Parallel complex systems simulation

Schoneveld, A.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3
On the Complexity of Task Allocation

"... just because a model doesn't capture all features of a specific system, this need to imply that the model doesn't capture any aspects at all."

—Henrik J. Jensen, Self-Organized Criticality

3.1 Introduction

An essential problem in parallel computing is the so called Task Allocation Problem (TAP). Given a set of parallel communicating tasks 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 communication and calculation components. These two components can not be regarded independently, but rather are strongly related. Equal distribution of the set of parallel tasks over the available parallel processors, without taking into account the inter task communication leads to optimal work load balancing. On the other hand, if all tasks are placed on a single processor the amount of communication is optimal.

We use the term frustration for the fact that optimization of one term conflicts with optimization of the other, in analogy to physical systems that exhibit frustration (e.g., spin glasses). Increasing dominance of either term reduces the amount of frustration in the system.

In most cases complex behavior is caused by quenched disorder and frustrated, non-linear interactions, between the set of elements constituting a complex system [107]. The quenched disorder refers to the randomness in the dynamic rules or interactions.

It is well known that these system ingredients result in unpredictable emergent behavior [139]. In general, the bulk properties of these systems are analytically intractable. Examples of such properties are asymptotic behavior and the exact location and value of the energetically optimal states. The latter characteristic often causes the corresponding optimization problems to be NP-hard
On the Complexity of Task Allocation

(see Chapter 1). An example where parallelism is discussed in a complex systems context, is the work by Macready et al. [97]. In this work a general optimization algorithm is studied for finding optima of NP-complete problems. By gradually increasing the number of simultaneous (parallel) changes, used for searching from a current solution, a continuous phase transition occurs. At a certain value for the number of parallel changes, a phase transition occurs from “able” to “unable” to find low energy configurations.

In this chapter we will study the TAP problem on task graph instances. An example of a task graph is the interconnected Virtual Particles of the parallel computation framework P-CAM introduced in Chapter 2. Using a statistical mechanics approach, the TAP problem is analytically and empirically analyzed. The nature of the TAP solutions structure as well as the optimization process itself are thoroughly studied. The theoretical analysis is complemented with a initial verification of the results by means of a meta model implemented on the P-CAM framework.

### 3.1.1 Phase transitions in combinatorial optimization

Combinatorial optimization problems can be found in a large number of disciplines, ranging from mathematics (e.g., graph coloring) to biological problems (e.g., DNA folding). A common characteristic of these problems is that a great deal of conflicting constraints must be satisfied in order to find the optimal solutions, most obviously present in the well known Satisfiability problem [58] (see below).

The nature of these problems often require an exhaustive search through a high dimensional phase space to find the optimal solution. The classification of these search problems is usually approached by a worst case analysis by finding an algorithm that can solve the general problem or reducing it to a known difficult problem by methods described in Section 1.2.3.

Computationally hard problems, characterized by exponential running time scaling of their algorithms or memory requirements, are known to be related to the ground state properties of spin-glass like models [107]. As a consequence, a large set of tools and concepts from statistical physics and complex systems theory has been used to shed new light on the complexity of such problems. It even led to the definition of new optimization algorithms such as simulated annealing [83].

In analogy with complex physical systems, phase transitions are found in studying combinatorial search problems if a statistical approach is used. Like physical matter, the qualitative properties of a problem instance may undergo dramatic changes when parameters pass through particular values.

Phase changes are characterized by the appearance of singularities in observables such as specific heat, sensitivity to change and long range order.

An important side note that should be made is that the practical applicability of the statistical mechanics approach to combinatorial optimization is limited. The constructed statistical ensembles of the problem instances do often not cor-
3.2 The Task Allocation Problem

The study in this chapter will be constrained to a specific set of TAP instances: random task graphs allocated to fully connected homogeneous processor topologies. Complementary, an energy function, which quantifies the cost of a task allocation, is constructed. We explore the characteristics of the TAP in terms of phase space and optima structure. Consider a parallel computer consisting of identical processors and a finite speed communication network. Increasing the CPU performance continuously from 0 flop/s to $\infty$ flop/s, induces a transition from optimal parallel- to sequential allocation. It is shown that a paramount characteristic of the TAP is the presence of a sudden transition from sequential to parallel optimal allocation, for specific model constraints. 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 of the TAP. That is, the difficulty of finding optimal solutions peaks near the transition region. The specific correlation structure of the corresponding energy landscape is used as a justification to select simulated annealing as the optimization heuristic [147].
3.2.1 Random task graphs

In order to facilitate a study on abstract parallel applications, a random graph representation as a model of \( n \) communicating parallel tasks is introduced. Each task (vertex) is assigned a work load and every pair of tasks in the task graph is connected with a probability \( \gamma \). A message size is assigned to each link (edge) between two communicating tasks. The size of the message only contributes to the communication time if the two connected tasks are allocated to different processors. Work loads and message sizes are kept constant. The target parallel computer is assumed to be fully connected and homogeneous. That is, all \( P \) processors have identical constant performance. Moreover, the \( P(P-1)/2 \) communication channels are bi-directional and have equal bandwidths. An example corresponding to these models, is a parallel molecular dynamics simulation with Coulomb interactions (long range interactions, leading to global communication patterns) on the IBM SP2 (fully connected and homogeneous topology) [3]. The time evolution of the particles is always preceded by a data exchange phase. The total execution time is determined by the time spent in the communication phase and the calculation phase.

3.2.2 TAP Hamiltonian

A well known NP-complete problem is graph bi-partitioning (GBP) [58]. 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:

\[
\sum_i s_i = 0, \tag{3.1}
\]

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 cost function or energy of a configuration, denoted by \( H \) (Hamiltonian, in analogy with physical systems), can be expressed as follows:

\[
H = \sum_{i<k} J_{ik} (1 - s_i s_k)/2. \tag{3.2}
\]

The constraint (Eq. 3.1) introduces competition, without the constraint the cost would be minimal for all vertices in one partition.

We stress the fact that the TAP is fundamentally different from “constrained optimization.” The number of processors that is used in a TAP allocation is not constrained to a certain value, other than a maximum number of processors to choose from. This approach is in contrast with conventional decomposition and mapping methods, where the utilized number of processors is fixed (see e.g., [101]).

We use the following Hamiltonian to quantify the quality or cost of a task allocation, which is inspired by a similar expression introduced by Fox et al. [59]:
$H = (1 - \beta) \sum_{i > k}^n J_{ik} (1 - \delta(s_i, s_k)) + \beta \sum_i^n W_i^2,$ \hspace{1cm} (3.3)

\begin{align*}
\delta(s_i, s_j) = \begin{cases} 
1 & \text{if } s_i = s_j \\
0 & \text{otherwise}
\end{cases}
\end{align*}

The processor to which task $i$ is allocated, is denoted by $s_i \in \{1...P\}$ 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. $W_i$ the total calculation weight on processor $i$, following from the individual workloads of all allocated tasks. An optimization process that is steered by Eq. 3.3, implicitly minimizes both the variance in the work load distribution and the total communication “surface.” In nature we can observe analogous processes, for example the minimization of the surface/volume ratio in a droplet of water due to surface tension. In our case the communication term can be compared to the surface of the droplet, while the work load variance is similar to its volume.

The $\beta$ parameter can be varied in the range $[0,1]$, in order to tune the competition between the calculation and the communication terms. Variations of $\frac{\beta}{1-\beta}$ can be interpreted either as alterations in an application’s calculation-communication ratio or a computer’s processor speed-bandwidth ratio [51]. The connection probability $\gamma$ in a random graph, can be considered as a dual parameter for $\beta$. Also $\gamma$ can be increased in the range $[0,1]$, which is equivalent to augmenting the average communication load. Additionally, Eq. 3.3 has the locality property, which means that local changes in a task allocation can be propagated into the Hamiltonian without recalculation (see Section 4.5.2). This is specifically useful if an optimization algorithm is applied that is based on incremental changes, and as such can exploit the direct consequence of these increments, reducing the computational cost associated with the optimization process. In the next section we will discuss the Simulated Annealing algorithm [83], the optimization heuristic that is used to find solutions to the TAP, in which the incremental change feature can be exploited. Besides the computational argument of exploiting these incremental changes, a more fundamental motivation to use Simulated Annealing is given in Section 3.3.

### 3.2.3 Simulated Annealing

Annealing is the physical process of heating up a solid until it melts, followed by a slow cooling down until it crystallizes into a state with a perfect lattice. This perfect lattice is the lowest energy state of the solid. Practice shows that the cooling must be done very slow in order not to get trapped in locally optimal structures with crystal imperfections.

Simulated Annealing (SA) is an optimization algorithm, formulated by Kirkpatrick et al. in 1983, based on the annealing of solids [83]. Many problems
originating from physics, chemistry and mathematics can be formulated as optimization problems, which are often NP-complete. A fast majority of these problems involve the determination of the absolute minimum of an underlying multidimensional function. Usually optimization of these complex systems is far from trivial since the solution must be attained from a very large irregular candidate space, containing many local extrema. Because it is not feasible to examine all solution candidates, approximation methods are required. The SA algorithm is a stochastic optimization procedure. There are also non stochastic optimization algorithms such as steepest descent where the path leading to the nearest minimum is taken. The consequence of this method is that is gets stuck in the nearest minimum, which does not need to be the global minimum of the function that is optimized. The stochastic nature of the SA method makes sure that the system does not get trapped in the nearest local minimum but is able to search the phase-space for deeper minima. In theory the method can guarantee that the global minimum of the function is found. The function that is to be optimized is called the cost-function. The SA procedure searches for the state of the system with the lowest cost.

We will give the general idea of how the SA technique works. Analogous to physical annealing we start the system at a high temperature. With the Metropolis algorithm we let the system make a random walk through the phase-space at a fixed “temperature” $T$. This random walk entails creating (random) states of the system given the current state, i.e., a Markov chain. The Metropolis algorithm is defined in Fig. 3.1.

<table>
<thead>
<tr>
<th>Generate a new state $j$ (based on the current state $i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate the energies of the state points $i$ and $j$ :</td>
</tr>
<tr>
<td>$E(i)$ and $E(j)$</td>
</tr>
<tr>
<td>Calculate $A_{ij} = \min[1, \exp(-\frac{E(j)-E(i)}{T})]$</td>
</tr>
<tr>
<td>If $A_{ij}$ is 1 then the proposed state is accepted</td>
</tr>
<tr>
<td>Else Take a uniformly distributed random number, $R$, $0$</td>
</tr>
<tr>
<td>$\leq R &lt; 1$</td>
</tr>
<tr>
<td>If $A_{ij} \geq R$ then the proposed state is accepted as the new state</td>
</tr>
<tr>
<td>If $A_{ij} &lt; R$ the old situation is reused as the new state</td>
</tr>
</tbody>
</table>

**Figure 3.1:** Pseudo code for the Metropolis algorithm

During the random walk we slowly lower the temperature. That is, the temperature is decreased by some cooling schedule and the Metropolis algorithm again generates a sufficiently long Markov chain. A popular cooling is called geometric cooling:
3.2 The Task Allocation Problem

\[ T^* = \alpha^* T \]  

The SA procedure is stopped when a certain stop criterion is met. The result of this is that at a high temperature the costs of the states, given by the cost-function, is not important as can be seen by the transition probabilities \( A_{ij} \) (see pseudo code 3.1). The system can explore the phase-space virtually without constraints. If the temperature is lowered the value of the cost-function begins to be more and more important and features of the function start to be visible in the probability distribution. The system will spend on average more time in states that have a low cost then in states with a higher cost. At this stage the system is reacting to the coarse features of the function. If the system is at a low temperature the details of the function to be optimized come within view. The system will spend most of the time in states that have a low cost, although it is still possible that a number of successive acceptations of cost increasing steps will bring the system into a new region. The process can be stopped if no lower cost has been found in a (large) number of temperature steps. An extended overview of various versions of parallel SA algorithms is given in the PhD thesis by Jeroen Voogd [155].

SA applied to TAP

In this paragraph we will explain how the SA algorithm works as it is applied to the Task Allocation Problem. Constructing an SA algorithm for an arbitrary optimization problem requires three ingredients: an energy function, encoding of candidate solutions, and a so called perturbation mechanism (for generating a random walk). We have already defined the energy function in Section 3.2.2 by Eq. 3.3. Solution vectors are a common method to store potential solutions for discrete problem instances. A proper perturbation mechanism, to generate new solution vectors is constrained to valid solutions. Some problem instances are more strict than others. Consider for example a simple mutation mechanism, which randomly chooses a new value for a vector element. If we were to apply this method as the perturbation mechanism for the well known Traveling Salesman Problem, a serious problem arises. Valid TSP tours are constrained to travel through all available cities, hence a mutation mechanism would generate invalid solutions. A city conserving mechanism, like for example a swap operator does generate valid solutions. A prominent example of such a swap perturbation mechanism is the 2-opt method, which breaks two connections (edges) of a tour and then cross connects them [93].

For the TAP the perturbation problem is easily solved, because we do not have a constrained such as the conservation of the total number of processors. Therefore, we apply a mutation operator. This operator first selects a random task from a solution vector and subsequently assigns it to a random processor. In Fig. 3.2 a task graph, a processor graph and a corresponding arbitrary solution vector are depicted. The 12 tasks, represented as vector elements, are assigned
to one of the 4 available processors. Given the task graph and its attributes (work load per task and communication load per link) the cost of this allocation can be evaluated by Eq. 3.3.

In addition, three SA specific parameters are required to formulate the SA process: the starting temperature \( T_0 \), a “cooling” exponent \( \alpha \) and a convergence criterion. The starting temperature is chosen such that almost all generated allocations are accepted, avoiding any configurational bias. A typical value for \( \alpha \) is 0.95 and the process converges as soon as the optimal configuration has not changed for a fixed number of temperature steps.

The SA process applied to the TAP can now be formulated by the pseudo code in Fig. 3.3.

### 3.3 TAP phase space structure

A random walk through some landscape, can be used to characterize its structure [157]. For landscapes that are self-similar it is known that the corresponding random walk auto-correlation function is a decaying exponential, with correlation length \( \lambda \).

Some basic properties of the TAP phase space are summarized in Section 3.3.1. In the subsequent section we will study the relaxation function of random walks through the task allocation configuration space. This function indicates at what rate a random walk through the space deviates from the starting point, analogous to e.g., relaxation of diffusion processes in physical systems. This function can be related to the auto-correlation function, which quantifies the ruggedness [157] of the TAP energy landscape. Using these functions, it can be shown that the landscape is AR(1) with a correlation length that is linearly proportional to the number of tasks \( n \).
3.3 TAP phase space structure

Allocation := Random Allocation
Cost := CalculateCost(Allocation)
T := T₀

Do
  For length of Markov Chain
    Mutate Allocation := Mutate(Allocation)
    New Cost := CalculateCost(Mutated Allocation)
    If (New Cost < Cost) or (exp((Cost - NewCost)/T) > Random Number)
      Allocation := Mutated Allocation
      Cost := New Cost
    Endif
  Endfor
  T := T × α
End

Figure 3.3: Pseudo code for the SA algorithm as applied to TAP

3.3.1 Configuration space

The configuration space C of the TAP consists of all possible task allocations of the n tasks to the P processor machine. 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 identifier. The Hamming distance \( d_H(A,B) \) (number of differing letter positions) between two sequences A and B is used as the distance metric on C. The corresponding Hamming graph \( \Gamma \) can be constructed by connecting every sequence pair (A,B) with \( d_H(A,B) = 1 \).

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, \( \text{diam} C \) are easily found to be:

\[
N(P,n,d) = \binom{n}{d} (P - 1)^d \tag{3.5}
\]

\[
\#C = P^n \tag{3.6}
\]

\[
\text{diam} C = n \tag{3.7}
\]
3.3.2 Random walk relaxation

The relaxation functions $q_k(s)$ ($k = 1, 2$) of a random walk through an arbitrary configuration space are given by (see [149]):

$$ q_k(s) = 1 - \frac{\Delta_k(s)}{\Delta_k(\infty)} , $$

(3.8)

with $\Delta_1(s)$ the average distance and $\Delta_2(s)$ the average squared distance of a random walk of length $s$.

In previous work we have derived the following relaxation functions for $\Gamma$ [145]:

$$ q_1(s) = \left( 1 - \frac{1}{n} \right)^s = e^{-s/\tau_1} , $$

(3.9)

$$ q_2(s) = \frac{(n-1)(1-P)}{1-n+nP} e^{-s/\tau_2} - \left( \frac{(n-1)(1-P)}{1-n+nP} - 1 \right) e^{-s/\tau_1} , $$

(3.10)

with $\tau_1 \approx n$ and $\tau_2 \approx \frac{n}{2}$.

3.3.3 Auto correlation function

The following auto-correlation function, associated with a random walk through an energy landscape $H : \Gamma \rightarrow \mathbb{R}$, can be used to characterize the structure of the landscape [157]:

$$ r(d) = \frac{\langle (H(x) - \langle H \rangle)(H(y) - \langle H \rangle) \rangle_{d(x,y) = d}}{\sigma^2} , $$

(3.11)

where $d$ is the number of random walk steps between configurations $x$ and $y$, and $\sigma^2$ is the variance of $H$ over all possible allocations.

Apart from totally uncorrelated landscapes, where $r(d) = \delta(d, 0)$, the simplest class consists of the nearly fractal or self-similar landscapes. For such landscapes it is known that the corresponding random walk auto-correlation ($r(d)$) function is a decaying exponential, with correlation length $\lambda$:

$$ r(d) = r(1)^d - e^{-d/\lambda}, d \ll n . $$

(3.12)

Such landscapes are classified as AR(1) (or elementary) landscapes and have been identified in various fields, for example in (bio)physics and combinatorial optimization [157, 149, 150].

According to Eq. 3.12, the auto-correlation function of an AR(1) landscape can be determined from the 1-step auto-correlation. Let $r$ and $r'$ be two configurations with $d(r, r') = 1$, with the Hamiltonian having values $H$ and $H'$:

$$ r(1) = 1 - \frac{\langle (H - H')^2 \rangle}{2\sigma^2} = 1 - \xi . $$

(3.13)

If $\xi$ is sufficiently small, which is reasonable, since only small variations in $H$ are expected.
3.3 TAP phase space structure

\[
\lambda = -\frac{1}{\ln(r(1))} = -\frac{1}{\ln(1 - \xi)} \approx \frac{1}{\xi}, \quad (3.14)
\]

or equivalently,

\[
\lambda = \frac{2\sigma^2}{\langle (H - H')^2 \rangle}. \quad (3.15)
\]

As previously stated, a processor topology is assumed 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. A general class of simple random task graphs is considered. Each pair of tasks is connected with probability \( \gamma \). 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 (3.3). If the allocation number of task \( k \) is mutated (for an arbitrary initial configuration) the following formula for the change in cost \( \delta H = H - H' \) can be derived:

\[
\delta H = 2w_k(W_o - W_p - w_k) + 2R \quad (3.16)
\]

if task \( k \) gets assigned a new allocation number. Else \( \delta H = 0 \).

\( w_k \) is the work associated with task \( k \), \( o \) is the previous allocation number, \( p \) the new one, \( W_o \) is the calculation time due to the work on processor \( o \) and equivalently for processor \( p \). 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).

After some algebra the following expression can be obtained for \( \langle (\delta H)^2 \rangle \) (including the fact that only a fraction \((P-1)/P\) of the mutations contributes indeed the amount given by equation (3.16)):

\[
\langle (\delta H)^2 \rangle = \frac{P-1}{P} (4(1 - 2\langle R \rangle + \langle R^2 \rangle) + 2(\langle W_p^2 \rangle - \langle W_oW_p \rangle + \langle W_pR \rangle - \langle WoR \rangle))) \quad (3.17)
\]

So, in order to obtain an analytical expression for equation (3.17) six quantities need to be derived: \( \langle R \rangle, \langle R^2 \rangle, \langle W_p^2 \rangle, \langle W_oW_p \rangle, \langle W_oR \rangle \) and \( \langle W_pR \rangle \).

Let us first derive an expression for \( \sigma^2 \), before continuing with the derivation of the one-step auto-correlation:

\[
\sigma^2 = \langle H^2 \rangle - \langle H \rangle^2 \quad (3.18)
\]

The simplest of the two terms is \( \langle H \rangle^2 \). We can obtain immediately:

\[
\langle H \rangle = \sum_r \langle W_r^2 \rangle + \sum_{r,s} \langle C_{rs} \rangle \quad (3.19)
\]
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 the number of tasks per processor as a binomial distribution:

The probability that \( k \) tasks get assigned to a specific processor number is therefore given by:

\[
\binom{n}{k} q^k (1-q)^{n-k}
\]  
(3.20)

Using the fact that \( q = \frac{1}{P} \). The expectation value for \( k \) is given by \( <k> = nq = n/P \), whereas the variance \( <k^2> - <k>^2 \) of \( k \) is equal to \( \frac{n}{P} \left( 1 - \frac{1}{P} \right) \). For \( <k^2> \) the following expression can be derived:

\[
<k^2> = \frac{n}{P} \left( \frac{n}{P} + 1 - \frac{1}{P} \right)
\]  
(3.21)

which is equal to \( <W_r^2> \) in the case that all tasks have unit weight.

Next, consider \( <C_{rs}> \). We are interested in the probability of having \( l \) tasks on some processor \( r \), and \( k \) tasks on another processor \( s \), sharing \( x \) edges. This leads to the following expression for the expected communication between an arbitrary processor pair:

\[
<C_{rs}> = \sum_l \binom{n}{l} q_1^l (1-q_1)^{n-l} \sum_k \binom{n-l}{k} q_2^k (1-q_2)^{n-l-k} \sum_x \binom{lk}{x} \gamma^{(1-\gamma)lk-xx}
\]  
(3.22)

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

\[
<C_{rs}> = \gamma nq_2 <l> - \gamma q_2 <l^2>
\]  
(3.23)

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

\[
<C_{rs}> = \gamma \frac{n(n-1)}{P^2}
\]  
(3.24)

This leads to an expression for \( <H> \). Not that the \( <W_r^2> \) term counts \( P \) times, and the \( <C_{rs}> \) term counts \( P(P-1) \) times.

\[
<H> = n \left( \frac{n}{P} + 1 - \frac{1}{P} \right) + \gamma \frac{(P-1)n(n-1)}{P}
\]  
(3.25)

Analogously an expression for \( <H^2> \) can be derived.

\[
<H^2> = <\sum_{rs} W_r^2 W_s^2> + 2 <\sum_{rs} W_r^2 C_{rs}> + <\sum_{stuv} C_{st} C_{uv}>
\]  
(3.26)

After proper evaluation of all the relevant terms \([37]\) an expression for \( \sigma^2 \) is found:
Note that, due to the appearance of $\gamma^2$ terms, equation (3.27) can only be used to predict the variance of an ensemble of random graphs with fixed $\gamma$. This is a consequence of the following fact:

$$\left( \sum_{i} \text{deg}(i) \right)^2 \neq \sum_{i} (\text{deg}(i))^2$$

(3.28)

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 the variance over multiple graph instances must be determined. The $\gamma^2$ term is not present in the expression for the average cost (equation (3.25)), which implies that it is valid for a single specific random graph instance.

The remaining $\langle (\delta H)^2 \rangle$ can be expressed as follows:

$$\langle (\delta H)^2 \rangle = 4 \frac{(P-1)}{p} \left( <R^2> - 4 <lR> + 2(<l^2> - <kl>) \right)$$

(3.29)

In the averaging procedure, $\delta H$ is only considered for those cases that one processor has at least $(i+1)$ tasks, and the destination processor has $k$ tasks. The following expressions for the individual terms can be derived:

$$<R^2> = \frac{2 \gamma (-1+n)}{p}$$

$$<lR> = \frac{\gamma (-1+n)}{p}$$

$$<l^2> = \frac{(-1+n)(-2+n+P)}{p^2}$$

$$<kl> = \frac{(-2+n)(-1+n)}{p^2}$$

(3.30)

which leads to

$$\langle (\delta H)^2 \rangle = \frac{8(-1+\gamma)(1-n)(-1+P)}{p^2}$$

(3.31)

And thus for the one-step auto correlation function:

$$\rho(1) = 1 - \frac{\langle (H-H')^2 \rangle}{2\sigma^2} = 1 + \frac{2}{n(-1+\gamma-\gamma P)}$$

(3.32)

After applying equation (3.15):

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

(3.33)
For fixed $\gamma$ and $P$, $\lambda$ is linearly proportional to the number of tasks $n$. Note that it is assumed that $P > 1$, otherwise $p(1)$ is not defined.

It is important to observe that there are no dependencies of $\gamma^2$ in equation (3.31), which implies that the variance in $\gamma$ (due to $\sigma^2$) does not get eliminated. Experimental validation (see Section 3.6.1 has shown that there is no dependence of $\lambda$ in $\gamma$ and $P$. There is only a dependence in $n$. Strictly speaking this means that the derived formula for $\lambda$ does not correctly predict the correlation structure of the landscape for single task graph instances. However, the $n/2$ term is obviously present in equation (3.33), which corresponds to the correlation time $\tau_2$ derived in Section 3.3.2. In Section 3.6.1 we shall see that this also corresponds to the experimentally determined correlation length.

For a more detailed derivation of the relaxation and auto-correlation functions we refer to [35].

### 3.3.4 Discussion

Summarizing, it has been established that the TAP energy landscape is AR(1), with correlation length $n/2$. Since the TAP landscape has a self-similar structure, we use simulated annealing to find (sub) optima, which is known to be an efficient search method for such landscapes [147].

The Simulated Annealing algorithm will work well on those landscapes where gradual cooling will confine the search to regions in the domain of attraction of low-lying local minima. Gross features of the optimal solutions are selected in the early part of the SA process, while the end part of the search is confined to a small regions of the configuration space. At low temperatures, the algorithm will only allow for minor rearrangements.

The SA process effectively increases the resolution at which the space is searched by lowering the temperature. At the same time, the size of the region of space that is explored decreases. Due to the fractal structure of the TAP search space, SA is able to effectively use its ability to “zoom” into regions of increasingly deeper local minima (see Sorkin [147]).

### 3.4 Phase transition in Simulated Annealing

By applying SA to the optimization of the TAP problem, we are effectively cooling the configurations to the ground states of the TAP Hamiltonian, which corresponds to the (sub optimal) solutions of a TAP instance. This can be compared to generating Markov chains at decreasing temperatures in for example Ising spin or spin glass models. It is well known that the Ising spin model undergoes a second order phase transition at the critical temperature to reach its ground state. The Simulated Annealing algorithm accomplishes the same effect when applied to a combinatorial optimization problem such as the TAP. In this section we will analyze the phase transition to the ground state of the TAP. In the context of complex systems a phase transition often denotes the onset of maximum
information processing. By studying the information processing capabilities of such systems signs of globally coherent behavior may be determined. With information theoretic measures it will be shown that this transition can be compared to the Ising spin phase transition.

### 3.4.1 Information theoretic measures

Some probability distributions are intrinsically more uncertain than others. A uniform distribution carries more doubt on the outcome than for example a Gaussian distribution. We need a mathematical entity that quantifies the uncertainty or doubt of an arbitrary probability distribution by some number. In 1949 Shannon came up with a mathematical theory to deal with compression and transmission of data. Since then his mathematical framework, coined Information theory, has been used in a variety of other fields, most prominently in the area of physics (statistical mechanics) and computer science (algorithmic complexity) [168].

Consider a sequence $S$ of symbols $s_i$ emitted by some source. Each symbol is drawn from an alphabet $\Gamma = 1, \ldots, b$. Now the uncertainty that a symbol occurs depends on its probability $P(s_i)$. Note that the amount of information that we gain from reading a symbol is larger as its probability is smaller. The average information over all outcomes $s_i \in \Gamma$ of an experiment $\mathcal{A}$ is now measured by the entropy [137]:

$$H(\mathcal{A}) = - \sum_{i=1}^{b} P(s_i) \log P(s_i) = \sum_{i=1}^{b} P(s_i) \log \frac{1}{P(s_i)}$$

(3.34)

where $P$ is a discrete probability distribution $P = P(1), \ldots, P(b)$ for $\mathcal{A}$. Two important aspects of this equation are: $H = 0$ if and only if one of the symbols has probability 1 (most certain case), and $H = \ln b$ (maximum value) if all symbols have an equal probability of $1/b$ (most uncertain case).

Given again a sequence $S$ of values from an alphabet $\Gamma$, let $S_n$ denote words of length $n$ from the space $\Gamma^n$ occurring in the sequence. The $n$-block entropy of a sequence of length $n$ over the alphabet $\Gamma$ is defined as:

$$H_n = - \sum_{S_n[S]} P(S) \log P(S)$$

(3.35)

The amount of information that a sequence of length $n$ carries about a sequence of length $n$, the block entropy [65], is now defined as:

$$h_n = H_{n+1} - H_n$$

(3.36)

From Eq.3.36 follows the measure entropy [65]:

$$H = \lim_{n \to \infty} h_n$$

(3.37)

or
On the Complexity of Task Allocation

The measure entropy gives the average information content per site of a given sequence. It can be shown that \( H \) is equivalent to the thermodynamic entropy [48],

\[
S(E) = \log N(E),
\]

(3.39)

where \( N(E) \) is the number of accessible micro-states as a function of the energy \( E \). Micro-states of equal energy have are equally likely to occur; the probability of the \( i \)-th state is:

\[
Pr(i) = \frac{1}{N(E)}, \quad \forall i
\]

(3.40)

If Eq. 3.40 is plugged into Eq. 3.34 the thermodynamic entropy (Eq. 3.39) results. The connection with thermodynamic entropy made Shannon to call \( H \) measure entropy, after he was encouraged by John von Neumann to do so. Von Neumann stated that since no one really understands what entropy is, giving the name entropy to this new measure would give Shannon a “big edge in the debates” [48].

Analogous to the spatial or temporal entropy, one can define temporal/spatial block entropy, where blocks of \( n \times t \) sites are considered:

\[
H_{nt} = \lim_{n,t \to \infty} h_{nt}
\]

(3.41)

Eq. 3.41 decreases monotonically with \( n \), while \( h_{nt} \) decreases also with \( t \). The following difference,

\[
\delta h_n = h_n - h_{n+1}
\]

(3.42)

is the amount of information by which a state \( s_{i+n} \) of a site \( i + n \) becomes less uncertain if the site state \( s_i \) gets known. \( \delta h_n \) is called the \( n \)-th order mutual information. Intuitively one could regard mutual information as the stored information in one site about another site and the degree of predictability of a second site by knowing the first. Another definition of mutual information, block-to-block mutual information, is defined as the mutual information between two \( L \)-blocks[90]. Let \( P_\alpha \) be the probability for an \( L \)-block and \( P_{\alpha \beta}(d) \) be the joint probability for two blocks separated by a distance \( d \):

\[
M(d) = \sum_\alpha \sum_\beta P_{\alpha \beta}(d) \log \frac{P_{\alpha \beta}(d)}{P_\alpha P_\beta}
\]

(3.43)

Note that \( \sum_\beta P_{\alpha \beta}(d) \) reduces to \( P_\beta \) and mutatis mutandis for summing over \( P_\beta \). Hence the Eq. 3.43 can be simplified to:
3.4 Phase transition in Simulated Annealing

\[ M(d) = \sum_{\alpha} \sum_{\beta} P_{\alpha\beta}(d) \log P_{\alpha\beta}(d) = \sum_{\alpha} P_{\alpha} \log P_{\alpha} - \sum_{\beta} P_{\beta} \log P_{\beta} \]

\[ = H_\alpha + H_\beta - H_{\alpha\beta} \] (3.44)

Where \( H_{\alpha\beta} \) is the joined entropy of sequences \( \alpha \) and \( \beta \). Note that this definition can be used to quantify both spatial as well as temporal correlations.

In the case of temporal correlations, the past-future mutual information is often used as a measure of complexity (see e.g., [90], [65]). Also it has been used by Shaw to study symbolic sequences generated by a dripping faucet [138]. Using Eq. 3.38 and Eq. 3.44, the past-future mutual information is:

\[ C = \lim_{m,n \to \infty} (H(S_m) + H(S_n) - H(S_{m+n})) \] (3.45)

The above definition captures the amount of information that a word of length \( m \) contains about its continuation of length \( n \). Sequences with long range correlations will have a high value of \( C \). Eq. 3.45 can also be written as:

\[ C = \lim_{n \to \infty} (H_n - nH) \] (3.46)

Hence, for large \( n \)

\[ H_n \approx C + nH \] (3.47)

An interpretation for Eq. 3.47 is based on the analogy of a computer program and its data. The total information \( H_n \) in a sequence \( S_n \) of \( n \) symbols is composed of a part \( nH \) which equals the amount of information carried by \( n \) symbols (the data) and a part \( C \) which is interpreted as the amount of information inherent in the code itself (the program), thus a measure for the difficulty of decoding the sequence [7].

3.4.2 Experimental results

In this paragraph some experimental results are presented on the disorder-order phase transition that occurs in the SA algorithm as the temperature \( T \) is slowly cooled. The information theoretic measures described in Section 3.4 will be used to quantify the transition and to show the anomalous behavior of long range correlations around the transition point.

In order to measure the disorder and mutual information in the optimization process of the TAP, we will have to extract a time series. The processor value of the first task in the string representation of the current allocation in the Markov chain is monitored. In order to define “time” in the time series, the value is monitored every \( P \) Markov steps. The generated time series is used to measure both \( H \) and \( C \) as defined in Eq. 3.47.
Figure 3.4: Relationship between block entropies $H(S_n)$ and the block length $n$. The slope of the limiting straight line is the value of the entropy $H$, while the $y$-intercept determines the mutual information $C$. The figure has been generated of a TAP instance with $P = 2$, $N = 64$ and $\beta = \gamma = 0.5$ at $T \approx 3.05$.

Figure 3.5: Stored information $C$ and disorder $H$ as a function of $T$, for a TAP instance with $N = 64$, $\beta = \gamma = 0.5$ and $P = 2$.

In Fig. 3.4 we have calculated the block entropies (Eq. 3.36) of a TAP instance with $P = 2$, $n = 64$, $\beta = 0.5$ and $\gamma = 0.5$ from a Markov chain at a fixed temperature of $T = 3.05$. This specific value of the temperature is in the region of the cooling
process where a phase transition takes place in the SA algorithm. The slope of the limiting line in this figure corresponds to the entropy $H$ (Eq. 3.38). The $y$-intercept of this line minus $H(S_i)$ determines the past-future mutual information $C$ (Eq. 3.46).

In Fig. 3.5 both the entropy $H$ as well as the past-future mutual information $C$ is calculated for the entire range of temperatures of the SA algorithm. Around a temperature of $T = 4$ the equilibrium state of the TAP allocations undergo a phase transition from disordered to ordered, as can be seen from the sudden decrease of the entropy $H$. Around this phase transition, the past-future mutual information increases sharply, indicating the appearance of large range correlations. The results of these experiments can be compared with the results found in [7], where Ising spin systems are subjected to an information theoretic analysis.

### 3.5 TAP phase transition

#### 3.5.1 TAP extremes

Although the task allocation problem is NP-hard [13], the two extremes, $\beta = 0$ and $\beta = 1$ are easy to solve. For $\beta = 0$ (infinitely fast CPUs), 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 (with a corresponding lowest energy state of value zero), the number of optima is equal to $P$. In the case of $\beta = 0$ the $P$ optima are at maximum distance in terms of the defined distance metric (see Section 3.3). The $P$-ary inversion operation (analogous to spin-flipping in spin glass models) 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.

For $\beta = 1$ (infinitely fast network) only a repulsive work load term is present, which will force the variance in the work load distribution to be minimized. This results in an equal partitioning of the total work load over all available processors. It can easily be shown that the total number of optima in this case equals:

\[
\prod_{k=1}^{P} \left( \frac{n}{k(n/P)} \right) = \frac{n!}{(n/P)!^P},
\]  \hspace{1cm} (3.48)

where it has been assumed that $n/P$ is integer. The corresponding optimal value of the Hamiltonian is equal to $n^2/P$. In case of $\beta = 1$, the optima are relatively close to one another. Again, two types of operations can be distinguished that leave the value of the Hamiltonian invariant, which are rotation of the sequence and permutation of two arbitrary tasks.
3.5.2 Locating the transition

A transition from sequential to parallel allocation can be observed when \( \beta \) is increased from 0 to 1 (or equivalently, if \( \gamma \) is decreased from 1 to 0). In order to quantify this (phase) transition we define an order parameter, which expresses the degree of parallelism present in an optimal allocation. Since all tasks and connection weights are unity, the order parameter \( \mathcal{P} \), quantifying the parallelism in a given optimal allocation can be defined as follows:

\[
\mathcal{P} = 1 - \frac{(\langle W^2 \rangle - \langle W \rangle^2)P^2}{n^2(P-1)},
\]

where \( W \) is the time spent in calculation and \( n^2(P-1)/P^2 \) is the maximal possible variance in \( W \). Eq. 3.49 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. Another measure for \( \gamma \) can be given by a kind of entropy measure:

\[
\gamma = -\sum_i p_i \log p_i
\]

where \( p_i \) is the fraction of tasks present on processor \( i \) and \( P \) the total number of available processors. Eq. 3.50 takes the value 1 in the case of optimal parallelism and the value 0 in the case of a sequential allocation. This order parameter corresponds to the 1-block entropy of optimal allocations.

Using Eq. 3.25 we can calculate whether the average value Eq. 3.3 either increases or decreases by using more processors. Using a mean field argument, the transition from sequential to parallel allocation will approximately occur for those values of \( \beta \) and \( \gamma \) for which Eq. 3.3 will change from a monotonically decreasing function to a monotonically increasing function of \( P \). In other words, setting

\[
\frac{\partial \langle H \rangle}{\partial P} = 0,
\]

with the additional constraint that either \( \gamma \) or \( \beta \) is fixed, we obtain the following transition values:

\[
\beta_c = \frac{\gamma}{1+\gamma}, \quad \gamma_c = \frac{\beta}{1-\beta}
\]

The values of \( \beta_c \) and \( \gamma_c \) are interpreted as the critical values of \( \beta \) and \( \gamma \) in analogy with the critical temperature \( T_c \) in thermal phase transitions or the percolation threshold \( p_c \).
3.5.3 Critical slowing down

Many search methods show anomalous behavior for certain critical parameters of combinatorial search problems [158, 159, 71, 23]. For example in the case of graph coloring it has been observed that the “difficulty” of determining if a graph can be colored, increases abruptly when the average connectivity in the graph is gradually increased to some critical value [158]. In Ising model simulations, the difficulty of equilibrating increases when the critical temperature is approached (critical slowing down).

In analogy, we expect that in the TAP comparable phenomena can be found in a critical region of the $\beta$ and $\gamma$-domain. For both $\beta$ extremes the optima are known in advance. The difficulty to find these optima is therefore trivial. If the calculation and the communication term in the Hamiltonian (Eq. 3.3) are of comparable magnitude, the system is said to be in a critical area. Moving away from this critical region, one term becomes small noise for the other.

We will use the following empirical method to estimate the computational cost of finding optima. The number of local optima are measured, in which independent steepest descent (SD) runs get stuck. A specific search space is considered to be “simple” if it contains a relatively small number of local optima, otherwise it is classified as “difficult.” The distinction between local optima is based on the value of the Hamiltonian of the corresponding task allocations. That is, two local optima $i$ and $j$ are called distinct if:

$$H(i) \neq H(j). \quad (3.54)$$

Due to the fact that the TAP energy landscape is AR(1), we do not expect large plateaus in which SD can get stuck in its search for a true local minimum (a minimum with the lowest value of the Hamiltonian in its local neighborhood). Therefore, it is to be expected that plateau states will not have a major contribution to the cost of the search.

Of course, one may wonder how this cost heuristic relates to a standard search measure like the number of convergence steps taken in SA. In Section 3.6, we will present empirical evidence that both measures are strongly correlated.

3.5.4 Finite size scaling

In many physical systems the sharpness and the location of transition points depend on the system size. This dependence can be analyzed by a method from statistical physics called finite size scaling. The existence of scaling parameters for transitions at different system sizes is a direct evidence for critical behavior at the transition. The system is indistinguishable at all sizes, except for a change of scale. The scale change can be found by analyzing the shift of the TAP transition (with fixed $\gamma$):

$$\beta_c(n) - \beta_c(\infty) \propto n^{-1/\nu} \quad (3.55)$$
where $\beta_c(n)$ is the location of the transition for a TAP instance of size $n$ and $\nu$ is called a critical exponent. The parameter $\nu$ can be used to re-scale $\beta$ as follows:

$$\beta^* = n^{1/\nu} (\beta - \beta_c(\infty))/\beta_c(\infty) \quad (3.56)$$

### 3.5.5 Spatial mutual information

The spatial mutual information between the processor allocation of two (closely) connected tasks can be used to indicate “how much” a different processor allocation of a one task influences the allocation of the other. The mutual information, $MI(A,B)$, is a function of the individual task allocation entropies, $S(A)$ and $S(B)$ ($= P$ of Eq. 3.50), and the entropy of the two tasks considered as a joint allocation, $S(A,B)$. In Section 3.4.1 we have already defined mutual information using Eq. 3.44:

$$MI(A,B) = S(A) + S(B) - S(A,B) \quad (3.57)$$

The entropies are calculated by generating several “optimal” allocations, from which the allocation probabilities, $p_i$ and $p_{ij}$ (see Eq. 3.50) are derived. Because the inherent structure of a task graph is not a 1-dimensional string but given by the corresponding connectivity graph, the tasks $A$ and $B$ are chosen such that they are nearest neighbors.

### 3.6 Experimental results

In this section experimental results regarding correlation length, phase transition and search cost for the TAP are presented. In order to validate the theoretical results of the various TAP quantities that have been derived, such as the average cost and the correlation length, a paragraph has been devoted to present some validation results. A more detailed suite of these experiments can be found in [35]. Several experiments on the TAP phase transition are summarized in paragraph 3.6.2. The theoretical prediction of transition location (Eqs. 3.52 and 3.53 is validated for several parameter values. It is shown that the transition are amenable to finite size scaling and several other experiments are presented to show the critical behavior in the transition region. The last part of this section is devoted to the special character of $\gamma = 1$ or fully connected task graphs. It is shown this transition does not display critical behavior, due to its exact nature.

#### 3.6.1 Correlation length and statistical quantities

In Fig.3.6 measured and predicted correlation functions are displayed, with parameters $n = 100$, $P = 8$ and $\gamma = 0$. In the second experiment a TAP instance with a non-zero connection probability ($\gamma = 0.5$), $n = 64$ and $P = 4$ is used. The theoretical correlation functions with correlation lengths 50 and 32 respectively ($n/2$),
3.6 Experimental results

**Figure 3.6:** Analytical (dashed lines) and experimental values for the auto-correlation function \( r(s) \) with \( n = 100, P = 8 \) and \( \gamma = 0.0 \) (diamonds) and \( n = 64, P = 4 \) and \( \gamma = 0.5 \) (pluses). The experimental auto-correlation functions are generated from a random walk of 640000 steps.

**Figure 3.7:** Density of states of TAP instance with \( n = 64, P = 8, \beta = 0.5, \gamma = 0.9 \). Experimental data is depicted by a solid line. A Gaussian distribution is plotted (dashed line) with mean and variance according to Eqs. 3.25 and 3.27 respectively.

are plotted as dashed lines. Clearly, the predicted correlation functions match the experimental data.
The predicted statistical quantities, $\langle H \rangle$, the variance $\sigma^2$, and $\langle (H - H')^2 \rangle$, have been validated extensively [37] by random walks through the energy landscape. In Fig. 3.7 we plot the "density of states" $\mathcal{N}(H)$, indicating the probability of an allocation with a certain energy. The results of a TAP instance, with $n = 64$, $P = 8$, $\gamma = 0.9$ and $\beta = 0.5$ are depicted in Fig. 3.7. From Fig. 3.7, it can be observed that the experimental data is in reasonable correspondence with a Gaussian distribution with mean $\langle H \rangle$ (Eq. 3.25) and variance $\sigma^2$ (Eq. 3.27). A more detailed validation of the individual statistical averages constituting $\langle H \rangle$ is given in [37].

### 3.6.2 TAP phase transition

Next, several experiments are conducted to demonstrate the existence of a phase transition. Furthermore, the location of the transition, as predicted by Eqs. (3.52) and (3.53), is checked.

![Figure 3.8: Two phase transitions with fixed $\gamma$ and increasing $\beta$, with $\gamma = 0.2$, $n = 256$ and $P = 32$, and with $\gamma = 0.5$, $n = 128$ and $P = 8$ respectively. The vertical lines indicate the location of the transition as predicted by Eq. 3.52. The $\beta$ domain is scanned with steps of $\Delta \beta = 0.01$. For each value of $\beta$, $P$ is estimated by averaging over 25 simulated annealing runs.](image)

In Fig. 3.8 $\beta$ is varied in the range $[0, 1]$ and $\gamma$ is fixed at two different values (0.2 and 0.5). In Fig. 3.9 the dual experiment is performed, where $\gamma$ is varied in the range $[0, 1]$ and $\beta$ is fixed at the value 0.25. The results presented are comparable with those found for arbitrary parameter values (data not shown). The mean field transition points (Eqs. (3.52) and (3.53)) are plotted as vertical lines. As shown in Figs. 3.8 and 3.9, the approximate location of the phase transition that is induced by variation of $\beta$ or $\gamma$ can be predicted by a mean field argument (Eq. 3.51).
3.6 Experimental results

Figure 3.9: A phase transition with $\beta = 0.25$, $n = 64$ and $P = 8$. The vertical solid line indicates the location of the transition as predicted by Eq. 3.53. The $\gamma$ domain is scanned with steps of $\Delta \gamma = 0.025$. For each value of $\gamma$, $P$ is estimated by averaging over 10 simulated annealing runs.

Figure 3.10: A phase transition and the search cost (with standard deviations) with $\gamma = 0.5$, $n = 32$ and $P = 4$ with $\beta$ varied with steps of $\Delta \beta = 0.025$. (The cost is determined for $\beta \in \{0.1, \ldots, 0.5\}$.) The vertical line indicates the location of the transition as predicted by Eq. 3.52. The values for $P$ are estimated by averaging over 10 simulated annealing runs. Each point in the search cost is estimated over 10 random graph instances, where for each instance 10 steepest descent runs are conducted. The cost value is scaled to fit in the range [0.1].
Fig. 3.10 displays the search cost and the order parameter $T$, for task graphs with $n = 32$, $P = 4$, $\gamma$ fixed to 0.5 and $\beta$ varied in the range $[0, 1]$. In Fig. 3.11 the same experiment is carried out, now with $\gamma$ varied in the range $[0, 1]$, $n = 64$, $P = 8$ and $\beta$ fixed to 0.2. The divergence of the search cost near the transition point can be observed in both plots. The method described in Section 3.5 is used to quantify the search cost. In Fig. 3.10 the standard deviation over the order parameter, i.e., the order parameter susceptibility, is depicted. It can be observed that the order parameter susceptibility increases at the phase transition, which can be compared with for instance increasing magnetic susceptibility of Ising spin systems near the critical temperature. A additional smaller peak in the search cost can be observed around $\beta = 0.45$. Currently we do not have a valid explanation for this peak, but we suspect that it could be explained by an additional geometric order phase that is not captured by the order parameter $T$.

Figure 3.11: A phase transition (with standard deviations) and the search cost with $\beta = 0.2$, $n = 64$ and $P = 8$ with $\gamma$ varied with steps of $\Delta\gamma = 0.025$ (The cost is determined cost for $\gamma \in \{0.1, \ldots, 0.5\}$). The vertical line indicates the location of the transition as predicted by Eq. 3.53. The values for $T$ are estimated by averaging over 10 simulated annealing runs. Each point in the search cost is estimated over 10 random graph instances, where for each instance 10n steepest descent runs are conducted. The cost value is scaled to fit in the range $[0.1]$.

To illustrate the correlation between the cost heuristic, introduced in Section 3.5.3, and the average number of SA convergence steps, an experiment has been carried out with the following parameter setting: $n = 64$, $P = 4$ and $\gamma = 0.5$, while $\beta$ is varied between 0 and 1. We applied the following convergence criterion for SA: if the optimal allocation does not improve with more than 5% during 50 consecutive temperature lowerings, SA has reached convergence. Fig. 3.12 shows that both measures peak near the predicted transition point $\beta_c$. The results in-
3.6 Experimental results

Figure 3.12: The average number of SA steps to reach convergence and the search cost, with \( n = 64, P = 4, \gamma = 0.5 \) and \( \beta \in [0, 1] \). The number of SA steps is averaged over 25 runs. The vertical line indicates the location of the transition as predicted by Eq. 3.52. Each point in the search cost is estimated over 10 random graph instances, where each instance 10n steepest descent runs are conducted. Both measures are scaled to fit in the range \([0,1]\).

Data indicate that it takes more temperature steps to find an optimal allocation in the critical \( \beta \) region. In other words, the “freezing temperature” of SA appears to behave anomalously near the TAP phase transition. Again additional peaks in both the SA convergence steps and the search cost can be observed.

Finite size scaling

The results of a finite size scaling experiment are summarized in Fig. 3.13 for task graphs with \( P = 8 \) and \( \gamma = 0.2 \). For increasing task graphs sizes \( n \in \{80, 160, 240, 320, 400, 480, 560, 640\} \), the order parameter \( \mathcal{P} \) has been estimated for 10 different values of \( \beta \) near the predicted phase transition. Each point in the figure is averaged over 25 SA runs. It can be observed that all functions intersect in one point. Assuming that all task graph order parameter functions intersect at a common point, this must be the exact value of \( \beta_c \). The critical value of \( \beta \) found experimentally, closely corresponds to the value as predicted by Eq. 3.52. In order to test the finite size scaling hypothesis, we have plotted the data from Fig. 3.13 against the rescaled \( \beta \) parameter, \( \beta^* \) (see Section 3.5.4). We define \( \beta_c(n) \) to be the value of \( \beta \) at which \( \mathcal{P} = 0.5 \) and \( \beta_c(\infty) \) is given by Eq. 3.52. In Fig. 3.14, we observe that all rescaled plots fall on a \( n \)-independent curve. The inverse experiment, i.e., a transition with \( \beta \) fixed, has also been conducted. In Fig. 3.15 the results of a finite size scaling experiment with \( \beta = 0.2 \) and \( P = 8 \).
Figure 3.13: For task graphs sizes $n \in \{80, 160, 240, 320, 400, 480, 560, 640\}$, $P = 8$ and $\gamma = 0.2$, $P$ is calculated around the mean field prediction of $\beta_c$ for $\beta$ varied in the range $[0.1, 0.19]$ and a step size of $\Delta \beta = 0.01$. The vertical line indicates the location of the transition as predicted by Eq. 3.52.

Figure 3.14: Finite size scaling. The curves of Fig. 3.13 are plotted against the rescaled $\beta^* = n^{1/v}(\beta - \beta_c(\infty))/\beta_c(\infty)$, where $v \approx 1.83 \pm 0.04$ has been estimated by fitting.

is shown. It is found that, using the same scaling exponent as in the previous experiment, all 8 functions can be scaled to the same universal function. To further show the universality of the scaling exponent an additional experiment
3.6 Experimental results

Figure 3.15: Finite size scaling with $\beta = 0.2$ and $P = 8$. $\gamma^* = n^{1/v}(\gamma - \gamma_c(\infty))/\gamma_c(\infty)$, with $v = 1.83$.

Figure 3.16: Finite size scaling with $\beta = 0.2$ and $P = 16$. $\gamma^* = n^{1/v}(\gamma - \gamma_c(\infty))/\gamma_c(\infty)$, with $v = 1.83$.

with the same parameters as in the previous one except for $P = 16$ is shown in Fig. 3.16.
Sensitivity to perturbations

The optimal allocation is reached with an SA algorithm. Subsequently, a few connections in the task graph are removed (perturbation) and the first local minimum is found by a steepest descent (SD) strategy. For several values of $\beta$, the number of SD steps needed to reach a new local minimum is recorded. The results of a perturbation experiment using a TAP instance with $n = 120, P = 8$ and $\gamma = 0.3$ are displayed in Fig. 3.17 together with the corresponding values of the order parameter, $\mathcal{P}$.

![Normalized number of SD steps (dotted line) needed to reach a new local minimum for a TAP instance with $n = 120, P = 8, \gamma = 0.3$. The phase transition from optimal sequential ($\mathcal{P} = 0$) to optimal parallel allocation ($\mathcal{P} = 1$) is rendered by the solid line.](image)

**Figure 3.17:** Normalized number of SD steps (dotted line) needed to reach a new local minimum for a TAP instance with $n = 120, P = 8, \gamma = 0.3$. The phase transition from optimal sequential ($\mathcal{P} = 0$) to optimal parallel allocation ($\mathcal{P} = 1$) is rendered by the solid line.

Spatial mutual information

Due to the fact that the processor graph is fully connected, symmetrical allocation exists, i.e., allocations in which processor ids can be swapped. Therefore, the allocations are “normalized” before the entropies are calculated. The normalization allocates tasks to processor ids by subsequently assigning increasing processor ids (starting from 0) to processor clusters sorted by size. This process eliminates invariant or symmetrical allocations.

The spatial mutual information $MI(A,B)$ (Eq. 3.57) between two next-nearest neighbor tasks is experimentally calculated for a set of TAP instances with $n = 96, 160, 320, 400, P = 4$ and $\gamma = 0.2$. $MI(A,B)$ reaches a maximum near the (scaled) transition point $\beta_c$. After rescaling the 4 different plots, resulting from the experimental data, with the scaling factor $\nu = 1.83$, we obtain Fig. 3.18. The peak
3.6 Experimental results

Figure 3.18: Next-nearest neighbor spatial mutual information. \( n = 96, 160, 320, 400 \), \( P = 4, \gamma = 0.2 \), the mutual information is calculated for each value of \( \beta \) for 100 identical task graphs, which is again averaged over 20 different instances. The mutual information functions are scaled with \( \nu = 1.83 \) as obtained from the finite size scaling analysis.

at the transition indicates that the influence on the mutual allocation of two closely connected tasks is large.

Cost susceptibility

The “cost susceptibility” is calculated using several optimal allocations:

\[
\chi = \langle C^2 \rangle - \langle C \rangle^2
\]  

(3.58)

The results presented in this section can be used to argue that the TAP is most difficult to solve for critical values of either \( \beta \) or \( \gamma \). Outside the transition region optimal allocation can be found more easily. The sharp increase of the cost susceptibility at the transition indicates that the optimal allocation becomes most sensitive to the mutual allocation of individual tasks (see Fig. 3.19).

3.6.3 Fully connected task graphs

According to the expressions above, some interesting properties for fully connected task graphs (\( \gamma = 1 \)) can be given. First, consider the correlation length for the critical value, \( \beta_c = 1/2 \) (From Eq. 3.52). The correlation length \( \lambda \) of the TAP landscape becomes indeterminate for \( \gamma = 1 \) and \( \beta = 1/2 \). It can easily be shown that these are the only values of \( \beta \) and \( \gamma \), resulting in \( \lambda = \frac{\theta}{2} \). The reason for this behavior is that the TAP landscape becomes flat, which can be seen
On the Complexity of Task Allocation

Figure 3.19: Cost susceptibility $\chi$ for a TAP instance with $N = 64, P = 4, \gamma = 0.4$. $\beta$ is varied between $[0, 1]$.

From the zero variance for this specific $(\gamma, \beta)$-pair in Eq. 3.27. Furthermore the following holds for $\gamma = 1$ at $\beta_c$:

$$<H>_{\beta_c} = H_{min} = H_{max} = \beta_c n^2 = \frac{1}{2} n^2$$

(3.59)

This means that the density of states $N(H)$, expressing the probability of finding an allocation with energy value $H$, collapses into a Dirac $\delta$ for $\beta = \frac{1}{2}$ and $\gamma = 1$:

$$\lim_{\beta \rightarrow \frac{1}{2}} N(H) = \delta(H - \frac{1}{2} n^2)$$

(3.60)

$P$ undergoes a discontinuous transition from sequential ($P = 0$) to parallel ($P = 1$) allocation, which can be shown as follows. Let us discuss the case $\beta < \beta_c$, i.e., the “sequential region.” For these values of $\beta$ the tasks are optimally allocated using a single processor. In order to see what happens close to the transition point, consider the smallest deviation from a sequential allocation: $n - 1$ tasks on 1 processor, 1 task on an arbitrary other and a small perturbation $\delta \beta$ from $\beta_c$. For this specific situation we can calculate the corresponding cost $C_\delta$ explicitly for $n$ tasks as well as for the sequential allocation cost $C_{seq}$ (all tasks on one processor):

$$C_\delta = (\beta_c - \delta \beta) (\frac{(n-1)^2}{2} + 1) + (1 - \beta_c + \delta \beta) (2(n-1))$$

(3.61)

$$C_{seq} = (\beta_c - \delta \beta) n^2$$

(3.62)
It is easy to see that $C_{\text{seq}} < C_{\beta}$ for all $\delta \beta > 0$ and $\beta_c = 1/2$. For the case $\beta > \beta_c$ a similar argument holds. The corresponding parallel allocation cost, $C_{\text{par}}$ (all tasks evenly distributed) is found to be:

$$C_{\text{par}} = \beta_c P \left( \frac{n}{P} \right)^2 + (1 - \beta_c) \left( \frac{n}{P} \right)^2 (P - 1)P$$

(3.63)

**Figure 3.20:** A discontinuous transition with $n = 64$, $P = 4$ and $\gamma = 1.0$. The search cost is indicated by the solid line. The dashed line depicts the order parameter. Note that the value of the “search cost” remains zero.

The transition for the fully connected graph on the other hand (Fig. 3.20), is obviously discontinuous and lacks a peak in the search cost, i.e., the optimal allocation problem is trivial over the entire spectrum of $\beta$. For $\beta < \beta_c$ the optimal cost can be calculated by $C_{\text{seq}}$ (Eq. 3.61). The optimal cost for the case $\beta > \beta_c$, is given by $C_{\text{par}}$ (Eq. 3.63). At $\beta = \beta_c$, all solutions have equal cost, hence any solution is optimal.

### 3.7 Experimental validation of the TAP phase transition

In this section we will conduct a validation of the inherent property of the TAP problem, the existence of a phase transition from sequential to parallel optimal allocation. It is important to note that we do not expect to validate all quantitative results of the previous section for various reasons, which we will discuss later. We will apply the P-CAM framework to implement a meta-model
of a TAP instance, i.e., a random task graph will be used as a parallel application topology on which fictive work is done with fictive communication between the interconnected tasks. A "physical optimization" algorithm will be applied to find "optimal" allocations, hence the task migration primitives of the P-CAM framework can be used conveniently.

3.7.1 Meta modeling with P-CAM

The TAP cost-function is not a realistic model in the sense that it does not model the actual cost of a parallel application on a real parallel machine. The TAP model is essentially based on the physical analogy of the minimalization of surface tension, to steer the optimization process. It is only the final allocation that is expected to be optimal from the viewpoint of parallel computation, the quantification of this allocation alone does not give any clues to the actual run time of the parallel application when executed on a parallel computer. The source of the discrepancy is twofold. Obviously, the quadratic workload term in the TAP cost-function is not conform the actual cost of work. The situation for the communication term is even worse. Apparently, the assumed communication model does not take into account communication latencies, which may contribute for a major part to the total communication time. Secondly, the accumulation of the interactions between the tasks is not really correct. Whenever a task $i$ on a processor $P_0$ is connected to several other tasks on a processor $P_1$, all these links are added in the TAP cost-function. However, in reality, that is in P-CAM, the messages of task $i$ are only send once to the other processor $P_1$, not for every task separately. Hence, the expression for the expected communication volume between two processor pairs (Eq. 3.24), derived for the TAP model is not valid in the P-CAM framework. Our objective in this section is not to validate all the quantitative results derived for the TAP model, but a more modest exercise: to validate whether a transition from sequential to parallel allocation is inherently present in the allocation of random task graphs.

A fairly simple meta model to simulate the process of work done alternated by a communication step can be defined, acting on arbitrary task graphs. Pseudo code for this meta model is given in Fig. 3.21.

In order to use this meta model in the P-CAM framework, an instance of a random graph is created given the parameters $n$ and $\gamma$. The specification of this random graph instance is passed P-CAM in addition with an initial decomposition specification. Subsequently the meta model of Fig. 3.21 is executed on each individual processor. The work (DoWork) step is performed on each allocated task. The communication pattern is based on the random task graph instance.

3.7.2 "Physical" optimization

In this paragraph we will define the physical optimization process. There is still much room for a debate on a proper definition of the algorithm, but our purpose is just to validate a part of a theoretical result and as such we are not interested
3.7 Experimental validation of the TAP phase transition

\[
\text{CommunicationVolume} := (1 - \beta) \times \text{BASE\_COMMUNICATION\_LOAD}
\]

\[
\text{TotalWork} := \beta \times \text{BASE\_WORK\_LOAD}
\]

Do

\[
\tau_{\text{comm}} := \text{CommunicateBorderCells(CommunicationVolume)}
\]

\[
\tau_{\text{calc}} := \text{DoWork(TotalWork)}
\]

\[
\tau_{\text{total}} := \tau_{\text{comm}} + \tau_{\text{calc}}
\]

Until Number of iterations

Figure 3.21: Pseudo code for the TAP meta model

in an optimal algorithm. The optimization process is conducted by alternating a meta model iteration and a task migration process. The functionality of the P-CAM framework is well suited to exploit the task migration process necessary to perform a local search optimization procedure.

We have defined a simple local search optimization procedure to optimize the allocation of the meta model of the previous section. The initial decomposition allocates all tasks onto 1 processor. The idea is to randomly migrate tasks from one processor to the other, and to execute the meta model after each migration step. If reallocation is fruitful, that is if the run-time of the new allocation is smaller than that of the previous allocation the allocation is kept, otherwise the migration process is discarded (performed in the opposite direction). The migration decision is based on several random processes. The first step is to choose either for an emigration or an immigration of tasks, that is a processor \(P'\) pushes tasks away to another one or it pulls tasks from another processor. The processor \(P'\) is chosen randomly. The migration volume, that is the total number of tasks that are to be engaged in the migration process, is also chosen randomly. The last step in the migration phase consists of randomly selecting task for migration until the total migration volume has been resolved. A pseudo code of the optimization procedure is given in Fig. 3.22.

3.7.3 Experimental results

In this section we will find “optimal” allocations of the meta model using the optimization procedure of the previous section. All experiments have been performed on a Parsytec CC-40 machine, consisting of 40 PowerPC 601 processors. The goal of the experiments is to validate whether a phase transition as defined in this chapter actually occurs for the task allocation problem if real programs and parallel machines are considered. The results of a random task graph instance with \(n = 64\) and \(\gamma = 0.2\) is given in Fig. 3.23 as \(\beta\) is varied. Obviously a sharp transition is present from sequential to parallel allocation. Another result, for a larger task graph with 128 tasks and \(\gamma = 0.5\) is presented in Fig. 3.24.
Start with a decomposition on 1 processor

diviser := 1

Do

MigrationDirection := random(Emigrate, Immigrate)
P' := random(0, P-1)
n' := Number of tasks on P' / diviser
x := random(0, n')

For n'

  Migrate random task in MigrationDirection from/to P'

  t_total := DoMetaModel()

  If (t_total > t_prev)
      Redo migration
  Else
      t_prev := t_total
      If (EvolutionIteration mod SomeNumber = 0)
          diviser := diviser + 1

Until Converged

Figure 3.22: Pseudo code for the TAP optimization process

This task graph has been subjected to optimization on a 4 and an 8 processor partition. Again a sharp transition can be observed for both processor partitions. Not that the transition point of all three experiments approximately coincide.

3.7.4 Discussion

In this section we have constructed a meta model and an optimization procedure for the allocation of parallel applications with random interactions between the tasks. The purpose of this section has been to show that the qualitative results of the TAP model can be validated for “real” parallel applications. The P-CAM framework is shown to be of great value for this research. The arbitrariness of the interconnection structure and the migration functionality offered by this framework have enabled the implementation of the meta model and a local search optimization procedure.

The experimental results of several random interaction structures subjected to a local optimization procedure unambiguously demonstrate the presence of
transition from sequential to parallel optimal allocation. Moreover, the results show that the number of tasks ($n$) and the number of processors ($P$) do not influence the location of the transition significantly. This result has also been derived for the theoretical TAP model. However, the random task graph connectivity ($\gamma$) does not induce a transition when it is varied. This is caused by the
way tasks communicate with tasks on other processors. If a task has multiple connections to another task on a different processor, P-CAM sends its contents only once to the other processor. This is not modeled correctly in the TAP cost function, where each connection to another processor contributes to the communication volume. Other shortcomings of the TAP cost functions are its inability to model communication latencies and the difficulty of matching the application and machine parameters to corresponding values of the $\beta$ and $\gamma$ parameters.

### 3.8 Conclusions

The results presented in this chapter clearly show that the task allocation problem exhibits a variety of interesting properties similar to complex behavior found in complex physical systems. Our approach to study the task allocation problem, by analyzing “ground states” of parameterized instances, can be considered new and may be used to study other combinatorial optimization problems as well.

For specific parameter sets the task allocation problem only exists in a small parameter range. Outside this range the problem is trivial. The problem becomes complex in the region 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.

Different allocation regimes are summarized in Fig. 3.26. The **sequential allocation** region only contains optima where all tasks are allocated to one processor. The **semi-parallel allocation** region corresponds to the situation, where not all available processors are necessarily used, due to the high competition between the calculation and communication terms. 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 onto 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 a weakly connected task graph. For increasing task graph sizes, the transition region narrows. This implies the existence of exactly two regions of task allocation order in the limit $n \to \infty$. Hence, for large task graphs that display long range interactions, the TAP is trivial for allocation on fully connected parallel computers.

The parallel- and sequential phases and the separation as predicted by Eqs. (3.52) and (3.53) are depicted in a phase diagram (see Fig. 3.25).

To validate the key characteristic of the TAP, the phase transition, we have optimized the allocation of a meta model defined on random task graphs. The meta model has been implemented straightforwardly in P-CAM, exploiting the capability of the framework to use arbitrary graphs and to migrate tasks between different processor partitions. Again a sudden transition from a sequential to a parallel phase is found by increasing the $\beta$ parameter. However, the meta model also shows shortcomings to the theoretical TAP cost function. Increasing the connectivity ($\gamma$) of the task graph does not induce a phase transition and mod-
Figure 3.25: The phase diagram of the TAP. The circle corresponds to the situation \((\beta, \gamma) = (1/2, 1)\). In this case \(\lambda = 0/0\) (indeterminate). The separating line between the two phases corresponds to \(\beta(\gamma) = \gamma / (1 + \gamma)\).

Figure 3.26: The different allocation regimes in the task allocation problem for varying \(\beta\).

3.8 Conclusions

Modeling the correct values of \(\beta\) and \(\gamma\) is a very difficult task, also keeping in mind that the cost function does not model communication latencies.

It is clear that this new way of dealing with computer science problems is still in its infancy. Besides a realistic formulation of a TAP cost function many fundamental issues remain unsolved. A more detailed analysis of the geometric properties of TAP solutions will probably reveal a plethora of other phases other than just “sequential” and “parallel.” Our current definition of the order parameter does not capture the specific ordering of the tasks to processors. More realistic task graphs need to be studied. For example the effect of introducing short range locality in both the task graph as well as the processor topology. Furthermore, the TAP cost function needs to be adapted such that it can be applied to load balancing of dynamic and heterogeneous parallel applications on dynamic and heterogeneous parallel computers.
On the Complexity of Task Allocation

3.8 Conclusions

The results presented in this chapter clearly show that the task allocation problem exhibits a variety of interesting properties similar to complex behavior found in complex physical systems. Our approach to study the task allocation problem by analyzing parameterized TAP cost functions, can be considered new and may be used to study other combinatorial optimization problems. The TAP model can be extended to incorporate additional features such as the use of parameterized TAP cost functions. This allows for a more flexible approach to controlling the behavior of the system. The results presented in this chapter are complemented by the parallel algorithms that we have developed for solving the TAP model. We believe that these algorithms will find applications in various fields such as computer science, engineering, and artificial intelligence. The parallel and sequential phases and the separation as predicted by Eqs. (3.33) and (3.34) are depicted in a phase diagram (see Fig. 4.2).

To validate the key characteristics of the TAP model, the phase transition, we have optimized the allocation of a meta model defined on random task graphs. The meta model has been implemented straightforwardly in PCAM, exploiting the capabilities of the framework to use arbitrary graphs and to migrate tasks between different processor partitions. Again, a sudden transition from a sequential to a parallel phase is found by increasing the $\beta$ parameter. However, the meta model also shows shortcomings in the theoretical TAP cost function. Increasing the connectivity $\gamma$ of the task graph does not induce a phase transition and can