A COMBINATORIAL MODEL FOR SELF-ORGANIZING NETWORKS

Yuri Dimitrov, (e-mail: yuri@math.ohio-state.edu)
Mario Lauria, (e-mail: lauria@cse.ohio-state.edu)

The Ohio State University,
Columbus, Ohio 43210, USA

1 Introduction

The organization of many complex biological and social systems has been explained in terms of the aggregations of a large number of autonomous entities that behave according to simple rules. According to this theory, complicated patterns can emerge from the interplay of many agents — despite the simplicity of the rules [7, 5]. The existence of this mechanism, often referred to as *emergence*, has been proposed to explain patterns such as shell motifs, animal coats, neural structures, and social behavior. In particular, complex behaviors of colonial organisms such as social insects (i.e. ants, bees) have been studied in detail, and their applications to the solution of classic computer science problems such as task scheduling and TSP has been proposed [6, 1].

We have developed a framework called the Organic Grid [3, 2, 4] for deploying and scheduling computation on desktop grids in a decentralized and self-organizing manner.

Our design is based on the following design assumptions. First, very few assumptions (if any) can be made about the systems, in particular about the amount of knowledge available about the system. Second, since the system is constantly changing (in terms of operating parameters, resource availability), self-adaption is the normal mode of operation and must be built in from the start. Third, the deployment of the components of an infrastructure is a non-trivial issue, and should be one of the fundamental aspects of the design. Fourth, any dependence on specialized entities such as schedulers, masters nodes, etc., needs to be avoided unless such entities can be easily replicated in a way that scales with the size of the system.

In our design we tried to address all these points simultaneously with a unified design methodology based on the principle of emergence. Nature provides numerous examples of the emergence of complex patterns derived from the interactions of millions of organisms that organize themselves in an autonomous, adaptive way by following relatively simple behavioral rules. In order to apply this concept to the organization of computation over large complex systems, a computation must be broken into small self-contained chunks, each capable of expressing autonomous behavior in its interaction with other chunks.

Our approach was to encapsulate computation and behavior into mobile agents, which deliver the computation to available machines. These mobile agents then communicate with one another and organize themselves in order to use the resources effectively. Once an application is started at a node, e.g., the user's laptop, other nodes are called in to contribute resources. New mobile agents are created that, under their autonomous control, readily colonizes the available resources and start computing. Only minimal support software is required on each node, since most of the scheduling infrastructure is encapsulated along with the application code inside an agent. In our experiments we only deployed a JVM and a mobile agent environment on each node.

Computation organizes itself on the available nodes according to a pattern that emerges from agent-agent interaction. In the simplest case, this pattern is an overlay tree rooted at the starting node; in the case of a data intensive application, the tree can be rooted at one or more separate, presumably well-connected machines at a supercomputer center. More complex patterns can be developed as required by the applications requirements, either by using different topologies than the tree, and/or by having multiple overlay networks each specialized for a different task.

In our system, the only knowledge each agent relies upon is what it can derive from its interaction with its neighbor and with the environment, plus an initial *friends list* needed to bootstrap the system. The nature of the information required for successful operation is application dependent and can be customized. E.g., for our first (data-intensive) application, both neighbor computing rate and communication bandwidth of the intervening link were important; this information was obtained using feedback from the ongoing computation.

Agent behavior completely determines the way computation is organized. In order to demonstrate the feasibility and generality of this approach, we built a prototype Organic Grid. In the course of several experiments we designed and tested agent behavior specific for two representative applications: the NCBI BLAST code for sequence alignment, and Cannon's algorithm for matrix multiplication. The results of these experiments are reported elsewhere [3, 2, 4].

The Organic Grid experiments taught us that it was crucial for our prototypes to be able to dynamically reorganize themselves. The two most important reasons for self-organization in massive systems are i) tolerance to faults, and ii) independence from the initial conditions. One of the challenges we faced in implementing a distributed reorganization scheme was how to study its behavior for arbitrarily large and complex topologies. The limited size of our experiments allowed us to adopt empirical design rules and a trial-and-error approach to testing, but obviously more powerful methods of synthesis and analysis are required for real world systems. As a consequence, we have embarked on a project to provide some mathematical foundations to the field of self-organizing computation based on emergent design.

In this document we report one of our efforts on developing methods for correlating small scale agent behavior with large scale system patterns of organization.

Specifically, during our Organic Grid experiments we asked ourselves if it were possible to forecast the distribution of node performance at every level of the tree, given i) the rule of tree reorganization built into the agent behavior and given ii) an initial distribution of node performance. We have approached this problem by performing a combinatorial analysis of possible tree configurations; given the complexity of this type of analysis, we have started considering very regular tree topologies (complete symmetric binary trees; only two levels of node performance, low ("0") and high ("1")) and a simplified parent-child exchange rule. In other words, we have been able to solve the following problem:

The binary tree with $N = 2^d - 1$ nodes has a number 0 or 1 assigned at random to each node. An edge is chosen at random and the numbers at the nodes for this edge are switched if the lower node has value 1 and the higher node has value 0. The values are switched with probability p if the lower node has value 0 and the higher has value 1.

Find the steady state distribution of the 1s at all d levels of the tree.

In this report we describe the solution to this problem and the approach we followed. Even with these simplifications, the number of configurations with n nodes and k "ones" is $\binom{n}{k}$. We have been able to tackle this complexity by exhaustively studying small trees of increasing depth, and then deriving a general formula through induction. The formula has been validated by comparing its value with those obtained through simulation performed on Matlab.

This approach is reminiscent of the analysis of thermodynamic ensembles using Boltzmann statistics, and in fact our final closed formula is in the form of a Boltzmann sum if the probability p is replaced with the exponential function of the energy (in our case a configuration of the tree represent a microstate, and a microstate is the subset of configurations with the same number of "ones" in a level.)

It is our hope that in the future we will be able to generalize these initial findings by studying more complex topologies and with increasing levels of node performance, and validating the results with those obtained through simulation.

2 Configurations

Let's denote by T_d the balanced tree with d levels and $N = 2^d - 1$ nodes. First we assign to each node a number 0 or 1 at random. The nodes with number 1 have more computational power than the nodes with number 0. The number of nodes which have number 1 is a random variable which has binomial distribution with probability for success 0.5 and takes values $0, 1, ..., 2^d - 1$. Let n be the number of nodes which have number 1. The total number of nodes is N and there are $\binom{N}{n}$ possible states of the system. We call each state "configuration". Let $G_d(n)$ for $n = 1, 2, ..., 2^d - 1$ be the graphs of configurations of T_d with n ones. The graph $G_d(n)$ has $\binom{N}{n}$ vertices corresponding to the configurations with n ones. Two configurations C_1 and C_2 are adjacent if they are consecutive states of the system. Then there exists an edge $e = \{v_1, v_2\}$ of T_d where the vertex v_1 has number 0 in C_1 and v_1 is 1 in C_2. The vertex v_2 has number 1 in C_1 and v_2 is 0 in C_2. The

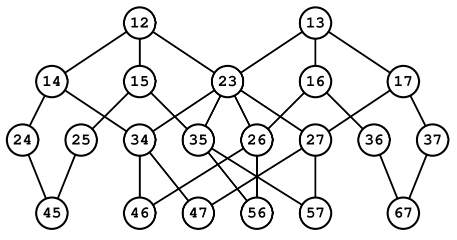

Figure 1: The graph $G_3(2)$.

numbers of all other vertices is the same in both configurations C_1 and C_2. The graph $G_d(1)$ is the same as the initial graph T_d and the graph $G_d(2^d-1)$ consists of a single vertex. The graph $G_3(2)$ is given in Figure 1. The vertices of $G_3(2)$ correspond to the configurations of T_3 with two ones. The top level has two configurations C_1 and C_2 where C_1 has ones at vertices 1 and 2 and C_2 has ones at vertices 1 and 3. We denote the levels of T_d with $0, 1, ..., d-1$ and enumerate the nodes of T_d with numbers $1, 2, ..., 2^d-1$. The root has a number 1 and the nodes on row i have numbers $2^i, ..., 2^{i+1}-1$ from left to right.

Definition 2.1. *The maximal configuration with n ones is the configuration where the ones are in the nodes $1, 2, ..., n$. The minimal configuration with n ones is the configuration where the ones are in the nodes $2^d-n, 2^d-n+1, ..., 2^d-1$.*

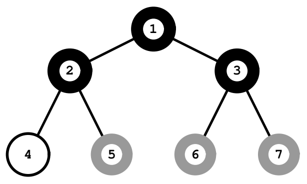

Figure 2: The maximal and minimal configurations with three $1s$ in T_3

The maximal and minimal configurations of T_d with three ones is given in Figure 2. Let's denote by $(i_1, i_2, ..., i_n)$ the configuration in T_d where nodes $i_1, i_2, ..., i_n$ have number 1.

Claim 2.1. *The graphs $G_d(n)$ are connected for all $d \geq 2$ and $n = 1, ..., 2^d-2$.*

Proof. We use Induction on the number of levels d. When $d=2$ the graph $G_2(1)$ has three vertices corresponding to the configurations $(1), (2)$ and (3) and two edges $\{(1),(2)\}$ and $\{(1),(3)\}$. Therefore $G_2(1)$ is connected. The graph $G_2(2)$ also has three vertices $(1,2), (1,3), (2,3)$ and two edges $\{(1,2),(2,3)\}$ and $\{(1,3),(2,3)\}$. Therefore $G_2(2)$ is connected as well. Suppose that the graphs $G_d(n)$ are connected for all $n = 1, ..., 2^d-2$. Let C_1 and C_2 be two configurations with n nodes with number 1 in T_{d+1}. Let's denote by T'_d and T''_d the subtrees of T_{d+1} with d levels and roots at nodes 2 and 3 respectively. Let k_1 and l_1 be the number of nodes of C_1 and C_2 with number 1 in the subtree T'_d and k_2 and l_2 be the number of nodes of C_1 and C_2 with number 1 in T''_d. Then $k_1+k_2=n$ and $l_1+l_2=n$ when C_1 and C_2 do not contain the root. First we define configurations K_1 and K_2 which have n ones and don't contain the root in the following way. If C_1 doesn't contain the root then $K_1=C_1$. Suppose that C_1 contains the root. Let C'_1 be the configuration with n nodes which contains the root and is a minimal configuration with k_1 nodes in T'_d and minimal configuration with k_2 nodes in T''_d. Then there is a path between C_1 and C'_1, because the graphs of configurations of T'_d and T''_d are connected. At least one of the nodes 2 or 3 doesn't belong to C'_1 because $n \leq 2^d-2$. Suppose that node 2 is not in C'_1. Then define K_1 to be the configuration obtained from C'_1 by removing the root and adding node 2. Then configuration K_1 doesn't contain the root and C_1 and K_1 are connected. We define configuration K_2 from C_2 in a similar way. Now we want to show that K_1 and K_2 are connected. Let K'_1 be the configuration with n ones which consists of a maximal configuration with k_1 nodes in T'_d and a minimal configuration with k_2 nodes in T''_d. There exists a path between K_1 and K'_1 because by the inductive assumption the graphs of configurations of the subtrees T'_d and T''_d are connected. Suppose that $k_1 > l_1$. Let us apply k_1-l_1 times the following algorithm. Take the smallest node of T'_d with number 1 and move it to the largest node in T''_d with number 0. This is possible because the nodes on the path between these two nodes consists of nodes with number 0. Let's denote by K_2' the configuration that we obtain after applying this algorithm. The configuration K_2' consists of a minimal configuration with l_2 nodes in T_d'' and K_2' has l_1 nodes in T_d'. Then there exists a path between K_2' and K_2. Therefore there exists a path between C_1 and C_2 in $G_{d+1}(n)$. □

Now we give a recursive definition for the levels of the graphs $G_d(n)$. Let C_M be the maximal configuration with n ones. We assign level 0 to the maximal configuration: $l(C_M) = 0$. Let L be a list of configurations which contains initially only the maximal configuration C_M. We apply repeatedly the following algorithm on the graph of configurations $G_d(n)$.

Algorithm. *Choose a configuration C which is not in L and there is a configuration $C' \in L$ adjacent to C. Let $e = \{v_1, v_2\}$, where $l(v_1) < l(v_2)$ be the edge of T_d such that the numbers of C and C' at the vertices v_1 and v_2 are different. If the number of v_1 in configuration C is 1 we assign $l(C) = l(C') - 1$. Otherwise we assign $l(C) = l(C') + 1$. Then we add configuration C to L.*

By Claim 1.1 the graph $G_d(n)$ is connected and the algorithm assigns a level to each node. In Section 3 we assign weights to the nodes of T_d and the configurations of $G_d(n)$. The configurations on the levels of $G_d(n)$ have equal weight. Note that if two configurations with n ones are adjacent then their dual configurations with $2^d - n - 1$ ones are also adjacent. The graphs $G_d(n)$ and $G_d(2^d - n - 1)$ are dual. They are the same graph with their levels reversed.

Lemma 2.1. *Let C_k and C_{k-1} be two adjacent configurations in $G_n(d)$ such that $l(C_k) = k$ and $l(C_{k-1}) = k - 1$. Then the probability of C_k is p times smaller than the probability of C_{k-1}.*

Proof. Let $\{v_1, v_2\}$, be the edge of T_d such that the numbers of C_k and C_{k-1} at the vertices v_1 and v_2 are different and $p(C_k)$ and $p(C_{k-1})$ be the probabilities that the network is in configurations C_k and C_{k-1}. The probability to choose edge $\{v_1, v_2\}$ at each step of the process is $q = \dfrac{1}{2^d - 1}$. Let's consider the probability for the transition from configuration C_k to C_{k-1}. Suppose that the network is in configuration C_k. Then it will move to C_{k-1} only when edge $\{v_1, v_2\}$ is chosen. Therefore the probability for the transition from C_k to C_{k-1} is $qp(C_k)$. Suppose that the network is in configuration C_{k-1} and consider the probability that the network was in configuration C_k on the previous step. This is possible only if edge $\{v_1, v_2\}$ is chosen. The the probability for the transition is $pqp(C_{k-1})$. Therefore $p(C_k) = pp(C_{k-1})$. □

Corollary 2.1. *Let C and C' be two configurations in $G_d(n)$ with $l(C) = l(C')$. Then $p(C) = p(C')$.*

Proof. By Claim 1.1 there exists a path $\{C, C_1, ..., C_{2n-1}, C'\}$ in $G_d(n)$ between configurations C and C'. The levels of the configurations on the path increase n times and decrease n times because $l(C) = l(C')$. By Lemma 1.1 every time the level increases the probability for the configuration decreases by p and when the level decreases the probability of the configuration on the path increases by a factor of $\dfrac{1}{p}$. Therefore $p(C) = p(C')$. □

We want to determine the steady-state distribution of $0s$ and $1s$ on each level of T_d. Let $prob_{d,n,r,w}(p)$ be the probability to have w ones on row r in T_d from the configurations with n ones . Then

$$prob_{d,n,r,w}(p) = \frac{num_{d,n,r,w}(p)}{den_{d,n}(p)}$$

where $num_{d,n,r,w}(p)$ and $den_{d,n}(p)$ are polynomials of p. Let D be the number of levels of the graph $G_d(n)$. The polynomial $den_{d,n}(p)$ has degree $D-1$ and the coefficients the number of elements on each level of $G_d(n)$. The coefficients of $num_{d,n,r,w}(p)$ are determined by the number of configurations on the levels of $G_d(n)$ which have w ones on row r. We want to find formulas for $num_{d,n,r,w}(p)$ and $den_{d,n}(p)$. In section 3 we show that $num_{d,n,r,w}(p)$ and $den_{d,n}(p)$ are coefficients of the polynomials $q(x,y)$ and $q_{r,w}(x,y)$ defined by formulas 3.4 and 3.5. In section 4 we find recursive formulas to compute these probabilities. The probability to have w points on row r is

$$prob_{d,r,w}(p) = \sum_{n=0}^{2^d-1} \frac{\binom{2^d-1}{n} prob_{d,n,r,w}(p)}{2^{2^d-1}} = \sum_{n=0}^{2^d-1} \frac{\binom{2^d-1}{n} num_{d,n,r,w}(p)}{2^{2^d-1} den_{d,n}(p)}$$

3 Polynomial Solutions

In section 2 we described the states of the system as configurations of n nodes where n is the number of the nodes which have number one. We showed that the configurations on each level of the graph $G_n(d)$ have equal probability. The graph T_d has d levels $0, 1, ..., d-1$ where the root is on level 0. Let's now assign weights $1, 2, ..., d$ on the nodes of T_d so that the nodes on level i have weight $d-i$. Let $M(d,x)$ be the sum of the weights of the nodes of the first x levels of T_d.

$$M(d,x) = \sum_{k=0}^{x-1} 2^k (d-k) = 2^x(d-x+2) - d - 2$$

When $x = d$ we obtain that the sum of the weights of all nodes of T_d is

$$M(d) = 2^{d+1} - d - 2$$

The weight of a configuration is the sum of its nodes. In this way the configurations on each level of $G_d(n)$ have equal weight. The number of configurations on each level of $G_d(n)$ is given by the solutions of the following problem.

Problem 1. *Let $x_i = 0$ or $x_i = 1$ for $i = 1, 2, ..., 2^d - 1$ and $r_t = \sum_{i=2^t}^{2^{t+1}-1} x_i$ for $t = 0, 1, ..., d-1$. Let $s(d,n,k)$ be the number of sequences x_i such that*

$$\sum_{i=0}^{d-1} r_i = n \quad \text{and} \quad \sum_{i=0}^{d-1} (d-i) r_i = k \tag{3.1}$$

Determine the numbers $s(d,n,k)$.

Lemma 3.1.

$$s(d,n,k) = s(d, 2^d - n - 1, M(d) - k)$$

Proof. Let C be a configuration with n ones and weight k. Then the dual configuration of C has $2^d - n - 1$ ones and weight $M(d) - k$. $\square$

For each sequence of numbers r_i which satisfy 3.1 there are $\prod_{i=0}^{d-1} \binom{2^i}{r_i}$ solutions of Problem 1 because r_i is the number of nodes which have number one on row i and row i has 2^i nodes. Now we show that the numbers $s(d, n, k)$ are coefficients of the polynomials $p_d(x, y)$ defined bellow.

$$p_d(x, y) = \prod_{i=0}^{d-1} (1 + xy^{d-i})^{2^i}$$

Claim 3.1. *The coefficient of $p_d(x, y)$ of the term $x^n y^k$ is $s(d, n, k)$.*

Proof. We have that

$$p_d(x, y) = \prod_{i=0}^{d-1} \sum_{r=0}^{2^i} \binom{2^i}{r} x^r y^{r(d-i)}$$

The coefficient of $x^n y^k$ is $\prod_{i=0}^{d-1} \binom{2^i}{r_i}$ where $\sum_{i=1}^{d} r_i = n$ and $\sum_{i=1}^{d} r_i(d - i) = k$. Therefore the coefficient of $x^n y^k$ is $s(d, n, k)$. $\square$

For given values of d and n let $k(d, n)$ and $K(d, n)$ be the minimal and the maximal values of k for which $s(d, n, k)$ is not equal to 0. Let l be the largest integer such that $2^l - 1 < n$. The maximal value of k is attained at the maximal configuration. Then $2^l - 1$ of the nodes occupy the first l levels of the tree and the remaining $n - 2^l + 1$ nodes are on level l. Then

$$K(d, n) = M(d, l) + (d - l)(n - 2^l + 1)$$

$$K(d, n) = 2(2^l - 1) + dn - l(1 + n)$$

The minimum value of k is attained at the minimal configuration with n ones. The dual configuration of the minimal configuration with n ones is the maximal configuration with $2^d - n - 1$ ones. Then

$$k(d, n) = M(d) - K(d, 2^d - n - 1)$$

$$k(d, n) = 2(2^d - 2^l) + d(n - 2^d) + l(2^d - n)$$

In Claim 3.3 we show that the solutions of Problem 1 are the coefficients of $den_{d,n}(x)$. In Claim 3.4 we derive a formula for $num_{d,n,r,w}(x)$ using the numbers $s(d, n, t, k)$ defined in Problem 2.

Problem 2. *Let $x_i = 0$ or $x_i = 1$ for $i = 1, 2, ..., 2^d - 1$ and $r_t = \sum_{i=2^t}^{2^{t+1}-1} x_i$ for $t = 0, 1, ..., d - 1$. Let $s(d, t, n, k)$ be the number of sequences x_i which satisfy (3.1) and $r_t = 0$. Determine the numbers $s(d, t, n, k)$.*

Lemma 3.2.

$$s(d,t,n,k) = s(d,t,2^d - 2^t - n - 1, M(d) - 2^t(d-t) - k)$$

Proof. If C is a configuration of $G_d(n)$ with n ones and weight k which is a solution of Problem 2, its dual configuration has $2^d - 2^t - n - 1$ ones and zeros on level t and weight $M(d) - 2^t(d-t) - k$. $\square$

Similarly to Problem 1, the number of solutions of Problem 2 are coefficients of the following polynomials.

$$p_{d,t}(x,y) = \frac{\prod_{k=0}^{d-1}(1+xy^{d-k})^{2^k}}{(1+xy^{d-t})^{2^t}}$$

Claim 3.2. *The coefficient of $p_{d,t}(x,y)$ of the term $x^n y^k$ is equal to $s(d,t,n,k)$.*

Proof. We have that $p_{d,t}(x,y) = \dfrac{p_d(x,y)}{(1+xy^{d-t})^{2^t}}$. Therefore the coefficient of $x^n y^k$ is equal to the number of the solutions of Problem 1 where $r_t = 0$. Hence the coefficient of $x^n y^k$ is $s(d,t,n,k)$. $\square$

Claim 3.3.

$$den_{d,n}(x) = \sum_{k=k(d,n)}^{K(d,n)} s(d,n,K(d,n)+k(d,n)-k)x^{k-k(d,n)} \tag{3.2}$$

Proof. The coefficients of the polynomial $den_{d,n}(x)$ are the number of nodes on the levels of $G_d(n)$. The number of nodes on level k of $G_d(n)$ is $s(d,n,k+1)$. $\square$

Claim 3.4.

$$num_{d,n,r,w}(x) = \binom{2^{r-1}}{w} x^{K(d,n)-K(d,r-1,n-w)-w(d-r+1)} \tag{3.3}$$

$$\sum_{k=k(d,r-1,n-w)}^{K(d,r-1,n-w)} s(d,r-1,n-w,K(d,r-1,n-w)+k(d,r-1,n-w)-k)x^{k-k(d,r-1,n-w)}$$

Proof. There are $\binom{2^{r-1}}{w}s(d,r-1,n-w,K(d,r-1,n-w)-k)$ with weight $K(d,r-1,n-w)-k+w(d-r+1)$. The maximal weight for the configurations with w nodes on row r is $K(d,r-1,n-w)+w(d-r+1)$ so we adjust the numerators with $x^{K(d,n)-K(d,r-1,n-w)-w(d-r+1)}$. $\square$

Now we express the polynomials $den_{d,n}(x)$ and $num_{d,n,r,w}(x)$ as coefficients of the polynomials $q(x,y)$ and $q_{r,w}(x,y)$ defined bellow. Let

$$q(x,y) = \prod_{s=1}^{d}(y+x^s)^{2^{d-s}} \tag{3.4}$$

and

$$q_{r,w}(x,y) = \binom{2^{r-1}}{w} y^w x^{(d-r+1)(2^{r-1}-w)} \frac{\prod_{s=1}^{d}(y+x^s)^{2^{d-s}}}{(y+x^{d-r+1})^{2^{r-1}}} \tag{3.5}$$

The polynomials $q(x,y)$ and $q_{r,w}(x,y)$ are polynomials in two variables. The coefficient of y^n is a polynomial of x.

Claim 3.5. *The coefficient of y^n in $q(x,y)$ is $x^{k(d,2^d-n-1)}den_{d,n}(x)$.*

Proof. We have that

$$q(x,y) = \prod_{s=1}^{d}(y+x^s)^{2^{d-s}} = \prod_{s=1}^{d}\sum_{i=0}^{2^{d-s}} \binom{2^{d-s}}{2^{d-s}-i} y^i x^{s(2^{d-s}-i)}$$

The coefficient of $y^n x^k$ is equal to $\prod_{s=1}^{d}\binom{2^{d-s}}{2^{d-s}-i_s}$ where

$$\sum_{s=1}^{d} i_s = n \quad \text{and} \quad \sum_{s=1}^{d} s(2^{d-s}-i_s) = k$$

Let $j = d-s$ and $r_j = 2^{d-s} - i_s = 2^j - i_{d-j}$. Then $s = d-j$ and

$$\sum_{j=0}^{d-1} r_j = 2^d - n - 1 \quad \text{and} \quad \sum_{j=0}^{d-1}(d-j)r_j = k$$

Therefore the coefficient of $y^n x^k$ is $s(d, 2^d-n-1, k)$. The coefficient of y^n is $\sum_{k=k(d,2^d-n-1)}^{K(d,2^d-n-1)} s(d,2^d-n-1,k)x^k$. We have that

$$\sum_{k=k(d,2^d-n-1)}^{K(d,2^d-n-1)} s(d,2^d-n-1,k)x^k = \sum_{k=k(d,2^d-n-1)}^{K(d,2^d-n-1)} s(d,n,M(d)-k)x^k$$

$$= \sum_{r=0}^{K(d,n)-k(d,n)} s(d,n,M(d)-k(d,2^d-n-1)-r)x^{k(d,2^d-n-1)+r}$$

$$= x^{k(d,2^d-n-1)} \sum_{r=0}^{K(d,n)-k(d,n)} s(d,n,K(d,n)-r)x^r$$

$$= x^{k(d,2^d-n-1)} \sum_{r=k(d,n)}^{K(d,n)} s(d,n,K(d,n)+k(d,n)-k)x^{k-k(d,n)}$$

$$\sum_{k=k(d,2^d-n-1)}^{K(d,2^d-n-1)} s(d,2^d-n-1,k)x^k = x^{k(d,2^d-n-1)}den_{d,n}(x)$$

□

Claim 3.6. *The coefficient of y^n in $q_{r,w}(x,y)$ is $x^{k(d,2^d-n-1)}num_{d,n,r,w}(x)$.*

Proof. Let

$$q'_{r,w}(x,y) = \frac{\prod_{s=1}^{d}(y+x^s)^{2^{d-s}}}{(y+x^{d-r+1})^{2^{r-1}}} = \prod_{\substack{1\le s\le d\\ s\ne d-r+1}} \sum_{i=0}^{2^{d-s}} \binom{2^{d-s}}{2^{d-s}-i} y^i x^{s(2^{d-s}-i)}$$

The coefficient of $y^{n-w}x^k$ in $q'_{r,w}(x,y)$ is equal to $\prod_{1\le s\le d}\binom{2^{d-s}}{2^{d-s}-i_s}$, where

$$\sum_{1\le s\le d} i_s = n-w \quad \text{and} \quad \sum_{1\le s\le d} s(2^{d-s}-i_s) = k \quad \text{and} \quad i_{d-r+1} = 2^{r-1}$$

Let $j = d-s$ and $q_j = 2^{d-s} - i_s = 2^j - i_{d-j}$. Then

$$q_{r-1} = 2^{r-1} - i_{d-r+1} = 2^{r-1} - 2^{r-1} = 0$$

and

$$\sum_{j=0}^{d-1} q_j = \sum_{s=1}^{d}(2^{d-s}-i_s) = 2^d - n + w - 1$$

$$\sum_{j=0}^{d}(d-j)q_j = \sum_{s=1}^{d} s(2^{d-s}-i_s) = k$$

Therefore the coefficient of $y^{n-w}x^k$ in $q'_{r,w}(x,y)$ is $s(d,r-1,2^d-n_w-1,k)$. Let A_n be the coefficient of y^n in $q_{r,w}(x,y)$. Then

$$A_n = \binom{2^{r-1}}{w} x^{(d-r+1)(2^{r-1}-w)} \sum_{k=k(d,r-1,2^d-2^r-n+w-1)}^{K(d,r-1,2^d-2^r-n+w-1)} s(d,r-1,2^d-2^{r-1}-n+w-1,k)x^k$$

Let

$$A_n = \binom{2^{r-1}}{w} x^{-w(d-r+1)} B_n$$

where

$$B_n = \sum_{k=k(d,r-1,2^d-2^r-n+w-1)}^{K(d,r-1,2^d-2^r-n+w-1)} s(d,r-1,2^d-2^{r-1}-n+w-1,k)x^{k+2^{r-1}(d-r+1)}$$

By Lemma 3.2 we have that

$$s(d,r-1,2^d-2^{r-1}-n+w-1,k) = s(d,r-1,n-w,M(d)-2^{r-1}(d-r+1)-k)$$

Then

$$B_n = \sum_{k=k(d,r-1,2^d-2^r-n+w-1)}^{K(d,r-1,2^d-2^r-n+w-1)} s(d,r-1,n-w,M(d)-2^{r-1}(d-r+1)-k)x^{k+2^{r-1}(d-r+1)}$$

Let's substitute

$$M(d) - 2^{r-1}(d-r+1) - k = K(d,r-1,n-w) + k(d,r-1,n-w) - l$$

Then

$$k + 2^{r-1}(d-r+1) = l + M(d) - K(d,r-1,n-w) - k(d,r-1,n-w)$$

and

$$B_n = \sum_{l=*}^{**} s(d,r-1,n-w,K(d,r-1,n-w)+k(d,r-1,n-w)-l)x^{l-k(d,r-1,n-w)+d_l}$$

where $d_l = M(d) - K(d,r-1,n-w)$. We have that

$k(d,r-1,2^d-2^r-n+w-1) + K(d,r-1,n-w) = M(d) - 2^{r-1}(d-r+1)$

Then $* = k(d,r-1,n-w)$ and $** = K(d,r-1,n-w)$. Therefore

$$B_n = \frac{num_{d,n,r,w}(x)x^{M(d)-K(d,r-1,n-w)}}{\binom{2^{r-1}}{w}x^{K(d,n)-K(d,r-1,n-w)-w(d-r+1)}}$$

$$B_n = \frac{num_{d,n,r,w}(x)}{\binom{2^{r-1}}{w}}x^{M(d)-K(d,n)+w(d-r+1)}$$

$$B_n = \frac{num_{d,n,r,w}(x)}{\binom{2^{r-1}}{w}}x^{k(d,2^d-n-1)+w(d-r+1)}$$

Hence

$$A_n = \binom{2^{r-1}}{w}x^{-w(d-r+1)}B_n$$

$$A_n = \binom{2^{r-1}}{w}x^{-w(d-r+1)}\frac{num_{d,n,r,w}(x)}{\binom{2^{r-1}}{w}}x^{k(d,2^d-n-1)+w(d-r+1)}$$

$$A_n = num_{d,n,r,w}(x)x^{k(d,2^d-n-1)}$$

□

4 Recursive Algorithm

In section 3 we described the solutions of Problem 1 and Problem 2 as well as numerator and the denominator of the probability function $prob_{d,r,w}(p)$ as polynomial coefficients. In Claim 4.1 and Claim 4.4 we find recursive formulas to calculate the number of the solutions of Problem 1 and Problem 2. We calculate the numerator and the denominator of the probability function with formulas 3.2 and 3.3.

Claim 4.1.

$$s(d,n,k) = \sum_{r=0}^{n}\binom{2^{d-1}}{r}s(d-1,n-r,k-n)\epsilon(r)$$

where

$$\epsilon(r) = \begin{cases} 1 & if \quad k(d-1,n-r) \le k-n \le K(d-1,n-r) \\ 0 & otherwise \end{cases}$$

Proof. The bottom level of the tree contains 2^{d-1} points corresponding to the numbers $x_{2^{d-1}}, ..., x_{2^d-1}$. When r of them are equal to 1, the points of the top $d-1$ levels satisfy

$$\sum_{i=1}^{d-1} r_i = n - r \quad \text{and} \quad \sum_{i=1}^{d-1} (d-i-1) r_i = k - n$$

The above equations have $s(d-1, n-r, k-n)$ solutions when

$$k(d-1, n-r) \leq k - n \leq K(d-1, n-r)$$

□

We defined the sum of the weights of all nodes as $M(d) = 2^{d+1} - d - 2$. Let $M_t(d)$ be the sum of the weights of the nodes of the tree except the nodes on level t. Then

$$M_t(d) = M(d) - 2^t(d-t)$$
$$M_t(d) = 2^{d+1} - 2^t(d-t) - d - 2$$

Let $K_t(d,n)$ and $k_t(d,n)$ be the maximal and minimal values of k for which $s(d,t,n,k) \neq 0$. The function $K_t(d,n)$ is determined from $K(d,n)$ in the following way.

Claim 4.2.

$$K_t(d,n) = \begin{cases} K(d,n) & if \quad n < 2^t \\ K(d, n+2^t) - 2^t(d-t), & if \quad n \geq 2^t \end{cases}$$

Proof. The maximal value of $k = \sum_{i=1}^{d} (d-i) r_i$ occurs when the nodes which have a number 1 are on the top levels of the tree. When $n < 2^t$ r the nodes with number 1 are in the top $t-1$ levels and we have that $K_t(d,n) = K(d,n)$. The tree has $2^t - 1$ nodes on levels $1, 2, ..., t-1$. When $n \geq 2^t$ some of the nodes with number 1 will be on levels $t+1, t+2, ..., d$. The sum of the weights of the points on level t is $2^t(d-t)$. Then $K_t(d,n) = K(d, n+2^t) - 2^t(d-t)$. □

Claim 4.3.

$$k_t(d,n) = M_t(d) - K_t(d, 2^d - 2^t - n - 1)$$

Proof. The minimum of $k = \sum_{i=1}^{d} (d-i) r_i$ occurs when the n nodes with number 1 are in the bottom levels of the tree. Then the remaining $2^d - 2^t - n - 1$ nodes with number 0 occupy the top levels of the tree. The sum of the weights of all points is $M_t(d)$. Then $k_t(d,n) = M_t(d) - K_t(d, 2^d - 2^t - n - 1)$. □

Claim 4.4. *The numbers $s(d,t,n,k)$ are calculated recursively as follows*

$$s(d, d-1, n, k) = s(d-1, n, k-n)$$

$$s(d,t,n,k) = \sum_{r=0}^{n} \binom{2^{d-1}}{r} s(d-1, t, n-r, k-n) \epsilon(r)$$

where $\epsilon(r) = 1$ *when*

$$k(d-1,t,n-r) \leq k-n \leq K(d-1,t,n-r) \& n < 2^{d-1} - 2^t + r$$

and $\epsilon(r) = 0$ *otherwise.*

Proof. When $t = d-1$ the nodes with number are on the top $d-1$ levels of the tree. In this case the solutions of Problem 2 satisfy:

$$\sum_{i=1}^{d} r_i = n \quad \text{and} \quad \sum_{i=1}^{d}(d-i)r_i = k \quad \text{and} \quad r_d = 0$$

Then

$$\sum_{i=1}^{d}(d-i)r_i = \sum_{i=1}^{d-1}(d-i)r_i = \sum_{i=1}^{d-1}(d-1-i)r_i + \sum_{i=1}^{d} r_i = \sum_{i=1}^{d-1}(d-1-i)r_i + n$$

$$\sum_{i=1}^{d-1}(d-1-i)r_i = k-n$$

because $\sum_{i=1}^{d}(d-i)r_i = k$. Therefore when $t = d-1$ the solutions of Problem 2 correspond to the solutions of Problem 1 for the tree with $d-1$ levels which has n nodes with number 1 and sum of the weights $k-n$. Then $s(d, d-1, n, k) = s(d-1, n, k-n)$. Let $t < d-1$ and r be the number of nodes with number 1 on level $d-1$. The remaining $n-r$ nodes are on the top $d-1$ levels of the tree. These configurations are solutions of Problem 1 for the tree with $d-1$ levels and $n-r$ nodes with number 1 which have total weight $k-n$. Problem 1 has a solution with these parameters when $n-r < 2^{d-1} - 2^t$ and

$$k(d-1,t,n-r) \leq k-n \leq K(d-1,t,n-r) \& n < 2^{d-1} - 2^t + r$$

□

5 Final Remarks

In section 4 we found recursive formulas to compute the values of the probability function for the number of 0*s* and 1*s* on the levels of the tree T_d. In the appendix we give the values of the probabilities for the first three levels for $d = 1, 2, 3, 4, 5, 6, 7$ and $p = 0.25, 0.5$. The probability that all nodes have number 1 is the rectangle in the barplots colored in black. This probability increases and approaches 1 as n increases. In section 2 we used the tree structure of the network to show that the graphs $G_d(n)$ are connected. In sections 3 and 4 we developed polynomial and recursive formulas for the probability function $prob_{d,,r,w}(p)$. These formulas use only the connectivity of the graphs $G_d(n)$. This observation suggests that we can extend our results to much wider class of networks with arbitrary number of nodes on each level and arbitrary number of connections between the neighboring levels provided that they are connected and there are no connections within each level. The model of self-organizing networks that we proposed here can be generalized to more realistic one by considering networks with arbitrary number of nodes on each level as well as assigning different probabilities for the computational power of the nodes.

6 Appendix

$d=3, p=.25$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.1237	0.8763			
Level 2	0.1316	0.4347	0.4336		
Level 3	0.2222	0.3736	0.2812	0.1065	0.01654

$d=4, p=.25$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0393	0.9607			
Level 2	0.0213	0.2347	0.7440		
Level 3	0.0286	0.1448	0.3185	0.3498	0.1584

$d=5, p=.25$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0107	0.9893			
Level 2	0.0018	0.0792	0.9190		
Level 3	0.0006	0.0122	0.0951	0.3561	0.5359

$d=6, p=.25$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0028	0.9972			
Level 2	0.0001	0.0217	0.9781		
Level 3	$< 10^{-5}$	0.0003	0.0102	0.1486	0.8409

$d=7, p=.25$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0007	0.9993			
Level 2	$< 10^{-5}$	0.0056	0.9944		
Level 3	$< 10^{-5}$	$< 10^{-5}$	0.0007	0.0430	0.9563

Table 1: The values of the probability function for the first three levels of the tree for $p = 0.25$ and $d = 3, 4, 5, 6, 7$.

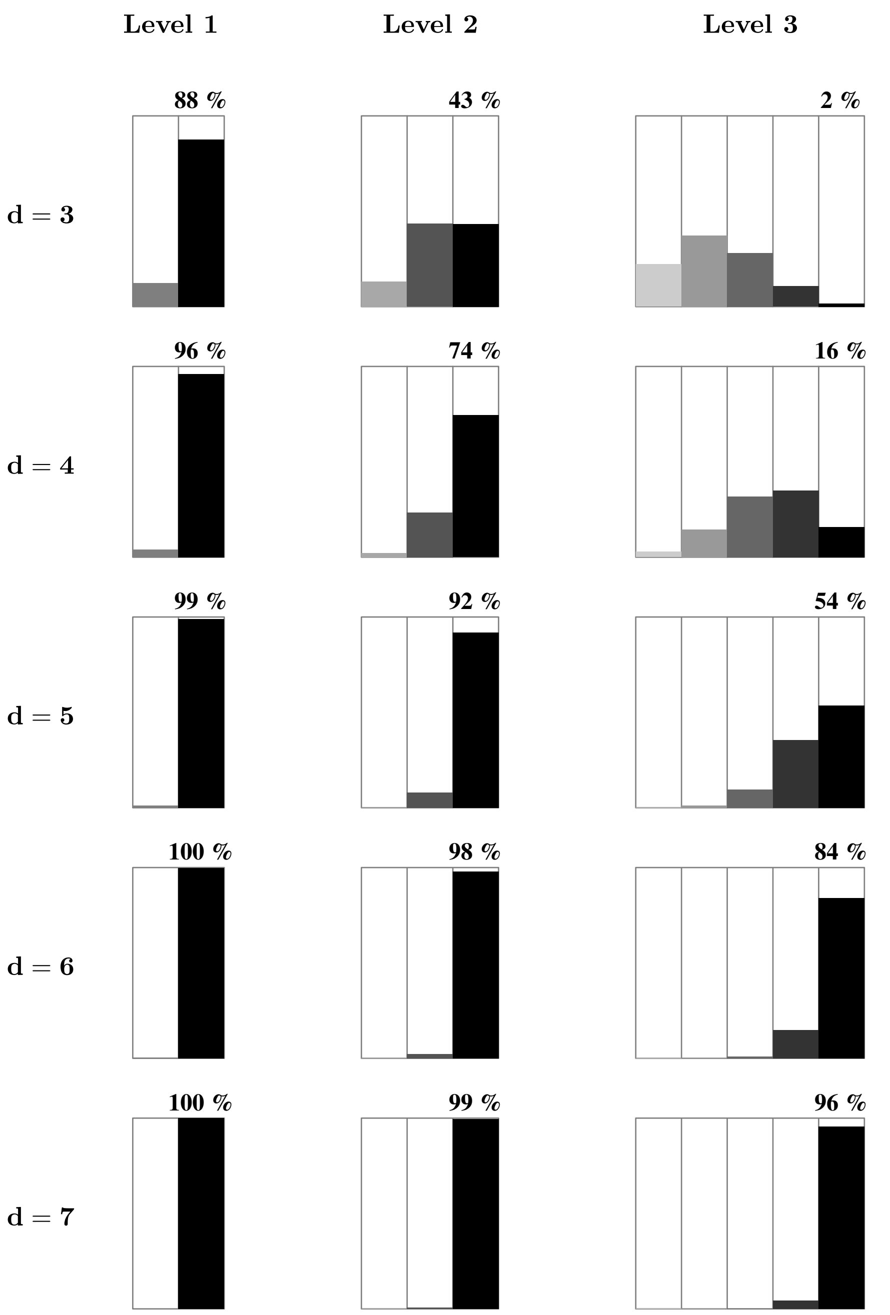

Figure 3: The graphs of the probability function from Table 1 for $p = 0.25$.

$d=3, p=.5$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.2712	0.7288			
Level 2	0.1832	0.4835	0.3333		
Level 3	0.1304	0.3372	0.3425	0.1607	0.0292

$d=4, p=.5$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.1706	0.8294			
Level 2	0.0858	0.4098	0.5044		
Level 3	0.0430	0.2006	0.3621	0.2992	0.0952

$d=5, p=.5$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0981	0.9019			
Level 2	0.0323	0.2924	0.6753		
Level 3	0.0088	0.0783	0.2657	0.4081	0.2391

$d=6, p=.5$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0530	0.9470			
Level 2	0.0102	0.1810	0.8088		
Level 3	0.0012	0.0203	0.1341	0.3977	0.4468

$d=7, p=.5$	$X=0$	$X=1$	$X=2$	$X=3$	$X=4$
Level 1	0.0276	0.9724			
Level 2	0.0029	0.1018	0.8953		
Level 3	0.0001	0.0039	0.0505	0.2952	0.6503

Table 2: The values of the probability function for the first three levels of the tree for $p = 0.5$ and $d = 3, 4, 5, 6, 7$.

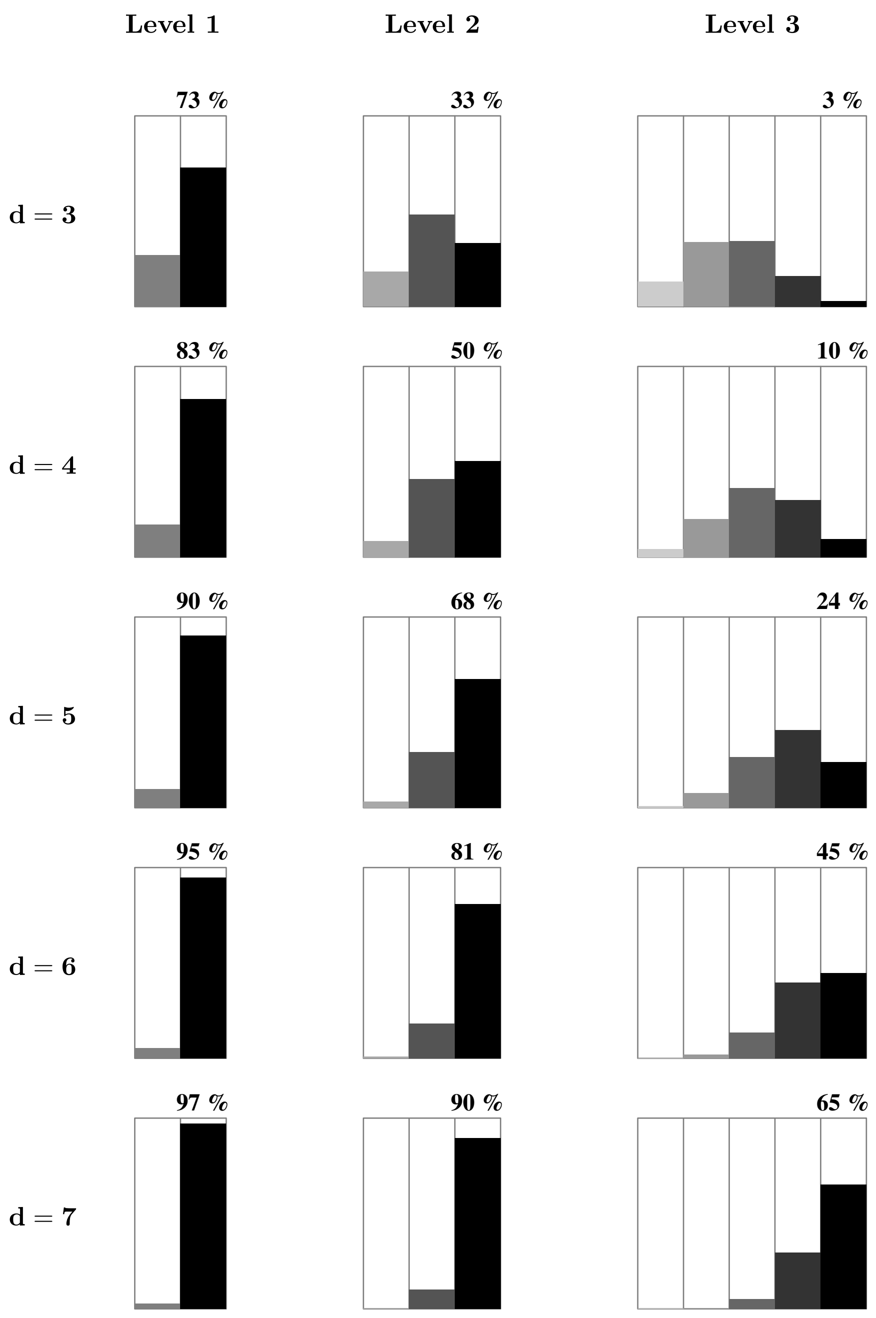

Figure 4: The graphs of the probability function from Table 2 for $p = 0.5$.

References

[1] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Santa Fe Institute Studies in the Sciences of Complexity, 1999.

[2] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. Application-specific scheduling for the Organic Grid. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)*, pages 146–155, Pittsburgh, November 2004.

[3] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. The Organic Grid: Self-organizing computation on a peer-to-peer network. In *Proceedings of the International Conference on Autonomic Computing*, pages 96–103. IEEE Computer Society, May 2004.

[4] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.

[5] A. Gierer and H. Meinhardt. A theory of biological pattern formation. *Kybernetik*, 12:30–39, 1972.

[6] A. Montresor, H. Meling, and O. Babaoglu. Messor: Load-balancing through a swarm of autonomous agents. In *Proceedings of 1st Workshop on Agent and Peer-to-Peer Systems*, number 2530 in Lecture Notes in Artificial Intelligence, pages 125–137. Springer-Verlag, July 2002.

[7] A. Turing. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. London*, 237(B):37–72, 1952.