Winter 1995

Developing an Exchange Network Simulator

Barry N. Markovsky
University of South Carolina - Columbia, barry@sc.edu

Publication Info

Published in *Sociological Perspectives*, Volume 38, Issue 4, Winter 1995, pages 519-545.
http://www.ucpressjournals.com/journal.asp?j=sop

Sociological Perspectives
Vol. 38, No. 4, pp. 519-545

ISSN 0731-1214

DEVELOPING AN EXCHANGE NETWORK SIMULATOR

BARRY MARKOVSKY*
University of Iowa

ABSTRACT: *"X-Net" is a computer simulation that I developed in conjunction with Network Exchange Theory. Users of X-Net can explore the effects of different network structures, rules of exchange, and negotiators' strategies on the dynamics and outcomes of resource exchanges in social networks. This article recounts the process of X-Net's development, in addition to key substantive, theoretical, and design issues that motivated its form and content. It concludes with a discussion of the relationship between theory, simulation, and empirical tests.*

INTRODUCTION

Most social scientists who are exposed to computer simulations only see the end results of long developmental phases. In general, investigators report on the behavior of a simulated process under various configurations of modeling assumptions and parameters. This is the *science* of computer simulation. In the course of developing the simulation, however, the author invariably proceeds in an evolutionary fashion, building layer upon program layer in a progressive, albeit trial-and-error-laden, series of operations. This is the *art* of simulation.

My purposes here are to review both the evolution and present form of X-Net,[1] a simulation that has become a useful adjunct to a particular program of theory-driven research. Although I also discuss briefly some substantive findings, these are not the focus. Instead, I review how the program of theory and empirical research helped to spawn the simulation and how the simulation, in turn, has been able to promote theory development and empirical testing. The discussion is primarily aimed at: (1) those who have not written a simulation, but also (2) those who have done so but may not have considered its connections to the theory-building process, and even (3) those who may have a particular interest in social exchange network simulations. The framework for my discussion is chronological,

* Direct all correspondence to: Barry Markovsky, Department of Sociology, University of Iowa, Iowa City, IA 52242. *e-mail:* barry-markovsky@uiowa.edu

highlighting the recent intellectual history of "network exchange theory" and X-Net, the simulation.

BACKGROUND

For over a decade I have been intrigued by social exchange processes—particularly, how network restrictions on those processes alter behaviors and exchange outcomes. I was first inspired by a prominent article by Cook, Emerson, Gillmore, and Yamagishi (1983). Arguing from the perspective of Richard Emerson's (1972) "power-dependence" (P-D) framework, the authors presented results for several laboratory experiments and computer simulations, finally offering an explicit, mathematical formula (called the "vulnerability" model) for calculating network exchange outcomes. Although the computer simulation program was little discussed, its key features were described briefly in a footnote.

The research findings were fascinating. They clearly demonstrated that experimental subjects in off-center positions in a variety of networks fared much better in their exchanges than centrally located subjects. For example, in a network configured as A—B—C—D—E, positions B and D obtained significantly higher profits than A, C, and E, and C's profits were approximately equal to those of A and E. It is important to note, however, that what we now call a "one exchange rule" was in force: Every subject was restricted to making just one exchange per round of negotiation, even if connected to more than one potential exchange partner. It can be seen that under such conditions, A, C, and E all risk being excluded from exchanging in a given round, whereas B and D will always have at least one willing and available partner. Although not discussed in such terms by Cook et al., it seems that a position's proneness to exclusion may be *the* determining factor regarding its occupant's ability to extract advantageous exchange outcomes: Minimum excludability imparts maximum power.

Despite the interesting findings, Willer (1986) noted and published a discussion of even more serious problems in the Cook et al. (1983) article. He demonstrated that "applying vulnerability leads to logically and empirically impossible inferences in a wide variety of applications." At the time, he offered no theoretical alternative to vulnerability for predicting power in social exchange networks. Soon thereafter, he and I began collaborating on alternative models, thus embarking on a long-term program of theory development and research.

The more we worked on our own theory, the more nagging questions seemed to emerge with respect to Cook et al.'s approach. For instance, although their laboratory experiments were straightforward enough, reasons for certain design features were not always apparent, and their larger purpose vis-à-vis the power-dependence program was not so clear. Did the experiments *really* test the P-D "framework"? The authors provided a set of explicit hypotheses for the experiments, ostensibly obtained from P-D. Yet there were no propositions from which predictions could have been logically derived. The hypotheses were ad hoc. Did the experiments *really* test the vulnerability model? The data were consistent with

vulnerability predictions, but the model was only given post hoc. There was no claim that vulnerability was used to generate the hypotheses. Why the one-exchange rule? Nothing in P-D mentions such a restriction, yet it seemed absolutely essential to generate the experimental results. Finally, what was the purpose of the computer simulations? Were the experiments intended to *test* them? Although the experimental results roughly conformed to the simulations, the authors did not claim or imply that the experiments provided support for the conclusions drawn from the simulations. In fact, the simulation results were treated as having the status of empirical data, which in turn were treated as providing *even further corroboration* for the P-D theory. These researchers are not alone in their failure to realize that computer simulation output and human behavioral data are not substitutable for one another. As trivial as it may sound, it is apparently necessary to emphasize that theories of human social behavior can be neither corroborated nor refuted by computer simulations because simulation output is not human social behavior.

To better understand how computer simulations could be put to more legitimate use in the study of exchange networks, I began developing my own network exchange simulator. Writing in the Microsoft QuickBASIC programming language and taking clues from the relevant Cook et al. (1983) footnote, I found it remarkably easy to reproduce their findings. I then began to explore alternative specifications for key elements in the program. The theoretical payoff was almost immediate. My little simulations made it eminently clear that power was not a result of network structural patterns alone, but also depended on assumptions made about the actors in the network.

For instance, consider a network consisting of three people linked via two relations: A—B—C. Each relation represents a channel through which negotiations may take place and, thus, the locus of a potential resource exchange. These early simulations revealed that, depending upon how actors' negotiation strategies were programmed, any actor's profits could exceed those of any other actor. Thus, under some conditions the "intuitive" finding holds: $A = C < B$. Under other conditions, however, other orderings occurred, for example, $A = B = C$, $A > C > B$, and so forth. Contrarily, all network theories of which I knew asserted unconditionally that B's centrality should accord that position an advantage over the others (Markovsky 1987).

These simulations clearly showed that much was being left implicit in network exchange theories. In a sense, the simulations said *more* than those theories that try to predict exchange outcomes based on network structure: The theories make assumptions about the decisions and behaviors of the actors in the network. In another sense, they say less: They make no assumptions about structural factors—the very phenomena that motivated exchange network theories to begin with! The simulated actors were not programmed to somehow take account of the larger social network in which they were embedded. The differential resource distributions *emerged* from the process of interaction.

The theoretical status of computer simulations became most apparent to me while simultaneously working with David Willer on a mathematical model of

power in exchange networks. *The mathematical model and the computer simulation was each the heart of a different theory.* Ironically, and apparently without realizing it, Cook et al.'s (1983) footnote on their simulation algorithm offered a rigorous alternative to their informal, subjective, and interpretive PD framework. That is, their simulation "results" were logical derivations from a set of programmed assumptions, and these provided ready-made but untapped derivations that, when operationalized, could have served as hypotheses for their experiments. The experimental results in fact provided empirical corroboration for their simulation *qua* theory!

The simulations differed in at least one very significant way from Cook et al.'s vulnerability model and the "graph-theoretical power index" (GPI) that we later published (Markovsky, Willer, and Patton 1988): The vulnerability model and GPI used an *analytic* approach to predicting exchange outcomes, whereas the simulations took an *iterative* approach. In general, analytic approaches are more mathematically elegant, allowing derivations to be calculated simply by assigning any needed initial conditions and/or parameter values. In contrast, the iterative approach requires a series of calculations that frequently correspond to the temporal unfolding of a process. Calculations for a given iteration of the model depend upon results from the previous iteration. The iterative approach has the potential to more easily model more "realistic" aspects of a process—for example, its complex dynamics—and to accommodate more factors without creating mathematical intractability. Each approach has its benefits and costs, and each emphasizes different properties of the phenomena it models. To further highlight this contrast, the following sections review the network exchange theory in greater detail, followed by a discussion of the development of the X-Net simulation.

NETWORK EXCHANGE THEORY

Network exchange theory (NET) has developed incrementally over a number of years, rooted in the "elementary theory" and research of Willer and his colleagues (Willer and Anderson 1981; Willer and Markovsky 1993). The first version of NET (Markovsky et al. 1988) was an attempt to correct problems in Cook et al.'s (1983) vulnerability model and to develop a more explicit and general formulation. It resolved the logical problems noted by Willer (1986), was consistent with virtually all of the results of both old and new network exchange research, and predicted new classes of phenomena that were not addressed by other theories—for example, power reversals and the emergence of sub-networks when the "one-exchange rule" is relaxed. Later versions have built from this core formulation and now include a variety of further refinements and insights (Markovsky et al. 1993; Skvoretz and Willer 1993; Lovaglia, Skvoretz, Willer, and Markovsky in press).

The Network Exchange Theory is fully explicated in the Appendix. The heart of the theory consists of the Graph-theoretic Power Index (GPI, or Axiom 1) and three associated axioms. The GPI serves to generate a numerical value for the structural power of a given position relative to that of other positions to which

it is connected. It is based on the idea that a position gains power from having more relations; a position loses power when those relations have other relations; the position gains power from those relations' relations that have other relations, and so on. The GPI for every position in the network can thus be calculated based on simple path-counting rules.

Having thus calculated power indices, the second axiom specifies when actors will be interested in exchanging—that is, when two actors represent one another's most profitable (or least costly) alternatives. Axiom 3 allows us to predict breaks in the network that result when the structure of relations makes some of them unprofitable. Finally, the last axiom relates relative power to relative exchange outcomes.

In general, GPI is sensitive to very robust power differences. For example, assume that actors in each relation of the B_1—A—B_2 network are engaged in a series of negotiations over the division of a pool of 24 resource units. An exchange occurs for a given period when A reaches agreement with one of the Bs as to who gets how much of the pool. On the next round, the pool is replenished. However, one of the Bs was excluded from the prior exchange and will try to re-enter by making better offers to A. Over time, a bidding war between the two Bs results in profit divisions that approach 23-1 favoring A.

Recently, my colleagues and I published an extension of this theory that generates finer-grained predictions for a class of networks in which some power differences are predicted to be much weaker (for supportive tests, see the Appendix, part 2; Markovsky et al. 1993). The idea is that in some networks, ongoing exchanges produce temporary changes in the number of an actor's available exchange partners. As a result, power can shift temporarily, creating small power differences between actors who are otherwise equal in structural power.

Most recently, extensions to the theory provide methods for predicting actual profits for each position—not just profit orderings (see Appendix, part 3; Skvoretz and Willer 1993; Lovaglia et al. in press). These have also been supported through experimental tests. Current research involves further refining, expanding, and testing the network exchange theory, paying greater attention to dynamics of negotiations and the properties of individual negotiators (Lovaglia, Skvoretz, Markovsky, and Willer 1995).

X-NET SIMULATIONS

In 1988 I received a grant to further develop the simple program I had published the previous year (Markovsky 1987). Thus, almost from the beginning of my collaborative work on the analytic model, I was also exploring the simulation approach. In fact, the plan was to use NET's experimental setting and scope conditions as my guide for designing X-Net.

Starting Up

My previous experiences in computer programming were rather limited. I took a course in PASCAL while in graduate school, but I never became a regular user

nor owned a PC version of that powerful language. Apart from the simulations noted above, I had only written some small programs in Microsoft BASIC, mostly for controlling the presentation of instructions and collection of data from subjects in experiments. I decided to use QuickBASIC (QB), another Microsoft product. Alternatives included high-level languages favored by serious programmers such as Pascal and C, or others such as SimScript that were specially designed for simulation programming. QB's advantages included: (1) It developed from the older BASIC with which I was already familiar and thus shared much of its command language and syntax. (2) It greatly enhanced BASIC by incorporating powerful new language elements that permitted more highly structured programs. Structured programming permits larger and better organized programs to be built or "chunked" from a number of smaller semi-isolated subprograms, where each subprogram performs a simpler subtask. With this expansion in capabilities, even some "serious" programmers began to adopt QB. (3) QB was part of a wave of very similar BASICs that were becoming increasingly popular (e.g., TrueBASIC and TurboBASIC), making programs written in QB relatively transportable. (4) Its grammar and syntax were quite intuitive, relatively easy to learn and teach. Students and colleagues with little or no programming experience can look at QB code and get a pretty good sense of how a program works. If a program is well-structured and internally documented with comments and instructions, users with minimal training can even explore the effects of program modifications. I have never regretted settling on QB. The C language and its variants are certainly speedier and more powerful, and still hold sway among programmers. However, the ease of developing and modifying X-Net, along with its comprehensibility to students and colleagues, more than compensate for the relatively small sacrifice in execution speed.[2]

Having chosen my language, I can remember very well sitting at the computer ready to begin work on the project. It was at this point that I decided to call the program "X-Net," short for e**X**change-**NET**works. The fact that I named the program before writing any of it hints at the anxiety I felt over possibly having bitten off more than I could chew. So my first task became staving off something akin to writer's block. That first day, I accomplished little more than writing the code for a big, silly, graphic display designed to flash "X-Net" on the screen at the start of the program. Although I soon discarded that part of the program, at the time it served both to remove the sense of facing a tabula rasa and to familiarize me with some of QB's graphics capabilities. It also got me thinking about X-Net from the user's point of view. As I further developed the program, I was motivated to make X-Net a *user's* program, as opposed to a *programmer's* program. I spent almost as much time developing a simple, informative, and visually clean interface as I did the substantive algorithms. Later on, this paid dividends in that a number of people who may otherwise not have done so have used the program and provided feedback.

The first truly substantial programming involved writing routines to handle the networks themselves. These capitalized on the fact that network configurations

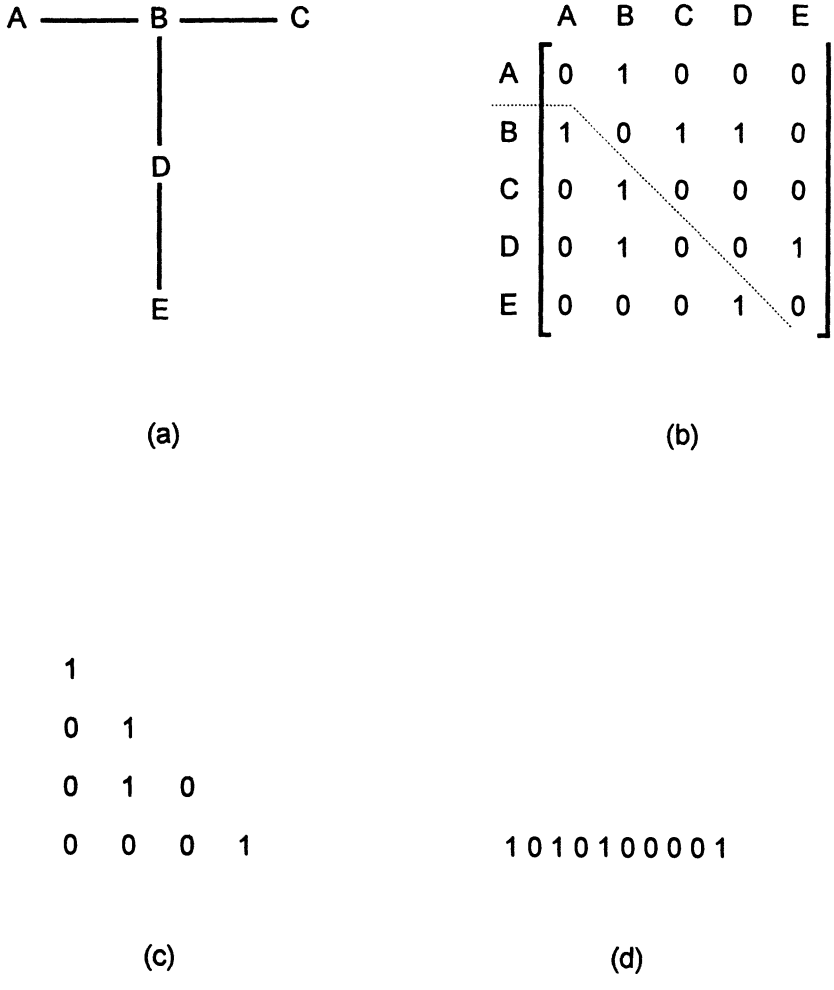

Figure 1
Representing a Network

can be readily stored as matrices of 1*s* and 0*s*. The five-actor network in part a of Figure 1, for example, can be fully described via the connection matrix in part b. The 1*s* indicate relations; the 0*s* nonrelations. When relations are mutual—for example, A is related to B whenever B is related to A—then matrices such as shown in part b contain redundant information. In fact, the network can be completely reconstructed based only on the nonredundant information below the dashed line,

as shown in part c of Figure 1. Finally, reading across the successive rows of part c, all necessary information on the structure of the network can be reduced to the single line shown in part d. In general, all networks with symmetric relations can be so represented using exactly $N(N-1)/2$ digits, where N is the number of positions in the network.

With network configurations reducible to mere strings of numbers, the first routine was a subprogram for reading these numbers from "network information files." I thought that it would be nice to have a second routine that would allow users to easily create such a file from a back-of-the-envelope drawing of the desired network. This routine grew into a completely separate, interactive program, NETMAKE, that lets the user describe and store a network simply by naming it, declaring the number of positions, and answering a series of $N(N-1)/2$ yes-or-no queries concerning which positions are connected to which others. Later, I incorporated another subprogram (described below) into NETMAKE that graphically displays the completed network. The user is then asked to either verify the network, start the process again, or exit NETMAKE without storing the network.

While at this stage of the project, I made an interesting discovery about my own capacities for doing this work that others have since confirmed applies to them as well: I could work longer and with greater efficiency when I switched back and forth between the serious, nitty-gritty, programming on the one hand and the more visually stimulating user-interface on the other hand. It is quite common to "get stuck" when programming—to not be able to locate a bug or to not be able to muster sufficient concentration, for example. At such times, it is usually best to set aside the problem for a while and go on to other things. When "other things" include a different set of programming tasks, which was the case with the program's graphic displays, other parts of the program continue to be developed during the "break," and time is thus used more efficiently. For me, the user interface displays were generally easier to program than the network exchange processes, and they provided the added benefit of immediate gratification. As a result of this strategy, the user interface evolved in step with the more technical components.

Program Structures and Functions

The programming described thus far was very straightforward. It was not long, however, before I had to begin developing routines for negotiations and exchanges among actors in the networks. This meant devising a simulated world that satisfied the scope conditions of the Network Exchange Theory (Appendix, part 1). The laboratory experiments provided a useful model. In these experiments, actors negotiate and exchange within each of a series of rounds. A single experiment involving the same group of actors could consist of, say, 10 rounds. Multiple experiments conducted with multiple groups of subjects would then be repeated and the results aggregated to permit more powerful inferences regarding effects of positions and other variables.

According to the theory's first scope condition (SC1), all actors must use the same strategy. This was certainly a handy simplification for purposes of programming. Any differential exchange outcomes that emerged in simulations would then *have* to be the result of structural differences among the positions that actors occupied and not due to idiosyncratic behaviors of those actors. SC2 asserted that every position must be related to, and seek exchange with, one or more other positions. The network description routines did not prevent the user from describing an isolate. However, that isolate would obviously never engage in negotiations or exchanges, so this condition posed no special problem. SC6 was satisfied by providing a fixed resource pool for each relation, with 24 units by default (because most experiments used 24 units per pool).

The remaining scope conditions were trickier in that they delimited various aspects of the exchange settings and processes to which NET was deemed applicable, but without actually specifying any concrete features. That is fine for scope conditions but not as useful for writing a simulation. We return to this problem below.

At this point, I had a set of actors, network relations, and resource pools. The next problem was how to breath life into this system and have the actors negotiate over and divide up those resources. The theory provided no solution to this problem. After all, the theory sought to use structures to predict exchange outcomes; it did not set out to model negotiation processes. Thus, whereas the theory had essentially determined the program up to this point, from here on the program was underdetermined vis-à-vis the theory. For instance, the theory did not say how actors should decide with whom to negotiate, how much to offer, how to evaluate offers received, when to accept another's offer, or how to adjust their offers in view of past outcomes.

The need to fill in missing details forced me out of the theory and into an exploratory mode. I embarked on a series of trial-and-error attempts to make the simulation do *something* that seemed interesting. One of the first features to develop during this phase was the overall program structure, thanks to the laboratory experiments that served as a template. There are plainly multiple nested "levels" of activity in network exchange research, as illustrated in Figure 2. Level 1, or the "top" level, consists of the *Experiment*. This corresponds to an entire network exchange experiment involving multiple groups of subjects run in the same network configuration, but at different times. The results of these different groups can be described via aggregated variables—that is, average profits per exchange across all subjects occupying a given network position. For instance, we might say that "after running five different groups of subjects in the B_1—A—B_2 network, position A averaged 20 points."

Level 2 consists of *Sessions*. One experiment contains multiple sessions, where each session involves a group of subjects (or simulated actors) engaging in a series of negotiations and exchanges.

Level 3 then consists of the *Rounds* that comprise a session. Within a session, subjects are usually restricted as to the number of deals they can make within

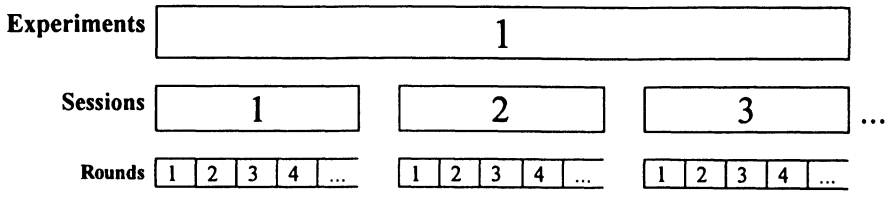

Figure 2
X-Net Program Levels

a given round, although they are often free to negotiate with multiple partners, making offers and counteroffers to each.

Prior to developing X-Net, these were the only program levels that seemed essential, with the exception of a "bottom" *Actions* level in which actors make offers and counteroffers. Each of the three levels would perform a few chores such as initializing variables (i.e., resetting to zero or one certain variables used the last time the program entered the particular level) and running a program loop that moves through tasks at the next lower level.

The specific subprograms developed as follows. At the *Experiment* level, a subprogram called MainMenu displays a set of selectable options for the user (see Figure 3)—that is, receive "General information" on the program, "Make or delete a network" from the list of stored networks, "Run a simulation," or "Exit the program." Choosing the first option results in a series of informational screens. The second choice "chains" to the NETMAKE program. Choosing the third option causes a menu of networks to appear (see Figure 4). After a network is chosen, the Simulations menu is displayed (see Figure 5). Here, the user may run the simulation with default parameter values, or change parameters prior to running the simulation. Parameters subject to manipulation include the size of the resource pool in all relations or in each relation (default = 24); the number of deals permitted all positions or particular positions (default = 1); the number of rounds per session (default = 25); the number of sessions (default = 10); and which of several available strategies would be employed by the actors. (Because actors are simulated, distinctions between the concepts of "strategy" and "tactic" are not crucial here.) After implementing any parameter changes, users are permitted to either run the simulation, make further parameter changes, or exit the program.

Figure 6, which illustrates the entire program structure, shows the MainMenu contained within the *Experiment* level, sitting atop all of the other levels of the program. When the user chooses to "Run a simulation," X-Net continues through the *Experiment* level and runs the SessionReady routine. This subprogram reads in the information on the chosen network, establishes which relations exist, sets up a graphical display for the user, and initializes several session-level variables. The graphical display, reproduced in Figure 7, places all network positions around an ellipse, draws lines to connect positions related in the network, labels position

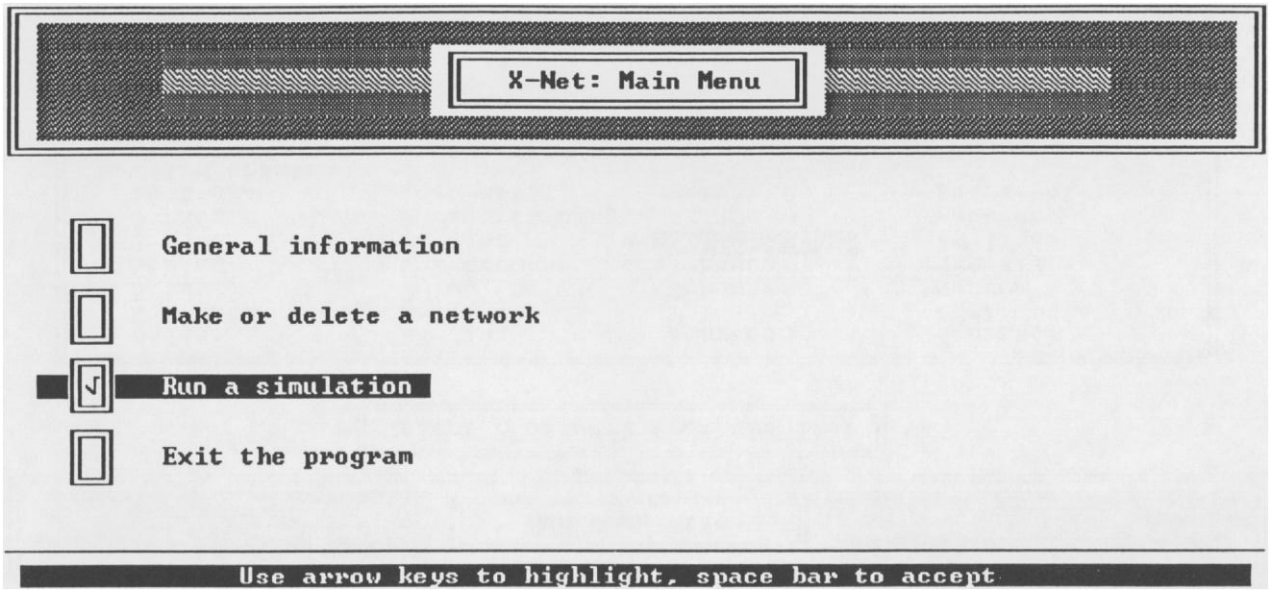

Figure 3
X-Net's Main Menu

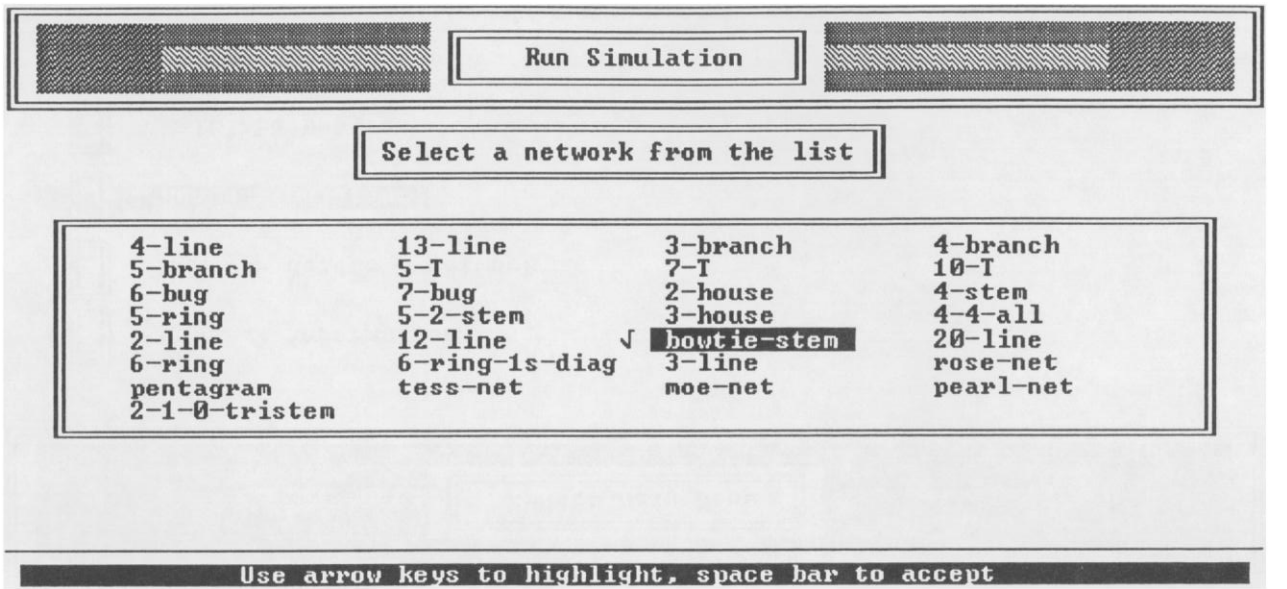

Figure 4
Network Selection Menu

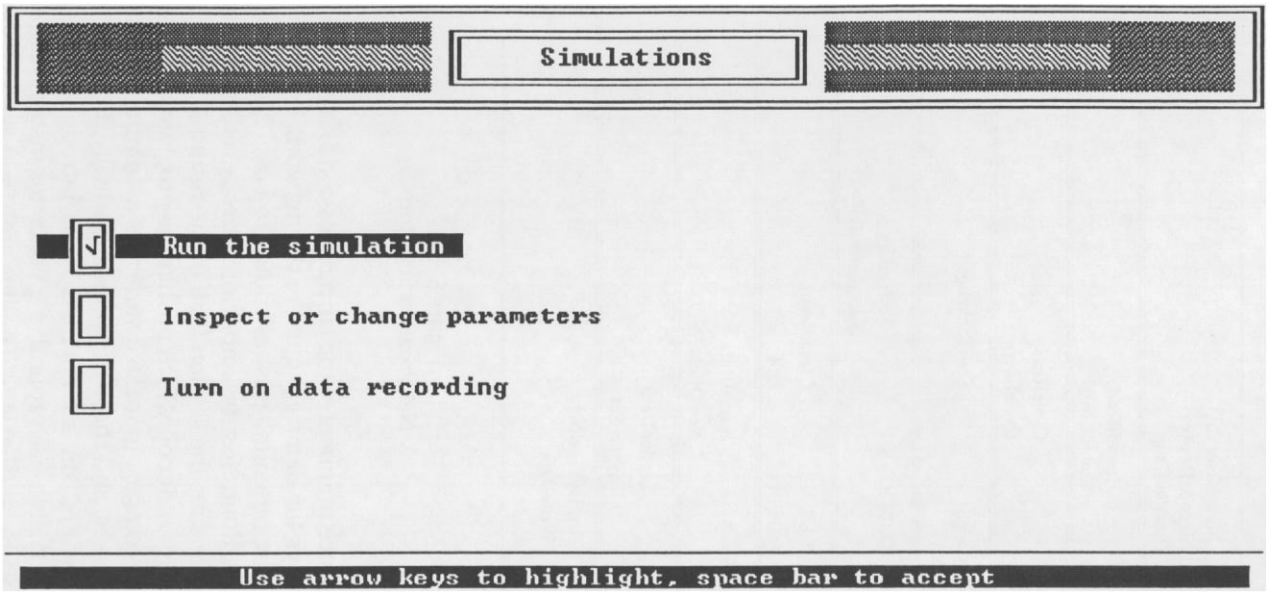

Figure 5
Simulation Control Menu

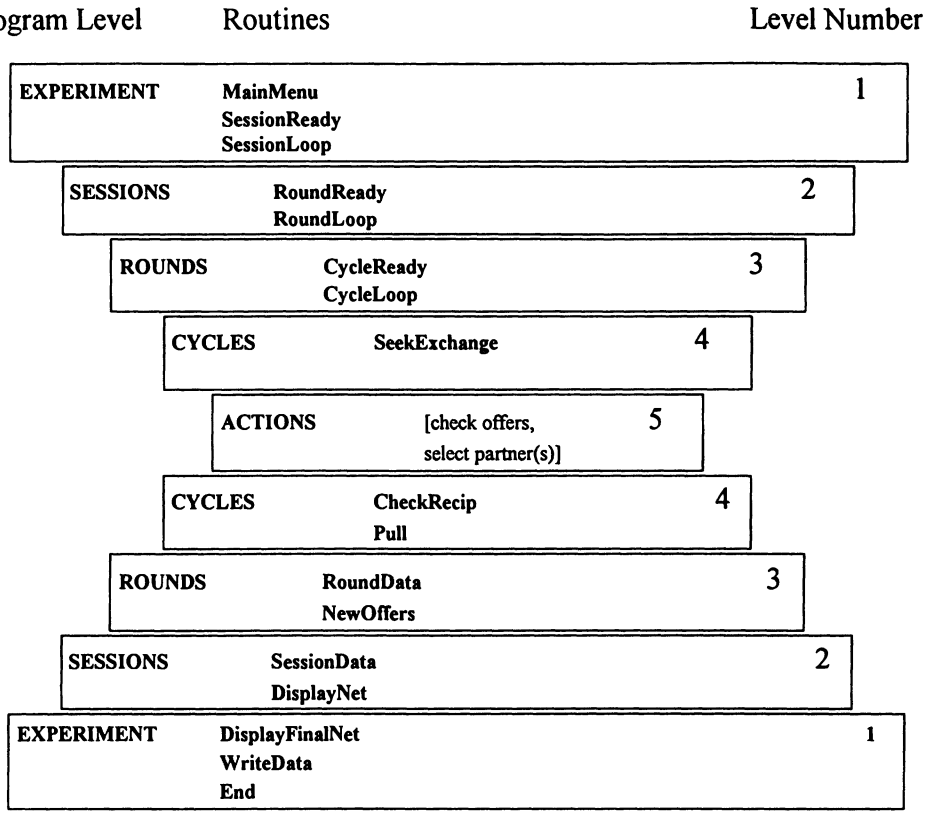

Figure 6
X-Net Program Structure

#1, shows the sequence numbers of the current session and round as the program executes, and displays the name assigned to the network. When the SessionLoop routine is entered, the program "pushes" down to Level 2, as shown in Figure 6.

At Level 2, *Sessions*, RoundReady simply initializes several data storage variables and actors' initial offers for the session (half of the resource pool size, by default) and then enters the RoundLoop which pushes the program down to Level 3. Here is where I found it necessary to make a variety of assumptions about negotiation and exchange processes, and these developed gradually, through trial and error. At some point, I settled on an approach that introduced one more logical programming level. Within each round, a given actor may go through a number of cycles of offers to others—that is, as if pulling each (and every) potential partner's name out of hat and checking to see if the other's offer is reasonable. Hence, I adopted the name *Cycles* for the fourth level. The *Rounds* level first prepares for

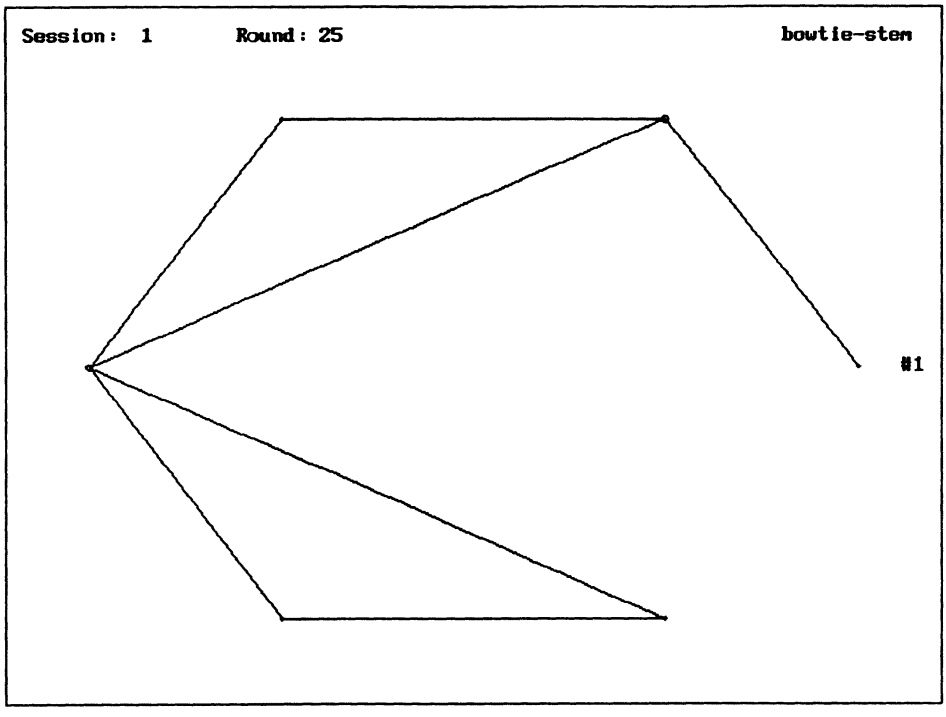

Figure 7
Early in a Simulation

these cycles with the CycleReady routine. Its procedures turned out to be somewhat elaborate. It "activates" actors who may have made early exchanges in the early cycles of a previous round and, thus, been "dormant" for later cycles. It also initializes variables that pertain to whether (1) a given other is sought for exchange, (2) another who is sought also seeks the given actor, and (3) a given strategy has been selected by the user. Then, the program enters the CycleLoop and pushes down to Level 4.

The *Cycles* level consists of three subprograms, starting with SeekExchange. This routine immediately enters the lowest level of the program, *Actions*, in which actors make judgments that depend upon their strategy as selected by the user. For example, one strategy flags those among an actor's alternative partners who have made acceptable offers, then notes (or "seeks exchange" with) a number of those others corresponding to the number of exchanges the actor is permitted in each round. By another strategy, compromises are calculated and then exchange is sought with the other(s) making the most profitable offer(s). After making this provisional selection of exchange partners, the program then "pops" up to the *Cycle* level (although we continue moving down in Figure 6), this time to run two more

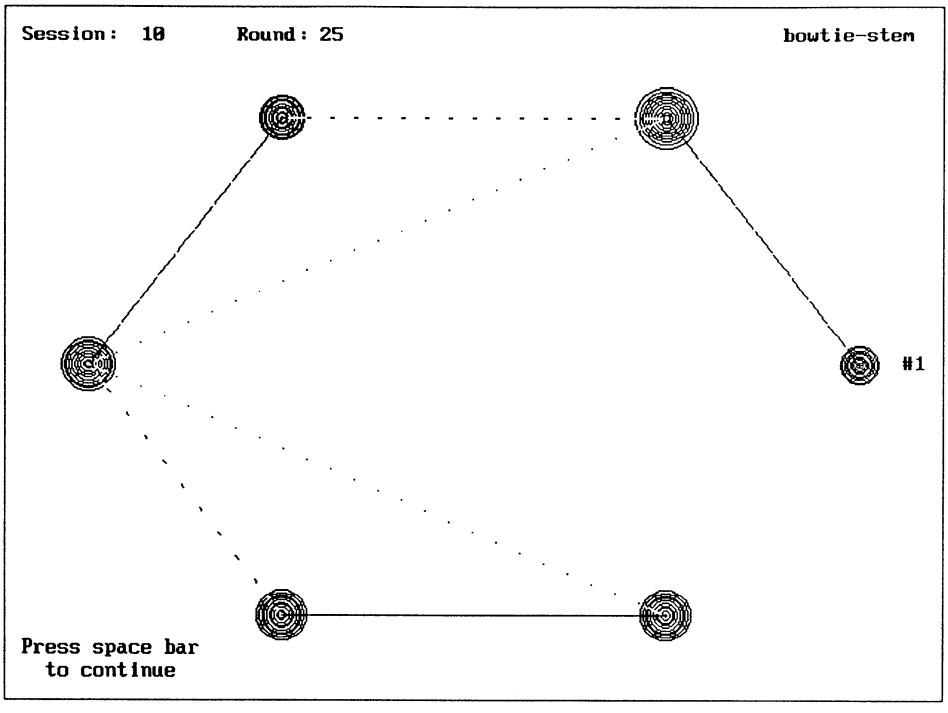

Figure 8
End of a Simulation

routines. CheckRecip moves through the network and flags relations in which both actors have chosen one another in the SeekExchange routine. When such mutual exchange-seeks are found, exchanges are declared to have occurred in those relations. Then the Pull routine temporarily "pulls" from the network those actors who have completed exchanges. The cycle continues to execute until all possible exchanges have occurred. This happens relatively quickly, for when a pair of actors has exchanged and been pulled from the network, those who remain no longer try to deal with them, thus speeding the process of searching for a viable partner in that round.

When no more exchanges can take place, the cycling has ended and, therefore, so does the round. The program pops back up to the *Rounds* level and stores the results of the last set of exchanges via the RoundData routine. Then the process enters the NewOffers routine in which actors decide how much they will offer in the next round of negotiations in light of what happened in the negotiations just completed.

Having decided upon NewOffers for the next round, the program loops back to CycleReady and CycleLoop, following the same process just described. When

the designated number of rounds has been completed, the program pops up to the *Sessions* level where it stores data for the now completed session and updates the graphical image on the screen. This consists of increasing the sizes of the network positions in proportion to the relative quantities of resources accumulated by each over its last five exchanges. When all of an experiment's sessions have been run, the program pops back up to the top *Experiment* level. There, DisplayFinalNet shows a final graphic display (Figure 8, for example) with results accumulated across all of the sessions, thereby providing an accurate depiction of average resource differentials by network position. It also redraws the network connections to indicate the frequencies of exchanges occurring in each relation: heavier lines indicate higher frequencies. Finally, if the user chose to do so earlier, WriteData stores numerical results to a disk file. The simulation run concludes with the opening "Main Menu" again displayed.

Developing the Graphical Display

Within a few weeks of starting to work on the program, I agreed to discuss it at a seminar. The seminar was to be on a Monday. The Friday before my talk, I realized that it would be nice to be able to demonstrate the simulation by showing networks on the screen and providing some type of dynamic image of resource accumulations. In my rush to have something on time, I stumbled on a nice display algorithm that is no doubt unoriginal, but works well for X-Net. The program uses trigonometric functions to evenly distribute the network positions—tiny circles, actually—around an imaginary ellipse on the screen. As shown in Figure 7, solid lines are added to indicate pairs of positions deemed to be in a relation. After each experiment, if a position earned resources, the size of the circle representing that position is increased. Actually, another circle is drawn around the previous circle, with the radius of the new circle proportional to the resources per exchange earned during the last five rounds of each session. At the conclusion of an Experiment, the solidity of the lines connecting actors is reduced according to the relative frequencies of exchanges between actors; the more frequently a relation is used, the more solid the line. There are sixteen gradations of line density, and these are normalized so that the most frequently used relation always has a solid line and the least often used relation is only sparsely dotted. The result of this *DisplayNet* routine is a representation that is easily interpretable for relatively small networks while still portraying a great deal of information. Because of this, I rarely opt to store numerical output.

Crucial Routines

Of the routines described above, *SeekExchange* and *NewOffers* are most crucial for determining the ultimate results from the simulations. *SeekExchange* controls the partner-selection process, whereas *NewOffers* controls offer-revision strategies. As one indication of their importance, these routines comprise less than 4% of the program code, but they took at least as long to write as the remaining 96%.

Moreover, these two routines were the sites for the new assumptions that were needed to compensate for the theory's inability to fully determine the program.

SeekExchange marks the beginning of every negotiation cycle. At the instant the program engages this routine, each actor already knows its outgoing and incoming offers for all of its relations. The problem at this point is to instruct actors how to deal with their information. I had anticipated having many choices when it came to programming concession strategies—that is, how actors modify their offers from round to round. As obvious at it now seems in retrospect, it did not occur to me that I would also have to devote a significant amount of time to devising another algorithm for having actors arbitrate multiple offers.

One simplification assumes that actors cannot change any of their offers during a round, even if there are multiple cycles within the round. A second checks to see whether mutual offers add up to the number of available units in the pool. Given a 24-unit pool, if actor A offers 12 units to B, and actor B offers to relinquish only 10, we find that $10 + 12 = 22$, two less than the 24 units available. A and B thus cannot complete a deal this round. In contrast, when offers sum to or exceed the pool, they are said to be *complementary*.

In every round, for each actor, *SeekExchange* determines and tallies a list of others with whom an actor's offers are complimentary. Next, it sends that list to a routine that returns it shuffled. Finally, given that M is the maximum number of deals an actor can make in a given round, each actor designates the first M others on its list as "others with which I will seek to exchange." Control then moves to the *CheckRecip* routine. Here, the network is scanned to find pairs of related actors who explicitly sought exchange *from each other*. Note that *SeekExchange* guarantees that no actor will have more than M reciprocations, and so every reciprocated exchange-seek is declared a completed deal. The number of resource units received by each actor is noted.

SeekExchange and *CheckRecip* do not guarantee that all actors who can make deals actually do so. For instance, consider the line of four actors A—B—C—D, each of whom can make one deal per round. In the first round, suppose each actor offers 12—12 pool divisions with those to whom it is related. All offers are thus complimentary. A has no choice but to seek exchange from B, and D from C. However, suppose that when B's shuffled list comes back, A is at the top, and when C's comes back, B is at the top. A and B complete a deal, but despite D's availability, C is left hanging. This is where the cycling process enters. The *Pull* routine will remove A and B from the list of active negotiators and determine that C and D are still active in this round. The program loops back to *SeekExchange* and *CheckRecip*, C and D make their deal, and *Pull* removes them from the active list. With the active list now empty, the round is over.

This algorithm emerged from a trial-and-error process. There are probably alternatives, and I recall my intuition telling me that some of them should have been able to accomplish the same tasks in a simpler way. However, each alternative that I tried ended up either introducing biases in actors' partner choices, being at least as complex as the method described above, or operating less efficiently.

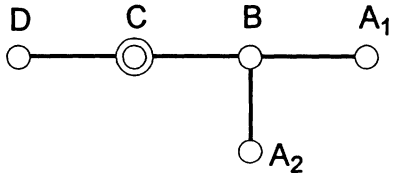

Figure 9
C May Exchange Twice

NewOffers modifies actors offers to others. It is enacted at the end of every exchange round, after all deals have been made. Once again, there are many possibilities for negotiation algorithms. After its initial development, some of my work with X-Net involved exploring the effects of different strategies when enacted by those in various network positions (e.g., Markovsky et al. 1993). In the early stages, however, I wanted to introduce the simplest possible strategy that would permit replication of laboratory experiments (Markovsky 1992). Generally, this meant specifying a minimal strategy that would allow actors in advantageous positions to profit from their structural advantages and that would prevent those in disadvantageous positions from giving up and withdrawing.

The first thing I tried was to simply have actors raise their offers to others by one unit if they were excluded from exchange on a previous round, and lower their offers by a unit if they were not excluded. This method was used in simulations reported by Cook et al. (1983). Although it is indeed simple, the strategy only works if actors are permitted at most one exchange per round. For example, in Figure 9, if actor C is permitted two exchanges but only completes one deal with D and cannot complete any with B, then C would have to be designated as excluded from exchange. In the next round, C would then have to make a better offer to D, despite having just successfully completed an exchange with D. Experiments with human subjects show that positions like C are members of two domains (Markovsky et al. 1988; the Appendix provides criteria for identifying domains). In this case, C has equal power in the domain shared with D, and low power in the domain shared with B and the As. Unless C is "intelligent" enough to somehow make these distinctions, it will suffer not only in exchanges with B but in those with D as well.

After exploring a variety of possibilities, I could devise only one algorithm that was both useful and extremely simple: (1) If an actor makes all of the deals it seeks, then it decreases all of its offers in the next round; (2) if the actor fails to make all the deals it seeks, then the actor (a) decreases its offers to those with whom it completed a deal, and (b) increases its offers to those with whom it did not. *NewOffers* thus satisfies two more of NET's scope conditions: SC2 and SC3 require that inclusion in exchange leads to decreased offers, and exclusion leads to increases. SC4 (accept best offer, randomly choose among ties) is not always

satisfied by X-Net decision strategy options. In the default strategy, for example, actors select randomly among not only tied offers but *any* offers that complement their own. They *satisfice* rather than maximize. Other user-selectable strategies do fully satisfy SC4. In many cases, having a fairly large number of rounds compensates for inefficiencies in the first strategy, and so long-run outcomes are not differentially affected by these alternative partner-selection strategies. This is not true in general, however. One of the major findings to emerge from X-Net is that the sensitivity of exchange outcomes to the strategic behaviors of actors is highly variable depending on structural features of the network.

With the addition of several "housekeeping" facilities, the completion of these crucial routines resulted in a workable program. I then spent some time smoothing out the user interface, developing the menu-generating routine, cleaning up the program code, and inserting more descriptive comments for people who may wish to read the program itself. I also created 20 or so starter networks that would appear on the Select a Network menu, wrote some external documentation, and wrote routines for calling up an online program description and for storing the numerical simulation results in formatted data files.

Starting with a later version of the program, X-Net reads a list of program parameter defaults from a separate text file. Thus, following editing instructions given in this "XNET.PAR" file, the user may establish his or her own program defaults. At the same time, I added the menuing system whereby, with the program running, the user can change the value of any relation's resource pool, the maximum exchanges per round for any position, and the number of rounds and experiments. Further, I provided user-selectable alternative decision strategies for actors. For example, rather than satisficing—that is, selecting randomly among complementary offers regardless of their values—actors may instead first reach a compromise in all of their relations by splitting the difference (making equal compromises) when offers are initially noncomplementary. Then, depending on the strategy chosen, actors either select randomly among their relations for explicit exchange-seeks or rank the offers they receive and only seek exchange from others offering the most. The effects of these relatively minor strategic changes have proven to be nil in some networks but rather profound in others. Again, the program is superb for illustrating the conclusion, carried forward from the first BASIC exchange simulations, that neither structure nor strategy alone is sufficient to predict exchange outcomes.

CONCLUSIONS: LINKING THEORY, SIMULATION, AND REALITY

While having a good deal of fun with this project, I have also been struggling with deeper questions: What is X-Net's relevance to NET or vice versa, and what is its connection to reality? It is a relatively simple matter to slap together a set of routines and produce a dynamic system. Toward what ends such activities are useful (aside from having fun) is another matter entirely.

Mathematics provides a variety of analytic approaches for generating outcomes ("solutions") for models of systems. For social exchange networks, too, there are alternatives to GPI (e.g., Bienenstock and Bonacich 1992; Friedkin 1992; Marsden 1983; Cook and Yamagishi 1992). Each provides a model that generates predictions for resource distributions based on explicit assumptions or axioms. Simulations, it may be argued, prove especially useful when such solutions cannot be achieved or do not exist. Ironically, an analytic solution may be unavailable because our knowledge about the phenomenon is poor or because our knowledge is so rich. In the first case, the lack of knowledge prevents explication of a model. In the second case, we want the model to be so complex that mathematical analysis is prohibited.

Traditionally, simulations attempt to model the essential components of a real-world system, to illustrate the workings of the system over time, and to generate summary information about the system and the behavior of its elements. To the extent that the relevant variables and relationships among them have been represented accurately, the simulation should generate outcomes that approximate those of the system it models. When one or more of those connections is probabilistic, then for a given set of starting conditions, over many runs the simulation should yield correct predictions for the relative likelihoods of the various possible outcomes.

This is, perhaps, the most common view of simulations, but it reveals only half the picture. Instead of operating on elements and connections that represent components of some real-world system, simulations may also operate on components taken from one or more *theoretical* systems (Hanneman 1988; Fararo and Hummon 1994). This need not be as divorced from reality as it sounds, for the theories that are being simulated are themselves presumably interpretable for empirical phenomena. What this amounts to is not so much a different type of simulation as a different way of regarding simulations. Rather than a method for examining empirical events and processes, simulations become a method for doing *theoretical analysis.*

The distinction between theoretical analysis and empirical analysis was made eminently clear by Jasso (1988). Whereas the latter involves primarily the examination of empirical data, theoretical analysis includes, among other activities, deriving logically implied consequences from a minimal set of premises. According to Jasso, this is the activity of longest duration and comprises most of what a mathematical theorist does. It is also a task that I believe may be greatly facilitated by the use of computer simulations. I would even argue that as our theories become more cumulative and sophisticated, simulations will become essential for exploring their logical consequences prior to empirical testing—just as in a number of branches of the physical sciences. In fact, as theoretical complexity increases and the likelihood of devising analytic solutions diminishes, simulation becomes the only tenable method for deriving testable theoretical consequences.

For me, the network exchange simulations illustrated ways that even a small-scale simulation project can yield payoffs. First, I developed a deeper appreciation of the beauty of simple and clear models that generate rich and varied

consequences. Sociology already has plenty of complex and nonparsimonious models for explaining phenomena. To generate a broad range of predictions from small, carefully selected sets of assumptions is an alternative with greater potential for generality and cumulation (Markovsky 1994). Simulations thus provide a means for exploring the *range* of a theory's explanations and predictions.

Additionally, translating even a relatively simple theory into a series of program steps imposes a strict form of honesty and integrity upon the theory. For the same reason that it is much easier to *claim* that your theory logically explains something than it is to *prove* it, it is far easier to *imagine* a simulation than it is to *construct* one that does something interesting. One is constantly forced to question theoretical assumptions when fitting together the pieces of the simulation. To varying degrees, we are probably all guilty of overstating the explanatory power of our favorite theories. Simulations-as-translated-theories do only what they do, and nothing more. Of course, nothing prevents an author from overgeneralizing his or her simulation. The added rigor imposed by the logic of a structured programming language should, however, ease the skeptic's task. The burden of proof is on the theorist to show how real-world elements manifest elements of the simulation. If the theory motivating the simulation is vague, such proof will not be possible.

Third, X-Net has led to insights that, in turn, have led to a broadening of the scope and refinement of the predictions of Network Exchange Theory. For example, several years ago, my colleague Michael Lovaglia was playing around with X-Net. He noticed a phenomenon that I had also found in a variety of types of networks: in certain networks whose positions were predicted by NET (the 1988 version) to be equal in power, some network positions would *consistently* have small but noticeable advantages over others. Whereas I had ignored those small differences under the assumption they were artifacts or program bugs, Lovaglia believed that the effect was *real*—that is, theoretically and empirically relevant. As it turned out, we later discovered the structural basis for those "weak power" effects (see Appendix, part 2) and went on to develop an analytic model and experimental tests that indeed verified the weak-power predictions (Markovsky 1992; Markovsky et al. 1993). Furthermore, predictions from the new analytic model converged precisely on X-Net outcomes.

Finally, and perhaps most controversially, one use of simulations can be to help us think *less* concretely about the world; to adopt a more abstracting and generalizing mode as opposed to a more journalistic and interpretive one. Each mode has its own interesting and useful products. My sense is, however, that sociology—and certainly the world outside of sociology—already appreciate the work of those who, for example, describe public opinions and beliefs. Less appreciated, and arguably far more beneficial, are the fruits of theoretical labor in the social sciences—theories that *explain* the emergence of opinion and belief systems, for example, rather than just describing them. We can learn much from piecing together an artificial social process from scratch (Webster and Kervin 1971). Some of what we learn may be useful for explaining American social phenomena, social phenomena in other nations, or perhaps even phenomena in nations not

yet existing or nations on other worlds. It is also quite a revelation to find that emergent and multilevel phenomena need not be mystical constructs. We can easily build our own, and these constructions teach us about emergent and multilevel phenomena in vivo.

APPENDIX

Network Exchange Theory

Part 1: Graph-theoretic Power Index (gpi)

Definitions of Key Terms

actor:	an entity with the capacity to observe conditions, make judgments, and act upon them
position:	a location that may be occupied by an actor
relation:	an exchange potential between a pair of positions
network:	a set of positions, their relations, and the actors in positions
exchange:	a mutually agreed-upon distribution of valued resources between actors.
power:	a structurally determined potential for obtaining relatively favorable resource levels

Scope Conditions

(1) all actors use identical strategies in negotiating exchanges
(2) actors consistently excluded from exchanges raise their offers
(3) those consistently included in exchanges lower their offers
(4) actors accept the best offer they receive and choose randomly in deciding among tied best offers
(5) each position is related to, and seeks exchange with, one or more other positions
(6) exchange rounds begin with equal pools of positively valued resource units in every relation
(7) two positions receive resources from their common pool if and only if they exchange

Terms in GPI Calculations

i: position in the network
e: number of others with which an actor may exchange (once each) in an exchange round
d: domain in the network. To calculate domain memberships, let i and j indicate two positions, and an e^+ position is one having more than e relations. Given the set V of all positions on a path between i and j, i and j are in the same domain if and only if there exists a path such that either (1) $V = \{\varnothing\}$, or (2) all members of V are e^+ positions.
k: length of a path. For example, $k = 3$ for the path from A to D in network A—B—C—D—E. Two paths from a position are nonintersecting when they

have only that position in common. Thus, C has two nonintersecting paths of length 2: C—B—A and C—D—E.

h: the longest non-intersecting path from a position

m_{idk}: the number of nonintersecting paths of length k in domain d from position i,

p_{id}: power index for position i in domain d.

Axioms

Axiom 1: $p_{id}(e_d) = (1/e_d) \sum_{k=1}^{h} (-1)^{(k-1)} m_{idk}$

Axiom 2: i seeks exchange with j if and only if $p_i > p_j$ or if $(p_i - p_j) \geq (p_i - p_k)$ for all k related to i.

Axiom 3: i and j can exchange only if each seeks exchange with the other.

Axiom 4: if i and j exchange, then i receives more resources than j if and only if $p_i > p_j$.

Part 2: Likelihood of Inclusion and "Weak Power"

This extension of NET builds on the GPI to generate refined predictions for weak power differentials. Ongoing exchanges can produce temporary changes in the number of an actor's available exchange partners, in the number of the partners' partners, and so on, and this formulation is able to take into account temporary power shifts that arise as some actors exchange in a given time period and leave behind altered substructures.

Step 1: Apply Axiom 1 to calculate initial GPI values for each position.

Step 2: Apply Axiom 2 to determine which positions seek exchange with which others.

Step 3: Apply Axiom 3 to identify and remove relations with nonmutual exchange-seeks.

Step 4: Apply Axiom 1 to the resulting substructures.

Step 5: Repeat Steps 1-4 until the GPI values stabilize.

If GPI $\neq 1$ for any positions in the network (or a given substructure), then resource distributions in the network (or substructure) will be ordered by GPI and approach maximum differentiation.

If GPI $= 1$ for all positions in the network (or a given substructure), then resource distributions in the network (or substructure) will be ordered by the likelihood of i's inclusion in exchange.

The likelihood of inclusion, l_i, is calculated as follows:

> Under an equiprobability assumption, determine the probability that the actor in position i and actors in each of its relations will mutually seek exchange (where actors are allocated e exchange-seeks). l_i is then the sum of these probabilities across i's relations.

Part 3: Exact Predictions for Weak Power Networks

This recent extension provides supplementary theoretical assumptions that build on the weak power formulation. An alternative formulation has also been

published by Skvoretz and Willer (1993); however, this version provides more accurate predictions. With this extension, we can derive exact predictions for exchange outcomes. It employs a modified *resistance* model for predicting negotiation outcomes. Then, modifications to the resistance model account for the assumed effects of (1) inclusion likelihoods, using the *Resistance-Likelihood Assumption*, and (2) relative degree—the number of an actors' direct relations relative to another's—using the *Resistance-Degree Assumption*. The *Profit Theorems* are used to generate the actual predictions for exchange outcomes at each network position.

P: total points available in resource pool
P_i: i's profit from exchange
M_i: i's maximum expectation or "best hope" for exchange profit
C_i: i's worst fear or "conflict outcome" for exchange profit
R_i: i's resistance to a given exchange profit P_i
t_i: i's number of network ties
d_{ij}: i's relative degree in the i–j relation:

Resistance Assumption:
$$R_i = \frac{M_i - P_i}{P_i - C_i}$$

Equiresistance Assumption: In equilibrated i–j exchanges, $P_j = P - P_i$, and P_i is obtained by solving:
$$\frac{M_i - P}{P_i - C_i} = \frac{M_j - P_j}{P_j - C_j}$$

Resistance-Likelihood Assumption:

(a) $$C_i = \frac{P}{2} l_i$$

(b) $$M_i = \frac{P}{2}(l_i + 1)$$

Resistance-Degree Assumption:
$$C_{ij} = \frac{P}{2} l_i d_{ij}$$

Profit Theorems: From the Equiresistance and Resistance-Likelihood Assumptions, we derive:

$$P_{ij} = (P + C_{ij} - C_{ji}) / 2$$
$$P_{ji} = P - P_{ij}$$

Acknowledgment: Some of this research was facilitated by National Science Foundation Grants SES 88-08289 and 90-22935. Most essential were the intellectual contributions of Michael Lovaglia (who also provided comments on an earlier draft), David Willer, and John Skvoretz.

NOTES

1. The X-Net program described herein is available upon request at no charge.
2. After a series of rapid upgrades leading to version 4.5, Microsoft ceased improving on QuickBASIC around 1988. Over the following two years, it was eclipsed by the more powerful and expensive BASIC Professional Development System, a superset of QuickBASIC with an identical interface. Shortly thereafter, BASIC PDS upgrades stopped and the price of the last version (7.1) was greatly reduced, apparently to make way for VisualBasic, designed to work in conjunction with Microsoft Windows.

REFERENCES

Bienenstock, Elisa Jayne, and Phillip Bonacich. 1992. "The Core as a Solution to Exclusionary Networks." *Social Networks* 14: 231-243.

Cook, Karen S., Richard M. Emerson, Mary R. Gillmore, and Toshio Yamagishi. 1983. "The Distribution of Power in Exchange Networks: Theory and Experimental Results." *American Journal of Sociology* 89: 275-305.

Cook, Karen S., and Toshio Yamagishi. 1992. "Power in Exchange Networks: A Power-Dependence Formulation." *Social Networks* 14: 245-265.

Emerson, Richard. 1972. "Exchange Theory, Part II: Exchange Relations and Network Structures." Pp. 58-87 in *Sociological Theories in Progress,* Volume 2, edited by Joseph Berger, Morris Zelditch, Jr., and Bo Anderson. New York: Houghton-Mifflin.

Fararo, Thomas J., and Norman P. Hummon. 1994. "Discrete Event Simulation and Theoretical Models in Sociology." Pp. 25-66 in *Advances in Group Processes,* Volume 11, edited by Barry Markovsky, Karen Heimer, and Jodi O'Brien. Greenwich, CT: JAI Press.

Friedkin, Noah. 1992. "An Expected Value Model of Social Power: Predictions for Selected Exchange Networks." *Social Networks* 14: 213-229.

Hanneman, Robert. 1988. *Computer-assisted Theory Building.* Newbury Park, CA: Sage.

Jasso, Guillermina. 1988. "Principles of Theoretical Analysis." *Sociological Theory* 6: 1-20.

Lovaglia, Michael, John Skvoretz, David Willer, and Barry Markovsky. In press. "Negotiated Exchanges in Social Networks." *Social Forces.*

Lovaglia, Michael J., John Skvoretz, Barry Markovsky, and David Willer. 1995. "Assessing Fundamental Power Differences in Exchange Networks: Iterative GPI." *Current Research in Social Psychology* 1(2): 8-16, http://www.uiowa.edu/~grpproc.

Markovsky, Barry. 1987. "Toward Multilevel Sociological Theories: Simulations of Actor and Network Effects." *Sociological Theory* 5: 100-115.

Markovsky, Barry. 1992. "Network Exchange Outcomes: Limits of Predictability." *Social Networks* 14: 267-286.

Markovsky, Barry. 1994. "The Structure of Theories." Pp. 3-24 in *Group Processes: Sociological Analyses,* edited by Martha Foschi and Edward J. Lawler. Chicago: Nelson-Hall.

Markovsky, Barry, David Willer and Travis Patton. 1988. "Power in Exchange Networks." *American Sociological Review* 53: 220-236.

Markovsky, B., J. Skvoretz, D. Willer, M. Lovaglia, and J. Erger. 1993. "The Seeds of Weak Power: An Extension of Network Exchange Theory." *American Sociological Review* 58: 197-209.

Marsden, Peter V. 1983. "Restricted Access in Networks and Models of Power." *American Journal of Sociology* 88: 686-717.

Skvoretz, John, and David Willer. 1993. "Exclusion and Power: A Test of Four Theories of Power in Exchange Networks." *American Sociological Review* 58: 801-818.

Webster, Murray, Jr., and John B. Kervin. 1971. "Artificiality in Experimental Sociology." *Canadian Review of Sociology and Anthropology* 8: 263-272.

Willer, David. 1986. "Vulnerability and the Location of Power Positions." *American Journal of Sociology* 92: 441-448.

Willer, David, and Bo Anderson. 1981. *Networks, Exchange and Coercion: The Elementary Theory and its Applications.* New York: Elsevier.

Willer, David, and B. Markovsky. 1993. "Elementary Theory: Its Development and Research Program." Pp. 323-363 in *Theoretical Research Programs: Studies in the Growth of Theory,* edited by J. Berge. Stanford, CA: Stanford University Press.