion for how to distribute the rest term is to distribute it in proportion to the current allocation. In the worst case this may require n iterations, where n is the number of agents. In practice, however, we can expect a much better figure, especially after a few iterations when the rest terms are expected to be small.

Before each iteration, except for the first one, the following algorithm ensures that only the proper allocations are updated. Some comments are given after the pseudo-code.

```

boolean at_least_one_off_bound[nr_commodities] =
    {false, false, ..., false};
for each agent alpha
  for each commodity i
    if ((alpha.allocation[i] == alpha.lower_bound[i]) and
        (alpha.price[i] <= expected_price_from_prev_iter[i]))
      or
      ((alpha.allocation[i] == alpha.upper_bound[i]) and
        (alpha.price[i] >= expected_price_from_prev_iter[i]))
    then
      alpha.price_gradient[i,i] = infinity;
      alpha.price_gradient[i,j] = 0 for i != j;
    else
      at_least_one_off_bound[i] = true;
    endif
  endfor
endfor
for each commodity i
  if not at_least_one_off_bound[i] then
    exclude the commodity completely for this iteration
    compute the expected price as the average
      of the price of the agent at its lower boundary
      holding the highest price
      and the price of the agent at its upper boundary
      holding the lowest price
  endif
endfor
run the resource update algorithm

```

Hence, for each commodity i of an agent α that should not be affected in the next iteration, the rule is to set $(\nabla \mathbf{p}_\alpha(\mathbf{z}_\alpha))_{ii} = \infty$. (In practice, just use a value that is considerably higher than any gradient element that should be affected. For instance, we have employed a factor 10^{30} in our simulations.) This ensures that the allocation of that specific commodity will not be updated during the next iteration. If all agents avoid being affected with respect to one specific commodity, this commodity should simply be removed during the next iteration. Then, all agents being at their lower boundary giving a higher price than some agent being at its upper bound will reenter the scheme (and correspondingly for reentering agents at the upper bound).

Admittedly, the above is an *ad-hoc* algorithm, and even though it has turned out to work very well in practice, we have no theory guaranteeing convergence. For the two-commodity case, a more rigorous approach is, e.g., the BRELAX2 algorithm [Ibaraki and Katoh, 1988, pp. 29 – 30], which requires the unconstrained problem to be solved at most n times. In practice, however, this algorithm is likely to be far less efficient than the algorithm presented above, since the unconstrained problem is completely solved before boundaries are checked. The above algorithm,

on the other hand, checks the boundaries after each iteration of the resource update algorithm. Typically this will result in far fewer iterations. More theoretical studies are, however, required to verify that the approach always converges and to investigate whether or not the approach is always more efficient than BRELAX2. Presumably such an analysis is dependent on the used resource update algorithm. We are not aware of any approach corresponding to BRELAX2 for the multi-commodity case, even though it is most likely that such an algorithm exists.

The case, where there are boundaries on every resource, is considerably harder, and we do not have an efficient solution for it. A pragmatic approach to the problem might be to let $\partial u_\alpha / \partial z_n$ increase significantly when being close to, and below, the lower boundary and let it decrease correspondingly for the upper limit. This approach somewhat resembles so-called penalty functions that are sometimes used in constrained numerical optimization problems. In this way the allocation for the k th commodity will be forced not to cross the boundaries.

2.3 Discussion

Above we demonstrated how price-oriented as well as resource-oriented schemes can be used for the search for equilibrium in a multi-commodity market. The price-oriented scheme takes its inspiration from price tâtonnement, and the resource-oriented scheme is somewhat related to quantity tâtonnement (for a detailed discussion, see [Ygge, 1998]). We saw that the resource-oriented scheme is somewhat more complicated, but at the same time the inputs to the two approaches are different. In the first case the demand is the input and in the latter case the inputs are (some derivatives of) the utility function, the endowments and the boundaries. In standard micro-economic theory the utility function is the primary concept and the demand is derived from the utility function and the endowments. In the special cases where this can be done analytically, price-oriented approaches are normally preferred. However, in the general case, the demand function can not be analytically derived from the utility function and the endowments, and numerical methods must be used for obtaining the demand. In those cases resource-oriented approaches can be highly competitive. Another important difference is that resource-oriented algorithms are normally constructed such that the allocations are always feasible, i.e. after each iteration it is possible to allocate the computed allocations. Maintaining feasibility is a very useful property for time-critical computations [Ygge, 1998].

This presentation would, however, not be complete without a word of warning. Even though we have shown that the introduced resource-oriented algorithm performs excellently, a major drawback is its relatively complicated nature. Debugging the algorithm is a major effort. Few people have reliable intuitive feelings for what, e.g., different gradient matrices should look like, and, e.g., finding a bug resulting in linear rather than quadratic convergence can be very hard. Thus, when selecting an algorithm simplicity is an important factor, and high performance Newton algorithms are mainly interesting when simpler alternatives fall short for performance reasons. (For a more extensive discussion, see [Ygge, 1998].)

2.3.1 Simulations

In this section some simulation results are presented in order to give an impression of the properties of the algorithms. We use a simple agent model where each agent is holding a utility function

described by

$$u_{\alpha}(\mathbf{x}) = \sum_{i=1}^k a_{\alpha i} \left(1 - e^{-b_{\alpha i} x_i}\right). \quad (11)$$

The algorithm has been analyzed for a number of different settings. For each setting a simulation of 50 different markets (i.e. 50 different configurations of $a_{\alpha i}$, $b_{\alpha i}$, and initial allocations) has been performed. The parameters a , b and the initial allocation have been assigned randomly and the lower and upper boundaries have been set to 0.01 and 100 respectively for each commodity except for the k th commodity.

We use three measures for evaluating the resource update:¹⁰

1. The total amount of reallocated resource between any agents at each iteration. This figure is divided by the total amount of resource on the market in the presentations.
2. The worst allocation of any agent. This is normalized against the initial allocation. The formula is $\text{relative_utility} = (\text{current_utility} / \text{initial_utility}) - 1$. That is, the value of the initial allocation is 0.
3. The average normalized utility, computed with the above formula.

Figure 1: The total relative reallocation at each iteration. The numbers are average values for 50 different markets. Leftmost, the total reallocation with 25 agents and 3, 5 and 10 commodities is shown. Rightmost, the same measure with 3 commodities and 5, 10 and 25 agents is shown.

The total reallocated resource at each iteration is shown for some different configurations in Figure 1. The total reallocation is normalized against the total amount of resource on the market. In all cases, the reallocated amount decreases very rapidly. Note that the vertical scales are logarithmic. The normalized reallocation decreases faster with few commodities, whereas it seems to be practically independent of the number of agents. One should however note that as an average value the number might be somewhat misleading. As for some markets the reallocated

¹⁰It is easy to come up with criticism against each of these three measures. Measure 1: The different commodities can be of different scales. One unit of a certain commodity should maybe be compared with ten units of another. Measure 2&3: As utility is only a preference ordering, discussing different levels of improvement can be questioned. However, coming up with better alternatives is not as easy. We have also considered to use the sum of the absolute price differences as a measure, but we have concluded that this measure is of little importance – the quality of the outcome is the only important measure, the distance to the equilibrium is only secondary.

resource will decrease very rapidly, the average can often be regarded as the largest reallocation divided by 50 (the number of simulated markets). The variation is quite large and two extreme curves are shown in Figure 2. However, in all cases we have a very rapid decrease.

Figure 2: Two extreme curves representing the relative reallocation at each iteration. For one of the functions the quadratic decrease of the error starts already after 4 iterations, whereas it takes until iteration 80 for the other function.

The next interesting measure, the worst allocation for any agent is shown in Figure 3. As soon as the worst utility is above zero, the algorithm has led to a Pareto improvement, i.e. the utility has increased for every agent. We see that for all cases there is a Pareto improvement already after very few iterations.

Figure 3: Leftmost the worst relative utility of any agent in the 50 random markets with 25 agents and 3, 5 and 10 commodities is plotted. Rightmost is the worst relative utility of any agent in the 50 random markets with 3 commodities and 5, 10 and 25 agents.

In Figure 4, the third interesting measure, the average relative utility improvement, is shown. The average relative utility increases very rapidly during the first few iterations. The major improvement is indeed during the first 3 to 10 iterations. As a typical example, with 10 commodities and 25 agents, the improvement of the second iteration is approximately half the improvement of the first iteration, and the improvement of the eighth iteration is approximately a tenth of the improvement of the first iteration.

In sum, the simulations show that for this example, *very few* iterations (typically less than 10) are required in order to achieve a high quality outcome.

A natural question is how well a price-oriented algorithm would perform for this example. The issue is, however, not how to implement the algorithm for this example, but what to compare. Intermediate solutions in price-oriented algorithms are non-feasible and measures such as measure 2 and 3 above have no meaning – the only measurable quantity is the distance to equilibrium in terms of aggregate excess demand. In passing we just mention that there are highly efficient price-oriented algorithms as well (if the demand function is easily obtained). For example, with the algorithm in Eq. (2) the complexity of each iteration scales with the number of agents, n and then number of commodities, k as $\mathcal{O}(n \cdot k^{2.496})$, and the number of iterations scales with the acceptable error, ϵ as $\mathcal{O}(\log(-\log \epsilon))$ (computations of demand not considered). Thus, asymptotic behavior for these price- and resource-oriented approaches are the same, though constant factors may of course be different. We refer to [Ygge, 1998] for a more thorough discussion.

Figure 4: Leftmost we have the average relative utility with 25 agents and 3, 5 and 10 commodities. The process of improvement is somewhat slower when the number of commodities grows, but still excellent results are obtained in order ten iterations. Rightmost we have the average relative utility with 3 commodities and 5, 10 and 25 agents. The average utility improvement seems somewhat larger with a larger number of agents.

3 Conclusions

Multi-commodity markets are conventionally computed by means of price-oriented algorithms. In this paper we have demonstrated in detail how resource-oriented schemes can be successfully adapted to deal with such multi-commodity markets.

The most delicate aspect of resource-oriented schemes is to properly deal with the local resource boundaries. On the other hand, an advantage of the resource-based schemes is that they can start directly from (derivatives of) the agents' utility functions, in contrast to the price-oriented algorithms. The latter includes solving an optimization problem to obtain the demand functions which is not always easily possible. Thus, a search for general equilibrium prices can result in an iterative search over prices *plus*, for each step of the algorithm, iterative searches for the demands of the involved agents. This can of course be very inefficient.

Another difference is that intermediate results of price-oriented algorithms are non-feasible. The fact that the presented resource-oriented approach maintains feasible allocations is a useful property in time-critical market computations. Such real-time aspects are relevant in industrial applications such as power load management, in the context of which the present work is carried

out [Ygge and Akkermans, 1996, Ygge and Akkermans, 1997, Ygge, 1998].

Our new resource-oriented algorithm has been simulated and tested on markets with a varying number of both agents and commodities. In general the calculated results are very good; convergence is essentially achieved already within ten iterations. Also the scaling properties with the number of both agents and commodities appear to be very satisfactory: asymptotically the complexity of each iteration scales as $\mathcal{O}(n \cdot k^{2.496})$, where k is the number of commodities and n is the number of agents, and the number of required iterations scales as $\mathcal{O}(\log(-\log \epsilon))$, where ϵ is the error. This corroborates the view that resource-oriented algorithms represent a useful alternative to the standard price-directed schemes in computational markets. Price- and resource-oriented algorithms are dual, complementary approaches to multi-commodity market computations.

Source Code The source code of the presented algorithm, and in particular the demonstrated example, is downloadable through the world-wide-web from: <http://www.enersearch.se/~ygge/>

Acknowledgements We thank Hans Ottosson at EnerSearch and Rune Gustavsson at the HKR for all their support, and Hans Olsson at Lund University for help on several numerical analysis issues.

References

- [Andersson and Ygge, 1998] A. Andersson and F. Ygge. Managing large scale computational markets. In H. El-Rewini, editor, *Proceedings of the Software Technology track of the 31th Hawaiian International Conference on System Sciences (HICSS31)*, volume VII, pages 4 – 14. IEEE Computer Society, Los Alamos, January 1998. ISBN 0-8186-8251-5, ISSN 1060-3425, IEEE Catalog Number 98TB100216. (Available from <http://www.enersearch.se/ygge/>).
- [Cheng and Wellman, 1998] J. Cheng and M. P. Wellman. The WALRAS algorithm – a convergent distributed implementation of general equilibrium outcomes. In *Computational Economics*, 1998. To appear. (Available from <http://ai.eecs.umich.edu/people/wellman>).
- [Clearwater and Huberman, 1994] S. Clearwater and B. A. Huberman. Thermal markets for controlling building environments. *Energy Engineering*, 91(3):25–56, 1994.
- [Fletcher, 1987] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987. Second Edition.
- [Golub and Loan, 1991] G. H. Golub and C. F. Van Loan. *Matrix Computation*. John Hopkins University Press, 1991. Second Edition.
- [Hu and Wellman, 1996] J. Hu and M. P. Wellman. Self-fulfilling bias in multiagent learning. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems ICMAS'96*, pages 118–125. AAAI Press, Menlo Park, CA, December 9–14 1996.
- [Huberman and Clearwater, 1995] B. A. Huberman and S. Clearwater. A multi-agent system for controlling building environments. In *Proceedings of the First International Conference on Multi-Agent Systems ICMAS'95*, pages 171–176. AAAI Press / The MIT Press, Menlo Park, CA, June 12–14 1995.
- [Hurwicz, 1986] L. Hurwicz. On informationally decentralized systems. In C.B McGuire and R. Radner, editors, *Decision and Organization*, chapter 14, pages 297–336. University of Minnesota Press, 1986. Second edition.

- [Ibaraki and Katoh, 1988] T. Ibaraki and N. Katoh. *Resource Allocation Problems – Algorithmic Approaches*. The MIT Press, 1988.
- [Kurose and Simha, 1989] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.
- [Lenting and Braspenning, 1994] J. Lenting and P. Braspenning. An all-pay auction approach to resource allocation. In A. Cohn, editor, *Proceedings of The 11th European Conference on Artificial Intelligence, ECAI '94*. John Wiley & Sons, Ltd, 1994.
- [Mas-Colell *et al.*, 1995] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [Miller and Drexler, 1996] M. S. Miller and K. E. Drexler. Markets and computation: Agoric open systems. Available at: <http://www.agorics.com/agorics/agoricsPapers/aos/AOS0.html>, 1996.
- [Pan, 1984] V. Pan. How can we speed up matrix multiplication? *SIAM Review*, 26:393–416, 1984.
- [Press *et al.*, 1994] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1994. Second Edition.
- [Sandholm and Ygge, 1997] T. W. Sandholm and F. Ygge. On the gains and losses of speculation in equilibrium markets. In *Proceeding of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97*, August 23–29, 1997. (Available from <http://www.enersearch.se/ygge>).
- [Shoven and Whalley, 1992] J. B. Shoven and J. Whalley. *Applying General Equilibrium*. Cambridge University Press, New York, 1992.
- [Takayama, 1985] A. Takayama. *Mathematical Economics*. Cambridge University Press, 1985.
- [Wellman, 1993] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research* (<http://www.jair.org/>), 1(1):1–23, 1993.
- [Wellman, 1994] M. P. Wellman. A computational market model for distributed configuration design. In *Proceedings of AAAI '94*, pages 401–407. Morgan Kaufmann Publishers, San Francisco, CA, 1994. (Available from <http://ai.eecs.umich.edu/people/wellman/Publications.html>).
- [Wellman, 1996] M. P. Wellman. Market-oriented programming: Some early lessons. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*, chapter 4. World Scientific, 1996. (More info about the book at http://www.wspc.co.uk/wspc/Books/compai_hilites.html#mbc. The paper is available from <http://ai.eecs.umich.edu/people/wellman/Publications.html>).
- [White, 1994] J. E. White. Telescript technology: The foundations for the electronic market place. Technical report, General Magic Inc., 2465 Latham Street, Mountain View, CA 94040, USA, 1994.
- [Yamaki *et al.*, 1996] H. Yamaki, M. P. Wellman, and T. Ishida. A market-based approach to allocating QoS for multimedia applications. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems ICMAS'96*, pages 385–392. AAAI Press, Menlo Park, CA, December 9–14 1996. (Available from <http://ai.eecs.umich.edu/people/wellman/Publications.html>).
- [Ygge and Akkermans, 1996] F. Ygge and J. M. Akkermans. Power load management as a computational market. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems ICMAS'96*, pages 393–400. AAAI Press, Menlo Park, CA, December 9–14 1996. (Available from <http://www.enersearch.se/ygge>).
- [Ygge and Akkermans, 1997] F. Ygge and J. M. Akkermans. Making a case for multi-agent systems. In M. Boman and W. Van de Velde, editors, *Proceedings of MAAMAW '97*, pages 156–176. Springer Verlag, Berlin, May 13–16 1997. ISBN-3-540-63077-5, (Available from <http://www.enersearch.se/ygge>).
- [Ygge, 1998] F. Ygge. *Market-Oriented Programming and its Application To Power Load Management*. PhD thesis, Department of Computer Science, Lund University, 1998. (Available from <http://www.enersearch.se/ygge>).

A Proofs

In this section we prove that

$$\mathbf{z}^{i+1} = \mathbf{z}^i - s \cdot \left(\nabla \mathbf{f}(\mathbf{z}^i) \right)^{-1} \cdot \mathbf{f}(\mathbf{z}^i),$$

with

$$\mathbf{f} = \begin{bmatrix} \mathbf{p}_1(\mathbf{z}_1) - \mathbf{p}_n(\mathbf{z}_n), \mathbf{p}_2(\mathbf{z}_2) - \mathbf{p}_n(\mathbf{z}_n), \dots, \\ \mathbf{p}_{n-1}(\mathbf{z}_{n-1}) - \mathbf{p}_n(\mathbf{z}_n) \end{bmatrix}^T,$$

can be reduced to

$$\mathbf{z}_\alpha^{i+1} = \mathbf{z}_\alpha^i - s \cdot \diamond \mathbf{p}_\alpha^i \left(\mathbf{p}_\alpha^i - \mathbf{p}_n^i - \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i (\mathbf{p}_\beta^i - \mathbf{p}_n^i) \right),$$

where $\mathbf{p}_\alpha^i$, $\nabla \mathbf{p}_\alpha^i$, and $\diamond \mathbf{p}_\alpha^i$ are abbreviations for $\mathbf{p}_\alpha(\mathbf{z}_\alpha^i)$, $\nabla \mathbf{p}_\alpha(\mathbf{z}_\alpha^i)$, and $(\nabla \mathbf{p}_\alpha(\mathbf{z}_\alpha^i))^{-1}$ respectively.

First introduce the matrices $\mathbf{A}$, $\mathbf{S}$, $\mathbf{T}$, and $\mathbf{R}$ of dimension $(n-1 \times n-1)$, defined by

$$\begin{aligned} A_{rc} &= \begin{cases} \nabla \mathbf{p}_r^i, & r = c \\ \mathbf{0}, & r \neq c \end{cases}, & S_{rc} &= \begin{cases} \mathbf{E}, & r = c \\ \mathbf{0}, & r \neq c \end{cases}, \\ T_{rc} &= \begin{cases} \mathbf{E}, & c = 1 \\ \mathbf{0}, & c \neq 1 \end{cases}, & \text{and} & R_{rc} = \begin{cases} \nabla \mathbf{p}_n^i, & c = 1 \\ \mathbf{0}, & c \neq 1 \end{cases}, \end{aligned}$$

where $\mathbf{E}$ is the unit matrix defined by

$$E_{rc} = \begin{cases} 1, & r = c \\ 0, & r \neq c \end{cases}.$$

Then $\nabla \mathbf{f} = \mathbf{A} + \mathbf{RST}^T$, and from the Sherman-Morrison formula [Golub and Loan, 1991] we have that $(\nabla \mathbf{f})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{R} \mathbf{U}^{-1} \mathbf{T}^T \mathbf{A}^{-1}$ where $\mathbf{U} = \mathbf{S}^{-1} + \mathbf{T}^T \mathbf{A}^{-1} \mathbf{R}$. Since the inversion of $\nabla \mathbf{f}$ now only contains inversion of diagonal matrices, it can be done analytically. The result is

$$(\nabla \mathbf{f}(\mathbf{z}^i))_{rc}^{-1} = \begin{cases} \diamond \mathbf{p}_r^i \left(\mathbf{E} - \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \diamond \mathbf{p}_c^i \right), & r = c \\ -\diamond \mathbf{p}_r^i \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \diamond \mathbf{p}_c^i, & r \neq c \end{cases}$$

Then, the change in the net demand for agent α is

$$\begin{aligned} \Delta \mathbf{z}_\alpha &= \\ &-s \cdot \diamond \mathbf{p}_\alpha^i \cdot \left(\mathbf{E} - \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \diamond \mathbf{p}_j^i \right) (\mathbf{p}_\alpha^i - \mathbf{p}_n^i) \\ &+s \cdot \diamond \mathbf{p}_\alpha^i \cdot \left(\sum_{1 \leq \gamma \leq n-1 \wedge \gamma \neq \alpha} \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \diamond \mathbf{p}_\gamma^i \right) (\mathbf{p}_\gamma^i - \mathbf{p}_n^i) \end{aligned}$$

$$\begin{aligned}
&= \\
&-s \cdot \diamond \mathbf{p}_\alpha^i \left(\mathbf{p}_\alpha^i - \mathbf{p}_n^i - \nabla \mathbf{p}_n^i \left(\sum_{\beta=1}^n \diamond \mathbf{p}_\beta^i \right)^{-1} \diamond \mathbf{p}_n^i \sum_{\gamma=1}^n \diamond \mathbf{p}_\gamma^i \left(\mathbf{p}_\gamma^i - \mathbf{p}_n^i \right) \right).
\end{aligned}$$

□