Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Parallel Computing 29 (2003) 1281–1295

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Geocomputation's future at the extremes: high performance computing and nanoclients

K.C. Clarke

Department of Geography/NCGIA, University of California, Santa Barbara, CA 93106-4060, USA

Received 11 June 2002; accepted 27 March 2003

Abstract

Recent developments in computing are reviewed, with consideration given to the nature of the human–computer interface and its development. The emergence of the desktop metaphor and the Windows–Icons–Menus–Pointers interface, while suitable for most computing tasks, has now played out its course as far as innovation in hardware is concerned. The next phase of computing is only now emerging, and is being fed by changes at the "extremes" of computing, the parallel supercomputer at one end, and the extremely thin or "nano" client at the other. The paradox, of fat servers and ultrathin clients implies a further separation toward the extremes as systems develop. Examples are given of nanoclients, including wearable computers, smart dust, microelectromechanical systems and wireless integrated network sensors. Clearly, the user interface for nanoclients must explicitly or implicitly include geographical space, because the primitive of location, coordinates, will often be a critical part of the data stream. At the parallel and HPC end, the advantages and types of parallel computing are discussed, and examined in the light of what they can do for geographical systems. Some examples of the computing extremes are given in terms of application, including wearable computers, and geo-smart dust. Finally, the relationship between massive combined computing power at both the client and server side are considered in the light of tractability in geoscience. It seems quite likely that a new suite of methods, based in computing, will emerge to replace the standard set of mathematical tools, such as optimization methods, in use today.

Keywords: Human–computer interface; Nanoclient; High performance computing; Tractability; Geocomputation

E-mail address: kclarke@geog.ucsb.edu (K.C. Clarke).

0167-8191/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2003.03.001

1. Introduction

> "*In 2010 MEMS sensors will be everywhere, and sensing virtually everything.*" (Kris Pister, Professor of Electrical Engineering and Computer Science, UC Berkeley, 2000)

> "*Computing is not about computers any more, it is about living.*" (Nicholas Negroponte [16, p. 6])

Computation as a process has evolved significantly since its origins in generating rapid solutions to mathematical formulae. Both inspired and made deeper by decades of more direct use of computers to substitute for human reasoning, computation has slowly but surely moved from an analytical to an algorithmic paradigm. An algorithm can be expressed as an heuristic or as a program, and the fact that such a program works is equivalent to the mathematical proof that a solution exists. Decades of research on the methodology of problem solving have shown that if a problem can be reduced to a set of tasks, and that these tasks can be stated symbolically, then these tasks can be automated. Formalized as the Church–Turing theorem, that "there is no computational procedure that determines, given any schema of first order logic, whether that schema is valid", the implication is that "any computation that can be done at all can be done by a universal system such as a universal Turing machine" [20, p. 1125]. Thus, the conclusion is that programs can be considered forms of reasoning. Wolfram stated "Church's thesis should best be considered a statement about nature and about kinds of computations that can be done in our universe" [20, p. 1126]. At the heart of programs, of course, lie algorithms.

Algorithms are "methods for solving problems which are suited for computer implementation" [19]. Sedgewick divided algorithms into mathematical, sorting, searching, string processing, geometric, graph and advanced. It is the latter type, into which Sedgewick included special purpose hardware, dynamic programming, linear programming, exhaustive search, and NP-completeness, that is of interest in this paper. The thesis under examination is: is there a class of geographical problems that are not computable, that is, there are no algorithms for their solution? This is a problem of significant importance, because the use of computing for geographical and environmental description and problem-solving involves several orders of magnitude more complexity than that characterizing computing in the past. The fact that the thesis is even posable is due to advances in computing that will be described here. The implications of boundless geocomputation are both of great interest theoretically and practically. If new problems can now be solved, which problems and with what methods? In this paper, I suggest that the breakthroughs will come from computing's extremes, not its core. These are the extremes of power, size, and function. Ironically, as far as algorithms are concerned, there is only marginal difference between these extremes.

This paper begins with a review of the evolution of computing that gave us the extremes, illustrating a paradox of client–server architecture. Trends in computing, at the high and low end extremes are examined with some examples. The discussion

then moves back to computability and tractability of geocomputational algorithms, and discusses some of the methods that might be of value for further progress in geocomputation. I conclude with some speculations on future hardware and software, and with a suggestion that the first breakthroughs will be at the client end of the geocomputational extremes.

2. Evolution in geocomputation

The early days of computing were characterized by large, slow, bulky mainframe computers. They were typically expensive and inaccessible. Kept behind walls and glass, sustained by raised floors, white coated technicians with pocket protectors, and cooled by bulky air conditioners, they typically involved user interfaces that included accounting, command line interpretation, non-interactive feedback, dial-up telephone access through 300 baud modems, and limited capability. Computational problems attempted with these machines were reflected by the tools: FORTRAN was for formula translation. Nevertheless, the algorithms in use varied little from those of today. Early overambitious attempts to use computers for computing tasks that were more complex, even geographical, created negative reactions in cartography to the ugliness of their maps and in urban planning for their "seven sins" [15] (Fig. 1).

A second era in computing began with the introduction of the personal computer. While the PC led to a radical shift in computing, most importantly a massive increase in availability, at first its user interface was little better than the mainframe. This changed with the introduction of the WIMP (Windows, Icons, Menus and Pointers) interface and the desktop metaphor. New was the fact that each user now had a dedicated computing machine that could wait for each command from the user. However, using the computer still consumed the user's whole attention, and interaction consisted solely of input from the keyboard and mouse, and output to a screen. This was the computational environment that saw the spectacular growth of Geographic

Fig. 1. Computing eras. Left: mainframe (Console of BRLESC-II. US Army Photo http://ftp.arl.army.mil/~mike/comphist/); center: Classic IBM-PC; right: UCSB Wearable computer (Project Battuta: http://dg.statlab.iastate.edu/dg/).

Information Systems (GIS), the first data integration tools available for geographic data in digital form. As time progressed, the desktop machine shrank to laptops, palmtops and Personal Digital Assistants (PDA), yet without a substantial change in the human–computer interface.

A third era of computing began with the growth of the Internet that came from the development of what is now known as the World Wide Web. While the user interface remained essentially unchanged, the idea that data, resources, computing and storage could be separated in space and pulled together by a network on demand became widespread, so that today few, if any, classrooms or homes are without this basic functionality.

And so comes the most recent computing era, that of ubiquitous or pervasive computing. This era is characterized by the connection of things in the world through computation and computer devices. The era features hardware devices that are small and lightweight Internet access devices, with some computing capability. This era has put computing clients into mobile (telephones, cars and buses) and fixed (sensors, appliances and traffic lights) devices. In terms of the user interface, this era allows computing anywhere, at the users demand. It is an era of ubiquitous, highly mobile and fully embedded client computing devices, dispersed throughout our living environments. The average new luxury car, for example, has about a dozen computers, monitoring fuel use, playing music, or detecting the deployment of an airbag. This era also includes wearable computing. Wearable computers have the potential to monitor the life of the user in a first-person sense, an input capability known as context awareness. Context aware functionality completely alters the user interface model. In wearable computing, for example, the computer is working even when the user is not giving explicit commands, and so the user can be doing something besides interacting with the computer. This means that the computer does not demand the full attention of the user, and plays a passive and supportive role.

3. The paradox of computing at the extremes

The network and ubiquitous eras of computing are characterized by the client–server model. In the client/server model, a term applied both to components and programs, a server is a computer program that fulfills requests from client programs running on the same or a networked computer. A server is a computer program that provides services to other computer programs on one computer, or across a network. A client is the program or user that sends requests to the server. In recent times, especially in computing for GIS and mobile computing, functionality has become increasingly specialized at the client end. This is true especially when the client has specialized functions itself, or has capabilities that are limited by its function (e.g. A PDA). However, spatial analysis and the query tasks that create network data flows from increasingly large distributed data bases require ever more powerful servers. The result is a paradox in computing. Clients demand more services from their servers as they become more specialized, yet users want all of the compute power of the server on the client side. Logic would dictate that improving the network solves

the problem using the existing classic client/server model, leading to the classic thin client, fat server, and fat pipe solution. Taken to the extreme, this leads to extremely thin clients, obese servers, and overtaxed networks, in short what characterizes early 21st century computing.

If we simply assume that improvements to networks, in time, will sustain the paradox and give users the best of both worlds, then we can assume that both client and server will continue toward the extremes. At the server side, this implies a trend toward enhanced computational power, most likely to come from a combination of high performance computing (HPC), including supercomputers and massively parallel processors; and the methods that will make these tools more accessible. On the client side, the extremes will be of flexibility and miniaturization, marked by high mobility computing, access to the wireless internet, and wearable computing—all elements of what is now called ubiquitous computing. Combining these client size trends leads to what could be termed the nanoclient, nanoscience being a term much favored today.

The nanoclient is at the logical end of the road in diminution of size and specialization of function, because eventually the size of computing devices will approach the constraints of physical and materials science. Before true nanodevices, which may even be too small to see with the human eye, will come millimeter size devices, based on the centimeter sizes of today. As clients become extremely small, and lose the burdens of general purpose computing, then they will also become increasingly specialized, culminating in a single function. This is the case with dedicated computing devices in appliances, consumer products and cars. A nanoclients life could consist of monitoring its context or environment for one specific event or action, then transmitting the event's existence to a server. A model would be a smoke alarm, yet the range of functions and widespread use of the devices could be on a wholly different scale. Nanoclients bonded into sheetrock, for example, could feed a room's thermostat. Floating nanoclients dropped into floodwaters could feed models or mapped reports of flood danger. Nanoclient sensors poured with road tarmac could operate traffic lights.

The closest existing comprehensive example of a nanoclient is smart dust. Smart dust is groups of computing devices made up of cubic millimeter sized computer dust motes containing power supplies, processors, sensors, and laser and radio-based networking. If small and easy dispersable, smart dust motes could create vast interconnected parallel computers, each element of which would consume minscule amounts of energy (pJ per bit.) Kris Pister, working on a smart dust project at the University of California, Berkeley, envisions a complete sensor network node, including power supply, processor, sensor and communications mechanisms, in a single cubic millimeter [7]. The smart dust components or motes would be small and cheap enough to be "scattered from aircraft for battlefield monitoring, stirred into house paint to create the ultimate home sensor network or swallowed with your breakfast cereal for health monitoring." The goal of the project is to build a self-contained, millimeter-scale sensing and communication platform for a massively distributed sensor network. The dust motes will be the size of a grain of sand and will contain "sensors, computational ability, bi-directional wireless communications, and a power supply,

while being inexpensive enough to deploy by the hundreds'' [14]. Applications for the technology include environmental monitoring on Mars, internal spacecraft monitoring, earth and space-based communications, chemical/biological sensors, weapons stockpile monitoring, defense-related sensor networks, inventory control, product quality monitoring, smart office environments, and sports.

Smart dust depends on two technologies, microelectromechanical systems (MEMS) and wireless integrated network sensors (WINS). Both use semiconductor manufacturing techniques to make analog devices with electronics components. Such devices are often called localizers. WINS provide distributed network and Internet access to sensors, controls, and processors that can be deeply embedded in equipment, facilities, and the environment. At their extreme, MEMS and WINS take computing to the nanoscale, and allow smart dust.

Two extraordinary potentials for smart dust come to mind. First, if computational problems can be adequately parallelized, they can be distributed to innumerable smart dust motes, each acting as a processor. This mimics human thought, both in numbers of neurons in the brain and in the ability to repair, add, and replace neurons. Secondly, because the smart dust motes can be dispersed into the environment, they are the ultimate tools for using the environment itself as a computer, and so computing environmental problems.

4. Context-awareness and geo-smart dust

Research on wearable computing has introduced the concept of ''context awareness'' into computation [1]. Context-awareness is ''the use of context to provide task-relevant information and/or services to a user, wherever they may be'' [10]. From a geocomputational perspective, it is the ''wherever'' that is of most interest. Perhaps the most fundamental primitive of event or existence-based information are time and place. The timestamp is well known, and fortunately is a component of the most effective positioning technology, the Global Positioning System (GPS). So context awareness implies a constant input stream containing at the least a latitude/longitude pair or their projected equivalent and an elevation [6]. Required of the context aware computing environment at a minimum is the ability to differentially act based on geographical location. Such functionality is now increasingly common in web sites, in-vehicle navigation systems, consumer GPS equipment, and even cellular telephones. These location technologies include GPS, cellular, broadcast, microwave, Infra Red readers, etc., and along with their software now constitute a new field of technology often termed ''location-based services.''

Consider such technologies at the nanoscale, what might be called geo-smart dust. Given the existing technologies, the problems to be overcome for such a smart-dust variant are evident, and include continuity (GPS systems today do not work indoors, and WINS rely on burst transmissions), network bandwidth, miniaturization, and systems integration. Software will also be no minor problem. Nevertheless, geo-smart dust applications are even more vast in scope than smart dust. They include personal location tracking, geolocation of objects in space (at last, find those car keys

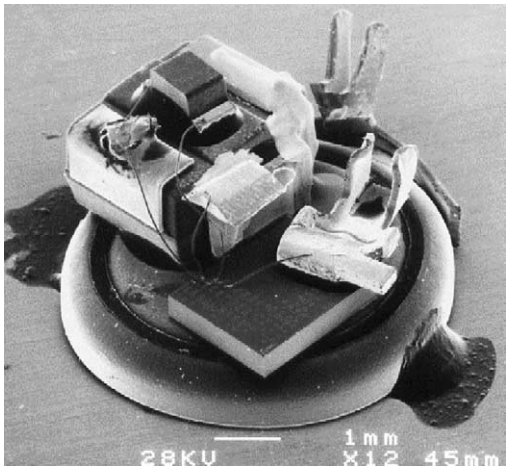

Fig. 2. Smart dust WINS mote. Source: http://robotics.eecs.berkeley.edu/~pister/SmartDust (used with permission).

or glasses), autonomous mobile robots (what might be called geoprobes), geo-tags built into consumer goods (locate and map every pair of jeans ever sold from a particular store), the addition of highly localized intelligence for personal and vehicle navigation (highways could announce closed exits, or the road conditions ahead), machine control, self-reporting infrastructure inspection, smart homes and a whole range of geo-biosensors that could radically change medical imaging (Fig. 2).

Geocomputation is the "eclectic application of computational methods and techniques to portray spatial properties, to explain geographical phenomena, and to solve geographical problems" [8, p. 17]. Extending this concept to geo-smart dust as a computational vehicle creates an extraordinary array of new applications, and gives rise to some intellectual and theoretical problems that will require solutions if the technology is to flourish. With geo-smart dust embedded in our natural landscape, space itself can be a metaphor for space in a computational-geographic operating environment. A landscape, map or image, then, can be a model of itself. Geographical objects can have knowledge of their own proximity, distance, path, connectivity, etc. Yet just as the client extreme shrinks to literally a speck of dust, quite obviously the demands on WINS, demands on the network and the server would become massive. Already we have Internet II, research into high band width communications, fiber optic cables, and the rise of the server farm. As the client gets thinner, it will also become more specialized. Server end tasks will then take on general purpose computing, searching and browsing, partitioning, sorting, analysis and modeling. This will require major improvements in power. Yet if we also solve these problems, the extremes of geocomputation will run into the limits of tractability. I consider the tractability problem after an equivalent examination of trends in high performance computing.

5. Trends in high performance computing (HPC)

The "dot-com" era saw the emergence on a massive scale of dedicated and specialized high end information servers, the so-called "server farms." Simultaneously, the personal computer and broad acceptance of Internet service providers have also greatly expanded the pool of highly distributed low end servers, that collectively represent astonishing computing power if considered a single parallel machine. Scientists working on compute-intensive tasks have moved their work to a third type of high end machine, the "compute server," normally a network of workstations, a parallel computer or a cluster. A fourth type of high performance is the more traditional supercomputer, which itself is highly parallel. With the end of the Cold War, supercomputers have become increasingly devoted to geocomputation and environmentally related tasks, such as climate simulation and sub-surface geophysical modeling.

Of these, it is parallel computing that seems to be moving ahead fastest. After a first generation of only partially successful application, parallel computing has matured. With accessibility has come a need to expand software capability for parallelism. "It could be argued that the field of parallel processing, in both hardware and software components, is simply relearning at an accelerated pace the earlier lessons of computing as a whole: namely, that viability is a function of generality" [12]. This is probably exemplified by two examples. First is the "Stone Soupercomputer", a Beowulf-style parallel machine made from assembled Intel processors running Linux [11]. With a theoretical 1.2 Gflops performance, and a monetary cost close to zero, the price/performance ratio is almost infinite. Second is the distributed solution that more resembles the smart-dust trend, the SETI@home approach, where users contribute CPU cycles to a screen saver that processes chunks of data from a network of radio telescopes seeking contact from extraterrestrial intelligence (www.setiathome.ssl.berkeley.edu) (Fig. 3). Checked at 3:15 pm Pacific on June 6, 2002, the site showed that 3,762,920 users had collectively and cumulatively contributed 1,005,685 CPU years for about a 37.64 TeraFlops/second level of performance. This would be a small fraction of what a distributed geo-smart dust system might be capable of.

There are practical consequences of moving to parallel approaches to computing. The first is that specialist software with the same user friendliness of PC systems is far from present. Many software solutions exist, some in the public domain: Parallel virtual machine (PVM), the message passing interface (MPI), Linux and Unix tools, High Performance Fortran and some home brew systems (e.g. SETI@home). In practice, this means that not only must code be modified (for example, definitions included and libraries linked), but a parallelization controller must be used to execute code. At a minimum, identifiers for each available microprocessor must be added and attached to tasks. The result, however, can be remarkable increases in actual code performance.

Combining geocomputation with the capabilities of parallel and highly distributed computing implies using the raw power of HPC to solve problems unsolvable using current methods. A goal is to seek pattern and form unavailable by traditional

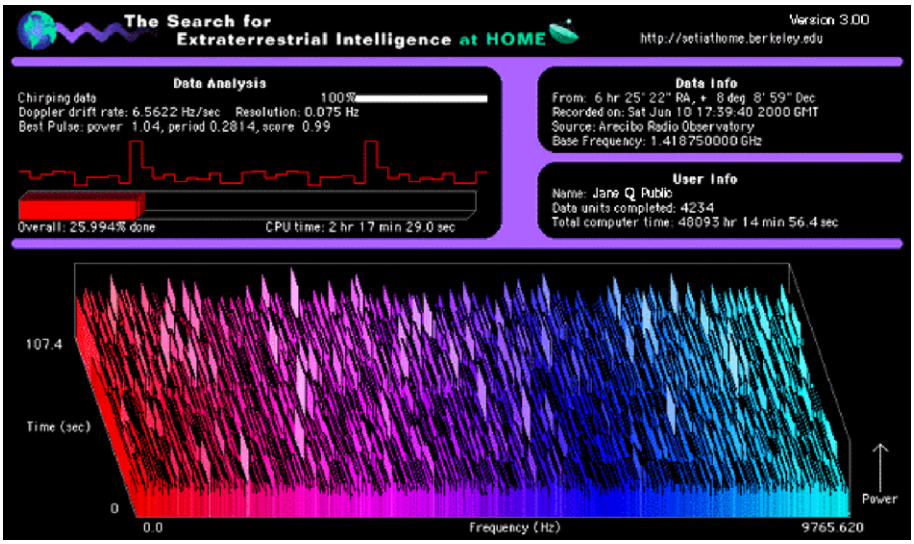

Fig. 3. SETI@home. A screen-saver application capable of massively distributed massively parallel computing. See: http://www.setiathome.ssl.berkeley.edu.

means, such as stable state modeling, or statistical analysis. With so many problems at stake, some solutions may need HPC server-side solutions, and some the type of client parallelism possible with geo-smart dust or SETI@home type approaches. The approaches have already tackled problems of massive scope: encryption, for example. One wonders whether or not the extraordinarily complex problems of the natural environment can yield to the methods? Might it be possible to effectively model global climate change in such a way that alternative policies could be tested prior to implementation? Given the earth's environmental problems, such power could have critical implications. Floods, snowstorms, and tornados could be anticipated, and crop and fishing yields managed sustainably.

6. Parallel geocomputation

Parallelization is the partitioning of serial computation task into tasks which can be performed on different processors independently. Ideally, of course, the independence is complete, that is there is no necessary communication between processes. Often tasks are quasi-independent, and so require communication or synchronization. The goals of parallel computing are to process larger data sets, at higher resolution, and to process more scenarios in less time. In functional parallelism, each processor performs different operations on its own data. Alternatively, in domain decomposition, data are partitioned between processors. Both methods suit geographical data, yet domain decomposition is known to have problems working with random numbers and at map and image edges [17].

A process is perfectly (or embarrassingly) parallel if the processors work completely independently. While this is often not possible for algorithms [2], it is sometimes entirely feasible for methods like model calibration, Monte Carlo simulation, and cellular automaton modeling. In work on modeling urban growth under project Gigalopolis (www.ncgia.ucsb.edu/projects/gig), a project goal has been to use historical data for urban areas to simulate present day urbanization. The simulation uses a cellular automaton model (called SLEUTH), which is calibrated for a given city using hindcasting, and then run into the future to forecast possible futures or scenarios [5]. SLEUTH uses environmental data layers showing the topographic slope of an urban area, land use at different times, zones excluded from development such as parks, the extent of the urban area at different time periods, and historical maps of the transportation network as inputs. These are then used to train a cellular automaton to simulate the urban growth process over time. More details about SLEUTH are on the project web site at http://www.ncgia.ucsb.edu/projects/gig, including the C-language source code. Essential to the model's realism, accountability and repeatability is the process of calibration [4]. Not only is the model completely sensitive to its initial conditions (since the data are fixed, this resolves to a parameter set), the desired solution should minimize variance across multiple Monte Carlo simulations. Model calibration for a medium sized data set and minimal data layers requires about 1200 CPU hours on a typical workstation. Such a solution raises issues of tractability. Obviously, an important consideration is whether a simulation can be created in a time suitable for its effective use as a planning tool, otherwise the whole approach is suspect.

In the most recent version of the model, the SLEUTH model has been recoded into modular flow ANSI C, the dynamic memory allocation returned to a flat memory structure optimized for the Cray memory model, and the code modified to use the MPI. An important check was to create bit-wise identical results to an earlier model version. SLEUTH uses a brute force calibration methodology, in which it partitions and explores the model's parameter space in increasing parametric and spatial detail. While early work also used coarsened data resolution to reduce CPU time, this has since been proven non-scalable and the whole brute force calibration process is now used on the entire dataset [13].

SLEUTH calibration has been shown to be embarrassingly parallel, giving a linear speed up in performance with processors. Implementations to date have included on a cluster of 15 SGI O2s, on a Beowulf Linux Cluster and on the Environmental Protection Agency's Cray C-90 and T3D and the Cray T3E-1200 at the National Environmental Supercomputing Center in Research Triangle Park, North Carolina. The T3E at NESC (Hickory) is a massively parallel processor, with 64,600 MHz processor elements giving a peak 1.2 Gflops/second/PE, with 16 GB total memory on a 3-D Torus. For an application of SLEUTH at 1 km resolution to the entire Mid Atlantic Integrated Assessment region (all or parts of eight eastern states), full calibration was achieved in one CPU hour.

Similarly, SLEUTH calibrations were run at the United States Geological Survey's Rocky Mountain Mapping Center (RMMC) on an 8-node AMD Athlon Beowulf PC Cluster. Runs were executed for the Seattle Area Natural Hazards

Project using image arrays up to 2000×2400. The Seattle 1/4th-size calibration run took three days to complete on the 8-node Cluster, whereas, the Seattle full-size calibration run required 10 days of processing. As a comparative test, RMMC initiated a Seattle full-size calibration run on the SGI Origin 2000 server using only one CPU and benchmarked that the 8-node Beowulf PC Cluster required eight times less CPU hours than the SGI to execute 100 iterations.

Clearly, a variety of solutions can work for brute force methods, and give results that are highly amenable to parallel computation. Having one set of code run on a large variety of platforms is highly desirable, because cross-calibration and sensitivity tests can be conducted, and the value of the modeling enhanced [3].

7. Tractability in geocomputation

Given that geocomputational methods have great potential, what "unsolvable" problems—that is unsolvable using deterministic methods—do we solve? Computational and scientific problems can be divided into five types: trivial, analytical, intrinsically hard, those solvable in polynomial time (P), and those solvable in non-deterministic polynomial time (NP). NP problems are definitely hardest to solve, because their solution requires more time than is available. If they can be proven to require non-deterministic polynomial time, they are termed NP-complete. Nevertheless, major jumps in geocomputation should crack whole sets of NP hard problems with methods that vary markedly from the simpler methods for the first types of problems.

In an incisive examination of analytical problem solving in cartography, a set of possible methods were suggested [18]. These include the solution of a simplified problem (such as a special case), determination of an approximate solution, finding a "good enough" solution that may not be (or even be provably) optimal, or to relax the problem's constraints. Bodies of theory that may be of use in tackling such intractable problems include rough sets, fuzzy logic, computational complexity theory (with its Big-Oh notation), approximation theory, complex systems theory, brute force methods, and divide and conquer (problem partitioning). Is it reasonable to assume that new solutions will be yielded from these methods and superior computing power? After all, a problem is intractable if no polynomial-time solution is known. Yet this does not preclude an unknown solution, as the Church–Tukey thesis implies. Similar logic applies to NP-complete and NP-hard problems. Indeed, many simple problems, such as those in computational geometry are intractable, and often use some of the above methods. For example, the most fundamental spatial operation (overlay of two vectors to find intersection) has an intractable special case (parallel lines) that requires special case solutions, as also does the construction of Voronoi polygons (circumscribed circle) [9].

Seeking special cases is an excellent way to begin to work on intractable problems, because the parameter space is sufficiently narrowed that other methods can be employed, perhaps even analytical methods. Other techniques are to change the problem statement, to change the problem domain, to seek sub-optimal (good enough)

solutions or approximation. These methods are highly likely to solve whole families of problems, simply because Moore's law, the power of parallelism, and a new generation of supercomputers will push us beyond the geographical tractability threshold. For example, the Pacific Northwest Laboratory of the Department of Energy recently signed a contract with Hewlett-Packard to produce an 8.3 TeraFlops Linux supercomputer, with 1.8 TB of RAM, 170 TB of disk, and 1400 Intel McKinley and Madison processors. Lawrence Berkeley Laboratories recently took installation of IBM's ANSI White, capable of 12 Tflops. With such power available, and with the geographical and environmental systems problem threshold likely to be transcended, perhaps by an order of magnitude, it is clearly the time to think big. Even using existing tools and methods, such as GIS software, much could be done that was before impossible. Leading the way could be models such as TRANSIMS [22], an individual-based model that simulates the position on a street map of every car in a city every 10 min. Just extending existing agent-based models could create a hillslope erosion model with soil particles or molecules as independent agents. Social science applications could simulate urban land use change, real estate transactions, city-wide employment, or intercity migration using each person, or each land ownership parcel as an independent agent with a different behavior profile. GIS applications could be repeated for all possible combinations and permutations of error, for an exhaustive sensitivity analysis of the final results, especially for critical or sensitive siting decisions, such as locating toxic waste facilities.

8. Nanogeocomputation

Smart dust and its successor, geo-smart dust, provide many exciting opportunities at the other computing extreme. While massive computing power can provide models capable of simulating entire earth systems, geo-smart dust will provide a way to instrument the earth to calibrate these models and link them to reality. If geo-smart dust motes are sufficiently small, at the nanoscale, then the measurement instruments can either join the environment itself as a sample of reality or become a scaled model of the same reality. For example, a geomorphologist studying the erosion of hillslopes can calibrate a soil particle scale independent agent based model in two ways. First, a real hillslope could be seeded with geo-smart dust, each mote reporting its position and movement during erosion events such as soil creep and landslides. Alternatively, a laboratory based pile of geo-smart dust particles could be assembled in a standard laboratory flume, and erosion simulated at a smaller scale, with the purpose of calibrating a supercomputer model that can duplicate the real world.

Individual humans, animals or vehicles could be carriers of geo-smart dust, leaving "strands" or paths through space and time showing their daily activities and behavior, a gold mine for social scientists. Early version of this are already under way using personal GPS by saving individual's movements over extended periods, often users of wearable computers. Probably the best aspects of the geographical environment to model and measure in this way are those areas of dynamics where change in particularly difficult, costly, or dangerous to observe directly. Hydrologic systems for

surface water and contamination plumes in the groundwater could be monitored with nanofloats or geo-smart molecules. MEMS technology would be useful to manufacture special purpose agents that could not only measure but also intervene with the environment. A zebra-mussel detector and killer, for example, or a nanoagent that attaches itself to heavy metals in river sediments, reporting its location as it does so, and then seeking out a safe disposal location.

With so much potential, it will be hard to measure success. One way to do so would be to use the example of the human genome project, and attempt to model at the atomic level one type of natural environmental system completely, explaining all observed spatial and temporal variation. While problems like the weather may not yield immediately, other less complicated systems (say the life cycle of the mosquito and its spreading of malaria) may prove more susceptible to the methods described. Geography and geocomputation have so far not been ''big science'' disciplines, like Physics. However, once the ability to demonstrate the power of geospatial representation becomes evident, large scale science on the order of the human genome project may be appropriate, and the benefits to everyday life could be immense, not the least of which could be homeland security. While the first breakthroughs will come on the server (compute engine) side, the true revolution lies on the client side.

The broader point here is that environmental problems can be better understood through next-generation computing. Where systems are non-linear, the majority of complex earth-related systems, likely future scenarios can be better modeled, along with estimates of system uncertainties [21]. Thus imperfect planning solutions to complex environmental problems can indeed still be forthcoming.

9. Conclusion: geocomputation and relevance

This paper was first formulated as a conference keynote lecture, given in Brisbane Australia at Geocomputation 2001 less than two weeks after the events of September 11th. At that meeting, Brian Lees opened the conference with a set of comments encouraging the attendees to seek relevance in their work. A similar but less sinister theme was evident at the author's keynote at the Annual Meeting of the Society of South African Geographers at Goudini Spa, Western Cape the previous July. With so many hard problems, we should work on the most relevant first. Geocomputation has been defined as the art and science of solving complex spatial problems with computers (www.geocomputation.org). As GIS has moved beyond software and applications to become an approach to science [6], and as increasingly the geo-spatial world has gone from data-poor to data-rich, the potential of geocomputational methods, and coupled with the awesome power of supercomputing and parallelization, it is not ambitious to suggest that whole sets of geographical problems will yield to geocomputation in the near future. Which problems will we choose to solve? At first, obviously, application will depend on our ability to select intractable problems that are susceptible to our methods just beyond the geocomputation threshold. Nevertheless, with the methods ready to go, scholars and scientists should

not limit themselves to geographical triviality. World poverty and hunger may not yield at all, but employment security and food distribution in one community, in one Cape Flats or Soweto, may follow from explicit computer models capable of exploring extraordinary solutions to these problems. If so, then we will have created something of both immediate relevance and lasting value to humankind.

References

[1] G.D. Abowd, A.K. Dey, A.K. Abowd, G.R. Orr, J. Brotherton, Context-awareness in wearable and ubiquitous computing. GVU Technical Report GIT-GVU-97-11, 1997. Available from <http://www.cc.gatech.edu/fce/pubs/iswc97/wear.html>.

[2] M.P. Armstrong, R.J. Marciano, International Journal of Geographical Information Systems 10 (6) (1996) 713–730.

[3] J. Candau, Calibrating a cellular automaton model of urban growth in a timely manner, in: 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4), Banff, Alberta, Canada, September 2000.

[4] K.C. Clarke, S. Hoppen, L. Gaydos, Methods and techniques for rigorous calibration of a cellular automaton model of urban growth, in: Proceedings, Third International Conference/Workshop on Integrating Geographic Information Systems and Environmental Modeling, Santa Fe, NM, January 21–25, 1996.

[5] K.C. Clarke, L. Gaydos, Loose coupling a cellular automaton model and GIS: long-term growth prediction for San Francisco and Washington/Baltimore, International Journal of Geographical Information Science 12 (7) (1998) 699–714.

[6] K.C. Clarke, Getting Started With Geographic Information Systems, Prentice-Hall, Upper Saddle River, NJ, 2001.

[7] L. Collins, Network in a dust storm, EE Times. Available from <http://www.eetuk.com>, April 5, 2002.

[8] H. Couclelis, Geocomputation in context, in: P.A. Longley, S.M. Brooks, R. McDonnell, B. Macmillan (Eds.), Geocomputation: A Primer, John Wiley, Chichester, 1998, pp. 17–29.

[9] M. Deberg, M. van Kreveld, M. Overmars, O. Schwartzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag, Berlin, 1997.

[10] A.K. Dey, G.D. Abowd, D. Salber, A context-based infrastructure for smart environments, in: Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), 1999, pp. 114–128. Available from <http://citeseer.nj.nec.com/dey99context-based.html>.

[11] W.W. Hargrove, F.M. Hoffman, T. Sterling, The Stone Soupercomputer: ORNL's first Beowulf-style parallel computer: The Do-It-Yourself Supercomputer, Scientific American 265 (2) (2001) 72–79.

[12] R. Healey, S. Dowers, B. Gittings, M. Mineter (Eds.), Parallel Processing Algorithms for GIS, Taylor and Francis, London, UK, 1998.

[13] C. Jantz, M. Shelley, S. Goetz, A. Smith, Modeling future growth in the Washington, DC, Area, Report to the Chesapeake Bay Foundation, Univesrity of Maryland Mid-Atlantic Regional Earth Science Applications Center, 2002. Available from <www.resac.umd.edu>.

[14] J.M. Kahn, R.H. Katz, K.S.J. Pister, Emerging challenges: mobile networking for smart dust, Journal of Communications And Networks 2 (3) (2000).

[15] D. Lee, A Requiem for large scale modeling, Journal of the American Institute of Planners 39 (3) (1973) 163–178.

[16] N. Negroponte, Being Digital, Knopf, New York, 1995.

[17] J.E. Mower, Developing parallel procedures for line simplification, International Journal of Geographical Information Systems 10 (6) (1996) 699–712.

[18] A. Saalfeld, Complexity and intractibility: limitations to implementation in analytical cartography, Cartography and Geographic Information Science 27 (3) (2000) 239–250.

[19] R. Sedgewick, Algorithms, Addison-Wesley, Reading, MA, 1983.
[20] S. Wolfram, A New Kind of Science, Wolfram Media, Champaign, IL, 2002.
[21] W.-N. Xiang, K.C. Clarke, The use of scenarios in land use planning, Environment and Planning B, in press.
[22] J.L. Casti, Would-be Worlds: How Simulation is Changing the Frontiers of Science, Wiley, New York, 1997.