Ygge and Akkermans: Background

Yevgeniy Vorobeychik

1 Introduction

It is a very natural habit of mankind to search for easy answers to hard problems. However, such discoveries are sufficiently rare to be on guard whenever a new one appears, and yet it is all too easy to allow the power of apparent convenience lure a sound mind from careful inspection of the solution. Thus, when Huberman and Clearwater [2, 6] discovered a market-based method for controlling a building environment that considerably outperformed a mainstream controller, few questioned the miraculous improvement and fewer yet wondered precisely which of the techniques used by the two authors effected the improvement. To step back for a moment, it seems a matter of common sense that any centralized approach, given sufficient information and computation time, should produce a result at least as good as any decentralized approach. Thus, the fact that the Huberman and Clearwater's market-based controller produced a better solution than a centralized controller should have been suspicious. Furthermore, there was little attempt on the part of the authors to explain their results beyond attributing it to the powers of market-based problem solving techniques. Interestingly, despite their use of the techniques from the field of Economics to solve a resource allocation problem, many of the critical details, such as utility and bid formation, were arbitrary enough to preclude any useful analyses using well-known tools from the latter field.

As Ygge and Akkermans [29] dissected the techniques and results from Huberman and Clearwater, they discovered that indeed it was not the market-based techniques that produced a remarkable improvement in the controller performance, but the information that was utilized. They then went on to create a centralized and a market-based controller that performed even better. The mystery was solved: a centralized controller with global information will do as well as any decentralized controller, and, conversely, a market-based controller can be designed to find an equivalent solution.

So now that the problem of building environments had been sufficiently analyzed, two questions remain: what kinds of generalizations can be made based on this and other experience and what lessons can we learn from this? Before attempting to answer these, we must first understand the context in which they are raised. Thus, we will begin with Economics, the field that for many years now has been devoted to understanding the principles of markets. This will be the subject of Section 2. Afterwards, we shall proceed to understand the tradeoffs between centralized and decentralized systems in Section 3. Section 4 will define the general resouce allocation problem and present a few common examples from Computer Science and Economics, and in Section 5 we will survey the approaches to resource allocation, providing a brief overview of classical Computer Science and Operations Research techniques, and moving on to discuss in some detail the many market-based approaches to date.

2 Overview of Economic Principles

Since market-based approaches to resource allocation are clearly rooted in Economics, this seems like a good place to start our discussion. Fundamentally, Economics is concerned with allocating the limited resources in the world, such as food and raw materials, among the many self-interested economic agents. On the micro (per agent) level, each agent derives some amount of happiness from being allocated some bundle of resources (goods), and the assumption is typically that the agents are autonomous. Given some initial edowment of goods, agents are presumed to trade until the *market* reaches *equilibrium*. Intu-

itively, when the market is at equilibrium, it is in some way stable. This stability is usually in terms of *prices*, since anyone who tries to charge more for some good than its equilibrium price would somehow be made unhappy by the resulting transaction. The abstract model of Economics markets just described obviously leaves out many important details, but it is the idea that interests us right now, and we'll fill in the details relevant to our discussion later. What is important to note is that if markets can reach this wonderful equilibrium, and if it so happens that this equilibrium is also *optimal*, the world can be a fairy tale where everyone is happy and pizza grows on trees. But even if such worlds may not in reality exist, as some may speculate, Engineers who prefer creating microcosms of reality may just find it useful to apply Economic models and analysis tools to solving artificial problems. It is the latter use that concerns us here, and, after describing the relevant ideas and concepts from Economics, we will discuss some of the research that makes use of these.

2.1 Utility and Value

In our discussion in the previous paragraph, we omitted a critical detail: how is it that an agent is made happy? Or, more specifically, what does it mean for an agent to attain different levels of happiness? We will now fill in this little detail. Intuitively, happiness of an agent should increase as it gets more of some resource, or good. Furthermore, some combinations of goods will make it more happy than others. Since we often want to reason rigorously about economic scenarios (especially if we are trying to design markets to solve our problems, as Engineers would like to do), we need to formalize happiness and relative happiness. Utility is just such a formalization, as it converts an abstract notion of happiness to a real number, with a higher utility value meaning a higher relative happiness.

It turns out that in our analyses we are usually concerned with *marginal utility*, which is simply the first derivate of the utility function[1]. Thus, marginal utility indicates how much better off, or happier, we will be if we get an additional unit of the good. By the way, we will use utility and value interchangeably, as is common in relevant literature. Also, note that utility provides us with a convenient way to understand the meaning of rational as applied to agents. A rational agent can be defined as any agent that maximizes its utility.

[1] As just described, this function maps amounts of a good to real numbers indicating relative preference for this amount, or bundle, to others. It is usually assumed that only relative magnitudes matter.

2.2 Money and Prices

We have alluded to prices in the discussion of economic mechanisms, and will now state specifically what they generally signify in Economics. A price of a good is the amount of a particular other good that can be traded for one unit of this good. Obviously, if there is only one good in an economy, prices are a moot point. However, in general they signify tradeoffs for agents. The idea of money is related to prices. In the above definition, one may need to have a price for every possible exchange of two goods. This is clearly inefficient, since, for n different goods, one would need to publish $O(n^2)$ prices! A way around this is to use a level of indirection, which is the purpose that money plays. Instead of providing a price of each good in terms of every other, we could simpy give a price in terms of some common reference good.

2.3 Supply and Demand

Economists refer to supply as a function that maps prices to quantities of a good. In other words, for a set of prices, supply function (or curve) would produce quantities of this good that will be provided for purchase at the market for the corresponding prices. Similarly, demand function produces the quanities that will be purchased at each price point.

2.4 Efficiency

Allocating resources among agents is not inherently difficult, since one could arbitrarily assign these to agents. What makes the problem difficult (and interesting) is the optimality requirement, i.e. how can we allocate resources such that the global utility is maximized?

Economic efficiency is synonymous with optimality, as it provides a lower bound on the meaning of a desirable outcome. The most common idea of efficiency in Economics is that of Pareto efficiency. By

definition, a Pareto efficient outcome provides no opportunity to reallocate resources among agents with the result that makes some agent better off without making some other worse off. To see why this is optimal, one can envision an outcome that is not Pareto efficient. In such a scenario, one could improve global utility by reallocating resources in a way that makes some agent better off and no one else worse off. Trivially, then, the non-Pareto efficient outcome just described cannot be considered optimal.

2.5 Equilibrium

Equilibrium usually connotes some form of stability. In Economics, this is not necessarily the case, since equilibrium merely refers to an outcome in which total quantity of a good supplied equals total quanity demanded. Since both supply and demand are functions of price, equilibrium price and quanitity fully describe the market equilibrium point.

In an Economy with many goods, it is convenient to talk about *partial* and *general* equilibrium. *Partial Equilibrium* is an equilibrium for a single good. Under *General Equilibrium*, each good in the market is in Partial Equilibrium.

2.6 Perfectly and Imperfectly Competitive Markets

Perfect is a very positive, optimistic word. And for good reasons: if something is perfect, it can't possibly be bad. Appropriately, Economists reserved the word "perfect" to describe a world very special indeed, in which, as we've mentioned, various appealing artifacts grow on trees. Perfect competition in Economics refers to markets that are teeming with rational agents, each having negligible impact on the price. Given the market price that is taken as exogenous by the agents, each agent strives to maximize local utility and, as a result, a market emerges that, upon reaching equilibrium is Pareto efficient. While this isn't the entire story (some assumptions need to be made about utility functions of agents, etc.), it is sufficient to make an important point: a system that is completely decentralized can, under certain assumptions, achieve an optimal state. Since the resulting optimal outcomes can be analyzed using standard tools from Economics, it is a very useful property to system designers: problem solving can be decentralized with no loss in solution quality!

The complement to perfect competition is, well, imperfect competition, which includes all the other possibilities, like monopoly (one producer of a good) or oligopoly (few producers of a good). We will not concern ourselves with these market forms here, but will instead concentrate for the remainder of this discussion on perfectly competitive markets. For a more detailed discussion of Microeconomics principles in general and market organization in particular, see [18].

3 Centralized versus Decentralized Systems

Having devoted Section 2 to the discussion of Economics, we would like to change gears and devote some attention to the argument of centralized versus decentralized systems and control. While we present these as orthagonal subjects, it may be of some value to keep in mind that competitive markets that we had just described are examples of fully decentralized systems.

Systems are said to be centralized or decentralized with respect to the location of control functions. In centralized systems, control resides one a single processor, while in decentralized systems it is distributed across mutliple processors [4]. An important observation here is that, given sufficient information, time, and computation resources, centralized solutions are at least as good as decentralized. It is easy to see that this is the case, since any functionality that is decentralized can be simulated centrally. Of course, the assumption of sufficient information, time, and computation resources does not generally hold, and so it does often make sense to decentralize. For example, detailed global information may be decentralized across many agents and it is either too costly to communicate all of it to a central authority, or the agents may be unwilling to reveal their private preferences. Additionally, diffucult but (relatively) easily parallelizable problems will often lend themselves to a decentralized approach, as long as the resulting communication overhead is not too high.

Due to the conceptual attractiveness of decentralized problem solving, many arguments have been

made in its favor that result in undue generalizations. One of such arguments is that centralized systems suffer from a single point of failure, implying that decentralized systems do not. The former argument is certainly true, but the implication is a red herring. Ygge [27] points out that it is often the case that decentralized systems have a lower success probability than the equivalent centralized systems. Making decentralized systems fault tolerant is often extremely expensive (again, Ygge analyzes the fault tolerant example from Kurose and Simha [8]). Tannenbaum [17] also shows that a distributed algorithm for mutual exclusion turns a single point of failure into n points of failure, and even with several tricks to eliminate this problem, the equivalent centralized algorithm is superior to the distributed one. The bottom line here is, this is the game of tradeoffs, kind of like the general resource allocation problem...

4 Resource Allocation Problem

As we have mentioned earlier, resource allocation problem is finding the optimal allocation of limited resources. In this problem, we are partitioning the resources among the many agents with different utilities for these resources. We want to find a partition such that the sum of all resulting utilities is maximized. Note that Pareto efficiency is a sufficient property for a particular allocation to be optimal by this definition.

Researchers in Computer Science, Operations Research, and Economics are all too familiar with the typical resource allocation problems and have developed a myriad of techniques for solving them. While Computer Science and Operations Research are mostly concerned with optimal allocation in the context of system design, Economists tend to look at things from an observer's perspective. Economics, thus, attempts to explain how resources are allocated among rational agents in a decentralized Economy. Whether Economics is accurate in describing the world is of supreme importance to Economists, but a lot less important to Engineers who often have control over the mechanism as well as the agents that are used to solve a problem. We will return to this when we discuss market-based approaches to resource allocation below.

5 Approaches to Resource Allocation

As we have just mentioned, the problem of resource allocation occurs in its most explicit form in three disciplines: Computer Science, Operations Research, and Economics. This is not to say that other disciplines do not encounter this problem, but it appears to have annoy the researchers in these three fields enough to develop a number of solution or approximation techniques. Since we are here concerned with approaches that either originated in or made their way into Computer Science, we will restrict our attention to techniques involving computational systems.

5.1 Classical Computer Science Approaches

Anyone who has ever seen or heard their computer thrash understands the true struggle for limited resources. It is often a vicious struggle, with heroes and villains on both sides, Microsoft Office and Visual Studio against Windows Media Player and Kazaa, with a poor wreck of a humanoid watching in despare the frozen windows and mouse pointer and listening to the waves of curses from the hard drive. Limited resources hardly ever stop anyone from abusing the little they may have–after all, what could be wrong with installing Windows 2000 on a computer with 32MB of RAM and a 2GB hard drive, and then running a web and file server on top of it? While some problems are not subject to any feasible solutions, in general it pays in Computer Science to sqeeze out as much efficiency from the limited computing resources. However, it is recognized that finding a completely optimal solution is often infeasible, since searching the large solution space (which is the worst case scenario) is not realistic, and many heuristics have emerged as reasonable approximations.

Let's take caching as an example. Assuming that utility in this context is inversely proportional to latency, the optimal allocation is such that caches a block (or a page) that will be accessed next. The problem, of course, is that one would have to predict access patterns with 100% accuracy to achieve this, which is clearly impossible. The most common approximation is LRU, or Least Recently Used cache replacement policy, which relies on the assumption

that there is high access locality, and data accessed most recently will likely be accessed again in the near future. Enhancements to this approximation have been attempted, such as better predictive algorithms (e.g. [7]) and a more sophisticated cost-benefit analyses to optimize the use of caches through prefetching as seen in [19, 10].

Another common problem faced by Computer Scientists is process scheduling, which is just a specialization of resource allocation problem (time shares being the limited resources in the system). The simplest and probably most common way is some form of First Come First Serve. A slight enhancement to FCFS is to combine it with priority scheduling. Priority scheduling tends to be a much better approximation of the optimal outcome, since it actually takes utilities into account (it's a greedy algorithm that schedules in decreasing order of utility). Unfortunately, priority scheduling suffers from starvation: some processes may never be scheduled. Additionally, greedy strategy is merely an approximation. Finally, detailed information about process utilities may not be available. An interesting approach presented by Waldspurger and Weihl [20] is to use lottery scheduling. In this approach, processes are provided a number of lottery tickets proportional to their utility (priority). Each scheduling decision selects the winner randomly. Probability of each process being selected is proportional to the number of tickets it holds. While a considerably more flexible scheme than the classic FCFS with priority, lottery scheduling still suffers from some of the same shortcomings: processes are still subject to starvation (although in the limit each process should get its proportional share of CPU time), and this, as well as the other schemes we have thus far described, is centralized. Furthermore, none of these schemes is easily generalizable, and outcomes of these do not easily lend themselves to analyses.[2]

5.2 Operations Research Techniques

The optimization techniques from Operations Research are widespread and many other disciplines rely on these to solve their resource allocation problems. These may be used for local as well as global optimization, and, thus, can be used in combination with the other methods discussed in Section 5. As the techniques in OR are not important for our discussion (they are just the tools we could use), we merely gloss over several of them here and leave it to the interested reader to pursue other sources (e.g. [25]) for more information.

One of the best known linear optimization techniques is Linear Programming. A linear program consists of an objective function which is to be maximized (minimized) and a set of inequalities signifying constraints. Linear Programming is known to be P-Complete, and so is relatively computationally expensive. In dealing with discrete domains, linear programs become linear integer programs, and the problems becomes NP-Complete (non-linear optimization equivalents to these are Nonlinear Programming and Nonlinear Integer Programming).

Another common optimization technique is Dynamic Programming, which attempts to decompose problems and then solve subproblems and recursively collect solutions.

Yet another approach is gradient climbing. This technique simply follows the direction of steepest ascent or descent and hopes to arrive at an optimum when the value function levels off. Unfortunately, gradient descent (ascent) may find poor local minima (maxima), and random restart algorithms are often used to alleviate this problem.

5.3 Market-Based Approaches

Finally, we have reached what my be the focal point of this paper: market-based approaches to resource allocation. The first observation about these is that they rely on the ideas and methods from Economics to solve problems that arise in the context of Computer Science (at least as far as we are concerned). In using these ideas and methods, researchers can fall back on the immense body of work and expertise in developing and analyzing decentralized models of computation. As the concepts of object- and agent-oriented programming gain ground, the synthesis of the two fields, and, hopefully, synergy that results from it, become inevitable.

While Economics does not prescribe a particular mechanism design, it is often convenient for Computer Scientists to use auctions for this purpose. Auc-

[2] For a more detailed treatment of classical Operating System problems such as caching, memory management, and scheduling, consult [14].

tions allow systems designers to control rules of negotition and outcomes, and, as a result, be able to reason about the outcomes. The latter property is paramount to designers, who would usually prefer knowing exactly what behavior the systems they design will exhibit.

5.3.1 Auction-Based Systems

Sutherland was one of the early venturers from the domain of Computer Science into Economics. In his paper [16] he describes an auction-based approach for assigning time slots on a computer to people based on their bids. Money with which bids were made was artificial, with amounts assigned based on project priority.

In 1995 Gagliano published a paper that described the aution-based resource allocation algorithm and simulation results [4]. This paper was primarily a study of feasibility of a decentralized resource allocation algorithm, and the finding was positive: auction-based scheme worked reasonably well.

Another simulation study was presented by Steiglitz et al [15]. The authors of this work simulated an Economy with two goods: food and gold. Each agent had exponential utility functions for food and gold, though the choice of utility functions as well as bids that were derived from them was somewhat arbitrary. Agents would also have some production capacities for both of these goods and each day would decide based on food reserves whether to offer food for sale or to purchase food from other selling agents. The trades between agents took place using an auction to which bids for food were submitted (utilities normalized by prices of food in terms of gold). Clearing rules in this auction select the price at which several constraints are satisfied (supply approximately equals demand).

Huberman and Clearwater [2, 6] used the utility and bid functions chosen by Steiglitz et al. to bootstrap their market-based building environment controller. Again, auction received supply and demand bids from agents who either had their valves open too much or not enough both with respect to their local setpoints as well as average deviation from setpoint in other offices [3]. Having achieved considerable improvement over the standard PID controller, Huberman and Clearwater did not attempt to pinpoint the precise sources of this improvements. They also did not explain why their approach would be inherently better than a centralized one, and, in fact, this was shown to be a fallacy in the later work by Ygge and Akkermans [29]. Finally, the market-based approach used by Huberman and Clearwater did not lend itself to any meaningful Economic analysis, and so their outcomes were, well, a kind of magic...

5.3.2 Resource and Price Adjustment Approaches

In our earlier discussion of Economics we had noted that ultimate global happiness takes place at equilibrium. Pareto efficiency, which is the formal name for ultimate global happiness, is highly desirable in problem solving, and as a result several methods emerged for iterative convergence to equilibrium. These methods can be classified as resource-directed and price-directed approaches [8]. In resource-directed approaches, agents compute their marginal utilities given current allocations and then trade units of resource between low and high marginal utility agents. The process repeats until no viable trades can take place. Price-directed approaches let each agent compute its quantity demanded or supplied (which can be expressed as negative quantity demanded) as a function of price. The market then goes through a series of price adjustments until total quantity demanded equals total quantity supplied. To achieve General Equilibrium, the price adjustment needs to be repeated iteratively for all goods, until all markets reach equilibrium.

Kurose and Simha [8] used a resource-directed approach to solve a decentralized file allocation problem. They explored three algorithms for file allocation: one using a first derivative of marginal utility in addition to marginal utility itself, another using also a second derivative, and the third using pairwise interaction between nodes.[4] They found that their algorithms converge to optimal allocation, and the second derivative algorithm converges consider-

[3] Indeed, it was the addition of global information to the problem that made their controller superior to the standart controllers.

[4] File allocation problem (FAP) determines how a file (which is divisible) or a file system can be allocated among nodes in the network to minimize expected cost of file access, which is a combination of communication cost and delay. Utility in this case is negative cost.

ably faster than the other two.

Cheng and Wellman [1] describe a WALRAS price-directed algorithm, the idea for which was suggested by Leon Walras in 1874. In this algorithm, agents compute their demand functions for each good and send them to auctioneers for these goods. Auctioneers computes a market clearing prices for their respective goods and notify agents of these. Agents adjust their demand functions to account for this new information and submit new bids to the auctioneers. This iterative process continues until changes in prices are below some threshold. Since this algorithm is asynchronous, it is very unlikely that many agents will simultaneously bid for a single good, and as a result there is less price oscillation[5]. Cheng and Wellman prove that the WALRAS algorithm converges to equilibrium if there are no complementarities between goods and preferences are strictly convex. While the publication date of this paper is 1998, the WALRAS algorithm was used by Wellman et al. in their work on Market-Oriented Programming since 1993 [9, 22, 23, 24, 26]. In the next section, we delve into this area of research in some more depth.

5.3.3 Market-Oriented Programming

As market-based methods slowly encroached upon Computer Science in the context of distributed problem solving, Market-Oriented Programming (MOP) emerged as a programming paradigm. The goal behind it was to create a constrained agent-based framework that would decentralize problem solving and could, under some assumptions, be guaranteed to produce Pareto optimal outcomes. As a programming paradigm, MOP has its roots in Object- and Agent- Oriented Programming, and as a problem solving tool, it can be viewed in the context of Davis and Smith's Contract Net [3, 12], as well as other market-based approaches that predated it. In the subsection entitled "Roots", we will attempt to trace out the roots of MOP, and in the following subsections we will describe the paradigm in more detail and survey several applications of it.

Roots As we have already mentioned, MOP's roots can be traced back to Object- and Agent-Oriented Programming.

[5]Prices could oscillate if many agents attempt to bid for the same good after receiving price updates.

Object-Oriented Programming (OOP) has become a dominant programming technique in Computer Science, as it allows considerable flexibility and modularity in software design. The central idea of OOP is that systems can be viewed in terms of components, or objects, that have some internal (private) state and expose a well-defined interface which allows components to act upon each other.

It may often be useful to impose constraints on top of OOP in order to create a paradigm that is more meaningful in a particular domain. This was done by Shoham in Agent-Oriented Programming (AOP) [13]. AOP uses an agent as a unit of abstraction. Agents, just like objects, have internal state, but this state is constrained to the agent's beliefs, commitments, and choices. The interface of the agent in AOP is constrained to specific actions like inform, request, offer, promise, decline, and the agents are assumed to be honest and consistent.

On the problem-soliving side of the Computer Science world, MOP borrows ideas from Contract Net [3, 12]. The idea behind Contract Net is fairly simple: different agents may specialize in solving different subproblems, and if some agent has a problem to solve, it may as well distribute it among agents that can solve parts of it (subproblems) very well. To do this, the agent that has a problem (manager) broadcasts subproblems to other agents (contractors), who may then submit bids to the manager for a subproblem they would like to solve. After a round of bidding, the manager assigns each subproblem to the contractor with the lowest bid. Since each subproblem can be further subdivided into more subproblems, each contractor can subsequently become a manager and each problem may in the end be recursively distributed among many agents. One may notice that the Contract Net protocol does not specify how the bids are formed. This detail was filled in by Sandholm in [12], where he proposed to use Marginal Cost as the basis for bidding.

Environment Market-Oriented Programming environment separates the world into goods, which are sold at auctions, and agents, which submit bids (sell or buy) for goods. The actual bids are demand functions, which agents compute using their local utility functions and current prices of goods. All auctions proceed according to the WALRAS algorithm

described above and in Cheng and Wellman, [1]. Thus, once prices at all auctions no longer change, or once the change in prices is below some threshold value, General Equilibrium is presumed and auctions clear. The resulting allocation of resources among the agents is the solution to the original resource allocation problem produced by MOP.

Agent Design As in Agent-Oriented Programming, agents are constrained to have some state and specific functionality which allows them to interact with the environment. Internal state of the agents includes prices of goods and other information relevant to computing their utilities. Agents interact with the environment through bidding for goods at the auctions and receiving current price quotes from the auctions.

Applications of Market-Oriented Programming The original paper on Market-Oriented Programming applied this paradigm to solving a multi-commodity flow problem that involved moving cargo over a given transportation network. Goods in this problem model capacity on each origin-destination pair in the network, as well as generic transportation resources. After introducing special carrier agents that internalize the cost of transporting cargo over links in the network, MOP produced an minimum cost allocation.

In later years, Wellman et al. have solved several other problems using the MOP framework. Wellman [23] used it to solve distributed configuration design problems. In this example, goods modeled resources needed by design components, as well as performance attributes that measure the capabilities of the designed product. The work by Mullen and Wellman [9] simulated a computational market to decide when and where Network Information Services should be mirrored. The goods in this work are network and computing resources and mirrors. Yamaki, Wellman, and Ishida [26] used MOP to solve QoS allocation problem in the FreeWalk multimedia application. In this example, Bandwidth and QoS at different points in time served as goods for the computational economy.

Ygge et al. used Market-Oriented Programming to manage electric power load [28, 30, 27]. In this work, commodities represented power at different time slots. Agents, or HOMEBOTS as Ygge et al. referred to them, are expressions of customer preferences and as such bid at the auctions for the aforementioned commodities. Bidding ends when the market equilibrium is reached, and the agents are allocated the resources they had won.

5.4 Discussion of Market-Based Approaches

Having described the various market-based approaches to resource allocation, we can now revisit the question we hinted at earlier: why would we use or not use such an approach?

Let us address the pros first. One of the more common justifications for market-based approaches is that it is a convenient way to implement a decentralized approach. Given no other motivation, this argument is reasonable when a decentralized approach is indeed reasonable in that setting. That may be, as we had already suggested earlier, because information itself is distributed, and it is more costly to communicate it to a central processor than to compute solutions locally (possibly, compressing information into a more manageable form as a result). It may also be that a problem is, indeed, easily parallelizable, and so a decentralized approach offers great computational advantage. Finally, we could in some circumstances (but, as we had noted previously, not in general) distribute failure among multiple entities, increasing robustness of the system. But even if the argument for decentralization holds, we must still justify market-based approaches as advantageous over other possibilities that are specific to particular problem domains. An obvious advantage is the generality of the concept of market-based computation: it can be relatively naturally applied to a very wide array of resource allocation problems, as the examples in the previous section demonstrate. In addition to generality, under certain assumptions market-based approaches promise to give us an optimal solution, and so we do not hope to do better by using a technique specialized to a problem we are trying to solve.

As expected, there are also many downsides to using market-based approaches. First of all, decentralizing computation is often undesirable, whether it is because it reduces reliability of the system, increases communication costs, or simply makes it more

difficult to reason about the solution.[6] But even if one may want to decentralize solution to a given problem, market-based approaches introduce several significant hurdles that may discourage a researcher from applying them. For example, it is often not trivial to design a rational agent. Utility functions do not generally "stand out" from the problem description, and verifying that the derived utilities correctly represent the problem at hand may be somewhat laborious. In addition, as Wellman points out in [24], transforming a problem into a form of market economy is often non-trivial due to the difficulties involved in identifying precisely what parts of the problem the agents and goods will represent.

6 Conclusion

After such a lengthy discussion of the various approaches to resource allocation, we seem to be at an impasse. Yes, market-based approach may be good sometimes, bad other times, and it unclear what to do the rest of the time. Unfortunately, such is life, and as tempting as easy answers may be, they will often be misleading. But in conglomerating all this information, we have, hopefully, achieved an increased level of coherence. Clearly, there are many situations where market-based approaches are useful. Just as clearly, there are many where they are not, but through some preliminary analysis we can usually discern relatively quickly into which of these two bins a given problem is likely to fall. It may not be the easy answer we would have hoped for, but it is progress.

References

[1] John Q. Cheng and Michael P. Wellman. The walras algorithm: a convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

[2] Scott H. Clearwater, Rick Costanza, Mike Dixon, and Brian Schroeder. Saving energy using market-based control. In Scott Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, New Jersey, 1996.

[3] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[4] Ross A. Gagliano, Martin D. Fraser, and Mark E. Schaefer. Auction allocation of computing resources. *Communications of the ACM*, 38(6):88–102, June 1995.

[5] John Kenneth Galbraith. *The Age of Uncertainty*. Houghton Mifflin Company, 1977.

[6] Bernardo Huberman and Scott H. Clearwater. A multi-agent system for controlling building environments. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, pages 171–176, Menlo park, California, June 1995. AAAI Press / MIT Press.

[7] Thomas M. Kroeger and Darrell D. E. Long. Predicting file-system actions from prior events. In *Proceedings of the USENIX 1996 Annual Technical Conference*, pages 319–328, 1996.

[8] James F. Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, May 1989.

[9] Tracy Mullen and Michael Wellman. A simple computational market for network information services. In Victor Lesser, editor, *Proceedings of the First International Conference on Multiagent Systems.*, pages 283–189, Menlo park, California, June 1995. AAAI Press / MIT Press.

[10] R. Hugo Patterson, Garth A. Gibson, Eka Ginting, Daniel Stodolsky, and Jim Zelenka. Informed prefetching and caching. In Hai Jin, Toni Cortes, and Rajkumar Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, pages 224–244. IEEE Computer Society Press and Wiley, New York, NY, 2001.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[6] However, there have been numerous arguments that decentralizing computation may lead to emergent properties of the system, which, while virtually impossible to reason about, are nevertheless desirable.

[12] Tuomas W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 295–308, Hidden Valley, Pennsylvania, 1993.

[13] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[14] Abraham Silberschatz and Peter Galvin. *Operating System Concepts*. Joh Wiley & Sons, 5th edition, January 1998.

[15] Ken Steiglitz, Michael L. Honig, and Leonard M. Cohen. A computational market model based on individual action. In Scott Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, New Jersey, 1996.

[16] I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, June 1968.

[17] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[18] Hal R. Varian. *Intermediate Microeconomics: A Modern Approach*. W. W. Norton & Company, 4th edition, 1996.

[19] Vivekanand Vellanki and Ann Chervenak. A cost-benefit scheme for high performance predictive prefetching. In *Proceedings of SC99: High Performance Networking and Computing*, Portland, OR, 1999. ACM Press and IEEE Computer Society Press.

[20] Carl A. Waldspurger and William E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Operating Systems Design and Implementation*, pages 1–11, November 1994.

[21] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.

[22] Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[23] Michael P. Wellman. A computational market model for distributed configuration design. In *National Conference on Artificial Intelligence*, pages 401–407, 1994.

[24] Michael P. Wellman. Market-oriented programming: Some early lessons. In Scott H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, New Jersey, 1996.

[25] Wayne L. Winston. *Operations Research: Applications and Algorithms*. Duxbury Press, 3rd edition, 1994.

[26] Hirofumi Yamaki, Kyoto University, Michael P. Wellman, and Toru 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*, pages 385–392. AAAI, 1996.

[27] Fredrik Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Department of Computer Science, Lund University, 1998.

[28] Fredrik Ygge and Hans Akkermans. Power load management as a computational market. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi–Agent Systems*, pages 393–400. AAAI, 1996.

[29] Fredrik Ygge and Hans Akkermans. Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333, 1999.

[30] Fredrik Ygge, Rune Gustavsson, and Hans Akkermans. Homebots: Intelligent agents for decentralized load management. In *Proceedings of DA/DSM Europe 96*, Vienna, Austria, October 1996.