its present power would not have sufficed for what he had in mind. Choosing to calculate the motions of the inner planets that whiz around their orbits in as little as 86 days, he took the faster but more approximate approach of neglecting motions on a time scale of less than 10 years or so. In effect, he smeared the mass of each planet around its orbital path and treated only the long-term behavior of this orbital "wire." Using this averaging approximation, he could take a 500-year rather than a 32-day time step in his 200-million-year calculation on a conventional supercomputer. The Orrery would have taken a couple of years to make the same calculation in its own way.

Laskar found several indications of chaos. Over long time scales the inner planets displayed many periodic gravitational interactions called resonances. The more resonances, the more likely chaotic behavior. The pattern of orbital variations of the inner planets shifted from time to time as those of the outer planets remained steady. And two nearly identical test orbits diverged exponentially with time, demonstrating that precise predictions of solar system behavior are impossible beyond 10 million years into the future. "It's a nice result," says Wisdom, "and the solar system probably is chaotic, but it needs to be confirmed with a different method. Everyone knows averaging works, but no one can prove it."

There are other new signs of chaos in the solar system. Martin Duncan of Queen's University at Kingston, Ontario, and Thomas Quinn and Scott Tremaine of the University of Toronto have used a different shortcut—distinguishing between crucial near-planet and less influential distant interactions—to search for orbits between the planets that might still harbor as yet undetected debris from the formation of the solar system. They found that many orbits lying between Uranus and Neptune become chaotic, supporting Sussman and Wisdom's suggestion that Pluto may have entered its present odd orbit through chaotic shifts.

The principal investigators of project LONGSTOP (Long-term Gravitational Stability Test for the Outer Planets)—Anna Nobili and Andrea Milani of the University of Pisa and Mario Carpino of the Astronomical Observatory of Milan—have run a 100-million-year calculation of the motion of the outer planets on a standard computer. The run is a short one, but by careful analysis they believe they see evidence of chaos even in the massive outer planets. These planets do not seem to have a limited number of orbital variations having discrete periods, as simple interactions of resonances would induce. Instead, there is a continuous variation of periods characteristic of chaos.

"In every case of a long-term integration," notes Tremaine, "researchers have found pretty strong evidence of some sort of chaotic behavior." That suggests to him that chaos is pervasive. Speculating a little further, Tremaine notes that the general character of our solar system, in which there are nine systematically spaced major planets with little debris between them, might have been determined by chaotic processes. Perhaps only those bodies whose chaotic behavior has tight limits survive without being thrown into collision courses with other bodies. Once the recent results are confirmed, the next step will be to investigate the exact limits of present-day chaos and what confines it to those limits. Approximation methods run on conventional supercomputers will help, but the race will probably be won by the next generation of computer dedicated to simulating the solar system. ■ RICHARD A. KERR

ADDITIONAL READING

J. Laskar, "A numerical experiment on the chaotic behavior of the solar system," *Nature* **338**, 237 (1989).

A. M. Nobili, M. Carpino, A. Milani, "Fundamental frequencies and small divisors in the orbits of the outer planets," *Astron. Astrophys.* **210**, 313 (1989).

G. J. Sussman and J. Wisdom, "Numerical evidence that the motion of Pluto is chaotic," *Science* **241**, 433 (1988).

PARC Brings Adam Smith to Computing

Part computer virus and part market theory, Spawn is both an efficiency tool and a laboratory for experimental economics

SOMEHOW, IT COMES AS NO SURPRISE to realize that Spawn is a housebroken computer virus. This is the Xerox Palo Alto Research Center (PARC), after all, the rambling hillside laboratory that has already brought forth such inspired flights of lunacy as the personal computer and the "Macintosh" graphics interface. "Spawn is a *useful* computer virus," laughs Bernardo Huberman, the Argentine-born physicist who leads the group* that is developing the system. Indeed, Spawn is able to fan out through a computer network and infect the machines in just such a way that they all do productive work together—without the help of a central planner.

Huberman, not a man to contain his enthusiasm, is deep into an explanation of Spawn as soon as a visitor reaches his seat.

"What we want to study is how to design programs that collaborate and cooperate on a given problem," he says. Look around the laboratory: big-screen, high-powered graphics workstations everywhere, all connected by a high-speed data network. It is a scene that is increasingly common both in the business world and in academia. And yet at any given time, says Huberman, you will always find a few people running problems that strain their computers to the limit, while everyone else is just typing away on low-intensity applications such as word processing, or even letting their machines sit idle. What's needed is a way for the high-intensity users to capture some of that wasted capacity.

And that, he says, is precisely what he and his colleagues have designed Spawn to do. Basically, it balances the load by setting up a kind of automated, on-line marketplace within the network. And in the process, not incidentally, it provides them with a new kind of laboratory for empirical research into economics and ecology.

For specificity, says Huberman, he and his colleagues have tested the system on just two problems: the formatting of large, complex documents, and Monte Carlo calculations, the latter being a statistical approximation technique widely used by physicists. But they could just have well worked on the factorization of large numbers, or cryptography, or indeed, any other problem that can easily be broken into pieces. Whatever the problem, he says, the key issue is to figure out how the computers are supposed to organize themselves to accomplish the task.

The most obvious solution would be to institute some kind of central planning mechanism, using one master computer to give detailed instructions to all the subsidiary computers. But, when collaborative computers systems are actually designed this way, they quickly bog down. The machines spend so much time in communication and coordination with the central computer that they have very little time left to do the work. The trick, says Huberman, is to get the

*Carl A. Waldspurger, Tad Hogg, Jeffrey O. Kephart, and Scott Stornetta.

computers to coordinate with each other *without* centralized planning. Indeed, under the rubric of "distributed computing," this is emerging as a major issue for computer science as a whole.

Enter market theory. If Adam Smith's "invisible hand" of supply and demand allows humans to coordinate their economic activity without centralized planning, then it should do the same for computers trying to coordinate their computational activity. Spawn is not the first program to pursue this idea, says Huberman—the Enterprise system developed in the early 1980s at the Massachusetts Institute of Technology was a notable predecessor—but it does carry the idea farther than ever before.

To see how it works, he says, imagine that you want to carry out a big Monte Carlo calculation. Sitting down at your workstation, you first give Spawn the project's "budget"—that is, its priority level measured in dollars. (In the prototype system these "dollars" are actually just arbitrary tokens used internally by the computers themselves. But in a working system, says Huberman, they could just as easily represent real money.) Then, once the problem itself has been specified, you simply tell Spawn to go ahead. While it is working there is little to see; you could spend the time writing a report in another on-screen window, or you could get a cup of coffee.

In the background, however, the software is busily chopping the original calculation into pieces and spawning a series of independent subcalculations, each with its own budget. (It is this procedure that gives the program its name.) These subprocesses, in turn, will be sent out over the network to take up residence in idle machines, where they will commandeer the spare capacity to actually get the work done. (Thus the reference to useful computer viruses.)

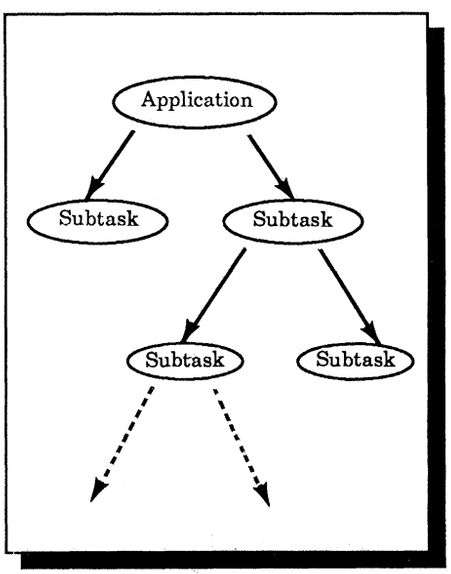
Rather than simply invading computers at random, however, these freshly spawned subprocesses follow the protocols of the market. Before they go anywhere, they send out what are, in effect, sealed bids offering to buy processing time in blocks of 60 seconds apiece. Meanwhile, each idle or underutilized machine on the network is likewise broadcasting an offer to sell as much time as it thinks it can spare. Just as in a human market, a price is quickly agreed upon, and the winning subprocess moves in for the allotted time. If its workload is still too high—and if it still has any money left—it can spawn a new generation of subprocesses and start a whole new round of bidding.

Now, if you were the only person on the network trying to use Spawn, says Huberman, then none of this bidding paraphernalia would make much difference to you. But suppose someone down the hall tries to launch a separate Spawn project unbeknownst to you. For that matter, suppose you yourself wanted to open up a few more calculations before the first one were finished. The market mechanism continues to produce an efficient allocation of resources no matter how many users are bidding.

Indeed, says Huberman, experiments with Spawn running on workstations around PARC have confirmed that the automated marketplace behaves just about as one would expect. The more subprocesses that are engaged in bidding for computer time, for example, the higher is the going price for that time. So the resources are automatically allocated to the processes with the highest priority. By the same token, if one subprocess enters the market with lots of money, it quickly bids up the price and beats out processes with a lower priority.

In addition, says Huberman, the automated market is efficient in a very practical sense: the network as a whole spends less than 10% of its time on the overhead of bidding, leaving more than 90% of the computer time free for doing useful work.

Of course, there are some important caveats to all this, warns Huberman. Spawn is less than a year old, for example. It is not a product, nor does Xerox currently have any plans to make it a product. "At the moment it's the software equivalent of a physics lab with wires hanging out all over the place," he says. Perhaps the most notable gap lies in the area of security. "We're all nice and friendly here," he says. But a practical system will have to contain strong safeguards to keep one person from using Spawn to break into another's computer with malicious intent. Such a system will also have to have safeguards to prevent a bug in a Spawn subprocess from destroying its host computer software accidentally.



Spawning. *The application sends subtasks out to different machines—if they win the bid. This can repeat indefinitely.*

Still, it is entertaining to speculate where Spawn and its descendants might lead. In its current rough state, for example, the Spawn user has to issue some rather cumbersome commands to get it to work. However, Spawn team member Carl A. Waldspurger, who is now a graduate student at MIT, is working on a new computer language specialized for Spawn-style distributed computing. Known as ARGUS, this language would allow programmers to couple applications to Spawn in an utterly transparent way: all the user would see on the screen is the application itself. He or she would neither know nor care which machine the application was actually running on.

In fact, says Huberman, when you follow that logic it is only one step to the high-speed national networks now being planned by the National Science Foundation and others. If that network contained Spawn or some facility like Spawn, then any desktop personal computer hooking into it would have exactly the same power as a supercomputer—provided, of course, that the supercomputer were also connected to the network, and that the desktop user were willing to pay for the time.

Meanwhile, the PARC group is also beginning to use Spawn for their real research interest: developing new kinds of computer models for economics and ecology. Indeed, Spawn itself grew out of their ongoing efforts to understand the dynamical behavior of large systems of competing agents, as in the stock market. And that work, in turn, is part of a much broader, if embryonic, movement among physicists, biologists, computer scientists, and economists to study what some researchers are calling "the emergent properties of complex systems."

For example, says Huberman, Spawn could be used to test what happens to the market equilibrium when each agent tries to anticipate what the others will be doing. There are indications from PARC's earlier work that this could be destabilizing. Is it?

Or, says Huberman, one could study the onset of diversity: "Just by chance, one machine might notice that its last five jobs all involved routines for floating point processes. So it might decide to cache those routines in memory and use that fact in future bids, giving itself a comparative advantage for floating point tasks."

In sum, it is still too early to know where Spawn will really lead. But as the economists and industry watchers contacted by *Science* agree, it is an intriguing beginning.

■ M. MITCHELL WALDROP