Instructional Manual for Burden Sharing and Public Goods Investments in Policy Reform

Glenn W. Harrison and Leo K. Simon

GATT Research Paper 90-GATT 29

Center for Agricultural and Rural Development
Iowa State University
Ames, Iowa 50011

Glenn W. Harrison is with the department of economics, College of Business Administration, University of South Carolina, Columbia. Leo K. Simon is with the department of agricultural and resource economics at the University of California, Berkeley.

This material is based upon work supported by the Cooperative State Research Service, U.S. Department of Agriculture, under Agreement No. 89-38812-4480. Any opinion, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessary reflect the view of the U.S. Department of Agriculture.

1. INTRODUCTION

This paper describes the basic laboratory experimental design that has been developed to assess a new Multilateral Bargaining institution that may be used for trade negotiations. Our overall objective is to implement a series of laboratory experiments that tests the ability of this institutions to resolve trade disputes.

In Section 2 we provide a brief overview of the new institution that is being studied; details may be found in Rausser and Simon [1991a] [1991b]. In Section 3 we discuss several aspects of our experimental design, especially the careful wording required in our instructions to subjects. In Section 4 we document how we have been able to solve the model numerically, using flexible and powerful software that will allow much richer models to be solved. Finally, in Section 5 we document the use of computer software that has been developed to actually conduct the laboratory experiments.

2. THE MULTILATERAL BARGAINING INSTITUTION

The MB institution can be characterized by a model of noncooperative multilateral bargaining with a central player. The model has n + 1 players, called the player set. The zero'th player is distinguished from the others and is called the central player. Players 1 through n are peripheral players. The players participate in a sequential, multilateral bargaining game, similar in spirit to Rubinstein's classic [1982] bilateral game. Their objective in bargaining is to form a coalition, which is just a subset of the player set, and to choose an m-dimensional vector from a set of feasible vectors, called the choice set and assumed to be compact. The choice set may be different for different coalitions. The central player is distinguished from the others in that she must be included in every coalition. Each player has a utility function defined on the choice set. We assume that utility functions are continuous and strictly quasi-concave "in the variables that count".[1]

Problems of this kind are typically formulated as cooperative games. Cooperative game theorists specify some solution concept that satisfies certain appealing properties and then study the set of choices that satisfy the given criterion. Perhaps the most

[1] Player i may care only about certain components of the m-dimensional vector. For example, if x is a vector of shares of a two-dimensional pie, then i may care only about his own share. In this case, i's utility must be strictly quasi-concave in the two components of x that represent i's share of the pie.

familiar cooperative solution concept is the Core. In the context of the MB institution, a vector x is in the Core if it is feasible for some coalition and if, for every coalition C, there is no feasible vector that is weakly preferred to x by each member of C and strictly preferred by one member.

Noncooperative bargaining theory differs from cooperative game theory in that it attempts to model the actual process of negotiation, rather than just the outcome of the negotiation. A noncooperative model of multilateral bargaining includes an extensive form, which stipulates a particular set of negotiating rules that players must follow.

A natural research program, referred to as the "Nash Program" after Nash [1953], is to study the cooperative and noncooperative versions of a game in conjunction with each other. First one studies a particular cooperative solution concept, then one asks whether the equilibria (usually the subgame perfect equilibria) of some noncooperative model implement the cooperative solutions. Following this approach, we study the relationship between the Core of various bargaining games and the subgame perfect equilibria of our noncooperative version of these games.

2.1 The Noncooperative Model

The game has a finite number of periods T, each of which is divided into three sub-periods. In the first sub-period a player is chosen by Nature to be the proposer. Nature makes it's choice according to a probability distribution over the player set that is

prespecified as past of the description of the game. In the second sub-period the proposer announces a coalition, of which he must be a member, and a vector that is feasible for that coalition. In the third sub-period the remaining members of the proposed coalition each choose whether to accept or reject the proposed vector. If all accept, the game ends. If not, the next period begins and a new proposer is selected. If agreement is not reached by the T'th period then players receive a predetermined disagreement payoff.

Note that the game is well specified whether or not there is a central player. As we shall see, however, the presence of the central player guarantees that the model has a solution. Nonetheless, it can be instructive to compare our model to the corresponding one in which the central player is excluded (see section 2.3).

A strategy for player i specifies the vector that he will announce in each period if selected to be the proposer, as well as a set of vectors that i will accept in each period if he is a member of a coalition announced by some other proposer. A strategy profile is a list of strategies, one for each player. Each strategy profile defines an outcome for the game, which is just a function assigning to each element of the choice set the probability that the game will end with an agreement to select this vector. Note that only a finite number of these probabilities will be positive. Moreover, these positive probabilities need not sum to unity, since the players may never reach an agreement.

A subgame perfect equilibrium for a game is a strategy profile

with the property that at every sub-period of the game each player's choice is optimal given the strategies specified by the other players. Every T-period game has a subgame perfect equilibrium. Moreover, this equilibrium is generically unique. A striking feature of the model is that there are equilibria in which players fail to agree until the final rounds of bargaining. An equilibrium outcome is the outcome defined by a subgame perfect equilibrium. Note that since agents may fail to agree at the beginning of the game, the equilibrium outcome need not coincide with the distribution over first period proposals.

We will be concerned with the equilibrium outcomes of games with an arbitrarily large number of periods. Accordingly, our bargaining model is defined as a sequence of T-period bargaining games, with T growing to infinity. A solution to our bargaining model is a limit of the equilibrium outcomes of the T-period games.

The first main result for our model is that a solution exists. That is, the outcomes for the T-period games always converge as T grows large. It is here that the central player has a crucial role: when there is no player that is a member of every coalition, T-period outcomes will not in general converge.

A second main result is that, generically, this solution is deterministic. More precisely, there is generically a unique vector x with the property that for every ϵ there exists a T sufficiently large that the agreed upon vector in any game with more than T periods is within ϵ of x with probability one. When such a vector x exists we will refer to it as the solution vector.

Our final main result is that the solution vector is always in the Core of the corresponding cooperative game.

3. EXPERIMENTAL DESIGN

A general feature of our experimental design is a mixture of positive and normative objectives. To some extent we are interested in how well self-interested actors perform in the new institution (a "positive" concern), but we are also interested in finding out what it takes by way of training and real-time computer assistance to get actors to behave rationally in the institution (a "normative" concern). It is important to be clear at the outset that we have several objectives in our design of experimental treatments, the wording of our Instructions, and the flexibility of our computer software.

3.1 Planned Treatments

In the first series of experiments it is appropriate to examine a simple environment in which we have human players take on the roles of each of the private agents. This is a natural place to begin an experimental evaluation of the MB institution, although our software allows much more complicated designs (such as computer-simulating one or more players in a session).

We propose several experimental features that will be common to all of our sessions. In principle these could be variable treatments, but we do not believe them to be sufficiently important as to warrant variation.

The first common treatment is to provide subjects with **unpaid training** in the institution being implemented. This training would

take the form of playing against simulated opponents, including the Government if need be (i.e., if this subject is not to play the role of the Government). In all essential respects this training would provide "hands-on" and self-paced instruction in the operation of the institution.

The second common treatment is the method of paying subjects. We use a **duplicate cardinal tournament** method in which agents receive a share of a fixed pie based on their performance relative to others in a similar situation. There are two major reasons for wanting to consider alternatives to the standard method. The tournament method provides enhanced incentives for all subjects to behave rationally even "down to the last penny", thereby avoiding payoff dominance problems that Harrison [1991a] [1991b] argues have plagued previous experiments in this area. The second reason is that it enables us to easily define a fixed sum of money to be paid out per session, enhancing our ability to plan a complete series of experiments that meet some overall budget constraint and ensure comparable incentives for all players (the total payoff under the direct method is endogenous to any particular set of parameters, although one could re-scale direct earnings to match some overall pot per session).

The third common treatment is the **number of private agents bargaining** in any game. We propose initial benchmark games of five agents, simply to focus on the role of various treatments in the simplest interesting setting that has been widely studied in the political science literature (see Harrison [1991b] for a review of

that literature). Our ultimate focus is on the role of the institutions when we have "many" bargainers, but it would be inefficient to focus on the large numbers case initially.

The first variable treatment we plan to evaluate is **experience** with the institution. By this we mean paid experience, as distinct from the unpaid training that all subjects will receive. Moreover, the unpaid training would be against simulated opponents, whereas experience as we define it here would involve participating twice in the same series. There is evidence from many fields of experimental economics that institution-specific experience is an important treatment variable.

The second variable treatment is the **availability of computer assistance** during the experiment. We anticipate being able to provide a user-friendly interface to some software that has been developed to allow subjects to gain some estimate of the likely outcomes for various proposals that they might make. In the simplest form, this could just involve subjects being able to see how rational computer-simulated opponents would respond to any given proposal, and what the payoffs would be for the subject of alternative responses. The major advantage of this treatment is that it would address an important doubt that one might have as to the limited computing abilities of student subjects in laboratory settings. This doubt is often expressed as a suspicion that lab institutions would be subject to strategic manipulation in the field if the stakes ever became high enough to warrant the use of skilled strategists or computers to determine optimal responses.

The final variable treatment is allowance for **face-to-face discussions** between participants. There is ample evidence from previous bargaining and committee experiments that such discussions can greatly facilitate agreements in bargaining situations (see Harrison [1991a] [1991b]). We propose that agents would be allowed a certain fixed period of time for discussions every period, and then be asked to return to their terminals to complete the round. The most interesting sequence would be to allow discussion after the proposer is selected, before he or she has nominated a coalition and before the policy variables have been selected. This is closest to our image of "backroom negotiations": an initiator picks a select group and attempt to work out a fait accompli with them.

3.2 Instructions

The experimental instructions that have been developed to implement the MB experiments are the result of many iterations, often involving pilot experiments. They are as follows:

1. Introduction

Please read through these instructions before you begin. You will have ample opportunity to practice and ask questions before we begin playing for money. If you follow the instructions carefully and make good decisions, you might earn a considerable amount of money. The experiment should take between one and a half and two hours. There is no advantage in rushing to finish sooner. You will not receive your cash payment until the end of the session. You have already earned three dollars just for showing up as agreed.

2. Ranking the Alternatives

You are about to participate in a group decision making

experiment in which one alternative will be chosen from numerous competing alternatives. Each alternative will be represented by two numbers, a horizontal coordinate and a vertical coordinate. The group consists of five imaginary women, named Lisa, Paula, Jenny, Christine and Stephanie. You will be asked to play the role of one of these women. Each woman ranks the alternatives in different ways. Each has a "favorite" alternative, and prefers points closer to that alternative to points further away. If a particular alternative is chosen by the group, then each woman earns points: the number of points is greater, the more she likes the alternative. In our practice example (Experiment 0), Lisa's favorite alternative is "0,0" (i.e., the horizontal and vertical coordinates are both zero). If "0,0" is chosen by the group, Lisa will earn 100 points.

3. The Decision Making Process

Our five women set aside several days to meet in Fresno in order to make a group decision. At the beginning of the first day, one of the women will be chosen randomly to be the PROPOSER. The choice is made by the computer. Some of the women are chosen with a greater frequency than others. You will be informed of the frequencies that apply in today's experiment.

The proposer does two things: she invites two or more of the other women to join her in a COALITION, and she proposes a POLICY ALTERNATIVE to this coalition. For example, the computer might pick Stephanie at random to be the proposer. Stephanie might then pick a coalition consisting of herself, Paula and Jenny, and propose the alternative "1.6, 1.3," i.e., a horizontal coordinate of 1.6 and a vertical coordinate of 1.3.

Next, each of the women who has been invited to join the coalition decides whether to accept or reject the proposed alternative. If each member of the coalition accepts the proposal, then the group decision will have been made, and the women return to their homes that night. In that case, each woman, whether or not they were included in the coalition, is assigned the points corresponding to the proposed alternative. In our example above, if both Paula and Jenny accept the alternative (1.6, 1.3) then the process ends and the computer calculates the points corresponding to that alternative for each of the five women.

If any member of the coalition rejects the proposal the women will stay overnight in Fresno and reconvene the next day, when another woman will be chosen randomly as the proposer. The process described above will be repeated again. And so on. If agreement has not been reached by the end of the last day set aside for meeting, the women return to their homes and each will earn zero points.

4. Earning points

You have been given a set of graphs. The first graph indicates each woman's favorite alternative. The others show in more detail how many points each woman earns from various alternatives. To find out exactly how many points you will earn from any given proposal, you can use your computer. In fact, your computer will tell you how many points each of the women earns from the alternative. To practice using this computer assistance, answer YES when the computer asks if you want to evaluate payoffs. Then computer will prompt you for a "horizontal coordinate" and a "vertical coordinate." Type in the numbers representing the alternative you have chosen. You will have access to this kind of assistance during the experiment. This will be useful in helping you assess how each woman is affected by the different policy values.

5. Description of the Experiment

The experiment consists of an interactive session in which you instruct the computer how to play the game on your behalf, followed by a computer simulation using the instructions you have provided. Instructions for the other women have been specified either by other participants in this experiment or by the computer.

During the interactive session we will ask you to tell us what coalition and alternative you would specify if the computer selected you as the proposer in the first day of the game. We will also ask you to indicate the minimum payoff that you would accept if you were invited by somebody else to join a coalition. We will then ask you to specify the same information for the second and subsequent days of the game.

In more detail, we will ask you to make three choices in every day of the interactive decision making process:

(1) Indicate a list of women that you would invite to join you in a coalition if you were chosen to be the proposer. (You must choose at least two other women besides yourself)

(2) Indicate an alternative (i.e., a pair of numbers) that you would propose to your coalition if you were chosen to be the proposer.

(3) Indicate the minimum payoff that you would be willing to accept in order to vote for an alternative. For example, suppose the minimum payoff you select is 45 points. This means that if some other player invites you to be in her coalition and proposes an alternative, you will be willing to accept it if and only if it earns you a payoff of at least 45 points.

The computer simulation phase follows the interactive session. We will "play out" the group decision making process a large number of times on the computer. The computer will make decisions on your behalf, using the instructions you have specified during the interactive session. When making decisions on behalf of the other women, the computer will either follow instructions that have been specified by other participants in the experiment or it will follow its own set of rules for playing the game. You will be informed by the experimenter whether the other women in today's experiment are being represented by the computer or by other participants.

We will simulate several different scenarios. We first assume that the women all arrive in Fresno as planned. The computer begins by choosing a woman at random and implementing the instructions she has specified for DAY 1 of the meeting. If her proposal is rejected by a coalition member the game proceeds to DAY 2. If there is rejection again, we proceed to DAY 3 and continue this process as described in Section 3 above.

For the next simulation we imagine that, because of an earthquake, everybody's trip to Fresno is delayed until the SECOND day of the scheduled meeting. In this scenario, the computer again chooses a woman at random, but this time it will begin with the instructions specified for DAY 2 of the meeting. We then imagine that the trip to Fresno is postponed for two days, beginning on DAY 3, etc. For example, if the women have set aside FIVE days for the meeting, we shall run five different simulations, beginning successively on DAY 1, DAY 2, DAY 3, DAY 4 and DAY 5.

Note that your performance may be quite different depending on which day the simulation starts. To see why, suppose that in DAY 1 each woman selects a proposal that is certain to be accepted. If in this case the simulation begins on DAY 1, the women will all return home at the end of the first day, and the instructions that have been prepared for DAY 2 will never be implemented. When the simulation begins on DAY 2, however, the instructions for DAY 2 will be implemented, and may lead to a quite different outcome.

Note that whenever the computer picks a different proposer the outcome and hence possibly your payoff will be different. We will be concerned with the AVERAGE of your payoffs in each simulation. Remember that for each simulation the computer "plays out" the group decision making process a large number of times. Your FINAL payoff for the experiment will be the SUM of your average payoffs in each of the simulations beginning on a different day of the proposed meeting. For example, if five days were set aside for the meeting, your final payoff will be the sum of your average payoffs in the simulations that begin in each of the five days. The amount of money you earn in the session will depend on the relative size of your FINAL payoff.

6. Turning Points into Money

You will receive $7 just for completing the session, which includes the $3 payment for showing up. The remaining money allocated to this session ($11 per person) will be divided between participants according to performance. You will be compared only to other participants who are in the same "class" as you, i.e., that represent the same woman. We use the following formula to divide up the money.

(1) At the end of the game, after you have participated in a number of experiments, we will add up all of the average payoffs you have earned during the session. This will be your score for the session.

(2) We subtract from your score the lowest score earned by anybody in your player class. This is your adjusted score. The fraction of the total pot that you receive is the ratio of your adjusted score to the sum of the adjusted scores of all players of the same type.

For example, suppose that three people all play Lisa and they earn, respectively, 500, 550, and 610 points. The lowest score is 500, so we subtract 500 from every player. The adjusted scores are 0, 50, and 110, which sum to 160. The three players receive $7 plus, respectively, 0/160, 50/160 and 110/160 of $33, i.e., a total of $7, $17, and $30. (If necessary, we will round up to the nearest dollar.)

YOU SHOULD NOTE THAT THE DIFFERENCE BETWEEN PARTICIPANTS' EARNINGS MAY BE CONSIDERABLE, EVEN IF THE DIFFERENCE BETWEEN THEIR FINAL PAYOFFS IS VERY SMALL. IN PREVIOUS EXPERIMENTS THERE HAVE BEEN PARTICIPANTS WHO HAVE EARNED MORE THAN $40.

We will now play a number of practice games so that you can get used to the rules. Please feel free to ask any questions, and to make as much money as possible. Good Luck!

The Figures referred to in these Instructions are reproduced at the end of this paper, and refer to the "training" environment we developed to give all subjects some initial experience with the institution.

4. SOLVING THE MULTILATERAL BARGAINING GAME

It is not possible to solve the MB game analytically for any interesting classes of preferences. We have therefore developed some computer software to compute the full solution for wide classes of games. Details on the use of this software are discussed separately in the next section. It is worthwhile, however, discussing the use of the non-linear programming package GAMS to solve these problems (see Brooke, Kendrick and Meeraus [1988] for documentation on GAMS).

One of the programs developed by us and discussed later, SOLVE, generates a number of GAMS problems which represent the problem faced by a proposer in our game. Specifically, this proposer must find the policy values for any given coalition that maximizes his or her utility while ensuring that all other agents who are in the coalition would be willing to vote for the proposal. We solve this problem for all possible coalitions and all agents in any period. The GAMS problem is "self-documenting" and is as follows:

```
$TITLE MULTILATERAL BARGAINING GAME

$OFFUPPER
$OFFSYMXREF

SETS
   I  Agents             / Gordon
                           Lisa
                           Jenny
                           Paula
                           Christine
                           Stephanie  /
   J  Policy Dimensions  / Horizontal
```

```
                          Vertical  /
   C  Coalitions          / C1*C16 /
   T  Periods             / T0*T0 / ;

ALIAS   (I,K), (I,II), (C,CC), (T,TT)  ;

* Define the ideal points (or bliss points) of each agent, around which
* their indifference curves over policies will be defined.

PARAMETER  A(I,J)   Ideal Points of Agents
   /      Gordon    .Horizontal                    0
          Gordon    .Vertical                      0
          Lisa      .Horizontal                    39
          Lisa      .Vertical                      68
          Jenny     .Horizontal                    30
          Jenny     .Vertical                      52
          Paula     .Horizontal                    25
          Paula     .Vertical                      72
          Christine.Horizontal                     62
          Christine.Vertical                       109
          Stephanie.Horizontal                     165
          Stephanie.Vertical                       32    /  ;

* Define the intercept of the utility functions of agents.

PARAMETER  INTERCEPT(I)   Intercept of Utility Functions
   /      Gordon                                   0
          Lisa                                     90
          Jenny                                    70
          Paula                                    70
          Christine                                90
          Stephanie                                110     /  ;

* Define the coefficient of the utility functions of agents.

PARAMETER  COEFF(I)   Coefficient of Utility Functions
   /      Gordon                                   0
          Lisa                                     1
          Jenny                                    1
          Paula                                    1
          Christine                                1
          Stephanie                                1     /  ;

* Define the access weights of each agent.

PARAMETER  ACCESS(I)   Access weights
   /      Gordon                                   0
          Lisa                                     12
          Jenny                                    1
          Paula                                    1
          Christine                                1
          Stephanie                                1     /  ;
```

```
* Define the default policy values.

PARAMETER  DEFAULT(J)   Default policies
   /     Horizontal                              0
         Vertical                                0    /  ;

* Define the default utility levels.

PARAMETER  UDINPUT(I)   Default utility levels as input
   /     Gordon                                  0
         Lisa                                    0
         Jenny                                   0
         Paula                                   0
         Christine                               0
         Stephanie                               0    /  ;

* Define the possible coalitions.

PARAMETER  COALITIONS(C,I)   Feasible Coalitions
/ (C1*C16) . Gordon      1
(C1, C2, C3, C5, C7, C8, C9, C12, C14, C15, C16) . Lisa       1
(C1, C2, C4, C5, C6, C8, C11, C13, C14, C15, C16) . Jenny      1
(C1, C2, C3, C6, C7, C10, C11, C12, C13, C14, C15) . Paula     1
(C1, C4, C5, C6, C7, C9, C10, C12, C13, C15, C16) . Christine  1
(C1, C3, C4, C8, C9, C10, C11, C12, C13, C14, C16) . Stephanie  1  / ;


* This file contains the essential problem logic for the MB problem. It will
* be included with a generating file which contains the specific parametric
* instance to be solved.

* The scalar SIGMA defines the substitutability of agents in the governments
* utility function.

* The scalar SELECTG will indicate if we are picking out the government (=1)
* or not (=0).

* The scalar SACCESS is used to hold the sum of the access weights.

* The scalar IBEGIN indicates (if > 0) that we begin the procedure by
* directly stating default utility. Otherwise (if < 0) we start by
* defining default policy values and derive default utility. This indicator
* in turn defines values for BIGT and NOTBIGT.

* The next two scalars will provide a switch for the evaluation of default
* utilities. When we are in the last period (BIGT=1 and NOTBIGT=0) we use
* the default policy options to evaluate default utility. Otherwise we are
* in an earlier stage of the game (BIGT=0 and NOTBIGT=1) and we use the
* expected utility from next period's proposals. This may be clearer below
* when you look at the UDEFAULT equations.
```

```
SCALARS
      SIGMA    SUBSTITUTABILITY OF AGENTS IN GOVERNMENT UTILITY     /2.0/
      SELECTG  INDICATOR THAT WE ARE SELECTING THE GOVERNMENT AGENT /1.0/
      SACCESS  SUM OF THE ACCESS WEIGHTS                            /0.0/
      IBEGIN   INDICATOR THAT WE DEFINE DEFAULT UTILITY DIRECTLY    /1.0/
      BIGT     INDICATOR THAT WE ARE IN THE TERMINAL PERIOD
      NOTBIGT  INDICATOR THAT WE ARE NOT IN THE TERMINAL PERIOD     ;

BIGT $ (IBEGIN GT 0) = 0.0     ;
BIGT $ (IBEGIN LT 0) = 1.0     ;
NOTBIGT $ (IBEGIN GT 0 ) = 1.0 ;
NOTBIGT $ (IBEGIN LT 0 ) = 0.0 ;

* Re-normalize the access probabilities to sum to one.

SACCESS = SUM(I, ACCESS(I))    ;
ACCESS(I) = ACCESS(I) / SACCESS;

* These arrays will facilitate the looping as well as the solution report.

PARAMETERS
     UNEXT(I)          RESERVATION UTILITY FOR AGENT IN NEXT PERIOD
     SELECTI(II)       WEIGHTS TO SELECT AGENTS
     SELECTC(CC)       WEIGHTS TO SELECT COALITIONS
     UREP(CC,TT,II,K)  OPTIMAL UTILITY LEVELS FOR EACH COALITION
     XREP(CC,TT,J,K)   OPTIMAL POLICY PROPOSALS FOR EACH COALITION
     UDREP(TT,K)       RESERVATION UTILITY OF AGENTS
     CHOOSE(TT,II)     UTILITY IN COALITION CHOSEN BY COLUMN AGENT
     BESTC(TT,I,II)    UTILITY OF ROW AGENT IN PROPOSAL BY COLUMN AGENT;

* Initialize UNEXT() at values for UDINPUT. To be re-initialized as time
* goes by...

UNEXT(I) = UDINPUT(I) ;

* Define the variables used to construct the problem.

VARIABLES
          GU           GOVERNMENT UTILITY
          GUDEF        DEFAULT GOVERNMENT UTILITY
          U(I)         UTILITY OF AGENT I
          UDEF(I)      DEFAULT UTILITY OF AGENT I
          X(J)         POLICY PROPOSALS
          OBJ          OBJECTIVE FUNCTION (UTILITY OF PROPOSER);

* Define each of the equations of the problem.

EQUATIONS
          GOVT          DEFINE UTILITY FUNCTION OF THE GOVERNMENT
          GOVTDEF       DEFINE DEFAULT UTILITY OF THE GOVERNMENT
          UTILITY(I)    DEFINE UTILITY FUNCTION OF AGENT I
          UDEFAULT(I)   DEFINE DEFAULT UTILITY OF AGENT I
```

```
            PROPOSE        DEFINE UTILITY OF PROPOSER AS OBJECTIVE
            UVOTERS(C,I)   ENSURE UTILITY OF VOTERS EXCEEDS DEFAULT
            VETO           ENSURE GOVERNMENT VETO POWER;

* Define the government's utility functions as a CES function of the
* utility of all agents. Define the government's default utility in a
* similar fashion. In this version we will simplify things by just assuming
* perfect substitutability between individual agent utilities. Similarly
* for the GOVTDEF definition below.

GOVT..

*  GU     -E- SUM(K $ (ORD(K) NE 1),
*                (U(K)     ** ((SIGMA-1)/SIGMA))) ** (SIGMA/(SIGMA-1));
   GU     -E- SUM(K $ (ORD(K) NE 1), U(K));

GOVTDEF..

*  GUDEF -E- SUM(K $ (ORD(K) NE 1),
*                (UDEF(K)     ** ((SIGMA-1)/SIGMA))) ** (SIGMA/(SIGMA-1));
   GUDEF -E- SUM(K $ (ORD(K) NE 1), UDEF(K));

* This is the Euclidean distance metric being used to define the utility of
* each agent as we move away from his ideal point A(,). We also get the
* government utility "defined" here, since we use U() for deciding on the
* best proposals as well as the reservation utilities.

UTILITY(I) $ (ORD(I) NE 1)..

      U(I) -E- INTERCEPT(I) - COEFF(I) * SQRT( SUM(J,
                              ( (X(J) - A(I,J)) * (X(J) - A(I,J))  ) ) );

* Use the foregoing utility functions to evaluate the default utility level.
* In the final period (when BIGT=1 and NOTBIGT=0) we use the default policy
* options to define default utility. Otherwise (when BIGT=0 and NOTBIGT=1)
* we use the expected value of utility next period which was solved for.

UDEFAULT(I) $ (ORD(I) NE 1)..

  UDEF(I) -E- BIGT * ( INTERCEPT(I) - COEFF(I) * SQRT( SUM(J,
                       ((DEFAULT(J) - A(I,J)) * (DEFAULT(J) - A(I,J)) ))))
            + NOTBIGT * UNEXT(I) ;

* The next set of constraints ensure that each voter in active coalition C
* gets more utility than his default, but weight each by (a) whether or not
* the voter is in the coalition (COALITIONS()=1), and (b) whether
* or not this coalition is being considered just now (SELECT()=1).
* Note that the government is not included here.

UVOTERS(C,I) $ (ORD(I) NE 1)..

      U(I)    * SELECTC(C) * COALITIONS(C,I) -G-
```

```
      UDEF(I) * SELECTC(C) * COALITIONS(C,I);

* Let the government have veto power.

VETO..

      GU =G= GUDEF  ;

* This is the objective function, which will depend on the agent
* making the proposal (picked out by SELECT() as we loop over II, which is
* aliased with I, below).

PROPOSE..

      OBJ =E= (1.0 - SELECTG) * SUM(I $ (ORD(I) NE 1), SELECTI(I) * U(I) )
                  + SELECTG  * GU;

* Define the model.

MODEL BARG / ALL /      ;

* Initialize the pointer arrays for agents and committees at zero.

SELECTI(I) = 0.0        ;
SELECTC(C) = 0.0        ;

* Solve the model, looping over all time periods TT, agents II and coalitions
* CC. This is a conservative solution approach which will ensure that we
* have found the best coalition.

LOOP (TT,

         LOOP (II $ (ACCESS(II) GT 0.0),

               SELECTI(II) $ (ORD(II) NE 1) = 1.0;
               SELECTG     $ (ORD(II) EQ 1) = 1.0;
               SELECTG     $ (ORD(II) NE 1) = 0.0;

               LOOP (CC $ (COALITIONS(CC,II) EQ 1),

                       SELECTC(CC) = 1.0;

                       X.L(J)   $ (ORD(II) GT 1) = A(II,J)        ;
*                       X.LO(J) = 0.0             ;

                       SOLVE BARG USING NLP MAXIMIZING OBJ;

* If the model solves then save the solution...

                       U.L(I)   $ (ORD(I) EQ 1) = GU.L;

                       UREP(CC,TT,I,II)  $ ((BARG.MODELSTAT EQ 2) OR
```

```
                                        (BARG.MODELSTAT EQ 7))
                                       - U.L(I);
                        XREP(CC,TT,J,II)  $ ((BARG.MODELSTAT EQ 2) OR
                                        (BARG.MODELSTAT EQ 7))
                                       - X.L(J);

* ... but if it does not solve then set the values to the expected
* utility of going into the next period (i.e., passing). This will happen
* as we approach a solution of the overall multiperiod game, so it is
* important not to "abort" at this stage. The following "abort" code is
* remarked out but is useful for debugging purposes. Note that not being
* able to find a solution means that the agent and coalition being considered
* in this loop cannot find a proposal that would be voted in.
*
*                        ABORT $ ((BARG.MODELSTAT NE 2) AND
*                                 (BARG.MODELSTAT NE 7))
*                               "**** THE MODEL DID NOT SOLVE";

                        UREP(CC,TT,I,II)  $ ((BARG.MODELSTAT NE 2) AND
                                        (BARG.MODELSTAT NE 7))
                                       - UNEXT(I);
                        XREP(CC,TT,J,II)  $ ((BARG.MODELSTAT NE 2) AND
                                        (BARG.MODELSTAT NE 7))
                                       - 0.0;

                        SELECTC(CC) - 0.0                    );

* Now find the best coalition for this proposer.

                CHOOSE(TT,II)  - SMAX((K,CC), SELECTI(II) * UREP(CC,TT,II,K)
                                            + SELECTG     * UREP(CC,TT,II,K));

* This next line is not correct ... it picks out the best values for each
* agent, rather than picking out the best coalition from the perspective
* of the proposing agent. Have the program SOLVE read in the entire set of
* results and do this itself .... FIX later!

                BESTC(TT,I,II) - SMAX((K,CC), UREP(CC,TT,I,II) $
                                      (CHOOSE(TT,II) GT 0.0) );
                SELECTI(II) - 0.0       );

          BIGT - 0.0;
          NOTBIGT - 1.0;
          UNEXT(I) - SUM(K, (ACCESS(K) * BESTC(TT,I,K)) );
          UDREP(TT,I) - UNEXT(I); ) ;

DISPLAY COALITIONS, ACCESS, BESTC, CHOOSE, UDREP, UREP, XREP;
```

The program SOLVE essentially controls the sequence of such

GAMS problems that must be solved. First it solves the series of problems for each agent and each coalition for the terminal bargaining round. Then, using the expected payoffs for each agent from the last round, it can set up the problems for the next-to-last round (since we know the "reservation payoff" that each agent must receive in order to accept a proposal rather than force play into the terminal round). It does this until we have solved the game as specified in the "CNF" file discussed later, or until some convergence tolerance has been attained.

The output of the program SOLVE is called a "report file", and has the suffix "REP". We go through two of these report files in companion studies (see Harrison and Simon [1990] [1991]). The report tells us what coalition and policy proposals each agent would make in each round of negotiations. It also tells us what the payoffs would be to each agent from these proposals, as well as the expected payoff to each agent in each round. A related output file, called a "detail file" and having the suffix "DET", tells us the same information for all possible coalition proposals. Thus we will be able to see if human subjects can propose optimal policies even if they do not select the very best coalition. The detail file has the same structure as the report file.

5. COMPUTER SOFTWARE

A number of computer programs have been developed to conduct the laboratory experiments. These programs have been designed to facilitate the evaluation of a wide number of alternative treatments in a range of economic environments. Although several features of our experiments have been "hard-wired" into the software, by and large the user interacts with the programs by means of a single ASCII file defining the type of experiment to be conducted. In this section we outline the sequence of programs to be used (section 5.1) and the way in which the user instructs the computer system to run a particular experiment (section 5.2).

5.1 The Computer Programs

There are five programs that are used to conduct and evaluate an experiment: (i) SOLVE, which solves the game assuming rational players, (ii) INIT, which initialize the experiment; (iii) MAIN, which monitors the actual experiment and keeps track of what each individual is doing at all times; (iv) IND, which presents each individual subject with instructions and all messages during the experiment; and (v) OUTPUT, which generates and processes the raw output of the experiment in a (reasonably) transparent manner.

In brief, one can think of MAIN and IND as the two programs that actually conduct the experiment. MAIN is akin to the traffic policeman that makes sure that all of the IND programs obey certain rules when talking to each other using the computer network. It

also keeps all of the IND programs up to speed, prompting them for messages when they are not responding. Finally, MAIN keeps the experimenter appraised of the "real-time" progress of the experiment, in case there is some need for experimenter intervention (e.g., if one subject falls asleep, this will become apparent to the experimenter watching MAIN, even if he in turn is asleep ... there are many loud "beeps" to alert the experimenter to possible problems).

IND is the program that each of the subjects interact with. Apart from giving them all of the messages necessary to conduct the experiment, it prompts them for messages to send back. In terms of the "look and feel" of the experiment, IND is the most important from the perspective of the subjects.

SOLVE provides a complete solution to the MB game for the particular parameters being used in the experiment. If there are simulated players then this program must be run prior to the experiment, since it provides the basis for our simulated responses. For most interesting games the amount of time required for a complete solution is quite high. We strongly recommend that this stage of the experimental design be undertaken several days before the actual experiment, at the very least.[2] We discuss the way in which SOLVE actually computes a solution in some detail in Section 4.

---

[2] We have also written a special-purpose computer program to solve these dynamic programming problems. This program is considerably faster than the solutions generated by SOLVE, but is not adequately documented to allow readers to understand the logic of our calculations.

INIT simply sets up the experiment before the actual day of the experiment. There is nothing in INIT that cannot be done by MAIN when actually running the experiment, but for some larger experiments with many computer-simulated players there are advantages in having the experiment completely initialized before the day of the experiment. Specifically, the generation of random allocations of players of specified "types" to particular replications of the game can be time-consuming when there are over 30 subjects; this occurs frequently when we have, say, 19 "human" subjects each playing against 5 "simulated" players (resulting in 114 subjects as far as the program is concerned). INIT also checks that the ASCII input file contains all of the information needed to run the experiment.

The final program, OUTPUT, reads the results of the experiment as generated by MAIN and produces a "report" file that is relatively easy to read and interpret. It can also produce analyses of the output as needed.

In terms of the sequence of usage, the experimenter would begin the actual experiment by running MAIN. To do this he would enter the command "MAIN expid" where "expid" is the identifier of the experiment. The program then looks for a file called expid.CNF, the structure of which is described in section below. This file defines the experiment to be run. The program MAIN is typically run on a computer that subjects do not have visual access to, since it may contain some keywords that subjects should not see (so as to ensure control over their motivation).

After all of the files have been initialized by MAIN, it will "beep" several times and alert the experimenter that he may now log in each of the individual subjects. To do this the experimenter simply gives the command "IND" at each of the computers assigned to subjects.

The only important thing that the experimenter must do at this stage is to enter the experiment ID number for each subject. This is typically a sequence of 1, 2, 3, etc., but need not be in particular experimental configurations explained later. We strongly urge the experimenter to initialize this number rather than allowing the individual subject to, since subjects often make mistakes and this can needlessly delay the beginning of the experiment.

One attractive design feature of the experiments is that one can "re-start" any experiment in progress if there is some reason (such as a temporary power failure) to do so. If it is only an IND program that must be re-started, such as if some idiot subject turned off his computer or entered CTL-ALT-DEL, then this can be done without interrupting the other programs. This IND program will just pick up where it left off. If the MAIN program must be re-started then all of the subjects must be logged in again in the original sequence (i.e., wait until the MAIN program says that they may be logged in). The only difference is that the experimenter must tell MAIN to begin in some period other than period 1. Note that the experiment must be re-started in the period in which the system went down: no data from that period will be reliably stored,

although all data from the earlier (completed) periods will be. The software is extremely conservative in saving data, doing so at the end of every period. If the system goes down and the experiment must be terminated for that day, then the data will still be saved for all previous periods providing the disk has not been corrupted.

When the experiment is finished the MAIN program will tell the experimenter how much each subject is to be paid. It is wise to write this information down quickly on the record sheet, so that subjects can be paid and dismissed efficiently. As soon as this information has been recorded, the experimenter could run the program OUTPUT to begin the collation and processing of the results. In this way it is typically possible to get some instant feedback on the experimental outcomes, slating one's natural curiosity!

5.2 Configuring an Experiment

The experimenter must initially generate an ASCII file which contains the configuration of the experiment to be conducted. The structure of this file has been carefully chosen so as to minimize the amount of information that needs to be entered, as well as keeping the file as non-cryptic as possible. In this way we hope that the essential experimental environment under study in any particular experiment can be quickly determined by simply inspecting the relevant configuration file.

All data and information may be entered in "free-format", using upper or lower case as seems natural. By "free-format" we

mean that there may be one or more spaces between data, to enhance the readability of the file.

There are a number of "delimiters", enclosed in square brackets [], that tell the software that the next lines contain particular pieces of information. It is the job of the experimenter to define all of the necessary delimiters as well as to enter the relevant data in succeeding lines. The delimiters must begin in column 1, and are "reserved words" that should not be used in square brackets that begin in column 1. There are a number of default options for some of the delimiters, so not all of them are required. However, we urge the use of all delimiters simply to be certain that the experiment has been configured as desired.

It is pedagogically easiest to introduce the structure of a CNF file by means of an example. We will discuss the default options later. The following file is called T.CNF, and implements a small multilateral bargaining experiment:

```
==> T.CNF  ... TEST configuration file for FIRST experiment (with a Core)

[nagents]              ' Number of agents, including Government
6

[agents]               ' Names of agents (up to 60 characters).
Gordon
Lisa
Jenny
Paula
Christine
Stephanie

[npolicies]            ' Number of policy dimensions
  2

[policies]             ' Names of policy dimension (up to 60 characters)
Horizontal
Vertical
```

```
[ngroups]                ' Number of experimental groups (or clones)
19

[simulated]              ' Agent or player ID and an asterisk if simulated
gordon          *
lisa            *
jenny           *
paula           *
christine       *
stephanie

[voting power]           ' Agent or player ID and number of votes
gordon          0
lisa            1
jenny           1
paula           1
christine       1
stephanie       1

[access]                 ' Agent or player ID and access prob.(to be normalized)
gordon          .0
lisa           12
jenny           1
paula           1
christine       1
stephanie       1

[matched proposals]      ' Have proposals from the same agents over replications
no

[u-default]              ' Agent or player ID and default utility level
gordon          0
lisa            0
jenny           0
paula           0
christine       0
stephanie       0

* NOTE: alternatively, user can enter the [p-default] values

[u-squo]                 ' Agent or player ID and status quo utility levels
gordon          0
lisa            0
jenny           0
paula           0
christine       0
stephanie       0

* NOTE: alternatively, user can enter the [p-squo] values

[u-ideal points]         ' Ideal points of Euclidean Utility function
gordon          0        0
```

```
lisa            39      68
jenny           30      52
paula           25      72
christine       62     109
stephanie      165      32

[u-intercept]           ' Intercepts of Euclidean Utility function
gordon          0
lisa            90
jenny           70
paula           70
christine       90
stephanie       110

[u-coefficient]         ' Coefficients of Euclidean Utility function
gordon      0.00
lisa        1.0
jenny       1.0
paula       1.0
christine   1.0
stephanie   1.0

[nperiods]              ' Number of periods per game (T)
 5

[nrepetitions]          ' Maximal number of times we play the whole game
10

[time]                  ' Maximal number of seconds per period
3600

[shuffle]               ' Shuffle players from game to game ("yes" or "no")
yes

[path]                  ' Path for all messages (this is system-specific)


[solver]                ' Call to GAMS solver
gams

[government]            ' Indicate whether or not there is a Government
no
```

Each of these options is now discussed in the sequence presented above.

The first line, defining the name of the file, is not required but is a useful convention to follow so that readers know from a

hardcopy printout which file they are reading (in this case, T.CNF).

The number of agents in the experiment is defined with the **[nagents]** delimiter. The line after the delimiter should simply have a number corresponding to the number of agents, including the "center" or "Government" agents. In this case we have five private agents and one Government agent, hence 6 agents in all.

The **[agents]** delimiter allows the entry of names for each of the agents, one name per line. There should be as many names here as there are agents. We always identify the government agents first. The private agents then follow, in any order. The name of any agent may be up to 8 characters, and cannot have spaces and any other funny ASCII character.

The **[npolicies]** delimiter defines the number of policy dimensions that will be bargained over. The following delimiter, **[policies]**, allows the experimenter to provide names for each of the policies. Again, up to 8 characters are allowed. It is recommended that short policy names such as "tax" and "subsidy" be employed to improve readability for subjects (these names will be used in the screen displays given to subjects).

The **[nplayers]** delimiter defines the number of players in the experiment. This is quite distinct from the number of agents per game, since there will typically be several games played simultaneously in any given experiment. Thus, if we have 20 computer terminals and six agents per game, we will typically play three games each with six agents (one terminal is to be used for

the monitor program, MAIN). It is easiest to think of [nplayers] as defining the number of experimental subjects in the session. Note that the number of players must be some integer-multiple of the number of agents defined by the [nagents] delimiter.

Each player, or experimental subject, will have an identification number defined by the **[players]** delimiter. These ID's need not actually be numbers, but this is by far the best way to identify subjects for our purposes.

The **[simulated]** delimiter signals that one or more agents or players will be computer-simulated in the experiment. Following this delimiter is a list of either the agent names or the player names, and an asterisk if the agent/player is to be computer-simulated. The reason for the option here of giving agent or player names is that one or other may be more convenient for different experimental purposes. It is not possible to list some agent names and some player names: one or the other is required. All names must be listed, whether or not the agent/player is to be simulated. The names must also be listed in the same order as defined in the [agents] or [players] fields. If there is no asterisk beside an agent/player name, then that agent/player will be a "human" subject. In the above example, note that the two government agents and the final private agent are being simulated.

The **[voting power]** delimiter signals the number of votes that each agent or player has in the game. Again, it is possible to list either agent names or player names depending on whichever is more convenient. In the above example each of the private agents have

one vote.

The **[access]** delimiter signals information on the access probability that each agent/player has. This refers to the probability that the agent will be called upon to make a proposal in any period. In the above example each private agent has equal probability. In this case each of the government agents get to make proposals with five times the frequency of the private agents. The program normalizes all of these probabilities so that they sum to unity: it is often easier simply to enter relative integer weights as is done here.

The **[matched proposals]** delimiter, which admits of a "yes" or "no" option", controls the way in which proposers are selected across different games being played simultaneously. If the "yes" option is invoked then the same agent will be chosen in all of the games being played in the same period. Note that this option does not mean that agents are not randomly chosen in each period (according to the access probabilities), but it does add some experimental control over the independent replications being conducted in any given session.

The **[u-default]** array defines the default utility level for each agent or player. An alternative way to input essentially the same information is to define the default policy values using the **[p-default]** delimiter: the program then computes the implied default utility values using these policy values. If the [p-default] option is used then one simply lists the policy names and their default values. In the above example each player's

default utility is set at some arbitrarily large negative number.

The **[u-squo]** delimiter defines the status quo utility levels, in the same manner as the [u-default] field. Again, it is possible to enter **[p-squo]** values defining the status quo level of each of the policies.

Three delimiters are used to specify the Euclidean utility function of each agent: **[u-ideal point]**, **[u-intercept]**, and **[u-coefficient]**. The interpretation of these coefficients is documented in the GAMS file generated by the SOLVE program.

The **[nperiods]** delimiter defines the number of periods per game. Each game is played with the same players being assigned to given agent types: thus the same four human subjects would be playing against each other for each game, and then would be "shuffled" if this is requested (see below). In the above example there are 5 periods per game.

The **[nrepetitions]** delimiter defines the maximum number of times that the whole game is played. In the above example, we ask for 10 repetitions. Note that player assignments to agent type will be shuffled from repetition to repetition, not from period to period "within" each game as defined here.

The **[time]** option allows the initialization of a time limit, in seconds, on the length of any period. This option causes the programs to start warning subjects that the time limit is approaching. If the time limit is reached without any proposal or response then the next period is immediately implemented. The maximal number of seconds that may be initialized is 30000, which

is just over 8 hours and well beyond the maximal bladder size of any known experimental subject! In effect, then, one can turn the time limit "off" by setting this value arbitrarily high. In this case it is important for the experimenter to be monitoring the MAIN program to see if one or more subjects is not responding (this sometimes happen if subjects do not realize that they are being prompted for an input). In the above example we configure the experiment for 1 hour periods (i.e., 3600 seconds), in effect allowing an unlimited time for subjects to respond.

The **[shuffle]** option controls whether or not players are to be shuffled from repetition to repetition. This is an important option to minimize (practically speaking, to eliminate) "reputation effects" from the same players playing in the same games. Note again that we do want the same players to participate in each game, which may well last a number of periods, but we do not necessarily want the same players to meet each other in game after game (i.e., in all of the repetitions of each game). This option, which admits of "yes" or "no" values, will be crucial to our experimental assessment of reputation effects.

The delimiter **[path]** defines the system-specific path in DOS for all file communications. This path is a DOS sub-directory that all subjects, or at least their computer terminals, can access. It is system-specific to particular microlab configurations, but should be easy to ascertain from a system operator and should be constant from experiment to experiment (in the same lab, of course). In the present example we leave this blank, implying that

the default DOS sub-directory is used.

The delimiter **[gams]** defines the call to the GAMS software package on the user's system. This package is used in the SOLVE program to set up a series of non-linear programming problems. This program generates a report file, with the suffix ".REP", that is used by the MAIN and IND programs when implementing computer-simulated strategies. We have also developed a faster program to undertake the same calculations as SOLVE but without having to access GAMS; this program is not documented in this report, but the results have been extensively checked with those generated by the SOLVE program, which is documented.

All of these options may be entered in any order: the computer program reads in the basic "dimensioning" delimiters first and then passes through the CNF file a second time looking for the other arrays. The INIT program is designed to warn you if any delimiter is not read in correctly by the program -- it will inform you if any array is missing or if "default" values have been assigned.

REFERENCES

Brooke, Anthony; Kendrick, David, and Meeraus, Alexander, **GAMS: A User's Guide** (Redwood City, CA: Scientific Press, 1988).

Harrison, Glenn W., "Multilateral Bargaining in Economics Experiments: A Survey", **Unpublished Manuscript**, Department of Economics, University of South Carolina, January 1991a.

Harrison, Glenn W., "Multilateral Bargaining in Political Science: A Survey", **Unpublished Manuscript**, Department of Economics, University of South Carolina, January 1991b.

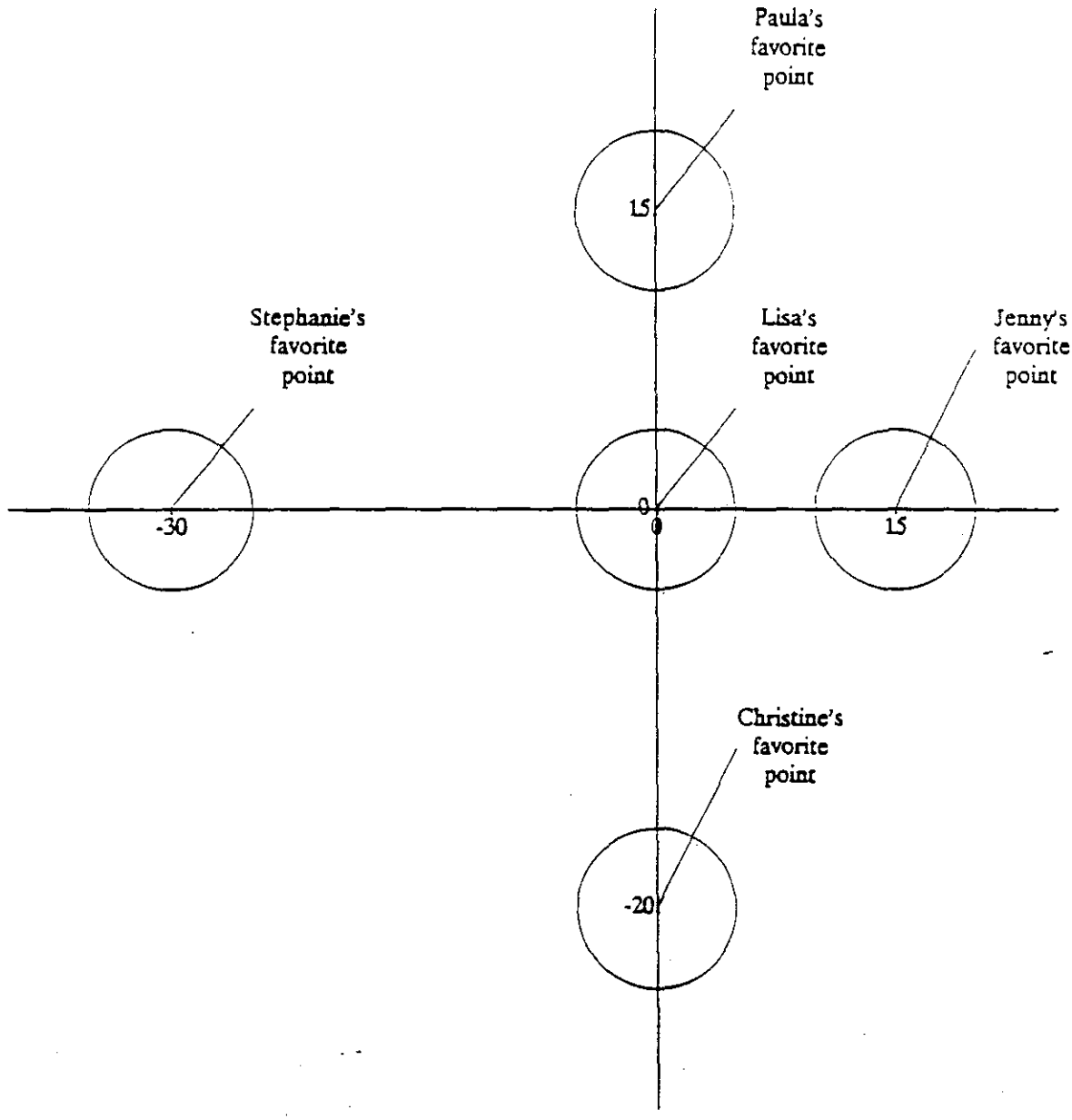
Harrison, Glenn W., and Simon, Leo K., "Experimental Assessment of Multilateral Bargaining: A 'Horizontal' Political Economy Model", **Unpublished Manuscript**, Department of Economics, University of South Carolina, December 1990.

Harrison, Glenn W., and Simon, Leo K., "Experimental Assessment of Multilateral Bargaining: Some Initial Results", **Unpublished Manuscript**, Department of Economics, University of South Carolina, January 1991.

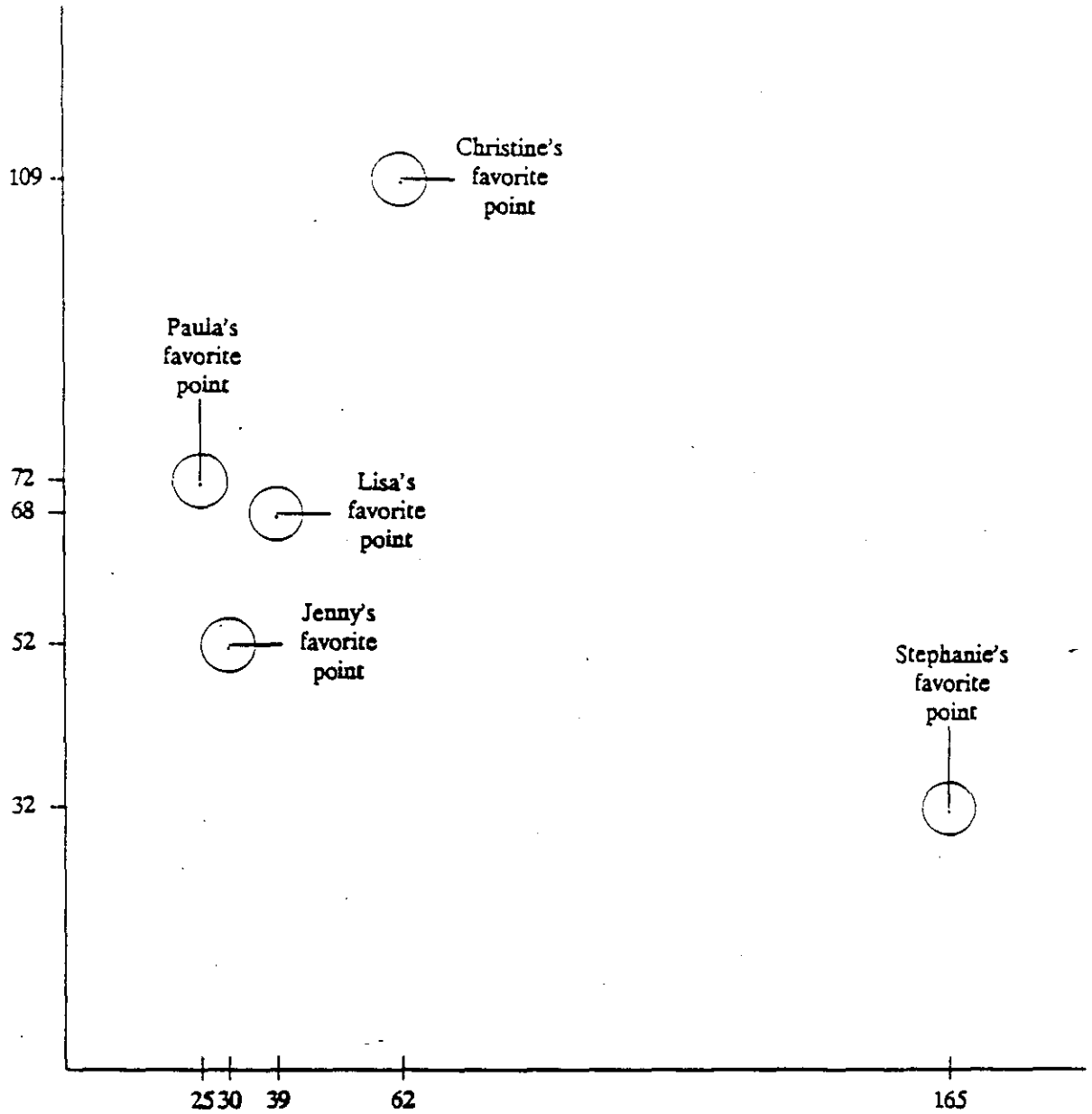
Rausser, Gordon C., and Simon, Leo, "A Noncooperative Model of Multilateral Bargaining", **Unpublished Manuscript**, Department of Agricultural & Resource Economics, University of California at Berkeley, January 1991a.

Rausser, Gordon C., and Simon, Leo, "Burden Sharing and Public Good Investments in Policy Reform: A Multilateral Bargaining Approach", **Unpublished Manuscript**, Department of Agricultural & Resource Economics, University of California at Berkeley, January 1991b.

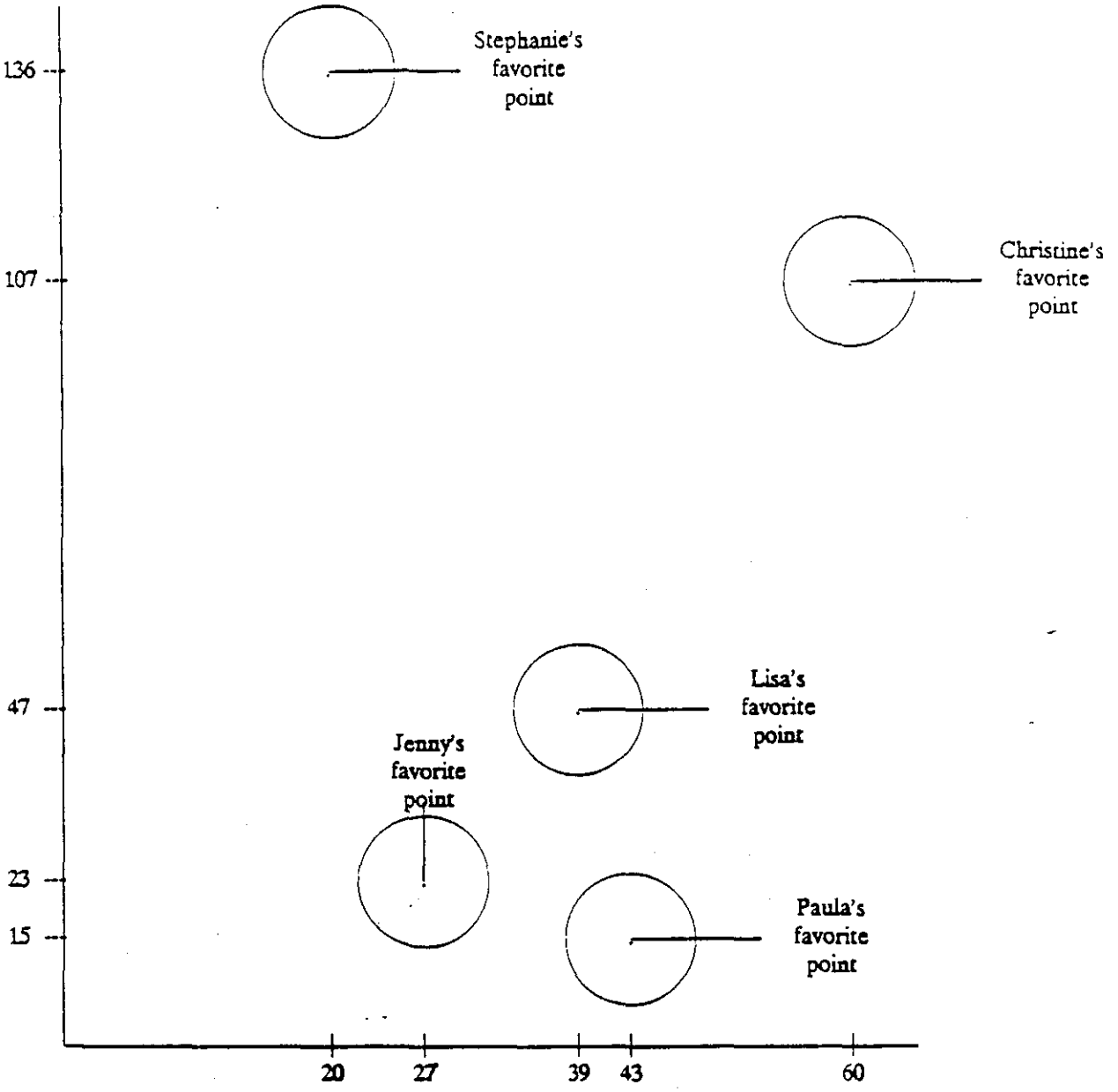
Rubinstein, Ariel, "Perfect Equilibrium in a Bargaining Model", **Econometrica**, v. 50, January 1982, pp. 97-109.



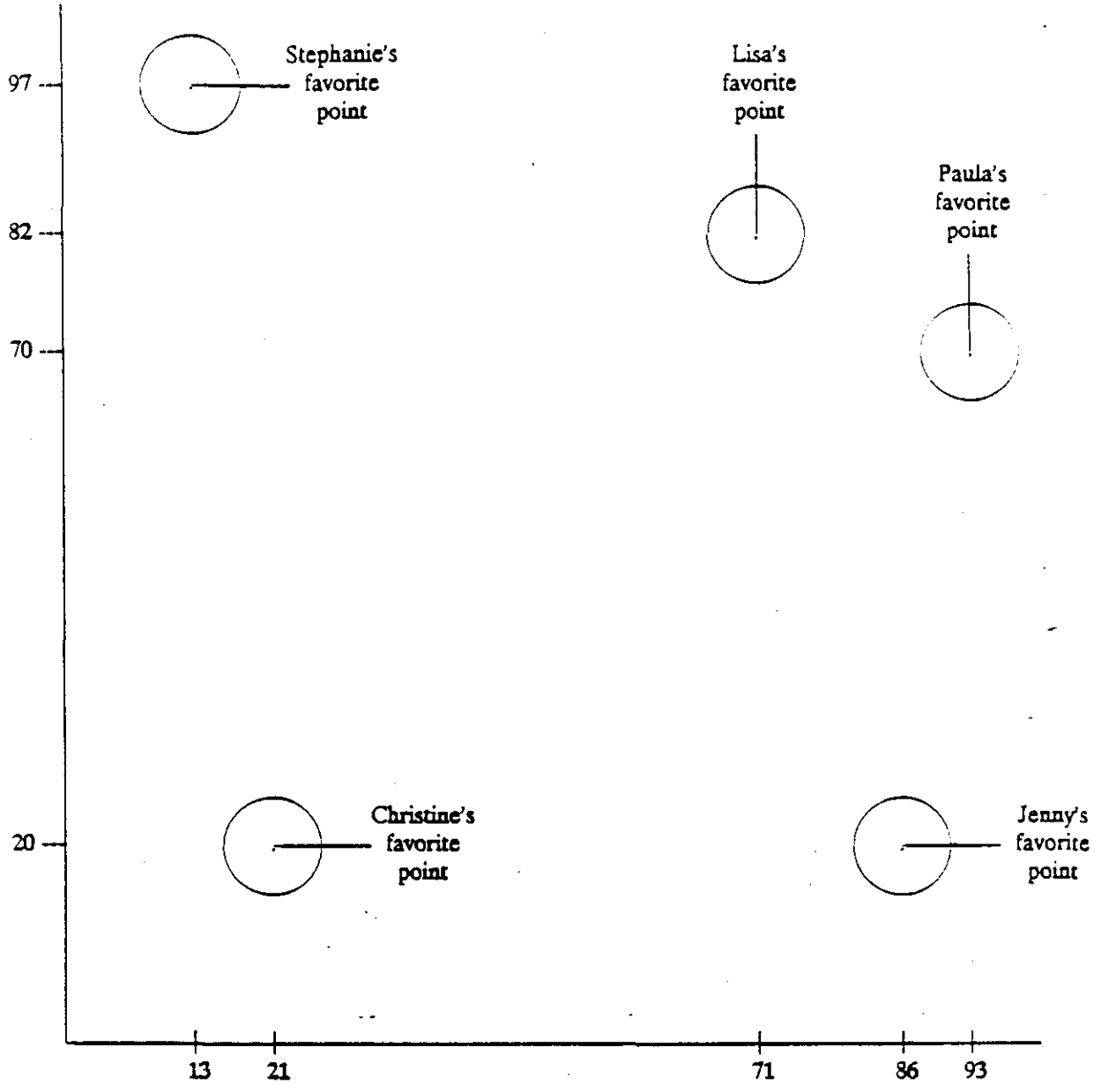
Experiment #0



Experiment #1



EXPERIMENT #2.



Experiment 3.