Economic Theory 13, 41–92 (1999)

Economic Theory

© Springer-Verlag 1999

Decentralized computation procurement and computational robustness in a smart market[★]

Paul J. Brewer

Department of Economics, Georgia State University, Atlanta, GA 30303, USA
(e-mail: pjb@gosolveit.com)

Received: November 5, 1996; revised version: September 30, 1997

Summary. Several 'smart market' mechanisms have recently appeared in the literature. These mechanisms combine a computer network that collects bids from agents with a central computer that selects a schedule of bids to fill based upon maximization of revenue or trading surplus. Potential problems exist when this optimization involves combinatorial difficulty sufficient to overwhelm the central computer. This paper explores the use of a computation procuring clock auction to induce human agents to approximate the solutions to discrete constrained optimization problems. Economic and computational properties of the auction are studied through a series of laboratory experiments. The experiments are designed around a potential application of the auction as a secondary institution that approximates the solution to difficult computational problems that occur within the primary 'smart market', and show that the auction is effective and robust in eliciting and processing suggestions for improved schedules.

JEL Classification Numbers: D44, D83, D82, C92.

Introduction

Advances in computer technology, networking technology, and the understanding of decentralized market forces – driven by both theory and experiment – create a unique opportunity for economists to apply market processes to allocation and decision problems in areas where markets were

[★] The author wishes to acknowledge the financial support of the California Institute of Technology Experimental Economics and Political Science Laboratory and the Georgia State University Experimental Economics Laboratory. Helpful comments were received from (in alphabetical order) 3 anonymous referees, Richard McKelvey, Scott Page, Charles R. Plott, David Porter, and Rubin Saposnik.

not previously practical. The ability of the internet to bring together distant agents for competition coupled with the application-specific market rules that can be programmed into a computerized system allows the construction of decentralized processes that before could exist only in theory. Such markets, called "smart markets"[1] because of their high-tech implementation, predate the recent growth of online commerce and started being proposed in the early 1980s. Although not all of the early proposals survived to implementation, the list of potential applications that has been suggested by the literature is quite broad and the experimental data has suggested that a potential for success does exist.

Proposed smart market applications include airport time slot allocation,[2] resource planning for space missions,[3] project management,[4] assignment of time slots for equipment usage,[5] markets for trading and transporting natural gas and electricity,[6] and the allocation of access to railroads.[7]

The economic philosophy that markets should be more efficient than administrative processes has many sources. Hayek (1945) suggested that a preference for markets should depend on the structure and nature of information. He argues that if important information is decentralized, then a decentralized economic system will be better at processing it than a system that tries to collect the information to a central location. Hurwicz (1972) attempted to formalize the idea of decentralized economic systems, and along with the contributions of numerous others a theory of mechanism design[8] has evolved in the literature.

Just as the early laboratory findings[9] of high efficiencies in classical market environments prompted researchers to ask whether markets could be extended to non-classical problems,[10] the smart market research creates new

[1] The term "smart market" for the description of these resource allocation processes seems to originate with McCabe, Rassenti, and Smith (1988).

[2] See Rassenti, Smith and Bulfin (1982) and Grether, Issac, and Plott (1989).

[3] See Banks, Ledyard, and Porter (1989) and Plott and Porter (1990, 1996).

[4] See Ledyard, Porter, and Rangel (forthcoming).

[5] See Olson and Porter (1994).

[6] See McCabe, Rassenti, and Smith (1987, 1989), and Rassenti, Reynolds, and Smith (1994).

[7] See Brewer and Plott (1996), Brewer (1995), and Nilsson (1991, 1993, 1994).

[8] For a survey, see Groves and Ledyard (1987). For a look at the basic concepts of the mechanism design approach, see Ledyard (1991, 1993).

[9] Smith (1962) reports on the efficiency of experimental markets in a number o f different supply and demand configurations. A brief selection of recent articles related to efficient information aggregation in laboratory markets would include Plott and Sunder (1982, 1988), Friedman (1984), Forsythe, Nelson, Neumann, and Wright (1992) and Cason and Friedman (1996). This selection covers a variety of economic environments in which information aggregation is important, and is by no means an exhaustive list.

[10] For instance, the airport slots, space mission, and railroad problems all share a common property in that non-convexity of preferences and indivisibility of resources are present. Also, see Plott (1994).

questions that require an answer. Two related questions are explored in this paper.

The first question involves what one should do when increases in scale of use combine with the complexities of certain mathematical problems to cause a runaway failure of a smart market design. The term "runaway" is used to describe a computer program that has not crashed per se. There is no bug or technical flaw. The problem being solved is simply too hard, and the computer takes longer to solve it than practical. Such a failure might be attributed to a lack of computing power, but computer scientists have identified a class of NP-complete optimization problems for which computing requirements grow so quickly that additional hardware technology is not usually the answer. Optimization of bid revenue or of trading surplus is a routine calculation in many smart market designs, and such designs can involve computation of NP-complete problems.

The second question poses an unusual response to the first question. Instead of using a *central* computer, can a secondary auction be used to create a *decentralized* computer that runs the primary market? Equations would be sent to the auction for solution, agents would compete to provide solutions, and solutions would be checked before rewarding the agents who submitted them. In this way, the computing bottleneck might be eliminated from the smart market design.

Ideally, the computation procuring auction could provide 3 specific benefits in correcting runaway smart markets. The first benefit is that the auction transfers computational difficulties onto a broader class of agents, who will compete to solve them. Competition among agents and solution techniques should yield lower computational costs than solution via a fixed algorithm. The second benefit is that the strategic private information of agents might be applied to the computational problem, rather than solving it centrally in the absence of this information. The third benefit is that a properly designed auction might force the benefits and costs of continued computation to be assessed, preventing the runaway failure problem that was possible with centralized computing.

Creating a decentralized computer that can solve any of the mathematical problems that a centralized electronic computer can solve is a challenging proposition. The primary goal of this research is not to solve this problem with a single study. Instead, this research initiates an exploration into existence. Can the idea be seen in operation? Can an auction procure at low cost, the intellectual good of solution or approximate solutions to optimizations? Laboratory experiments can help to provide the answers to these questions.

Limiting the types of problems that the decentralized computer must solve is crucial to both the success and the applicability of the research. The problem to be solved will involve an optimization problem from a smart auction called BICAP. The optimization involves maximizing bid revenue from a set of known bids relative to a known feasibility constraint. BICAP stands for Binary Conflict Ascending Price auction, and was an auction constructed for allocating access to railroads. The BICAP application was chosen for two reasons: it is easy to describe and understand,[11] and it can involve significant computational complexities at large scales.

The conclusion of this paper will be a demonstration, via experimental methods, of a particular auction mechanism created in this paper. This mechanism will be called the "Computation Procuring Clock Auction" or CPCA, and will be shown to be effective at solving or approximating the maximum bid revenue in the BICAP smart auction.

In these experiments, CPCA will be merged with BICAP. The BICAP rules will continue to apply to the bidding procedures for the trains, but CPCA will attempt to find the train schedule. The schedule is supposed to be the set of bids for trains that maximizes bid revenue and satisfies a complex set of feasibility constraints (basically, no collisions between the trains), but this requirement will be relaxed somewhat. CPCA will accept any feasible suggestion, but will enforce an improvement rule and reward agents according to the level of improvement in their scheduling suggestions.

The experiments are intended to provide a "proof of concept" or "proof of principle" and also provide data on the operation of the auction that can be used to argue design consistency. If the auction is observed to work, works repeatedly, and works for reasons that are consistent with economic theory, then we might initially trust the design to function consistently in similar applications until counterexamples begin to be revealed. In this way, the domain of applicability of the techniques can be discovered in a process of future theoretical and laboratory research.

The demonstration problem that CPCA solves is in itself a significant problem from both an applied and a theoretical perspective. The problem comes from a question about privatizing access to railroads in Sweden and whether the BICAP could be scaled up. The BICAP auction requires a "market maker" to calculate the set of winning bids from a complex set of safety and other contingency constraints. This calculation will be taken over by CPCA – functioning as an auction within an auction.[12] In a broader sense the issues about scaling up are common to many of application-specific, "smart" market mechanisms that have been appearing in the literature since the early 1980s.

The broader applied perspective is that the development of something like CPCA may broaden the applicability of "smart" market techniques by eliminating the requirement that the market maker choose a scale of computing resources to make available. Typically, a market maker must provide the computing power for the market to function. He must choose in advance what kind of computing facilities to provide. This choice becomes a short run constraint on computing resources that has consequences in terms of

[11] Rubinstein (1996) suggests that binary relations are common in natural language (e.g. to the left of, above, conflicts with) for exactly such utilitarian reasons.

[12] Thus, in the combined BICAP + CPCA process, an agent desiring to run a train submits a BICAP bid and competes with other bidders. An agent who can compute schedules submits scheduling suggestions to CPCA and will compete with other agents for cash bonuses that will be awarded under the CPCA process.

transaction costs and capacity[13] of the market to handle more complex situations. CPCA could improve the applicability of these new market techniques by decentralizing the operations of the market maker. CPCA could provide the market maker with access to a broader range of facilities for problem solving.

The broader theoretical perspective is that difficult computational burdens can be lightened by integrating strategic information, possessed by human agents, into attempted solutions of the problem. In a sense the design of the auction follows Hayek's [1945] lead, in that it insures that information about "the peculiar circumstances of time and place" can compete with expert technical knowledge. Modern ideas about heterogeneous computing agents may also play a significant role in understanding the operation of the process but are beyond the scope of this paper.

The use of the incentive system of an auction to create a kind of "computer" that, instead of using a fixed hardware facility or software algorithm, uses an auction to determine who solves what pieces of the problem, on what hardware, and how is an exciting new idea from a computational perspective as well. Such a "computer" should have some interesting properties when compared to the typical operation of an ordinary electronic computer. Systems of equations sent to the auction would be publicly solved by agents in the marketplace, who would be paid for their efforts at a price level determined by performance and competitive forces. Least cost agents or techniques would have an advantage in the competition, but other techniques would also be involved in the competition. This could create a type of robustness or backup, so that if one solution technique fails another could still succeed in providing a solution. In addition, such a "computer" could use information that might be unavailable in a traditional, centralized setting. When privately held strategic information is important to solving a problem, the auction could give incentives for this private information to be applied,[14] reducing solution cost or solution time. The laboratory experiments will make it possible to demonstrate these ideas in operation.

Creating an effective computation-procuring auction is challenging. There is a "lemons"[15] difficulty with more general approaches, a difficulty that can be overcome for the limited class of mathematical problems described in this paper. The lemons result involves the quality of goods and

[13] Thus, one view is to limit smart markets to economic environments that are likely to be solvable. Rothkopf, Pekec, and Harstad (1995) consider such an approach.

[14] Disclosure of the private information might otherwise never occur in these cases. The private information might remain completely with an agent, who might consider privacy valuable towards their strategic position.

[15] As shown theoretically by Akerlof – if the quality of goods is costly to the seller, valuable to the buyer, and unobservable to the buyer before the sale, least cost procurement can result in acquiring low quality goods. Exact "solutions" to mathematical equations differ in quality by being either "right" or "wrong", so an auction could result in an incorrect answer to the system of equations. Approximate solutions could be inaccurate.

suggests that quality of solutions submitted to the auction should be monitored. Monitoring for accuracy of solutions could involve considerable computational effort on the part of the buyer and possibly introduce additional complications in the transaction between the buyer and seller. Keeping solution quality high, using competition to lower costs, and avoiding large transactions or monitoring costs are the challenges that the any auction-based approach at computation must address.

Investigating the performance of CPCA will involve the creation of a new economic criteria necessary to describe CPCA as a computational process. This criteria – to be called "Computing Effectiveness" – is different from the social welfare criteria of efficiency of allocation, which is typically used to evaluate new kinds of market processes. Constraints on the kinds of data observable in the experiments motivates analysis in terms of this new criteria. The current series of experiments will only allow observation of the outcomes of CPCA and not the internal workings of the human agents involved in the computation. One can not know how different agents were thinking or what mental steps they took to solve problems and submit suggestions to the CPCA process. Because of this difficulty, there will be no way to insure that the human agents are computing in the most socially efficient way, and there are many reasons to believe that some inefficiencies will occur.[16]

The current research, as a proof-of-principle, is primarily concerned with whether a decentralized computing technique can be effective at all. Thus, the data analysis concentrates on particular aspects of CPCA's computational capabilities, such as accuracy of solutions and costs of incentives paid to agents, with the question of overall social efficiency left for future research.

The remainder of the paper will be organized as follows. Section 2 introduces notation and concepts regarding scheduling problems. Both scheduling in general and railroad scheduling will be addressed. General computing techniques for optimal scheduling will be discussed, as well as the implications for agents with limited computational abilities. The Computation Procuring Clock Auction [CPCA] mechanism is defined in section 3 and it is a simple variation of a well known clock auction mechanism. Section 4 considers how to interface CPCA to the BICAP rail auction. Issues are listed and an interface between the two processes – which will run simultaneously – is defined. An experimental testbed environment is defined in section 5. Human subject experiments are conducted using the economic environment from section 5. The results of these experiments are reported in section 6. The final section, section 7, contains concluding remarks.

2 Scheduling: Concepts and notation

Management of access to a common facility, such as railroad tracks, a spacecraft, or a laboratory typically involves feasibility constraints and some

[16] For example, there could be duplication of effort. CPCA does not attempt to coordinate agents to work on separate parts of a problem.

notion of value or priority. Because some uses can be carried on simultaneously while other uses can not, a manager interested in finding the best schedule for sharing the facility among the several uses and users faces a decision problem with a particular mathematical structure. This section provides an introduction to the mathematical notation for such scheduling problems.

2.1 General scheduling environments

A schedule S is a set of uses for a facility. Extensions of this notation where a schedule could be a function giving a set of uses at various times, a vector, a matrix, or some other mathematical object are possible but are beyond the scope of this work. It is useful to assume that the set of schedules is finite,[17] though perhaps large.

Let $\mathscr{S}$ be the set of all schedules, and let $\mathscr{F}$ indicate a subset of schedules that are feasible. Feasibility of a schedule is determined by a function $F(\ ) : \mathscr{S} \rightarrow \{0, 1\}$. This function contains all of the technical, application-specific constraints for determining feasibility of schedules. If $F(S) = 1$, then s is a feasible schedule and so $S \in \mathscr{F}$. If $F(S) = 0$, then S is not a feasible schedule and so $S \notin \mathscr{F}$.

Defining feasibility via a function[18] $F(\ )$ captures an essential difference between feasibility constraints and the feasible set that will be important later when considering computational costs and agents with limited abilities. In certain difficult optimization problems, it is easy to determine if a particular schedule is feasible but difficult to search for and enumerate the set of all feasible schedules. If computational abilities are quite costly then an agent can determine $\mathscr{F}$ from $F(S)$ and $\mathscr{S}$, but only at a prohibitively high cost.

The value of a schedule is given by a function $Q(\ ) : \mathscr{S} \rightarrow \Re^{+}$. $Q(S)$ is a positive real number giving the value[19] of a particular schedule S. The manager's problem in choosing the best schedule can be described via a triple $(\mathscr{S}, F, Q)$ which will be called a *constrained optimization environment.*

[17] Although optimization over a continuum involves interesting issues, a number of applied scheduling problems can be posed as optimizations over a [large] finite set. For example, in scheduling the uses of a facility over time, one could consider only 1 second, 1 minute, or 1 hour blocks of time as appropriate. If there are a finite number of possible uses at any time, and a finite number of times, then there are a finite number of schedules that can be chosen. The fineness of such scheduling will depend on details of a particular application. As the purpose of this section is primarily notational rather than theoretical, there is no reason for the reader to be burdened with the consideration of technicalities involved with infinite or dense sets.

[18] The use of this notation does not reduce generality. If one begins a problem knowing the feasible set of schedules $\mathscr{F}$, it is easy to construct a feasibility function $F(\ )$ that faithfully generates that feasible set. $F(S) = 1$ if S is in $\mathscr{F}$, $F(S) = 0$ if S is not in $\mathscr{F}$.

[19] In this ana lysis, $Q(\ )$ need not depend on whether S is feasible or not. For example, $Q(\ )$ might be an amount of money that could be collected if this schedule were implemented, and this amount might not depend formally on feasibility. In the analysis that follows, only feasible schedules can be selected. Values of $Q(\ )$ outside of the feasible set of schedules will not affect the schedule that is chosen.

Definition. *A constrained optimization environment $\mathscr{E}$ is a triple $(\mathscr{S}, F, Q)$, where*:

$\mathscr{S}$ *is a set of elements* $\{S_1, S_2, \ldots\}$ [*the solution space*]

$F(S)$: $\mathscr{S} \rightarrow \{1 = \text{“}\textit{feasible}\text{”}\}, 0 = \{\text{“}\textit{not feasible}\text{”}\}$ [*the constraint function*]

$Q(S)$: $\mathscr{S} \rightarrow \Re^+$ *is a positive*, *real*-*valued function of* S [*the criterion function*].

2.2 Example: Railroad scheduling

The initial tests of the techniques developed in this paper will involve a model scheduling problem involving access to railroad tracks. Auctions for rail scheduling will be seen to involve computation al issues described by the general framework defined above.

Definition. A *railroad scheduling environment* is specified by:

(a) a *physical configuration* $PC = (R, C)$, where
  R is a set of train routes
  $C \subset R \otimes R$ a set of train pairs that conflict; *AND*
(b) an *economic configuration* $EC = (I, V)$, where
  $I = \{1, 2, 3, \ldots, i, \ldots, I\}$ *a set of agents*
  V is a matrix of agent's private values for trains, with elements $V_i[r]$ defined for each $i \in I$ and $r \in R$.

The physical configuration of the railroad determines a portion of the constrained optimization environment $\mathscr{E} = (\mathscr{S}, F, Q)$, such that:

(i) $\mathscr{S}_{\mathrm{PC}} = 2^R = \{S : S \subseteq R\}$ = set of all subsets of R.
(ii) $F_{\mathrm{PC}}(S) = 1$ if and only if $[(S \otimes S) \cap C = \varnothing]$
  $F_{\mathrm{PC}}(S) = 0$ otherwise.

Two of the three elements of $\mathscr{E}$ are determined by the physical configuration alone. The remaining element, the criterion function $Q(\ )$, depends on the type of public information that is present. The information that is available depends on institutional features. Two examples, key to the later arguments of the paper, are considered below.

Efficient scheduling $\mathscr{E}_V$

The criterion for efficient scheduling is to attempt to maximize the value of agents' uses of the track. The values V of the trains to each operator must be known to the manager.[20] Any trains that are scheduled can then be assigned

[20] Whether the manager can obtain the necessary information about values, especially when the values are privately held confidential information, is often a key point of debate about whether a centralized or decentralized system of management is appropriate. If the values are known, then the efficient scheduling problem will involve the constrained optimization environment defined above.

to the agent with the highest value. For each $r \in R$ let $\mathrm{HV}[r] = \max_{i \in I} V_i[r]$ be the highest value for each train. Then in the general scheduling framework, $Q(S) = Q_v(S) = \sum_{r \in S} \mathrm{HV}[r]$, and

$$\mathscr{E}_v = (\mathscr{S}_{\mathrm{PC}}, F_{\mathrm{PC}}(\ ),\ Q_v(\ ))\ .$$

Auction-based scheduling $\mathscr{E}_B$

The criterion for auction-based scheduling is to attempt to maximize the total bid revenue for use of the track. The manager does not need to know the values V. Instead, under auction-based scheduling, agents bid for the right to run the various trains.[21] Suppose for each train route $r \in R$, $B[r]$ is the bid on that train. Then in the general scheduling framework, $Q(S) = Q_B(S) = \sum_{r \in S} B[r]$, and $\mathscr{E}_B = (\mathscr{S}_{\mathrm{PC}}, F_{\mathrm{PC}}(\ ),\ Q_B(\ )\ )$.

This paper will concentrate primarily on the computational challenges inherent in auction-based scheduling rather than on a comparison of auction-based scheduling and efficient scheduling. This distinction will become important in the design of the experiments and the techniques for data analysis.

2.3 Computational perspectives

Attempting to maximize the criteria $Q(S)$ over the feasible schedules could be done in many ways. The best way of mathematically or computationally setting up a particular problem involves many different, possibly conflicting issues. For example, if a computing machine is to be used, there are tradeoffs involved between programmer time and machine time in choosing an elegant versus a brute force style of solution. Details on how a problem is set up certainly affect the likelihood of obtaining a solution as well as its accuracy. These details become important when computational capabilities are severely limited or extremely costly.

An instance of a computational problem is an environment $\mathscr{E}$ together with conditions a solution must satisfy. Thus, a problem has a formal definition. The computational perspective that a particular problem imposes is not rigorously defined, but involves intuitions about tradeoffs between the kinds of solutions likely to be obtained in practice and under harsh conditions. Two types of problems are defined below.

Definition. An *instance of OPTIMIZATION* ($\mathscr{E}$) is the problem: find a solution $S^* \in \mathscr{S}$ such that

(a) $F(S^*) = 1$, and

[21] In theory, the agents with high values will outbid the agents with low values. Whether that in fact occurs is a question for empirical field and laboratory research. In any case, the computational problem for the scheduling authority is to maximize the bid revenue over the feasible schedules.

(b) $Q(S^*) \geq Q(S)$ for *every* $S \in \mathscr{F}$
or return the null solution $\varnothing$.

Definition. An *instance of IMPROVEMENT* $(\mathscr{E}, S)$ is the problem: find a solution $S^* \in \mathscr{S}$ such that
(a) $F(S^*) = 1$, and
(b) $Q(S^*) > F(S)Q(S)$
or return the null solution $\varnothing$.

The problems of IMPROVEMENT and OPTIMIZATION are related in a straightforward way. If a non-null solution to OPTIMIZATION $(\mathscr{E})$ exists, let this solution be S^*. Then the solution to IMPROVEMENT $(\mathscr{E}, S^*)$ must be $\varnothing$ because no feasible schedule can have a higher $Q(\,)$ value than S^*. This suggests that an algorithm that repeats the improvement procedure, improving the schedule until a feasible schedule with a higher value can not be found, could yield solutions for OPTIMIZATION.

Definition. An instance of the *ITERATIVE-IMPROVEMENT* $(\mathscr{E}, S_0)$ algorithm involves finding a solution $S^* \in \mathscr{S}$ via the following procedure:

```
for $k = 1, 2, 3, \ldots$
        repeat $S_k$ = solution to IMPROVEMENT $(\mathscr{E}, S_{k-1})$
until $S_k = \varnothing$
end with the solution $S^* = S_{k-1}$
```

When agents have infinite computational abilities, the ITERATIVE-IMPROVEMENT algorithm will solve the OPTIMIZATION problem. An improvement solution exists if and only if the optimum has not been reached. ITERATIVE-IMPROVEMENT can only stop if the optimum schedule is calculated, and thus, its solution is equivalent to OPTIMIZATION.

2.4 Limited rationality

Limits on agents' computational abilities create a distinction between the problems OPTIMIZATION, IMPROVEMENT and ITERATIVE-IMPROVEMENT. IMPROVEMENT involves evaluating the functions F and Q until a suitable schedule is found or until the agent "gives up". OPTIMIZATION could require calculating the entire feasible set $\mathscr{F}$, and this increased difficulty increases likelihood that an agent will "give up".

Suppose that either OPTIMIZATION or IMPROVEMENT can fail by returning a solution of $\varnothing$ when in fact some solution does exist. This provides agents involved in computing a way out of a prohibitively costly situation, and it provides mechanism designers a way to model the failure of computations.

If a difficult computation is structured as OPTIMIZATION, the final result may be $\varnothing$. In scheduling resources, a null schedule may be feasible but not particularly desirable. A more useful result is likely if the computation is structured as ITERATIVE-IMPROVEMENT, because as long as some

IMPROVEMENT solution is found, a non-null schedule solution will be returned. The return from ITERATIVE-IMPROVEMENT may or may not be optimal, but it is more likely to be higher in $Q(\,)$ value than a solution of $\varnothing$. Furthermore, while a null solution to OPTIMIZATION is not particular useful for further attempts at solution, the schedule solution to ITERATIVE-IMPROVEMENT can be used to start another instance of the algorithm,[22] with no loss of the partial solution information.

3 The computation procuring clock auction [CPCA]

3.1 Overview

The CPCA mechanism involves a modification of the ITERATIVE-IMPROVEMENT algorithm of the previous section. The primary modification is to invite the participation of many different agents by providing payment incentives similar to those in a clock auction. Thus, the name of the mechanism – the Computation Procuring Clock Auction. This section is concerned with defining and explaining the formal rules of this auction.

CPCA can be summarized as follows. At the beginning of the auction there is some status quo feasible schedule. Agents who can find and submit an improved solution are paid a bonus equal to some percentage of the improvement. While this is happening, a clock counts down from some initial value. This clock indicates the time remaining in the auction if no improvements are submitted. As the clock counts down, the percentage of the surplus offered to any agent submitting an improvement increases. The percentage of the surplus offered to agents approaches 100% as the time remaining approaches zero.

If an improved solution is submitted, the clock is reset and more improvements are sought. If no improved solution is found, then the procedure ends with a recommendation to implement the best solution found so far. The idea is that if no agent is willing to provide an improvement, even when they are given all the surplus from the improvement, then further improvement in the solution is not practical, at least not in a decentralized framework with the given agents.

Some CPCA parameters, such as the choice of the time scale and the way that the bonus increases as time ticks down, might be adjusted in order to "fine tune" CPCA to a specific need. At the current time, there is no well organized theory about how this might be done. In choosing a means for approximating maxima, one typically wants to minimize solution time, and solution cost, but maximize solution accuracy, all else being equal. But how

[22] In his survey of bounded rationality, Conlisk (1996) mentions that iterative models are in use in the theory of bounded rationality. In the bounded rationality literature, ideas similar to iterative improvement are – under other names – offered as a behavioral theory of human activity. Thus the ideas presented here are not totally unfamiliar to the economic literature. When the manager does not have perfect rationality, it is reasonable for him to attempt an iterative improvement calculation rather than a full optimization.

to trade off these qualities against each other is not obvious. Different application-specific needs might dictate different tradeoffs one could make in solving specific instances. The goal here was to find a common mechanism or adaptation of a common mechanism that would respond to all three of these dimensions of the computing problem.

3.2 Formal description: states, messages, and transitions

In this section, the rules of CPCA are described using an event-driven approach that should be familiar to those who study dynamical systems or state-machines. In this approach, the mechanism has a *state* that reacts to *events*. *Events* are outside stimuli, such as messages from agents or the passage of time, that cause the state of the mechanism to change. *Transition rules* indicate how events affect the states of the mechanism.

States have importance in this model of CPCA for two reasons. (1) The *current state* of the mechanism is *public information and common knowledge*. The state is available to all participants and everyone knows that the other agents are seeing this information. (2) Given the current state in formation and the transition rules, agents can determine the immediate effect that their messages can have upon CPCA states and outcomes, as well as what will occur if agents take no further action.

CPCA is described by 4 *classes of states*: *null* $\emptyset$, *listening* L, *verification* V, and *closing* C. The role of each of these classes will be made clear shortly. A particular state is defined by a letter for the class ($\emptyset$, L, V, or C) together with the values of certain parameters that are relevant for that state. These parameters are put in braces after the letter, e.g. $L\{Q, F, S, \beta\}$ denotes a listening state with parameters Q, F, S and β. The following paragraphs will define further the various classes of states and their parameters.

Classes of states

The null state $\emptyset$ denotes a pre-existing state of the world before the mechanism is placed into operation. In the null state, there is no public information about the mechanism's operation. "Starting" the mechanism involves a transition away from the null state to a state where public information exists.

"$L\{\mathscr{E}; S, \beta\}$" denotes a listening state where $\mathscr{E}$ is a constrained optimization environment, S is the best solution so far, and β is the current bonus rate.

"$V\{\mathscr{E}; S, \beta, S^*, i\}$" denotes a verification state. The environment $\mathscr{E}$ and the variables S, and β have the same interpretation as in the listen states described above. In this state, agent i has suggested that S^* is better than the solution S. That is, the agent claims that $Q(S^*)$ exceeds $Q(S)$. The mechanism remains in the verification state until this claim is verified or rejected.

$C\{\mathscr{E}; S_{\text{FINAL}}\}$ denotes a closing state. The closing state parameters are the environmental parameters and the final solution S_{FINAL} obtained by the CPCA mechanism. The closing state is a terminal state. It is a final outcome of the mechanism.

Events

When events involve a particular agent, the notation for the event will involve an arrow. For example, the notation "*actor* $\rightarrow$ *event* 1" shows that a particular actor caused event 1. There are 4 *events* that affect the current state of CPCA. Two of these events are messages from the market maker. The event mm $\rightarrow E(\mathscr{E}; S_0)$ indicates that the market maker made public the details of the improvement problem to be solved. The market maker must also send a message to verify or reject suggestions that are submitted by the agents. The other two events are suggestion messages from agents for improved solution, and the passing of time with no such suggestions. Details of these messages are shown in Appendix A.

Transition rules

The transition rules for the various states and events are illustrated in Figure 1 and more formally defined in Appendix A. The figure is complete in the sense that it shows the features of the rules necessary to the operation of CPCA. The notation of the figure can be read as follows. States are given by circles. Dashed lines indicate potential steps of how CPCA responds to events that might or might not occur. Solid lines indicate procedural steps that must occur. Boxes give details of procedures that are required in the transition from one state to another.

Figure 1 can be interpreted as follows. The mechanism initially exists in a null state $\varnothing$, where nothing is happening and agents have no information. An agent called the "market maker", who is responsible for the maintenance of CPCA, starts the mechanism by sending the message "mm $\rightarrow E(\mathscr{E}, S_0)$" [dashed line]. This message specifies the constrained-optimization environment $\mathscr{E}$ (the functional form $Q(S)$ of the criterion to be maximized, and the function form $F(S)$ of the feasibility constraint) along with the an initial feasible schedule S_0. The CPCA mechanism is then reset [solid box] so that the information on the CPCA public display is then reset to indicate this information. A particularly important variable, the bonus rate β, is set to 0%. This bonus rate determines the rate of payment to agents who submit suggestions to the message. The mechanism then enters a Listen state where it waits for one of two events to occur. Either (1) some agent, say agent k, sends in a suggestion message for an improved solution to the constrained-optimization environment or (2) time passes without any agent sending in such a message. These two possibilities are considered separately below.

If time passes without a suggestion message from an agent, then the indicated path is followed. If agents do not provide suggestions, the bonus rate β steadily increases from 0% to 1%, 2%, 3%, up to 100%. When β reaches 100%, the mechanism closes The bonus rate β is rising in a familiar way, similar to the increase in price level in a clocked auction. As will be seen in the next paragraph, β is such a price. β gives the proportion of the improvements surplus to be awarded to the agent submitting the suggestion.

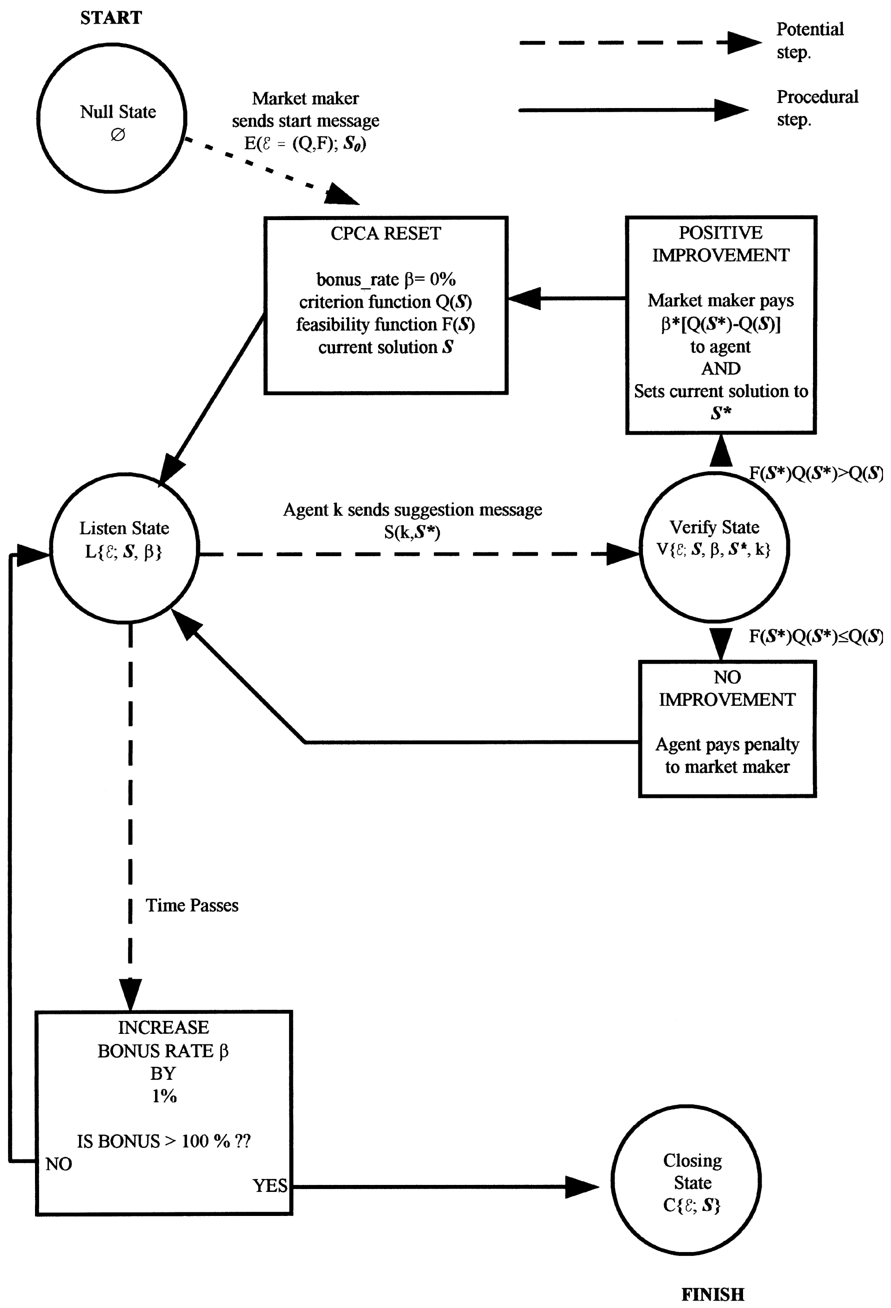


Figure 1. CPCA flowchart

As β increases, an agent might know or find a solution to the problem $IMPROVEMENT(\mathscr{E}, S)$ of finding an improved schedule that is better than the current schedule S. If agent k believes that the schedule S^* is such a solution, then agent k has the option, but not the requirement, to send a suggestion message $k \rightarrow S(k, S^*)$. Such a message causes the mechanism to leave the listen state and enter a verification state $V\{\,\}$ where the suggestion (k, S^*) is stored until it can be verified. The market maker then verifies whether the suggestion is an improvement $[F(S^*)Q(S^*) > Q(S)]$ or not. If the suggestion is an improvement, then a counterclockwise path is followed from the verify state back to the listen state. First, the box "POSITIVE IMPROVEMENT" is entered, the market maker pays the agent the bonus rate times the amount of the improvement, sets the current solution to S^* and then sets the bonus rate back to β zero. The mechanism once again enters a Listen state. If the suggestion is not an improvement, then the clockwise path is followed. The agent pays a penalty to the market maker, and then the mechanism goes back to a listen state.

If no agent sends in an improvement, the bonus rate β climbs to 100% and then the mechanism enters the (terminal) closed state. Stated another way, if no agent can provide an improved solution when offered the entire improvement $[\beta = 100\%]$, then the process is terminated. The intuition behind this termination rule is that the system should stop when costs of further improvements exceed the benefit in the improved solution.

3.3 Theoretical properties of the process

The purpose of this section is to present some rough ideas about properties CPCA might be expected to possess. Because there is no fully worked out theory about the behavior of such complicated mechanisms in such complex environments, what follows must necessarily be only a beginning in evaluating the properties of the process.

These properties are logical properties that follow from the rules of the CPCA process. Theories of agent behavior in a particular CPCA application will be addressed later in section 4.

CPCA creates a competitive game. Given several agents willing to submit suggestion messages, the structure of rewards in the CPCA mechanism creates a competitive game between these agents. There are several dimensions to this competition. Among agents who can find and submit similar suggestions, only the first agent to submit the suggestion will obtain the bonus payment. Similarly, agents who submit suggestions that involve greater amounts of improvement may tend to receive larger payments than agents who are only able to find smaller improvements.

CPCA provides incentives to lower costs. Because the CPCA mechanism involves resetting the bonus rate to 0% and increasing it gradually for each new improvement, agents who have the lowest costs for submitting an improvement will be the first to be offered a sufficient incentive to submit a suggestion. Thus agents who can perform low-cost searches or agents who already have information are favored by CPCA over those who have high costs of participation.

CPCA terminates in finite time. The CPCA mechanism always terminates when $Q(\,)$ is bounded from above and agents' costs of finding improvements are positive and bounded from below. CPCA can terminate with the initial solution S_0 when the constrained optimization environment is too difficult for the agents to solve it.

CPCA has monotone accuracy. The CPCA mechanism encourages more accurate solutions over less accurate solutions. In a scheduling environment $\mathcal{E}, S^{**}$ is more accurate than S^* if $F(S^{**})Q(S^{**}) > F(S^{**})Q(S^*)$. If the solutions S^{**} and S^* are submitted to CPCA, then no matter the order of submission, CPCA will result in S^{**} and never S^*.

CPCA is outcome-oriented. The CPCA mechanism rewards agents for information no matter the source. Participation in and rewards from the mechanism are not related to the level of effort that went into creating the solution, or whether the solution is a result of existing knowledge, new knowledge, private or public information. All that matters is whether the suggested solution S^* has $F(S^*)Q(S^*)$ greater than the value $Q(S)$ of the best schedule S that the process had found.

CPCA relies upon voluntary participation. With the exception of the market-maker, agents under CPCA are not required to participate in any way. Agents will voluntarily participate in order to affect the schedule or to receive bonus payments.

CPCA is incentive compatible. Since an agent submitting a suggestion can be paid as much as 100% of the improvement, if an agent has or can obtain a suggestion that improves the value of the schedule by more than the cost to the agent, then eventually CPCA provides a sufficiently high incentive for the agent to submit the information. Similarly, if an agent has or can obtain a suggestion, but only at a cost higher than the benefit, then CPCA does not provide a sufficient incentive to obtain or submit the information. Efficiency of CPCA is compatible with individual incentives and the incentives created by the process.

Under CPCA, information cascades may lead to competitive behavior. The onset of competition might emerge as a kind of cascade effect, under the following argument. If agent *j* knows of a valid CPCA improvement and believes with certainty that others will find and submit the improvement, then agent *j* should submit the improvement before another agent submits it. Only in that way does agent *j* collect the bonus. If competition is fierce, then immediate submission might be warranted. Agents will believe competition is fierce if a number of agents submit often and appear to submit immediately. Thus, a common belief in fierce competition produces incentives under which fierce competition would occur.

4 Applying the CPCA to the BICAP railroad auction

The CPCA procedure will be merged with a bidding procedure known as BICAP.[23] In BICAP, agents bid for access to railroad tracks and are free to increase their bids in an attempt to compete for the available rights. A central computer continuously calculates potential allocations based on maximizing the total bid from trains relative to some feasibility constraints. This information is reported as feedback to the agents. Agents can, if they wish, increase their bids, causing a new allocation to be calculated. The auction ends when a certain time period elapses with no further increases in the bids. When the auction ends, the potential allocation becomes the actual allocation of track rights, and the agents receiving an allocation must pay their bid(s).

Replacing the electronic central computer in BICAP with CPCA will require careful consideration about how these two auction processes interact. Ideally, CPCA should be used merely to replace the central computer in BICAP. CPCA should, ideally, not affect the incentives to bid generated in BICAP. Attempts to achieve a minimal interaction will define the interface between BICAP and CPCA. Data obtained in the laboratory experiments that follow will provide an opportunity to examine both CPCAs computing performance and its effect on BICAPs economic properties.

4.1 Issues

Level of feedback to agents. BICAP with a central computer provided feedback on the potential scheduling immediately after each bid. BICAP + CPCA, by depending upon others to calculate schedules, may take much longer, provide less accurate schedules that are subject to change, and may provide a much lower level of feedback concerning the effect of recent bids on the potential schedule that might be implemented.

[23] Recall that BICAP stands for Binary Conflict Ascending Price, and is an auction for allocating access to railroad tracks that was described in Brewer and Plott (1996).

Minimal BICAP/CPCA interaction. Even slight changes in the rules of auction procedures have been known to cause dramatic and debilitating results.[24] Only by avoiding any obvious pitfalls will the resulting BICAP + CPCA mechanism be suitable for testing via laboratory procedures.

Elimination of arbitrage. One obvious pitfall involves the following scenario. If the bonus rate β is non-zero immediately after a bid, the bidder might be able to submit a suggestion message and reclaim a portion of the bid. Suppose, for example, the bids on A and B are 300 and 299 respectively, and that the schedule is $\{A\}$, and that the bonus rate β is 50%. A bidder who bids 400 on B could try to submit the suggestion $\{B\}$. The CPCA bonus to this bidder would be $50\%^*(400 - 300) = 50\%^*100 = 50$. This payment of 50 should be seen as an arbitrage opportunity. Such an arbitrage opportunity, although it provides information to CPCA, distorts bidding in BICAP. Agents might bid much more than their value for trains, because they anticipate recapturing a portion of bids.

Simultaneous or sequential operation. CPCA scheduling could operate only after BICAP bidding has concluded, or the two mechanisms could operate simultaneously. Simultaneous operation involves greater challenges to the incentive structures of both processes, because it involves additional interactions that can not occur if the mechanisms are operated in sequence.

4.2 Interface

Simultaneous operation of BICAP and CPCA was chosen, with the CPCA mechanism reset immediately after every BICAP bid. One interpretation of such a rule is that every time the bid changes, the scheduling environment $\mathscr{E}_B$ has changed through the change in the function $Q_B(S)$, which is a sum depending directly on the bids in B. In resetting CPCA, the current best schedule is retained. The function $Q(\,)$ is changed to reflect the new bids.

The most important effect of this rule, which reduces the possibility of arbitrage, is that the bonus rate β is reset to 0%. Thus, submitting a CPCA suggestion immediately after a BICAP bid results in CPCA processing the suggestion, but awarding a zero bonus.

This method of reducing the BICAP + CPCA arbitrage relies on a degree of competition in the CPCA process. It is possible that an agent might submit a BICAP bid, know how it changes the bid maximizing schedule, wait a time and then submit the CPCA suggestion message. In such a case an arbitrage is still possible, but it is limited by the competition among agents to

[24] For an example, see Cason and Plott (1996) regarding an EPA pollution permit market.

submit suggestions. The amounts received in this way are likely to be small, and will not drastically affect the incentives to bid under BICAP.

4.3 Summary of BICAP + CPCA

Operationally, the BICAP + CPCA mechanism can be summarized as follows. The mechanism starts with a null potential allocation $S^* = \emptyset$, no bids, and a clock showing $\tau = \tau_0$ seconds remaining in the auction unless some agent sends in a bid. Certain BICAP and CPCA information is publicly available throughout the process – public knowledge includes the physical configuration of the tracks $\mathrm{PC} = (R, C)$ (which is sufficient to determine the feasibility function $F_{\mathrm{PC}}(S)$), and the high bids $\underline{B}$ (which are sufficient to determine $Q_{\underline{B}}(S)$). As time elapses the clock τ counts down toward zero and the CPCA bonus rate β increases. If someone submits a bid on a train route that is higher than current high bid for that route, $\underline{B}$ is altered. Any change in $\underline{B}$ requires resetting CPCA, because the criterion function $Q_{\underline{B}}(S)$ has changed. Therefore, the clock is reset to $\tau = \tau_0$ seconds and the bonus rate β is reset to 0%. If someone submits a CPCA suggestion S^* such that $F(S^*)Q_{\underline{B}}(S^*) > F(S^*)Q_{\underline{B}}(S)$, then the potential allocation is changed to S^{**}, the agent is paid their CPCA bonus $\beta[Q_{\underline{B}}(S^*) - Q_{\underline{B}}(S)]$, the clock is reset to $\tau = \tau_0$ seconds and the bonus rate is reset to $\beta = 0\%$. At some point, agents stop submitting bids and suggestion messages, the clock τ counts down to $\tau = 0$ (and β reaches 100%) and the auction terminates. Upon termination, the potential allocation becomes the final allocation. Agents then pay their bids in $\underline{B}$ on the allocated trains in S^* and receive the right to run their respective train(s).

4.4 Questions for empirical study

The purpose of this section is twofold: (i) to define criteria that will be used to measure the performance of the BICAP + CPCA mechanism in the experiments that follow, and (ii) to give a few rough theories about the behavior of agents under CPCA that might help to explain its failure or success as a tool for decentralized computing.

Is CPCA effective?

Here we are interested in knowing whether CPCA is effective at producing the same bid revenue maximizing schedules that a computer can. That is, if the final BICAP bids are given by B and the final CPCA schedule is S, does S produce the maximum bid revenue from the given bids and physical configuration of the trains? Put another way, does S solve OPTIMIZE($\mathscr{E}_B$) [where $\mathscr{E}_B$ defined in section 2.2]?

Definition. The Computing Effectiveness of CPCA is the ratio $\frac{Q(S^{\text{CPCA}})}{Q(S_B^{\text{OPT}})}$ where S the CPCA schedule, and S_B^{OPT} is a non-null solution of OPTIMIZATION $(\mathscr{E}_B)$.

Example. Suppose there are only three trains A, B, C, and that every pair of trains has a conflict. Then the set of feasible schedules is simply $\mathscr{F} = \{\{A\}, \{B\}, \{C\}, \{\emptyset\}\}$. Suppose that the bids B are 100 for A, 400 for B, and 500 for C, and that the CPCA schedule is $S = \{B\}$. The solution to OPTIMIZATION$(\mathscr{E}_B)$ is the schedule with the highest bid revenue, which is $\{C\}$. So $S_B^{\text{OPT}} = \{C\}$. The computing effectiveness ratio is then $Q(\{B\})/(Q\{C\}) = 400/500 = 80\%$.

Much of previous research uses a different ratio called "allocation efficiency" that involves comparing the sum of values $\underline{V}$ generated in auction-based scheduling experiments with the ideal, efficient scheduling of section 2.2. That comparison is not the primary goal here. Computing effectiveness is primarily concerned with the computational challenges inherent in constrained maximization of the sum of bids within auction-based scheduling. The computing effectiveness ratio defined here measures the extent to which CPCA has met this computational challenge.

Does CPCA work without destroying the bidding incentives in BICAP?

The key question behind this idea is whether the outcome of an auction mechanism depends on the means of computing. The following conjecture hypothesizes that this is not the case.

The CPCA computational invariance conjecture. If CPCA is computationally effective then CPCA will not affect closing bid prices in a combinatorial auction, so long as the bonus for submitting a proposal immediately after a bid is zero.

The conjecture involves a view that CPCA is solely a tool for computing the bid maximizing feasible schedule. As long as the computation is effective, the final bid prices should be dependent on the railroad environmental parameters (PC, EC) rather than on the method by which computing takes place.

The bid revenue and total surplus of BICAP + CPCA could be compared to these statistics for BICAP with central computing. This would necessitate some experiments that duplicate the Brewer and Plott (1996) experimental environments where BICAP was previously tested.

The relevant statistics are:

Bid-Revenue BR $= \sum_{s \in S^*} B[s]$

Total Surplus TS = Train Operators' Profit + Bid Revenue

$$= \left\{ \sum_{s \in S*} V_{I(s)}[s] - B[s] \right\} + \sum_{s \in S^*} B[s]$$
$$= \sum_{S \in S^*} V_{I(s)}[s]$$

Are CPCA costs competitive or monopolistic?

Under perfect competition, CPCA costs should be low for simple problems. The exact level of computational costs will be determined by the competitive supply of suggestion information.

Under monopoly, only one agent has the relevant CPCA suggestion information. This agent can wait until the bonus rate is 100% before submitting the information. Thus, under monopoly, one expects that the CPCA costs will be quite high and perhaps as much as 100% of BICAP bid revenue.

5 Experimental procedures

This section describes the design and conduct of experiments to examine bidding and computing behavior of BICAP + CPCA. The parameters of these experimental environments will be related, in a straightforward way, to an environment used for initial research on the BICAP auction. In this way some comparison of economic variables between the earlier research and this research will be possible – it will be possible to examine whether taking BICAPs computational chores out of the computer code and giving them to the agents through CPCA caused any unusual effects in the combined BICAP + CPCA mechanism. In addition, it will be possible to assess the performance of the combined BICAP + CPCA mechanism.

5.1 General

A total of six experimental sessions were conducted. Table 1 summarizes the environments studied in each experiment.

Table 1. Summary of experiments performed with three-track testbeds

Environment	Experiment	Proposers P	Periods	Subjects
Separable tracks	SIP1	3 independent agents	3	13 Caltech student
	S1	All buyers	3	10 Caltech students
	S2	All buyers	1[a]	10 Georgia State business students
Combined tracks	C1	All buyers	3	10 Caltech students
	C2	All buyers	2[a]	10 Caltech students
	C3	All buyers	1[a]	10 Georgia State business students

[a] Time did not allow running additional periods

As can be seen from the table, four of the sessions were conducted at Caltech and two of the sessions were conducted at Georgia State University. Subjects were undergraduate and graduate students, and were recruited via flyers and a computer network announcement.

Sessions proceeded as follows. Subjects were seated at a computer terminal. The experiment instructions (see Appendix C for a sample – there are slight differences across experiments) and train value sheets were passed out. The instructions were read, and there was a question and answer period. The subjects spent 5 to 10 minutes practicing with the experiment software, and then there was another question and answer period. Depending on time, 1 to 3 periods were conducted for actual cash payments. Periods ended when no agent bid for 30 seconds, so there was no fixed ending time for a period. In practice, periods took from 20 to 30 minutes and sometimes a bit longer. At the end of the experiment, subjects were paid their earnings. If a subject made an error entering a bid, they were urged to notify the experimenter so that the error could be corrected. A number of such errors were corrected.

Cash payment included a show up fee of from \$2 to \$5, plus their profits from the BICAP + CPCA mechanism. Subjects received a row of train values from a table like Table 2, and did not know anything about other subjects, values or the probability distribution. A subject's BICAP profit (or loss) was the difference between train value and final bid for each scheduled train for which they were the high bidder Their CPCA profit depended upon participation and submission of train system schedules according to the rules of section 3.2. Individual subject payments tend to fall in a broad range of \$5 to \$40 for a 1–3 hour session.

Subjects who lost money in a period had these losses deducted from earnings in profitable periods, and from the show up fee if necessary. A few subjects could not earn very much profit from equilibrium BICAP activity, and so their earnings were mostly from the show up fee and CPCA suggestions. Although no attempt was made to vary redemption value rates across subjects, a two-tiered, progressive franc to dollar ratio was used for experiments SIP1, $S1$, $C1$, and $C2$. Subjects earned 10 cents per franc for the first 20 francs, and then 2 cents per franc after that.[25] A flat 3 cent franc to dollar ratio was used for rewarding subjects in experiments $S2$ and $C3$.

Each of the four sessions lasted approximately 2–2.5 hours and used 10 subjects (except experiment SIP1, 13 subjects). Some sessions did not proceed as quickly as others, thus different amounts of data were collected: some sessions are 1 period, some 2 periods, and some are 3 periods.

5.2 The railroad testbeds: Physical configuration parameters

Recall from section 2 that a railroad scheduling environment consists of a physical configuration and an economic configuration. The physical config-

[25] Although it is possible that high initial franc values might lead 'poor' agents to compete more aggressively in CPCA and depress CPCA costs, such an effect was not noted.

uration determines the sets of trains that are feasible. A feasible set of trains can have simultaneous access to the railroad tracks without causing a conflict or collision.

Figure 2 shows the physical configuration of the tracks and the conflicts between the trains in the testbed. In the figure, each row defines a particular testbed that will be used in experiments. The row shows a name for the testbed along with a physical layout of tracks in this model railroad, followed by a diagram of the conflicts in this railroad. The diagram represents a conflict in C by a direct link between the two conflicting trains. For example,

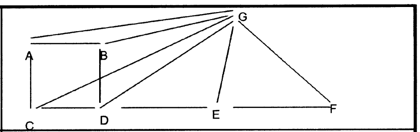
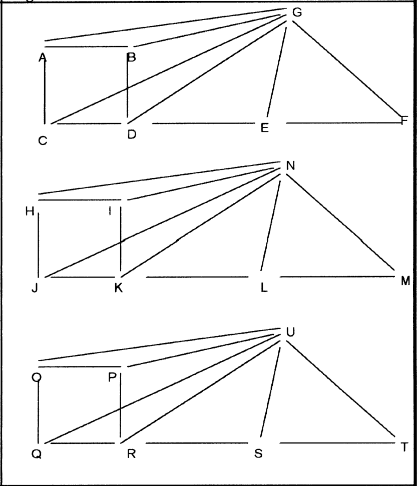
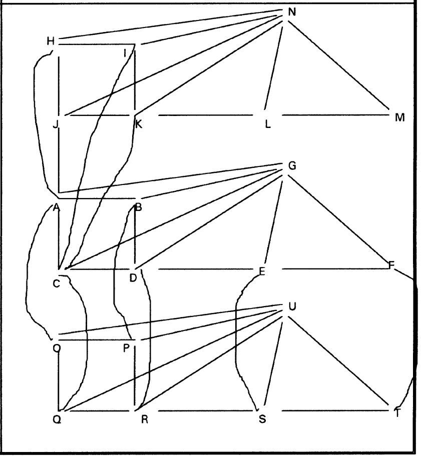
Environment "BP96" 1 Track, 7 Trains. Layout: ========== Track 1, Trains A-G	
Environment "S"/"SIP" 3 Separable Tracks, 21 Trains. Layout: ========== Track 1, Trains A-G ========== Track 2, Trains H-N ========== Track 3, Trains O-U	 A B C D E F G H I J K L M N O P Q R S T U
Environment "C" 3 Non-Separable Tracks, 21 Trains Layout: ========== Track 2, Trains H-N Track 1, Trains A-G ========== Track 3, Trains O-U	 H I J K L M N A B C D E F G O P Q R S T U

Figure 2. Testbed railroads for the experiments

in the single track "BP96" testbed, there is a conflict between trains $\{A, B\}$ and trains $\{D, E\}$ but not between trains $\{D, F\}$.

Environment "BP96" is a portion[26] of the environment used for the experiments reported in Brewer and Plott (1996). It consists of one track with trains $R = \{A, \ldots, G\}$. The layout shows a single track between two points. The conflicts are shown graphically in the box to the right. The *conflict graph* directly connects pairs of trains that are in conflict. Thus, trains A and B are in conflict because a line connects train A and train B in the conflict graph. But trains A and D are not in conflict because there is no line directly connecting A and D.

Environments "S" and "SIP" involve three independent tracks, where each track has a copy of the 7 trains in the BP96 environment. Because the tracks are not physically connected, the scheduling of one track does not affect the scheduling of the other tracks. Independence of the tracks is shown in both the physical diagram for the tracks and in the train conflict diagram.

Environment "C" involves a harder problem. Each of three tracks still contain the conflicts of environment S, but there are additional conflicts because the tracks now cross. The additional conflicts in the conflict diagram show that the scheduling of one track is now dependent on the scheduling of the other tracks. For example, if trains $\{A, D, F\}$ are scheduled on track 1, then the trains $\{O, R, T\}$ can not be scheduled on track 2 and train H can not be scheduled on track 3.

The difficulty of computing optimal schedules in these environments is related in a straightforward way. Complexity theory says that this type of scheduling problem is NP-complete, meaning that solution time can (but need-not) scale exponentially with the number of conflicting trains. This suggests that environment "C" is more complex than "S" which is similar in complexity to "BP96". Page (1996) takes an alternative view and gives two measures of computational difficulty, one of which is important for the train problem. The measure of *cover size* defines difficulty relative to the hardest separable sub-problem that must be solved. This type of difficulty is most important to parallel processes like CPCA. Each physical configuration will now be examined briefly in terms of the difficulty of the associated scheduling problem.

When each train has a positive additive value, the BP96 environment contains 5 schedules that, depending on the values of the trains, could be optimal: $\{A, D, F\}$, $\{A, E\}$, $\{B, C, E\}$, $\{B, C, F\}$, and $\{G\}$. Any other schedule either leaves out a possible train or causes conflict.

The difficulty of the "S" environment is determined by the 3 separate copies of the BP96 environment that are contained. Although there are $5^3 = 125$ schedules that need to be searched for the best schedule, optimal

[26] In the original experiments, two additional trains H and I were in the testbed. Because these trains did not conflict with any of the trains in $\{A, \ldots, G\}$, they were not relevant to the scheduling of the conflicting trains, and are not relevant to the discussions of this paper.

schedules can be determined in steps. The fact that the tracks are independent suggests that only 15 comparisons need to be made: a schedule for track 1 can be determined first, then a schedule for track 2, and then a schedule for track 3.

The difficulty of the "*C*" environment is much higher, because the tracks can no longer be scheduled in isolation. There are 139 different schedules that need to be considered. While there may be ways to reduce the number of comparisons from 139 to some smaller number, the interdependence of the conflicts on the tracks does not make any obvious methods apparent.

5.3 The railroad testbeds: Economic configuration

The economic configuration of the testbeds are identical, and are given by the set of agents $I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ along with an assignment of train values for each train and agent such as that shown in Table 2. Each agents values are private information, and agents are not given any information about the distribution of values.

Table 2 gives the train values for period 1 of the experiments. It is important to note that these train values are exactly the same values as were used in the BP96 experiments. The values for period 1 of the BP96 experiments became the values of the trains on track 1 in the experiments reported here. The values for period 2 of the BP96 experiments became the values of the trains on track 2 in Table 2. The values for period 3 of the BP96 experiments became the values of the trains on track 3 in Table 2.

Periods 2 and 3 of the experiments use different train values, which in the interests of space are not shown. However, the link to the BP96 experiments is maintained. Period 2 of the new experiments used periods 4, 5, and 6 of the BP96 experiments for the train values for tracks 1, 2, and 3 respectively. Period 3 of the new experiment used periods 7, 3, and 4 of the BP96 experiments to obtain the train values on tracks 1, 2, and 3.

Isomorphic physical and economic configurations of the BP96, "*S*", and "SIP" scheduling environments imply that important BICAP variables, such as final bid prices, bid revenue, allocation efficiency, and aspects of bidding behavior, should be related across these 3 sets of experiments. For example, bidding behavior observed in the BP96 experiments in period 5 should correspond-under the isomorphism – to bidding behavior to be observed in period 2, track 2 of the "*S*" or "SIP" experiments.

The "*C*" environments, although they involve identical train values to the "*S*" environments, do not involve an identical physical configuration. The substantially different nature of the feasible set will make the comparison of economic variables impossible between the earlier BP96 environment and the "*C*" environment. The purpose of the "*C*" environment is to provide observations with a harder problem.

Table 2. Train redemption values $V_j[r]$ for Period 1 in the experiments

Period 1 – Track 1

Agent id#	Train A	Train B	Train C	Train D	Train E	Train F	Train G
0	332	232	878	708	746	426	***2619***
1	946	521	321	241	739	265	2491
2	302	198	307	270	1013	645	1329
3	***1699***	***645***	307	206	306	217	509
4	1282	454	***1634***	***1447***	341	134	2543
5	801	354	933	465	936	561	2339
6	389	242	387	117	583	348	423
7	320	132	1405	974	528	360	594
8	708	332	309	188	***1635***	***1421***	2005
9	372	277	341	138	395	284	1549

Period 1 – Track 2

Agent id#	Train H	Train I	Train J	Train K	Train L	Train M	Train N
0	368	133	***683***	***346***	320	108	1604
1	***1124***	***980***	319	269	340	291	93
2	303	219	335	168	***1359***	***641***	373
3	305	171	371	149	524	177	466
4	403	325	463	237	475	382	124
5	692	487	320	267	1027	515	1625
6	405	315	370	194	375	284	570
7	413	311	417	343	430	377	531
8	558	340	354	270	577	224	304
9	362	154	320	96	312	206	***1710***

Period 1 – Track 3

Agent id#	Train O	Train P	Train Q	Train R	Train S	Train T	Train U
0	425	365	360	116	500	310	598
1	319	241	337	263	463	194	***1843***
2	528	382	350	117	306	206	1570
3	***1858***	***615***	840	662	384	264	412
4	456	376	***1227***	***964***	315	105	206
5	660	405	342	217	328	169	1336
6	413	227	314	248	368	257	382
7	448	290	371	274	***943***	***774***	1387
8	312	267	1025	657	482	341	247
9	300	109	451	244	309	257	1731

5.4 BICAP + CPCA Configuration

Train scheduling agents

The set of scheduling agents, who could submit CPCA proposals, was the same as the set of train operator agents, who could bid, in all the experiments except experiment SIP1.

In experiment SIP1, the scheduling agents were 3 additional subjects that did not bid on the trains. Only these 3 agents could make proposals. Several questions are addressed by this design. Would CPCA still work with such a small group of scheduling agents? Would agents, somehow, implicitly collude to extract large bonuses from the system? Should agents who suggest schedules be independent of the agents who are bidding, or is a system where bidders can suggest schedules actually more desirable?

Timing

In the experiments, the BICAP + CPCA timer was set from 30 to 60 seconds depending on the subjects. Longer times reduced subjects anxiety about rushing to bid, but longer times also allow subjects to delay the process when they increase bids by only 1 Fr at a time.

5.5 Software

Identical versions of the BICAP + CPCA rules were implemented in two different versions of the software. The differences in the software that are relevant to the experiments reporter here primarily revolve around the user interface.

Version 1

The experiments that ran at Caltech involved a DOS version of the software. This software displayed a table of bids on the screen and a set of keyboard commands that could be used to enter a new BICAP bid or send a CPCA scheduling suggestion. Color codes indicated the current CPCA scheduling information. The software was a modification of the software used in the BP96 experiments.

Version 2

The experiments that ran at Georgia State involved a LINUX[27] version of the software. This software displayed bids and proposals on separate screen windows. The mouse was used instead of the keyboard to select a command. The use of the mouse and graphical windows made describing and using the software much easier.[28] The LINUX software, while not DOS compatible,

[27] LINUX is a full-featured, free version of UNIX that was developed by a collaboration of thousands over the internet. It is named for its principle architect, Linus Torvalds. This operating system was chosen because at the time, DOS/Windows based systems could not easily provide the desired multi-user internet capabilities.

[28] The subjects did not need to know anything about LINUX. As the displays were mouse driven, the subject simply clicked on a button corresponding to the desired action – bidding or suggesting a schedule.

could be run on the internet and thus allows the potential for larger experiments in the future.

6 Experimental results

This section will provide analysis, support, and discussion for four primary results of the experimental laboratory research.

Principal findings

Result 1. The Computation Procuring Clock Auction (CPCA) exhibited *computational effectiveness* inducing agents to compute schedules that maximize bid revenue.

Result 2. CPCA exhibited *computational robustness* in inducing some agents to make up for the shortfalls of others. Such robustness involves the ability of the CPCA process to aggregate information from different sources.

Result 3. CPCA exhibited *low total costs* for suggestion incentives. Costs are relatively low even over a wide range of concentrations of computing activity. A form of competition seems to be present even when only a few agents are submitting suggestions.

Result 4. Observation of BICAP + CPCA bidding behavior sustains a conjecture of *computational invariance* in comparison with behavior of BICAP + Central Computin. BICAP bidding incentives, as evidenced by BICAP bidding behavior revenue, do not seem to be adversely affected by the introduction of CPCA as the computing technique.

The argument involved in this series of results is one of proof of principle and design consistency. Result 1 shows that CPCA can be effective as a computational technique in this testbed railroad scheduling application. The Computation Procuring Clock Auction provides sufficient incentive to induce the agents to compute the bid maximizing potential allocations and to submit this information as CPCA proposals before the end of the round in every experiment studied.

It is not enough to know that CPCA exhibited adequate performance as a decentralized computing mechanism. The mere fact that a mechanism can work in a few cases does not logically imply that it will work in the general case or even in similar cases where it has not been tested. For this reason, laboratory researchers have acknowledged the importance of examining the design consistency of a mechanism. The idea is to answer not only the question 'Did CPCA work?', but to also examine other questions: 'Why did it work?'; 'Did it work for the right reasons?'. Results 2 through 4 will show that CPCA is operationally consistent with economic principles implicit in its design.

6.1 Result 1. Computational effectiveness

Data

Section 4 provides a discussion of the measure of CPCA computing effectiveness that will be used here to analyze the experimental data.

Recall that

$$\textit{Computing Effectiveness} = \frac{Q_B(S^{\mathrm{CPCA}})}{Q_B(S_B^{\mathrm{OPT}})},$$

where S^{CPCA} is a schedule produced by the CPCA process and S_B^{OPT} a schedule that produces maximum bid revenue at BICAP bids B. The S_B^{OPT} schedules were not directly observable to agents or the experimenter, but can be calculated from the observed bids in the experiment, and are the standard against which the performance of the CPCA process is being judged. Recall that the function $Q_B(S)$ is the sum of current bids for the schedule S.

If the CPCA process works correctly – if CPCA is computationally effective – then the total bid revenue from the schedule that CPCA produces, $Q_B(S^{\mathrm{CPCA}})$, will equal the maximum possible bid revenue from these bids, $Q_B(S_B^{\mathrm{OPT}})$. Thus, computational effectiveness is indicated by a ratio value of 100% and lack of effectiveness is indicated when the ratio is below 100%. Ratio values above 100% are not possible.

Table 3 shows the calculation of the CPCA Computing Effectiveness at the close of each experimental period. From left to right, the table is read as follows. The first two columns give the experimental environment (environment S or environment C),[29] experiment number and the period number. Column 3 gives the observed final BICAP + CPCA allocation. Column 4 gives the allocation that would maximize bid revenue to the scheduling authority given the bids submitted by the agents in that round. This bid maximizing schedule in Column 4 is the schedule that CPCA is supposed to induce agents to compute. Column 5 gives the related measure of CPCA computing effectiveness: the ratio corresponding to the sum of bids at the observed [col. 3] allocation divided by the sum of bids at the bid revenue maximizing [col. 4] allocation.

Analysis and support

Table 3 shows that in each experiment, CPCA achieved 100% computing effectiveness for the final schedules and allocations. This data provides the necessary support to establish that CPCA is computationally effective in the testbed environments.

[29] Recall that environment "C" is more difficult than environment "S", and that within an experimental environment the matrix of train values V were varied from period to period within an experiment, giving rise to different patterns of bidding and different computational problems in each case.

Table 3. Computing effectiveness of CPCA: Final allocations

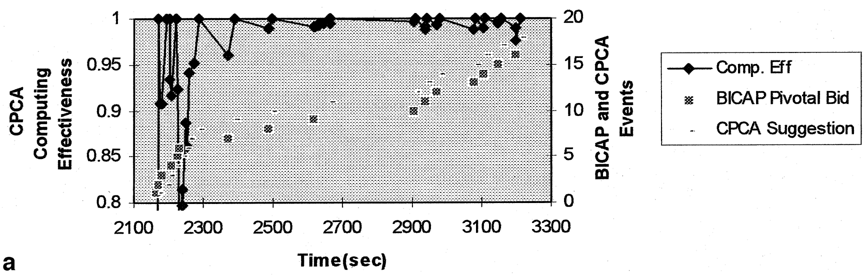
Experiment	Period	System allocation realized in experiment S^*_{OBS}	Bid revenue maximizing allocation at closing bids on trains $S^*_{MAX}(\underline{B}_{OBS})$	CPCA computing effectiveness $\frac{Q(S^*_{OBS}:\underline{B})}{Q_{MAX}(\underline{B})}$
S1	01	BCEIJLPQS	BCEIJLPQS	1.0
	02	BCEHKMPQS	BCEHKMPQS	1.0
	03	ADFHKMPQS	ADFHKMPQS	1.0
SIP1	01	ADFIJLPQS	ADFIJLPQS	1.0
	02	BCEHKM*PQS	BCEHKMPQS	1.0
	03	ADFHKMPQS	ADFHKMPQS	1.0
S2	01	ADFIJLORT	ADFIJLORT	1.0
C1	01	ADFILPQS	ADFILPQS	1.0
	02	GHKMPQS	GHKMPQS	1.0
	03	ADFNPQS	ADFNPQS	1.0
C2	01	BCEHLORT	BCEHLORT	1.0
	02	BCEHLORT	BCEHLORT	1.0
C3	01	BCEHLORT	BCEHLORT	1.0

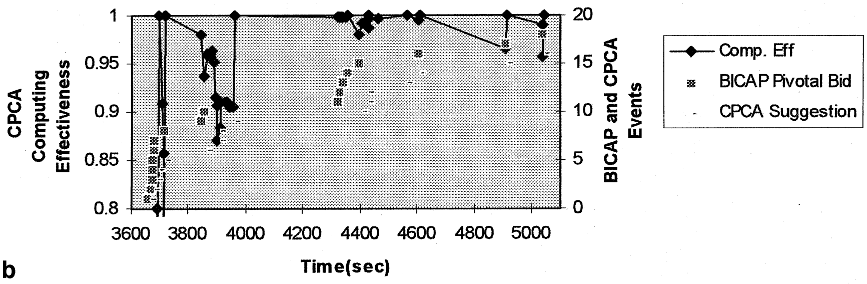
Discussion

Table 3 clearly shows that CPCA always terminated at the bid maximizing feasible allocation in each period of each experiment. But it is possible to show more. In this discussion it will be shown that CPCA finds the 100% bid revenue maximizing schedule many times in a period, changing in response to the bids submitted by the agents. It does not calculate these new schedules instantly, and it occasionally misses a few opportunities to provide feedback, with most of the missing feedback occurring at the first minutes of each period. However, from the middle to the end of the each period, the CPCA process provided, with a several second delay, similar scheduling feedback to what a central computer could provide.

While CPCA was shown to be effective in calculating the final schedules, it is also important to ask how CPCA performed in calculating the potential schedules during the BICAP + CPCA process. Recall that in the BICAP process with central computing, the computer calculates – after each bid – a new schedule showing the effect of that particular bid on the potential scheduling outcome. CPCA decentralizes this task, relying on agents to submit suggestions about which schedules will provide higher revenues. Proper feedback to bidders in the BICAP process depends critically on the computing effectiveness of the CPCA process.

Time series data for computing effectiveness within a period tend to resemble Figures 3.1–3.3 in that 100% computing effectiveness is lost and regained many times during a period. This pattern in the figure is caused by a repeating pattern of events in the data: some agents submit pivotal bids that change the bid maximizing feasible allocation and thus cause the denominator of the computing effectiveness ratio to increase. This causes the com-





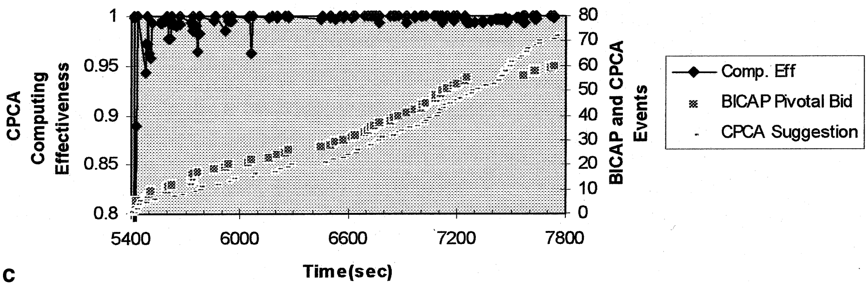


Figure 3. **a** CPCA computing effectiveness during period 1, experiment C1. **b** CPCA computing effectiveness during 2, experiment C1. **c** CPCA computing effectiveness during 3, experiment C1

putation effectiveness to drop. CPCA then gives agents an incentive to find and submit new scheduling proposals appropriate to these new bids. The agents eventually respond each time, increasing the computing effectiveness ratio back to 100% . Figure 3 shows that the CPCA schedule is acting as an effective running approximation of the bid maximizing feasible allocation.

Another measure of the interim effectiveness of CPCA involves comparing the time of arrival of *pivotal bids* – bids which change S_B^{OPT} – and the time of arrival of subsequent CPCA suggestion messages. The figures give an idea but not the entire pattern of the data. For that reason, the raw data was examined in detail. A different experiment – experiment $S1$ – was examined.

For the first 3 to 4 minutes of experiment $S1$, period 1, ($T = 1756$ to $T = 1960$), the CPCA schedule is being updated but does not follow the bid maximizing schedule very well. Intuitively, this suggests that the agents are only beginning to learn how to solve the relevant constrained improvement problem. However, there is a time when CPCA begins to track the changes in the bid maximizing potential allocations. After 5½ minutes ($T = 2071$), in experiment $S11$, each pivotal bid is followed, within 7 to 23 seconds, by a suggestion message that updates the CPCA schedule to once again match the bid revenue maximizing schedule. This pattern continues for the remainder of the period, which lasted just over 17 minutes. The pattern of data in the other experiments, while not presented due to space, is fairly similar. CPCA does occasionally skip improvements later in the period, but most of the skipped improvements are concentrated in an initial "learning" phase of each round.

6.2 Result 2. Computational robustness

Computational robustness means that a number of agents participate in the CPCA process, and that when some of these agents fail to calculate or submit information, other agents correct for these failures.

Demonstrating CPCA computational robustness involves identifying features in the experimental data that reveal failures of individual agents and corrections by other agents.

Data

The data analysis identifies the omissions of some agents involved in the CPCA process and how other agents corrected for these omissions. A new table categorizing the CPCA suggestion messages was compiled from the raw data of various experiments and is included as Table 4. A CPCA suggestion message is categorized a "Complete Solution" if the suggested schedule is in fact the bid revenue maximizing schedule. Otherwise, the suggestion message is categorized a "Partial Solution". A "Partial Solution" is still an improvement over the current schedule but is not the bid revenue maximizing schedule.

The number of partial and complete solution suggestion messages is tallied for each experiment and period, and by what type of agent submitted the suggestion.

A *pivotal bidder* is the agent whose recent bidding activity has changed the schedule that ought to be adopted. For example, if the schedule can be $\{A\}$ or $\{B\}$, and the current schedule is $\{A\}$, and some agent places a very high bid on $\{B\}$ that ought to change the schedule, then that agent would be a pivotal bidder. The schedule need not actually change from $\{A\}$ to $\{B\}$. *Others* are simply agents who were not the pivotal bidder at the time the new schedule was suggested.

Analysis and support

The argument supporting CPCA computational robustness involves three elements.

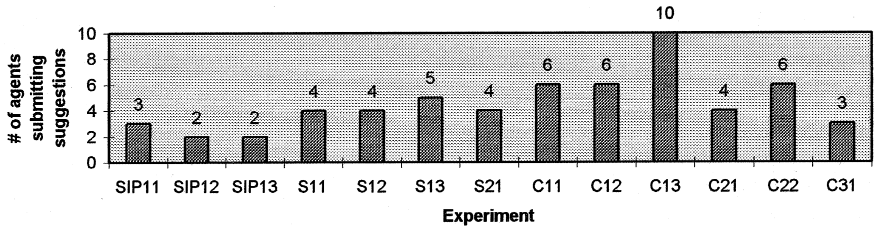


Figure 4. CPCA participation

(A) A number of different agents participate in the CPCA process as shown by Figure 4.

(B) The categorization of CPCA suggestion messages in Table 4 indicates that some pivotal bidders who did submit a CPCA suggestion message after their bid failed to submit a complete solution, but only submitted a partial solution. A partial solution means that an improved schedule was submitted, but this schedule did not maximize bid revenue from the available high bids. This indicates that agents are not always submitting[30] scheduling information that is associated with their bids.

(C) The computational effectiveness of CPCA exhibited in Result 1 shows that a complete solution is eventually submitted to CPCA. At the end of the period, the CPCA schedule matches the bid revenue maximizing schedule because some other agent submitted a complete solution. Thus, some agents in CPCA are shown to correct for the failure of others.

Discussion

Result 2 shows that in the laboratory testbed environments, some agents can and do correct for the omissions of others. Intuition involved in the design of the process suggest two factors relevant towards extending this result to future research and possible application.

Result 2 suggests that the payment of bonuses to agents who submit information to CPCA is probably an important part of the process. In a world of ideally rational bidders, one could consider abandoning the CPCA bonus payments. Pivotal bidders would always understand how their bid affected the bid revenue maximizing schedule, and would automatically provide updates to CPCA because otherwise their pivotal bids would not be included in the final CPCA schedule. A pivotal bidder who did not submit a CPCA suggestion message would not change the schedule, and their bid would be worthless.

Table 4 shows that when pivotal bidders send CPCA suggestion message, the suggested schedule is often a full solution for the bid maximizing schedule. However, the same table shows that the pivotal bidders sometimes

[30] It is also true that the agents might be unaware of whether their bid is sufficiently high to change the schedule. However, we can not observe what agents are aware of – only their bids and suggestion messages are observable.

Table 4. Partial vs. full solution CPCA proposals

Experiment	CPCA proposals by pivotal bidders		CPCA proposals by others	
	Partial solution	Full solution	Partial solution	Full solution
SIP11			4	4
SIP12			9	4
SIP13			6	4
S11	0	4	7	1
S12	1	6	8	8
S13	4	1	15	5
S21	0	0	11	2
C11	2	6	5	5
C12	1	1	7	5
C13	1	7	24	38
C21	2	4	20	3
C22	1	12	27	28
C31	0	1	7	0
Total S+C	12	42	131	95

submit partial solutions, which suggests that they either do not fully understand the effect of their pivotal bid or are acting in some strategic manner. In these cases, it is important that other agents are rewarded for finding improved schedules.

CPCA functions well because it does not allow the bid revenue from a schedule to go down but it accepts any schedule change that causes bid revenue to increase. Because CPCA provides no rules for evaluating how an agent calculated an improved schedule or who the agent is, the design of the CPCA process makes possible both omissions and the correction of omissions. The fact that a mistake is an omission by a pivotal bidder, who might normally be assumed to know the intent of his bid, and the fact that a correction was submitted by someone who merely guessed a better schedule, are facts never evaluated formally within the process. In this way, CPCA impartially accepts and aggregates different kinds of information.

6.3 Result 3. Costs of CPCA incentives

CPCA exhibited *low total costs* for suggestion incentives. Costs are relatively low even over a wide range of concentrations of computing activity. A form of competition seems to be present even when only a few agents are submitting suggestions.

Data

In a given period of a given experiment, the sum total CPCA bonuses paid across the agents ranged from 10 Fr to 350 Fr. The total bid revenue was in the range of 6000–8000 Fr.

Figure 5 explores the relationship between the concentration of rewarded CPCA activity in the experimental environment, and the total cost of CPCA. On the horizontal axis, the code for the experiment and period is given (e.g., *C*12 indicates period 2 of experiment *C*1) along with the HHI concentration of the bonus payments. The vertical axis shows the total cost of CPCA incentives that were paid out to agents in that period, as a percentage of the bid revenue.

This HHI index is a standard Hirschmann-Herfandahl index, familiar in industrial organization theory, applied to the CPCA bonus payments received by the agents. In this case, the HHI equals the sum of the squares of the CPCA bonus shares of all the agents in a particular period. For example, suppose in a period agent 1 received a 30 Fr CPCA bonus payment, and agent 2 received a 20 Fr CPCA bonus payment, and other agents received nothing. Then the sum total of the bonus payments is 50 Fr across the various agents, and agent 1 received 60% of this whereas agent 2 received 40% and other agents received nothing. The HHI is the sum of the squares of the percentage shares. In this example the $\mathrm{HHI} = 60^2 + 40^2 + 0^2 = 3600 + 1600 = 5200$. Thus in pure monopoly the HHI would be $100^2 = 10,000$. If all ten agents received an equal 10% share of the CPCA bonus payments, the HHI would be $10^2 * 10 = 100 * 10 = 1000$. As shown in the graphs, the observed HHI of the CPCA bonuses varies among the different experiments and periods.

Analysis and support

Figure 5 shows that the CPCA bonus payments typically consume 1% –5% of BICAP bid revenue. This is a low cost when compared to the monopoly outcome. A monopolist could wait until the bonus rate was 100%, and collect 100% of BICAP bid revenue as a CPCA bonus.

The figure shows that even when the CPCA activity was fairly concentrated, costs do not rise as might otherwise be expected. This supports a conjecture that competition exists and induces agents to reveal information

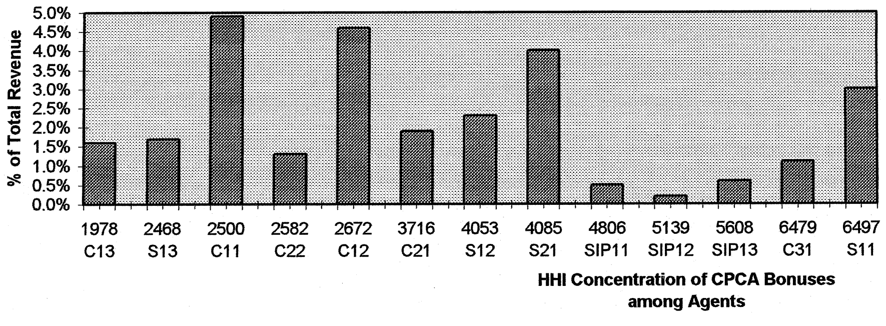


Figure 5. Total cost of bonuses vs. concentration

at low bonus rates. Agents do not tend to wait for higher bonus rate levels, even when only 2 or 3 agents are submitting suggestions.

Discussion

While the result shows that CPCA costs are low, an understanding of the nature of the competition within the CPCA process is desirable in order to understand why the costs are low.

Pivotal bidders play a role in keeping CPCA costs low in two ways. First, from Table 4 many pivotal bidders submit full CPCA solutions that detail how their bid changes the bid revenue maximizing schedule. These submissions are often made shortly after a bid, and at a zero or low CPCA bonus rate. Attempts from pivotal bidders to insure that their bid is included in the schedule present serious competition to agents who are merely watching the process, and might keep overall CPCA costs low.

The fluctuating nature of the bonus rate timer is another factor that might keep CPCA costs low. Figure 6 shows the bonus rate for the *C*1 experiment. Other experiments display similar data. Recall that the bonus rate uniformly increases from 0% to 100%, so long as no bid or suggestion messages are received. Any BICAP bid or CPCA suggestion message that meets the improvement rule requirements will cause the CPCA bonus rate to be reset to

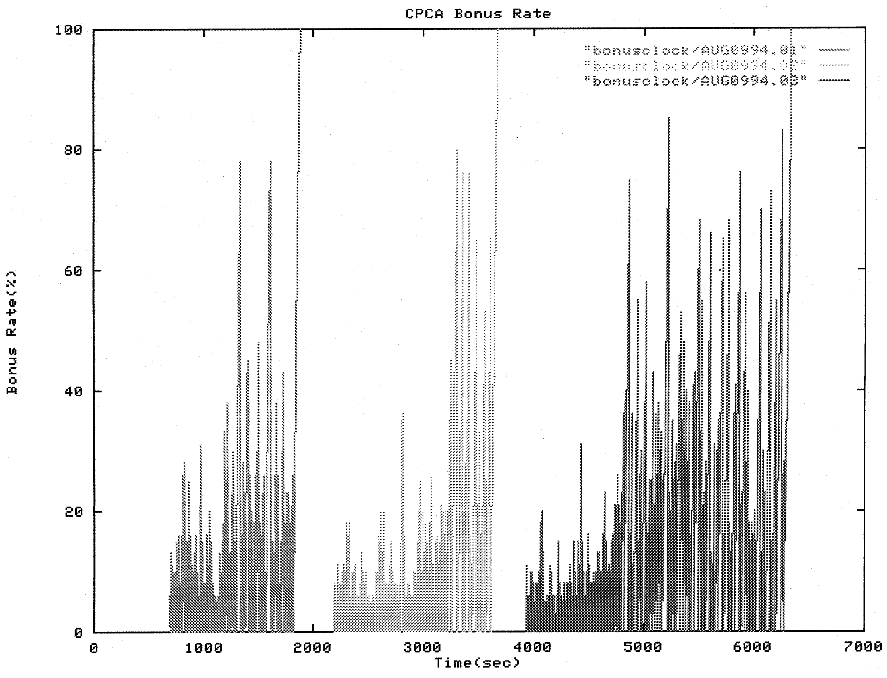


Figure 6. CPCA bonus rate (experiment C1, periods 1–3)

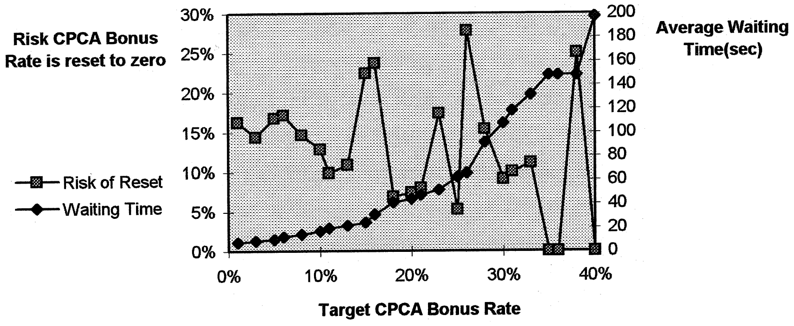


Figure 7. Risks faced by CPCA participants seeking larger CPCA bonuses (experiment C1, period 1)

0%. Thus, a graph of the bonus rate will indicate low bonus rates at times of high bidding activity and high bonus rates at times of low bidding activity.

The noisy nature of the bonus rate introduces some uncertainty for an agent who has information to submit to CPCA and wants to maximize their payment for the information.

Figure 7 provides a graphical depiction of two risks caused by the noisy bonus rate. The data shown is taken from period 1 of experiment *C*1.

One risk involves the fact that the agent does not know how long he will have to wait for the bonus to climb to 20%, 50%, or some other target level. The longer an agent must wait, the higher the likelihood is that someone else will claim the bonus for the information.

A second risk that agents face when submitting CPCA suggestions occurs when the bidding is very active. An agent wanting to submit a CPCA suggestion does not know if another agent will be pressing the bid key a few tenths of a second before he presses the suggestion key. If another agent submits a bid, the computer running the BICAP + CPCA process will reset the CPCA bonus rate to 0%. If the agent submitting the CPCA suggestion does not notice this, and submits the suggestion anyway, then they will receive a bonus of zero. While this does reduce CPCA costs, it does increase the risks for agents who compute improved schedules. Questions from agents during the experiment showed that agents did encounter this effect in the process.[31]

6.4 Result 4. Conjecture of computational invariance

A conjecture that CPCA does not affect BICAP bidding revenues can be sustained.

[31] Complaints from experimental agents usually took the form of a "computer bug" report or a report that something unfair was happening. Agents were told individually that a chance existed that someone else placed a bid at roughly the same time that they entered a suggestion. The bonus rate would be determined by which event was processed first by the mechanism.

Data

Figure 8 shows a comparison of the total BICAP revenue from the experiments of Brewer and Plott (1996) [BP96] and the new experiments reported here [SIP1, $S1$, and $S2$]. Figure 8 shows the revenues in economically comparable environments.

From the discussion of section 5.3, the BP96 environment and the SIP1, $S1$, and $S2$ experiments should produce comparable patterns of bidding. There is a one to one correspondence between the physical and economic configurations of the trains on a particular track of the S environment and of the trains $\{A, \dots, G\}$ of the BP96 environment.

Analysis and support

Figure 8 shows that the BICAP bid revenue is a slightly higher in the new experiments, where CPCA was used for train schedule calculation, than in the BP96 experiments, where a central computer was used for train schedule calculation.

Comparable track environments produce similar revenues in both the BP96 and the BICA + CPCA experiments.

This data shows that CPCA did not noticeably reduce the BICAP bid revenue. If anything, the bid revenue increased slightly.

The data also shows that in later periods, the revenue followed theoretical predictions just as in earlier periods. There is no trend away from the theoretical predictions. Thus, CPCA does not somehow encourage agents to collude over time.

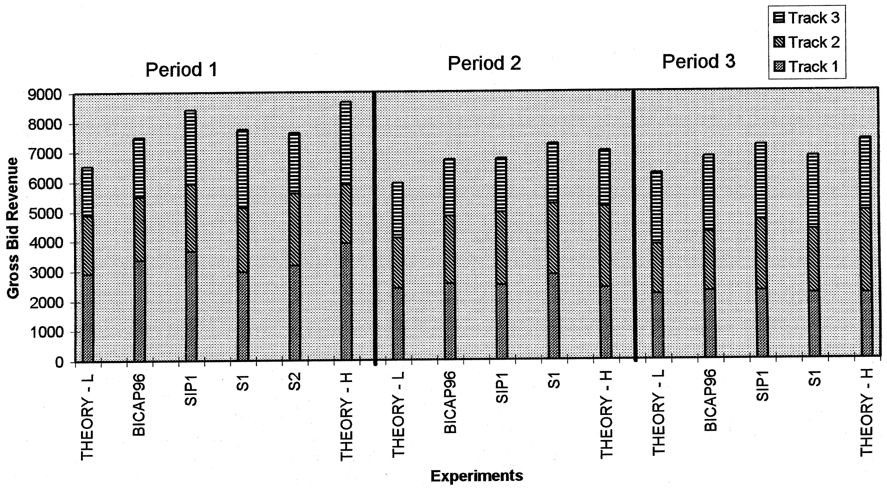


Figure 8. Comparison of bid revenue from BICAP, BICAP + CPCA and theoretical upper and lower bounds

Discussion

While results 1–3 involve relatively “clean” data and strong results, result 4 is stated as a sustained conjecture. Several pieces of evidence exist that point to mixed results. Future experiments in a variety of environments may help to resolve the questions at hand.

What does it mean for the BICAP mechanism to be relatively ‘unharmed’ by a transition from central computing of schedules to CPCA computing of schedules? The idea is that the same economic principles that make BICAP relatively efficient should also be observed to operate in BICAP + CPCA.

Thus, the bid revenue is only a part of the possible evidence that might need to be examined in order to make a complete determination. Other sources of evidence that could be compared would include the final train schedules for each agent, the final bid prices, whether or not a one stage nash equilibrium was reached, and the nature of agent bidding behavior during the auction.

Mixed results from the analysis of this evidence is summarized below.

- Allocation efficiencies[32] were in the range 90–100% in 10 of the 13 experimental periods. On average, the allocation efficiency with BICAP + CPCA is fairly high: 91.5% in the “S” environments and 93.6% in the “C” environments. With BP96, in a much simpler environment, the average was 97%. The somewhat lower efficiencies were expected as a result of studying the much more complex environments and as a possible cost of using a procedure like CPCA.
- Low allocation efficiencies exhibited in 3 periods point to possible causes of difficulties in the mechanism. The two lowest cases of allocation efficiency are 67% in period 1, experiment SIP1; and 77% in period 3, experiment $C1$. In the case of SIP1 period 1, the overbidding of agent 0 and agent 2 caused the low efficiency. In the case of $C1$ period 3, overbidding did not occur. However, agents 1 and 4 could have submitted bids that would have greatly improved their individual profits and the overall allocation efficiency, but failed to do so.
- Overbidding was a problem in 5 of the 13 experimental periods. Overbidding means that some agent bid over his redemption value by a large amount. Because 3 of these “overbid” periods occurred in the SIP1 experiment, this overbidding does not seem to be related to arbitrage capture strategies between BICAP and CPCA as mentioned in section 4. Recall that in the SIP1 experiment, bidders could not enter scheduling proposals. Thus, overbidding is not directly related to a strategy to overbid and then submit a related scheduling proposal. When overbidding occurred, other agents usually took advantage by not bidding as

[32] Allocation efficiency is determined by comparing the sums of the values (e.g., from table 1) at the final schedule to that obtained by solving the efficient scheduling problem of section 2. Thus an allocation efficiency of 100% means that the most valuable trains are scheduled to be run by those agents who value them the most.

high on complementary schedules.[33] Thus, overbidding did not always affect the total bid revenue by very much.

- Nash-like (NE1) bidding equilibrium were frequently not obtained. In contrast, in the BP96 experiments, NE1 were frequently obtained. A NE1 profile of bids exists when no agent can submit a pivotal bid that increases his or her own profit. Not counting the 5 periods where overbidding occurred, 6 of the 8 remaining periods involved a failure by one or more agents to recognize a 'valuable' pivotal bidding opportunity. In this case, 'valuable' means that the agent could have increased their profit by at least 100 Fr by submitting another bid. Instead, the agents allowed the clock to run out and the mechanism to close.

Mistakes by subjects may be the problem here and certainly exist in many other types of experiments.[34] The tasks which subjects must perform are fairly complex, and include watching a constantly updating screen and comparing the bids with their own values on separate redemption value sheets. The subjects must form a bidding strategy that is consistent with their own objectives and the complex interdependencies of the feasibility constraints, and they must execute this strategy on an unfamiliar[35] software system. Thus the fact that BICAP + CPCA works at all is strong evidence for the value of future research. The 2nd version of the software, which was used for the experiments at Georgia State, had a friendlier[36] user interface – and data from a future series of experiments may resolve many of the problems reported here.

[33] For example, Figure 2 shows that in all environments, trains A and D are in competition with trains B and C. If trains A and D are scheduled, B and C can not be scheduled, and vice versa. If someone overbids on A, then the bidder for train D need not bid as high in order to help insure that the *total* bid on A and D beat the *total* bid on B and C.

[34] For example, in a large number of experiments that use market systems, agents sometimes trade at prices far from the equilibrium simply because one of the agents mistyped his bid or ask. Usually, successive trades return to the equilibrium price and there is little effect on total revenues or efficiency; just a decrease in the profit of error prone agents. In auctions, mistakes can have a more permanent effect on the data. An effort was made to correct errors reported by subjects, but subjects did not always announce their overbidding as a typo.

[35] Although a practice period was provided after reading the instructions, it is safe to say that the subjects had never seen this software before and were unfamiliar with its weaknesses.

[36] In a new series of 1997 experiments currently in progress, changes in the software have improved the ability of subjects to cope with their environment. The addition of redemption values to the screen displays and the display of warning messages about overbidding has reduced it considerably. Bidding over the redemption value causes a box to appear where the subject is warned and then asked if they still wish to enter the bid. Overbidding still sometimes occurs, but not at the levels reported in this initial series of experiments. These developments suggest that subject error is responsible for much of the problems reported here.

7 Conclusions

This paper began with questions concerning the large scale feasibility of smart markets. The question posed was whether it might be possible to develop techniques for finding or calculating bid maximizing feasible allocations in a way that might allow smart markets to function at larger and larger scales. In particular, the technique was to rely on economic rather than technical concepts – the idea was not to find a better mathematical algorithm for finding optimal solutions, but rather, to develop an incentive structure that would cause partial solutions and, eventually, a best known solution to be revealed. At the same time, the technique for computing bid maximizing potential allocations should only minimally affect the incentives for bidding in the market.

The answer is that such an incentive system can be constructed. The Computation Procuring Clock Auction (CPCA) is such an incentive system. Experiments were performed where CPCA successfully replaced the central computing aspects of the BICAP smart auction. In several testbed environments, CPCA always obtained the bid maximizing potential allocation by the close of each period. Often, CPCA potential allocations tracked the bid maximizing schedule within several seconds of any changes. The variety of environments in which CPCA was tested shows that its ability to find optimal solutions is not a fluke or an accident of choice of experimental procedures.

A conjecture can be sustained that CPCA provided effective computation because it provided incentives similar to those mentioned by Hayek (1945) for problems of much larger scale. CPCA functioned effectively because:

- CPCA gives agents the incentives to compute improvements or changes to small parts of the big problem of how to schedule the railroad network. Computation occurs in parallel by many agents.
- CPCA provides for an information transfer capacity through the potential allocation. The potential allocation summarizes information about the best known schedule – an agent does not need to know why the solution has changed in order to react with new BICAP bids for trains and/or a new CPCA proposal to change the solution for scheduling trains on the tracks.
- Profitability in the marketplace regulates the activities of the agents in regards to providing effective information or computation. Whenever an information or computation activity can be profitable, CPCA encourages it. Whenever information or computation activity is unprofitable, the incentives in the market discourage it. In CPCA we do not know whether agents' computational activities were profitable, because the costs are the costs of human effort and are probably varied and unobservable.
- CPCA does not strictly prevent or prohibit activities on the grounds that they are not the best technical means of solution. Many types of computation and information are provided by agents in the marketplace that are very different from what would be suggested by technical experts desiring a complete, central solution. CPCA solutions may partially depend on agents'

abilities in understanding the strategy of other agents' BICAP bids, a type of knowledge that would be very difficult to program into a computer.

As an algorithm for optimization, CPCA has some interesting features that might be useful in many other applications besides smart market applications. The algorithm by which CPCA computes the bid maximizing potential allocation is likely to be very different than the one that a computer programmer or operations research specialist would construct for this type of problem.[37] In CPCA calculation of optima are not centrally controlled or dispatched and there is no control to prevent the duplication of effort – several agents might be simultaneously searching for very similar solutions.

Whether duplication of effort would necessarily point to an inefficiency is a potential point for future debate about computing mechanisms. Huberman and Hogg (1995) and Huberman, Lukose, and Hogg (1997) have developed the idea of *computational portfolios*, where a number of different techniques are simultaneously tried for solving the same problem. The idea is that on a multiprocessing computer that can support a number of simultaneous programs, choosing a level of time sharing between a number of solution techniques is similar to choosing a portfolio of stocks in financial theory. The solution techniques each have an uncertain ex ante yield in terms of time to a solution. Thus, a combination of techniques would involve a tradeoff between speed of solution and variance of solution time, just as a portfolio of stocks involves a tradeoff between performance and variance. CPCA would appear to be compatible with such techniques for computation, as it causes a market portfolio of solution techniques to evolve based on performance.

In addition, evidence exists suggesting that CPCA also procures substitutes for computation when they are available. In the experiments, agents often knew what effect a bid had on the value of certain allocations and were able to gain a small bonus by immediately revealing this information. These agents might be said to have what Hayek (1945) referred to as the knowledge of particular circumstances of time and place. The CPCA procedure caused the revelation of this special type of information that would be difficult to otherwise incorporate into a computing algorithm – information such as personal, strategic reasons for a particular bid and the ability to guess strategic reasons for others' bids. In addition, CPCA also gave agents incentives to undertake more extensive searches of the allocations when such simple rules-of-thumb would be ineffective.

While the current series of experiments shows that CPCA is an effective computational tool that can also elicit knowledge that is difficult to include in formal computer programs, future series of experiments are required to further determine CPCA's properties. Can CPCA cause agents to appro-

[37] Usually the push is for better algorithms and/or faster computers. The novel idea of *interactive optimization*, as suggested by Fisher (1986), involves linking human and computer abilities in solving optimization problems. No one, to my knowledge, proposes linking competing humans who have various computing abilities at their disposal.

priately use available computing tools when these tools are costly? In the current experiments, the only costs to computation for the agents was the cost of their own mental effort. Because CPCA does not place restrictions on what an agent does to construct a solution proposal, any algorithm or computer program for finding or approximating optima could be used with CPCA. In more complicated computational environments, various computer programs or other tools might have real economic costs as well as potential benefits. The CPCA procedure would help determine a parallel algorithm where agents were willing to undergo computing costs associated with the use of certain tools or techniques in order to gain the possible CPCA bonus. More complex environments need to be studied to determine how effective CPCA would be in these situations. Eventually, field tests might even be warranted. Perhaps CPCA, or something like it, could eventually be used as a means of rewarding teams of sophisticated, optimization research specialists who compete and/or collaborate to solve problems of importance to industry.

Appendix A

The many potential uses of CPCA as a *computation* mechanism, and in particular its use within allocation mechanisms such as smart markets, requires that the CPCA rules be stated in a sufficiently clear and general fashion so that its compatibility with other mechanisms can be ascertained.

Here we take an approach familiar to those who work with dynamical systems or state-machines. The mechanism has a state. In CPCA, the state is assumed to be public information and common knowledge (e.g. agents know that other agents have access to the state).

The operation of a mechanism, in general, can be described as follows. The mechanism starts in a universal initial state we will call $\varnothing$. The mechanism changes states only by processing messages, which may arrive asynchronously from agents or be a special "message from time". The message T will be considered as a message that some uniform time interval has passed. In this way, the description of a mechanism's reaction to the passing of time can be placed in the same framework as its reaction to messages from agents. The mechanism may or may not have terminal states, from which further changes are impossible. The terminal states, when they exist, may occur in operation either through messages from the agents or messages from time. In some allocation mechanisms, the terminal state provides a final allocation that is to be implemented. In CPCA, the terminal state will give the approximate mathematical solution of a constrained optimization problem.

The reaction of a mechanism to a message is given by a *transition rule*. The transition rule specifies an initial state, a message or messages, and a final state. If the mechanism is at the initial state, and the message or sequence of messages is received, then the state of the mechanism changes to the final state. If a transition rule does not exist for a given mechanism initial state and agents message, then a transition does not occur – the message is ignored.

A taxonomy of transition rules can make useful distinctions in required and optional transitions in mechanisms. When only one transition rule exists for a given state, that transition is called a *required transition*. A required transition must occur in order for processing in the mechanism to continue. When many possible transition rules exist for a given state, then at this state there are many *optional transitions*. This taxonomy of transition rules may help to identify bottlenecks that occur because of the limited abilities of agents involved in required transitions.

The notation used for required and optional transitions follows below.

Required transitions

notation

$$\text{State}\{\text{variables}\} \xrightarrow{\text{sender}\to\text{messages}} \text{New\_State}\{\text{variables}\}$$

interpretation

If the mechanism is at the initial State, then for the mechanism to continue operations, the sender must send the indicated message or messages, and then the mechanism state changes to the New_State.

notation

$$\text{State}\{\text{variables}\} \wedge [\text{applicability condition}] \xrightarrow{\text{sender}\to\text{messages}} \text{New\_State}\{\text{variables}\}$$

interpretation

If the mechanism is at the initial State, and the applicability condition is satisfied, then for the mechanism to continue operations, the sender must send the indicated message or messages, and then the state changes to the New_State.

Optional transitions

notation

$$\text{State}\{\text{variables}\} * \xrightarrow{\text{sender}\to\text{messages}} \text{New\_State}\{\text{variables}\}$$

interpretation

If the mechanism is at the initial State, and the sender sends the indicated messages, then the mechanism state changes to New_State.

notation

$$\text{State}\{\text{variables}\} \wedge [\text{applicability condition}] * \xrightarrow{\text{sender}\to\text{messages}} \text{New\_State}\{\text{variables}\}$$

interpretation

If the mechanism is in the initial State, and the applicability condition is satisfied, and the sender sends the indicated messages, then the mechanism state changes to New_State.

A rough description of the CPCA mechanism is as follows. The states of CPCA are a null initial state $\emptyset$, a listening state $L\{\ \}$, a verification state V $\{\ \}$, and a terminal closed state $C\{\ \}$. The agents include a market maker and a number of computing agents. The market maker starts the process by providing details of the constrained optimization environment. This message causes a transition from the $\emptyset$ state to a $L\{\ \}$ state. In the state $L\{\ \}$, CPCA will accept improvement messages from agents, and proceed to state $V\{\ \}$, where the improvement will be verified by the market maker, and cause CPCA to enter another $L\{\ \}$ state. Time affects the $L\{\ \}$ states in a manner that makes it more profitable for agents to send improvement messages. Eventually, if no improvement messages are sent, time will cause the mechanism to enter a terminal $C\{\ \}$ state.

CPCA states

$\emptyset$ – Null state
$L\{Q, F, x_n, B\}$ – Listening State
  variables for $L\{\ \}$
  Q = value function (maximand)
  F = feasibility function (constraints)
  x_n = n-th iterated potential solution
  B = bonus rate
$V\{Q, F, x_n, B, i, x^*\}$ – Verification State
  variables for $V\{\ \}$
  $Q, .F.x.B$ – as above
  i, x^* – potential improvement
$C\{Q, F, x_n\}$ – Auction closed.

Transition rules

The market maker has the option to start the auction by providing the details of the constrained optimization environment.

$$\emptyset * \xrightarrow{mm \to E(Q,F)} L\{Q, F, x_0 = \emptyset, B = 0\%\}$$

Time raises the bonus to 100% then closes the auction if no messages arrive from the agents.

$$L\{Q, F, x_n, B = 0\%\} * \xrightarrow{T} L\{Q, F, x_n, B = 1\%\} *$$
$$\xrightarrow{T} \ldots * \xrightarrow{T} L\{Q, F, x_n, B = 99\%\} *$$
$$\xrightarrow{T} L\{Q, F, x_n, B = 100\%\} * \xrightarrow{T} C\{Q, F, x_n\}$$

The market maker has the option to monotonically increase the value function

$$L\{Q, F, x_n, B\} * \xrightarrow{mm \to \Delta E(|\Delta Q|)} L\{Q + |\Delta Q|, F, x_n, B\}$$

Agents have the option to suggest improvements x^* to the current potential solution x_n

$$L\{Q, F, x_n, B\} * \xrightarrow{i \to S(i,x*)} V\{Q, F, x_n, B, i, x^*\}$$

The market maker must verify and pay agents who submit bonafide improvements.

$$V\{Q, F, x_n, B, i, x^*\} \wedge [F(x^*) \wedge (Q(x^*) > Q(x_n))] \xrightarrow{mm \to A(x*) + \Pi(i, B \cdot (Q(x*) - Q(x_n)))} L\{Q, F, x_{n+1} = x^*, B = 0\%\}$$

The market maker must reject and penalize agents for incorrect improvement submissions.

$$V\{Q, F, x_n, B, i, x^*\} \wedge \neg[F(x^*) \wedge (Q(x^*) > Q(x_n))] \xrightarrow{mm \to R(x*) + \Pi(i, -penalty)} L\{Q, F, x_n, B\}$$

Formal description of the BICAP mechanism
Transition rules
The market maker has the option to start the BICAP mechanism by providing the details of the trains and conflicts.

$$\varnothing * \xrightarrow{mm \to E(R,C)} L\{R, C, \underline{B} = 0, \underline{H} = 0, S = \varnothing\}$$

Agents have the option to place a bid b on a train $r \in R$.

$$L\{R, C, \underline{B}, \underline{H}, S\} * \xrightarrow{i \to B(b,r)} V\{R, C, \underline{B}, \underline{H}, S, i, b, r\}$$

The market maker must verify that a new bid is above the current high bid, and if so, must announce the new high bids. The mechanism then enters a wait state while the new bid maximizing feasible schedule S, is computed.

$$V\{R, C, \underline{B}, \underline{H}, S, i, b, r\} \wedge [b > B_r] \xrightarrow{mm \to A(b,r)} W\{R, C, \underline{B} = [\ldots, b, \ldots], \ \underline{H} = [\ldots, i, \ldots]\}$$

The market maker must reject any new bid that does not exceed the current high bid for that train.

$$V\{R, C, \underline{B}, \underline{H}, S, i, b, r\} \wedge \neg[b > B_r] \xrightarrow{mm \to R(b,r)} L\{R, C, \underline{B}, \underline{H}, S\}$$

A computing agent (which might be the market maker or a machine controlled by the market maker) must compute the new bid maximizing feasible schedule whenever a wait state is entered.

$$W\{R, C, \underline{B}, \underline{H}\} \xrightarrow{comp \to Sched(S)} L\{R, C, \underline{B}, \underline{H}, S\}$$

Formal description of the BICAP + CPCA mechanism
Transition rules

The market maker has the option to start the BICAP + CPCA mechanism by providing the details of the trains and conflicts to BICAP. The CPCA environment $(Q(x), F(x), x_0)$ is determined from R and C.

$$\emptyset * \xrightarrow{mm \to E(R,C)} L\{R, C, \underline{B} = 0, \underline{H} = 0, S = \emptyset\}$$

Agents have the option to place a bid b on a train $r \in R$.

$$L\{R, C, \underline{B}, \underline{H}, S\} * \xrightarrow{i \to B(b,r)} V\{R, C, \underline{B}, \underline{H}, S, i, b, r\}$$

The market maker must verify that a new bid is above the current high bid, and if so, must announce the new high bids and send a CPCA environment change message.

$$V\{R, C, \underline{B}, \underline{H}, S, i, b, r\} \wedge [b > B\_r] \xrightarrow{mm \to A(b,r) + \Delta E(\Delta Q = \Delta \underline{B})}$$
$$L\{R, C, \underline{B} = [\ldots, b, \ldots], \underline{H} = [\ldots, i, \ldots], S\}$$

The market maker must reject any new bid that does not exceed the current high bid for that train.

$$V\{R, C, \underline{B}, \underline{H}, S, i, b, r\} \wedge \neg[b > B_r] \xrightarrow{mm \to R(b,r)} L\{R, C, \underline{B}, \underline{H}, S\}$$

Appendix B: Experimental instructions. BICAP + CPCA 8/9/94, 8/10/94, Experiment C1, C2

The instructions that follow are original. In experiments S1 and SIP1, similar instructions were used but the conflict diagram was replaced with the appropriate diagram from Figure 2.

In experiments $C3$ and $S2$, the two-tiered redemption values were not used. In these experiments there was a constant payment in cents per franc.

Instructions

This is an experiment in the economics of market decision making. The instructions are simple, and if you follow them carefully and make good decisions you might earn money which will be paid to you in cash.

In this experiment, we are going to conduct a computerized market over a sequence of trading periods. The items to be sold are called projects, and are designated by letters of the alphabet (project A, project B, project C, etc...). You may try to purchase any number of projects as you wish. The value to you of any particular project is detailed on your attached set of redemption value sheets. The redemption values vary from period to period and from person to person. You must pay careful attention to make sure you are using the correct period number sheet in evaluating which project(s) you wish to purchase. [note: the information on the redemption sheets is your own private information, do not reveal it to anyone.] At the end of each period, project(s) you have purchased are redeemed by the experimenter for the amounts indicated on these sheets.

Your trading profits in a period are determined by the difference in the redemption amount you receive for the projects you purchased and the amount you paid for them.

i.e. trading profit = (total project redemption value) – (total purchase price)

For example, if BUYER 43 purchases project *C* in the market for 500 and project *N* for 200 and her redemption value from her sheet is 750 for *C* and 300 for *N*, then BUYER 43's trading profit is

750 (value of *C*) $-$ 500 (payment for *C*) $+$ 300 (value of *N*) $-$200(payment for *N*) $=$ 350 (profit) .

Each project can be sold to one and only one buyer during each period. The projects are sold via an auction, carried out using the computer terminals. Buyers will have an opportunity to bid on each project as many times as they wish. To bid, follow the instructions at the bottom of the screen. Bids are not binding until the SEND key is hit. Bids which are lower than the current bid on the screen are ignored. Once a bid for a project is sent into the system, and becomes the current bid, the bidder is obligated to honor it until someone else bids higher on the same project, at which point the lower bid is deleted from the system.

There is an additional complication. Not all combinations of projects are possible. For example, it could be that if *X* is sold, that *Y* or *Z* cannot be sold. Incompatible groups of projects are detailed on an attached sheet.

The computer will accept proposals for which objects should be sold. The PROPOSERs earn profit for making proposals which are ACCEPTED for consideration. At the end of the period, the computer will use the best proposal submitted to determine which projects it will sell.

A PROPOSAL consists of a list of proposed objects to be sold by the computer. The computer allways keeps the current proposal on display. Projects included in the proposal are green on the display and items which were not included are red. At the beginning of the period, the current proposal is the proposal that none of the projects are sold.

A new proposal is ACCEPTED if there are no incompatible groups of projects in the proposal, and if the value of the new proposal given the current bids is higher than the value of the current proposal.

The proposer earns profit for ACCEPTED proposals. For each ACCEPTED proposal, a PROPOSAL BONUS is paid. Bonuses accumulate over the period.

PROPOSAL BONUS $=$ amount of improvement $\times$ bonus percentage.

The bonus percentage starts at 0% and rises as time left on the PERIOD TIMER decreases. When the timer indicates half the time left, the bonus will be 50%, and when the timer indicates 1 second left, the bonus will be close to 100%.

If you wish to make a proposal, type in the proposal by listing the object letters which should be accepted, then hit the [F1] key. You may propose any set of projects that you wish, but it will not be ACCEPTED unless it meets the criteria above (no conflicts, improves sum of bids). The period begins with the current proposal being to sell nothing.

It is important to emphasize the difference between Proposals and Bids. Remember that a PROPOSAL is a recommendation to the computer concerning which projects it should sell, given the BIDS already entered into the system. These bids might be your own, or they might be another BUYERs bids. Since projects are sold to the highest bidder, it is allways necessary to BID on projects which you are attempting to purchase. The projects you wish to purchase must also be in the best proposal received by the end of the period in order for you to actually purchase the projects. However, this proposal does not need to be made by the same person who is bidding on the projects.

At the beginning of each period, a PERIOD TIMER is set to _60_ seconds and is reset to this value whenever an acceptable bid or proposal is made. When the timer reaches 0, the period closes.

At the end of each period, the computer notifies each buyer of any successful bids. Successful bid(s) must be paid, and the bidder receives the indicated projects.

Unsuccessful bids are not displayed. Unsuccessful bidders pay nothing, and receive nothing.

At the end of the period, buyers should fill out their BUYER RECORD SHEET and calculate any profits (or losses) from the period. The total bonus from proposals is also displayed on the screen when the period closes, and this should be included in the profit calculation.

Currency:

The currency used in these markets is "francs". At the end of each period of the experiment francs will be converted to dollars. This will occur according to the following formula:

Losses: $0.02 * francs

1–10 francs: $0.20 * francs, or 5 francs = $1

10-infinity francs: $1.80 + 0.02 * francs.

Therefore, if you gain 10 francs either formula gives $2.00

Gain 100 francs, $3.80.

Gain 500 francs, $11.80.

If you lose 50 francs, then thats $1.

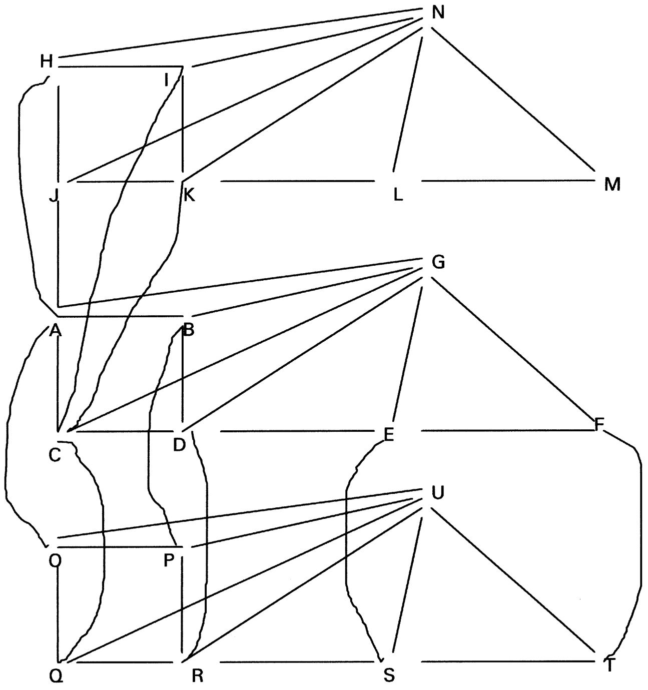
Remember, conversion to dollars occurs at the end of each PERIOD of the experiment.

Incompatible projects

Any proposal that contains an incompatible pair is not feasible.

The incompatible pairs of projects are shown via the following graph(s).

An incompatible pair of projects are directly joined by a line. For instance A, C is an incompatible pair because a line directly connects A and C, but A and E are compatible because there is not a line between A and E.



Examples:

$\{A, D, F\}$ is feasible since neither A, D, A, F or D, F are connected by lines in the figures above.
$\{B, D, F\}$ is not feasible since B, D is connected by a line.

References

Banks, J. S., Ledyard, J. O., Porter, D. P.: Allocating uncertain and unresponsive resources: An experimental approach. Rand Journal of Economics **20**(1), 1–25 (1989)

Brewer, P. J.: Allocation and computation in rail networks: A binary conflicts ascending price mechanism (BICAP) for the allocation of the right to use railroad tracks. Ph.D. Thesis. California Institute of Technology (1995)

Brewer, P. J., Plott, C. R.: A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. International Journal of Industrial Organization **14**, 857–886 (1996)

Cason, T. N., Friedman, D.: Price formation in double auction markets. Journal of Economic Dynamics and Control **20**, 1307–1337(1996)

Conlisk, J.: Why bounded rationality? Journal of Economic Literature **34**, 669–700 (1996)

Fisher, M. L.: Interactive optimization. Annals of Operation Research **5**, 541–556 (1986)

Forsythe, R., Nelson, F., Neumann, G. R., Wright, J.: Anatomy of a political stock market. American Economic Review **82**, 1142–1162 (1992)

Friedman, D.: On the efficiency of experimental double auction markets. American Economic Review **74**(1), 60–72 (1984)

Grether, D. M., Isaac, R. M., Plott, C. R.: The allocation of scarce resources: Experimental economics and the problem of allocating airport slots. Underground classics in economics. Boulder: Westview Press 1989

Groves, T., Ledyard J.: Incentive compatibility since 1972. In: Groves, T., Radner, R., Reiter, S. (eds.) Information, incentives, and economic mechanisms: Essays in Honor of Leonid Hurwicz, pp. 48–111. Minneapolis: University of Minnesota Press 1987

Hayek, Friedrich A.: The use of knowledge in society. American Economic Review **35**(4), 519–630 (1945)

Huberman, B., Hogg, T.: Distributed computation as an economic system. Journal of Economic Perspectives **9**(1), 141–152 (1995)

Huberman, B., Lukose, R. M., Hogg, T.: An economics approach to hard computational problems. Science **275**, 51–54 (1997)

Hurwicz, L.: On informationally decentralized systems. In: McGuire, C.B., Radner, R. (eds.) Decision and organization, pp. 297–336. Amsterdam: North-Holland 1972

Ledyard, J. O.: Coordination in shared facilities: A new methodology. Journal of Organizational Computing **1**(1), 41–59 (1991)

Ledyard, J. O.: The design of coordination mechanisms and organizational computing. Journal of Organizational Computing **3**(1), 121–134 (1993)

Ledyard, J. O., Porter, D. P., Rangel, A.: Using computational exchange systems to solve an allocation problem in project management. Journal of Organizational Computing (forthcoming)

McCabe, K. A., Rassenti, S. J., Smith, V. L.: An experimental mechanism for the simultaneous auction of gas and pipeline rights in a transmission network. Report prepared for the Federal Energy Regulatory Commission, Economic Science Laboratory, University of Arizona (1987)

McCabe, K. A., Rassenti, S. J., Smith, V. L.: A new market institution for the exchange of composite goods. Discussion paper 88–13, University of Arizona (1988)

McCabe, K. A., Rassenti, S. J., Smith, V. L.: Designing 'smart computer' assisted markets: an experimental auction for gas networks. European Journal of Political Economy **5**, 259–283 (1989)

Nilsson, J. -E.: Konkurrens pa sparen. Effektivite pa en avreglerad jarnvagstrafik. Swedish Transport Research Board (1991)

Nilsson, J. -E.: Regulatory reform in swedish railways: Policy review with emphasis on track allocation issues. Presented at the third international conference on competition and ownership in surface passenger transport, Missisauga, Ontario (1993)

Nilsson, J. -E.: Personal communications with P. Brewer and C. R. Plott (1994)

Olson, M., Porter, D.: An experimental examination into the design of decentralized methods to solve the assignment problem with and without money. Economic Theory **4**, 11–40 (1994)

Page, S. E.: Two measures of difficulty. Economic Theory **8**, 321–346 (1996)

Plott, C. R.: Market architectures, institutional landscapes and testbed experiments. Economic Theory **4**(1), 3–10 (1994)

Plott, C. R., Porter, D. P.: An experiment with space station pricing policies. Social Science Working Paper 704, California Institute of Technology (1990)

Plott, C. R., Porter, D. P.: Market architectures and institutional testbedding: An experiment with space station pricing policies. Journal of Economic Behavior and Organization **31**, 237–272 (1996)

Plott, C. R., Sunder, S.: Efficiency of experimental security markets with insider information: an application of rational-expectations models. Journal of Political Economy **90**, 663–698 (1982)

Plott, C. R., Sunder, S.: Rational expectations and the aggregation of diverse information in laboratory security markets. Econometrica **56**(5), 1085–1118 (1988)

Rassenti, S. J., Reynolds, S. S., Smith, V. L.: Cotenancy and competition in an experimental auction market for natural gas pipeline networks. Economic Theory **4**(1), 41–65 (1994)

Rassenti, S. J., Smith, V. L., Bulfin, R. L.: A combinatorial auction mechanism for airport time slot allocation. Bell Journal of Economics **13**, 402–417 (1982)

Rothkopf, M. H., Pekec, A., Harstad, R. M.: Computationally manageable combinatorial auctions rutcor research report 13–95 (1995)

Rubinstein, A.: Why are certain properties of binary relations relatively more common in natural language? Econometrica **64**, 343–355 (1996)

Smith, V. L.: An experimental study of competitive market behavior. Journal of Political Economy **70**, 111–137 (1962)

Wilson, R.: Design of efficient trading procedures. In: Friedman, D., Rust, J. (eds.) The double auction market. Santa Fe Institute Studies in the Sciences of Complexity Proceedings Volume XIV. New York: Addison-Wesley 1993