How Smart Does an Agent Need to Be?

Scott Kirkpatrick[1] and Johannes J. Schneider[1,2]

[1]The Hebrew University of Jerusalem, Israel

[2]Johannes Gutenberg University of Mainz, Germany

January 8, 2004

Abstract

The classic distributed computation is done by atoms, molecules, or spins in vast numbers, each equipped with nothing more than knowledge of their immediate neighborhood and the rules of statistical mechanics. Such agents, 10^{23} or more of them, are able to form liquids and solids from gases, realize extremely complex ordered states, such as liquid crystals, and even decode encrypted messages. We'll describe a study done for a sensor-array "challenge problem" in which we have based our approach on old-fashioned simulated annealing to accomplish target acquisition and tracking under the rules of statistical mechanics. We believe the many additional constraints that occur in the real problem can be folded, step by step, into this stochastic approach. The results have applicability to other network management problems on scales where a distributed solution will be mandatory

1 Introduction

A very old idea in distributed computing and communication is the network that self-organizes to perform some local function of moderate complexity and then communicates its discoveries to some wider world that is interested. In military communications, this might take the form of sensor arrays of simple, inexpensive units "sprinkled" behind a rapidly advancing force in battle to provide communications, or around a sensitive installation to create a defensive perimeter. We recently participated in a study of an application of this sort. From our experience, we concluded that understanding the complexity of these self-organizing systems is critical to achieving their objectives, and is sorely neglected in the most popular approaches to programming them. The

study which we joined (in midstream) had been envisioned as an exercise in multi-agent distributed programming with negotiation protocols. These were to be employed to resolve a difficult optimization problem, which would frequently require settling for locally sub-optimal results in order to meet challenging real time constraints with acceptable solution quality.

The sensors in this array are Doppler radars, rather useless individually but capable of tracking targets if three or more of the radars can lock onto a single foreign object for a sufficient time. In possible applications, such as protecting a aircraft carrier during refueling, one could imagine having a few thousand such antennas floating in the sea. But for research and demo purposes, controlling four to twenty such antennas was considered quite challenging. Thus the temptation was strong to write very special-case agents, and evolve very specific negotiation protocols as overloads and other problems were discovered. One of the other groups in this study was planning to develop a target selection protocol, with negotiation, that would support 64 antennas observing a dozen or so targets, and simulate it on a 32-processor server cluster, but not in real time.

Working with Bart Selman and the Intelligent Information Systems Institute of Cornell University, we were asked to determine if there might be phase transitions, or other threshold phenomena blocking the way to success in this endeavor, and if so, what to do about it. We wrote a simple simulation of the target assignment and tracking problem. Because of our concern with complexity and its scaling, we treated the antennas as spins, not agents, and looked first for a Hamiltonian that would give them the right behavior, counting on statistical mechanics and temperature to handle the negotiation part. It worked rather well, although there were a few inventions required to find a way to make everything work. What follows is lightly edited from our progress reports. After describing our effort, we conclude with comments on the role of phase transitions in this sort of large scale engineering problems as they continue to grow in scale over future decades.

2 Physics-inspired negotiation for Sensor Array self-organization

We implemented a family of models which borrow insights and techniques from statistical physics and use them to solve the sensor challenge problem under a series of increasing restrictive real-time constraints. We describe results with fairly difficult communications constraints and moving targets, covering the phases from target acquisition to tracking. The method can

deal with momentary overloads of targets by gracefully degrading coverage in the most congested areas, but picking up adequate coverage as soon as the target density decreases in those areas, and holding it long enough for the target's characteristics to be determined and a response initiated. Our utility function-based approach could be extended to include more system-specific constraints yet remained efficient enough to scale to larger numbers of sensors.

Our model consists of 100s of sensors attempting to track a number of targets. The sensors are limited by their ability to communicate quickly and directly with neighboring sensors, and can see targets only over a finite range (which probably exceeds the range over which they can communicate with one another). This is modeled probabilistically. Initially we assumed that there is 90% probability of a communication link working between two neighboring sensors, 50% probability at twice that distance, and 10% probability at three times that distance, using a Fermi function, or logistic function for the calculation at an arbitrary distance. Subsequently, we explored other characteristic cutoff ranges for communications, using the same functional form for the probability of communication. At the start of each simulation we use this function to decide which communications links are working. The range at which a target can be tracked is a parameter which we vary from a minimum of 1.5 times a typical neighbor distance to 4 times this distance. Sensors are placed on a regular lattice, with positions varied from the lattice sites by a random displacement of up to 0.1 times the lattice spacing. We considered three lattice arrangements to give different spatial densities – honeycomb was the least dense, then square lattice, then triangular lattice. But the precise lattice geometry made little difference beyond the changes in sensor density that resulted, so most work was carried out with a square array of sensors, randomly displaced as described.

Since three sensors must track a given target before an action decision can be taken, this means that at most 33 targets/100 sensors can be covered when no other constraints appear. We treat each sensor as capable of making independent targeting decisions. Moreover, each sensor is aware of what targets each of its immediate neighbors is tracking. We define a utility function which is a sum over the quality of coverage $F(n)$ achieved on each target, and depends on n, the number of sensors that are tracking that particular target. If exactly three sensors are tracking the target, the function is minimized. If more than three are tracking the target, the function increases (becomes less good) slowly, to discourage wasting sensors that could be helping to track other targets. If less than three are tracking the target, the function increases sharply, since this is the situation that we want to avoid. We have developed a version of this utility function which works well on the easy problem with

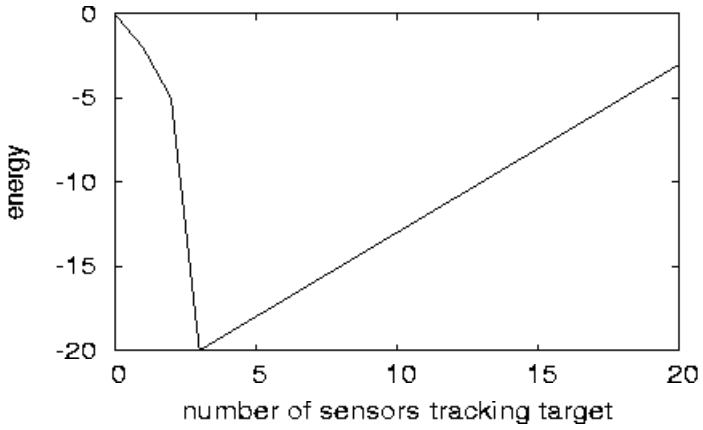

Figure 1: Utility function for a single target tracking cluster

no communications or target range constraints. (Think of this as having all the sensors on one mountaintop, all connected to each other, and all targets at a distance.) We have then applied the same utility function to the harder problem of an extended array of sensors, with targets moving between them. The function is plotted as Fig. 1.

We look for Target tracking Connected clusters, called TC clusters of 3 or more sensors (all connected and all tracking the same target). Since any connected cluster of 3 or more sensors has at least one sensor which has two neighbors, this assures that an action decision can be reached. To calculate the utility function, we find, for each target, the TC clusters tracking it. Each TC cluster, with n sensors, contributes $F(n)$ to the utility of the whole solution. Note that there may be more than one cluster of sensors tracking a given target, with no direct communications links between members of the different groups, if the distance over which a target can be seen exceeds the sensor spacing. There may be benefits of having this occur. This will make the calculation more readily distributable, and the extra TCs may help to ensure successful handoff of fast moving targets.

Initially we assign each sensor to a randomly selected target within range. We then use simulated annealing [1] to improve the solution. The annealing process consists of each sensor independently and asynchronously considering switching to tracking some other target, and accepting the possible change if the utility function's value will be decreased by the change. If the utility function increases at simulated temperature, T, the move may still be accepted, but with probability $\exp(-(E_{\text{final}} - E_{\text{initial}})/T)$. This is a standard, robust method from optimization theory [2] and it turns out to be a very

good starting point for deriving fully distributed search algorithms.

The simulated annealing process can work extremely fast. For the problem without communications constraints very high quality solutions were obtained when each sensor tried 1 to 3 target choices at each of only 3 temperatures. We have tuned our approach to the constrained problems to determine how robust this method will be at different points in the "phase diagram" of sensor to target range, sensor communications range and delays. Our phase diagram parameters are:

- Average number of sensors communicating with a given sensor
- Average number of sensors in range of an arbitrary target
- Ratio of total number of targets to total number of sensors

We can use these normalized parameters to map out a phase diagram of feasible solutions (when the targets are uniformly distributed), and we use exact methods, limited to solving for very small numbers of sensors and targets, to identify the point at which a solution is 50% likely to be possible for a random, roughly uniform arrangement of targets. We then use our dynamic solution to treat a more difficult case – a swarm of targets arrive from outside the target array, then spread out, turning randomly as they proceed. In this case we must deal at first with a load which is higher than average, and targets which keep appearing from outside the boundaries of the array (to keep the total number constant as other targets exit). These results can be viewed in two ways – impressionistically, as movies at
`http://www.cs.huji.ac.il/~jsch/beautifulmovies/movies.html`
`http://www.cs.huji.ac.il/~jsch/beautifulmovies/movies.html`,
or more quantitatively by measuring the actual length of time that each target is adequately covered by one or more target connected clusters (TC clusters) in the sensor array. Each movie is actually an animated GIF file and rather large. When we first constructed these, each required about 20 MB. Subsequently we found better encodings that reduced them to about 2 MB each. They provide the raw material of our study. Finding good ways of analyzing the performance of the whole system from the scenarios in the movies was one focus of our research in this area. We developed arguments for using these overall performance metrics as an assay for the rough location of the phase boundary to the region where good solutions could be found.

The conventions for the movies are as shown in the sample frame (Fig. 2) below. Each sensor is identified by a cross. The sensors that are tracking a target and part of a TC show the antenna sector as a wedge. The targets are dots, and each target is surrounded with a circle showing the range within

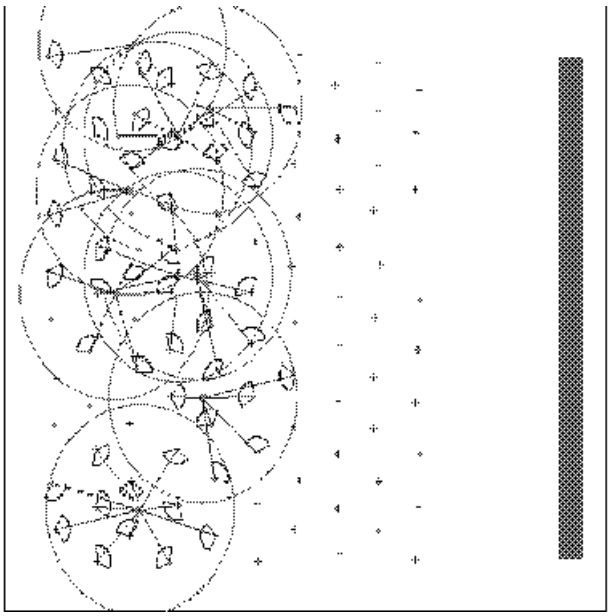

Figure 2: Sample frame from our movies of target tracking

which it can be sensed – every sensor within that circle can see the particular target if it chooses. Lines connect sensors and targets tracked in a TC. The thermometer bar on the right represents the fraction of targets covered by one or more TC's. The wedges representing antenna sectors reflect a "real-world" issue that we inserted at this point without actually modeling it as a constraint, and were later able to include within our optimization approach. Each of the prototype Doppler radars in the study had three antenna cones, each cone capable of observing 120 degrees of azimuth. Only one cone could be active at a given time, and there were power and delay penalties for switching the sensor from one sector to another. We assumed that the sectors were oriented at random, with their angles fixed during the time that was simulated. This sort of complication is extremely hard to add at late stages of an explicit programming approach, as would be typical of multiagent negotiation methods, but we were easily able to explore it in our simulations.

Our test scenario is that a fixed number of targets impinge on the array from one side and cross the array while turning at random. Each time a target exits the array, a new target is introduced at a random position on the entry side to keep the total number constant. We run the "movie" simulation for 100 time steps, roughly three times the time it takes for each target to cross

the array. We have developed several ways of evaluating the overall results of the simulations, but it may help to watch the actual movies. They may be downloaded from the group's website at the Hebrew University, `http://www.cs.huji.ac.il/~kirk`.

Before each movie starts, we show the communication links that were in effect for the duration of the scenario. To summarize all the frames in a movie, we show a coverage summary at the end of 100 iterations. The numbers summarize the parameters of the movie, while the horizontal lines (one for each target) code for coverage during the 100 iterations. The color code is: no color – uncovered; red – one TC covers the target; green – two TCs; blue – three TCs... Purple and colors indicating still more TCs are occasionally seen in the very easy cases as in Fig. 3a. As target density increases, the gaps with no TCs forming for a given target get longer, and the periods of multiple coverage shorter, as seen in Fig. 3b.

To analyze the effectiveness of coverage, we also plot at the end of each movie the amount of time that each target was either tracked, or tracked continuously, against the time that it was in view of the sensor array. An example of this is shown in Fig. 4 and is discussed in more detail below.

At first, in the movies, we took many more simulated annealing iterations than were necessary for optimization alone, in order to measure physical quantities like susceptibilities that characterize the process. This is useful in determining the temperatures to be used for annealing [4]. We then reran some of the same tests with factors of 1000 to 100,000 fewer sensor target assignment changes attempted during each iteration. The payoff for this is in reducing the message traffic between neighboring sensors, since they need to communicate with each other each time they choose to focus on a different target. The quality of the results, measured in terms of fraction of targets covered, was almost unaffected by our 1000-fold speedup, and slightly decreased by our 10,000-fold speedup, but the message traffic becomes quite reasonable, as the Figures 5 and 6 show:

For 1000-fold speedup, (shown in Fig. 5 on a case with 20 targets, square lattice, target range = 3 nearest neighbor distances), we need to exchange 10-12 messages per sensor during each iteration step. The actual fraction of sensors which retain their target assignment from one iteration to the next is only 20-30%. In our 10,000-fold speedup (Fig. 6, for the same overall parameters), the bandwidth required has been reduced to one or two messages per sensor per iteration, and typically half of the sensors change their target assignment every iteration, but the quality of coverage is reduced.

To solve the problem time step after time step when the targets have moved for some short interval, we start with the previous solution. As the targets move, we simply "bounce" the temperature back up to a high value,

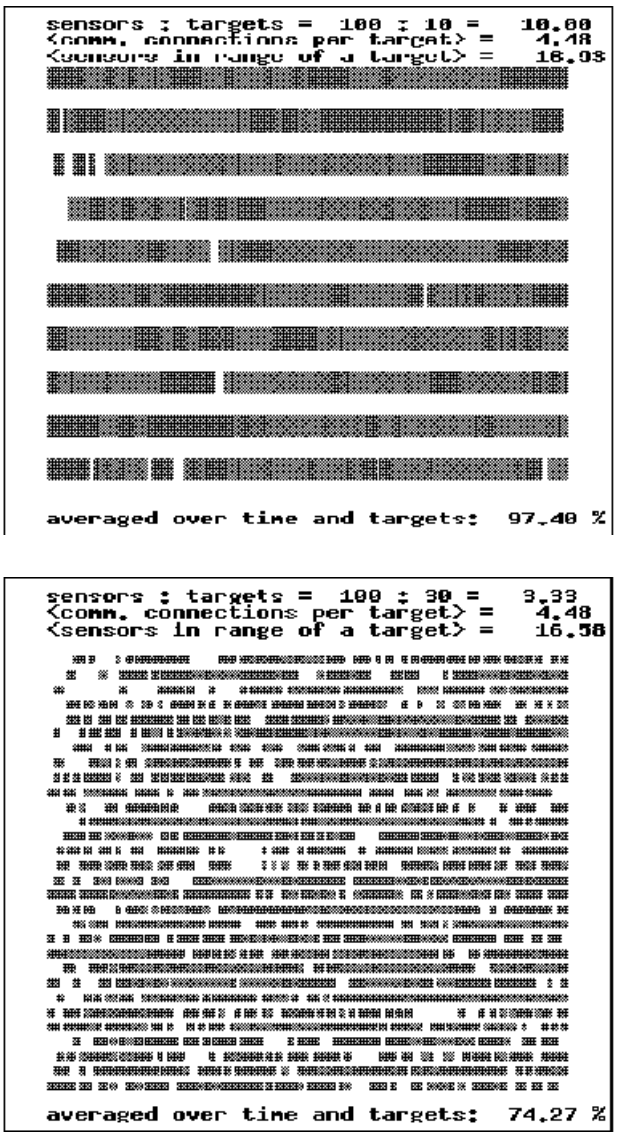

Figure 3: Sample movie frame summarizing the number of TCs covering each of 10 targets (a) on top, or 30 targets (b) on bottom.

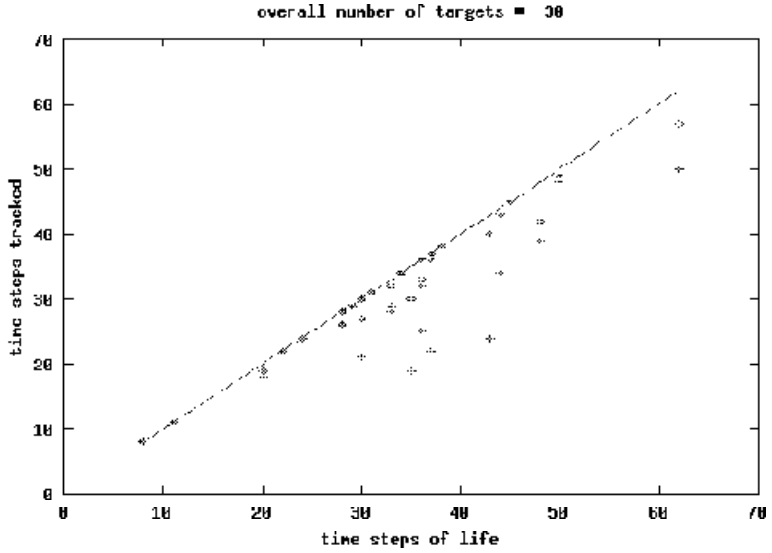

Figure 4: Total (circles) and maximum continuous (plus symbols) target coverage achieved with 10 targets present at any one time. A total of 30 targets were tracked during the 100 timesteps. 100 sensors, CR10.

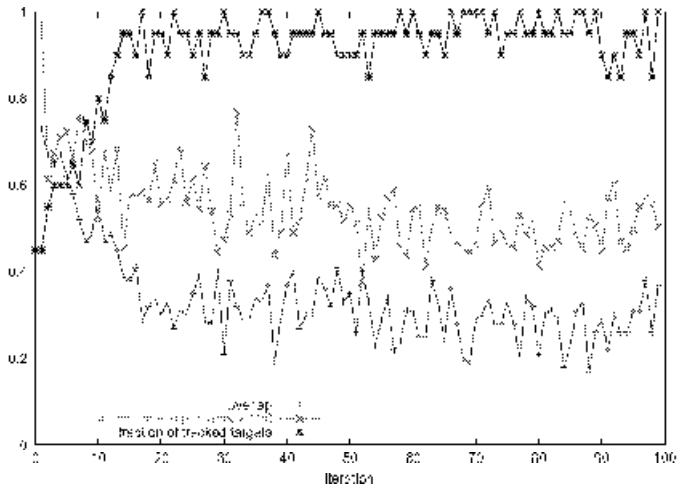

Figure 5: Quality of solution, plus measures of communications cost for 1000-fold speeded-up simulated annealing.

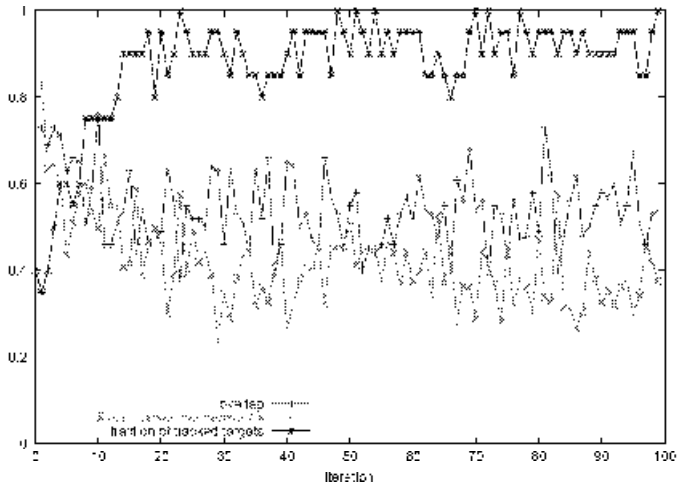

Figure 6: Quality of solution for 10,000-fold speeded up simulated annealing.

and anneal again to convergence. This process can continue as long as there are targets to be tracked. In a fully distributed implementation of our approach, the time steps for updates could either be roughly synchronized or asynchronous. At present, our simulations assume synchronous time steps.

In fact, this process of reevaluating our solution in the light of changes in target positions involves a "renegotiation" of the sensors' tracking assignments, very much like what all the multiagent methods carry out. By controlling the size of the temperature "bounce," we can compare the results of "greedy" one-step agents with what will happen with increasing degrees of renegotiation. Our conclusions are that the results of modest amounts of renegotiation sometimes help, and sometimes hurt, while more systematic renegotiation, with enough retries to work out mistakes introduced, is needed. Our systematic annealing to a modest temperature always gave still better results, but the further improvements available are very small for easy cases, and of greatest value close to the phase boundary, when the number of targets is sufficient to begin to overwhelm the sensor array. Another way to look at the tradeoffs involved is that the boundary of the parameter space in which good solutions are quickly found shrinks as the agents get simpler and are allowed less communications bandwidth and time to do their jobs.

We achieved rapid simulations of target assignment for tracking with both 100 and 400 sensor arrays, and more than 30 targets (with 100 sensors) or

up to 120 targets (with 400 sensors) moving rapidly and randomly through the arrays. We also imposed the restriction that changing the particular 120 degree sector into which a sensor is looking should be done as infrequently as possible to avoid delay and conserve power. While the quality of our results is reduced, the method still works well. We interpret the result of this restriction as a change in the boundaries of the "phase diagram".

In essentially all studies we assume that targets can be seen by the sensors up to a radius of 30 units (3 times the lattice constant of 10 units). We vary the range over which communications links go from certain to be working to certain to fail. We either quote this parameter as the "communications range (CR)" in the same units or plot data against the mean number of sensors in communication. We study values of CR from 1 to 20, of which the values CR= 7, 10, and 14 are relatively plausible cases. The numbers of neighbors accessible in each case are CR7 (2.8), CR10 (3.86), and CR14 (6.16).

Finally, we explore the phase boundary for the physical modeling approach, with gentle annealing during each time step. We then compare these results with the results in which "renegotiation" during each time step is restricted or eliminated. We then introduce methods of controlling antenna sector usage more precisely, and give some measurements of the solution quality changes that result.

3 Phase boundary for target assignment

The phase space of possible target swarm/sensor array combinations has at least three parameters which we can study by considering sample scenarios in simulation. These are

- Average number of sensors that communicate with a direct single hop
- Average number of sensors in range of an arbitrary target
- Ratio of total number of targets to total number of sensors

We have considered the effect of target detection range, and concluded that as long as targets are detectable well outside the range of fast communications, the first parameter, communications range, is the more critical. We study the effect of varying the ability to communicate from a situation where each sensor can reach only one or two neighbors in a single hop, to where a dozen or more neighbors can communicate directly.

The second summary figure (an example is shown in Fig. 4) after each movie is a scatter plot, for each of the targets that entered or crossed the

sensor array, which shows as a circle the total number of time steps of successful tracking by the array, and as a cross the longest consecutive string of time steps in which tracking was achieved. The horizontal axis in this figure is the number of time steps that the target was within range of the array. When all the data points cluster close to the diagonal of this chart, we are successful in our tracking. Without an accepted model of how the sensors' tracking information will be used, we do not know whether the total fraction of the time for which a target has been accurately tracked or the time for which tracking is continuous is more important. If we can afford the overhead of creating a target agent to develop information about a particular target and to maintain its identity as it moves from one group of sensors to another, the total fraction of time the target is resolved is probably critical. If we want to use a simpler, more ad hoc architecture, continuous tracking may be critical. The fraction of time that a target is tracked, averaged over all targets that appear during a particular scenario, is the easiest single measure of effectiveness to use in our analysis. For example, consider the loss of solution quality when running fewer iterations at each step in the annealing process (below). The average fraction falls only a little as we reduce the computing effort by a factor of 10/6 in an easy case (10 targets, 100 sensors, good communications). In a more difficult scenario (20 targets), the falloff of average coverage is faster at first, and then accelerates at speedups beyond 1000-fold. As a result of tests like that shown in Fig. 7, we chose 100-fold as our standard speedup factor, to ensure reasonable results in difficult cases.

To determine a phase boundary in the targets/sensor vs. average number communicating parameter space, we looked at both average coverage fraction and the distribution of continuous tracking times. In both the average coverage and in the number of targets which lived at least 30 time steps and were covered continuously for at least 20 time steps, a "knee" in the curves indicated a rapid worsening of results as we cross a phase boundary.

That boundary shows the importance of communications restrictions in pulling the limiting number of targets/sensor down from 1/3 to nearly 0, a result of the increasing difficulty of finding communicating neighbor sensors with which to share tracking information.

The resulting phase boundary that we determined from our movies is shown below for the case in which constraints on changing antenna sectors were ignored.

The next variation we have considered is control over antenna sector assignment, since in the present implementations there is a time and power penalty for shifting the Doppler radars from one 120 degree sector to another in order to follow a target which has moved out of view in azimuth. We compare our previous results with adding the restriction that each sensor

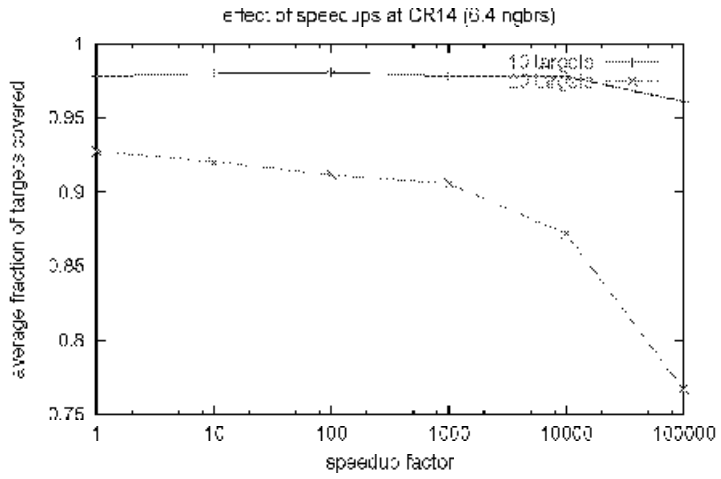

Figure 7: Falloff of quality of targeting solution as we speed up the simulated annealing search at each time step.

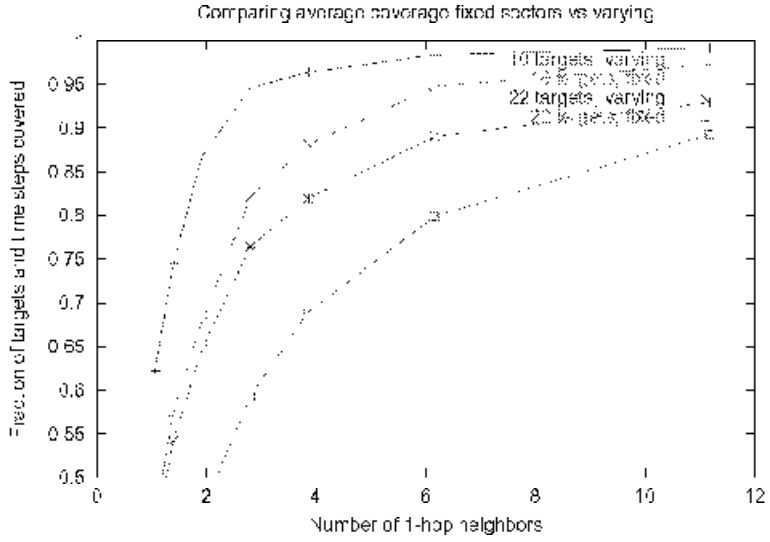

Figure 8: Decrease in coverage as a result of decreased communications range. Both cases in which antenna sectors are fixed at the beginning of each time step and those in which they allowed to vary during the optimization of that time step are shown.

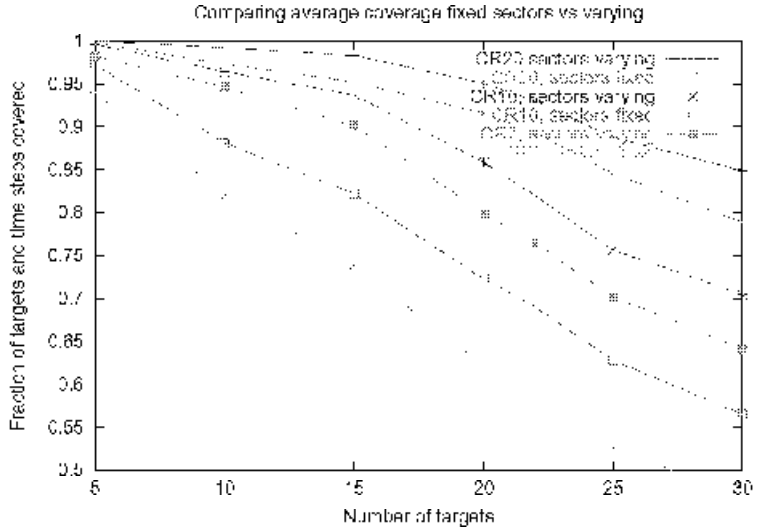

Figure 9: Decrease in average coverage as increased numbers of targets are tracked. Both fixed antenna sector and varying antenna sector results are shown.

chooses the sector in which it will focus at the outset of each time step. During the optimization that takes place over the ensuing time step, the sensor only considers targets which can be seen within the chosen sector. This makes optimization more difficult, although the program does run somewhat faster, so that perhaps shorter time steps could be used.

In Fig. 8, we see that the effect of restricting optimization to a single sector is a small decrease when there are many neighbors to form TCs with, and a severe decrease when there are few. We can turn this chart into an estimate of how much the phase boundary (or threshold of effectiveness) has moved in by comparing the points at which the average fraction of targets and time steps covered is 85%. For the easier case, with 10 targets, this boundary estimate moves from 1.8 to 3.6 neighbors as the sectors are fixed. For the harder case, with 22 targets, the boundary was at roughly 5 neighbors with sectors varying during each time step. With fixed sectors, 10 neighbors are required to get the same coverage. A similar analysis, looking in the direction of increasing target density, is shown in Fig. 9. If again we use 85% average coverage as our threshold estimator, we see that with essentially unlimited communications (CR20) the threshold moves down from 30 to 25 targets when we fix sectors. At CR10, which is somewhat communications-

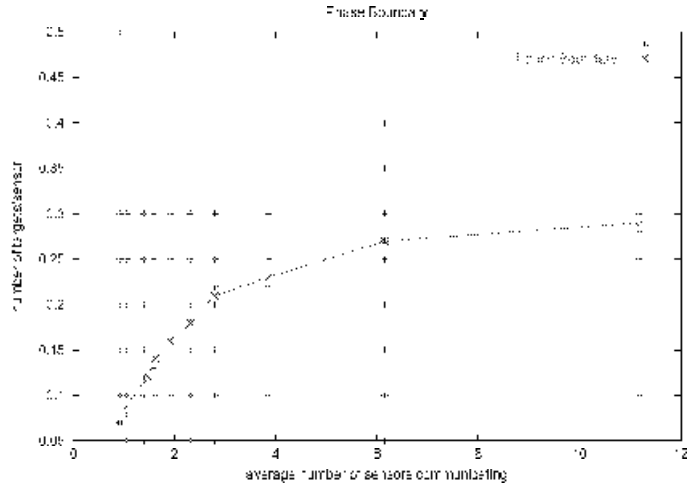

Figure 10: Phase boundary inferred from the movies. Sensors can see targets up to three lattice constants (30 units) away. Number of targets per sensor and number of sensors which can be reached in one hop from a given sensor are the parameters explored here.

restricted, the threshold moves from 20 targets down to about 17. For CR7, at which point each sensor can, on average, talk directly to less than three others, the threshold moves down from 13 to 9 targets.

Remapping the phase boundary shown in Fig. 10 using results in which sector changes were only allowed once within each time step showed only a small shrinkage of the phase boundary. About 8% in average coverage was lost in typical cases away from the phase boundary, but the maximum continuous coverage increased, as sensors were less likely to divide their attention between multiple targets on different timesteps.

4 Larger arrays and measures of solution efficiency

In the physical modeling approach, our total computational costs are at least roughly linear in the size of the problem. This means that a fully distributed computation sees a cost per sensor which is roughly constant as the size of the overall array increases. The only exposure in our approach is that if there are few targets, each TC which is being evolved as the targets move can be quite large, potentially growing to fill the size of the region over which the target is visible. A similar problem occurs in cellular communications, and is solved by power control, to keep cellular clients visible only as far as is necessary to ensure good handoff continuity. We did not implement such "visibility management" in our simulations.

How big might a sensor array get? Using rough numbers, if we deploy a floating sensor every 100 meters around a stationary asset such as a refueling aircraft carrier, up to a radius of a few kilometers, we can easily arrive at arrays of a few thousand sensors which need to be managed and their observations collected, evaluated, and communicated. Our simulations of 100 sensor arrays have required 10 ms per sensor per timestep to execute on 1GHz PIII Linux computers, using the conservative speedup of 100-fold. This can be reduced to 1 ms per sensor per timestep without significant loss of solution quality over most of the phase space. It was hard to resist simulating bigger arrays using the physical modeling approach, simply to make the scalability of the approach evident. We also wanted to see if there is an increase in the computational cost per sensor. We doubled the linear dimensions of the array, and show a series of "movies" with a 20×20 array of sensors at
`http://www.cs.huji.ac.il/~jsch/bigmovies/movies.html`.
One typical example from these, with 40 targets and a communications range of 14 can be found at

http://www.cs.huji.ac.il/~kirk/darpa/film.gif.
The computational cost of simulating arrays four times our previous size did increase, but by only about two times. We think the major compensating factor in this sublinear increase is the larger TC sizes encountered in regions with low target density, since that implies that sensors change their TC affiliation less often.

5 Stupid or Smart Agents?

As already mentioned, our first approach was to work only with our potential function shown in Fig. 1. This function knows that it is better for two antennas to look at the same moving target than to look at different targets and that it is optimal that three antennas look at the same target. This function is indeed very clever: if there are two pairs of antennas looking at two different targets, then it is energetically better that one of the two pairs breaks up, such that three antennas look at one target and the remaining antenna at the other target. This function also deals with all other cases in an excellent way. Since we have invented a very clever potential function for this problem, it is possible that we are also very clever.

However, the agents in these simulations cannot be considered to be clever at all. They change the target they are looking at simply by asking the potential function P. If P tells them that this is energetically better to do, they will do it. They are thus good at obedience, but have not developed or used individual intelligence. The question naturally arises whether the quality of the results can be increased if the agents are made smarter. We performed several trials to make them smarter, e.g.:

- If they were already looking at a target, we did not change the target even if the change led to an energetically better configuration. Thus, the antennas looked at the same target as long as it was within their detection ranges.
- We did the same for a whole TC.
- Agents told their neighboring agents that there was something at which they should look.
- If the targets were about to leave their detection ranges, the agents also propagated the information about the location and speed of the target to more distant antennas, such that they should take over and join the TC, such that the target was kept tracked.

- We used less elaborate utility functions, thinking that less cleverness from outside / our side would relatively increase the cleverness of the single agents.

The results of our experiments are disillusioning: we have to state that – at least for our problem – we have to keep the agents as simple as possible. Thus, the question "How smart does an agent need to be?" has to be answered with "Keep it small, simple, stupid."

6 Conclusion and comments

Statistical mechanics is more than a metaphor for complex systems. Our personal experience in developing optimization or solution schemes for large scale engineering problems is that stochastic methods have advantages in both ease of application and the possibility of generalization. While exact deterministic methods for carefully defined (and well understood) problems can be faster, these often do not generalize to handle constraints that subtly change the nature of the problem. In the real world, these usually get added before one is done developing a solution.

In this problem, one's physical intuition is clear. There are regions of the parameter space in which a solution is possible, and other regions in which it is not. Between them must lie a phase boundary. Obvious questions that this raises are: is it sharp in the limit of an extremely large system, and how far from this boundary are its effects visible in a finite system? Are the associated effects found in the phase transitions that occur in materials also present here, such as diverging correlation lengths (for continuous transitions) or nucleation phenomena (for discontinuous transitions)?

One partial answer comes from Friedgut's theorem about when thresholds are sharp and when they are not [6]. He considers monotonically increasing properties, such as the size of the largest connected cluster, which grows steadily as bonds are added to the sites in a random graph. Statements such as "the largest connected cluster has no more than $\mathcal{O}(\log N)$ sites when there are N sites and M bonds in the graph" pass from being almost always true to being almost always false over a range of ratios M/N which narrows as the numbers N and M increase. Loosely speaking, Friedgut has shown that this is true because the number of example subgraphs which can be used to either prove or disprove the statement grows enormously as N grows. This leads to a weaker definition of sharp threshold than physicists have in mind when they think about phase transitions. But its generality supports the basic physical intuition. A second test is due to A. B. Harris [7]. His criterion is essentially that the correlation length for the type of ordering

which is beginning to occur as one approaches an ordered phase from one in which the order is absent may not increase rapidly enough to allow the laws of large numbers to average the heterogeneous local environment into a sharply characterized equivalent of the phase transition seen in the absence of the disorder. As a result the transition is broadened and modified, but remains sharp. This criterion for transition broadening might apply to some of these heterogeneous optimization and engineering problems. We still do not know if problems become more or less homogeneous as we approach "Avagadro-scale" engineering?

We have observed differences in style in multiagent programming vs. defining a statistical physics-like system that gets the job done. In the physical modeling approach, we are satisfied with ergodic local moves and iterative computation which, when distributed, is quite inexpensive. Multi-agent programmers pay a large factor in programming cost up front, which they expect to recover through the ability to distribute the computation. However, coordination of activities on multiple scales occurs with only modest effort in the statistical mechanics approach. One simply observes correlated activity and may then add additional states. With agents, it seems that growing additional roles, in order to coordinate actions in a hierarchical manner, will be more difficult and computationally more expensive. An interesting but unproven possibility would be to allow the agents to evolve through a scheme such as genetic programming, which prunes a population of agents to leave behind only those which have specialized to the roles which are needed. However it will be difficult to create an environment which is stressful enough to force the agents to exhibit this evolution, yet permit the efficient execution of the agents' basic tasks.

Acknowledgment

This work was performed under a contract with the Defense Advanced Research Projects Agency.

References

[1] S. Kirkpatrick, C. D. Gelatt, Jr., and M. Vecchi, Science **200**, 671 (1983).

[2] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes in C: The Art of Scientific Computing," 2^{nd} Edition, Canbridge Univ. Press (1993).

[3] J. Schneider, I. Morgenstern, and J. M. Singer, Phys. Rev. E **58**, 5085 (1998).

[4] "Stochastic Optimization" – Book in preparation by J. J. Schneider and S. Kirkpatrick (Springer 2004).

[5] ANTS program proceedings, DARPA volume, 2003.

[6] E. Friedgut, J. Am. Math. Soc. **12**, 1017 (1999).

[7] A. B. Harris, J. Phys. **C7**, 1671 (1974).