Chapter 7

Parameter Space Exploration

7.1 The Necessity of Exploration

One major problem encountered in Artificial Society experimentation is the role of numerous exogenously defined parameters. In all but the simplest models, assumptions have to be made which can't be justified by reference to the substantive phenomena under investigation. Because of this fact and the fact that the production of a computational model forces the modeller to be more precise than can be justified by pre-existing theory (if any such theory exists), these kinds of assumptions are stated as exogenously defined parameters. This can be seen in the previous chapter (chapter 6) where many exogenous parameters were specified for the StereoLab artificial society.

Obviously, to execute a model, values have to be substituted for these exogenous parameters. The problem then arises: which values should be chosen? One popular method is to arbitrarily assign values which appear "sensible". This is often done without any justification of the values chosen. Although arbitrary, the method may be adequate for "existence proof" (see figure 3.1 in chapter 3) experiments. In these kinds of experiments

the salient fact is that at least *some parameter values produce a given observable model behaviour.*

If a greater understanding of the model is required, then the role and significance of the particular exogenous parameters need to be understood. For some given observable model behaviour it may be useful to know the *range of parameter values* that give rise to it. Such knowledge represents the beginnings of a domain theory of the model. However, even for a small number of discrete parameters the parameter space of the model may be huge (the product of the number of unique values over each parameter). Given that artificial society simulation models often require substantial amounts of processing time to execute, how can such a space be explored? The following techniques consider that a tractable set of runs could be in the order of thousands.

As stated recently by Teran et al [156], we wish to map the *envelope* of simulation trajectories over some given set of parameters. Teran et al propose a method which also takes account of individual stochastic events - effectively exploring all possible trajectories for some given model. This means that it is possible to *prove the necessity* of a particular emergent phenomena of interest for some given range of parameter values. By necessity is meant that *all possible* trajectories produce phenomenon of interest. They accomplish this exploration by translating the simulation model into a constraint based search over all possible agent trajectories for some given parameterisation. For a complex monolithic model over a large number of cycles this method would be computationally intractable. However, in the method presented, use is made of the modularity possible within a declarative model to automatically search all trajectories for a given subset of the rules which form the model. In this way it is possible to prove the necessity of a particular outcome for all possible trajectories (governed by random agent choices) for the given module of the simulation.

This method offers the exciting possibility of producing *proofs* rather than merely empirical observations from simulation models. However, the computational resources required for even relatively small sets of agents and choices is large (even for a single set of exogenous parameter values). In the StereoLab such a method would be computationally intractable.

7.2 Sampling the Space

In the following sections several methods of sampling the parameter space are discussed: systematic sweep, random sampling and heuristic search.

7.2.1 Systematic Sweep

One method of sampling a parameter space involves a scan or sweep of a range of parameter values in some systematic way [3], [29]. Such a technique may be visualised as the imposition of a grid or lattice structure over the parameter space with each line intersection representing a single simulation run (see figure 7.1). This technique allows some conclusions to be drawn concerning the sensitivity of observable phenomena to particular parameter values of the model. Given one or two parameters and some one dimensional output from a simulation run, results can be presented graphically. The input parameters and output can be used as dimensions in a two or three dimensional space and results plotted. Regions where particular observable phenomena are found can be identified visually (see figures 9.2 and 9.4 in chapter 9 for an example of this).

Such a scan can be easily run in parallel over several machines allowing for reasonably large sample sizes to be made. The Drone software system (as used in [3] and [29]) from Michigan University has been specifically designed to facilitate the task of making systematic scans over a parameter space of a simulation model using multiple machines in

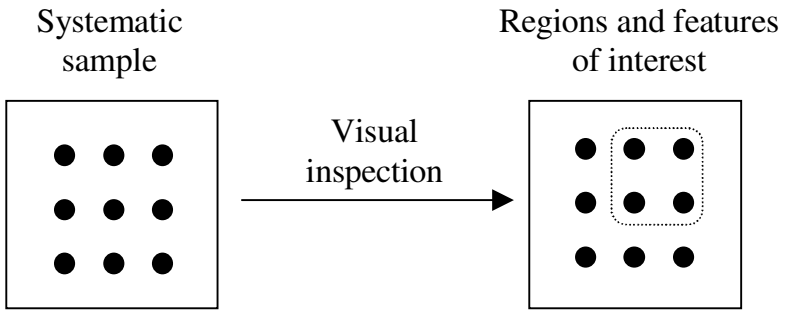

Figure 7.1: Systematic (or sweep scan) sampling of the parameter space and visual inspection is often used for low dimensional spaces.

parallel (see [11] for a detailed description of the Drone system).

Another benefit of systematic scanning of the space is that no assumptions need be made concerning the nature of the space *a priori.* This is particularly useful with output from non-deterministic simulations where noise and different evolutionary trajectories may produce a space with a very non-deterministic complex surface.

However for a large parameter space with many dimensions a meaningful systematic scan may not be possible within a feasible time (for example, if it is not possible to scan at least two values from each parameter). An alternative in such cases is to randomly select points within the space up to available computational resources.

7.2.2 Random Sampling

Random sampling involves selecting parameter values at random (from a uniform distribution) over their ranges. In this way all points have an equal chance of being selected. Given a random sample it is possible to estimate statistical characteristics of the space. In this way the entire space can be characterised (in a statistical sense).

As with a systematic scan, random sampling is easily amenable to parallel execution. The Drone system does not provide this facility, but software designed by the author

does (see section 7.5 below).

7.2.3 Heuristic Search

A more intelligent form of sampling the space may be employed. It is possible to utilise a heuristic search if it is known in advance what kinds of points in the space are desirable or interesting. For example, the simple local search method of "hill-climbing" may be employed if points of interest can be reached by maximising some observable measure from the simulation. Hill-climbing involves starting at some random point in the parameter space and then sampling neighbouring points until some higher value for the measure is found. If such a point is found then this becomes the new starting point for further samping. The process is iterated until no futher movement is possible or some computational limit is reached. Essentially then, hill-climbing attempts to ascend gradients in the space and it will work well in a space with long smooth gradients such that from random starting points in the space hill-climbing can lead to global maxima. A space with many local maxima would be less amenable to simple hill-climbing. There are many proposed methods for avoiding local maxima in a local search such as "simulated annealing" and "guided-local" search (see [159], [144]). The success of each different technique depends on the topology of the space - there is no universal "best search" method.

The use of a hill-climbing search therefore makes assumptions about the nature of the space which may or may not be valid. In general the best way to test the assumptions is to actually run hill-climbing and determine if the results are satisfactory.

Another intelligent way of sampling the parameter space is a novel method used in the "Weaver System" [81]. New points from the space are selected using an iterative method based on the "refutation of unsupported hypotheses". The unsupported hypotheses are

produced via inspection of an induced descision tree (see section 7.4 below for an overview of Weaver).

For the work which follows (in chapter 8) both random sampling and simple hill-climbing were used to sample points in the StereoLab parameter space. Random sampling was used to characterise the huge parameter space (combined with C4.5 - see section 7.3.1 below) and hill-climbing was used to locate rare points in the space (combined with cluster analysis - see section 7.3.2 below).

7.3 Characterising Regions in the Space

Given some sample of points from a high dimensional space (either randomly, systematically or intelligently gathered), the problem remains of to how the points can be interpreted such that the results can be presented in a meaningful form - relating the values of exogenous parameters to observed behaviours. As stated previously, for a low dimensional space with one or two dimensions, a visual method of presentation and inspection is often sufficient for this purpose. For a high dimensional space however, this option is not possible.

One approach to aid interpretation is to locate regions within the space which bound points with similar properties. If some subset of points in a sample can be categorised as having a given property of interest then methods can be applied which attempt to find approximately homogenous regions of such points in the space.

One method of specifying a region within a parameter space is to enumerate a set of intervals, one interval for each parameter. Such a region is therefore a cuboid with dimensions equal to the number of parameters and a volume equal to the product of all the interval sizes. The task of identifying regions bounding points with a given property of interest can be constructed as a process of recursive splitting of the parameter space until

regions are found that bound points which all have the property. This is essentially the task performed by a class of decision tree induction algorithms developed within the machine learning community (see section 7.3.1 below).

Another method of locating regions with high concentrations of points with a property of interest is to use some form of cluster analysis over only those points from a sample which have the property. The result of a cluster analysis is not a set of strictly bounded regions (as produced by decision tree induction methods) but an assignment of each point to a particular cluster (see section 7.3.2 below).

7.3.1 The C4.5 Classification Algorithm

The C4.5 classification algorithm [139] induces decision trees from a sample over a space of parameters (attributes) for a given categorical variable (in this case some category of observable phenomena of the simulation run). The algorithm works by recursively splitting the parameter space along the dimension which produces the highest "information gain" (based on information theory) over the sample. This process continues until a region is homogenous (based on the categorical variable) or some minimum size (based on the number of points in the sample which fall within a region) is reached. In the latter stopping condition, regions may be found which are not homogenous if the minimum size (or weight in C4.5 terminology) is more than 1. In order to apply C4.5 to a sample of the space it is necessary to categorise each point in the sample. For a one dimensional output value, thresholding can be used giving a set of classes over the output. Figure 7.2 shows schematically the application of C4.5 to a random sample producing class homogenous regions. Complex output requires sophisticated methods of categorisation. In the Weaver system (see below) where the output to categories is a time series and the phenomena of

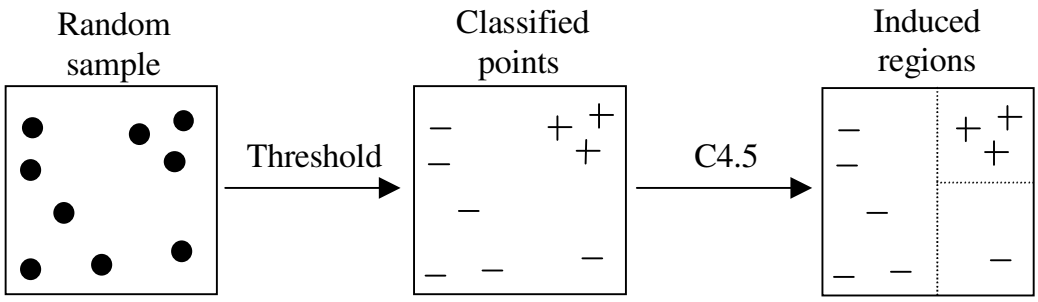

Figure 7.2: Points sampled randomly from the parameter space can (if categorised) be processed by the C4.5 algorithm to induce a set of category homogenous regions of the space.

interest are various properties of the series, a human observer categorises the output. This obviously puts practical limits on the size of any sample which can be processed since each point in the sample requires user intervention.

In the work presented in the following chapter (chapter 8) other classification algorithms could be used in place of C4.5 (e.g. a back-propagation multi-layer neural network). However, one of the main benefits of C4.5 is that the induced decision tree produced is generally more easily interpreted by a human user than the weights produced by neural network learning algorithms. A neural network does not produce a list of regions defined over the input space but rather a set of weights which implement a mapping. In order to find what regions have been induced an investigation of the weights is required. There is no simple method of extracting the induced regions.

7.3.2 Cluster Analysis

Cluster analysis techniques partition a set of points into some number of clusters. In the analysis which follows (see section 8.2 in chapter 8) k-means clustering was used [151]. This technique uses a heuristic search to minimise an objective function (this being a simple form of hill-climbing). The search space consists of all possible allocations of

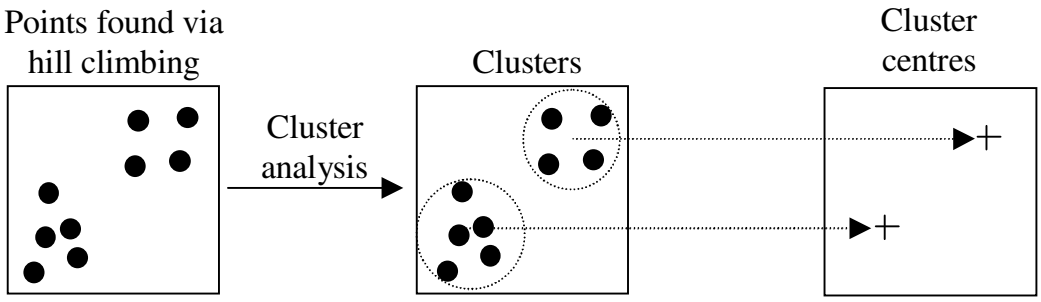

Figure 7.3: Cluster analysis applied to a sample of points found via hill-climbing may be used to find cluster centroids.

points from the sample between some specified number of clusters. Searching the space involves the movement of points between clusters to minimise the objective function. The objective function is based on the sum of the mean distances of all points from their cluster centroids. Various distance metrics may be employed[1], and obviously, data has to be sensibly normalised for the given metric. It should be noted that the k-means algorithm requires that the number of clusters is specified *a priori* and does not guarantee to find the global minimum. Since this is the case it is considered prudent to run the search several times with different initial starting conditions (initial centroid values). Also since an *a priori* choice for the number of clusters needs to be made it is also often considered useful to run k-means with a range of cluster numbers and make a choice based on the final value of the objective function found for each clustering [151]. In the analysis which follows (section 8.2 in chapter 8) hill-climbing was used to find points of maximum co-operation in the StereoLab parameter space (see section 6.5.4 in chapter 6). These points were then analysed using k-means cluster analysis. Figure 7.3 illustrates this process schematically.

[1] In the work presented in this thesis simple Euclidean distance is used.

7.4 The Weaver System

The Weaver system [81] is a recent application of automated induction (using C4.5) within an integrated theory development and testing methodology (see figures 3.3 and 3.4 in chapter 3). The Weaver system has been designed for the "development and exploration of existing knowledge in complex, knowledge- and data-poor domains" [81]. Specifically the system has been applied to an ecological domain: the possible causes of population cycles in red grouse (*Lagopus, Lagopus scoticus*).

Weaver works with assumptions specified as enumerated (nominal) variables representing code fragments specifying alternative model mechanisms. Weaver therefore allows the specification of various alternative model assumptions in the form of program code units. The system can then compose those units into a working simulation automatically (on-the-fly) during theory development. This allows for much greater flexibility in the form that assumptions can take than simply varying parameter values (as in the StereoLab, chapter 8). Both macro (population level) and micro (individual based) model mechanisms can be compared.

Figure 7.4 gives an overview of the semi-automated Popperian [136] method of "critical discussion" used by the system to explore, compare and refine theory (in the form of a C4.5 induced decision tree). This method can be compared to the theory construction methodology outlined in chapter 3 (see figure 3.4 in chapter 3). Notice that the user is responsible within Weaver for classifying the outcome of simulations. This is a recognition of the often highly qualitative and / or visual nature of such classifications. In the work which follows for the StereoLab (chapter 8) this problem was avoided since the measure used to classify the result of a simulation run was purposefully kept simple. However, it would seem that for all but such simple classifications user inspection would be required and

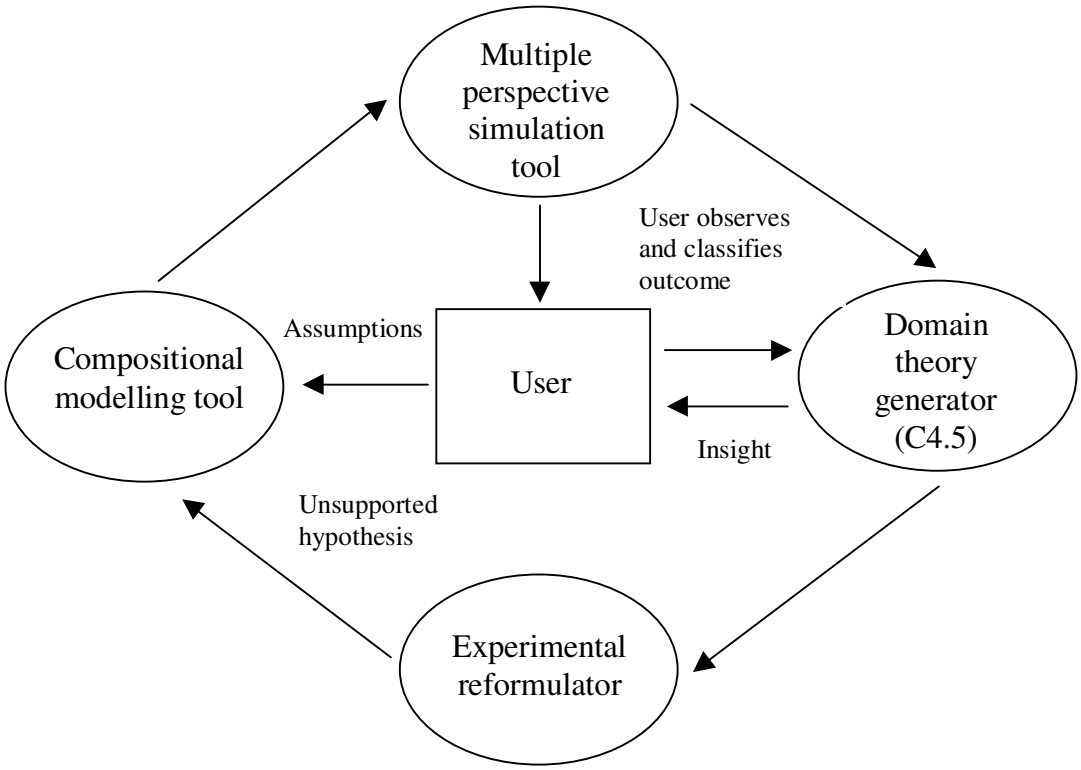

Figure 7.4: Theory development in the Weaver system using inductive learning and experimental reformulation (adapted from [81]). This is an application of a theoy building methodology (see chapter 3, figure 3.4).

this puts practical limits on the number of runs that can be processed. The application of Weaver to the ecological domain of red grouse population cycles involved the user classifying a population size time series into one of several category types (steady state, extinction or various yearly cycles).

Central to the Weaver system is the domain independent "experimental reformulator". Cleverly, Weaver searches the domain theory (induced decision tree) supplied by C4.5 for hypotheses (leaf nodes) which are *unsupported* by any experimental results. A new experiment (set of assumptions) is then generated based on the reformulation of the *most similar* past experiment (in terms of assumptions) such that it covers the unsupported hypothesis. Essentially then, Weaver attempts to refute unsupported generalisations made by the domain theory through experimentation. This process is continued until the user decides that the domain theory is "stable" (that is the induced tree has changed little over several refutation iterations).

Weaver requires that each parameter (assumption) is of a nominal enumerated type (i.e. takes one of a set of unordered values). Continuous numerical parameters have to be converted into nominal types which puts restrictions on the kinds of domain theory that C4.5 can generate. It would not be possible to induce a rule of the form A1 > C (where A1 is some numerical assumption and C is some constant).

Weaver was applied to an ecological domain in which the phenomenon of interest was not sparsely distributed in the parameter space. Consequently a few initial random samples and the application of the "critical discussion" method of theory construction was adequate to produce a stable and useful domain theory without requiring thousands of runs. However, in the work which follows for the StereoLab (chapter 8) this was not the case. High co-operation (the phenomenon of interest) was rare and therefore had to be

actively searched for. The Weaver tool-set would therefore not be directly applicable to all domains. Weaver is a step in the direction of semi-automation of the theory construction methodology. It suggests the possibility that for some domains, certain methodologies can be automated to some extent[2].

7.5 The SampTool Software

Software (SampTool) was constructed which allows for sweep (linear and log scales) and random sampling from the parameter space of a simulation. Additionally SampTool provides simple hill-climbing over the parameter space for a specified number of steps against a given output measure. Hill-climbing involves starting from a random point in the parameter space (by executing a simulation run) and then selecting a neighbouring point at random (executing a simulation run) and moving to that point if the result of the run is better than the initial point. This process is iterated the specified number of steps. The neighbourhood is defined as all points within one unit (on any number of dimensions).

Figure 7.5 shows an overview of the SampTool system. The simulation parameter specifications file defines a parameter space for a simulation. Figure 7.6 shows an example simulation parameter specification file for the StereoLab. Notice that the nature (input or output), type (float or int), treatment (fixed or variable) the scale (linear or log) and the increment interval for each input parameter is specified. The user activates SampTool by selecting the type of sample required (scan, random or hill-climb) and the number of sample points to make. SampTool then executes the required simulation runs. The results of the runs are written to a sample file. SampTool provides a set of utilities that allow a sample file to be converted for loading into other packages for analysis, visualisation, processing

[2]It would be interesting to apply other machine learning systems, than C4.5, to simulation runs. For example, BACON [105] which attempts to generate mathematical "laws" which link variables.

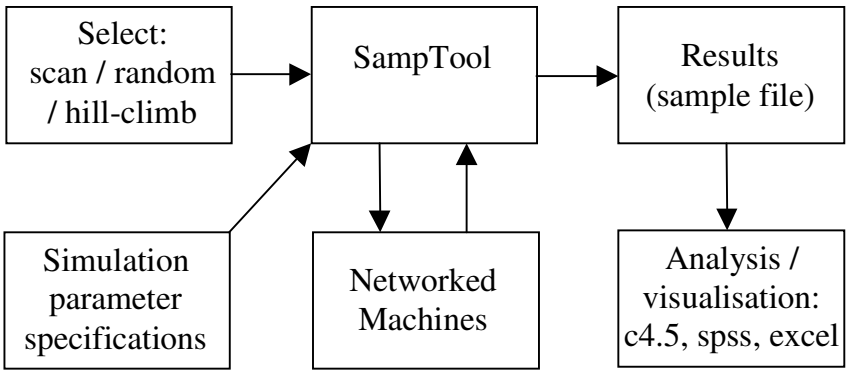

Figure 7.5: A schematic of the SampTool system. A general purpose tool for sampling a parameter space. Figure 7.6 shows an example specification for the StereoLab simulation.

or storage (e.g. SPSS, MS-ACCESS, C4.5, MS-EXCEL). Each individual run within the sample file is identified by a unique ID. SampTool can be used to re-execute any individual run from a sample file by supplying the ID (either with the original or an alternative random number seed). This is useful for more detailed post-hoc analysis of individual runs (to collect time series data for example).

The SampTool schedules and executes simulations in parallel over any number of networked machines sharing a common file-store. In the context of hill-climbing, each machine may run an independent hill-climbing process. SampTool, like Drone (see section 7.2 above), is a general purpose tool and can interface to most simulation models[3].

7.6 Summary

When exploring a large parameter space of a model two issues need to be addressed: 1) how to sample the space, 2) how to analyse the sample. In a low dimensional space it might be possible to systematically scan the space and then visually examine the results. This is a common method used in much simulation work. However, in a high dimensional

[3]See Appendix A for details.

```
DEF 40 "Stereotypes simulation" 1 1

    # "EXECMD"  stereo
    # "EXEPATH" stereotest/
    # "RESPATH" ./

# "name  type  trt low hi  scale inc fix    Desc"

INP "S"  int   fix  0    0  lin  1   101    "number of locations in environment"
INP "N"  int   fix  0    0  lin  1   101    "number of agents in environment"
INP "Nc" int   fix  0    0  lin  1   100    "number of system cycles to execute"
INP "Nr" int   fix  0    0  lin  1     1    "number of runs to execute"
INP "T"  float fix  0    0  lin  0.1   3    "satisfaction threshold"
INP "Pp" float fix  0    0  lin  0     1    "PD outcome payoff value P"
INP "Pt" float fix  0    0  lin  0     5    "PD outcome payoff value T"
INP "Pr" float fix  0    0  lin  0     3    "PD outcome payoff value R"
INP "Ps" float fix  0    0  lin  0     0    "PD outcome payoff value S"
INP "P"  float fix  0    0  lin  0.1   1    "prob of satisfaction test"
INP "B"  int   var  4    8  lin  1     0    "number of bits in label string"
INP "M"  int   var  2   10  lin  1     0    "number of stereotypes agent can hold"
INP "Pm" float var  0    1  lin  0.1   0    "probability of meme propogation"
INP "Mt" float var  0    1  lin  0.1   0    "mutation rate for label bits and rules"
INP "Ci" float var  0    1  lin  0.1   0    "factor by which to increase confidence"
INP "Cr" float var  0    1  lin  0.1   0    "factor by which to decrease confidence"
INP "Ms" float var  0    1  lin  0.1   0    "mutation size range for strategy parts -Ms to +Ms"
INP "Fg" float var  .1   1  lin  0.1   0    "prob. of a game interaction in cycle"
INP "Fc" float var  0    1  lin  0.1   0    "prob. of a cultural interaction in cycle"
INP "Fm" float var  0    1  lin  0.1   0    "prob. of a random agent movement in cycle"
INP "Bf" float var  0    1  lin  0.1   0    "proportion of bits that are fixed"
INP "Bg" float var  0    1  lin  0.1   0    "game label bias"
INP "Bc" float var  0    1  lin  0.1   0    "cultural label bias"
INP "Tg" int   var  1   10  lin  1     0    "refusals before forced interaction for game"
INP "Tc" int   var  1   10  lin  1     0    "refusals allowed forced interaction for cultural"
INP "Vc" float var  0    1  lin  0.1   0    "cultural interaction window"
INP "Vg" float var  0    1  lin  0.1   0    "game interaction window"

# "name   type  Desc"

OUT "CC"  float "prop of mute cooperation"
OUT "DD"  float "prop of mute defection"
OUT "DC"  float "prop of DC interactions"
```

Figure 7.6: An example simulation parameter specification for the StereoLab. See appendix A for details of the format and meaning of the parameter specification file.

space, visualisation becomes problematic and the size of the space may make meaningful systematic scans computationally intractable. One possible solution is to randomly sample the space and then apply an induction algorithm to characterise regions in the space with common properties. Another approach is to actively search the space in some way for regions of interest. Alternatively, when a phenomena of interest is sparse within the parameter space then a form of *hill-climbing* may be applicable. If hill-climbing is used then some set of homogenous points may be located. In order to help interpret those points, *cluster analysis* may be used to identify spatially distinct regions within the space. In order to facilitate the sampling process for the StereoLab the SampTool software was constructed. SampTool allows for systematic, random and hill-climbing sampling to be made over a parameter space utilising multiple machines over a network. SampTool schedules execution of simulation runs and collates the results, providing conversion utilities allowing those results to be loaded into existing analysis packages. SampTool is a general purpose utility that can be interfaced to most simulation software.

In the next chapter the SampTool software is applied to the StereoLab (as outlined in chapter 6) to find regions within the parameter space which produce high co-operation. Both random sampling combined with C4.5 and hill-climbing combined with cluster analysis are employed.