A combinatorial model for self-organizing networks

Yuri Dimitrov[1], Carlo Giovine[2*], Gennaro Mango[2*], Mario Lauria[3]

[1] Dept. of Mathematics
The Ohio State University, Columbus, OH
yuri@math.ohio-state.edu

[2]Dip. di Informatica e Sistemistica
Università di Napoli "Federico II", Naples, Italy
{gennaro.mango, carlo.giovine}@unina.it

[3] Dept. of Computer Science and Engineering
The Ohio State University, Columbus, OH
lauria@cse.ohio-state.edu

[3] Telethon Institute of Genetics and Medicine
(TIGEM), Naples, Italy
lauria@tigem.it

Abstract

In previous works we have proposed to use of self-organization based on emergent design as a model for the programming of very large aggregates of heterogeneous computing resources. In our approach, a large scale computation is divided into small independent units of computation, each provided with its own uniform, autonomous behavior; only local information is used by each unit of computation to take all the decisions needed to carry out the computation. One of the challenges of this novel approach is to provide some theoretical foundation that can assist in the rational design of new systems.

In this paper is to demonstrate the use of combinatorial techniques for obtaining quantitative analytical models of the organization pattern emerging from a specific type of self-organizing computation. Specifically, in a previous experiment we have demonstrated a computation in which mobile agents organize themselves around an overlay tree, that constantly restructures itself in response to changing node availability and performance levels. In this paper we derive an analytical expression describing how nodes distribute themselves over the tree based on their performance, in a simplified version of the above problem. This result represents an instance of a theoretical tool that can be used to predict global patterns emerging as a result of a self-organizing design, and to establish a direct connection between global features and local behavior parameters.

[1]This work was performed while C. G. and G. M. were visiting Mario Lauria's group at the Ohio State University

1-4244-0910-1/07/$20.00

1 Introduction

A growing number of computer science fields relies on systems composed of large numbers of independent computing elements. Desktop grids composed of thousands to millions of PCs have been used to perform some of the largest computations in the world [30, 37, 16]. The current trend in Massively Parallel Processor (MPP) systems is exemplified by machines like IBM's Blue Gene/L which will have more than 100K processors at full configuration [14]. High-end PC/workstation clusters of increasing size have been built over the years and the ones currently appearing near the top of the Top500 supercomputer ranking have around 8K nodes [33]. Sensor networks of 10K nodes are being prototyped [15], and much larger numbers of elements are expected for MEMS-based projects like Smart Dust [23].

We have proposed to use self-organization as a model for the programming of very large aggregates of heterogeneous computing resources, for the purpose of running high performance computing (HPC) applications that are too large for even the largest current supercomputers [10, 9, 11]. Self-organizing computation based on emergent design, in the literature also referred to with other names such as emergent computation [17] or amorphous computing [1], is a type of computation in which the desired global organization of the system emerges from local and long range interactions among component parts. This concept is very general and has been applied to areas such as, among others, artificial neural networks, adaptive systems, connectionist learning, cellular computing, organization of social insects, and biological networks [35, 6].

In our approach, a large scale computation is divided into small independent units of computation, each provided with

its own uniform, autonomous behavior; only local information or very limited amount of non-local information is used by each unit of computation to take all the decisions needed to carry out the computation. Allowing global coordination to emerge from a distributed collection of simple components has important advantages over explicit central control in both natural and artificial systems information processing systems. There are well known costs in associated to centralized coordination, such as i) speed (a central coordinator can become the performance bottleneck), ii) robustness (if the coordinator fails, the whole systems halts), and iii) control overhead (the overhead incurred in gathering and updating the state information about the system).

By using a decentralized, adaptive scheduling approach, we attempt to enable a number of tasks to be run on large numbers of machines connected by an unreliable network. Our approach can be used to broaden the class of applications that can be run on a large desktop grid, or to extend a traditional Grid computing approach to platforms with highly unreliable connections or highly unpredictable levels of availability.

In the course of this project we are investigating the theoretical foundations of self-organization applied to HPC. The challenges of developing a theoretical framework for the self-organizing approach can be summarized by the two following fundamental questions:

- Given a description of the local behavior, is it possible to make predictions on the resulting global organization of computation, that emerges as a result of the assigned behavior?
- Once identified a desirable feature of the global system, is it possible to provide design guidelines for the local behavior needed to achieve the emergence of the assigned behavior?

The objective of this paper is to report on one of our first attempts toward the development of the theoretical foundations that will ultimately enable the rational design of large scale system. Based on our experiments with a proof of concept prototype called the Organic Grid, we have identified key aspects of the design process that would benefit from methods of rational design and analysis. Our approach to the development of these methods has been to study and adapt techniques developed in other fields of science for the treatment of large collections of simple elements. In this paper we will present the preliminary results for the case of combinatorial analysis reminiscent of the study of thermodynamic ensembles.

1.1 Related Work

Despite an obvious interest and the numerous practical applications, the engineering of general purpose large scale computing systems is still largely matter of research. In the following we briefly review current approaches and how they relate to ours.

Internet computing and Desktop grids. These large scale systems are based on variants of the master/workers model [4, 30, 37, 18, 16, 12, 26, 27, 21, 25, 7, 24]. The level of technical maturity reached by some of these projects is demonstrated by the fact that they have spawned commercial enterprises. However they have two fundamental problems. The achievable computing power of this approach is constrained by the performance of the master (especially for data-intensive applications) and by the difficulty of deploying the supporting software on a large number of workers. Second, despite the spectacular success of some Internet computing projects, these systems are also severely restricted in the choice of suitable applications; the underlying master/worker model combined with the extreme bandwidth requirements for the master makes these large scale deployment suitable only for independent task applications with high granularity individual tasks.

Grid scheduling. Research on grid scheduling has focused on algorithms to determine an optimal computation schedule based on the assumption that sufficiently detailed and up-to-date knowledge of the systems state is available to a single entity (the metascheduler) [20, 5, 2, 31]. While this approach results in a very efficient utilization of the resources, it does not scale well to large systems. Maintaining a global view of the system becomes prohibitively expensive and on unreliable networks might even make it impossible.

2 Self-organization of Complex Systems

This section contains a brief introduction to the crucial concepts and technologies used in our work on autonomic scheduling, as well as the related work in these areas. These include: Peer-to-Peer and Internet computing, self-organizing systems and the concept of emergence.

Peer-to-Peer and Internet Computing The idea of utilizing the CPU cycles of idle machines was first put to work in the Worm project [22] at Xerox PARC. Further progress was achieved by academic projects such as Condor [26]. The growth of the Internet and the wide availability of inexpensive broadband connections to home and the office made large-scale efforts like GIMPS [37], SETI@home [30] and folding@home [16] feasible. Recently, commercial solutions such as Entropia [12] and United Devices [36] have also been developed.

Still, all these approaches suffer from a fundamental limitation: computation must be divisible into a large number of independent tasks to be organized according to a master/worker model. Instead, the idea of combining Internet resources for general purpose large scale computations using a peer-to-peer approach is attractive because of the potential for almost unlimited computational power, low cost, ease and universality of access — a true Computational Grid. Among the technical challenges posed by such an architecture, scheduling is one of the most formidable — how to organize computation on a highly dynamic system at a planetary scale while relying only on local knowledge of its state.

Self-Organization of Complex Systems 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 [34, 19]. 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 [29, 6].

In a departure from the methodological approach followed in previous projects, we did not try to accurately reproduce a naturally occurring behavior. Rather, we started with a problem and then designed a completely artificial behavior that would result in a satisfactory solution to it. Our work is somewhat closer to the self-organizing computation concept explored in the Co-Fields project [28]. The idea behind Co-Fields is to drive the organization of autonomous agents through artificial potential fields.

Our work was inspired by a particular version of the emergence principle called Local Activation, Long-range Inhibition (LALI) [32]. The LALI rule is based on two types of interactions: a positive, reinforcing one that works over a short range, and a negative, destructive one that works over longer distances. We retain the LALI principle but in a different form: we use a definition of distance which is based on a performance-based metric. Nodes are initially recruited using a friends list (a list of some other peers on the network) in a way that is completely oblivious of distance, therefore propagating computation on distant nodes with same probability as close ones. During the course of the computation agents behavior encourages the propagation of computation among well-connected nodes while discouraging the inclusion of distant (i.e. less responsive) agents.

2.1 The Organic Grid

We have developed a framework called the Organic Grid [10, 9, 11] 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.

Our approach was to encapsulate computation and behavior into Java mobile agents, which deliver the computation to available machines. Java mobile agents provided a convenient implementation platform that allowed us to concentrate on the design issues of a self-organizing computation. However self-organization is a very general principle and other type of implementations are possible; for example the use of virtual machines and virtual network connections, possibly augmented with a mechanism for checkpointing/recovery, would provide comparable functionality in terms of computation isolation, code distribution, process migration.

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 briefly describe our experience in designing agent behavior for running two representative applications on an Organic Grid: the NCBI BLAST code for sequence alignment, and Cannon's algorithm for matrix multiplication.

In the first Organic Grid prototype we demonstrated how to apply our decentralized approach to a class of applications representing the main staple of grid scheduling research, namely an *independent task application* (or ITA) [10]. The specific application we used was BLAST, a popular sequence alignment tool. The results of our experiments using the ITA are available in [10].

For an ITA, the computation spreads out from its source in the form of a tree. The source distributes the data in the form of computational subtasks that flow down the tree; results flow towards the root. This same tree structure was used as the overlay network for making scheduling decisions. In general, there could be separate overlay networks: for data distribution, for scheduling, and for communication between subtasks. In the case of an ITA, there is no communication between subtasks while the overlay trees for data distribution and scheduling overlap.

In order to demonstrate the generality of the self-organizing approach and the flexibility of the Organic Grid scheduling framework, we selected a second application characterized by a highly regular and synchronous pattern of communication — Cannon's matrix multiplication algorithm [8]. This application employs three different overlay networks: a star topology for data distribution, a torus for the communication between subtasks, and the tree overlay of the scheduling framework.

More details on these experiments are available in [9]

In both experiments we implemented an adaptive tree mechanism in the agent behavior in order to select in a decentralized manner the best available machines. Machine performance was constantly evalued with a mechanism based on passive feedback, represented by the time taken by each child to perform a known amount of computation.

In the Cannon's algorithm case, we experimented with a desktop grid of 20 agents that whose behavior was designed to build a tree overlay network, of which the first 16 to contact the distribution agent were included in a torus with 4 agents along each dimension; the remaining agents acted as extras in case any faults occurred. The initial tree and torus can be seen in Figure 1 with 4 slow nodes in the torus and 4 extra, fast nodes.

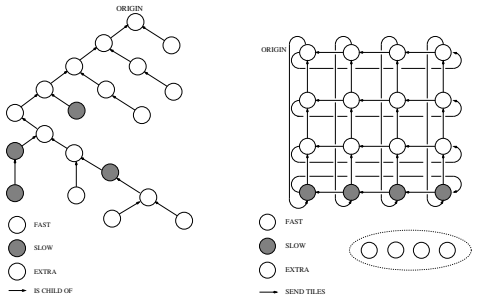

Figure 1. Original tree and torus overlays

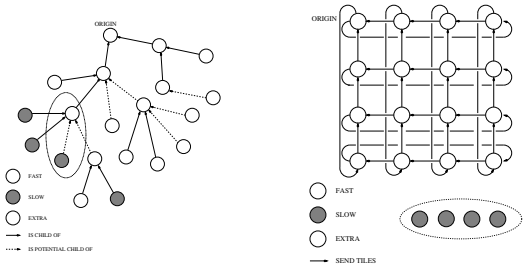

Figure 2. Tree and torus overlays after Fourth Swap

The structure of the tree continually changed and the high-performance nodes were pushed up towards the root. When a fast, extra node found that one of its children was slower than itself and part of the torus, it initiated a swap of roles. The topology of the tree and the torus before and after the fourth swap are shown in Figure 2. With a few modifications the same mechanism was also employed to rebuild the torus on the fly after a node failure.

3 Combinatorial Model

During our Organic Grid experiments an important performance parameter turned out to be the percentage of high performing nodes promoted to the upper levels of the tree. We resorted to perform large number of runs to collect statistics, and eventually we asked ourselves if it were possible to analytically forecast the distribution of node performance at every level of the tree, given a rule for tree reorganization as part of the agent behavior and given an initial distribution of node performance.

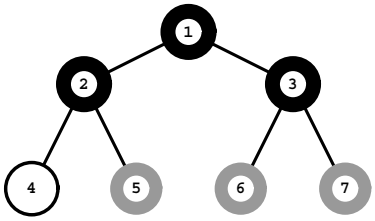

Figure 3. The maximal and minimal configurations with three $1s$ in T_3

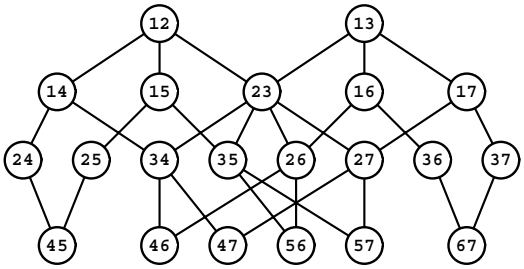

Figure 4. The graph $G_3(2)$.

We approached the problem by performing a combinatorial analysis of the possible tree configurations; given the complexity of this type of analysis, we started with very simple tree topologies (complete binary trees; only two levels of node performance, low ("0") and high ("1")) and a simplified parent-child exchange rule. In other words, we were able to solve the following problem:

The binary tree T_d with d levels and $2^d - 1$ 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 each level of the tree T_d.

3.1 Configurations

The tree T_d has d levels $0, 1, ..., d-1$ where the root is on level 0 and level i consists of 2^i vertices. For convenience let's assign weights to the vertices of T_d such that the root has weight d and the vertices on level l have weight $d - l$. We also enumerate the vertices with numbers $1, 2, ..., 2^d-1$. The root has number 1 and the vertices on level i have numbers $2^i, ..., 2^{i+1} - 1$ from left to right. Initially we assign randomly $0s$ and $1s$ to the nodes of the tree which represent the computers in a hierarchically organized network. Let n be the number of $1s$ in the nodes. The possible values for n are $1, 2, ..., 2^d - 1$, binomially distributed with probability of success 0.5. The states of the network are all configurations with n ones. The weight of a configuration is the sum of the weights of its nodes.

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: $\{2^d - n, 2^d - n + 1, ..., 2^d - 1\}$.

Let $G_d(n)$ be the graphs of the configurations with n ones. The graphs $G_d(n)$ have $\binom{2^d-1}{n}$ vertices and two configurations are adjacent if they are consecutive states of the system. If C_1 and C_2 are two adjacent configurations 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. Let $l(v)$ be the level of vertex v in T_d. The levels of the graphs of configurations $G_d(n)$ are induced by the levels of T_d in the following way:

$$\text{If } l(v_2) = l(v_1) + 1 \text{ then } l(C_1) = l(C_2) + 1$$

In Figure 4 we show the graph $G_3(2)$ which consists of the configurations with two ones of T_3.

The first level of $G_3(2)$ has two vertices corresponding to the configurations with two ones in T_3 in vertices $1, 2$ and $1, 3$. The graphs $G_d(n)$ and $G_d(2^d - n - 1)$ are dual. They are the same graph with its levels reversed. Now we outline the steps which lead to recursive formulas for calculating the distribution of $1s$ on the levels of T_d. Additional details and proofs are available in [13].

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

Let's denote by $p(C)$ the probability of configuration C in the steady state distribution of the network.

Lemma 1. *Let C and C' be two adjacent configurations in $G_n(d)$ such that $l(C') = l(C) + 1$. Then $p(C') = pp(C)$.*

Corollary 1. *Let C and C' be two configurations in $G_d(n)$ with $l(C) = l(C')$. Then $p(C) = p(C')$.*

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 the levels 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. The probability to have w points on row r is

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

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 configurations on each level of $G_d(n)$ have equal weight. The number of configurations on the levels 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*

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

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

We use the numbers $s(d,n,k)$ to determine the denominators of the probability function.

Claim 2.

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

The numerators of the probability function are determined in Claim 3 using the number of solutions of the following problem.

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* (1) *and $r_t = 0$. Determine the numbers $s(d,t,n,k)$.*

Claim 3.

$$num_{d,n,r,w}(x) = \binom{2^{r-1}}{w} x^{k_*} \sum_{k=k(d,r-1,n-w)}^{K(d,r-1,n-w)} s_* x^{k-k(d,r-1,n-w)}$$

where $k_* = K(d,n) - K(d,r-1,n-w) - w(d-r+1)$

$$s_* = s(d,r-1,n-w,k_{**})$$

and

$$k_{**} = K(d,r-1,n-w) + k(d,r-1,n-w) - k$$

3.2 Recursive Algorithm

In Claim 2 and Claim 3 we found formulas for the polynomials $den_{d,n}(x)$ and $num_{d,n,r,w}(x)$. 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

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

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

Now we describe a recursive algorithm for calculating the number of solutions of Problem 1 and Problem 2.

Claim 4.

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

where

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

and $k_* = K(d-1, n-r)$.

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 values of $K_t(d,n)$ and $k_t(d,n)$ are determined from $K(d,n)$ in the following way.

Claim 5.

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

and

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

In Claim 6 we use the values of $k_t(d,n)$ and $K_t(d,n)$ to compute the values of $s(d,t,n,k)$.

Claim 6. *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

$$\epsilon(r) = \begin{cases} 1 & if \quad k_t(d-1,n-r) \leq k-n \leq k_* \\ 0 & otherwise \end{cases}$$

and $k_* = K_t(d-1,n-r) \& n < 2^{d-1} - 2^t + r$.

In [13] we prove the statements from section 3 an we also show that the numerator and the denominator of the probability function are the coefficients of y^n of the following polynomials

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

and

$$q_{r,w}(x,y) = \binom{2^{r-1}}{w} y^w x^{(d-r+1)(2^{r-1}-w)} \prod_{\substack{1 \leq s \leq d \\ s \neq d-r+1}} (y + x^s)^{2^{d-s}}$$

3.3 Example

In Table 1 and Figure 4 we give the values and bar plots of the probability function $prob_{4,r,w}(0.5)$ for $r = 0, 1, 2$ and all values of w computed with the recursive formulas from section 3.2. The probabilities that all nodes on levels $0, 1$ and 2 have number 1 are approximately $83\%, 50\%$ and 10%. These probabilities approach 100% as the number of levels of T_d increases.

$p = .5$	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
r=0	0.171	0.829			
r=1	0.086	0.41	0.504		
r=2	0.043	0.201	0.362	0.299	0.095

Table 1. The values of the probability function on the first three levels of T_4 for $p = 0.5$.

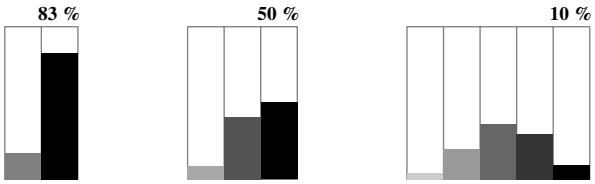

Figure 5. Bar plot of the probability function from Table 1

3.4 Future work

The results of our investigation into theoretical foundations for the Organic Grid will be tested in a variety of ways. An Organic Grid simulator currently under development will be the tool of choice to test large scale configurations. The simulator is under construction using Java, and it will be validated using our Organic Grid prototype, which is currently being ported to a higher performance mobile agent environment called ProActive [3]. On the analytical side, the logical steps used to obtain the probability function are based on the connectivity of the graph of configurations $G_d(n)$ and do not rely on the tree structure of the network. We therefore expect that our results can be generalized and applied to a much wider class of hierarchical self-organizing networks.

References

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, J. Thomas F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Commun. ACM*, 43(5):74–82, 2000.

[2] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with Nimrod/G: Killer application for the global grid? In *Proc. Intl. Parallel and Distributed Processing Symp.*, pages 520–528, May 2000.

[3] F. Baude, D. Caromel, F. Huet, and J. Vayssìere. Communicating mobile active objects in Java. In M. Bubak, H. Afsarmanesh, R. Williams, and B. Hertzberger, editors, *Proceedings of HPCN Europe 2000*, volume 1823 of *Lecture Notes in Computer Science*, pages 633–643. Springer Verlag, May 2000.

[4] Berkeley Open Infrastructure for Network Computing (BOINC).

[5] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive computing on the grid using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):369–382, 2003.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Santa Fe Institute Studies in the Sciences of Complexity, 1999.

[7] D. Buaklee, G. Tracy, M. K. Vernon, and S. Wright. Near-optimal adaptive control of a large grid application. In *Proceedings of the International Conference on Supercomputing*, pages 315–326, June 2002.

[8] L. Cannon. *A Cellular Computer to implement the Kalman Filter Algorithm*. PhD thesis, Montana State University, 1969.

[9] A. J. Chakravarti, G. Baumgartner, and M. 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.

[10] A. J. Chakravarti, G. Baumgartner, and M. 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.

[11] A. J. Chakravarti, G. Baumgartner, and M. 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.

[12] A. A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: architecture and performance of an enterprise desktop grid system. *J. Parallel and Distributed Computing*, 63(5):597–610, 2003.

[13] Y. Dimitrov and M. Lauria. A combinatorial model for self-organizing networks. Technical Report OSU-CISRC-1/07-TR02, Dept of Computer Science and Engeneering, The Ohio State University, Jan. 2007.

[14] N. R. A. et al. An overview of the bluegene/l supercomputer. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Press, 2002.

[15] ExScal (formerly Echelon). http://www.cse.ohio-state.edu/ duttap/echelon/start.html.

[16] folding@home.

[17] S. Forrest, editor. *Emergent computation*. MIT Press, Cambridge, MA, USA, 1991.

[18] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *Proc. IEEE Symp. on High Performance Distributed Computing (HPDC)*, pages 7–9, San Francisco, CA, August 2001.

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

[20] A. S. Grimshaw and W. A. Wulf. The Legion vision of a worldwide virtual computer. *Comm. of the ACM*, 40(1):39–45, Jan. 1997.

[21] E. Heymann, M. A. Senar, E. Luque, and M. Livny. Adaptive scheduling for master-worker applications on the computational grid. In *Proc. of the First Intl. Workshop on Grid Computing*, pages 214–227, 2000.

[22] J. A. H. John F. Shoch. The "Worm" programs — early experience with a distributed computation. *Comm. of the ACM*, 25(3):172–180, Mar. 1982.

[23] J. M. Kahn, R. H. Katz, and K. S. J. Pister. "next century challenges: mobile networking for smart dust". In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM Press, 1999.

[24] N. T. Karonis, B. Toonen, and I. Foster. MPICH-G2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 63(5):551–563, 2003.

[25] T. Kindberg, A. Sahiner, and Y. Paker. Adaptive Parallelism under Equus. In *Proceedings of the 2nd International Workshop on Configurable Distributed Systems*, pages 172–184, Mar. 1994.

[26] M. Litzkow, M. Livny, and M. Mutka. Condor — a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.

[27] M. Maheswaran, S. Ali, H. J. Siegel, D. A. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the 8th Heterogeneous Computing Workshop*, pages 30–44, Apr. 1999.

[28] M. Mamei and F. Zambonelli. Co-Fields: a Physically Inspired Approach to Distributed Motion Coordination. *IEEE Pervasive Computing*, 3(2), April 2004.

[29] 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.

[30] SETI@home.

[31] I. Taylor, M. Shields, and I. Wang. *Grid Resource Management*, chapter 1 - Resource Management of Triana P2P Services. Kluwer, June 2003.

[32] G. Theraulaz, E. Bonabeau, S. C. Nicolis, R. V. Sol, V. Fourcassi, S. Blanco, R. Fournier, J.-L. Joly, P. Fernndez, A. Grimal, P. Dalle, and J.-L. Deneubourg. Spatial patterns in ant colonies. *PNAS*, 99(15):9645–9649, 2002.

[33] TOP500 Supercomputing Sites project. http://www.top500.org/.

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

[35] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society (B)*, 237:37–72, 1952.

[36] United Devices. Grid computing solutions.

[37] G. Woltman. GIMPS: The great internet mersenne prime search.