Generation of on-line randomness by cellular automata

Patrizia Mentrasti, Silvio Capobianco
Department of Mathematics, Università di Roma "La Sapienza"
P.le Aldo Moro, 2 I-00185 Roma (Italy)
email: p.mentrasti@caspur.it, capobian@mat.uniroma1.it

29th September 2000

Abstract

We describe a cellular automaton expressly designed for random number generation and efficiently implemented on the CAM-8 engine. We may think of this system as a local and parallel "oracle" which, at each site x and time t, returns random a bit with probability p which may be assigned as an arbitrary function of the site.

Keywords: cellular automata, random sequences, oracle.

1 Introduction

We discuss a scheme of algorithms generating pseudorandom bit sequences, that can be quickly used by complex dynamical systems.

Roughly speaking, a complex system [2] describes the interaction of several moving individuals. The correct representation of a complex system is not a simple matter; in fact, several choices are needed: "how many are the variables?", "which is their meaning?", "how many different states (i.e. external displacements) can they assume?", "how to describe the interactions with each other?", "how to consider the actions of unknown agents?", "how to measure individuals state changes with respect both to the space and to the time?". So, the simulation of complex dynamic systems often requires "good" pseudorandom bits sequences [13, 21].

These random variables will often be represented by suitable "dice rolls". From a philosophical point of view, dice rolls, or equivalent procedures, can represent queries to an "oracle" [6], that is, they describe the fact of asking a "Sibyl" (a diviner who answers whatever question) something about an interesting but unknown thing; consequently, an individual, before making a decision, takes into account the Sibyl's response. Our interpretation both of "randomness" and of "oracle" is only one of the possibilities; concerning the behaviour under uncertainty conditions, we follow Chaitin's definitions [8, 9].

The most used random generators have the following features. The generation process is iterative and off-line; that is, the devices that produce sequences can be different from those requiring them; at least, even if all the algorithms for the simulation of complex systems evolution and the generation of random bits sequences do run on the same device, this is performed in different time units. In fact, the machine could even have prepared the required bits when the simulation starts. Then, each parallel processor possesses the same "random" bits at the same time.

Here, we want to introduce a new approach. Our random generation is parallel; still, it uses only a cellular automata machine both for the experiments and the sequences; bits are produced on-line, that is, at the instant when they are to be used; moreover, as the bits are produced locally, they can be different from site to site; still, they can follow different probability distributions, that may not necessarily be uniform; finally, the parameter of each distribution can be chosen by the user when starting and also be different from site to site.

As to our knowledge, the idea of using the same cellular automata device for complex system simulation and random sequence generation is already found in Toffoli and Margolus [26] with the STIR rule, that in fact we do use in a revised version. Felici and Mentrasti [11] proved the goodness of STIR as a generator for equidistributed sequences. On the contrary, the local and non uniform approach previously described is new and, in our opinion, fruitful for several applications. In particular, the authors yet used this idea in some applications, mainly concerning traffic evolution simulations [3, 4, 10].

Here, we want to characterize from a theoretical point of view the idea of "local oracle" and also give a practical demonstration of its feasibility. In fact, we describe a simple way to "locally" generate random distributions having any parameter. Then, we test the produced sequences of bits with the chi-square method. As we shall see later, test results satisfy the required expected value. We also make some considerations about the period.

The engine we used is the CAM-8 [17, 25, 16], an evolution of the older, simpler CAM-6 model [26] based on the "programmable matter" technology. A CAM-8 is made of several identical *CAMboxes*, connected in a three-dimensional structure, and driven by an external machine that hosts the control software; each CAMbox contains eight modules, that contain the basic elements like memory and processors, and operate in parallel. Once the space for the experiment has been defined, the available memory is equally distributed through all its points; the neighborhood and the rule of evolution are defined via software. The device offers great versatility and has a lot of models already implemented [24, 27].

Since the CAM-8 device can run several CA simultaneously, we can have one or m ore of them working at the generation of the sequences. This leads to a greater flexibility than that of the classical approach, and eliminates any need of an external, sequence-generating device.

We effectively implemented two models, which slightly differ in the use of STIR rule and both of them satisfy our initial requirements.

2 Cellular automata with oracles

Imagine an infinite grid. Think of the nodes of that grid, which we will call *cells* from now on, as entities (really, they are finite-state automata [19, 28]), all of the same kind, capable to assume various states from a finite set. Suppose that each of these cells watches some of the others, and modifies its own state according to theirs. Lastly, think of the change of state to happen at the same time for all cells. This can be a first idea of what a cellular automaton is.

More formally:

Definition 1 *A* cellular automaton *is a quadruple* $< L, S, N, f >$ *where:*

- L *is a d-dimensional regular lattice, called* support *of the automaton; its nodes are called* cells *of the automaton;*
- S *is a finite, non-empty set called* set of the states*;*
- N *is a finite subset of* L*, called* neighborhood index*, containing the null vector and such that, if* $\mathbf{v} \in N$ *and* $\mathbf{r} \in L$*, then* $\mathbf{v} + \mathbf{r} \in L$*;*
- $f : S^{|N|} \to S$ *is the* transition function*.*

Given a cellular automaton $< L, S, N, f >$ and a cell $\mathbf{r} \in L$, the set $N(r) = \{\mathbf{v} + \mathbf{r}, \mathbf{r} \in N\}$ is called the *neighborhood* of the cell $\mathbf{r}$: the elements of the neighborhood of a cell are called its *neighbors*. Each cell is neighbor of itself.

The previous definitions are intended for infinite cellular automata, but they can be quickly adapted to finite ones, having the topology of a torus.

Since the device used realizes finite toroidal automata, and the tests can be performed only on a finite number of finite sequence, *from now on we will always consider finite toroidal cellular automata.*

Definition 2 *We call* oracle *a device of any kind that answers in time* $O(1)$ *to a user-chosen problem.*

We don't require the answer to be favorable to us. The oracle can either be "global", giving the same response to each cell, or "local", giving possibly different responses to different cells. The classical definitions of an oracle are in the sense of a global one. On the contrary, we chose the local kind, for the following reasons:

- we want to be able to generate different probabilities in different places;
- we adhere to the "locality" idea of the machine;
- this way is closer to the reality evolutions.

Definition 3 *A* cellular automaton with oracle *[18] is an automaton with an additional device, acting as an oracle, and whose cells are given three special states, called* ask-oracle*,* oracle-yes*,* oracle-no*.*

Each time a cell enters the *ask-oracle* state, it examines the word, or *response*, given by the oracle and then goes into the state *oracle-yes* or *oracle-no*, according to the fact the word is or not in the oracle language.

Our goal is to construct a cellular automaton that can act as a local oracle: more precisely, we want that it generates binary pseudorandom sequences. We also want that each element of our sequences can be 1 with probability p or 0 with probability $1-p$; sequences generated at different places should be different from each other. Moreover, we want that the parameter p can be chosen at will, and that different regions of the automaton space can generate sequences with different parameter. Finally, we want the generation of the sequence to be performed with no use of external devices. In fact, we want to use the CAM-8 cellular automata machine.

All of these requests are effectively performed by our model.

3 The STIR cellular automata rule

A well known [26] unidimensional cellular automaton capable of producing highly chaotic configurations has von Neumann's neighborhood and follows the local evolution rule:

$$C_{t+1} = (C_t \vee E_t) \oplus W_t \tag{1}$$

where $A \vee B$ is usual OR rule and $A \oplus B$ is XOR rule, i.e. the modulo 2 addition.

This rule can be easily extended to a two-dimensional space. One of the most successful variants has been developed by Toffoli and Margolus [26]: it is called STIR and has the following scheme:

$$C_{t+1} = C_t \oplus N_t \oplus W_t \oplus (S_t \wedge E_t) \tag{2}$$

where $A \wedge B$ is the usual AND rule.

Felici and Mentrasti [11] proved that this rule, and other two-dimensional variants of (1), have properties which are considered suitable for pseudorandom sequence generation:

1. they generate very chaotic sequences, both in time and in space;
2. with several initial configurations, they do quickly enough converge to uniform distribution;
3. if the values of a cell and of its neighbors at time t are independent and uniformly distributed over $\{0,1\}$, then C_{t+1} is uniform and independent from C_t.

The second feature of the STIR rule makes it very good if the sequence to be generated should have $p = 1/2$, but also unsuitable for any other value of p: so we cannot achieve our aim by simply using it, and need something more.

4 The BBM gas model

Being conceived and developed as a universal computational model, BBM (Billiard Ball Model; see [26] for a complete description) derives from the classical kinetic gas theory. The BBM model considers solid balls, having equal finite size and mass, moving on straight lines and colliding with each other and with mirrors: this not only simulates the motion of perfect gases, even preserving reversibilities, but, by opportune interpretation of collisions and trajectories, can reproduce any logical circuitry.

To implement this model on the CAM-6 (and then the CAM-8) device, Toffoli and Margolus developed a substructure, that can be interpreted as a gas of particles that interact with each other only by collisions. However, these do not always preserve momentum: that is, a particle colliding with a wall "bounces" in the same direction whence it came, instead of performing a specular reflection. To overcome this, they devised a clever trick that makes possible to construct momentum-conserving balls.

This substructure, that will be referred to as "gas" from now on, has some interesting properties. Firstly, it is possible to delimit regions of the support using only aggregates of gas particles: these "mirrors", or "walls", can be thought of as pieces of frozen or crystalized gas. Still, the total number of particles is a time constant and this should give a chance to control the distribution along time. Finally, there is already a very good implementation of the BBM model on the CAM-8.

By only using BBM, we can delimit a region of the space and fill it with free gas particles. Having chosen a cell in that region, we state that it generates the digit "0" if there is no gas particle in it, and "1" otherwise; by doing this at each step of the time evolution of the gas, we generate a sequence of bits. To achieve the correct value for the parameter p, we properly choose the density of the gas.

So we may think that, by suitable initialization, we can easily have a random bit sequence generator with desired probability. But at this point, the BBM model shows its limitations:

1. particles can aggregate into blocks, thus modifying the probability distribution;

2. the model is totally (and easily) reversible, and so everyone can attack it.

Our new candidate is defective where STIR was good, and vice versa: neither of them can do the job alone.

5 In unity is strength

At this point, we have at our disposal:

- a generator that has very good statistically measured properties, but is able to realize only uniform distribution;

- a gas model that can be treated easily, but is not able by itself to pass the standard statistical tests.

We *merge* the two models to obtain a controllable and stable random sequence generator. This is the idea:

1. We use at the same time many regions of BBM gas delimited by walls and initialized with different densities of gas. We call the delimited region *cage*. These cages will be our local oracles. We use also a STIR-like generator, that acts as an "oracle for the oracle" and will never be seen by the underlying simulation system.
2. The STIR-like generator evolves normally.
3. The BBM gas evolves normally, except for when it is forced to ask the STIR-like generator, to know if it must alter its movement and, if it does, how.
4. The oracle response is positive or negative, according to the presence or absence of BBM gas particles in the appropriate region point.

We claim that this solution works.

6 The first model

Our first model uses BBM along with the following implementation of a 2-dimensional rule similar to STIR:

$$C_{t+1} = (C_t \wedge E_t) \oplus W_t \oplus N_t \oplus S_t \tag{3}$$

As we see below, the pair (C_t, C_{t-1}) can be read at time t and used as an oracle by the BBM automaton; if we want that the responses are independent, then a cell that asks the oracle at time t and uses the pair cannot ask it again at time $t+1$: to do this, we simply state that each BBM cell asks the oracle every third step.

Our rule is as that: if there are one or two particles, the pair (C_t, C_{t-1}) determines an alteration of the normal evolution in the BBM cell, in one of the following ways:

- no change;
- 90-degrees clockwise rotation;
- 90-degrees counter-clockwise rotation;
- reverse motion direction.

For slight problems deriving from the implementation, we state that there is no change if there are three or four particles in the cell.

At the aim of performing the necessary tests of statistical compliance, we created a statistically significative number of 2-dimensional configurations of the following kind:

- a cage of 20×20 cells is placed at the center of BBM space;
- BBM space inside the cage is initialized by a distribution $f(p)$ related to the parameter p of the Bernoulli distribution we are willing to reproduce; since presence or absence of balls in a cell is interpreted as 1 or 0 respectively, and the CAM-8 implementation of the BBM model requires four bits, we put $f(p) = 1 - \sqrt[4]{1-p}$ for each of them;
- the STIR-like generator is initialized with random uniform distribution.

Having done that, we select a cell in the cage, most often near its center, and collect a long sample by reading the presence (1) or absence (0) of particles in the cell. Then, the sample undergoes the *chi-square test* [15], a standard procedure of comparison with an expected distribution. To check if the distribution can degenerate to a different one, we repeat the test for the first 10000, 20000, 30000, 40000 bits of the sequence, and then for its whole. As we want to verify the improvement caused by the STIR-like rule, we repeat the same test to the sequences obtained by using the same initial configurations, but without the query to the STIR-like generator.

The chi-square test is very simple. Let's suppose we have a random quantity X that can take a finite number of values $x_1, x_2, ..., x_k$, and let p_i be the theoretical value of $P(X = x_i)$. If N_i is the number of times that, out of n tosses, $X = x_i$, then

$$V = \sum_{i=1}^{k} \frac{(N_i - np_i)^2}{np_i} \tag{4}$$

is a random variable. We observe n tosses of X, and then compute V, obtaining a real number v.

Now, we want to find the probability that $V \leq v$: it should not be too high, because this would mean that X does not follow the expected distribution at all; nor it should be too low, because this would suggest that we can easily guess the behavior of X. For n large enough, then *Pearson's Theorem* [15] states that:

$$P(V \leq v) \sim \frac{\left(\frac{1}{2}\right)^{\frac{k-1}{2}}}{\Gamma\left(\frac{k-1}{2}\right)} \int_0^v x^{\frac{k-1}{2}-1} e^{-\frac{x}{2}} dx$$

that is,

$$V \sim \chi^2_{k-1}$$

Since the density function of a chi-square distribution is positive, we can derive the response of our test simply by the value of v. In our case:

1. if $v \geq 0.00393$ and $v \leq 3.841$ the test *succeeds*;
2. if $v < 0.00016$ or $v > 6.635$ the test *fails*;
3. otherwise the result is considered *ambiguous*.

Table 1 shows outcomes from these tests. Each sample is shown with its expected probability (Dist.), number of steps performed (Steps), and expected number of 1s (Exp.). The variables: resulting number of 1s (Occ.), resulting value for V (V value), and response of the test (Res.: S stands for *success*, A for *ambiguous result*, F for *failure*), are shown both without and with interrogation of STIR-like rule.

By observing the table, we can see that while using BBM together with STIR-like rule the agreement with expected probability greatly improves, especially when p is not too big.

On the contrary, when p is near 1, probably because of molecule aggregation phenomenon, rejects are too many. Actually, this is not a real problem, since it is always possible to choose $p \leq 1/2$, simply *negating* the response.

Anyway, we want to explore another way, as we are going to see in § 7. Before discussing it, we want to make some considerations about the period length.

6.1 The period length

We are now going to discuss about the period length, that is, we want to limit the estimated number of different random bits we can generate. Roughly speaking, the period length describes how many not repeated bits occur before having a cycle.

Let we have a cage of size $4 \times h \times k$, initialized with parameter p; we can have at most

$$L_{h,k} = \left(\begin{array}{c} hk \\ hk\left(1 - \sqrt[4]{1-p}\right) \end{array} \right)^4 \tag{5}$$

distinct feasible configurations, over a total of possible 2^{4hk}. If the rule is deterministic, then formula (5) gives also the greatest possible period.

The Stirling's rule gives an upper bound for $L_{h,k}$, that is, $L_{h,k}$ has an asymptotic behavior

$$L_{h,k} \cong O\left(n^{-2}{\alpha_p}^{4hk}\right) \tag{6}$$

for an appropriate constant $\alpha_p > 1$.

On the other side, STIR uses a space of $512 \times 512 = 2^{18}$ cells. Not all global configurations are possible, but each of the possible ones can affect "gas" behavior, even if the cage only sees a "mask" of its own size.

Actually, we use only two generated bits among three; but this is equivalent to extract a subsequence in a "random" one and this, as it is known, is equally random.

7 A second attempt

Our second model is obtained by replacing the previous STIR-like generator (3) with the actual STIR rule (2) and also increasing time between interrogations to 7 steps.

To test the new model, we ran it with many of the old configurations and also with new ones. At the aim of examining if even the position in the cage could affect the behavior of the sequence, some of them were different only in the choice of the cell from which the sample was collected.

Our second rule not only succeeds in improving the performance, but also works when p is high: this also comes from adequate use of STIR rule.

Table 2 shows outcomes from performed tests. Variables are displaced as in Table 1.

8 Further developments

One possible improvement is a change in the number of steps between two interrogations.

Another possibility could be a change of gas model: for instance, FHP [28], a very good model of perfect gas, could be used in place of BBM. In this case, it would be necessary to revise the rules of collision, and possibly the shape of cages.

9 Conclusions

We designed a flexible, local, parallel, inside-located, independent random bit sequences generator which is easy both to use and to implement. In fact, our system shows the following properties:

1. paying careful attention while using all the devices, we produce a pseudo-random bit generator with desired expectation and endowed with the just described properties;

2. it is possible to place as many generators as we want, with different, user-chosen probability distributions, on the same cellular automata support.

10 Acknowledgements

We want to thank Fabio Spizzichino and Tommaso Toffoli for helping discussions and encouragements.

References

[1] T. Baker, J. Gill, R. Solovay, Relativization of the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question, SIAM J. Computing 4 (1975) 431-442

[2] Y. Bar-Yam, Dynamics of Complex System (Addison-Wesley, Reading, Mass. 1997)

[3] A. Benigni, Simulazione di una dinamica di traffico urbano mediante l'automa cellulare CAM-8, Tesi di laurea, Dipartimento di Matematica "Guido Castelnuovo", Università degli Studi di Roma "La Sapienza", Anno Accademico 1999-2000

[4] A. Benigni, P. Mentrasti, T. Toffoli, A Model of Urban Transport Simulation Using the Cellular Machine CAM8, in M. Delorme et J. Mazoyer eds. Procs. of Automata 99, Workshop on Cellular Automata, 4th IFIP WG 1.5 Meeting, Ecole Normale Supérieure, Lyon 1999

[5] C. H. Bennett, Quantum information and Computation, Physics Today (1985) 24-30

[6] C. H. Bennett, J. Gill, Relative to a random oracle A $\mathbf{P}^A \neq \mathbf{NP}^A \neq \mathbf{coNP}^A$ with probability 1, SIAM J. Comp., 10 (1981) 96-113

[7] A. Bertoni, M. C. Bollina, G. Mauri, N. Sabadini, On characterizing classes of efficiently parallelizable problems, in Procs. of VLSI: algorithms and architectures, Amalfi 1984 (North-Holland, Amsterdam-New York 1985)

[8] G. J. Chaitin, Information, Randomness and Incompleteness, World Scientific Series in Computer Science 8 (World Scientific Publishing Co., River Edge 1990)

[9] G. J. Chaitin, Randomness and Mathematical Proof, Scientific American, 232 (1975) 47-52

[10] F. M. D'Amore, P. Mentrasti, Simulazione con Automi Cellulari delle Dinamiche di Strutture di Stazionamento, in S. Bandini, R. Casati and G. Mauri eds., Procs. of ACRI '96, Milano 1996

[11] M. Felici, P. Mentrasti, A Secret Key Scheme with Cellular Automata, in S. Di Gregorio and G. Spezzano eds., Procs. of ACRI '94, Rende di Cosenza 1994, 209-219

[12] B. Hayes, The Wheel of Fortune, American Scientist, 81 (1993), 114-118

[13] B. Jansson, Random Number Generators (Almqvist & Wiksell, Stockholm 1966)

[14] W. Kirchherr, M. Li, P. Vitányi, The Miraculous Universal Distribution, preprint

[15] D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (Addison Wesley Longman, Reading, MA 1998)

[16] N. Margolus, Crystalline Computation, in A. Hey ed., Feynman and Computation: Exploring the Limits of Computers, Reading, MA 1999

[17] N. Margolus, T. Toffoli, STEP Technical Reference (The MIT Press, Cambridge 1993)

[18] P. Mentrasti, S. Capobianco, Cellular Automata with Local Oracles, in M. Delorme et J. Mazoyer eds. Procs. of Automata 99, Workshop on Cellular Automata Open problems session, 4th IFIP WG 1.5 Meeting, Ecole Normale Supérieure, Lyon 1999

[19] P. Mentrasti, A. Clementi, P. Pierini, L'automa cellulare: un modello di calcolo e di simulazione, Dipartimento di Matematica, Università degli Studi di Roma "La Sapienza", 1991

[20] P. Mentrasti, O. Tempestini, L'automa cellulare CAM-8: caratteristiche fisiche e dinamiche, Dipartimento di Matematica, Università degli Studi di Roma "La Sapienza", 37/99, 1999

[21] S. Micali, "Perfect" pseudorandom number generation, invited paper, Information Processing 89, IFIP, 1989

[22] R. Morris, Counting Large Numbers of Events in Small Registers, Communications of the ACM, 21 (1978) 841-842

[23] C. Papadimitriou, Computational Complexity (Addison-Wesley Publishing Company, Reading, MA)

[24] P. Pierini, Space-Time Structures for cellular Automata, in S. Di Gregorio and G. Spezzano eds., Procs. of ACRI '94, Rende di Cosenza 1994, 27-37

[25] T. Toffoli, Programmable Matter Methods, in D. Talia and P. Sloot eds. Future Generation Computer Systems, 1998

[26] T. Toffoli, N. Margolus, Cellular Automata Machine: A New Environment for Modeling (The MIT Press, Cambridge 1988)

[27] T. Toffoli, N. Margolus, Programmable Matter: Concepts and Realization, Physica D 47 (1991) 263-272

[28] J. R. Weimar, Simulation with Cellular Automata (Logos Verlag, Berlin 1997)

[29] S. Wolfram, Computation Theory of Cellular Automata, Comm. Math. Ph., 96 (1984) 15-57

[30] S. Wolfram, Origins of Randomness in Physical Systems, Physical Review Letters, 55 (1985) 449-452

Table 1

			Without STIR-like			With STIR-like		
Dist.	Steps	Exp.	Occ.	V value	Res.	Occ.	V value	Res.
10%	10000	1000	1045	1.6044	S	1038	2.2500	S
	20000	2000	2003	0.0050	S	2049	1.3339	S
	30000	3000	2937	1.4700	S	3056	1.1615	S
	40000	4000	3957	0.5136	S	4041	0.4669	S
	50000	5000	4919	1.4580	S	5044	0.4302	S
25%	10000	2500	2587	4.0368	A	2492	0.0341	S
	20000	5000	5241	15.4883	F	5086	1.9723	S
	30000	7500	7756	11.6508	F	7515	0.0400	S
	40000	10000	10355	16.8033	F	9959	0.2241	S
	50000	12500	12895	16.6427	F	12427	0.5684	S
47.8%	10000	4780	4891	4.9380	A	4824	0.7759	S
	20000	9560	9711	4.5690	A	9512	0.4616	S
	30000	14340	14607	9.5236	F	14304	0.1731	S
	40000	19120	19418	8.8976	F	19058	0.3851	S
	50000	23900	24221	8.2593	F	23758	1.6162	S
50%	10000	5000	5177	12.5316	F	5060	1.4400	S
	20000	10000	10185	6.8450	F	10144	4.1472	A
	30000	15000	15253	8.5345	F	15170	3.8533	A
	40000	20000	20224	5.0176	A	19943	0.3249	S
	50000	25000	25333	8.8711	F	24990	0.0080	S
50%	10000	5000	5113	5.1076	A	4978	0.1936	S
	20000	10000	10096	1.8432	S	9985	0.0450	S
	30000	15000	15041	0.2241	S	14878	1.9845	S
	40000	20000	19923	0.5929	S	19834	2.7556	S
	50000	25000	24871	1.3313	S	24741	5.3664	A
75%	10000	7500	7140	69.1200	F	7519	0.1925	S
	20000	15000	14295	132.5400	F	14914	1.9723	S
	30000	22500	21536	165.2082	F	22374	2.8224	S
	40000	30000	28663	238.3425	F	29873	2.1505	S
	50000	37500	35834	296.0593	F	37421	0.6657	S
90%	10000	9000	8804	42.6844	F	8892	12.9600	F
	20000	18000	17652	67.2800	F	17866	9.9756	F
	30000	27000	26427	121.6033	F	26825	11.3426	F
	40000	36000	35160	196.0000	F	35799	11.2225	F
	50000	45000	43786	327.5102	F	44797	9.1576	F

Table 2

			Without STIR			With STIR		
Dist.	Steps	Exp.	Occ.	V value	Res.	Occ.	V value	Res.
10%	10000	1000	1045	1.6044	S	1016	0.2844	S
	20000	2000	2003	0.0050	S	2050	1.3889	S
	30000	3000	2937	1.4700	S	3071	1.8670	S
	40000	4000	3957	0.5136	S	4027	0.2025	S
	50000	5000	4919	1.4580	S	4956	0.4302	S
25%	10000	2500	2288	23.9701	F	2270	28.2133	F
	20000	5000	4580	47.0400	F	4506	65.0763	F
	30000	7500	6810	84.6400	F	6705	112.3600	F
	40000	10000	9102	107.5205	F	8951	146.7201	F
	50000	12500	11318	149.0266	F	11342	143.0362	F
25%	10000	2500	2587	4.0368	A	2544	1.0325	S
	20000	5000	5241	15.4883	F	4955	0.5400	S
	30000	7500	7756	11.6508	F	7513	0.0300	S
	40000	10000	10355	16.8033	F	10026	0.0901	S
	50000	12500	12895	16.6427	F	12512	0.0154	S
33%	10000	3300	3393	3.9118	A	3247	1.2704	S
	20000	6600	6830	11.9629	F	6505	2.0409	S
	30000	9900	10220	15.4380	F	9901	0.0001	F
	40000	13200	13564	14.9815	F	13277	0.6704	S
	50000	16500	16930	16.7255	F	16538	0.1306	S
33%	10000	3300	3294	0.0163	S	3342	0.7978	S
	20000	6600	6495	2.4932	S	6672	1.1723	S
	30000	9900	9909	0.0122	S	9980	0.9649	S
	40000	13200	13286	0.8364	S	13251	0.2941	S
	50000	16500	16694	3.4044	S	16511	0.0109	S
33%	10000	3300	3464	12.1646	F	3355	1.3682	S
	20000	6600	7000	36.1827	F	6793	8.4236	F
	30000	9900	10386	35.6092	F	10128	7.8372	F
	40000	13200	13879	52.1304	F	13433	6.1385	A
	50000	16500	17375	69.2600	F	16731	4.8268	A
40%	10000	4000	3929	2.1004	S	4079	2.6004	S
	20000	8000	8036	0.2700	S	8225	10.5469	F
	30000	12000	11976	0.0800	S	12216	6.4800	A
	40000	16000	16231	5.5584	A	16133	1.8426	S
	50000	20000	20290	7.0083	F	20191	3.0401	S

Table 2 (more)

			Without STIR			With STIR		
Dist.	Steps	Exp.	Occ.	V value	Res.	Occ.	V value	Res.
40%	10000	4000	4079	2.6604	S	3893	4.7704	A
	20000	8000	8116	2.8033	S	7941	0.7252	S
	30000	12000	12126	2.2050	S	11869	2.3835	S
	40000	16000	16129	1.7334	S	15841	2.6334	S
	50000	20000	20233	4.5241	A	19856	1.7280	S
40%	10000	4000	4065	1.7604	S	4024	0.2400	S
	20000	8000	8195	7.9219	F	8097	1.9602	S
	30000	12000	12280	10.8889	F	12104	1.5022	S
	40000	16000	16325	11.0026	F	16182	3.4504	S
	50000	20000	20418	14.5603	F	20206	3.5363	S
47.8%	10000	4780	4891	4.9380	A	4807	0.2921	S
	20000	9560	9711	4.5690	A	9592	0.2052	S
	30000	14340	14607	9.5236	F	14404	0.5472	S
	40000	19120	19418	8.8976	F	19114	0.0036	A
	50000	23900	24221	8.2593	F	23846	0.2337	S
50%	10000	5000	5177	12.5316	F	4943	1.2996	S
	20000	10000	10185	6.8450	F	9789	8.9042	F
	30000	15000	15253	8.5345	F	14754	8.0688	F
	40000	20000	20224	5.0176	A	19758	5.8564	A
	50000	25000	25333	8.8711	F	24716	6.4525	A
50%	10000	5000	5113	5.0176	A	5046	0.8464	S
	20000	10000	10096	1.8432	S	10005	0.0050	S
	30000	15000	15041	0.2241	S	14923	0.7905	S
	40000	20000	19923	0.5929	S	19807	3.7249	S
	50000	25000	24871	1.3313	S	24809	2.9185	S
50%	10000	5000	4969	0.3844	S	5026	0.2704	S
	20000	10000	9969	0.1922	S	10182	6.6248	A
	30000	15000	15071	0.6721	S	15175	4.0833	A
	40000	20000	20001	0.0001	F	20326	10.6276	F
	50000	25000	24955	0.1620	S	25236	4.4557	A
50%	10000	5000	4956	0.7744	S	5021	0.1764	S
	20000	10000	9951	0.4802	S	10030	0.1800	S
	30000	15000	14961	0.2028	S	15042	0.2352	S
	40000	20000	19992	0.0064	S	19983	0.0289	S
	50000	25000	24964	0.1037	S	24997	0.0007	A

Table 2 (more)

			Without STIR			With STIR		
Dist.	Steps	Exp.	Occ.	V value	Res.	Occ.	V value	Res.
50%	10000	5000	4826	12.1104	F	5138	7.6176	F
	20000	10000	9626	27.9752	F	10210	8.8200	F
	30000	15000	14365	53.7633	F	15158	3.3285	S
	40000	20000	19106	79.9236	F	20148	2.1904	S
	50000	25000	24061	70.5377	F	25161	2.0737	S
50%	10000	5000	5024	0.2304	S	5073	2.1316	S
	20000	10000	9996	0.0032	A	10077	1.1858	S
	30000	15000	14905	1.2033	S	15114	1.7328	S
	40000	20000	19811	3.5721	S	20218	4.7524	A
	50000	25000	24771	4.1953	A	25266	5.6605	A
50%	10000	5000	5618	152.7696	F	5110	4.8400	A
	20000	10000	11311	343.7442	F	10385	29.6450	F
	30000	15000	17108	592.4885	F	15610	49.6133	F
	40000	20000	22290	892.6400	F	20681	46.3761	F
	50000	25000	28836	1177.1917	F	25875	61.2500	F
75%	10000	7500	7278	26.2848	F	7403	5.0181	A
	20000	15000	14638	34.9451	F	14892	3.1104	S
	30000	22500	22061	34.2615	F	22309	6.4855	A
	40000	30000	29491	34.5441	F	29701	11.9201	F
	50000	37500	36899	38.5281	F	37225	8.0667	F
75%	10000	7500	10000	3333.3333	F	7545	1.0800	S
	20000	15000	20000	6666.6667	F	15131	4.5763	A
	30000	22500	30000	10000	F	22683	5.9536	A
	40000	30000	40000	13333.333	F	30217	6.2785	A
	50000	37500	50000	16666.667	F	37769	7.7185	F
75%	10000	7500	6911	185.0245	F	7497	0.0048	S
	20000	15000	13788	391.7184	F	14969	0.2563	S
	30000	22500	20735	553.8178	F	22471	0.1495	S
	40000	30000	27769	663.6481	F	29998	0.0005	A
	50000	37500	34798	778.7524	F	37457	0.1972	S
75%	10000	7500	7359	10.6032	F	7560	1.9200	S
	20000	15000	14741	17.8883	F	15067	1.1971	S
	30000	22500	22152	21.5296	F	22658	4.4380	A
	40000	30000	29469	37.5948	F	30236	7.4261	F
	50000	37500	36980	28.8427	F	37771	7.8337	F

Table 2 (more)

			Without STIR			With STIR		
Dist.	Steps	Exp.	Occ.	V value	Res.	Occ.	V value	Res.
75%	10000	7500	7713	24.1968	F	7528	0.4181	S
	20000	15000	15201	10.7736	F	15032	0.2731	S
	30000	22500	22737	9.9856	F	22527	0.1296	S
	40000	30000	30328	14.3445	F	30049	0.3201	S
	50000	37500	37942	20.8388	F	37617	1.4601	S
90%	10000	9000	8821	35.6011	F	9106	12.4844	F
	20000	18000	17564	105.6089	F	18034	0.6422	S
	30000	27000	26307	177.8700	F	27022	0.1793	S
	40000	36000	34968	295.8400	F	36004	0.0044	S
	50000	45000	43725	361.2500	F	44909	1.8402	S
90%	10000	9000	8863	20.8544	F	9129	18.4900	F
	20000	18000	17742	36.9800	F	18110	6.7222	F
	30000	27000	26623	52.6404	F	27083	2.5515	S
	40000	36000	35390	103.3611	F	36196	10.6711	F
	50000	45000	43852	292.8676	F	45267	15.8420	F
90%	10000	9000	8719	87.7344	F	8997	0.0100	S
	20000	18000	17361	226.8450	F	18003	0.0050	S
	30000	27000	26167	256.9959	F	26982	0.1200	S
	40000	36000	35054	248.5878	F	35965	0.3403	S
	50000	45000	43790	325.3556	F	45040	0.3556	S
90%	10000	9000	8775	56.2500	F	8956	2.1511	S
	20000	18000	17520	128.0000	F	17909	4.6006	A
	30000	27000	26354	154.5615	F	26862	7.0533	F
	40000	36000	35230	164.6944	F	35899	2.8336	S
	50000	45000	44149	160.9336	F	44987	0.0376	S
90%	10000	9000	8862	21.1600	F	8946	3.2400	S
	20000	18000	17691	53.0450	F	17880	8.0000	F
	30000	27000	26635	49.3426	F	26847	8.6700	F
	40000	36000	35456	82.2044	F	35833	7.7469	F
	50000	45000	44437	70.4376	F	44908	1.8809	S