Large Scale Simulations of Complex Systems Part I: Conceptual Framework

P.M.A. Sloot[*], A. Schoneveld, J.F. de Ronde, J.A. Kaandorp

July 25, 1997

Parallel Scientific Computing and Simulation Group
Faculty of Mathematics, Computer Science, Physics & Astronomy
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands
Phone: +31 20 5257463
fax: +31 20 5257490
E-mail: peterslo@wins.uva.nl
http://www.wins.uva.nl/research/pscs/

Abstract

In this working document, we report on a new approach to high performance simulation. The main inspiration to this approach is the concept of complex systems: disparate elements with well defined interactions rules and non non-linear emergent macroscopic behavior. We provide arguments and mechanisms to abstract temporal and spatial locality from the application and to incorporate this locality into the complete design cycle of modeling and simulation on parallel architectures.
Although the main application area discussed here is physics, the presented Virtual Particle (VIP) paradigm in the context of Dynamic Complex Systems (DCS), is applicable to other areas of compute intensive applications. Part I deals with the concepts behind the VIP and DCS models. A formal approach to the mapping of application task-graphs to machine task-graphs is presented. The major part of section 3 has recently (July 1997) been accepted for publication in Complexity. In Part II we will elaborate on the execution behavior of a series of large scale simulations using the concepts presented in this document. Note that this is a working document, where we present ongoing work on a formal description of DCS and compile new ideas and results of our research group.

[*] Visiting Professor Santa Fe Institute New Mexico (USA), Summer 1997

Contents

1 Introduction **4**

2 Background and Review of Concepts **8**
2.1 Physical Requirements for Computability 8
2.1.1 Information Theory in CA 9
2.1.2 Computation in the intermediate regime of ordered and disordered behavior 10
2.1.3 Computational Behavior as an Attractor: Self Organized Criticality 12
2.2 Examples of Intractability and undecidability in some Physical Systems 14
2.2.1 Optical Beam Tracing 14
2.2.2 N-body Simulation 15
2.2.3 Non Equilibrium Growth Processes 16

3 Theoretical Framework **20**
3.1 Parallel Task Allocation 20
3.1.1 Introduction 20
3.1.2 Application and Machine Models 22
3.1.3 Correlation Structure of the TAP 24
3.1.4 Relaxation of random walks 24
3.1.5 Random walks through the TAP landscape 27
3.1.6 The one step correlation function for the Task Allocation Problem 28
3.1.7 Physics of Task Allocation 33
3.1.8 Spin Glasses and Graph bi-partitioning 34
3.1.9 Task Allocation Hamiltonian 35
3.1.10 The TAP Phase Transition 35
3.1.11 Experimental Methods 37
3.1.12 Simulated Annealing For Optima Search 37
3.1.13 Search Cost Estimation 38
3.1.14 Measuring Phase Space Structure 39
3.1.15 Experimental Results 39
3.1.16 Statistical Quantities and Correlation Length 39
3.1.17 Experimental Verification of Cost Terms 40
3.1.18 Analytical and Measured λ 40
3.1.19 Phase Transitions and Computational Search Cost 41
3.1.20 Summary and Discussion 42
3.1.21 Statistical Quantities and Correlation Length 43
3.1.22 Phase Transitions and Computational Search Cost 44
3.1.23 Concluding Remarks and Future Work 44
3.2 An Abstract Cellular Genetic Algorithm for solving optimization problems 46

3.2.1 The Abstract Cellular Genetic Algorithm: a Theoretical Framework . . . 46
3.2.2 Instance of the ACGA I: Cellular Genetic Algorithm . . . 48
3.2.3 Instance of the ACGA II: Cellular Simulated Annealing . 49
3.2.4 Experiments on the Convergence Behavior of C-GA . . . 50
3.2.5 ACGA on a parallel architecture . . . 51

4 Case studies 51
4.1 Diffusion Limited Aggregation . . . 51
4.2 Lattice Boltzmann solver for flow and diffusion phenomena . . 55
4.2.1 Macroscopic equations . . . 57
4.2.2 Lattice gases in 3D . . . 57
4.2.3 Theoretical background of the Lattice Boltzmann method 58
4.3 A comparison: Lattice Gases, Lattice Boltzmann method, and PDE's . . . 61
4.4 FE - Applications Formalized by the VIP Model . . . 62
4.4.1 The Finite Element Technique . . . 62
4.4.2 FE as VIP . . . 63
4.4.3 parallel architectures as VIP . . . 63
4.4.4 Mapping FE - Parallel architecture . . . 64

5 Conclusions and future work 65

6 Appendix:
Computability in Dynamical Complex Systems 67
6.1 Undecidability and Intractability . . . 67
6.1.1 Church-Turing Hypothesis . . . 67
6.1.2 Undecidability . . . 68
6.1.3 Intractability . . . 69
6.2 Difficulty of Predicting Behavior in Complex Systems . . . 70
6.3 Simulation, Parallelism and P-completeness . . . 72
6.4 Computability: Cellular Automata . . . 74
6.4.1 An example of intractability and undecidability in a universal CA . . . 76

1 Introduction

Until now much experimental simulation work has been done in physics and chemistry on parallel computing systems. The development however of formal methods, suitable for modeling complex systems from nature and for an efficient mapping of the complex system onto parallel computing systems, is still in its infancy. The development of techniques, suitable for modeling complex systems stemming from physics, chemistry, and biology, and mapping these onto parallel platforms, belongs to one of the real challenges of future research, as identified in several NSF-reports [1] and CEC-reports [88]. Many fundamental problems from natural sciences deal with complex systems. We define a complex system as a population of unique elements with well defined attributes. In the case that these elements have non-linear interactions in the temporal and spatial evolution of the system, from these microscopic interactions a macroscopic behavior can emerge. This emergent behavior can, in general, not be predicted from the individual elements and their interaction. A typical example of emergent behavior is self-organization, e.g. Turing patterns in reaction- diffusion systems. These problems are often irreducible[2] and can not be solved in an analytical way. The only available option to obtain more insight in these systems is by explicit simulation. Moreover many of these problems are intractable: in order to obtain the required macroscopic information extensive and computationally expensive simulation is necessary. Since simulation models of complex systems require an enormous computational effort, the only feasible way is to apply massive parallel computing techniques to these models. A major future challenge is to apply High Performance Computing in research on complex systems and, in addition, to offer a parallel computing environment easily accessible for applications[99][98].

Traditionally, science has studied the properties of large systems composed of basic entities that obey simple microscopic equations reflecting the fundamental laws of nature. These entities could be fluid fields in fluid dynamics, gravitating galaxies in astrophysics, or gluons in lattice gauge theory. These natural systems may be studied by computer simulation in a variety of ways. Generally, the first step in any computer simulation is to develop some mathematical model consisting of a finite number of discrete parts. The correspondence between the discrete components and the natural system is completely arbitrary. Often, the discretization is arrived at indirectly, by means of partial differential equations. An alternative, less widely used approach is to develop solvers that conserve the characteristic intrinsic parallel properties of the applications and that allow for optimal mapping to a massive parallel computing system. These solvers have the properties that they map the parallelism in the application via a simple transformation to the parallelism in the machine. With these simple

[1]Report on 'high performance computing and communications' can be obtained via: Committee on Physical, Mathematical and Engineering Sciences c/o NSF. Computer and Information Science and Engineering 1800G Street NW Washington DC 20550

[2]Irreducible problems can only be solved by direct simulation

transformations the necessity to express the application into complex mathematical formulations becomes obsolete. One example would be the modeling of a fluid flow. Traditionally this problem is simulated through mathematical description of the phenomena via Navier Stokes equations and discretisation of these equations into numerical constructs for algorithmic presentation on a computer. This process of simulation involves a number of approximations and abstractions to the real fluid flow problem. Moreover, in due course of the simulation model the intrinsic properties and explicit information of the physical phenomena is obscured. Even worse, the possible implicit parallelism of the problem becomes completely indistinct in the abstraction process. An alternative approach would be to model the microscopic properties of the fluid flow with cellular automata, where the macroscopical processes of interest can be explored through computer simulation. This approach has the advantage that the physical characteristics of the fluid flow problem remain visible in the solving method and that the method conserves the parallelism in the problem. Although these type of simulation methods are not yet completely understood and certainly not fully exploited they are of crucial importance when massive parallel computers are concerned. We define these type of solvers as "natural solvers" these techniques have in common that they are inspired by processes from nature [100]. Four important examples of natural solvers are Genetic Algorithms (inspired by the process of natural selection), Simulated Annealing (inspired by the process of cooling heated material which converges to a state of minimal energy), the Lattice Boltzmann method (a many particle system with a macroscopic behavior that corresponds to the hydrodynamic equations), and artificial Neural Networks. We argue that in parallel computing the class of natural solvers results in a very promising approach, since the physical characteristics of the original physical phenomenon remain visible in the solving method and the implicit and explicit parallelism of the problem remain conserved.

In Fig. 1 a 'bird's eye view' of the different steps of the mapping process from application onto parallel machines is presented. As can be seen, an application is first transformed into a solver method. Here detailed knowledge of the problem domain is obligatory. A solver can have a different nature and one particular problem can be represented by different solver methods (like the example of the fluid flow). Next the intrinsic parallelism in the solver is passed through the Decomposition layer that captures the parallelism and dependencies into objects and communication relationships. Finally these two classes are mapped onto a Virtual Parallel Machine model that allows for implementation on a large suit of parallel systems [78].

To be able to capture the generic aspects of parallel solvers and to express the basic properties of the natural system we will define our own abstract solver model indicated as the Virtual Particle model. The Virtual Particle (VIP) can be defined as the basic element in the simulation model. The VIP can be defined on several levels of abstraction. For example in a simulation model of a biolog-

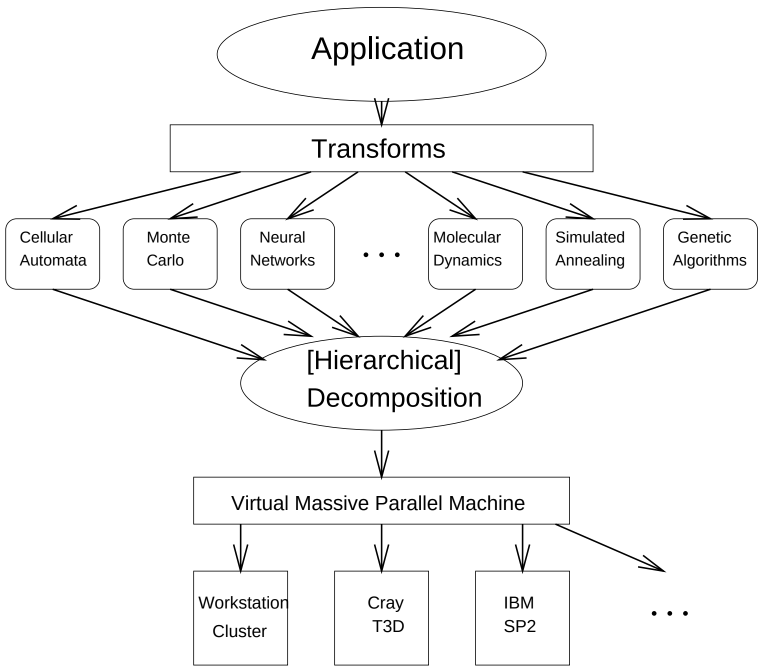


Figure 1: *Outline of a Parallel Programming Model for Dynamic Complex Systems on Massive Parallel Computers*

ical system, the VIP can correspond to a certain level of organization and aggregation in the system (e.g. molecule-organelle-cell -tissue-organ- organism-population). The choice of the abstraction level is determined by a combination of the desired refining of the model and the computational requirements. In the VIP model the microscopic, temporal or spatial, rules have to be specified in such a way that they approximate the microscopic rules as observed in the actual system. In the VIP model, the VIPs may correspond to the individual particles in the natural solver, as for example in lattice gases. Alternatively, the particles can be organized hierarchically, where VIPs can be an individual particle or clusters of VIPs. In such a hierarchical model interactions can occur between individual VIPs, clusters of VIPs, and individuals and clusters. An example where clustering of VIPs is applied is in N-body problems, where the N^2 number of long-range interactions between the particles can be approximated by $NlogN$ interactions through the use of hierarchical tree methods[3] [6] [38]. In the hierarchical algorithm the overall complexity of the problem is reduced, which allows for the simulation of relatively larger problems, while the information about the error introduced by this approximation is preserved. By allowing to cluster VIPs into a new VIP, it becomes possible to develop an efficient abstraction and consequent mapping of the application model onto the Virtual Parallel Machine model.
The natural system is represented by a DCS, the application model, where the individual elements are the VIPs whose interaction is possibly described by (random) graphs. The application model is mapped onto the Virtual Parallel Machine Model (see Fig. 2), which can be another instance of a DCS consisting of a population of processors, in this case both load-balancing and minimalisation of communication can be taken into account in a graph representation. In section 3.1 we will discuss this in detail, in section 4.4 we will demonstrate this by a case-study of mapping a finite element model onto a parallel machine model.

In the Background section of this paper we will review the physical requirements for computability and give some examples of intractability and undecidability in physical systems. We will use these notions througout the document. Then we will outline a theoretical framework for mapping parallel appications on parallel architectures. In the final sections we will discuss four case studies of modeling Dynamical Complex Systems. In the first example we show two alternative ways of modeling Diffusion Limited Aggregation: the "traditional" way by solving the underlying PDE, the Laplace equation and a VIP model, based on a multi particle system. In the second case study we will demonstrate how fluid flow and diffusion can be modeled using the VIP model based on the Lattice Boltzmann method. In the third example we will show how a natural solver, the genetic algorithm, can be casted into a VIP model, by using a cellular automata representation. In the fourth case study we will show the example of a finite element model where the application model consists of a population of clusters with finite elements, while the machine model is a set of

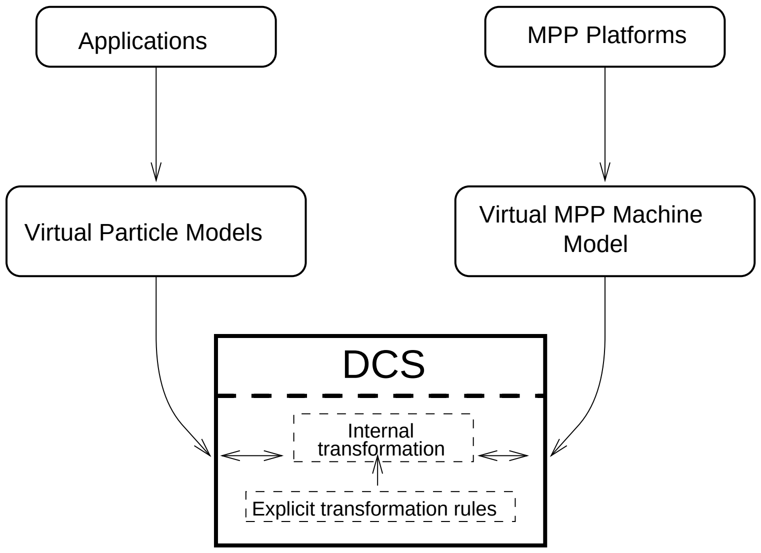


Figure 2: *Basic structure of Dynamic Complex System Paradigm: the mapping of the application model onto the machine model. The internal transformation denotes the mapping of the application graph onto the machine graph*

processors where the workload changes dynamically. The major issue in this section is to minimize the communication and to attain load balancing, where the most optimal mapping of the DCS with VIPs onto the DCS with processors has to be determined. In the conclusions we will summarize current and future research on DCS within the Parallel Scientific Computing and Simulation Group.

2 Background and Review of Concepts

2.1 Physical Requirements for Computability

In the appendix we identify systems which display all kinds of complex behavior, i.e. they can be computationally intractable or even formally undecidable. A classification in terms of their computational complexity is given and we discuss NP-complete, PSPACE-complete and P-complete problems. A question to ask at this point is: "When does a system display complex behavior?". By some authors[113][57][15] it is believed that when a system display complex behavior, universal computations can be performed. Mechanically speaking a computational system requires *transmission*, *storage* and *modification* of information. So, whenever we identify those three components in some dynamical system, the system could be computationally universal. But then the question remains when does this happen? Broadly said, using information theoretic results, we

can answer this question by saying that it must take place at an intermediate level of entropy: stored information, lowers the entropy, but transmission of information increases the entropy level. Therefore we will briefly review entropy measures of Cellular Automata in section 2.1.1. In a number of papers, Christopher Langton has tried to answer this question by considering Cellular Automata as a theoretical model for a physical system. The hypothesis "Computation at the Edge of Chaos" resulted from this research. Briefly it states that universal computations can take place at the border between order and chaos. This statement partly resulted from the observation that correlations can become infinite during or at a *second order phase transition* between for example a solid and a liquid phase. Recall that a discontinuous change in an order parameter of the system corresponds to a first order transition. A sudden, but continuous, change corresponds to a second order transition. At such a transition, the system is in a *critical* state. By some authors[57][50] it is believed that the resulting infinite correlations can be interpreted as long-term memory needed to store information. We will review these notions briefly in section 2.1.2.
Some non-equilibrium systems can display critical behavior without external parameter tuning. This critical behavior is analogous to the behavior of a equilibrium system at second order phase transitions, i.e. no characteristic length scales can be found. Systems with critical behavior as an attractor, are reviewed in section 2.1.3.

2.1.1 Information Theory in CA

In order to observe phase transitions in CA evolution, quantitative order parameters are needed. These order parameters need to distinguish between ordered and disordered states. A commonly used quantity for this purpose is the Shannon entropy[96], defined on a discrete probability distribution p_i:

$$H = -\sum_i p_i log p_i = \sum_i p_i log \frac{1}{p_i} \tag{1}$$

This measure H can be associated with the degree of uncertainty about a system. In a Cellular Automata, entropy can be defined on the k^N possible subsequences of N-length blocks in a k-state CA. In a random sequence all subsequences must occur with equal probability. With probabilities p_i for the k^N possible subsequences:

$$H_N = -\sum_{i=1}^{k^N} p_j log p_j \tag{2}$$

The *spatial block entropy*[37] is now defined as:

$$h_N^{(x)} = H_{N+1} - H_N \tag{3}$$

From Eq.3 follows the *spatial measure entropy*, or *entropy rate* [37]:

$$h^{(x)} = \lim_{N\to\infty} h_N \tag{4}$$

The superscripts (x) indicate that spatial sequences are considered. The measure entropy gives the average information content per site. Analogous to the spatial entropy, one can define *temporal entropy*, where blocks of $N \times T$ sites are considered:

$$h^{(t)} = \lim_{N,T \to \infty} h^{(t)}_{N,T} \tag{5}$$

Eq.3 decreases monotonically with N, while $h^{(t)}_{N,T}$ decreases with T. The difference,

$$\delta h^{(x)}_N = h^{(x)}_N - h^{(x)}_{N+1} \tag{6}$$

is the amount of information by which a state s_{i+N} of a cell $i + N$ becomes less uncertain if the cell state s_i gets known. $\delta h^{(x)}_N$ is called the N-th order mutual information in space. Intuitively one could regard mutual information as the stored information in one variable about another variable and the degree of predictability of a second variable by knowing the first. Another definition of mutual information, block-to-block mutual information, is defined as the mutual information between two L-blocks[59]. Let P_α be the probability for an L-block and $P_{\alpha\beta}(d)$ be the joint probability for two blocks separated by a distance d.:

$$\begin{aligned} M(d)^{[L]} &= \sum_\alpha \sum_\beta P_{\alpha\beta}(d) log \frac{P_{\alpha\beta}(d)}{P_\alpha P_\beta} \\ &= \sum_\alpha \sum_\beta P_{\alpha\beta}(d) log P_{\alpha\beta}(d) - \sum_\alpha \sum_\beta P_{\alpha\beta}(d) log P_\alpha log P_\beta \end{aligned} \tag{7}$$

where k is the number of states in a configuration of a CA or other symbolic sequence. Note that $\sum_\alpha \sum_\beta P_{\alpha\beta}(d)$ reduces to P_β and mutatis mutandis for summing over P_β. Hence the Eq.7 can be simplified to:

$$\begin{aligned} M(d)^{[L]} &= \sum_\alpha \sum_\beta P_{\alpha\beta}(d) log P_{\alpha\beta}(d) - \sum_\alpha P_\alpha log P_\alpha - \sum_\beta P_\beta log P_\beta \\ &= H_\alpha + H_\beta - H_{\alpha\beta} \end{aligned} \tag{8}$$

Where $H_{\alpha\beta}$ is the joined entropy of sequences α and β.

2.1.2 Computation in the intermediate regime of ordered and disordered behavior

If we have a k-state CA with a neighborhood size r, the total number of possible transition rules is k^{k^r}, which can become a very large, even for a moderate number of states and/or a small neighborhood. If a structure were present in this enormous space, it should be possible to identify areas of equal complexity (Wolfram classes, see section 6.4) and how these areas are connected to each other. Using this ordering one can locate those areas which support the transmission, storage and modification of information. Langton[57], suggested the λ parameter to structure the CA rule-space. An arbitrary state $s \in \Sigma$ is assigned the *quiescent state* s_q. Let there be n transitions to this quiescent state in an arbitrary transition rule. The remaining $k^r - n$ transitions are filled randomly by picking uniformly over the other $k-1$ states:

$$\lambda = \frac{k^r - n}{k^r} \tag{9}$$

If $\lambda = 0.0$ then all transitions in the rule will be to the quiescent state s_q. If $\lambda = 1.0$ there will be no transitions to s_q. All states are represented equally in the rule if $\lambda = 1.0 - 1/K$. With the aid of the λ-parameter it should be possible to examine the conjecture that *complex behavior* is located at intermediate regime between ordered and disordered behavior. The spectrum of dynamical behavior can be explored with the so called *table-walk-through-method* which increases the λ-parameter at successive time steps. At each new time step a transition table is incrementally updated using the transition table at the previous time step. Because the described method is actually a "random walk" through a coarse grained version of the CA state-space, each table-walk displays quantitatively different behavior. Several measures can be used to characterize the dynamical behavior of the CA at each new value of the λ-parameter. These measures include the numerical determination of block entropies and both temporal and spatial mutual information statistics. At intermediate values of λ, i.e at the edge between ordered and disordered dynamics several remarkable events occur:

- Transient lengths grow rapidly, analogously to the physical event of *critical slowing down*.
- Transient lengths depend exponentially on the size of the CA.
- Mutual information measures (see Eq. 8), reach their maximum values (both spatial and temporal mutual information), see Fig. 3(left) at the entropy transition, see Fig. 3(right)

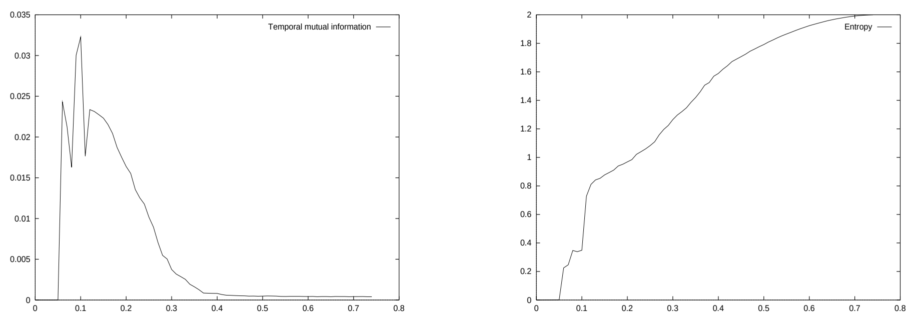


Figure 3: *Temporal mutual information between two sites separated by one time step and site entropy, both for 4-state 2-neighbor Cellular Automata*

The exponential dependence of transient lengths on the size of the CA is analogous to the exponential dependence on problem size in the NP and PSPACE complexity classes. As for the halting-computations, it will be formally undecidable for an arbitrary CA in the vicinity of a phase transition, whether transients will ever die out. The increase in mutual information indicates that the correlation length is growing, which implies further evidence for a phase transition in that region. Of course we cannot observe a real phase transition other than in the thermodynamic limit. Other evidence for the "Edge of chaos" hypothesis can be found in the work of Crutchfield on continuous dynamical systems[14] and the resulting ϵ-machine reconstruction. In [16] the so called *intrinsic computation* abilities of a continuous dynamical system are investigated. The output of the system (e.g. an iterative map, $x_{n+1} = f(x_n)$) in time is coarse grained into a sequence of zeros and ones. In other words the output domain x_n is divided into two regions, $P_0 = \{x_n < x_c\}$ and $P_1 = \{x_n \geq x_c\}$, where x_c is an arbitrary chosen division point. The complexity of the dynamical system is quantified by construction of the minimal regular language which accepts the generated sequence. The complexity and entropy (see Eq. 1) for the logistic map was examined in [16] using the method of regular language complexity (size of the corresponding finite automaton). It was found that the lowest values of complexity corresponds to the periodic and fully chaotic regimes of the map. The highest value of the complexity occurs where the period doubling cascade of the map meets the band-merging cascade, i.e. at the border between order and chaos. In [23] the work of Langton and Crutchfield is complemented by examining the dynamical behavior of a well known computational device: The Turing Machines (TM). A class of 7-state 4-symbol Turing machines, which also includes Minsky's universal Turing machine[72], was used to address the question whether universal computation is found between order and chaos. A large number of randomly created TM's was used to generate three different sequences: a sequence of symbols read, a sequence of states and a sequence of moves made by the TM head. For all these sequences the corresponding regular language complexity was calculated using the technique of ϵ-machine reconstruction and plotted against its block-entropy (see Eq. 3). They found that the most complex TM's are indeed located at intermediate values of the entropy, including Minsky's universal TM. Mitchell et al, reviewed this idea of computation at the 'edge of chaos' and reported on experiments producing very different results from the original experiment by Packard [80], they suggest that the interpretation of the original results is not correct [73].

2.1.3 Computational Behavior as an Attractor: Self Organized Criticality

Most of the time, equilibrium systems with short-range interactions, exhibit exponentially decaying correlations. Infinite correlations, i.e. scale invariance, can be achieved by fine tuning some parameters (temperature) to a critical value. An example of such a system is the Ising spin model of magnetization. On the other hand, a large class of non-equilibrium locally interacting nonlinear systems spontaneously develop scale invariance. These systems are subject to external driving: local states are increased from outside until some threshold condition holds. At the point where the critical value is reached the local state variables trigger a chain reaction by transporting its "stored energy" to neighboring sites. At the steady state of the system, assured by open boundary conditions, chain reactions or avalanches of all sizes can occur. The distribution of "avalanche sizes" obeys critical scaling:

$$P(s) \sim s^{-\tau} \tag{10}$$

where τ is a critical exponent and the state of the system has no intrinsic time or length scale. The size of an avalanche can be defined in different ways. It can be measured by the number of relaxation steps needed for the chain reaction to stop or the total number of sites involved in the avalanche. This phenomenon of spontaneously developing critical scaling has been called Self Organized Criticality (SOC)[5]. A lot of naturally occurring systems exhibit this kind of scaling- or self-similar behavior. The basic model with which SOC behavior is demonstrated, is with the use of a special kind of Cellular Automaton, which simulates the sliding and growing of sand piles[5]. A discrete variable h is assigned to each site on a d-dimensional lattice, representing the height of a "slope". Units of sand are added subsequently to a random site until some critical threshold h_c is reached. If the state of a site has reached the critical value, a sliding event takes place: the sand of the site is distributed among its nearest neighbors. The rules for a d-dimensional version of such a model are:

$$h(\mathbf{r}, t+1) \rightarrow h(\mathbf{r}, t) - 2d \tag{11}$$

$$h(\mathbf{r} \pm \mathbf{e_j}, t+1) \rightarrow h(\mathbf{r} \pm \mathbf{e_j}, t) + 1 \tag{12}$$

where $\mathbf{r}$ is a lattice position, $\mathbf{e_j}$ is used to denote the j-th neighbor, and t is the iteration counter. The rules actually describe a non-linear diffusion equation. If the system has relaxed, another random perturbation can made. We have repeated the experiments with a continuous version of this model where randomly chosen energy quanta ϵ_h ($[0, 1]$)are added to a randomly chosen site. If some site reaches the critical energy value 1.0, the system is relaxed until every site is below this critical value. At some point the system reaches a stationary state where the average energy has reached a steady value $< h >$. At this point, the addition of an amount of energy to some site can trigger avalanches of arbitrary sizes, i.e. from one shift to one comparable with the system size. From this point we measured the distribution of both the duration and size of the avalanches. Figs. 4 and 5 show the distribution of avalanche size and avalanche duration respectively. It is clear that the finite size distribution can be fitted by a function obeying a power law. The idea of Self-Organized Criticality is assumed to be an explanation of the emergence of critical scaling behavior in many naturally observed systems. If this scaling behavior is spatial the signature of the system is often a fractal. Many growth phenomena exhibit this self-similar behavior, e.g. DLA, Dielectric Breakdown (DB), Invasion Percolation, etc. Hence a logical step is to use SOC as a possible theory of the

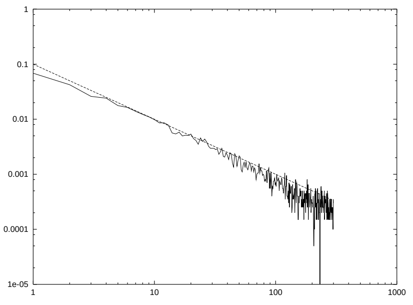


Figure 4: *Avalanche size distribution in a 100x100 lattice*

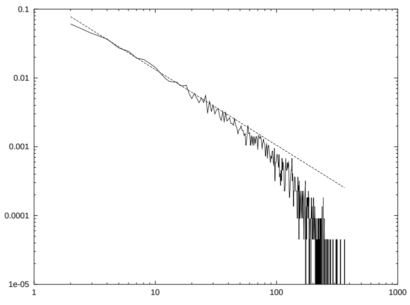


Figure 5: *Avalanche time distribution in a 100x100 lattice*

growth of fractal patterns. The idea that a dynamical system with spatial degrees of freedom evolves into a self-organized critical state could very well be applied to these familiar models of growth. Some work in this area has been done on the DB model[2][81][82]. Exactly at the point where extinction of a branch balances branching itself, the growth process is stable with respect to fluctuations. A stationary state is reached when branching has broken down to a level where the flow only barely survives[2]. The distribution of the sizes of extinct branches describe the self-organized critical state, corresponding to the avalanches in the sandpile. Another way of characterizing the SOC state is by keeping track of the electrostatic field and the number of iterations required to reach relaxation[81] after a growth step. The relaxation steps of a numerical procedure, e.g Jacobi iteration, can be used to quantify this measure. Alternatively one could measure the range of the disturbance, defined as the total number of lattice sites in which the electrostatic potential changes above a given threshold. If the growth is in the SOC regime, the range distribution can be described by a power-law function.

2.2 Examples of Intractability and undecidability in some Physical Systems

In this section we will discuss models of three different complex systems, namely an *optical system*[86], a *N-body system*[85] and *Non-equilibrium growth*. It can be shown that these systems exhibit undecidable and intractable behavior concerning their predictability. We will shortly describe these systems and their related hardness proofs. For a thorough description we refer to [86], [85], [48] and [95].

2.2.1 Optical Beam Tracing

First the optical system, which also forms a basis for the hardness proofs of the N-body system. In [86] the "Computability and Complexity of Optical Beam

Tracing" is explored. The *Ray Tracing Problem* (RTB) is stated as:

> *Given an optical system (a finite set of reflective or refractive surfaces) and an initial position and direction of a light ray and some fixed point* p*, does the light ray eventually reach the point* p*?*

We summarize the computability results for different optical models, for more detail see [86]:

1. In a three dimensional system which consists of a finite set of mirrors, half-silvered mirrors and quadratic lenses, RTB is undecidable. The proof remains valid for rational coordinates.

2. In a three dimensional system which consists of just a finite set of mirrors with irrational endpoint coordinates, RTB is undecidable. When the endpoints are restricted to rational coordinates the problem is PSPACE-hard.

3. For any system with dimension ≥ 2, which consists of a finite set of mirrors with rational endpoints which lie perpendicular to each other, RTB is in PSPACE. For dimensions ≥ 3 the problem is PSPACE-complete.

By showing that the total path of the light ray can be divided into sub paths and their corresponding rational equations, all models can be shown to be recursively enumerable. Hence the total path can be traced. In order to prove undecidability, an optical model must be able to mimic an unrestricted Turing Machine. This can be achieved by coding the contents of the tape by (x, y) coordinates of the position of the light ray. The states of the TM are "coded" by basic optical boxes (a specific set of mirrors and lenses), which implement the transition function δ for a particular state. The mapping of δ onto the operation of an optical model is done by simulating the two basic different types of transitions:

$$\delta(q, c) = (q^{'}, w, L) \tag{13}$$

$$\delta(q, c) = (q^{'}, w, R) \tag{14}$$

Where L and R represent a left and right move respectively, q and $q^{'}$ correspond to TM states, c to the read symbol, and w to the written symbol. The basic boxes implement the various transition functions, by manipulating the light ray through the use of mirrors and lenses, which in turn implement the functioning of an entire TM. Undecidability of RTB using some optical model can be proven by showing that an unrestricted TM can be mimicked. PSPACE-hardness can be proven by showing that some space bounded TM can be mimicked.

2.2.2 N-body Simulation

A second physical model to which a hardness proof is applied, is *N-body simulations*[85]. The *N-body simulation problem* is stated as follows:

Given initial positions and velocities of n particles that have pair-wise force interactions, simulate the movement of these particles so as to determine the positions of the particles at a future time.

The problem of determining whether a specific particle will reach a certain region at some specified target time is called the *N-body reachability problem*. The equations of motion for each body is given by Newton's laws of motion which constitute n ordinary differential equations. The corresponding solutions can be approximated by time stepping. The initial positions and velocities of the bodies are given by n^k-bit rational. The destination position is given by a ball, which is an n-bit rational (the ratio of two n-bit integers). A related result can be found in [29], where the "billiard ball computer" is introduced and which is proved to be PSPACE-hard. However this model depends on non-movable obstacles as does the optical model stated above and hence is not applicable to n-body simulation.
The hardness proof is done by constructing an analog of optical beam tracing in the N-body system. It is assumed that initial positions, velocities and position of the destination ball are rational. The global sketch of the proof is by reducing several problems to each other, starting with a known PSPACE-hard problem, namely that of optical ray tracing with reflective (only mirrors) surfaces and ending with the n-body simulation problem by subsequently reducing subproblems. For a detailed overview we refer to [85].

2.2.3 Non Equilibrium Growth Processes

In equilibrium growth processes, as for example found in a perfect crystal where the growth process is near or in equilibrium, molecules are exploring various sites of the crystal and are added to the crystal until the most stable configuration is found. In this type of growth process a continuous rearrangement of particles takes place, the process is relatively slow and the resulting objects are very regular [92]. In some cases the growth form which emerges is a normal object from Euclidean geometry whereas in other cases objects are formed that resemble regular fractal objects.
Many growth processes in nature are not in equilibrium. An extreme example is an aggregation process of particles where as soon as a particle is added to the growth form, it stops trying other sites an no further rearrangement takes place. The local growth probabilities are not everywhere equal on the aggregate and an instable situation emerges. The growth process in non-equilibrium is relatively fast and often irregular objects, characterized by a fractal dimension, are formed [91] [92] [90]. An example of a (non-equilibrium) growth process from physics, is viscous fingering. The phenomenon can be demonstrated in an experiment where air displaces a high-viscosity fluid between two glass plates. In Fig. 6 a diagram is shown of an experiment where air is injected between two glass plates at $y = 0$ and displaces a high viscosity fluid, which is only removed at the top of the plates (both sides are closed). The pressure P will be the highest at $y = 0$ and the lowest at $y = L$, where L represents

the length of the glass plates. In the fluid the pressure is given by the Laplace equation [25]:

$$- \nabla^2 P = 0 \tag{15}$$

In the air the pressure is everywhere equal, since its viscosity can be ignored. The pressure in the air equals to the input pressure $P(y = 0)$ and the consequence is that the largest pressure gradients occur at the tips of the fingers in Fig. 6, while the lowest gradients occur below the tips. The probability that the fingers continue to grow will be the highest at the tips and in a next growth stage the pressure gradients in the tips are still more amplified, resulting in an instable situation. In Fig. 7 an example of the resulting growth pattern is shown, it is an irregular shaped object, known in the literature as viscous fingering.

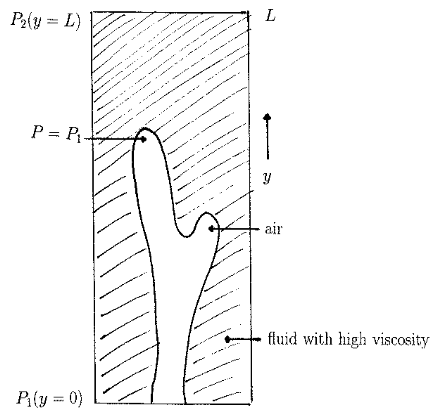


Figure 6: *Diagram of a viscous fingering experiment*

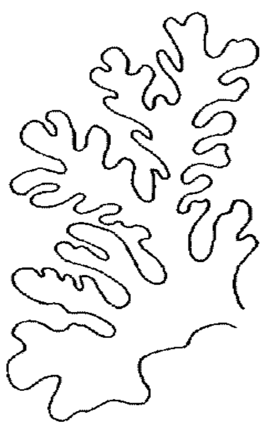


Figure 7: *Example of a viscous fingering growth pattern*

Another example of growth in non-equilibrium is growth of a bacteria colony (for example *Bacillus subtilus*) on a petri-dish [32]. The colony consumes nutrients from its immediate environment and the distribution of nutrients is determined by diffusion. When it is assumed that the concentration c is zero at the colony and that the diffusion process is fast compared to the growth process, the concentration field will attain a steady state in which the diffusion equation

$$\frac{dc}{dt} = \mathcal{D} \nabla^2 c \tag{16}$$

equals zero. In this equation $\mathcal{D}$ is the diffusion coefficient. The nutrient source may be, for example, a circle around the colony or a linear source where the concentration is maximal. The local nutrient concentration at sites between the colony and the source can be described with the Laplace equation:

$$\nabla^2 C = 0 \tag{17}$$

The growth process of a bacteria colony, viscous fingering, and various other growth patterns from physics as for example electric discharge patterns and growth forms of electro deposits, can be simulated with one model: the Diffusion Limited Aggregation model [91]. At the heart of all these growth patterns there is one Partial Differential Equation, the Laplace equation, which describes the distribution of the concentration (Eq. 17), pressure (Eq. 15), electric potential etc. in the environment of the growth pattern. The DLA-model is a probabilistic cellular automaton which resides on a square two-dimensional or three dimensional lattice. In section 4.1 various methods will be discussed for constructing DLA-clusters, an example of a DLA-cluster is shown in Fig. 27. The underlying Laplace equation can be solved numerically and a DLA cluster can be constructed using the nutrient distribution over the lattice. The cluster is initialized with a seed and the following boundary conditions are applied: $c = 0$ on the cluster itself and $c = 1$ at the nutrient source, which may be circular, linear etc. The probability p that a perimeter site with index k will be added to the DLA-cluster is determined by

$$p(k \in \text{perimeter sites} \rightarrow k \in \text{cluster sites}) = \frac{(c_k)^\eta}{\sum_{j \in \text{perimeter sites}} (c_j)^\eta} \tag{18}$$
$$\text{where } c_k = \text{ concentration at position } k$$

The exponent η applied in Eq. 18 describes the relation between the local field and the probability. This exponent usually ranges in experiments from 0.0 to 2.0. The sum in the denominator represents the sum of all local concentrations of the possible growth candidates. The probability that new sites will be added to the cluster will be the highest at the tips of the cluster, where the steepest nutrient gradients occur, and the lowest in the bays between the branches. In successive growth steps the nutrient gradients at the tips will even become steeper and a comparable instable situation is encountered as in the viscous fingering example. The effect of changing the exponent η in Eq. 18 is that the overall shape of the cluster changes. For the value $\eta = 0$ the shape changes in a compact cluster and it can be demonstrated that the DLA-model for this special case transforms into the Eden model. This model is one of the earliest probabilistic cellular automata to simulate growth. In the Eden model each possible growth candidate has the same probability to become occupied. For the value $\eta = 1$ the normal DLA-cluster is obtained, while for higher values more dendritic shapes are generated [68]. With the parameter η the effect of nutrient gradients on the growth process can be controlled, where the Eden model is an extreme example in which gradients have no effect on the local probability that a new site will be added to the growth form.
In order to model and simulate different natural growth processes, a variety of growth models have been suggested, each with different characteristics. Most

of these models display scaling or fractal behavior, analogous to characteristics observed in second order phase transitions. Examples of these models are: DLA, Eden growth, invasion percolation, and ballistic deposition. From a computational point of view these models can be divided in two classes known from computational complexity theory: NC and P (see section 6.3).
As usual complexity classes are defined for decision problems, so we in order to talk about the computational complexity of growth models, they have to be formulated as such. Consider a cluster growing on a lattice of N sites. The decision problem can be divided into N separate decision problems, corresponding to whether the N sites of a lattice will eventually be occupied or not by the spreading cluster. Through the equivalence of a hydrodynamic model[55] it can be shown that DLA also P-complete[60], hence DLA can be regarded as a universal computer for problems in P. On the other hand the Eden model and invasion percolation can be formulated as *waiting time models*[87] and as such can be proven to be in the class NC[62].
Let us first consider the equivalence between DLA and the Chamber-Tube (CT) model[55]. An approach to model two-fluid flow (an in-viscid fluid driving a viscous one) in a porous medium is to regard the porous medium as a system of chambers connected by tubes. The pressure of the viscous fluid driven by the invading fluid, satisfies a Laplace equation:

$$\nabla^2 p = 0$$

Basically this is the CT model, which can be shown to be equivalent to DLA. In [60] it is shown that this CT model is P-complete, from which a restricted planar version of this CT model is also shown to be P-complete. Intuitively the DLA model is inherently history-dependent, i.e. the growth at a given time step depends in the prior history of the system (Markovian). Because of this property, the DLA model does not lend itself for *exact* efficient parallelization. It is however possible, to parallelize the computational steps between two growth events and also to some extent the growth steps themselves[95]. Apparently also Eden models suffer from history dependence, at each time step a particle is added randomly to the perimeter of the cluster. In [62] it is shown that this history dependence can be overcome. In [87] a mapping is made between the Eden model and the growth of directed polymers in random media. The *directed polymer problem* is defined in a random medium, at zero temperature. To each site i of a lattice, a random number x_i from some probability distribution is assigned. This random number corresponds to the local interaction energy between the polymer and the embedding medium at this site. A directed polymer is a directed path P that spans the lattice. The energy of the polymer is the sum of the x's along the path P. At zero temperature the polymer will be in the lowest energy configuration, such that E will be:

$$E = min_P \left\{ \sum_{i \in P} \right\} \tag{19}$$

Subsequently an equivalence with Eden growth can be made. Set a clock on each site of the lattice and record the time during which the site has been a potential growth site, without being actually part of the cluster. Note that eventually all sites will be part of the Eden cluster. Call this delay τ_i for site i. From these delay times, the real time t_i at which the site became part of the cluster can be computed: it is equal to the time at which i became a neighbor to the cluster plus the time τ_i. The time ${t_i}^i$ at which the neighbor becomes part of the cluster is the minimum of the sum of delays to this neighbor:

$$t_i = min_{P_i} \left\{ \sum_{j \in P_i} \tau_j \right\} \tag{20}$$

The resemblance between Eq.20 and Eq.19 is obvious. The time should be compared with the minimum energy of a polymer. This so called waiting time model can in turn be mapped onto a Minimum-Weight Path Algorithm, which is known to be efficiently parallelisable. MWP is defined as:

> An undirected grap $G = (V, E)$, where V is a set of sites and E is a set of bonds connecting pairs of sites. Weights $w(i, j)$ are assigned to each bond $\{i, j\} \in E$. The problem is to find a matrix containing weights of the minimum-weight paths between every pair of sites in V.

It possible to describe both Eden and DLA with the same equation[82], only differing in one parameter. The growth is determined by a scalar field ϕ, which obeys the Laplace equation outside the lattice. The velocity of growth depends on the gradient of this field raised to a power:

$$v \propto |\nabla \phi|^\eta . \tag{21}$$

When $\eta = 1$ DLA is recovered, and when $\eta = 0$, the Eden model is obtained. It follows that, though both DLA and Eden can be described by the same equations, they are not equal in computational complexity. This transition to another complexity class can be established by changing this η parameter. It is interesting to known whether general properties and conditions for this difference in time complexity between growth models can be found.

3 Theoretical Framework

3.1 Parallel Task Allocation

3.1.1 Introduction

An essential problem in the field of parallel computing is the so called *Task Allocation Problem*(TAP): given a set of parallel communicating tasks (a parallel application) and a parallel distributed memory machine, find the optimal allocation of tasks onto the parallel system. The quality of an allocation is measured by the turn-around time of the application, which depends on various

components. In a parallel application, generally, one can distinguish phases dominated by communication components and calculation components.
A method that is used to minimize the turn-around time must optimise both components simultaneously. This is due to the fact that the two terms can not be regarded as independent components, rather they are strongly related to each other. A task allocation where all tasks are placed on a single processor obviously minimizes the amount of communication, while the calculation will be maximal. On the other hand equal distribution of the set of parallel tasks, without taking into account the communication term would lead to an optimized calculation term, while communication can become degraded. We toss the term *frustration* for the fact that optimization of one term conflicts with the other, in analogy to physical systems that exhibit frustration. Intuitively, it is clear that increasing dominance of either the communication or calculation term reduces the amount of frustration in the system.

Many fundamental problems from natural sciences deal with *complex systems*. A complex system can be described as a population of unique elements with well defined attributes and interactions. In most cases, such systems are characterized by *quenched* disorder and *frustrated*, non-linear interactions [71], between the set of elements constituting the system. It is well known that these system ingredients contribute to the emergent unpredictable behavior that is often demonstrated by such systems [97]. The quenched disorder is either present in the initial condition (e.g. in cellular automata) or in the interaction between elements (e.g. in spin glasses). In combination with the frustration occurring due to mutual conflicts, certain properties of these systems are often analytically intractable. Examples of such properties are its asymptotic behavior and the exact location of the (energetically) optimal states. The latter characteristic often causes the corresponding optimization problems to be NP-hard [101]. Given the fact that the TAP objective function (minimization of turn-around time) contains two competitive terms, behavior similar to other known *complex systems* is to be expected.
In order to deepen our knowledge about the TAP, we intend to explore its characteristics in terms of phase space and optima structure. Specifically, the degree of frustration in the TAP constitutes a fundamental difficulty with the problem. An important distinguishing aspect in the TAP is the presence of a transition from sequential to parallel optimal allocation. For example, consider a parallel machine topology consisting of identical processors, with a tunable performance rate. Increasing the peak performance continuously from 0 flop/s to ∞ flop/s, will induce a transition from optimal parallel- to sequential allocation, given a finite communication speed within the network. In analogy with other combinatorial optimization problems that exhibit frustration and phase transitions, we expect that a phenomenon, known as *critical slowing down*, can be observed in the transition region. That is, the difficulty of finding optimal solutions peaks near the transition region (see e.g. [111]).

In general, the selection of a suited heuristic method for finding (sub)-optimal solutions requires knowledge of the shape of the phase space. Great care has to be taken in selecting an optimization method, since searching the optimal solution to the TAP is known to be an NP-hard problem [8]. Hence, a study on the structure of the landscape of the TAP is necessary in order to identify effective optimization methods. Furthermore, the sensitivity of the TAP to a small set of machine and application specific parameters is investigated. We restrict our attention to a specific subset of the TAP. The focus will be on applications that can be described by static parallel task graphs. In addition we assume to have a static resource parallel machine that is homogeneous and fully connected.

This section is structured as follows. Section 3.1.2 introduces application and machine representations that are used to model the performance characteristics of parallel static applications on parallel machines. Section 3.1.3 gives a detailed study on the structure of the phase space (or landscape) of the TAP. Section 3.1.7 is dedicated to the geometrical phase transition occurring in the TAP. In section 3.1.11 the following experimental methods are presented: Simulated Annealing (SA) [53], for finding optima, and Weinberger correlation for phase space structure characterization [110]. In section 3.1.15 experimental results are presented, which are discussed in section 3.1.20. Finally, some concluding remarks and directions for future work are given in section 3.1.23.

3.1.2 Application and Machine Models

In order to facilitate our study on abstract parallel applications we introduce a random graph representation as a model of static communicating parallel tasks. Each task is assigned a workload and every pair of tasks (vertices) in the task graph is connected with a probability γ ($\gamma \in [0,1]$). A message size is assigned to each link between two communicating tasks. We restrict our attention to constant work loads and message sizes. Furthermore the target processor topology is assumed to be a static parallel machine that is fully connected and homogeneous. That is, communication channels between all processor pairs are bi-directional and have equal bandwidths. Moreover, the processors are homogeneous, i.e. they have identical constant performance.
The metric for deciding on the quality of a task allocation is the turn-around or execution time. A variety of cost models that are based on a graph representation can be found in literature. For example, the following cost function (22) [51], is known to model the actual execution time for a given task allocation with reasonable accuracy. Of course it is a simplification of the real situation, i.e. message latencies and network congestion are neglected.

$$H = \max_{q \in \mathcal{Q}} \left(\sum_{u_i \in \mathcal{U}^q} W_{u_i} S_q + \max_{u_i \in \mathcal{U}^q, u_j \in \mathcal{A}(u_i)} S_{pq} W_{u_i u_j} \right), \tag{22}$$

where

- u_i is a task in the parallel task graph

- $\mathcal{Q}$: the set of processors
- $\mathcal{A}(u_i)$: the set of tasks connected to task u_i
- $\mathcal{U}^q$: set of tasks u_i residing on processor q
- W_{u_i}: Work associated with task u_i (e.g. in terms of flop)
- S_q: $\frac{1}{processorspeed}$ for processor q (e.g. in s/flop)
- $W_{u_i u_j}$: Number of bytes to be sent, due to nodal connectivity, between host processor of task u_i and task u_j.
- S_{pq}: $\frac{1}{bandwidth}$ of route between processor p and q (in s/bytes)

A property of this specific function is that the execution time is determined by the "slowest" processor in the parallel machine. This cost function is a reasonable representation of the actual execution time.
Because the value of H (Eq. 22) can only change in task transfers that involve the slowest processor, it is not very sensitive to task rearrangements. Therefore it is unsuitable for local search optimization techniques like SA. Usage of SA for finding optimal solutions necessitates formulation of an alternative cost function like (23), see e.g. [67].

$$H = \sum_p W_p^2 + \mu \sum_{p \neq q} C_{pq} \quad , \tag{23}$$

where

- $W_p = A_p S_p$, with A_p: $\sum_{u_i \in \mathcal{U}^p} W_{u_i}$, total work on processor p in terms of flop.
- $C_{pq} = M_{pq} S_{pq}$, with M_{pq}: $\sum_{u_i \in p, u_j \in q} W_{u_i u_j}$.
- μ is a control parameter, expressing the communication/calculation ratio [26].

An incremental search from a given allocation (moving one task), requires a complete re-calculation of the cost for Eq. 22. On the other hand, Eq. 23 has the locality property, which means that incremental changes in a task allocation can be propagated into the cost without having to recalculate the whole cost function. Only a difference has to be calculated instead[28]. This is specifically useful if an optimization algorithm is applied that is based on incremental changes (e.g. SA), and as such can exploit the direct consequence of these increments. A disadvantage of using (23) is the fact that it is a not a correct model for the absolute cost. The objective is to minimize the variance in the workload distribution simultaneous with the communication volume of the whole system, opposed to optimization of the execution time of the slowest processor in Eq. 22.

3.1.3 Correlation Structure of the TAP

The configuration space C of the TAP consists of all possible task allocations of the n tasks to the P processor topology. A configuration can be encoded as a sequence of length n, which is composed of letters taken from the alphabet $\{1, 2, ..., P\}$. The index of a sequence letter corresponds to a task ID. The distance is given by the number of positions in which two sequences A and B differ; this metric distance measure is the *Hamming distance* [39] $d(A, B)$. The *Hamming graph* can be constructed by connecting every sequence pair (A, B) that has $d(A, B) = 1$.
The number of configurations with a given distance d from an arbitrary reference point $N(P, n, d)$, the total number of configurations #C, and the diameter in the configuration space, $diamC$ are easily found to be:

$$N(P, n, d) = \left(\begin{array}{c} n \\ d \end{array} \right) (P - 1)^d \tag{24}$$

$$\#C = P^n \tag{25}$$

$$diamC = n \tag{26}$$

A random walk through some landscape, can be used to characterize its structure [110]. For landscapes that are self-similar it is known that the corresponding random walk auto-correlation function is a decaying exponential, with correlation length λ. Such landscapes are classified as AR(1) landscapes and have been identified in various fields, e.g. (Bio)physics [110] and combinatorial optimization [103][104]. It has been shown that incremental search methods like Simulated Annealing perform optimally on landscapes that show a self-similar structure [102].
We will derive expressions for the relaxation and auto-correlation functions of random walks through the task allocation landscape. The relaxation functions indicate at what rate a random walk through the Hamming graph deviates from the starting point, analogous to e.g. relaxation of diffusion processes in physical systems.
The auto-correlation function is used to quantify the rugged-ness [110] of the landscape of the TAP. The landscape constitutes the Hamming graph with cost values that are assigned to all vertices according to Eq. 23. Using these expressions it is shown that the landscape is AR(1) with a correlation length that is linearly proportional to the number of tasks n.

3.1.4 Relaxation of random walks

The statistical properties of random walks on the graph C are completely contained in the probabilities ϕ_{sd}, where ϕ_{sd} denotes the probability that a random walk is within a distance d from the starting point after s steps. In general this probability distribution fulfills the following recursion relations on any distance transitive graph (following [103]).

$$\begin{aligned}\phi_{sd} &= a^+_{d-1}\phi_{s-1d-1} + a^0_d\phi_{s-1d} + a^-_{d+1}\phi_{s-1d+1} \\ \phi_{00} &= 1 \\ \phi_{sd} &= 0, \qquad \text{if} \qquad d > s\end{aligned} \tag{27}$$

The coefficients a^+_d, a^0_d and a^-_d denote the probability of making a step "forward", "side ward" and "backward", respectively, given the walk is within a distance d from the starting point. Therefore $a^+_d + a^0_d + a^-_d$ is equal to 1. For the TAP graph C we obtain the following expressions:

$$\begin{aligned}a^+_d &= \frac{(n-d)(P-1)}{nP} \\ a^0_d &= \frac{n+(P-2)d}{nP} \\ a^-_d &= d/(nP)\end{aligned} \tag{28}$$

Although we have no closed expression for the ϕ_{sd}, we can obtain some insight into the relaxation behavior of random walks from the expected values of the distance (first moment) and the squared distance (second moment) from the starting point after s steps along the walk:

$$\begin{aligned}\Delta_1(s) &= \sum_{d=0}^{s} d\phi_{sd} \\ \Delta_2(s) &= \sum_{d=0}^{s} d^2\phi_{sd}\end{aligned} \tag{29}$$

Using Eqs. 27 and 29, we can derive recursion relations for $\Delta_1(s)$ and $\Delta_2(s)$:

$$\begin{aligned}\Delta_1(s) &= \sum_{d=0}^{s} d(a^+_{d-1}\phi_{s-1,d-1} + a^0_d\phi_{s-1,d} + a^-_{d+1}\phi_{s-1,d+1}) \\ &= \sum_{d=0}^{s-1} \phi_{s-1d}(d(a^+_d + a^0_d + a^-_d) + (a^+_d - a^-_d)) \\ &= \sum_{d=0}^{s-1} \phi_{s-1d}(d + (a^+_d - a^-_d))\end{aligned} \tag{30}$$

And analogously:

$$\begin{aligned}\Delta_2(s) &= \sum_{d=0}^{s} d^2(a^+_{d-1}\phi_{s-1d-1} + a^0_d\phi_{s-1d} + a^-_{d+1}\phi_{s-1d+1}) \\ &= \sum_{d=0}^{s-1} \phi_{s-1d}(d^2 + 2d(a^+_d - a^-_d) + a^+_d + a^-_d)\end{aligned} \tag{31}$$

Filling in the explicit expressions for the coefficients (see Eq. 28) we obtain:

$$\Delta_1(s) = (1-\frac{1}{n})\Delta_1(s-1) + (1-\frac{1}{P}) \quad (32)$$

$$\Delta_2(s) = (1-\frac{2}{n})\Delta_2(s-1) + (2-\frac{2}{P}+\frac{2}{nP}-\frac{1}{n})\Delta_1(s-1) + (1-\frac{1}{P}) \quad (33)$$

The fixed points of these difference (or recursion) equations are unique and correspond to the limit $s \to \infty$, or equivalently, random sampling, They are found to be

$$\Delta_1(\infty) = < d(A,B) >_{random} = n(1-\frac{1}{P}) \quad (34)$$

$$\Delta_2(\infty) = < d^2(A,B) >_{random} = \frac{n(P-1)(1-n+nP)}{P^2} \quad (35)$$

where A and B are random configurations (with $d(A,B)$ the distance between A and B).
We can define the corresponding relaxation functions $q_k(s)$ [103] as follows:

$$q_k(s) = \frac{< d^k(A,B) >_{random} - < d^k(A_0, A_s) >}{< d^k(A,B) >_{random}} = 1 - \frac{\Delta_k(s)}{\Delta_k(\infty)} \quad (36)$$

where $k = 1, 2$ and A_0 and A_s are the initial and final point of a random walk of length s.
After rewriting, we arrive at the following recursion relations for the $q_k(s)$:

$$q_1(s) = (1-\frac{1}{n})q_1(s-1) \quad (37)$$

$$q_2(s) = \frac{q_1(s-1)(2-2n+2nP-P)}{n-n^2(1-P)} + q_2(s-1)(1-\frac{2}{n}) \quad (38)$$

Clearly for $q_1(s)$ we can obtain immediately:

$$q_1(s) = (1-\frac{1}{n})^s = e^{-s/\tau_1} \quad (39)$$

Where $\tau_1 = \frac{1}{ln(n/n-1))} \approx n$.
To arrive at a closed formula for $q_2(s)$ first the recursion relation for $q_2(s)$ is rewritten by the relation for $q_1(s)$:

$$q_2(s) = q_2(s-1)g + a^{s-1}b \quad (40)$$

where $b = \frac{(2-2n+2nP-P)}{n-n^2(1-P)}$, $a = (1-\frac{1}{n})$ and $g = (1-\frac{2}{n})$.
since $q_2(0) = 1$, we can derive that:

$$q_2(s) = g^s + ba^{s-1}\sum_{i=0}^{s-1}(\frac{g}{a})^i \quad (41)$$

In the sum term we recognize the geometrical series:

$$\sum_{i=0}^{s-1} x^i = \frac{1-x^s}{1-x} \tag{42}$$

which leads to the general expression:

$$q_2(s) = g^s(1 + \frac{b}{g-a}) - a^s \frac{b}{g-a} \tag{43}$$

which can be rewritten using two different relaxation times (τ_1 and τ_2).

$$q_2(s) = (1 + \frac{b}{g-a})e^{-s/\tau_2} - \frac{b}{g-a}e^{-s/\tau_1} \tag{44}$$

obviously,

$$\tau_1 = -\frac{1}{lna} \approx n \tag{45}$$

and

$$\tau_2 = -\frac{1}{lng} \approx \frac{n}{2} \tag{46}$$

3.1.5 Random walks through the TAP landscape

In the previous subsection we have restricted our attention to the correlation structure of distance sequences on the Hamming graph. In this section to every vertex in the graph a cost value will be assigned according to function H, e.g. Eq. 23.
Weinberger [110] proposed the autocorrelation function:

$$\rho(d) = \frac{< (H(A) - < H >)(H(B) - < H >) >_{d(A,B)=d}}{\sigma^2} \tag{47}$$

(where d is the number of random walks steps and σ^2 is the variance of H), as the most useful characteristic of a fitness landscape $H : C \to I\!R$. Apart from totally uncorrelated landscapes, $\rho(d) = \delta(d, 0)$, the simplest class consists of the nearly fractal AR(1) landscapes. A time series which is isotropic, Gaussian and Markovian will lead to an autocorrelation function of the form characterized by [110]:

$$\rho(d) \approx \rho(1)^d = e^{-d/\lambda}, d \ll n \tag{48}$$

where λ is the *correlation length*.
The definition of the autocorrelation function, (47), can be rewritten as

$$\rho(d) = 1 - \frac{< (H(A) - H(B))^2 >_{d(A,B)=d}}{2\sigma^2} \tag{49}$$

According to Eq. 48, the auto-correlation function for an AR(1) landscape can be determined from analysis of the 1-step auto-correlation. Let t and t' be two configurations with $d(t,t') = 1$ and corresponding costs H and H'. According to (49) we can write:

$$\rho(1) = 1 - \frac{< (H - H')^2 >}{2\sigma^2} = 1 - \xi \tag{50}$$

We assume that $\xi \ll 1$, which is reasonable, since we look at a small variation in H.
If ξ is sufficiently small we have

$$\lambda = -\frac{1}{ln\rho(1)} = -\frac{1}{ln(1-\xi)} \approx \frac{1}{\xi} \tag{51}$$

or equivalently,

$$\lambda = \frac{2\sigma^2}{< (H - H')^2 >} \tag{52}$$

3.1.6 The one step correlation function for the Task Allocation Problem

As previously stated, we consider the task allocation problem for a processor topology that is fully connected and homogeneous, so processor- and link speeds are set to unity. Furthermore the work per task is considered to be unity. We consider a general class of random task graphs. Each pair of tasks is connected with probability γ. In graph theoretical terms we consider simple graphs, so maximally one edge connects two vertices (tasks) and a task is not connected to itself.
The TAP phase space properties are studied using the cost function (23). If we mutate the allocation number of task k (in an arbitrary initial configuration) we can derive the following formula for the change in cost $\delta H = H - H'$:

$$\delta H = 2w_k(W_m - W_n - w_k) + 2R \tag{53}$$

if task k gets assigned a new allocation number. Else δH is 0.
w_k is the work associated with task k, m is the previous allocation number, n the new one, W_m is the execution time due to the work on processor m and equivalently for processor n. Both calculation time values are taken before the mutation. The term R denotes the change in the communication cost (communication cost before - communication cost after).
However, we are interested in $< (\delta H)^2) >$. After some algebra we obtain: (including the fact that only a fraction $(P-1)/P$ of the mutations contributes indeed the amount (53)).

$$< (\delta H)^2) > = \frac{P-1}{P}(4(1-2 < R > + < R^2 > +2(< W_n^2 > - < W_m W_n > + < W_n R > - < W_m R >))) \tag{54}$$

So, in order to obtain an analytical expression for (54) we need to calculate six quantities: $< R >$, $< R^2 >$, $< W_n^2 >$, $< W_m W_n >$, $< W_m R >$ and $< W_n R >$. Before continuing with our derivation of the one-step auto-correlation, first an expression for σ^2 will be derived.
We have

$$\sigma^2 =< H^2 > - < H >^2 \tag{55}$$

The simplest of the two terms is $< H >^2$. We can see that

$$< H >= \sum_p < W_p^2 > + \sum_{p,q} < C_{pq} > \tag{56}$$

The probability that a given task i gets assigned a specific allocation number j is denoted by q, consequently the probability that the task doesn't get the allocation number is equal to $1 - q$. So we can consider this as a binomial distribution.
The probability that k tasks get assigned to a specific processor number is therefore given by:

$$\left(\begin{array}{c} n \\ k \end{array} \right) q^k (1-q)^{n-k} \tag{57}$$

Obviously $q = \frac{1}{P}$. The expectation value for k is given by $< k >= nq = n/P$, whereas the variance $< k^2 > - < k >^2$ of k is equal to $\frac{n}{P}(1 - \frac{1}{P})$.
This leads us directly to the following expression for $< k^2 >$:

$$< k^2 >= \frac{n}{P}(\frac{n}{P} + 1 - \frac{1}{P}) \tag{58}$$

which is equal to $< W_p^2 >$ in the case that all tasks have unit weight.
Next, consider $< C_{pq} >$. We are interested in the probability of having l tasks on some processor p, and k tasks on another processor q, sharing x edges. We denote by $P(x \cap (l \cap k))$ the probability that the above event occurs. We can express this probability as a product of two other probabilities using Bayes theorem:

$$P(x|l \cap k) = \frac{P(x \cap (l \cap k))}{P(l \cap k)} \tag{59}$$

So, the probability that we look for is $P(x \cap (l \cap k)) = P(x|l \cap k)P(l \cap k)$; the product of the probability that we have l nodes on some processor p and k nodes on some processor q, times the probability that given the first restriction the tasks on these processors share x edges. This leads to the following expression for the expected communication between an arbitrary processor pair:

$$< C_{pq} >= \sum_l \left(\begin{array}{c} n \\ l \end{array} \right) q_1^l (1-q_1)^{n-l} \sum_k \left(\begin{array}{c} n-l \\ k \end{array} \right) q_2^k (1-q_2)^{n-l-k} \sum_x \left(\begin{array}{c} lk \\ x \end{array} \right) \gamma^x (1-\gamma)^{lk-x} x \tag{60}$$

Where, $q_1 = \frac{1}{P}$ and $q_2 = \frac{1}{P-1}$ which reduces to

$$< C_{pq} >=< x >= \sum_l \left(\begin{array}{c} n \\ l \end{array} \right) q_1^l (1-q_1)^{n-l} \sum_k \left(\begin{array}{c} n-l \\ k \end{array} \right) q_2^k (1-q_2)^{n-l-k} \gamma kl \tag{61}$$

And therefore

$$< C_{pq} >= \sum_l \left(\begin{array}{c} n \\ l \end{array} \right) q_1^l (1-q_1)^{n-l} \gamma l(n-l) q_2 \tag{62}$$

Simplifying to

$$< C_{pq} >= \gamma n q_2 < l > -\gamma q_2 < l^2 > \tag{63}$$

We already saw that $< l^2 >= \frac{n}{P}(\frac{n}{P} + 1 - \frac{1}{P})$ and $< l >= \frac{n}{P}$, so

$$< C_{pq} >= \gamma \frac{n(n-1)}{P^2} \tag{64}$$

This gives us the following expression for $< H >$, where we take into account that the $< W_p^2 >$ term counts P times, and the $< C_{pq} >$ term counts $P(P-1)$ times.

$$< H >= n(\frac{n}{P} + 1 - \frac{1}{P}) + \gamma \frac{(P-1)n(n-1)}{P}) \tag{65}$$

Next an expression for $< H^2 >$ will be derived.

$$< H^2 >=< \sum_{p,q} W_p^2 W_q^2 + 2 \sum_{p,o,m} W_p^2 C_{mo} + \sum_{q,r,m,o} C_{qr} C_{mo} > \tag{66}$$

or,

$$< H^2 >=< \sum_{p,q} W_p^2 W_q^2 > +2 < \sum_{p,o,m} W_p^2 C_{mo} > + < \sum_{q,r,m,o} C_{qr} C_{mo} > \tag{67}$$

The first term can be rewritten in two separate sums. We must distinguish the possibilities $p = q$ and $p \neq q$.

$$\sum_{p,q} W_p^2 W_q^2 = \sum_p W_p^4 + \sum_{p \neq q} W_p^2 W_q^2 \tag{68}$$

Let's consider the case of $p = q$. Assuming k tasks on processor p we have

$$< W_p^4 >=< k^4 >= \sum_k \left(\begin{array}{c} n \\ k \end{array} \right) q^k (1-q)(n-k) k^4 \tag{69}$$

For a binomial distribution the kurtosis (4th moment)

$$< (k- < k >)^4 >= (nq(1-q))^2 (3 + \frac{1-6q(1-q)}{nq(1-q)}) = m_4 \tag{70}$$

And thus,

$$< k^4 >= m_4 + 4 < k^3 >< k > -6 < k^2 >< k >^2 +3 < k >^4 \tag{71}$$

Furthermore the skewness (3rd moment) is given by

$$< (k- < k >)^3 >= nq(1-q)(1-2q) =< k^3 > -3 < k >< k^2 > +2 < k >^3 = m_3 \tag{72}$$

or,

$$< k^3 >= 3 < k >< k^2 > -2 < k >^3 +m_3 \tag{73}$$

Finally we find, since $< k >= nq$ and $< k^2 >= nq(nq + 1 - q)$ that

$$< W_p^4 >=< k^4 >= \frac{n\,\left(-6 + 11\,n - 6\,n^2 + n^3 + 12\,P - 18\,n\,P + 6\,n^2\,P - 7\,P^2 + 7\,n\,P^2 + P^3\right)}{P^4} \tag{74}$$

Next, consider $p \neq q$, that is $< W_p^2 W_q^2 >=< k^2 l^2 >$.
In an analogous manner one arrives at:

$$< W_p^2 W_q^2 >= \frac{(-1+n)\;n\;\left(6 - 5\,n + n^2 - 4\,P + 2\,n\,P + P^2\right)}{P^4} \tag{75}$$

In case of the interference term $< W_p^2 C_{qr} >$, we must consider the cases $p \neq q \neq r$ and $p = q \neq r$.
For the first case we get:

$$< W_p^2 C_{qr} >= \frac{\gamma\,n\;\left(2 - 3\,n + n^2\right)\;(-3 + n + P)}{P^4} \tag{76}$$

And for the second case :

$$< W_q^2 C_{qr} >= \frac{\gamma\;(-1+n)\;n\;\left(6 - 5\,n + n^2 - 6\,P + 3\,n\,P + P^2\right)}{P^4} \tag{77}$$

Finally, we are left with the $< C_{qr} C_{st} >$ terms.
For this case we can distinguish the following (contributing) cases:

1. $q \neq s \neq r \neq t$, leading to terms of the form $< C_{qr} C_{st} >$
2. $q = s \neq r \neq t$, leading to terms of the form $< C_{qr} C_{qt} >$
3. $q = s \neq r = t$, leading to terms of the form $< C_{qr} C_{qr} >$

Analogous to the method above the following expressions can be derived.

$$< C_{qr} C_{qr} >= \frac{\gamma\;(-1+n)\;n\;\left(6\,\gamma - 5\,\gamma\,n + \gamma\,n^2 - 4\,\gamma\,P + 2\,\gamma\,n\,P + P^2\right)}{P^4} \tag{78}$$

$$< C_{qr} C_{qt} >= \frac{\gamma^2\, n\, \left(2 - 3\,n + n^2\right)\, (-3 + n + P)}{P^4} \tag{79}$$

$$< C_{qr} C_{st} >= \frac{\gamma^2\, n\, \left(-6 + 11\,n - 6\,n^2 + n^3\right)}{P^4} \tag{80}$$

Having available all the essential terms for $< H^2 >$, we can now write down the full formula, taking into account the proper pre-factors:

$$\begin{aligned} < H^2 >= P < W_p^2 > + \\ P(P-1) < W_p^2 W_q^2 > + \\ 2(P(P-1)(P-2) < W_p^2 C_{qr} > + \\ 2P(P-1) < W_q^2 C_{qr} >) + \\ 2P(P-1) < C_{qr} C_{qr} > + \\ 4P(P-1)(P-2) < C_{qr} C_{qt} > + \\ P(P-1)(P-2)(P-3) < C_{qr} C_{st} > \end{aligned} \tag{81}$$

Filling out all terms and simplifying the expression we finally end up with the following expression for the variance σ^2:

$$< H^2 > - < H >^2= \frac{2\,(\gamma-1)\,(n-1)\,n\,(1+\gamma\,(P-1))\,(P-1)}{P^2} \tag{82}$$

Note that, because of the appearance of γ^2 terms, Eq. 82 can only be used to predict the variance of an ensemble of random graphs with fixed γ. This is due to the following fact

$$(\sum_i deg(i))^2 \neq \sum_i (deg(i))^2 \tag{83}$$

which states that the squared sum over the individual vertex degrees is generally not equal to the sum over the squared vertex degrees.
So in order to experimentally verify this result we must calculate the variance over multiple graph instances. The γ^2 term is not present in the expression for the average cost (Eq. 65), which implies that it is valid for a specific random graph instance.
Then let's turn to $< (\delta H)^2) >$. We can express this as follows:

$$< (\delta H)^2) >= 4\frac{(P-1)}{P}(< R^2 > -4 < lR > +2(< l^2 > - < kl >)) \tag{84}$$

In the averaging procedure, we consider δH for only those cases that one processor has $(l+1)$ tasks (so at least 1), and the processor that the transfer is to has k tasks.
The following expressions for the individual terms can be derived:

$$\begin{aligned}
< R^2 >= & \frac{2\,\gamma\,(-1+n)}{P} \\
< lR >= & \frac{\gamma\,(-1+n)}{P} \\
< l^2 >= & \frac{(-1+n)\,(-2+n+P)}{P^2} \\
< kl >= & \frac{(-2+n)\,(-1+n)}{P^2}
\end{aligned} \tag{85}$$

which leads to

$$< (\delta H)^2) >= \frac{8\,(-1+\gamma)\,(1-n)\,(-1+P)}{P^2} \tag{86}$$

And thus our one-step auto correlation:

$$\rho(1) = 1 - \frac{< (H - H')^2 >}{2\sigma^2} = 1 + \frac{2}{n\,(-1+\gamma-\gamma\,P)} \tag{87}$$

Applying Eq.(52) we find directly

$$\lambda = \frac{n}{2}(1+\gamma(P-1)) \tag{88}$$

We see that for fixed γ and P, λ is linearly proportional to the number of tasks n. Note that we have assumed that $P > 1$ in our derivation, otherwise $\rho(1)$ is not defined.
It is very important to observe that there are no dependencies of γ^2 in Eq. 86, which implies that the variance in γ (due to σ^2) does not get eliminated. Strictly speaking this means that the derived formula for λ does not correctly predict the correlation structure of the landscape for single task graph instances. However, the $n/2$ term is obviously present in Eq. 88, which corresponds to the correlation time τ_2 derived in section 3.1.4. In section 3.1.15 we shall see that this is also the correlation length found experimentally.

3.1.7 Physics of Task Allocation

It can be shown that the Hamiltonian (energy function) of a spin glass is similar to the cost function of a well known NP-complete problem: graph bi-partitioning [71]. The cost function of the graph bi-partitioning problem can be considered as a special instance of that of the TAP. In analogy with spin glasses and graph bi-partitioning the TAP Hamiltonian will be formulated.
Application and machine specific parameters are used to distinguish two different phases (a sequential- and a parallel allocation phase) in the spectrum of optimal task allocations. The location of the separation between the two phases as a function of the aforementioned parameters is determined by a mean field argument. This location gives a rough estimate of the transition region.

Many search methods have been shown to behave anomalytically for certain critical parameters of the instance of combinatorial search problems [111] (critical slowing down). We speculate on the existence of such an anomaly (often observable as a sudden increase in the search cost) in the spectrum of TAP instances.

3.1.8 Spin Glasses and Graph bi-partitioning

In the area of condensed matter physics, a canonical model to describe the properties of a magnet is the Ising model. In d dimensions this is a regular square lattice of atomic magnets, which may have spin up or spin down. Formally, we have n variables s_i, one for each individual magnet, where s_i can take on values $+1$ or -1. The Hamiltonian describing the magnetic energy present in a specific configuration, without an external magnetic field, is given by:

$$H = -\sum_{k>i} J_{ik} s_i s_k. \tag{89}$$

For the Ising spin model, the interaction strength J_{ik}, is constant. However, if the J_{ik}'s are independent negative and non- negative random variables, we obtain the spin glass Hamiltonian. The spin glass model exhibits frustration, opposed to the (square-lattice) Ising model. This specific characteristic of the Ising system causes only two ground states to be present in the Ising model (all spins up, or all spins down) and many (highly degenerate) ground states for the spin glass. While in the Ising model, each pair of aligned spins contributes the same amount of energy, this is not true for a spin glass. Alignment with one neighboring spin can result in an energetically unfavorable situation with another neighbor.
A well known NP-complete problem, graph bi-partitioning, has a cost function which is equivalent to the Hamiltonian of the spin glass model. We consider a graph, a set of n vertices and E edges. A configuration is an equal partition of the vertices. This can be expressed with the following constraint:

$$\sum_i s_i = 0, \tag{90}$$

where $s_i = 1$ if vertex i is in partition 0 and $s_i = -1$ otherwise. The edges can be encoded with a connectivity matrix J_{ik}. Such that $J_{ik} = 1$ if i and k are connected and $J_{ik} = 0$ if not. The Hamiltonian of a configuration can be expressed as follows:

$$H = \sum_{i<k} J_{ik}(1 - s_i s_k)/2. \tag{91}$$

Eq. 91 is equal to the spin glass Hamiltonian (Eq. 89), up to a constant value of $\sum_{i<k} J_{ik}/2$. The constraint (90) introduces frustration, otherwise the cost

would be minimal for all vertices in one partition. In other words, without the constraint we would have a simple Ising ferro-magnet.
For a detailed review of spin glass theory and graph bi-partitioning we refer to the book by Mezard *et al.* [71].

3.1.9 Task Allocation Hamiltonian

In analogy with the models above we can rewrite the task allocation cost function (23) as follows:

$$H = (1-\beta)\sum_{i>k}^{n} J_{ik}(1-\delta(s_i,s_k)) + \beta\sum_{l}^{P} W_l^2 \ , \tag{92}$$
$$\delta(s_i,s_j) = \begin{cases} 1 & \text{if} \quad s_i = s_j \\ 0 & \text{otherwise} \end{cases} ,$$

The processor to which task i is allocated, is denoted by $s_i \in \{1 \dots P\}$ and P is the number of processors. J_{ik} is a contribution to the communication between the host processors of tasks i and k, resulting from the connection between these tasks. W_l the total calculation weight on processor l, following from the individual workloads of all allocated tasks.
Note that we have introduced a parameter β into the Hamiltonian. This β-parameter can be varied in the range $[0,1]$, in order to tune the amount of "frustration" between the calculation and the communication terms. Variations of β can be interpreted either as variation in an application's calculation-communication ratio or a machine's processor speed - bandwidth ratio [26].
The connection probability γ in a random graph, can be considered as a dual parameter for β. γ can be increased in the range $[0,1]$, which is equivalent to augmenting the average communication load, which can also be realized by decreasing β.

3.1.10 The TAP Phase Transition

Although the task allocation problem is NP-hard [8], the two extremes, $\beta = 0$ and $\beta = 1$ are easy to solve. For $\beta = 0$, the only relevant term in the Hamiltonian is an attracting communication term, which will cause all *connected* tasks to be allocated to one processor. For this extreme, the number of optima is exactly P. The corresponding lowest energy state will have value zero. This situation corresponds to a parallel machine with infinitely fast processors.
For $\beta = 1$ there is only a repulsive workload term, which will force that the variance in the workload distribution is minimized. This results in an equal partitioning of the total workload over all available processors. It can easily be shown that the total number of optima in this case equals:

$$\prod_{k=1}^{P} \begin{pmatrix} n \\ k(n/P) \end{pmatrix} = \frac{n!}{(n/P)!^{P}}. \tag{93}$$

It is assumed that the n tasks have unit weight and that n/P is integer. The corresponding optimal cost value obviously will be n^2/P.
In the case of $\beta = 0$ the P optima are maximally distant in terms of the defined distance metric (see section 3.1.3). The P-ary inversion operation (analogous to spin-flipping in spin glass theory) and arbitrary permutations applied to a given optimal configuration leave the value of the Hamiltonian invariant. Note that, in this case, the TAP landscape is highly symmetrical. The entire landscape consists of P identical sub-landscapes. Each sub-landscape has only one optimum, which is automatically the global optimum.
In case of $\beta = 1$, the optima are relatively close to one another. Again, we can distinguish two types of operations that leave the value of the Hamiltonian invariant. The first type is trivial, that is, permutation of tasks allocated to the same processor, since this corresponds to the same point in phase space. The second type may change the point in phase space. Examples of such operations are rotation of the sequence and permutation of two arbitrary tasks.
From the perspective of parallel computing it is most ideal when all processors are engaged in a computation. However, the employment of all available processors does not always correspond to the optimal solution due to the communication overhead. Both machine and application specific parameters, which can be summarized as the ratio between the communication and calculation time, determine this optimal value.
We can observe a transition from sequential to parallel allocation when β is increased from 0 to 1 (or equivalently, if γ is decreased from 1 to 0). In order to quantify this transition we have to define an *order parameter*, which is a measure for the *degree of parallelism* present in an optimal allocation.
We assume that all tasks and connection weights are unity and define the order parameter $\mathcal{P}$, quantifying the *parallelism* in a given optimal allocation:

$$\mathcal{P} = 1 - \frac{(< W^2 > - < W >^2)P^2}{n^2(P-1)}. \tag{94}$$

where W is the time spent in calculation and $n^2(P-1)/P^2$ is the maximal possible variance in W. Eq. 94 takes the value 1 in the case of optimal parallelism ($\beta = 1$ or $\gamma = 0$) and the value 0 ($\beta = 0$ or $\gamma = 1$) in the case of a sequential allocation.
Using Eq. 95 which expresses the average cost, which was derived in section 3.1.3, we can calculate whether the average cost will either increase or decrease by using more processors for an allocation. Note that β has been included into Eq. 65. We expect that the transition from sequential to parallel allocation will approximately occur for those values of β and γ for which the average cost will change from a monotonically decreasing function to a monotonically increasing function of P.

$$< H >= \beta n(\frac{n}{P} + 1 - \frac{1}{P}) + (1-\beta)\gamma\frac{(P-1)n(n-1)}{P}. \tag{95}$$

We use Eq. 95 to predict for which values of γ and β the transition will occur approximately. In Fig. 8 an example of this transition is depicted, for a task graph

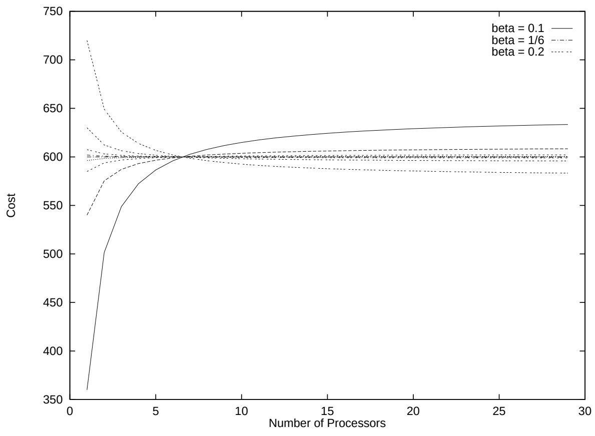


Figure 8: $< H >$ vs. P for increasing β, $n = 60$ and $\gamma = 0.2$

with $\gamma = 0.2, n = 60$. The transition point as predicted will approximately occur for the following values of β and γ keeping one of the two variables fixed with the additional constraint that $\frac{\partial <H>}{\partial P} = 0$.

$$\beta_c = \frac{\gamma}{1+\gamma} \tag{96}$$

$$\gamma_c = \frac{\beta}{1-\beta} \tag{97}$$

We interpret β_c and γ_c as the "critical" values of β and γ in analogy with e.g. the critical temperature T_c in thermal phase transitions or percolation threshold p_c in percolation problems. Note that in Fig. 8 there is a point where the average value of the Hamiltonian is independent of β (approximately at $P = 7$). This is due to the fact that Eq. 95 contains β independent terms and therefore the β dependent terms can be eliminated for certain values of P, given fixed n and γ.

3.1.11 Experimental Methods

In this section several experimental methods that will be used in our study are introduced. Firstly, SA which is used to find sub-optimal solutions to the TAP. Secondly, a method is presented to quantify the computational search cost. Thirdly, we will briefly discuss an experimental method to determine the correlation length of the phase space of a TAP instance.

3.1.12 Simulated Annealing For Optima Search

In simulated physical systems, configurations at thermal equilibrium can be sampled using the Metropolis algorithm [70]. The determination of the loca-

tion of the critical temperature, can be established by sampling at fixed temperatures over some temperature range.
In the case of task allocation we are not interested in finding equilibria, but optimal configurations. For this purpose, exhaustive search is the only correct method that can be used for finding global optima. Unfortunately, this can be very inefficient and in worst case requires exponentially large search times. Therefore another more effective search method has to be selected.
In previous work [18][94], we have applied a Parallel Cellular Genetic Algorithm (PCGA) to find optimal solutions to the TAP in a parallel finite element application. Another possibility is using SA. The usefulness of SA on the TAP depends on the shape of the phase space. In section 3.1.3 we argued that the landscape has a self-similar structure, which is an indication of good performance of local heuristic search techniques. Indeed we have found that SA, was superior to GA, both in efficiency and quality of the solution. Therefore SA is applied to find the (sub) optimal solutions.

3.1.13 Search Cost Estimation

In comparable (NP-hard) problems the computational cost of determining the (optimal) solutions shows a dependence on problem specific parameters [112][42][12]. For example, in the case of graph coloring it has been observed that the "difficulty" of determining if a graph can be be colored, increases abruptly when the average connectivity in the graph is gradually increased to some critical value [111].
Another example of a system where computational cost is affected by such parameters is that of a physical system where a thermal phase transition occurs (like the Ising model). The difficulty of finding the equilibrium value increases when the critical point is approached, and theoretically (in the thermodynamic limit) will become infinite at the critical point. This is generally referred to as critical slowing down.

In analogy with this behavior we expect that in the task allocation problem comparable phenomena can be found in a critical region of the β and γ-domain. For both β extremes the optima are known in advance. The difficulty to find these optima is therefore reduced to order unity. If the calculation and the communication term in the Hamiltonian (92) are of comparable magnitude we can say that the system is in a critical (or frustrated) area. Moving away from this critical region one term becomes small noise for the other.
We are interested in a method for obtaining an estimate of the computational cost (difficulty) of finding optima for given problem parameters. In order to quantify the search cost, we measure the number of local optima, in which independent steepest descent runs get stuck. A specific search space is considered to be "simple" if it contains a relatively small number of local optima. On the other hand, if the number of local optima is large the corresponding search space is classified as "difficult". The distinction between local optima is based

on the cost of the corresponding task allocations. That is two allocations i and j (that are local optima) are called distinct if:

$$H(i) \neq H(j) \tag{98}$$

In the experiments below, the number of steepest descent runs is taken to be $10n$, with n the number of tasks.

3.1.14 Measuring Phase Space Structure

The structure of the TAP phase space is characterized using the auto-correlation function (99) of a random walk.

$$\rho(d) = \frac{< H(A)H(B) >_{d(A,B)=d} - < H >^2}{\sigma^2}, \tag{99}$$

where $d(A, B)$ is the "distance" between two configurations A and B as introduced in section 3.1.3. The value for λ for the task allocation phase space can be directly determined from $\rho(1)$.

3.1.15 Experimental Results

In this section experimental results regarding the statistical quantities, correlation length, phase transition and search cost for the TAP are presented.
First a number of experiments, conducted to verify the analytical results derived in section 3.1.3 are given. It is established that the TAP landscape is AR(1), which supports the argument for using SA in the subsequent experiments for finding optimal allocations.
The occurrence of the phase transition for several parameter values is observed in the corresponding experiments. Complementary to the phase transition is the divergence of the computational cost, which is also shown to manifest itself.

3.1.16 Statistical Quantities and Correlation Length

In for example the *Traveling Salesman Problem* (TSP) [103] statistical quantities of the landscape of random TSP instances can be obtained by random walk averaging. This is not possible for TAP. Only for both connectivity extrema, $\gamma = 0.0$ and $\gamma = 1.0$, the random walk is self averaging, which means that the ensemble average can be obtained by a random walk. For other values of γ each instance of a random graph differs in connectivity from the other, which implies that statistical quantities can only be estimated by averaging over multiple instances of random graphs.
The determination of the auto-correlation functions is obtained using a specific instance of the TAP with fixed γ, n and P. We can not use the derived formula for the variance to predict the variance of a single TAP instance, this is due to the presence of γ^2 terms in the expression for σ^2 (see Eq. 82). Such terms are not present in the formulae for $< H >$ (Eq. 65) and $< (\delta H^2) >$ (Eq. 86).
In all figures error bars are displayed, if applicable.

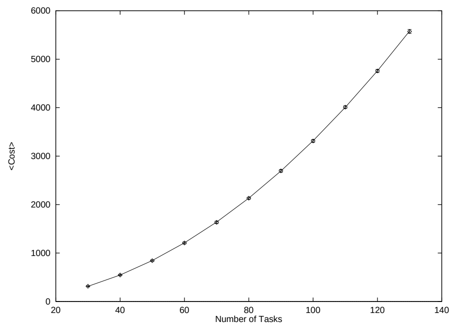


Figure 9: $< H >$ for different n

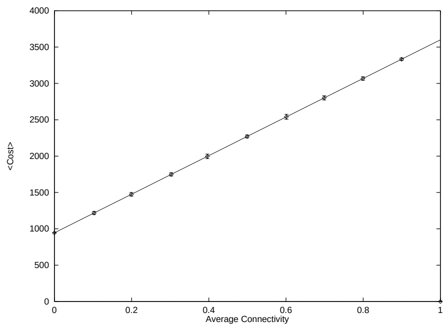


Figure 10: $< H >$ for different γ

3.1.17 Experimental Verification of Cost Terms

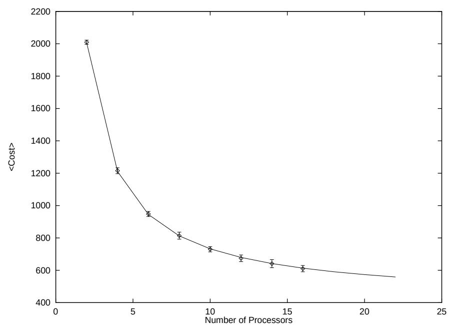


Figure 11: $< H >$ for different P

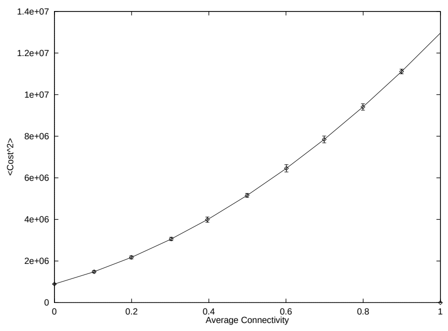


Figure 12: $< H^2 >$ for different γ

In this section we experimentally verify the derived expressions (Eqs. 65 and 81) for the expected cost and expected squared cost. Furthermore the equation for $< (\delta H)^2) >$ (Eq. 86) is experimentally verified. We have carried out experiments with variable number of processors (P), connectivity (γ) and number of tasks (n). For each variable parameter the other two parameters were kept fixed ($n = 60$, $\gamma = 0.1$ and $P = 4$). The results are shown in Figs. 9-17.

3.1.18 Analytical and Measured λ

In this section the correlation length is experimentally determined. For these experiments random walks through the TAP landscape with approximate lengths of 10^5 steps were generated. Subsequently, the autocorrelation functions using the values encountered were calculated.
In Fig. 18 two measured and predicted correlation functions are displayed. In the first experiment we have used 100 tasks, 8 processors and a connection probability of 0. In the second experiment a TAP instance with a non-zero connection probability ($\gamma = 0.5$), $n = 64$ and $P = 4$ was used.

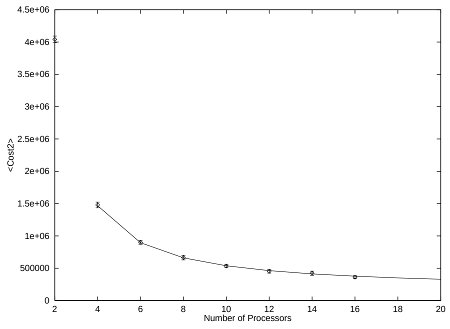


Figure 13: $< H^2 >$ for different P

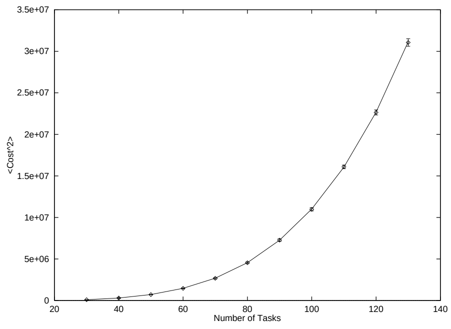


Figure 14: $< H^2 >$ for different n

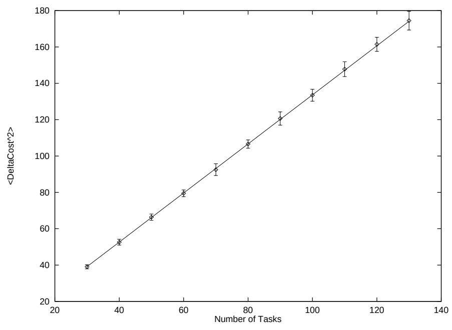


Figure 15: $< (\delta H)^2 >$ for different n

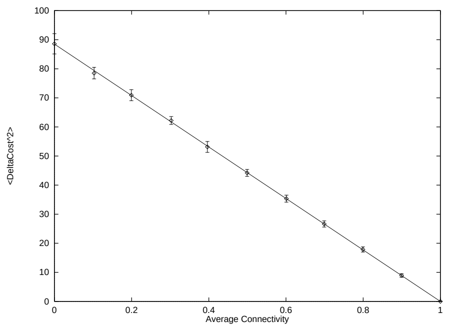


Figure 16: $< (\delta H)^2 >$ for different γ

3.1.19 Phase Transitions and Computational Search Cost

Several experiments are performed to demonstrate the existence of a phase transition, and the location of the transition as predicted by Eq. 96 is checked. The experiments to which the depicted data corresponds were carried out with $n = 64$ and $P = 8$. In Fig. 19, β is varied in the range $[0, 1]$ and γ is fixed at two different values (0.2 and 0.4). In Fig. 20 the dual experiment is performed, now γ is varied in the range $[0, 1]$ and β is fixed at the value 0.25. The results presented are comparable with those found for arbitrary parameter values. The mean field transition points are plotted as vertical lines.

In Figs. 21 and 22 the divergence of the search cost near the transition point can be observed. The method described in section 3.1.13 is used to quantify the cost. In Fig. 21, $n = 32$ and $P = 4$ and γ is fixed to 0.5. An increase in the number of local optima is found around the location of the phase transition. Another example is shown in Fig. 22, where $n = 64$, $P = 8$ and β is fixed to 0.2. Again the computational cost increases in the neighborhood of the phase transition.

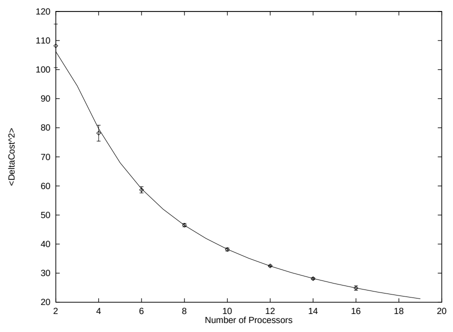


Figure 17: $< (\delta H)^2 >$ for different P

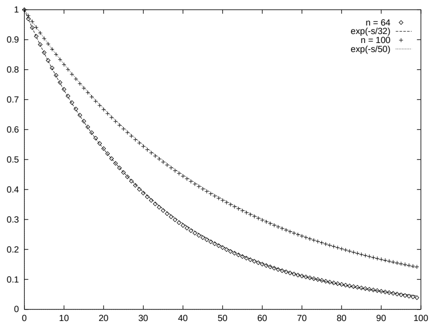


Figure 18: Analytical and experimental values for the autocorrelation function $n = 100, P = 8$ and $\gamma = 0.0$ and $n = 64$, $P = 4$ and $\gamma = 0.5$.

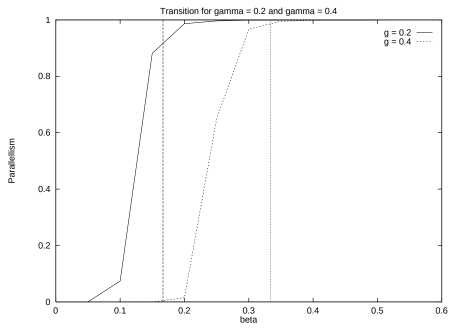


Figure 19: phase transition with fixed γ (0.2 and 0.4) and increasing β. The vertical solid lines indicates the location of the transition as predicted by Eq. 96

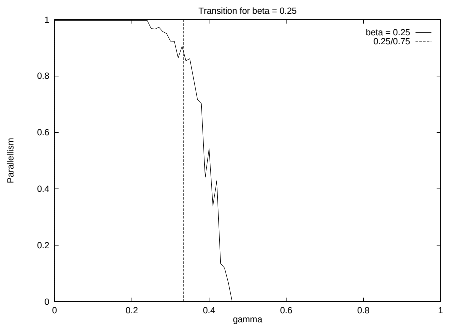


Figure 20: A phase transition with $\beta = 0.25$. The vertical solid line indicates the location of the transition as predicted by Eq. 96

3.1.20 Summary and Discussion

In analogy with graph bi-partitioning and spin-glass theory we have constructed a cost function that expresses task allocation quality into a Hamiltonian form. It has been argued that the TAP is an example of so called frustrated systems, and as such is expected to show typical complex behavior. The competition between the calculation and communication terms in the Hamiltonian is the source of frustration. In order to facilitate our study on frustration in the TAP a control parameter β was introduced into the Hamiltonian. The β parameter can be considered as a dual parameter for the degree of connectivity between tasks in the task graph. In the case of random task graphs γ is the connection probability between vertices (or tasks). The β parameter has an important interpretation in terms of high performance computing terminology. It either

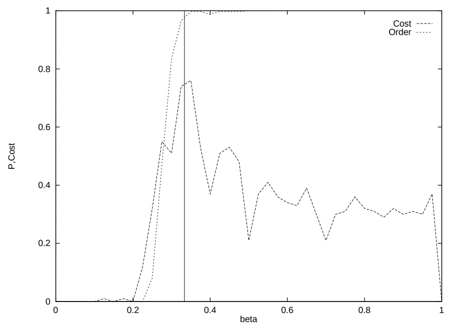


Figure 21: Computation cost diverges at the phase transition, $P = 4, n = 32, \gamma = 0.5$, β varied

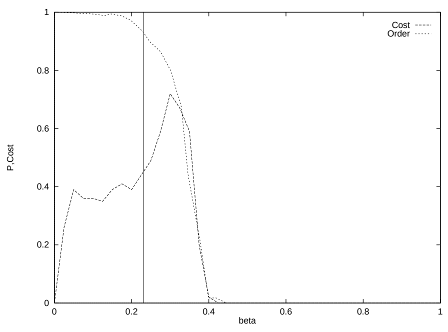


Figure 22: Another example with $n = 64$ and $P = 8$, $\beta = 0.2$, γ varied.

expresses an application's calculation-communication ratio or a machine's processor speed-bandwidth ratio.

In order to select a suitable method to find optima several aspects of the TAP phase space were investigated. Firstly, some basic characteristics like the size of the TAP phase space and its diameter were given. Secondly, the concept of AR(1) landscapes was discussed and the performance of SA on landscapes that exhibit such structure. We have derived the correlation length of a random walk trough the TAP phase space. First we derived analytical expressions for the relaxation functions on random walks through the TAP landscape. It was shown that the correlation length of the phase space corresponds to one of the two relaxation times (τ_2) found in this expression (Eq. 44). Secondly, a formal expression for both the variance of the cost and the squared difference in the cost between two subsequent allocations was derived.
The number of global optima for the extreme values of β in the Hamiltonian was discussed, as well as the invariance properties of the Hamiltonian in these cases. An order parameter $\mathcal{P}$, was introduced to quantify a degree of parallelism. Using an expression for the average value of the Hamiltonian (or cost) a rough measure was given for the location of the transition region where the optimal solution changes from sequential to parallel allocation.
Next, the observation was made that comparable systems show divergent behavior in the computational cost that is associated with finding optimal values in a critical region, e.g. near a phase transition. It was argued that the transition of sequential to parallel allocation, induced by varying β or γ, is expected to give rise to analogous critical behavior for the search cost in the TAP.

3.1.21 Statistical Quantities and Correlation Length

From Figs. 9-17 it is clear that the analytical formulae predict the corresponding average quantities to a high degree of accuracy. The specific choice of param-

eters does not influence the accuracy of the experiments. In other words, the specific problem instances for which the data are shown are indicative for the correctness of the derived expressions.
We only have an expression for the variance over an ensemble of random graphs with a fixed value of γ. This can not be used to predict the correlation length (λ) of the autocorrelation function for a random graph instance. Therefore, we can not derive an exact expression for the one step auto-correlation function. However the correlation time τ_2, found in Eq. 44 corresponds to the correlation length that is found experimentally. Empirically, it can be shown that the variance of H over a random graph instance is approximately described by the following equation:

$$< H^2 > - < H >^2 = \frac{2\ (\gamma - 1)\ (n-1)\ n\ \ (P-1)}{P^2} \tag{100}$$

Eq. 100 correctly predicts the variance for a single random graph instance (data not shown). Using this equation (100) and Eq. 86 we do find the correct prediction of λ:

$$\rho(1) = 1 - \frac{2}{n} \rightarrow \lambda = \frac{n}{2} \tag{101}$$

The corresponding correlation length $\lambda = n/2$ indicates that the TAP landscape is smooth in comparison with that of graph bi-partitioning ($\lambda = n/4$). The correlation length is strongly related to the chosen perturbation mechanism. Increasing correlation length indicates a phase space that is smoother and easier to search. The fact that the search process benefits from larger correlation lengths, has been established in [104].

3.1.22 Phase Transitions and Computational Search Cost

As shown in Figs. 19 and 20, the approximate location of the phase transition that is induced by variation of β or γ can be predicted by a simple mean field argument (A practical consequence of this observation is, that given real application and machine parameters this can provide an estimate of the "usefulness" of parallel computation of the problem at hand).
In Figs. 21 and 22 it is shown that the presence of a phase transition is accompanied by the anomalous behavior of the search cost. This behavior is analogous to that observed in comparable combinatorial optimization problems as argued above.

3.1.23 Concluding Remarks and Future Work

It has been shown that the landscape of the TAP can be classified as an AR(1) landscape. Given the fact that the correlation length is $n/2$ for the defined cost function and neighborhood structure, the TAP is an easier problem to solve using local search methods than e.g. graph bi-partitioning. In other work [103], it has been shown that both the landscape of spin-glasses and

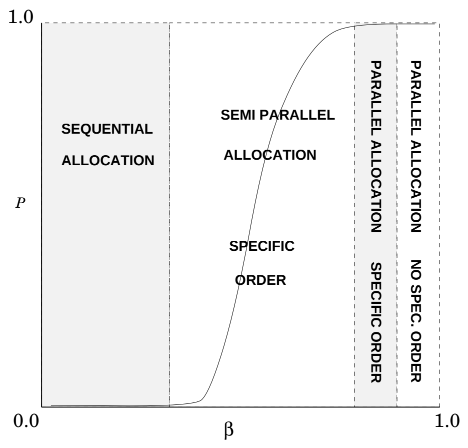


Figure 23: The different allocation regimes in the task allocation problem for varying β

graph bi-partitioning have a correlation length of $n/4$, which makes those landscapes less smooth than that of the TAP.
The results presented in this section show that the task allocation problem exhibits a variety of interesting properties. For specific parameter sets the task allocation problem exists only in a small parameter range. Outside this range the problem is trivial. The problem becomes complex in the range where the calculation and communication terms are of comparable magnitude. The location of this complex region is marked by the presence of a transition from sequential to parallel allocation. The different allocation regimes are summarized in Fig. 23. The *sequential allocation* region only contains optima where all tasks are allocated to one processor. The *semi-parallel allocation* region can correspond to the following situation. Not all available processors are necessarily used, due to the high competition with the communication cost. Also the locality in the task graph has its consequences for the allocation sequence. Tasks that are connected to one another "desire" to be grouped on the same processor. The last region, *parallel allocation*, corresponds to the mode where the inter task connectivity has become insignificant. This may either be due to a high speed communication network or weakly connected task graph. In this case optimal allocations can be realized using scattered decomposition.
In the near future we intend to investigate the effects of introducing locality in both the task graph as well as the processor topology. Also, we will study the critical behavior in more detail, by means of finite size scaling experiments [64]. Furthermore, our interest goes out to a thorough understanding of task allocation in dynamic and heterogeneous parallel applications and machines, in e.g. cluster computing [79].

3.2 An Abstract Cellular Genetic Algorithm for solving optimization problems

Many problems from the natural sciences can be considered as optimization problems. One approach to solve these problems, is to use numerical methods, another attractive approach is to use stochastic or natural solvers. Two typical natural solvers are Simulated Annealing [52] and Genetic Algorithms [43]. A fundamental problem is that both methods are difficult to parallelize to a level of high scalability. Classical GAs use global knowledge for their selection process. There does not exist a spatial relation between the different individuals. An essential problem in SA is that the method is inherently sequential. Our approach to parallelize both methods is to introduce locality by using a mapping to a Cellular Automata. Examples in which a GA is mapped on a Cellular Automata are given in [65], [36], and [107]. In the general case it is not possible to map SA on a Cellular Automata. However locality can be imposed to SA by applying a population based algorithm [35]. In simultaneous independent searches [4] basically the same method is used without interactions.
In [1] a generic algorithm, the so-called Abstract Genetic Algorithm, for both SA and GA was introduced. The AGA however was not designed to facilitate parallelization. The purpose of this section is to describe an Abstract Cellular Genetic Algorithm suitable for parallelization. In this ACGA the locality is introduced by mapping both GA and population based SA on a Cellular Automata. We will recognize a general framework of a VIP model, consisting of "active" individuals (VIPs) with interactions constrained by the lattice topology. In this section we address the theoretical considerations and give some preliminary results of a GA instance of the ACGA, implemented on a sequential machine.

3.2.1 The Abstract Cellular Genetic Algorithm: a Theoretical Framework

To avoid the use of global knowledge which is necessary in the Abstract Genetic Algorithm of [1] we introduce a local spatial neighborhood structure. In [1] a neighborhood is assigned to the value of an individual, not to the location of an individual. The main idea behind the ACGA is to make an analogy between the individual (or solution vector) and a cell in a Cellular Automata. Each individual is assigned to a cell, which explicitly defines its neighborhood structure. All communication is local, cells can only interact with their direct neighbors, Consequently we can formulate the ACGA:

```
Initialize
DO
FOR EACH cell in the population
DO IN PARALLEL
Choose a parent list (selection)
Recombine parent list (reproduction)
Mutate the offspring
Evaluate offspring
IF offspring meets some criterion (evaluation)
THEN
accept offspring
ELSE
leave the current individual in its place
ENDIF
ENDFOR
UNTIL some stop criterion
```

In order to define the ACGA in a formal way we are going to restrict the optimization problem to minimization problems. An object function $f : \mathcal{S} \rightarrow \mathbb{R}$ over the finite search space $\mathcal{S}$ has to be minimized in this type of problem.
The ACGA can be studied in the framework of Probabilistic Cellular Automata[89]. The PCA is a special case of a Probabilistic Automata Network. In the next sequel we will call cells (or individuals) nodes, using the definitions of [89].

Definition
Let V denote the set of nodes of an undirected graph $G = (V, E)$. Each node $v \in V$ is called an *automaton with a state* $s_v \in \mathcal{S}$. The *system state space* is called $S = \bigcup_{v \in V} \mathcal{S}$. Each $s \in S$ denotes a *system state*. The neighborhood $N_v^{(x)}$ of node v is determined by the *neighborhood structure* $N^s \subset \mathbb{Z}$, which is a finite set of offsets:

$$N_v^{(x)} = v + N^s = \{v + a : a \in N^s\}$$

All automata possess the same well-defined transition matrix P, that gathers the probabilities that a state i transitions to a state $j, \forall i, j \in S$. The new system state $s(t+1)$ at step $t+1$ depends on the previous system state $s(t)$ and the transition matrix P:

$$s(t+1) = g(s(t), P)$$

where $g(.)$ symbolizes the synchronous $updaterule$. A PCA is completely determined by the tuple $(V, S_0, N^s, P, s(0))$.

We already mentioned the phrase *spatial neighborhood* to denote the neighborhood $N_v^{(x)}$ we defined above. Let us call the other neighborhood structure that is used to find the new state s_v in the next time step the *temporal neighborhood*:

$$N^{(t)} : \mathcal{S} \to \mathcal{S}$$

A neighborhood associated with an $s_v \in \mathcal{S}$ is denoted by $N_s^{(t)}$.
L denotes the set of all finite lists of elements in $\mathcal{S}$. A *parent-list* $p_v \in P$ associated with a node v is a list of individuals capable of producing new elements. Following [1], we introduce the sets A, B, C and three parameters $\alpha \in A, \beta \in B, \gamma \in C$ that are chosen by random drawings.
A *selection function* $f_s^v : A \times N_v^{(x)} \to P$ is used to choose a parent list from a neighborhood $N_v^{(x)}$ such that:

$$\forall \alpha \in A \forall x \in N_v^{(x)} \forall y \in f_s^v(\alpha, x) : y \subseteq x$$

A *reproduction function* is used to produce new individuals from the parent list: $f_r^v : B \times L \to L$, such that the updated node is from the neighborhood:

$$\forall \beta \in B \forall x \in L : f_r^v(\beta, x) \subseteq \cup_{s \in x} N_s^{(t)}$$

A *evaluation function* is used to choose the final instantiation of the node v that is to be updated, from the parent list L extended with the newly produced individuals L': $f_e^v : C \times (L \cup L') \to \mathcal{S}$, such that:

$$\forall \gamma \in C \forall y \in (L \cup L') : f_e^v(\gamma, y) \subseteq y$$

Using the defined functions the Abstract Cellular Genetic Algorithm can be formally defined as follows:

```
Create an initial system state $s \in S$.
DO
FOR EACH $v \in V$ DO IN PARALLEL
draw $\alpha, \beta$ and $\gamma$
$q^v = f_s^v(\alpha, N_v)$
$y^v = f_r^v(\beta, q_v)$
$v' = N_v \cup y^v$
$v = f_e^v(\gamma, v')$
ENDFOR
$s = \bigcup_{v \in V} s_v$
UNTIL stop
```

3.2.2 Instance of the ACGA I: Cellular Genetic Algorithm

From the ACGA algorithm above a parallel C-GA with local selection can be derived straightforward. Following the formal definition of the ACGA we can specify the state-space and the spatial neighborhood:

$$N^s = \{-1, 0, 1\}^k, \;\; (k \in I\!N)$$

We need to specify the selection- (f_s^v), recombination-, mutation- (f_r^v) and evaluation (f_e^v) operator.

First the selection operator. A conventional GA uses a global fitness function in order to select the parents. With a C-GA one parent is explicitly selected, the current cell k, the other must be selected from the neighborhood with size r , using a local fitness function F. A cell is chosen as the second parent by picking a uniformly distributed random number $\xi \in [0, 1)$ if it satisfies the following rule:

$$\xi \ < \ \frac{F(x_k)}{\sum_{x_j \in N_k^{(x)}} F(x_j)}$$

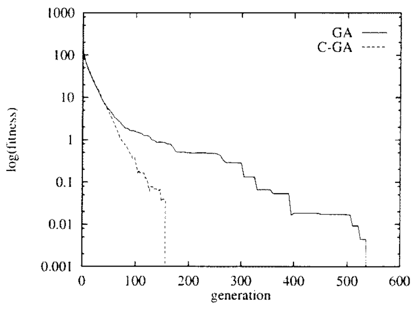
As a recombination operator we can take the unchanged GA-crossover operator. Also the GA-mutation operator can be used. Crossover between two cells is done by taking a uniformly distributed random number $\psi \in [0, length(individual)]$ as the splitting location. Mutation is done by "bit-flipping" every bit of an individual with a probability p_m. Evaluation means calculating the fitness of the new individual.

3.2.3 Instance of the ACGA II: Cellular Simulated Annealing

To introduce locality in the SA algorithm we use an alternative approach whereby several configurations exist together on a n-dimensional grid. These configurations only know of the existence of other configurations in their direct neighborhood. This neighborhood is defined as a spatial structure on the grid. If a new configuration has to be evaluated for acceptance not only the previous configuration is taken into account, but also its neighborhood set. Rejection of a new configuration can cause any of the configurations in a neighborhood set to take over the current spatial grid location. A configuration is chosen according to the Boltzmann tournament rule $btr()$ [35]. Of course evaluation is done, after the perturbation, by the Boltzmann rule. The conventional Boltzmann rule is modified in order to incorporate selection in the evaluation of several configurations. We define a neighborhood set $N_i^{(x)}$ of configuration i, which includes i itself and its spatial neighbors. The selected configuration from the neighborhood $N_i^{(x)}$ is given by $btr(N_i^{(x)})$. The change of the energy of configuration i is given by ΔE_i. The energy of configuration i by $F(i)$. A perturbed configuration i is generated by $mutant(btr(N_i))$. The Neighborhood Boltzmann Rule is now defined as:

$$\begin{cases} P(\Delta E_i, T) = exp\left[\frac{-(F(mutant(btr(N_i^{(x)}))) - F(btr(N_i^{(x)})))}{T}\right] & \text{if } \Delta E_i > 0 \\ P(\Delta E_i, T) = 1 & \text{if } \Delta E_i \leq 0 \end{cases}$$

Unaccepted configurations are replaced by $btr(N_i)$. The difference with conventional SA is that instead of looking only to one configuration, several configurations are used to evaluate a newly generated configuration. Note that by using an empty neighborhood structure N^s, we obtain a C-SA based on independent Markov Chain, i.e. without any interaction.



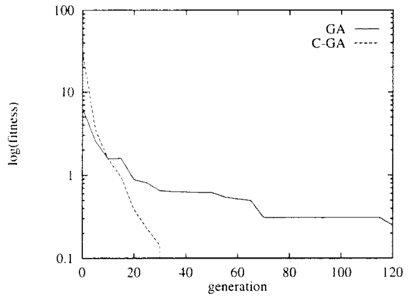


Figure 24: *Convergence of C-GA and standard GA using function f_4 from Eq. 102.*

Figure 25: *Convergence of C-GA and standard GA using function f_5 from Eq. 103.*

3.2.4 Experiments on the Convergence Behavior of C-GA

In order to compare the convergence behavior of C-GA to a standard GA [34] with elitist strategy. It is known that standard GA algorithms are sensitive to the shape of the search space. To reduce the influence of shape dependency we used a "smooth" shaped function with one local minimum and a more irregular shaped function with more local minima. In both algorithms we approximated optimal parameter settings.
As a test case we used de Jong's test functions [46]:

$$f_4(x) = \sum_{1}^{30} i{x_i}^4 + Gauss(0,1) \text{ for } -1.28 \leq x_i \leq 1.28 \tag{102}$$

$$f_5(x) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6} \text{ for } -65.536 \leq x_i \leq 65.536 \tag{103}$$

The function f_4 is a continuous, convex, unimodal, high-dimensional function which contains some Gaussian noise. The search space consists of $256^{30} \approx 10^{72}$ solutions. The global minimum is at $x_i = 0.0$ with $f_4(x_i) = 0.0$. The function f_5 is a continuous, non-convex, multi-modal function with 25 local minima. The total search space consists of $131072^2 \approx 16 * 10^9$ alternative solutions. The global minimum is $f_5(x_i) = 1.0$.
The results of a sequential implementation of a GA instance of ACGA are shown in Figs. 24 and 25. The experiments were done on an Intel 80486 computer. The plotted lines in both figures are calculated by averaging the fitness of the best individual in the population in 30 experiments. In both figures the convergence of the standard GA (dotted line) is compared to the C-GA (solid line).

3.2.5 ACGA on a parallel architecture

In this section we have developed a generic formal description for a parallel SA and a parallel GA by mapping them onto Cellular Automata. A formal model allows a mathematical and empirical analysis of the underlying algorithms. The convergence behavior of GA is still poorly understood. It is known that convergence of different GA algorithms may deviate and that convergence is also strongly influenced by the shape of the search space. GA and SA are attractive natural solvers for optimization problems but parallelization of these algorithms is an important prerequisite. The first test results of a sequential implementation show that when the parallel GA is compared to the standard GA, a faster convergence is obtained, despite the use of different shaped search spaces. These results show the validity of the model: it is possible to introduce locality without loosing convergence speed, an optimum obtained even faster. Experiments with C-SA are discussed in [93].
Future work includes the implementation the ACGA on a parallel architecture to fully exploit the parallel nature of the presented model and to test the scalability of the model. The formal model will be crucial in proving asymptotic global convergence. For now empirical tests are necessary to determine the finite time convergence behavior, which is the quantity that really matters for practical purposes. We intend to use the *Mapping Problem* (see section 4.4) as a major test case for the C-GA. We are especially interested in the performance/scalability of the algorithm using different ranges of locality.

4 Case studies

4.1 Diffusion Limited Aggregation

Many growth phenomena, for example the growth process of a bacteria colony, viscous fingering, electric discharge patterns (see section 2.2.3) and growth forms of electro deposits, can be simulated with one model: the Diffusion Limited Aggregation model [91]. At the heart of all these growth patterns there is one Partial Differential Equation,

$$\nabla^2 c = 0 \tag{104}$$

the Laplace equation, which describes the distribution of the concentration c , pressure, electric potential etc. in the environment of the growth pattern. First we will discuss the numerical solver for such a system, then the natural solver and finally parallelization aspects for the natural solver.

The numerical solver: Finite differencing
The Laplace equation can be solved numerically and a DLA cluster can be constructed using the nutrient distribution over the lattice. The cluster is initialized with a seed and the following boundary conditions are applied: $c = 0$ on the cluster itself and $c = 1$ at the nutrient source, which in itself may be circular, linear etc. The probability p that a perimeter site (the sites indicated with

an open circle in Fig. 26 with index k will be added to the DLA-cluster (black circles in Fig. 26) is determined by

$$p(k \in \circ \rightarrow k \in \bullet) = \frac{(c_k)}{\sum_{j \in \circ}(c_j)} \tag{105}$$
$$\text{where } c_k = \text{ concentration at position } k$$

The sum in the denominator represents the sum of all local concentrations of the possible growth candidates (the open circles in Fig. 26). The DLA cluster is constructed using the following rules:

1: solve the Laplace equation (Eq. 104), using the boundary conditions.
2: new sites are added to the cluster are added to the cluster with probability p (Eq. 105).
3: goto 1

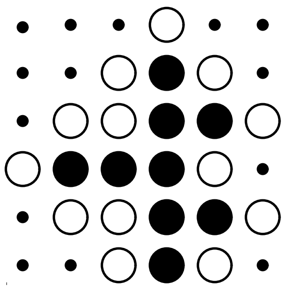


Figure 26: *First steps in the construction of the DLA-cluster. Sites which are part of the cluster are visualized as black circles, sites which are possible candidates to be added to the cluster in next iteration steps are indicated with open circles.*

The Laplace equation can be solved, using the boundary conditions mentioned above, by finite differencing and the successive over-relaxation method:

$$c_{i,j}^{n+1} = \frac{\omega}{4}(c_{i-1,j}^{n+1} + c_{i,j-1}^{n+1} + c_{i+1,j}^{n} + c_{i,j+1}^{n}) + (1-\omega)c_{i,j}^{n} \tag{106}$$

In this method the new local nutrient concentration $c_{i,j}^{n+1}$ in the lattice, at a site with lattice coordinates i, j , is determined in an iterative procedure which converges as soon as the difference between the new and old local nutrient concentration is below a certain tolerance level. The ω in Eq. 106 is the over relaxation parameter, which in general lies within the range $1 \leq \omega < 2$. After many construction steps this procedure results in a DLA-cluster as shown in Fig. 27

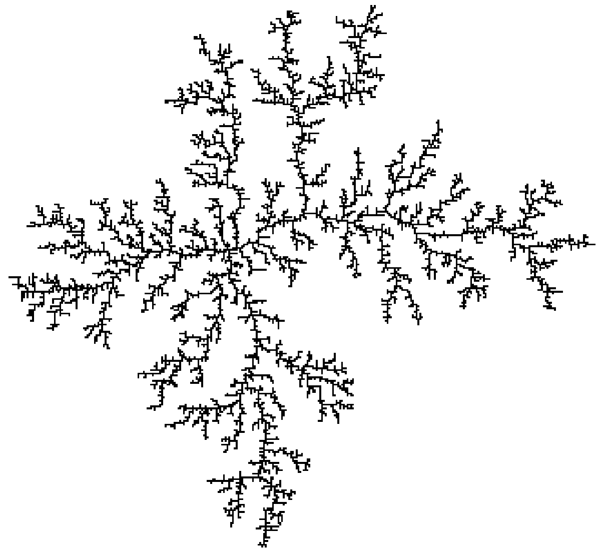


Figure 27: *DLA-cluster generated using a circular source of nutrient*

The natural solver: The random moving particle model
An alternative method to construct the DLA-cluster is a probabilistic cellular automaton which resides on a square two-dimensional or three dimensional lattice. The growth pattern can be constructed using the following Monte Carlo approach: The first step in the construction is to occupy a lattice site with a seed. After that, particles are released from a source which might be circular shaped (using the seed as a center) at a large distance from the seed. The particle starts a random walk, the walk stops when the particle leaves the circle or reaches a perimeter site of the seed and sticks. Then more random walkers are released from the source and are allowed to walk until the distance with respect to the cluster with occupied sites becomes too large or it reaches a perimeter site, neighboring to one of the previous particles, and it sticks. When this procedure is repeated many times a similar irregular growth pattern as shown in Fig. 27 is generated. It can be demonstrated that in this Monte Carlo method the underlying Laplace equation is correctly solved [91].

The parallel implementation of Diffusion Limited Growth
Modeling and simulation of Diffusion Limited Growth, especially in the 3D case, is computationally very expensive. The development of parallel growth models, especially in the case of DLA is not straightforward (see also references [61] [63]). The computationally most expensive step, step 1 solving the Laplace equation, of the numerical solver can be done in parallel. In a parallel implementation of Eq. 106 using SOR, the update order of the lattice has

to be taken into account. When the lattice site (i, j) is updated in the parallel implementation and is located at the border of the processor (see Fig. 28), its neighbors should be in the correct state (n or $n+1$). A parallel implementation, with a correct update order, can be made using the checkerboard strategy [27]. In this method the sub-lattice on one processor is subdivided into four blocks. In the parallel SOR update first all red regions are updated, followed by the black regions. Step 2 in the numerical solver, adding sites with probability p is an implicit sequential step.

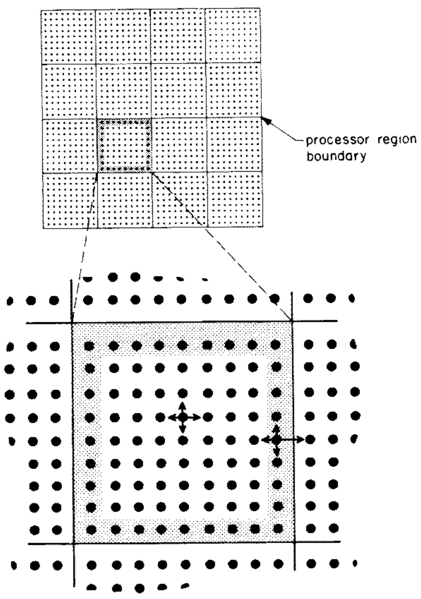


Figure 28: *Mapping of the lattice onto a grid of 4 x 4 processors, the shaded sites are situated in a region where information from neighboring processors is required for the update (after [27]).*

The attractive property of the random moving particle model is that it illustrates that the locality present in the model can be exploited in all steps of the algorithm. Both solving the Laplace equation and growth of the cluster by the addition of sites can be done in parallel. A major question is if such a parallel version of random moving particle model mimics the same physics as its sequential version. In a forthcoming paper [95] we will discuss this question in detail.

4.2 Lattice Boltzmann solver for flow and diffusion phenomena

An alternative approach to model physical systems in fluids and gases, for example complex flows, complex fluids, and reactive systems, are lattice gas methods. The basic idea is illustrated in Fig. 29. The behavior of the fluid surrounding the obstacles can be modeled as a continuum (Fig. 29A) using a set of partial differential equations, as a molecular liquid (Fig. 29B) using a Molecular Dynamics method, or alternatively using a discrete velocity (lattice) gas.

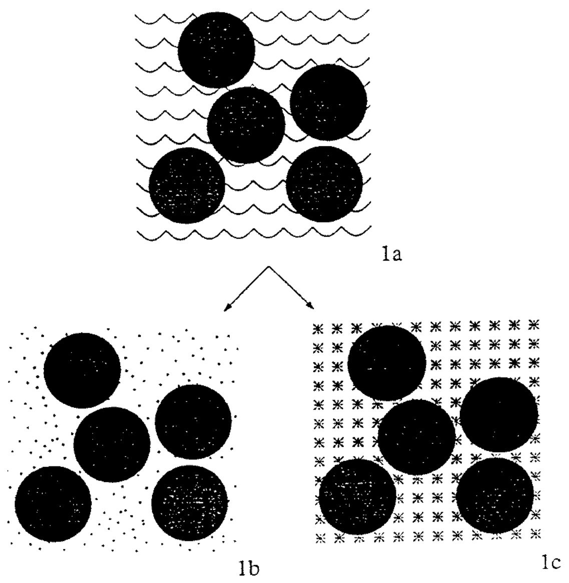


Figure 29: *Three different approaches to model the fluid surrounding the obstacles: the fluid is modeled using a) a set of partial differential equations b) molecular dynamics c) a lattice gas (after [56])*

A very simple lattice gas method with which an incompressible fluid in 2D can be modeled is the FHP Lattice Gas Automata (LGA) [31]. In this method the fluid is represented by "particles" which can move from a lattice site to neighboring lattice site. The particles reside on a triangular lattice and are located in the centers, the nodes, of the hexagons in the lattice. Each particle, or lattice site, is connected with six neighbors and can move in a next time step to one of these neighbors, or can be stationary. In the FHP model both time and velocities are discretized. Two successive time steps of the FHP model are shown in Fig. 30. As soon as the particles are at the point to occupy the same node a collision occurs. In order to obtain a physically correct model, in which mass and momentum are conserved, collision rules have to be included. Updating

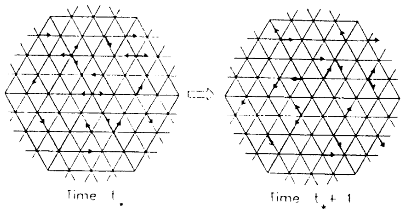


Figure 30: *Two successive time steps of the FHP model (after [31])*

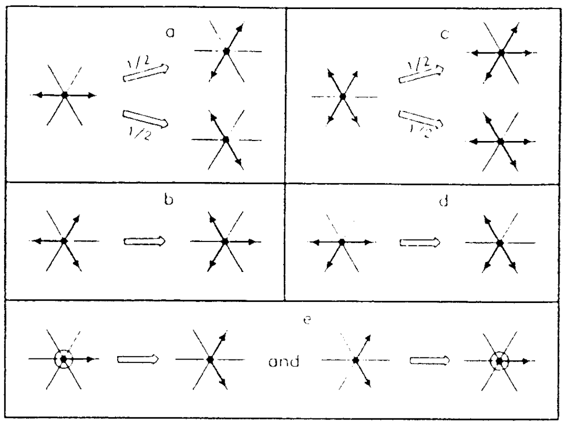


Figure 31: *Collision rules applied in the FHP model (after [31])*

of the lattice now involves a propagation step and collision handling. In Fig. 31 a simple version of the collision rules, used in the FHP model, is shown. Note that in the FHP model both velocities and time are discretized. Many types of boundary conditions can be relatively easy specified. For example the rigid no-slip condition can be specified by setting the lattice sites which represent the obstacle, to the state "stationary". The lattice gas method is especially suitable to determine flow patterns, for low velocities, about obstacles with a complex geometry. An example of such an experiment is depicted in Fig. 32 where flow about an irregular object is shown. The discreteness of the lattice gas approximation introduces noise into the results. In the computation of flow patterns this discreteness artifact can be partially avoided by taking spatial averages for the velocities. In practice the average velocity u in a 2D lattice gas model is determined in clusters with a minimal size of 64×64 nodes [21]. The limited range of velocities also restricts the allowed range of Reynolds numbers which can be simulated with the lattice gas method.
Due to the large scale separations it is not necessary to use a model which includes the detailed mechanics of the molecules surrounding the obstacles. The

fluid can be modeled with a lattice gas, the "particles" in the fluid are represented on a much higher magnitude scale than the actual molecules. It can be demonstrated that lattice gas methods still include most features of a fully molecular simulation, while it is order of magnitude faster in computation. The lattice gas methods fill the gap between MD and PDE methods, the LGA method is sometimes described as the "poor man's MD" [21].

4.2.1 Macroscopic equations

The flow of a time dependent incompressible fluid, a fluid with constant density as for example water under most conditions, can be described by the two basic macroscopic equations from hydrodynamics:

$$\nabla \cdot (nu) = 0 \tag{107}$$

$$\frac{\partial nu}{\partial t} + \mu n(u \cdot \nabla)u = -\nabla p + \nu n \nabla^2 u \tag{108}$$

where u is the velocity, n the particle density, t time, p the pressure, and ν the kinematic viscosity which will be assumed to be constant. The coefficient μ is usually constrained to have value 1 by Galilean invariance. For low velocities it can be demonstrated that the FHP model approximates the Navier Stokes Eq. 108 (see [31]):

$$\partial_t(u) = -(g(\rho) \cdot u \cdot \nabla)u - \nabla p + \nu \nabla^2 u \tag{109}$$

where ρ is the average number of particles per cell and $g(\rho)$ a function of this density ρ. The FHP model is not Galilean invariant: the parameter μ in Eq. 108 does not equal to the value 1. In the FHP model there exists a velocity dependency of the density $\rho : p = \frac{1}{2}\rho(1 - \frac{1}{2}u^2)$. Regions with constant pressure p and a high velocity p have a higher density ρ than regions with a lower velocity. This does not correspond to physical properties of an incompressible fluid which should have constant density everywhere. For low velocities this velocity dependency of ρ does not have strong effects. When in Eq. 109 the time t, the viscosity ν, and the pressure p are scaled with the factor $g(\rho)$ ($t' = g(\rho)t$, $\nu' = \nu/g(\rho)$), the equation transforms into the Navier Stokes equation.

4.2.2 Lattice gases in 3D

There also exist 3D extensions of the FHP-model [30]. In the 3D extension, the Face Centered hypercube (FCHC) geometry, the particles reside on a cubic lattice (see Fig. 33) where each node is connected with 24 other nodes. Along the dotted links between the nodes at most one particle can propagate, while at the solid links up to two particles can propagate. In practice this FCHC method suffers from several serious problems: to avoid discreteness effects by spatial averaging an enormous lattice is required and the implementation of

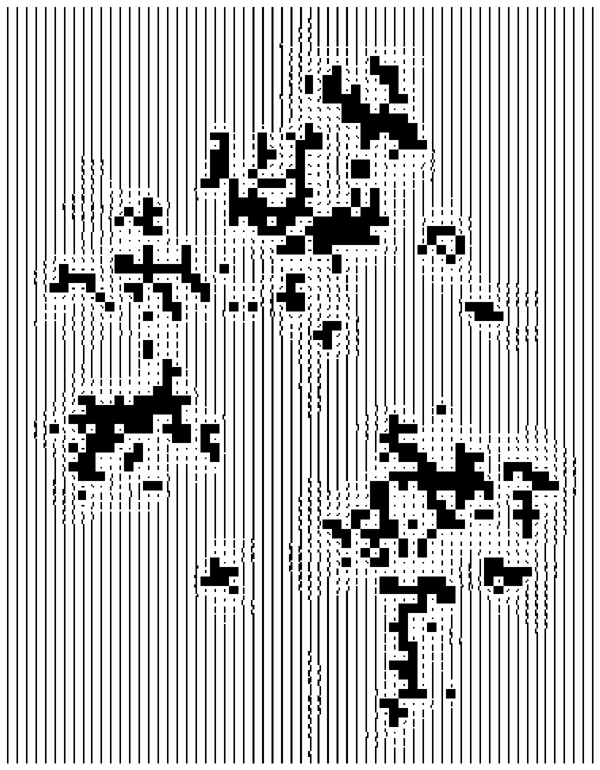


Figure 32: *Flow pattern around an irregular shaped obstacle using the lattice gas method*

the collision operator $\omega(n_j(x,t))$, due to large amount of possible collisions, is troublesome [105].
Part of the problems of LGA in 3D can be overcome by using an alternative lattice gas method based on the Lattice Boltzmann Equation (LBE). The difference with the LGA-method is that no longer individual particles are followed, but the average values are used. In the LBE method the velocities are no longer discrete, but represented in continuous space and the noise introduced by the discreteness of the LGA method is removed. Furthermore the problem of the velocity dependency of the density (Eq. 109) can be solved in the LBE method.

4.2.3 Theoretical background of the Lattice Boltzmann method

In the simulations a cubic lattice is used, where each node is connected with 18 other nodes. There are 6 links with the length of one lattice unit and 12 links with the length of $\sqrt{2}$ units. The mean populations of particles travel simultaneously from one node to one of the 18 neighbors. The evolution of the

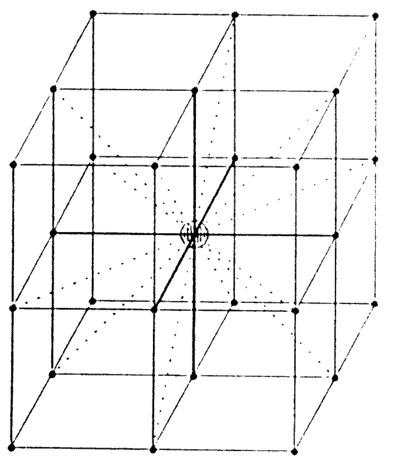


Figure 33: *Face Centered hypercube (FCHC). Along the dotted links between the nodes at most one particle can propagate, while at the solid links up to two particles can propagate.*

lattice is described by the following dynamical rule:

$$n_i(r + c_i, t + 1) = n_i(r, t) + \Delta_i(r, t) \qquad i = 1, .., 18 \tag{110}$$

where $n_i(r, t)$ is the continuous velocity distribution function, which describes the number of particles of particles at a node at position r, at time t and with direction c_i. The moments of the distribution function $n_i(r, t)$:

$$\begin{aligned} \rho &= \sum_i n_i \\ j &= \sum_i n_i c_i = \rho u \\ \Pi &= \sum_i n_i c_i c_i \end{aligned} \tag{111}$$

correspond to the hydrodynamic fields mass density ρ, momentum density j (u is the flow velocity at node r), and momentum flux Π.

A linearized collision operator $\mathcal{L}_{ij}$ about an appropriate equilibrium n^{eq} can be constructed:

$$\Delta_i(n) = \Delta_i(n^{eq}) + \sum_j \mathcal{L}_{ij}(n_j - n_j^{eq}) \tag{112}$$

where $\Delta_i(n^{eq}) = 0$ by definition, since the equilibrium is collision invariant and where n^{eq} is the Maxwell Boltzmann distribution for a gas. The distribution function can be split into an equilibrium and a non-equilibrium part:

$$n_i = n_i^{eq} + n_i^{neq} \tag{113}$$

Similar to Eq. 111 the moments of the equilibrium distribution function n_i^{eq} are:

$$\begin{aligned} \rho &= \sum_i n^{eq} \\ j &= \sum_i n^{eq} c_i \\ \Pi &= \sum_i n^{eq} c_i c_i = p1 + \rho uu \end{aligned} \tag{114}$$

where p is the pressure, the driving force of the system, and 1 is the unit vector. The particle collisions conserve mass and momentum but change the non-equilibrium part of the momentum flux:

$$\Pi^{neq} = \Pi + \Pi^{eq} \tag{115}$$

By using properties as conservation of mass, momentum, and symmetry, it is possible to construct a number of eigenvalue equations (see for a more detailed account of the collision process [56]) for the moments of $\mathcal{L}_{ij}$:

$$\begin{aligned} \sum_i \mathcal{L}_{ij} &= 0 \\ \sum_i c_i \mathcal{L}_{ij} &= 0 \\ \sum_i \overline{c_i c_i} \mathcal{L}_{ij} &= \lambda \overline{c_j c_j} \\ &\ldots \end{aligned} \tag{116}$$

The first two equations conserve mass and momentum. The eigenvalue λ is related to the shear viscosity:

$$\nu = -\frac{1}{6}\rho(2/\lambda + 1) \tag{117}$$

One computational update of the lattice now involves a propagation step in which n_i's travel from node r to node $r + c_i$ and a collision step in which the post-collision distribution $n_i + \Delta_i(n)$ is determined in the following three steps. First at each node ρ, j, and Π are calculated and the equilibrium momentum flux Π^{eq} is determined. In the second step the momentum flux, including the equilibrium and non-equilibrium part, can be updated using the moments of the linearized collision operator in Eq. 116. In the third step, after the post collision momentum is computed, the post-collision distribution $n_i + \Delta_i(n)$ can be calculated.

In simulations several types of boundary conditions can be used: at the borders of the lattice periodic boundary conditions can be applied, while in the nodes adjacent to the nodes representing the obstacle, solid boundary conditions can be used. Periodic boundary conditions can be implemented by exchanging the n_i's of the links at the borders of the lattice. Solid boundary conditions can be represented by exchanging the n_i's between the adjacent node and a neighboring fluid node. In the boundary nodes, near the obstacle, it is also possible to determine the force exerted by the fluid on the obstacle. Changes in this force are used in the simulations to detect if the flow pattern around the obstacle stabilizes and the lattice Boltzmann iteration converges.

After the lattice Boltzmann iteration a diffusion step can be applied where populations of tracer particles are released from certain source nodes, while the tracer particles are absorbed by sink nodes. The tracer particles can move from a node at site r in the lattice to one of the 18 adjacent nodes $r + c_i$, where the ratio of viscous forces to Brownian forces, the Péclet number

$$Pe = \frac{\nu}{D} \tag{118}$$

determines if flow or diffusion dominates.

4.3 A comparison: Lattice Gases, Lattice Boltzmann method, and PDE's

In the numerical solver the physical problem is modeled using partial differential equations, and then mapped onto a discrete model a finite differencing scheme in which both time and space are discretised. This finite differencing scheme falls within the formal definition of a cellular automaton and can be indicated as a computational CA for an independently formulated model represented by partial differential equations [41]. This is a fundamental difference with the lattice gas methods (LGA and LBE) where the problem is directly modeled using a cellular automaton. By applying mass and momentum conserving rules in the CA, a lattice gas can be used as a model of hydrodynamical phenomena. This difference has important consequences for the preservement of the physical characteristics of the original problem.
The most ideal form to represent a hydrodynamical phenomenon would be a MD model, within a MD model all physical properties remain preserved. In the numerical solvers these physical properties are lost in the mapping of the PDE's onto the computational CA. In the LGA method physical properties related to a many particle system, as for example particle correlations, remain preserved. For this respect we can arrange the different methods from microscopic to macroscopic: MD, LGA, LBE, PDE. This arrangement starts with

MD, where from a model on a microscopic level, the desired macroscopic information is derived. The PDE's represent the other extreme and are based on a macroscopic model. The LGA and LBE method can be classified as intermediate models, where from a mesoscopic model the desired macroscopic information is derived. In this sequence LBE should be arranged behind LGA, since by the transformation of the particle system into a population - of - particles system, physical properties as particles correlations are lost. This makes LBE less microscopic than LGA.

An important advantage of the lattice gas methods is that the computational code is much simpler than the code required for most of the numerical solvers. In lattice gas methods there are no problems with round off errors, which may cause instabilities in the numerical solvers. Unfortunately the problem of instabilities re-occurs in the LBE method, since in the dynamical rule Eq. 110 continuous values are used. It is possible to conserve momentum and mass exactly and the lattice gas methods do not suffer from slow converging iteration processes. For some problems, in which highly irregular geometries are involved as for example flows through porous media, the LBE method is the only available solver. In [13] the LBE method is compared to a numerical method, the spectral method, which is still considered a superior and efficient method for hydrodynamical problems.

In many biological systems the distribution of chemical agents and nutrients, caused by a combination of flow and diffusion, plays a fundamental role. An example is morphogenesis where the external distribution of nutrients [32] [47], as well as the internal distribution of morphogens [75] [69] by reaction-diffusion in the organism, can induce certain growth patterns. The Lattice Boltzmann method is especially suitable for modeling diffusion and flow phenomena in biological systems with highly complex boundary conditions. The Péclet number (the ratio of viscous forces to Brownian forces) and the Reynolds number (the ratio of inertial forces to viscous forces) can be varied independently [56],[48]. Because of the inherently parallel properties of the LBE method it can be expected that with the development of massive parallel computers, LBE will become a competitive solving technique [54], [49]. The fact that specification of boundary conditions with an irregular geometry is problematic in the spectral methods will be another argument in favor of LBE.

4.4 FE - Applications Formalized by the VIP Model

4.4.1 The Finite Element Technique

The finite element method is a widely applied technique in all kinds of research areas. An example of such a research area is structural mechanics. In that case the elastic and deformation characteristics of a complex shaped object are computed in their time evolution. One may think of calculating vibrational properties of buildings or e.g. car crash simulations.

The finite element method approximates the shape of a real-life object by a set of discrete elements that are interconnected. The physical properties of these individual elements are well understood. Simply said, the better the FE mesh covers the object, the better numerical results can be expected.
The basic equation that describes time evolution of a linear elastic finite element system is the equation of motion (119).

$$M\mathbf{a} + C\mathbf{v} + K\mathbf{d} = F \tag{119}$$

Where:

$$M = \int_V N^T \rho N dV \tag{120}$$

$$C = \int_V N^T \mu N dV \tag{121}$$

$$K = \int_V B^T D B dV \tag{122}$$

a is the acceleration vector, **v** is the velocity vector **d** is the coordinate vector.
M is the so called mass matrix of the system, C describes the viscous properties and K corresponds to the stiffness within the FE system. Furthermore F is the external force vector acting on the system.
In the integrals we consider integration over the systems volume V, N describes the numerical approximation (linear, polynomial) of individual elements, ρ is density in the system, μ is a friction coefficient, and finally B describes elastic-strain in the system.

4.4.2 FE as VIP

The FE mesh consist of a set of elements that have local interactions. The concept of the VIP model is well suited as a formalization of such an element mesh. The granularity of general finite element meshes is very small. When parallelization is considered it is therefore not realistic to parallelize on the finest grain level. A grain size has to be selected that better matches that offered by a parallel architecture. Therefore partitioning techniques that decompose the original FE mesh into clusters of elements are necessary. The VIP formalization can transparently be applied to the "element-clusters", which are just finite elements, but on a hierarchically higher level. Special attributes to the element-clusters are the interaction size (or connectivity) and the volume. These parameters are essential when optimal usage by the parallel FE application of parallel architectures is to be studied.

4.4.3 parallel architectures as VIP

A parallel architecture can also be described using a VIP formalization, in the same manner as for the FE system. Now we have an interconnected set of

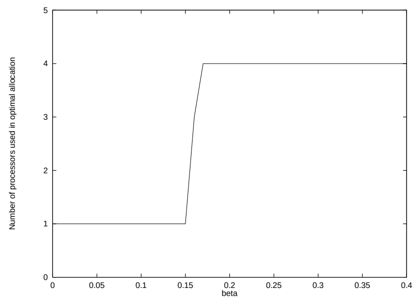


Figure 34: *A phase transition in the mapping process*

processors. The processor-processor connections are analogous to the FE interconnectivity, while the processor power is analogous to the element-cluster grain size or volume.

4.4.4 Mapping FE - Parallel architecture

In order enable a FE application to make optimal use of a given parallel machine one has to investigate how the general attributes of the FE-mesh (volume, connectivity) and the general attributes of the processor topology (processing power, connectivity) can be mapped on each other see also section 3.1. This mapping problem is in general NP-hard, which means that we have to consider solving an effectively intractable problem. No tractable solutions are known, hence an approximation approach has to be used. Some of the best suited optimization techniques that can be used fall into the class of Natural Solvers, e.g. genetic algorithms, neural networks. We have used a GA approach to map the element-clusters of a given FE mesh onto arbitrary topologies[19][18]. Certain phenomena that are often observed in DCS, such as *frustration*, can also be found in the process of mapping a FE element-cluster onto a parallel architecture. The optimization process is guided by a selection procedure which is biased to an evaluation function. An example of such an evaluation function is:

$$C = \beta\sqrt{(}\sum_{n=0}^{N_p-1} W^2(n)) + \frac{1}{2}\sum_{n=0}^{N_p-1}\sum_{m=0}^{N_p-1} C(n,m) \tag{123}$$

where

$W(n)$: Total time spent on calculation work on processor n

$C(n,m)$: $B(n,m) * M(n,m)$ where

$B(n,m)$: $\frac{1}{bandwidth}$ of route between processor n and m.

$M(n,m)$: Number of bytes to be sent from processor n to m.

The parameter β is used to indicate the importance of the workload relative to the communication. Varying β results in different optimal mappings. Furthermore it can be observed that in general only 1 or all processors are used in the mapping, there is only a small region in which a different number of processors is used. Hence, we can interpret the number of processors as a kind of order parameter in the mapping process, while β acts as the "temperature" of the system. The convergence of the GA increases in the neighborhood of this region, which indicates the presence of critical slowing down as observed in second order phase transitions. The observed phase transition is between a 1-processor mapping and an all-processor mapping. An example of such a phase transition is given in figure 34, where a torus mesh is mapped onto a fully connected processor topology. The phase transition indicates the onset of a complex region in which correlation of length and time scales can occur, because the competition between the work load optimization and communication minimization is very strong. There are a lot of suboptimal solutions which are very close to each other in terms of the cost value of Eq. 123. Note the resemblance of the complexity region at the phase transitions in the discussed systems like Diffusion Limited Aggregation (η as the growth "temperature") and complex dynamics in Cellular Automata (λ as activity parameter).

5 Conclusions and future work

In the previous sections we have demonstrated that several instances of natural DCS can be captured with a natural solver. Diffusion Limited Aggregation can be modeled with a multi-particle model, while flow and diffusion can be modeled with the Lattice Boltzmann solver. With the Lattice Boltzmann solver it is even possible to model some phenomena (for example complex fluids and diffusion and flow phenomena in highly complex geometries, modeling flow and diffusion by varying the Péclet and Reynolds number independently) which cannot be modeled using a numerical solver. In both cases the particles in the natural solver correspond to the particles in the VIP model, allowing for a straightforward mapping onto the Parallel Virtual Machine model. In some classes of natural solvers, as for example the genetic algorithms, it is necessary to introduce locality in order to obtain an efficient mapping onto the Parallel Virtual Machine model. In the case study on genetic Algorithms this locality was introduced by applying a Cellular Automaton representation of the GA as a VIP model. In some models of natural systems it is necessary to cluster the VIPs hierarchically to obtain an efficient mapping of the application model onto the machine model. By allowing to cluster the VIPs hierarchically it becomes possible to reduce the complexity of the problem , while the information about the error that is introduced in this approximation is conserved. In the section on the Finite Element models the VIPs (clusters of Finite Elements) are allowed to cluster hierarchically. In this case-study it is also demonstrated the necessity of a"Holistic" approach: to obtain an optimal mapping of the DCS of the application model onto the machine model without loosing information in the mapping process, both knowledge of the solver and the machine architecture are required.

Within the Parallel Scientific Computing and Simulation Group research on modeling and simulation of DCS on the application and machine level, is a central theme. On the application level research is being carried out on modeling and simulation of DCS from physics and biology in case-studies, where more insight in these systems can only be obtained through explicit simulation, as well on the theoretical framework of intractability and undecidability of DCS. On the application level 5 projects within the Parallel Scientific Computing and Simulation Group are being done:
a) The FOM (Foundation of fundamental Research of matter) project "Crystallization on a sphere", in which crystallization processes with spherical boundary conditions are studied [108] [109]. The VIP model in this study are particles on a sphere, the particles are abstractions of phospholipids or other macromolecules in spherical vesicles.
b) Contribution to the Esprit project "CAMAS Computer Aided Migration of Application Systems". In this project research is done on the most efficient mapping of Finite Element models on a Virtual parallel machine [19]. The VIP model in this study consists of clusters of Finite Elements [19]
c) The biophysics project "Elastic Light Scattering" in which light-scattering properties of white blood cells are studied. The VIP model model in this study consists of dipoles which are large, abstract, clusters of molecular and atomic dipols. [40]
d) The NWO (Dutch Science Foundation) project "Portable parallel simulation of nucleation and growth" in which diffusion and flow limited growth processes are studied. The VIP model is the Lattice Boltzmann solver. [48]
e) The locally supported project "Dynamic Complex Systems" in which methods are developed for the formal description of a DCS and research is done on intractability and undecidability of DCS [95]

On the virtual machine level research is done on the abstraction of the hardware model and the runtime support system. Four project are being carried on this level:
a) The EC project "ParaSol" (contribution to the Esprit project "CAMAS Computer Aided Migration of Application Systems") in which time complexity description and analysis methods for, existing, FORTRAN-code are developed. Time complexity analysis is only possible through explicit simulation of the behavior of clusters of instructions of FORTRAN-code, similar to simulations of clusters of VIPs in the application model.
b) Externally supported project "Dynamic PVM" a project in which research is being done on automatic load balancing using dynamic resource management of parallel tasks in a job[20][79][78].
c) Locally supported project "Simulation of a DCS through discrete event simulation". In this project parallel discrete event simulations are studied, using the Time Warp paradigm to circumvent causality errors which can occur as a

result of distributed generation of events [77].
d) Tools for developing parallel implementations of simulation models of DCS:
d1) Enhancement of communication kernels to support message passing computation (MPI, PVM).
d2) Automatic domain decomposition methods.
d3) MAP: tool for mapping a decomposed problem onto the machine.

In three forthcoming MPR-projects (Massive Parallel Computing projects funded by the Dutch science foundation) "Parallel I/O and imaging", "Portable parallel simulation of crystal nucleation and growth", and "The discreteness-continuity dichotomy in individual-based population dynamics using massively parallel machines" we are planning to bring both lines of research together. In a number of case studies we intend to test our ideas, as presented in this report, on modeling and simulation of DCS from nature.

Acknowledgments

This working document was prepared at the Santa Fe Institute in New Mexico in the summer of 1997. P.M.A. Sloot wishes to thank Melanie Mitchell and Jim Crutchfield for hosting him and thanks the Santa Fe Institute and its researchers for the scientifically stimulating visit. This research was supported in part by grants from the Dutch Society for Scientific Research (NWO) in the Massive Parallel Computing priority theme and by a European Committee grant for High Performance Simulation.

6 Appendix: Computability in Dynamical Complex Systems

In this section we will review the concepts of *Undecidability* and *Intractability* as used in formal machine models, in order to apply these concepts in real systems, e.g. physical, biological. The underlying motivation is to integrate our current knowledge of formal computation with the notion that physical processes can be viewed as "computations".

6.1 Undecidability and Intractability

6.1.1 Church-Turing Hypothesis

The intuitive notion of an algorithm can be formalized by a *Turing Machine* (TM) (Alan Turing, 1912-1954). It is assumed that any way of formalizing an algorithm is equivalent to formulating a Turing Machine. Formally the basic model consists of a finite control, an input tape and a head that can be used for reading or writing on that tape. A Turing Machine can execute an algorithm defined by a *string* on the tape. The string is constructed from a finite alphabet.

A TM can decide whether a given string is an element of a *language*. A language is here defined as a restricted set of strings. A Turing Machine can both be used for deciding membership and for computing functions (basically the two operations are identical). A language $L(M)$ that is accepted by a TM M, is said to be *recursively enumerable* (r.e.). The class of r.e. languages includes some languages $L(M)$ for which there does not exist an algorithm to determine the membership of a string, i.e. we can not tell whether the machine M will eventually accept the string if we let it run long enough, or whether M will run forever [44]. A subclass of the r.e. languages are the *recursive sets*, which are those languages accepted by at least one TM that halts on all inputs.
We can also view a Turing Machine as a computer of functions from integers to integers. Again we can identify two different classes. First, the so called *total recursive functions*, which are those functions that are computed by Turing Machines that halt on all inputs. For example all common arithmetic functions, like multiplication, addition, etc, are total recursive functions. The analogon of r.e. languages in the computation of functions are called *partial recursive functions*, i.e. those functions that are computed by Turing Machines that may or may not halt on a given input.
With this background information we can now define the *Church-Turing Hypothesis*, which states that the intuitive notion of an algorithm can be identified with the class of partial recursive functions and consequently computed by a Turing Machine.
Currently we cannot prove this hypothesis because of the informal notion of "computable". If we place no bound on the resources (time and storage), the partial recursive functions are intuitively computable. It is difficult to tell whether the class of partial recursive functions includes all "computable" functions. Many other formalisms that have been presented, λ-calculus, Post systems, etc, but all are equal to the class of partial recursive functions[44].

6.1.2 Undecidability

In 1931 the logician Kurt Gödel discovered an incompleteness in a axiomatic system of number theory. His theorem informally states that there are certain statements about numbers that cannot be proved within the axiomatic formalism. Gödel's original proof of the incompleteness theorem is based on the paradox of the liar: "This statement is false". By changing this statement to: "This statement is unprovable", he obtains a theorem instead of a paradox. If this assertion is unprovable, than it is true and the formal system is incomplete. If this assertion is provable, then it is false and the formal system is inconsistent. The original proof of Gödel was based on a specific formalization of number theory and was followed by a paper showing that the same methods applied to a much broader class of formal axiomatic systems. At that time he could not yet proof it for all axiomatic systems, due to the lack of a mathematical definition of an algorithm. After the definition of the Turing Machine it became possible to proceed in a more general fashion. In turn Turing showed that there is no algorithm for deciding whether or not a program ever halts, i.e. the *halt-*

ing problem is unsolvable. One can derive Gödel's theorem from the halting problem[11]
Using a technique called *diagonalisation* it is possible to construct a language that is not accepted by any Turing Machine [44]. In this technique an undecidable language is constructed that is not in the list of all systematically enumerated languages. Hence non-computable languages *do* exist.
It is a fact that any nontrivial property of r.e. languages are undecidable. The latter fact is known as Rice's theorem [44] and it actually states that virtually any question about the set H of sequences on which a Turing Machine will eventually halt is undecidable. These questions include questions whether H is empty, finite, regular, context free, etc.
Using Rice's theorem, we already make the observation that if we observe a physical phenomenon that is able to mimic a universal TM then most physical questions concerning idealized limits of infinity require arbitrary long computations and so be *formally undecidable*. Shortly, we will return to this observation. First, we will address another unpleasant property namely that some problems are so "complex", that it takes exponential time to solve them.

6.1.3 Intractability

Computational complexity describes the amount of time, space or other resources needed to accept a language or to solve a problem.
The space complexity $S(n)$ denotes the maximum number of cells that a TM M will scan for an input of length n. The language recognized by M is said to be of space complexity $S(n)$.
The time complexity $T(n)$ denotes the maximum number of moves that M will make for every input word of length n. M is said to be of time complexity $T(n)$.

In the time complexity regime we define two classes: P and NP. The class P is informally defined as the class of those languages for which there is a polynomial time deterministic TM. The Class NP is informally defined as the class of those languages for which there is a polynomial time non-deterministic TM. Furthermore, a language or a problem belongs to NP if and only if a given string or solution can be checked or verified in polynomial deterministic time, see also[10]. We can now define an NP-*complete* problem $L^{'} \in \mathrm{NP}$ as a problem to which every other problem $L \in \mathrm{NP}$ can be *polynomially reduced*. A problem L can be polynomially reduced to another problem $L^{'}$ if there exists a polynomial time transformation from an instance in L to an instance in $L^{'}$. Next we have to find the first NP-complete problem. Therefore we can not use the above mentioned approach, simply because we do not have a known problem to transform to. A common trick to prove $C-completeness$ is just to mimic a specific machine that accepts an arbitrary language $L \in C$, where C stands for a specific complexity class, i.e. P or NP. This is also the approach that can be used in deciding completeness results for certain physical phenomena. Thus in order to prove NP-completeness for our first language, we must show that this language is able to mimic the computation of an arbitrary language in NP on a

non-deterministic TM in polynomial time. The first problem for which this was done is the so-called *Satisfiability* citeBovet94[33]. In short, to show that every language in NP is reducible to *Satisfiability*, for each non-deterministic TM M that is time bounded by a polynomial $p(n)$, construct a polynomial-time algorithm that takes as input a string x and produces a Boolean formula E_x, which is `true` if and only if M accepts x.
Another important class of languages are called NP-hard. A language L is called NP-hard if all problems in NP can be reduced to it, but L is not necessarily in NP.
In the space complexity regime we can define the classes PSPACE and NPSPACE. The class PSPACE is informally defined as the class of those languages that are accepted by a polynomial space bounded deterministic TM that halts on all inputs. NPSPACE is defined as PSPACE, but now for non-deterministic TM's. As opposed to the time complexity classes P and NP, it is possible to proof [44][33] that PSPACE = NPSPACE. Hence deterministic polynomial space is as powerful as non-deterministic polynomial space. We can now define the class of PSPACE-complete problems: a language $L^{\text{`}}$ is PSPACE-complete, if $L^{\text{`}} \in \text{PSPACE}$ and for all $L \in \text{PSPACE}$, L can be polynomially reduced to $L^{\text{`}}$. It follows that if L is PSPACE-complete, then $L \in P$ if and only if P = PSPACE and $L \in$ NP if and only if NP = PSPACE. Note that we could have P = NP even if P $\neq$ PSPACE, indicating that if a problem is PSPACE-complete, it is an even stronger indication that it is intractable, than if it where NP-complete [33]. Note that every known NP-complete problem is also in PSPACE. In order to proof that a problem $L^{\text{`}}$ is PSPACE-complete we can either use the reducibility technique to a known PSPACE-complete problem, or we must proof that every language $L \in \text{PSPACE}$ is polynomial-time reducible to $L^{\text{`}}$. In short we must proof that an arbitrary polynomial-space bounded TM M can be simulated by $L^{\text{`}}$ in polynomial time. Note that all problems solvable in polynomial time can be solved in polynomial space. Or in other words, $L^{\text{`}}$ must be able to simulate a universal polynomial-space bounded TM, i.e. a TM which can compute every polynomial-space bounded function. The first problem that was proven PSPACE-complete is called Quantified Boolean Formulas (QBF)[33].
Analogous to the NP-hard problems, we can define a PSPACE-hard problem L as a problem to which every other problem in PSPACE can be reduced, but L is not necessarily in PSPACE.

6.2 Difficulty of Predicting Behavior in Complex Systems

Physical processes can be viewed as computations[114], where the difficulty of answering questions about these processes is equivalent to performing the corresponding computations. In principle the behavior of many complex system can be calculated by explicit simulation. However, the theoretical sciences are concerned with devising shorter calculations to reproduce the outcome of such systems. Assume a class of systems that can indeed be predicted by using a formula describing its future state. Then essentially, this means that the calculations performed by using the formula must be more sophisticated than

the calculations performed by the physical system itself. This implies that the formal system must be more powerful than the corresponding physical system. However, for some physical systems able to perform calculations that are as powerful as those performed by a Universal Turing Machine, no shortcuts are possible. Some physical processes behave *chaotic*[17] and therefore can not be viewed as computations. Here we are confronted with a kind of unpredictability: it is not possible to predict the future accurately, just because slight discrepancies in the initial configuration will grow exponentially in time[22]. This implies that, to accurately predict the system t steps in the future, we need approximately t digits in the specification of the initial configuration. So, the amount of information needed to predict the future grows exponentially with time. In a phase diagram of a complex system it is possible to distinguish *basins of attraction*. Initial trajectories in a phase space can eventually become periodic, fixed, chaotic or disappear to infinity.
A trivial example of a physical process that can be viewed as a computation, is the logic performed by a general purpose computer. There is no way of finding a general procedure for predicting future states. If no general predictive procedure is possible this computation is called *computationally irreducible* [114]. Using computation- and complexity theory it can be shown that such systems have a stronger kind of unpredictability than chaotical systems[114][74]. We can reformulate this notion in a particular physical version of the Church-Turing Hypothesis:
"Universal computers are as powerful in their computational capabilities as any physically realizable system can be, so that they can simulate any physical system"
[114]. If this hypothesis is true, no physical system can shortcut a computationally irreducible process. In this case simulation is the only way out. A system with bounded storage capacity that solves PSPACE-complete problems can be called universal. Formally, only systems having infinite memory capacity that can accept any r.e. language can be called universal, but this is not physically realizable. The question is whether we can find any physical system, other than an artificial computer, that is capable of universal computation. It is assumed that there are many of such systems[83][114][74]. A complex dynamical system, which can simulate any universal computation, is PSPACE-complete. The question whether a given initial state x will ever reach a particular set A is equal to the question whether a Universal Turing Machine ever halts for a given initial input. Even if x is known *exactly*, its basin of attraction is not a recursive set, i.e. there is no algorithm to test whether or not a point is in it[74]. Even in structurally very simple computational systems, very complex behavior can emerge, such that questions about their long time behavior are formally undecidable. In previous sections we came across a DCS with very simple microscopic interaction rules, which is able to display complex behavior, namely *Cellular Automata*.
Summarizing so far, we can can say that a dynamic complex system is formally intractable if it can simulate any polynomial space bounded TM and thus questions regarding its behavior can be PSPACE-complete. In addition it

is possible that some systems can simulate any polynomial time bounded non-deterministic Turing Machine. Imagine for example a system in which certain questions about it can be non-deterministically guessed and checked in polynomial time. Questions regarding such systems can be called NP-complete. The last, and most difficult type of behavior, is displayed by those systems which are capable of imitating *any* Turing Machine, with no resource bounds. Answering questions about these kind of systems can be formally undecidable, we can simulate for ever, without being certain of its behavior. Note that only systems having a continuum of states are potentially Turing universal, while in *deterministic* systems having a finite number of states, the limited precision will cause that after some period, the system will return exactly to some previous state.

6.3 Simulation, Parallelism and P-completeness

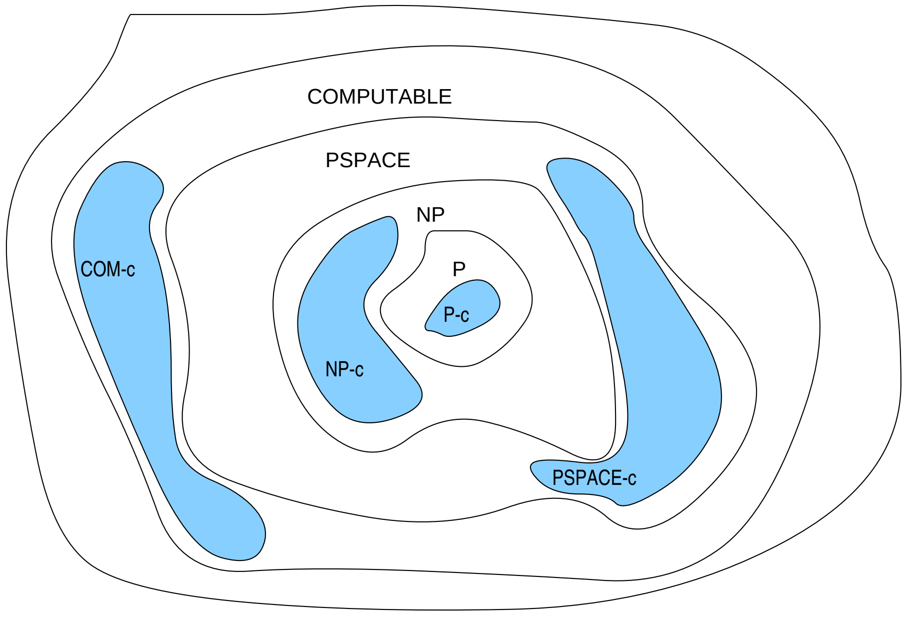


Figure 35: *The space of problem complexities*

We have indicated the hardness of certain kind of problems and are confronted with situations where it is fundamentally impossible to find shortcuts for solving problems, or even worse, solving those problems can be computationally intractable. In the previous section we already considered the subject of simulation. In this section we want to discuss this subject in more detail. How can we position simulation in the world of computation theory? For now let us define a *system* as some dynamical entity which evolves through "time". We already observed that a complex system can be called irreducible whenever

the system is capable of mimicking a Universal TM. Problems regarding the system's ultimate fate are often computationally intractable. Irreducibility implies a simulation approach since every computational step must be explicitly simulated; we can also state that the system suffers from history dependence. In other words we want to solve the problem in what state a system will be after “running” for t time steps. It is essential to know whether the simulation of those systems can be accelerated using multiple processors. Computational complexity theory offers different formal models for parallel computers, e.g. P-RAM and uniform circuit families[10]. A separate complexity class has been defined for problems that admit *efficient parallelization*[10]. The definition is as follows: Let Π be a problem of sequential time complexity $O(N^k)$, Π is in NC if and only if there exists a parallel algorithm which can solve the problem is poly-logarithmic time, $O(log^k N)$, using only a $O(N^k)$ processors. The class of P-complete problems now contains those problems which are “hardest” to solve in parallel, i.e. for which no efficient parallel algorithm exists (unless NC = P). A well known P-complete problem is the Circuit Value Problem (CVP). The problem is to compute the truth value of a so called Boolean circuit, which consists of a hierarchical connected topology of boolean gates, given an input string. Another interesting P-complete problem is Generic Machine Simulation (GSM): Given an input x, an encoding of a TM T, and an integer t coded in unary, decide whether T accepts x within t steps. The P-completeness result of GSM implies that the simulation of any system that is able to perform arbitrary computations is a P-complete problem.
As an example consider an N-body system consisting of a set of interacting particles. A question could be: “Will a particular particle ever reach a certain part of the embedding space?”. It can be shown that this problem is actually PSPACE-hard, hence the solution will not be found by a “fast” procedure (see section 2.2.2). As a consequence we call the *system* of interacting particles *irreducible* and the corresponding *problem intractable*. The natural system can perform “computations” that are as powerful as those performed by universal computers requiring polynomial space. This irreducibility result implies that the simulation problem itself, i.e. the state of the system t time steps away, is a P-complete problem.
Hence, according to our definition of computational irreducibility, for all systems which can only be explicitly simulated, a P-complete problem must be solved. This implies that efficient parallel simulation is *not* possible according to the given definition. This sounds like a devastating result, implying yet another hardness imposed on the class of irreducible systems. Besides the fact that those systems can not be solved by analytical means, they can also not be simulated efficiently in parallel. However, in practice we can already be satisfied by breaking up a problem in time or space. Consider for example a parallel method for finding the largest integer among a set of n integers. The problem can be solved by decomposing the problem in $\sqrt{n}$ sub problems of $\sqrt{n}$ elements each. In the first step a sequential algorithm is applied in parallel to determine the largest value of each sub problem. Subsequently the same algorithm is applied to determine the largest value among the $\sqrt{n}$ resulting

elements. The parallel time complexity of this method is $O(\sqrt{n})$, which is not efficiently parallel, but in practice we can still be satisfied (Note that for this example an efficient algorithm of $O(log(n))$ does exist[10]). P-completeness does not say anything about the possibility of using approximation methods, which may be amenable to efficient parallelization. Moreover, by simulating a system to solve problems, certain phenomena or properties can *emerge* during the simulation, phenomena that were not explicitly coded in the simulation model. By analyzing this emergent behavior, we can answer questions about those systems. In figure 35 we give a short schematic outline of the space of problem complexities. For every DCS we can define a set of properties we are interested in. It is the "hardness" of the properties that can be classified into computational complexity classes. As an alternative to computing the property we can simulate the system and "watch" whether a specific property emerges. Even non-computable problems can be "solved" by simulation, just by watching the system evolve. An example [84] is the problem of determining whether a certain point is a member of the Julia Set, which is the closure of the unstable equilibrium set of the following complex mapping:

$$z(n+1) = z^2(n) + c \tag{124}$$

It is undecidable whether a given point in the complex plane is an element of the Julia set [9], it is however possible to just simulate the mapping and observe the membership of a certain point.

6.4 Computability: Cellular Automata

Even in systems with a very simple structure, it is possible to observe complex behavior. Computational irreducible behavior is found everywhere, even in systems with simple constructions. Examples are systems with only a few degrees of freedom, e.g. the logistic map, and systems with many, but a finite degree of freedom, e.g. Cellular Automata (CA)[115]. The nice thing about CA is that they are easy to construct and are explicitly parallel, which make them good model candidates for implementation on massively parallel computers. CA can be seen as discrete approximations to partial differential equations[106] and can be used for modeling a wide variety of Dynamic Complex Systems[66]. It is possible to classify CA evolution by their behavior. A possible classification was given in [113]. Globally all classifications discriminate between four classes:

I. Fixed point behavior, eventually the evolution of a CA ends in a fixed point of the state space.

II. Periodic behavior, the evolution f is periodic with a period T, so $f(t) = f(t+T)$.

III. Chaotic behavior, evolution leads to chaotic patterns.

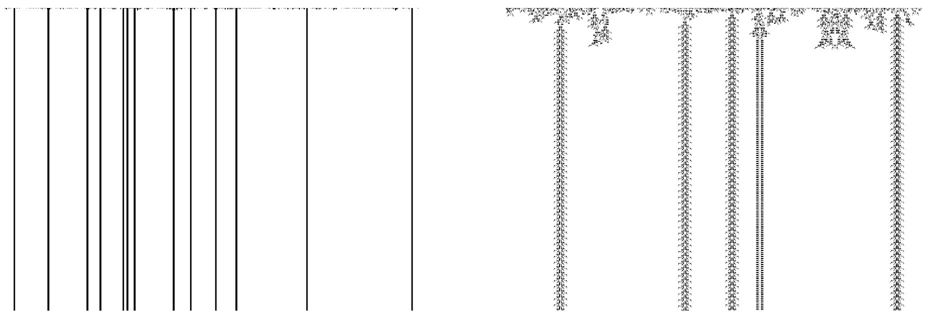


Figure 36: *Fixed point and periodic behavior in 1D CA*

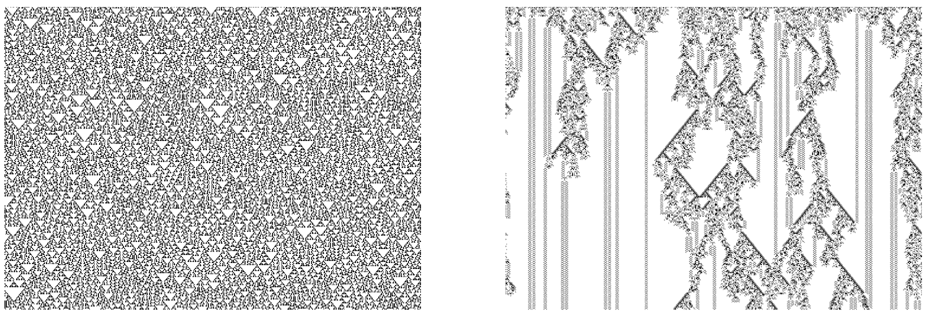


Figure 37: *Chaotic and complex behavior in 1D CA*

IV. Complex behavior, complicated localized patterns occur, sometimes they die out sometimes they don't.

The first three types of behavior are commonly found in dynamical systems, only until recently, behavior of the complex-type has been identified in dynamical systems[74][86][45][85]. It is assumed that dynamical systems which display complex behavior are capable of universal computation[113][58]. In the previous section of this paper, some motivation supporting this system characteristic was given. This section further concentrates on CA, as a type of physical model with the ability to perform irreducible computations.

In Figures 36 and 37, the four possible types of dynamical behavior are shown for a 1 dimensional CA. Dynamical behavior of the CAs in Fig. 37 are examples of computational irreducibility. The problem of determining the configuration of a CA after t time steps, given an initial seed and the CA-rule, can be solved in $O(log\ t)$ time for the CAs in Fig. 36. The evolution for the two other rules can only be found by explicit simulation with length $O(t)$[114]. The difference is that the former two are reducible and the latter two are irreducible, i.e. their ultimate behavior can only be predicted by explicit simulation. For infinite class IV CA it is *effectively* undecidable whether a particular rule operating on a particular initial seed will ultimately lead to a frozen state or not[58][114].

This is the CA analog of Turing's Halting problem. Obviously this decision problem is not undecidable for finite CA, because it is always possible to determine its fate in at most k^N steps, for k states and N sites. This is analogous to resource-bounded Turing Machines, which implies that the problem is PSPACE-complete for universal bounded CA.
Summarizing CA behavior:

- *Computational reducible behavior*: fixed points and periodic. Final state of the system may be predicted analytically.
- *Computational irreducible behavior*: chaotic dynamics are unpredictable and complex dynamics can even be undecidable. The evolution of these systems is its own simplest description and its own fastest computation[24].

As an example we will construct a universal Turing Machine on a cellular automata, showing the universal computing capabilities of CA.

6.4.1 An example of intractability and undecidability in a universal CA

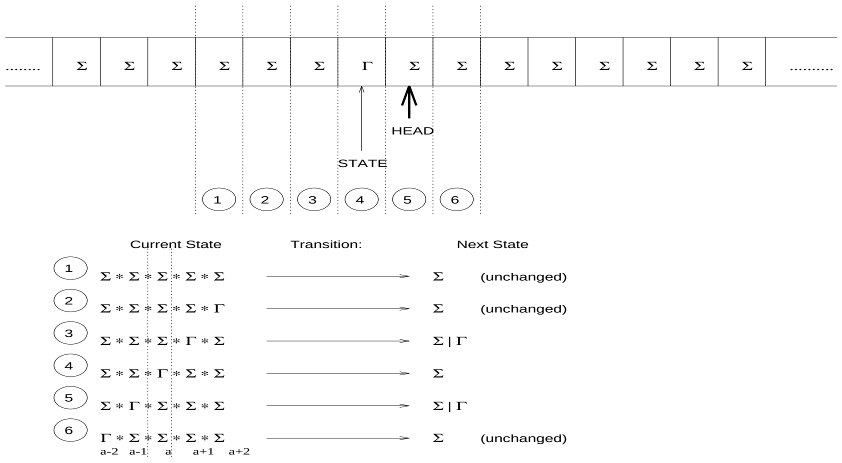


Figure 38: *Top: a TM tape coded as a cellular automaton configuration. Bottom: the CA transition table that implement six different TM transition functions F*

In this section we will construct a TM on a cellular automata using a straightforward method, without exploiting the parallel capabilities of a CA. The construction is merely meant as a proof of existence for computing universal cellular automata and differs from traditional universal CA proofs, in which usually logic gates acting on input signals are identified[7].
A Turing Machine works on a infinite tape on which symbols from an alphabet Σ can be written. The TM can be in one of the states of a set Γ. Furthermore a TM executes instructions using a transition table, which instructs the TM where to move its head (left or right), what to write on the leaving position and what

new state to enter given a read symbol and a current state. We identify the following operations:

- Record current state γ_n
- Read symbol on head σ_n
- Write symbol σ_{n+1} on head
- Do left (L) or right (R) move
- Change state to γ_{n+1}

We are now ready to simulate a TM on a cellular automaton. We simply associate the tape symbols Σ with states of the cellular automaton. The state of a TM will also be part of the state alphabet of cellular automaton. The suggested coding of a TM on a cellular automaton is depicted in Fig. 38. We will use a CA neighborhood with radius 2 and the number of CA states equals $\Gamma + \Sigma$. In Fig. 38, we have numbered the cells to be updated. The numbered cells represent all possible different neighborhoods in the cellular automaton. The associated neighborhoods are listed below the picture of the cellular automaton configuration. The transition from the current cell to the next state is schematically depicted. Some neighborhoods (1,2 and 6) are left unchanged by the transition function, i.e. the state of the cell does not change. At each computation step the TM moves its head either to the left or to the right. The "head" of a TM is positioned at the right of the cell that indicates the current "TM state", i.e. this is the symbol that is "read". The left or right moves have to be coded as cellular automaton transitions. A move of a TM is concurrently executed by three cells, e.g. cells with neighborhoods 3, 4 and 5. This is done for each of the three different neighborhoods:

Neighborhood 3:

Left move: the cell will become new "TM state" γ_{n+1}.
Right Move: No change.

Neighborhood 4:

Left move: The cell will become the new "TM head", reading symbol $a - 1$.
Right move: The cell will become the new written "TM head" symbol σ_{n+1}.

Neighborhood 5:

Left move: The cell will become the new written "TM head" symbol σ_{n+1}.
Right move: The cell will become the new "TM state" γ_{n+1}.

Using this recipe, the cellular automaton can simulate any Turing Machine with a state alphabet Γ and a symbol alphabet Σ, hence the cellular automaton can simulate a universal Turing Machine. This is important since it implies that questions regarding its ultimate state are formally undecidable. A space bounded version of the same cellular automaton can only simulate space bounded universal TM's, which implies that questions about its final behavior are formally intractable or PSPACE-complete.

References

[1] E.H.L. Aarts, A.E. Eiben, and K.H. van Hee. Global convergence of genetic algorithms: a Markov chain analysis. In H.P. Schwefel, editor, *Parallel problem solving from Nature I*, pages 4–12, Berlin, 1990. Springer-Verlag.

[2] P. Alstrom. Self-organized criticality and fractal growth. *Phys. Rev. A*, 41(12):7049–7052, 1990.

[3] A.W. Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6:?, 1985.

[4] R. Azencott. *Simulated annealing: parallelization techniques*. Wiley & sons, New York, 1992.

[5] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Phys. Rev. A*, 38(1):364–374, 1988.

[6] J. Barnes and P. Hut. A hierarchical order$(nlogn)$ force calculation algorithm. *Nature*, 324, 1986.

[7] E.R. Berlekamp, J.H. Conway, and R.K. Guy. *Winning Ways for your Mathematical Plays*. Academic Press, New York, 1982.

[8] D. Bernstein, M. Rodeh, and I. Gertner. On the complexity of scheduling problems for parallel/pipelined machines. *IEEE Transactions on Computers*, C-38(9):1308–1313, 1989.

[9] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: Np-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1), 1989.

[10] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice-Hall, 1994.

[11] G.J. Chaitin. Gödel's theorem and information. *International Journal of Theoretical Physics*, 21(12):941–954, 1982.

[12] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Computational complexity and phase transitions. In *Workshop on Physics and Computation*. IEEE Computer Society, 1992.

[13] S. Chen, Z. Wang, X. Shan, and G. Doolen. Lattice Boltzmann computational fluid dynamics in three dimensions. *Journal of Statistical Physics*, 68(3/4):379–400, 1992.

[14] J.P. Crutchfield. Critical computation, phase transitions and hierarchical learning. In M. Yamaguti, editor, *Towards the Harnessing of Chaos*, Amsterdam, 1994. Elsevier Science.

[15] J.P. Crutchfield and M. Mitchell. The evolution of emergent computation. *Proceedings of the National Academy of Sciences*, 92(23):10742, 1995.

[16] J.P. Crutchfield and K. Young. Inferring statistical complexity. *Phys. Rev. Lett*, 63:105–108, 1989.

[17] P. Cvitanovic. *Universality in Chaos*. Adam Hilger Ltd, Redcliffe Way, Bristol, 1984.

[18] J.F. de Ronde, A. Schoneveld, and P.M.A. Sloot *et al.* Load balancing by redundant decomposition and mapping. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, editors, *High Performance Computing and Networking (HPCN'96)*, pages 555–561, 1996.

[19] J.F. de Ronde, A. Schoneveld, B.J. Overeinder, and P.M.A. Sloot. Map final report. ESPRIT III-CEC CAMAS-TR-2.1.3.4, University of Amsterdam, Amsterdam, 1995.

[20] L. Dikken, F. van der Linden, J. J. J. Vesseur, and P. M. A. Sloot. In W. Gentzsch and U. Harms, editors, *Lecture notes in computer science 797, High Performance Computing and Networking*, pages 273–277, Berlin, 1994. Springer Verlag. vol. Proceedings Volume II, Networking and Tools.

[21] G.D. Doolen. *Lattice gas methods for partial differential equations*. Addison-Wesley, New York, 1990. Proceedings Volume IV, Santa Fe Institute, Studies in the Science of Complexity.

[22] P.G. Drazin. *Nonlinear Systems*. Cambridge University Press, 1992.

[23] P.A. Dufort and C.J. Lumsden. The complexity and entropy of turing machines. In *Workshop on Physics and Computation*, Dallas, Texas, 1994.

[24] B. Eckhardt, J. Ford, and F. Vivaldi. Analytically solvable dynamical systems which are not integrable. *Physica D*, 13:339–356, 1984.

[25] J. Feder. *Fractals*. Plenum Press, New York, London, 1988.

[26] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall, 1988.

[27] G. Fox, M. Johnson, G. Lyzinga, S. Otto, J. Salmon, and D. Walker. *Solving problems on concurrent processesors, Volume I, General techniques and regular problems*. Prentice-Hall International Inc., London, 1988.

[28] G. C. Fox. Achievements and prospects for parallel computing. *Concurrency: Practice and Experience*, 3(6):725–739, December 1991.

[29] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.

[30] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.P. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.

[31] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.*, 56(14):1505–1508, 1986. see also reprint in kn:Doolen.

[32] H. Fujikawa and M. Matsushita. Bacterial fractal growth in the concentration field of nutrient. *J. Phys. Soc. Japan*, 60(1):88–94, 1991.

[33] M. Garey and D.S. Johnson. *Computers and Intractibility; A guide to the Theory of NP-completeness*. W.H. Freeman and Co., San Fransisco, 1979.

[34] D. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, M.A., 1989.

[35] D. Goldberg. A note on Boltzmann tournament selection for genetic algorithms and population oriented simulated annealing. Technical report, University of Alabama, 1990. TCGA Report 90003.

[36] M. Gorges-Schleuter. A asynchronous parallel genetic optimization strategy. In J.D. Schaffer, editor, *3rd International Conference on Genetic Algorithms*, pages 422–427, San Mateo, 1989. Kaufmann.

[37] P. Grassberger. Long-range effects in an elementary cellular automaton. *J. Stat. Phys.*, 45(1/2):27–39, 1986.

[38] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73:?, 1987.

[39] R.W. Hamming. Error detecting and error correcting codes. *Bell Syst. Tech. J.*, 29:147–160, 1950.

[40] A.G. Hoekstra and P.M.A. Sloot. New computational techniques to simulate light scattering from arbitrary particles. *Part. Part. Syst. Charact.*, 11:189–193, 1994.

[41] P. Hogeweg. Cellular automata as a paradigm for ecological modeling. *Applied mathematics and computation*, 27:81–100, 1988.

[42] T. Hogg. Statistical mechanics of combinatorial search. In *Workshop on Physics and Computation*, Dallas, Texas, 1994.

[43] J.H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[44] J. E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[45] J.E. Hutson. Undecidability in an adaptive system. In Nadel and Stein [76].

[46] Jong K.A. de. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[47] J.A. Kaandorp. *Fractal modelling: growth and form in biology*. Springer-Verlag, Berlin, New York, 1994.

[48] J.A. Kaandorp, C. Lowe, D. Frenkel, and P.M.A. Sloot. The effect of nutrient diffusion and flow on coral morphology. *Phys. Rev. Lett.*, 77(11):2328–2331, 1996.

[49] D. Kandhai, A. Koponen, A. G. Hoekstra, M. Kataja, J. Timonen, and P.M.A. Sloot. Lattice boltzmann hydrodynamics on parallel systems. *Submitted*, 1997.

[50] S.A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.

[51] J. De Keyser and D. Roose. Load balancing data parallel programs on distributed memory computers. *Parallel Computing*, 19:1199–1219, 1993.

[52] S. Kirckpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Technical report, IBM, 1982. Research Report RC 9355.

[53] S. Kirkpatrick, C.D. Gelatt jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[54] A. Koponen, D. Kandhai, E. Hellen, M. Alvava, A. G. Hoekstra, M. Kataja, K. Niskanen, and P.M.A. Sloot. Permeability of three diemensional random fibre webs. *Submitted*, 1997.

[55] Z. Koza. The equivalence of the dla and a hydrodynamic model. *J. Phys. A: Math. Gen.*, 24:4895–4905, 1991.

[56] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation Part I. theoretical foundation. *J. Fluid Mech.*, 271:285–309, 1994.

[57] C.G. Langton. Studying artificial life with cellular automata. *Physica D*, 22:120–149, 1986.

[58] C.G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.

[59] W. Li. Mutual information functions versus correlation functions. *J. of Stat. Phys.*, 60(5/6):823–837, 1990.

[60] J. Machta. The computational complexity of pattern formation. *J. of Stat. Phys.*, 70(3/4):949–965, 1993.

[61] J. Machta. The computational complexity of pattern formation. *Journal of Statistical Physics*, 70(3/4):949–967, 1993.

[62] J. Machta and R. Greenlaw. The parallel complexity of growth models. *J. of Stat. Phys.*, 77:755–781, 1994.

[63] J. Machta and R. Greenlaw. The parallel complexity of growth models. *Journal of Statistical Physics*, 77:755–781, 1994.

[64] W.G. Macready, A.G. Siapas, and S.A. Kauffman. Criticality and parallelism in combinatorial optimization. *Science*, 271:56–59, 1996.

[65] B. Manderick and P. Spiessens. Fine grained parallel genetic algorithms. In J.D. Schaffer, editor, *3rd International Conference on Genetic Algorithms*, pages 428–433, San Mateo, 1989. Kaufmann.

[66] P. Manneville, N. Boccara, G.Y. Vichniac, and R. Bidaux, editors. *Cellular Automata and Modeling of Complex Physical Systems*, volume 46 of *Springer Proceedings in Physics*. Springer-Verlag, 1989.

[67] N. Mansour and G. Fox. Allocating data to multicomputer nodes by physical optimization algorithms for loosely synchronous computations. *Concurrency: practice and experience*, 4(7):557–574, 1992.

[68] P. Meakin. A new model for biological pattern formation. *J. Theor. Biol.*, 118:101–113, 1986.

[69] H. Meinhardt. *The algorithmic beauty of sea shells*. Springer-Verlag, Berlin, New York, 1995.

[70] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[71] M. Mezard, G. Parisi, and M.A. Virasoro. *Spin Glass Theory and Beyond*. World Scientific, 1987.

[72] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N.J., 1967.

[73] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Dynamics, computation, and the 'edge of chaos': A re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models, and Reality*, 1994.

[74] C. Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Let.*, 64(20):2354–2357, 1990.

[75] J.D. Murray. *Mathematical Biology*. Springer-Verlag, Berlin, Heidelberg, 1990.

[76] L. Nadel and D.L. Stein, editors. *1990 Lectures in Complex Systems*, volume III of *Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley, 1991.

[77] B. J. Overeinder and P. M. A. Sloot. Application of time warp to parallel simulations with asynchronous cellular automata. In E. J. H. Kerckhoffs A. Verbraeck, editor, *Proceedings European Simulation Symposium 1993*, pages 397–402, Delft, 1993. Society for Computer Simulation International.

[78] B.J. Overeinder and P.M.A. Sloot. Breaking the curse of dynamics by task migration: Pilot experiments in the polder metacomputer. *In press, PVMPI'97*, 1997.

[79] B.J. Overeinder, P.M.A. Sloot, R.N. Heederik, and L.O. Hertzberger. A dynamic load balancing system for parallel cluster computing. *Future Generation Computer Systems*, 12(1):101–115, 1996.

[80] N.H. Packard. Adaptation toward the edge of chaos. In J.A.S. Kelso, A.J. Mandell, and M.F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, 1988.

[81] O. Pla, F. Guinea, and E. Louis. Self-organized criticality in laplacian growth. *Phys. Rev. A*, 42(10):6270–6273, 1990.

[82] O. Pla, F. Guinea, and E. Louis. Self organized criticality in simple growth models. In J.M. Garcia-Ruiz, E. Louis, P. Meakin, and L.M. Sander, editors, *Growth Patterns in Physical Sciences and Biology*, pages 213–219, New York, 1993. Plenum Press.

[83] M. Boykan Pour-El and I. Richards. Noncomputability in models of physical phenomena. *International Journal of Theoretical Physics*, 21(6/7):553–555, 1982.

[84] S. Rasmussen and C.L. Barrett. Elements of a theory of simulation. In *European Conference on Artificial Life*, 1995.

[85] J.H. Reif and S.R. Tate. The complexity of n-body simulation. In A. Lingas, R. Karlsoon, and S. Carlsson, editors, *Automata, Languages and Programming Proceedings*, pages 163–175, 1993.

[86] J.H. Reif, J.D. Tygar, and A. Yoshida. The computability and complexity of optical beam tracing. In *Symposium on Foundations of Computer Science*, 1990.

[87] S. Roux, A. Hansen, and E.L. Hinrichsen. A direct mapping between eden growth and directed polymers in random media. *J. Phys. A: Math. Gen.*, 24:L295–L300, 1991.

[88] C. Rubbia. Report of the EEC working group on high-performance computing.

[89] G. Rudolph and J. Sprave. A globally convergent cellular genetic algorithm. Not yet published, 1993.

[90] E. Sander, L.M. Sander, and R.M. Ziff. Fractals and fractal correlations. *Computers in Physics*, 8(4):420–425, 1994.

[91] L.M. Sander. Fractal growth processes. *Nature*, 322:789–793, 1986.

[92] L.M. Sander. Fractal growth. *Scientific American*, 256(1):82–88, 1987.

[93] A. Schoneveld. An abstract cellular genetic algorithm. Master's thesis, University of Amsterdam, june 1994.

[94] A. Schoneveld, J.F. de Ronde, P.M.A. Sloot, and J.A. Kaandorp. A parallel cellular genetic algorithm used in finite element simulation. In H-.M. Voigt, W. Ebeling, I. Rechenberg, and H-.P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN IV)*, pages 533–542, 1996.

[95] A. Schoneveld, J.A. Kaandorp, and P.M.A. Sloot. Parallel simulation of growth models. In prep.

[96] W. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois, Urbana, 1949.

[97] D. Sherrington. Complexity due to disorder and frustration. In E. Jen, editor, *1989 Lectures in Complex Systems*, volume II of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 415–453. Addison-Wesley, 1990.

[98] P.M.A. Sloot. High performance simulation. *EUROSIM - Simulation News Europe*, (15):15–18, 1995.

[99] P.M.A. Sloot. Modelling for parallel simulation: Possibilities and pitfalls, invited lecture. In *Eurosim'95, Simulation congress*, pages 29–44, Amsterdam, the Netherlands, 1995.

[100] P.M.A. Sloot, J.A. Kaandorp, A.G. Hoekstra, and B.J. Overeinder. *Parallel Scientific Computing and Simulation*. Wiley Book Series, 1998.

[101] P.M.A. Sloot, J.A. Kaandorp, and A. Schoneveld. Dynamic complex systems (dcs): A new approach to parallel computing in computational physics. Technical Report TR-CS-95-08, University of Amsterdam, 1995.

[102] G.B. Sorkin. Simulated annealing on fractals: Theoretical analysis and relevance for combinatorial optimisation. In W.J. Dally, editor, *Advanced Research in VLSI*, Proceedings of the 6th MIT Conference, pages 331–351, 1990.

[103] P.F. Stadler and R. Happel. Correlation structure of the landscape of the graph-bipartitioning problem. *Journal of Physics A*, 25:3103–3110, 1992.

[104] P.F. Stadler and W. Schnabl. The landscape of the traveling salesman problem. *Physics Lettters A*, 161:337–344, 1992.

[105] S. Succi, R. Benzi, and F. Massaioli. A review of the lattice Boltzmann method. In R.A. de Groot and J. Nadrchal, editors, *Physics Computing '92*, pages pp 177–183, Singapore, 1992. World Scientific.

[106] T. Toffoli. Cellular automata as an alternative to differential equations in modeling physics. *Physica D*, 10:117–127, 1984.

[107] M. Tomassini. The parallel genetic cellular automata: application to global function optimization. In R.F. Albrecht, C.R. Reeves, and N.C. Steele, editors, *Artificial neural nets and genetic algorithms*, pages 385–391, Wien, 1993. Springer-Verlag.

[108] J.M. Voogd and P.M.A. Sloot. Simulated annealing on h.p.c. systems: Applied to crystallization with spherical boundary conditions. In A. Verbraeck and E. J. H. Kerckhoffs, editors, *European Simulation Symposium 1993*, pages 371–376, Delft, The Netherlands, 1993. Society for Computer Simulation International.

[109] J.M. Voogd, P.M.A. Sloot, and R. van Dantzig. Comparison of vector and parallel implementations of the simulated annealing algorithm. *FGCS*, 11:467–475, 1995.

[110] E.D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325, 1990.

[111] C.P. Williams and T. Hogg. Using deep structure to locate hard problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 472–477, San Jose, California, 1992.

[112] C.P. Williams and T. Hogg. Phase transitions and coarse-grained search. In *Workshop on Physics and Computation*, Dallas, Texas, 1994.

[113] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

[114] S. Wolfram. Undecidability and intractability in theoretical physics. *Phys. Rev. Let.*, 54:735–738, 1985. In: Cellular Automata and Complexity.

[115] S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.