Control of Autonomous Swarms Using Gibbs Sampling

John S. Baras and Xiaobo Tan

***Abstract*— A distributed control approach is proposed for self-organization of autonomous swarms. The swarm is modeled as a Markov random field (MRF) on a graph where the (mobile) nodes and their communication/sensing links constitute the vertices and the edges of the graph, respectively. The movement of nodes is governed by the Gibbs sampler. The Gibbs potentials, local in nature, are designed to reflect collective goals such as gathering, dispersion, and linear formation. The algorithm can be run completely in parallel, and hence it is robust and scalable. Simulation results are provided to illustrate the proposed method.**

I. Introduction

This paper deals with the self-organization of large networks of mobile nodes, including but not limited to mobile sensor networks, robotic swarms conducting wide-area search, and flocks of unmanned aerial vehicles (UAVs) performing reconnaissance. Given an initial configuration of nodes, one wishes to find a distributed maneuver strategy under which the nodes evolve into a new, desired configuration. One such example is in mobile sensor networks deployed by UAVs, where the initial locations of sensors are random and they need to move autonomously into some configuration suitable for the mission. Another example is mission switching, where the nodes need to reconfigure for performing a new task. The large number of nodes and the limited power for communication and signal processing as well as bandwidth limitations, prohibit centralized coordination in such networks and necessitate a distributed, scalable approach.

Inspired by the emergent behaviors demonstrated by swarms of bacteria, insects, and animals, control methods that yield desired collective behaviors based on simple local interactions have received great interest [1], [2], [3], [4], [5], [6], [7]. Potential functions [8], [9] have often been involved in these methods, where the motion of nodes is determined by the gradient flow. An essential problem with such approaches, however, is that the system dynamics may get trapped at local minima of the potential function.

A stochastic approach to the control of large-scale swarms is proposed in this paper based on the theory of Markov random fields (MRFs). An MRF is a mathematical generalization of the notion of a one-dimensional temporal Markov chain to a two (or higher) dimensional spatial lattice or graph, and it has been used extensively in statistical mechanics and quantum mechanics of interacting particles, as well as with great success in developing image processing algorithms, see [10], [11], [12] and the references therein. In this paper a swarm of mobile nodes is modeled as an MRF on a graph where the nodes and their communication/sensing links constitute the vertices and the edges of the graph, respectively. Global objectives, such as gathering at one place for intensive search, dispersion for maximum area coverage, or forming linear formations for blanket-scan, are reflected through the design of Gibbs potentials. The movement of each node is then governed by simulated annealing based on the Gibbs sampler. Since the Gibbs potentials consist of locally coupling terms, the computation requires only information about the neighboring nodes on the graph. Sequential sampling, where the nodes take turns to update their locations, is first described. Extension to parallel sampling, where each node computes its next move simultaneously, will also be discussed.

Comparing with the deterministic approaches, the approach based on the Gibbs sampling can yield desired configurations corresponding to global minima of the potential function, with only local interactions. The resulting distributed control law is robust. The computational load for each node does not increase as the number of nodes increases, hence it is particularly suitable for control of large scale mobile networks.

The paper is organized as follows. In Section II some background on MRFs and the Gibbs sampler is briefly reviewed. The algorithms for self-organization of swarms are presented in Section III. Simulation results for three example tasks are reported in Section IV. Section V concludes the paper.

II. Markov Random Fields and the Gibbs Sampler

Let S be a finite set of cardinality σ, with elements denoted by s and called *sites*, and let Λ be a finite set called the *the phase space*. A *random field* on S with phases in Λ is a collection $X = \{X_s\}_{s\in S}$ of random variables X_s with values in Λ. A *configuration* of the system is $x = \{x_s, s \in S\}$ where $x_s \in \Lambda$, $\forall s$. The product space Λ^σ is called the *configuration space*.

A *neighborhood system* on S is a family $\mathcal{N} = \{\mathcal{N}_s\}_{s\in S}$, where $\mathcal{N}_s \subset S$, and $\forall s \in S$, 1) $s \notin \mathcal{N}_s$, 2) $r \in \mathcal{N}_s$ if and only if $s \in \mathcal{N}_r$. $\mathcal{N}_s$ is called the *neighborhood* of site s. The random field X is called a *Markov random field* (MRF)

This research was supported by the Army Research Office under the ODDR&E MURI01 Program Grant No. DAAD19-01-1-0465 to the Center for Networked Communicating Control Systems (through Boston University), and under ARO Grant No. DAAD190210319.

J. S. Baras is with the the Institute for Systems Research and the Department of Electrical & Computer Engineering, University of Maryland, College Park, MD 20742, USA baras@isr.umd.edu

X. Tan is with the Institute for Systems Research, University of Maryland, College Park, MD 20742, USA xbtan@umd.edu

with respect to the neighborhood system $\mathcal{N}$ if, $\forall s \in S$,

$$P(X_s = x_s | X_r = x_r, r \neq s)$$
$$= P(X_s = x_s | X_r = x_r, r \in \mathcal{N}_s),$$

i.e., the conditional probabilities associated with the joint probability distribution of X are local in character and they obey the spatial Markovian relationship.

A random field X is a *Gibbs random field* if and only if its joint probability distribution is of the form:

$$P(X = x) = \frac{e^{-\frac{U(x)}{T}}}{Z}, \forall x \in \Lambda^\sigma,$$

where T is the temperature variable (widely used in simulated annealing algorithms), $U(x)$ is the potential (or energy) of the configuration x, and Z is the normalizing constant, called the *partition* function: $Z = \sum_{x \in \Lambda^\sigma} e^{-\frac{U(x)}{T}}$. One then considers the following very useful class of potential functions

$$U(x) = \sum_{s \in \Lambda} \Phi_s(x),$$

which is a sum of individual contributions from potentials Φ_s for each site. Furthermore one can incorporate the key idea of local interactions by decomposing $\Phi_s(x)$ into a sum of *clique* potentials $\Psi_c(x)$:

$$\Phi_s(x) = \sum_{c \in \mathcal{C}_s} \Psi_c(x),$$

where $\mathcal{C}_s$ is the collection of all cliques associated with site s. Formally, a clique is either a single site, or a set of sites in which every site is a neighbor of every other site.

The celebrated Hammersley-Clifford theorem establishes that: a Gibbs random field defined by a Gibbs distribution with the potential expressed in terms of clique potentials, leads to an MRF with a consistent set of conditional probabilities; and conversely, for any MRF with a consistent set of conditional probabilities, there exists an equivalent Gibbs distribution expressed in terms of local potentials on cliques.

The *Gibbs sampler* is a sampling algorithm that results in a Gibbs distribution Π as a limiting distribution of a Markov chain. Pick an enumeration scheme for $S = \{s_1, \cdots, s_\sigma\}$, called a *visiting scheme*. The Gibbs sampler (a basic version) works as follows:

- Step 1. Pick an initial configuration $x \in \Lambda^\sigma$;
- Step 2. Update the configuration by visiting each site s_i, $1 \leq i \leq \sigma$ in turn. When visiting s_i, hold fixed the values at s_j, $j \neq i$, and change x_{s_i} to $z \in \Lambda$ with probability $P(z)$ defined through the local characteristics:

$$P(z) = \frac{\Pi(z | r \in \mathcal{N}_{s_i})}{\sum_{z' \in \Lambda} \Pi(z' | r \in \mathcal{N}_{s_i})}.$$

- Step 3. Repeat Step 2.

Note that in the above procedure the evaluation of $P(z)$ is in general easy thanks to the local nature of potential functions. Step 2 essentially defines the transition probabilities for a random field-valued Markov chain $X(n)$. One sequential visit to all sites is called a *sweep*. It can be shown [10] that as the number of sweeps tends to infinity, the distribution of $X(n)$ approaches the Gibbs distribution Π.

As the temperature T in the Gibbs distribution approaches 0, the Gibbs distribution converges to a uniform distribution on the space of configurations achieving the minimum of $U(x)$. Simulated annealing using the Gibbs sampler, with an appropriate cooling schedule, yields configurations corresponding to the minimizer(s) of $U(x)$ [10]. In particular, by designing $U(x)$ so that its minimizer(s) correspond to desired configurations, one can (ultimately) achieve such configurations following successive site updates based on local rules.

III. Self-Organization of Swarms

A. Sequential Sampling

For ease of discussion, assume that nodes move on a two-dimensional plane (It will be clear from the description below that the extension to three-dimensional space is straightforward). Discretize the space into a lattice of square cells, and label each cell with its coordinates (i, j), where $1 \leq i \leq N_x$ and $1 \leq j \leq N_y$ for $N_x, N_y > 0$. At most one mobile node is allowed to stay in each cell at any time instant. Let the number of nodes be K and label them from 1 through K. Denote the location $x_k = (i_k, j_k)$ of node k by the coordinates of the cell where node k stays.

Each node k has a communication/sensing range R_s and a moving range $R_m < R_s$. Node k can detect (through sensing or communication) the locations of other nodes that lie within the circle with center x_k and radius R_s. These nodes are the *neighbors* of node k. The node can go at most R_m within one move. In general R_s and R_m can be node-dependent so that one can model a swarm consisting of nodes with different sensing and moving capabilities. Each node is also equipped with some computational capability to perform the calculation to be described next.

A natural neighborhood system on $\{1, 2, \cdots, K\}$ is induced by the aforementioned sensing constraint. This also leads to a (dynamic) graph where each mobile node is a vertex of the graph, and the neighborhood system prescribes the edges between the nodes. One can now define an MRF on this graph. Here each mobile node k represents a site, and the associated (dynamic) phase space Λ_k is the set of all locations not occupied by other nodes. Note the difference from Section II, where the phase space is invariant with respect to the site index and is static. An *admissible* configuration in the current setting is one in which each site takes a different location value.

The Gibbs potential $U(x) = \sum_k \Phi_k(x)$ is designed to reflect the global objective, where $x = \{x_k, 1 \leq k \leq K\}$ is the configuration of nodes. The local nature of $U(x)$ is guaranteed by further requiring that $\Phi_k(x)$ depends only on x_k and the values on its neighborhood $\mathcal{N}_k$, i.e., $\Phi_k(x) =$

$\Phi_k(y)$ whenever $x_k = y_k$, and $x_{k'} = y_{k'}, \forall k' \in \mathcal{N}_k$. Hence one can also write $\Phi_k(x)$ as $\hat{\Phi}_k(x_k, \{x_{k'} : k' \in \mathcal{N}_k\})$.

Simulated annealing with the Gibbs sampler can then be used for the evolution of the network. To be specific, the locations of nodes are updated *sequentially* with the following algorithm:

- Step 1. Pick a cooling schedule $T(\cdot)$ and the total number N of sweeps. Let the sweep number $n = 1$;
- Step 2. Conduct a sweep of location updates for node 1 through node K *sequentially*, where node k, $1 \leq k \leq K$, performs the following:
  - Determine the set L_k of candidate locations for the next move of node k:

    $$L_k \stackrel{\triangle}{=} \Lambda_k \cap \{(i,j) : \sqrt{(i-i_k)^2 + (j-j_k)^2} \leq R_m\};$$

  - For each $l \in L_k$, evaluate

    $$\Phi_k(x^l) \stackrel{\triangle}{=} \hat{\Phi}_k(x_k = l, \{x_{k'} : k' \in \mathcal{N}_k\}), \text{ and}$$
    $$P(x_k = l) = \frac{e^{-\frac{\Phi_k(x^l)}{T(n)}}}{\sum_{l' \in L_k} e^{-\frac{\Phi_k(x^{l'})}{T(n)}}}.$$

    Note that for the purpose of practical implementation, the current $\mathcal{N}_k$ has been used in the evaluation of local potential for each candidate move $l \in L_k$;
  - Update x_k by letting $x_k = l$ with probability $P(x_k = l)$, $l \in L_k$;
- Step 3. Let $n = n + 1$. If $n = N$, stop; otherwise go to Step 2.

B. Parallel Sampling

The sequential sampling scheme has two difficulties in implementation. First each node needs to know the total number K of nodes and its index to keep track of the token. This is challenging since K might be huge, and further more K might be changing as new nodes join in or current nodes fail. The second problem is that it takes a long time to finish one sweep. Both problems can be solved using parallel sampling.

The parallel scheme is identical to the sequential one except that all the nodes compute their next moves at the same time. This is possible, again due to the local nature of the Gibbs potential. The only issue is that two or more nodes may try to take the same spot for the next move. To resolve such conflicts, another sampling is taken with a uniform probability distribution on contending nodes. The "winner" takes its desired spot, while the "losers" stay put at their current locations.

IV. Simulation Results

In this section examples of Gibbs potentials are presented for three different missions together with the simulation results.

A. Gathering

The first example is that the nodes are required to get close to each other. This happens, for instance, when the nodes need to perform intensive search in a small area.

The Gibbs potential function for this task is picked to be: for $k \in \{1, \cdots, K\}$,

$$\begin{aligned} & \hat{\Phi}_k(x_k, \{x_{k'} : k' \in \mathcal{N}_k\}) \\ = \; & \lambda_1 \|x_k - z_0\| + \begin{cases} \frac{\lambda_2}{\sum_{k' \in \mathcal{N}_k} \frac{1}{\|x_k - x_{k'}\|}} \text{ if } \mathcal{N}_k \neq \emptyset \\ \Delta \text{ if } \mathcal{N}_k = \emptyset \end{cases} \end{aligned} \quad (1)$$

where z_0 is the reference location for gathering, and $\lambda_1 \geq 0$, $\lambda_2 > 0$ are scaling constants. The first term in (1) attracts the nodes close to z_0. The second term tends to cluster the nodes; in particular, it is smaller when node k has more, closer neighbors. $\Delta > 0$ is a relatively large constant and it represents the penalty for having no neighbors.

Fig. 1 shows the self-organization of 200 nodes on a 50×50 grid, where the goal is to gather around the cell (25,25). Sequential sampling is used with $\lambda_1 = 0.05$, $\lambda_2 = 1.0$, $\Delta = 10^3$, $R_m = 2\sqrt{2}$, $R_s = 6\sqrt{2}$, and a cooling schedule $T(n) = \frac{1}{4\log(400+n)}$. From the figure, the group completes the task within 30 sweeps.

Fig. 2 shows how the group evolves when $\lambda_1 = 0$. In this case there is no fixed attraction point to pull the nodes together. An interesting observation is that the nodes first organize locally into several clusters; finally these clusters merge into one big cluster. This demonstrates one of the strengths of the algorithm: it can get out of the local minimum of the potential function due to its stochastic nature.

B. Dispersion

The second example of mission is dispersion, where the nodes are desired to stay away from each other. This is useful, e.g., when the nodes deployed initially in a small area need to scatter around for wide area coverage. The following potential function is chosen for this purpose:

$$\begin{aligned} & \hat{\Phi}_k(x_k, \{x_{k'} : k' \in \mathcal{N}_k\}) \\ = \; & \begin{cases} \frac{\lambda}{\min_{k' \in \mathcal{N}_k} \|x_k - x_{k'}\|} \text{ if } \mathcal{N}_k \neq \emptyset \\ \epsilon \text{ if } \mathcal{N}_k = \emptyset \end{cases}, \end{aligned} \quad (2)$$

where λ is a scaling constant, and ϵ is a very small number (having no neighbors is now encouraged).

Fig. 3 shows the simulation results of dispersing 200 nodes. The parameters used are $\lambda = 0.6$, $\epsilon = 10^{-8}$, R_s, R_m, and the cooling schedule are as same as those in Fig. 1. One can see that the nodes move and spread themselves over the whole region. In Fig. 3(d), the minimum inter-node distance is 2.

C. Formation in Lines

Formation of nodes is important for various applications. For example, one usage of linear formations is blanket-search of the whole area. Here how to form linear formations is explored. In particular, an example will be given

below for linear formations with specified orientations. With a minor modification, the potential function can be used for self-organization into lines without directional constraints.

Assume that the formation of lines parallel to the $y-$axis is needed (other directional specifications can be accommodated similarly). For $k' \in \mathcal{N}_k$, let $d_{k,k'}$ be the distance between node k and k', $\theta_{k,k'}$ be the angle formed by the line segment from node k' to node k with the $x-$axis, and m_k be the number of elements in $\mathcal{N}_k$. The potential function $\hat{\Phi}_k$ is then expressed as:

$$\hat{\Phi}_k(x_k, \{x'_k : k' \in \mathcal{N}_k\}) = \begin{cases} \frac{\lambda}{m_k} \sum_{k' \in \mathcal{N}_k} \frac{d_{k,k'}}{R_s}(1 - |\sin(\theta_{k,k'})|)^2 \text{ if } m_k > 0 \\ \Delta \text{ if } m_k = 0 \end{cases} \quad (3)$$

where $\lambda > 0$ is a scaling constant, $\Delta > 0$ is a relatively large number penalizing node k on having no neighbors. The term $\frac{d_{k,k'}}{R_s}$ puts more weight on farther neighbors, which encourages the formation of long lines.

Fig. 4 through Fig. 6 show the line forming processes for different R_s, where sequential sampling is used. Other parameters are: $\lambda = 10$, $\Delta = 5$, the cooling schedule $T(n) = \frac{1}{4log(400+n)}$. When $R_s = 10\sqrt{2}$, the nodes form one single line (Fig. 4); when $R_s = 6\sqrt{2}$, two lines are formed (Fig. 5); when $R_s = 4\sqrt{2}$, the nodes evolve into three lines (Fig. 6). The explanation is that longer sensing range enables more nodes to interact and collaborate. This demonstrates that the sensing range R_s has strong impact on the global behavior, at least in some missions. Note that this dependence could be exploited to provide more freedom in self-organization, e.g., missions as concrete as "to form m lines" or "to have line-to-line distance q meters" may be accomplished by adjusting the R_s parameter in sampling.

So far the simulation results reported are all obtained using the sequential sampling. Fig. 7 shows the results of line-forming based on the parallel sampling. It should be noted that the parallel scheme delivers comparable results as the sequential one. In particular, the numbers of lines formed under the parallel scheme for different R_s are consistent with that in Fig. 4 through Fig. 6.

V. Conclusions and Discussions

A distributed control scheme has been proposed for self-organization of large networks of mobile nodes, where the key idea is simulated annealing based on the Gibbs sampler. Desired global behaviors can be encoded into the Gibbs potential function characterized by local interactions. For illustrative purposes, three examples of potential function design have been presented. Simulation results have shown that local movement of the nodes based on local information leads to emergent behaviors.

The proposed approach is scalable, which is crucial for autonomous swarms. The computational requirement for each node remains the same as the number of nodes increases. Furthermore, the parallel sampling scheme allows the computation to be carried out simultaneously at each node, which makes the self-organization process very efficient.

Another advantage is the robustness. Indeed from the simulation results, what can be expected is the global behavior of the nodes and not the configuration of any individual node. This eliminates the dependence on any single node and offers the robustness on aggregate performance through redundancy of nodes.

This scheme can easily accommodate various constraints. For instance, in the context of this paper the nodes can avoid collision with each other automatically. Other considerations, such as obstacle avoidance and threat evasion, can either be included similarly or encoded in the Gibbs potential function. Directional constraints or threats can also be modeled appropriately. In addition to the sensing range, the effect of sensing noise can be readily reflected through the final annealing temperature T_f: the lower T_f, the smaller the noise.

General theory of MRF provides conditions for convergence to the global minimizers of the Gibbs potential. Note, however, that the moving range constraint R_m poses difficulty in directly applying classical analysis results. In particular, due to this constraint, each node can only access a small subset of its phase space during the location update, and the sampling algorithm may not lead to the minimizing configurations of the potential function. Fortunately, in some cases, additional structures of the problems prove useful toward establishing the convergence to the global optimizer; and in some other cases, the configurations corresponding to local minima of the potential suffice for the mission purpose. Classification of these cases and detailed analysis of them are the subjects of future work.

References

[1] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[2] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, FL, 2001, pp. 2968–2973.

[3] H. V. Parunak, M. Purcell, and R. O'Connel, "Digital pheromones for autonomous coordination of swarming UAV's," in *Proceedings of AIAA 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles, Systems, and Operations*, 2002.

[4] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," in *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain, 2002.

[5] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[6] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents, Part I: fixed topology," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003, pp. 2010–2015.

[7] J. S. Baras and X. Tan, "Decentralized control of autonomous vehicles," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003, pp. 1532–1537.

[8] R. Shahidi, M. Shayman, and P. S. Krishnaprasad, "Mobile robot navigation using potential functions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp. 2047–2053.

[9] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.
[10] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.
[11] M. Beckerman, *Adaptive Cooperative Systems*, John Wiley & Sons, 1997.
[12] P. Bremaud, *Markov Chains, Gibbs Fields, Monto Carlo Simulation and Queues*, Springer Verlag, New York, 1999.

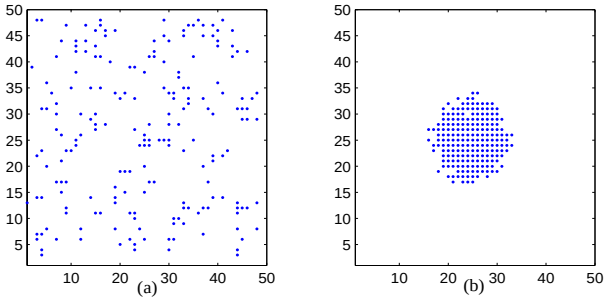

Fig. 1. Gathering around a specified location (sequential sampling). (a) Initial configuration; (b) configuration after 30 sweeps.

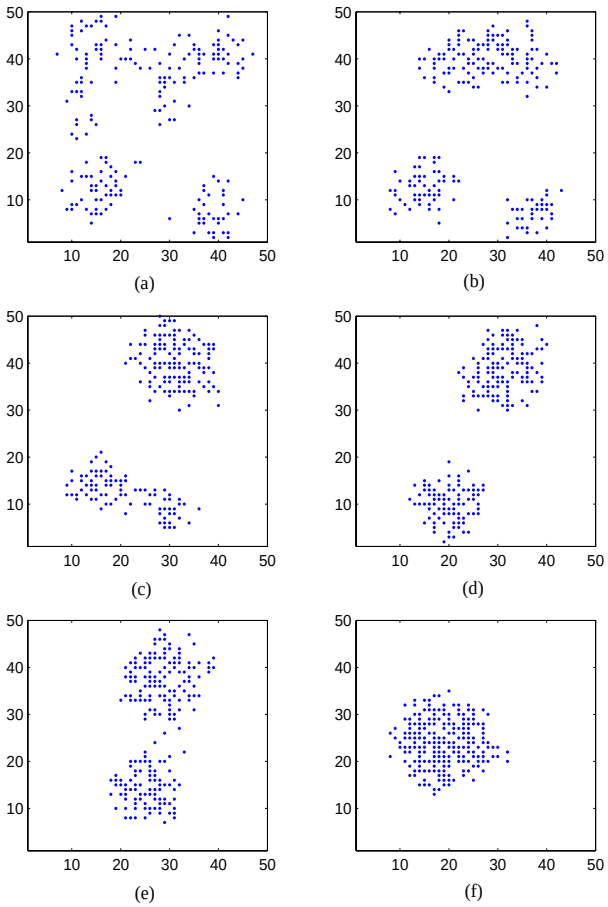

Fig. 2. Gathering without location specified (sequential sampling). Initial configuration as in Fig. 1 (a). Configurations after (a) 20 sweeps; (b) 50 sweeps; (c) 230 sweeps; (d) 300 sweeps; (e) 490 sweeps; (f) 2000 sweeps.

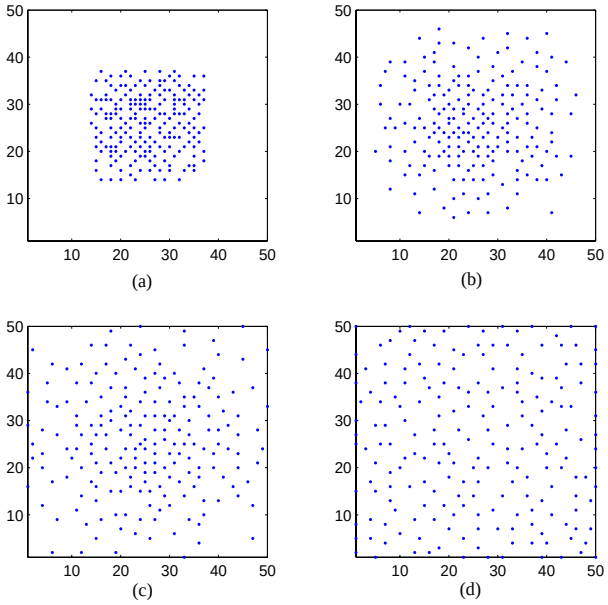

Fig. 3. Dispersion (sequential sampling). (a) Initial configuration; (b) after 5 sweeps; (c) after 10 sweeps; (d) after 70 sweeps.

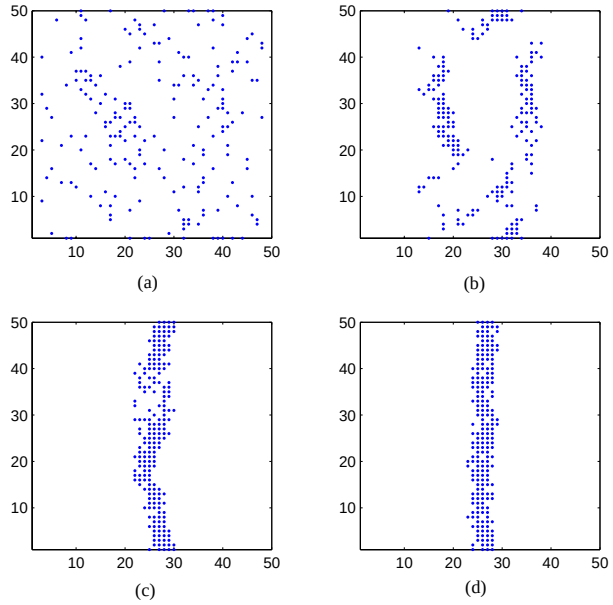

Fig. 4. Forming lines parallel to the $y-$ axis: $R_s = 10\sqrt{2}$ (sequential sampling). (a) Initial configuration; (b) after 5 sweeps; (c) after 10 sweeps; (d) after 70 sweeps.

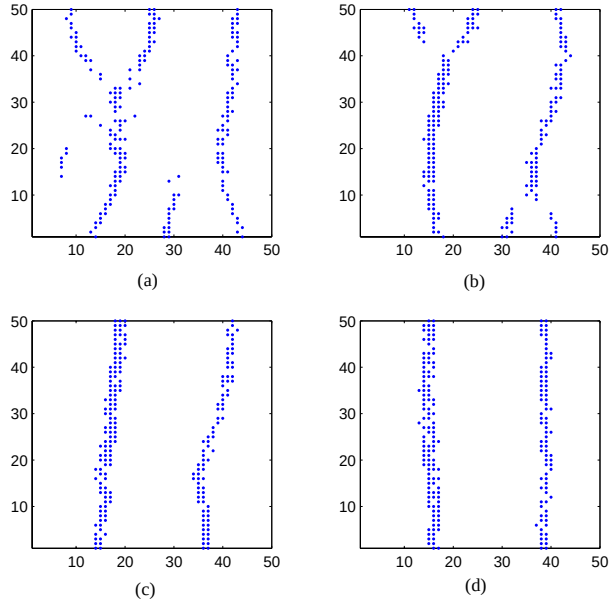

Fig. 5. Forming lines parallel to the $y-$ axis: $R_s = 6\sqrt{2}$ (sequential sampling). Initial configuration as in Fig. 4(a). (a) After 5 sweeps; (b) after 10 sweeps; (c) after 20 sweeps; (d) after 200 sweeps.

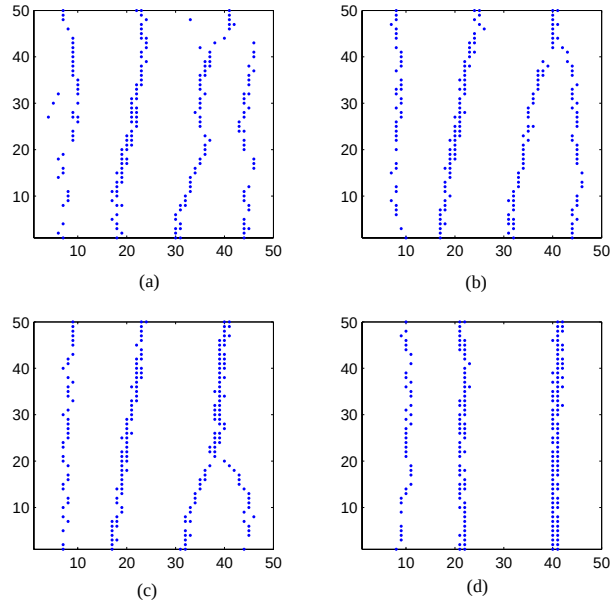

Fig. 6. Forming lines parallel to the $y-$ axis: $R_s = 4\sqrt{2}$ (sequential sampling). Initial configuration as in Fig. 4(a). (a) After 5 sweeps; (b) after 10 sweeps; (c) after 20 sweeps; (d) after 350 sweeps.

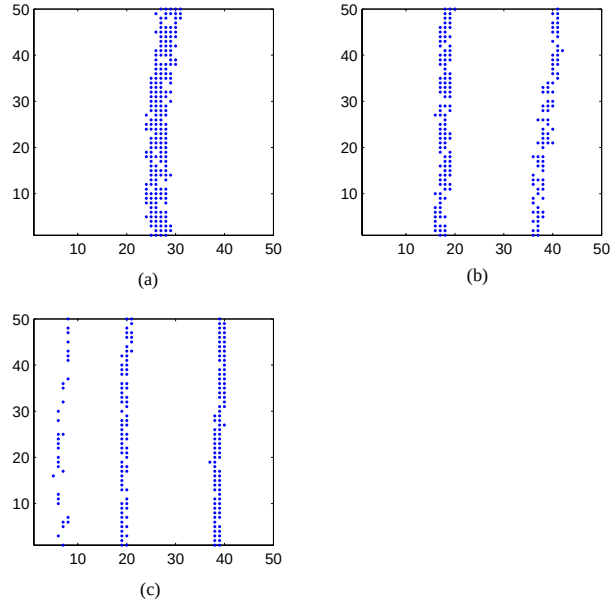

Fig. 7. Forming lines using *parallel sampling*. Initial configuration as in Fig. 4(a). (a) $R_s = 10\sqrt{2}$, after 70 sweeps; (b) $R_s = 6\sqrt{2}$, after 200 sweeps; (c) $R_s = 4\sqrt{2}$, after 350 sweeps.