Phase Transition in Multiprocessor Scheduling

Heiko Bauke,[1,*] Stephan Mertens,[1,2,†] and Andreas Engel[1,‡]

[1]*Institut für Theoretische Physik, Otto-von-Guericke Universität, PF4120, 39016 Magdeburg, Germany*
[2]*The Abdus Salam International Centre of Theoretical Physics, St. Costiera 11, 34100 Trieste, Italy*

(Received 5 August 2002; published 17 April 2003)

An "easy-hard" phase transition is shown to characterize the multiprocessor scheduling problem in which one has to distribute the workload on a parallel computer such as to minimize the overall run time. The transition can be analyzed in detail by mapping it on a mean-field antiferromagnetic Potts model. The static phase transition, characterized by a vanishing ground state entropy, corresponds to a transition in the performance of practical scheduling algorithms.

DOI: 10.1103/PhysRevLett.90.158701 PACS numbers: 89.20.Ff, 02.60.Pn, 64.60.Cn, 89.70.+c

Statistical mechanics has been dramatically expanding its range of applicability in the past years and continues to do so. Among the more promising directions is the analysis of problems in computational complexity [1,2]. Here the relevant questions can often be formulated in a manner surprisingly similar to well-established problems in statistical mechanics, they are concerned with the scaling behavior of systems in a suitably defined thermodynamic limit, and there is an impressive background of "experimental data" in the form of extensive simulation results produced by computer scientists.

A central issue in this field is the identification and characterization of different types of phase transitions in *randomly generated instances* of *NP*-hard problems. Contrary to the worst-case oriented methodology of computer science, statistical mechanics is able to describe the *typical* complexity of a problem which is often more relevant for practical applications. Often one may identify regions in the parameter space where typical instances can be solved without exponential search, separated from other regions where exponential search is mandatory. With increasing size N of the problem, the transitions between these regions become sharp and exhibit properties as known for phase transitions in statistical mechanics. A transition in the typical algorithmic complexity normally corresponds to a structural change in the typical instances of the random ensemble. The latter in turn can be analyzed using the methods and notions from statistical mechanics [1]. An outstanding example for the fruitfulness of this interdisciplinary approach is the analysis of K satisfiability with the replica [3] and the cavity method [4].

In the present Letter, we show that the problem of load balancing, i.e., of evenly distributing the workload on the processors of a parallel computer, exhibits such an "easy-hard" phase transition. We identify the control parameter of this transition and analytically calculate its value including finite size corrections by mapping the problem on an antiferromagnetic Potts model. Finally, we identify dynamical consequences of this phase transition in extensive numerical simulations using a particular algorithm.

In the multiprocessor scheduling problem (MSP), we are given q identical processors and N independent tasks with running times $a_i \in \mathbb{N}$. We are then required to find a *schedule*, i.e., an assignment of the N tasks to the q processors such as to minimize the largest task finishing time (makespan). MSP is an *NP*-hard problem [5,6], which basically means that no one has ever found a solution algorithm that is (for the worst case) significantly faster than exhaustive search through all $\mathcal{O}(q^N)$ possible schedules. On the other hand, numerical experiments [7] with a_i being *random* B-bit integers reveal two distinct regimes: For small values of $\kappa = B/N$, typical instances can be solved without exponential search, whereas for large values of κ exponential search is needed.

To analyze the situation analytically, we define a schedule as a map $s : \{1, \ldots, N\} \mapsto \{1, \ldots, q\}$ with $s(i) = \alpha$ denoting that task i is assigned to processor α. The problem is then to minimize the makespan,

$$T(s) = \max_{\alpha} \left\{ A_\alpha = \sum_{i=1}^{N} a_i \delta(s(i) - \alpha) \right\}. \tag{1}$$

A_α is the total workload of processor α and δ is the usual Kronecker symbol. The structural change in MSP that lies underneath the algorithmic phase transition is the appearance of *perfect* schedules. Let $r = \sum_j a_j \bmod q$ and $\lfloor x \rfloor = \max\{n | n \le x, n \in \mathrm{Z}\}$. A schedule with

$$A_\alpha - \left\lfloor \frac{1}{q} \sum_{j=1}^{N} a_j \right\rfloor = \begin{cases} 1 & \text{if } 1 \le \alpha \le r \\ 0 & \text{if } r < \alpha \le q \end{cases} \tag{2}$$

[and its $\binom{q}{r}$ equivalent rearrangements] is called perfect since it obviously minimizes the makespan T, Eq. (1). Whenever an algorithm runs into a perfect schedule, it can stop the search possibly before having explored an exponential number of schedules. Numerical simulations [7] in fact indicate that the probability that a random instance has a perfect schedule decreases from 1 for

 © 2003 The American Physical Society

$\kappa = 0$ to 0 as $\kappa \gg 1$, and for large N the probability jumps abruptly from 1 to 0 as κ crosses a critical value $\kappa_c(q)$.

It is convenient to encode the schedules using Potts vectors [8] since these reflect the symmetry of the problem: All that matters is whether two tasks are on the same processor or not. See [9] for other optimization problems that can be encoded with Potts vectors. Potts vectors $\vec{e}^{(\alpha)}$ are $(q-1)$-dimensional unit vectors pointing at the q corners of a $(q-1)$-dimensional hypertetrahedron (see Fig. 1 for the case $q = 3$). This implies that the angle between two Potts vectors is the same for all pairs of different vectors,

$$\vec{e}^{(\alpha)} \cdot \vec{e}^{(\beta)} = \frac{q\delta(\alpha - \beta) - 1}{q - 1}. \tag{3}$$

A schedule is encoded by N Potts vectors $\vec{s}_j$, where $\vec{s}_j = \vec{e}^{(\alpha)}$ means that task j is assigned to processor α, i.e.,

$$\vec{s}_j = \sum_{\beta=1}^{q} \delta(s(j) - \beta)\vec{e}^{(\beta)}. \tag{4}$$

The *target vector*,

$$\vec{E}(\{\vec{s}\}) = \sum_{j=1}^{N} a_j \vec{s}_j, \tag{5}$$

encodes the running time of all processors,

$$A_\alpha - \frac{1}{q}\sum_{j=1}^{N} a_j = \frac{q-1}{q}\vec{E} \cdot \vec{e}^{(\alpha)}, \tag{6}$$

and minimizing T, Eq. (1), is equivalent to minimizing $\vec{E}$ with respect to the supremum norm [10].

For integer values a_j, the minimal change of a schedule is to remove 1 from one processor and add it to one of the other $q-1$ processors. Hence, possible values of $\vec{E}(\{\vec{s}\})$ are points on a $(q-1)$-dimensional Bravais lattice with primitive vectors,

$$\vec{b}_\alpha = \vec{e}^{(\alpha)} - \vec{e}^{(q)} \qquad \alpha = 1, \ldots, q-1. \tag{7}$$

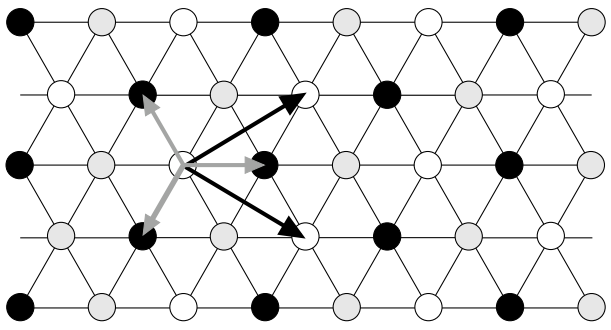

FIG. 1. Lattice of target vectors for $q = 3$ with the three Potts vectors (gray) and the two primitive vectors $\vec{b}_\alpha$ (black). The white (gray, black) lattice points correspond to $\sum a_j \bmod 3 = 0(1, 2)$.

These primitive vectors span a sublattice of the lattice generated by $q-1$ Potts vectors. The sublattice contains every qth point of the Potts lattice and, correspondingly, there are q classes of such lattice points depending on $\sum a_j \bmod q$ (Fig. 1). The volume $V(q)$ of the primitive cell in our sublattice can be calculated from the Gram determinant,

$$V^2(q) = \det(\vec{b}_\alpha \cdot \vec{b}_\beta) = \frac{q^q}{(q-1)^{q-1}}. \tag{8}$$

The average number Ω of schedules with target $\vec{E}$ is

$$\Omega(\vec{E}) = \mathrm{Tr}_{\{\vec{s}\}}\left\langle \delta\left(E - \sum_{j=1}^{N} a_j \vec{s}_j\right)\right\rangle, \tag{9}$$

where $\langle\cdot\rangle$ denotes averaging over the independently and identically distributed random numbers a_j. For fixed schedule $\{\vec{s}_j\}$ and large N, the sum $\sum_{j=1}^{N} a_j \vec{s}_j$ is Gaussian with mean

$$\langle \vec{E} \rangle = \langle (E_1, \ldots, E_{q-1}) \rangle = \langle a \rangle \sum_{j=1}^{N} \vec{s}_j =: \langle a \rangle \vec{M}, \tag{10}$$

and covariance matrix,

$$\langle E_i E_k \rangle - \langle E_i \rangle \langle E_k \rangle = \sigma_a^2 \sum_{j=1}^{N} (\vec{s}_j)_i (\vec{s}_j)_k =: \sigma_a^2 g_{i,k}, \tag{11}$$

with $\sigma_a^2 = \langle a^2 \rangle - \langle a \rangle^2$. The "magnetization" $\vec{M}$ and the variance matrix $\mathbf{g} = (g_{i,k})$ depend on the schedule only through the numbers $N_\alpha = \sum_{j=1}^{N} \delta(s(j) - \alpha)$ of tasks assigned to processor α:

$$\vec{M} = \sum_{\alpha=1}^{q} N_\alpha \vec{e}^{(\alpha)} \qquad g_{i,k} = \sum_{\alpha=1}^{q} N_\alpha e_i^{(\alpha)} e_k^{(\alpha)}. \tag{12}$$

The trace over $\{\vec{s}\}$ is basically an average over all trajectories of a random walk in $q-1$ dimensions. For large N, this average is dominated by trajectories with $\vec{M} = 0$, i.e., $N_\alpha = N/q$. For these trajectories, the matrix $\mathbf{g}$ is diagonal,

$$g_{i,k} = \frac{\delta(i-k)}{q-1}, \tag{13}$$

and we have basically an independent random walk in each of the $q-1$ directions of our lattice. The probability to occupy after N steps a position E_α away from the origin in direction α is therefore given by

$$p(E_\alpha) = \frac{\sqrt{q-1}}{\sqrt{2\pi N \langle a^2 \rangle}} \exp\left(-\frac{(q-1)E_\alpha^2}{2N\langle a^2 \rangle}\right). \tag{14}$$

The probability of finding the $q-1$ walkers at $\vec{E}$ reads

$$p(\vec{E}) = \left(\frac{(q-1)}{2\pi N \langle a^2 \rangle}\right)^{(q-1)/2} \exp\left(-\frac{(q-1)|\vec{E}|^2}{2N\langle a^2 \rangle}\right). \tag{15}$$

To get the number of schedules with given target vector

$\vec{E}$, we have to multiply the density $p(\vec{E})$ by q^N and the volume $V(q)$ of the primitive cell of our Bravais lattice:

$$\Omega(\vec{E}) = \frac{q^N q^{q/2}}{(2\pi N\langle a^2\rangle)^{(q-1)/2}} \exp\left(-\frac{q-1}{2N\langle a^2\rangle}|\vec{E}|^2\right). \quad (16)$$

Note that, as long as $|\vec{E}| = O(1)$ for $N \to \infty$, $\Omega(\vec{E})$ is almost independent of $|\vec{E}|$; i.e., there are as many perfect schedules as there are any suboptimal schedules with $|\vec{E}| = \mathcal{O}(1)$.

The density of *scalar* quantities like $|\vec{E}|$ gets a factor $|\vec{E}|^{q-2}$ from the volume element in $(q-1)$-dimensional spherical coordinates. For $q > 2$ this leads to a maximum of the microcanonical entropy at some value $|\vec{E}| > 0$, and this maximum gets sharper with increasing q, a scenario that has been observed in Monte Carlo simulations [13]. However, this implies no fundamental difference between $q = 2$ and $q > 2$ as claimed in [13] but is of purely geometrical origin.

For the location of the phase transition, we can concentrate on perfect schedules, i.e., we set $|\vec{E}| = \mathcal{O}(1)$. In accordance with common simulation practice, we assume that the a_j are uniformly distributed κN-bit integers. From

$$\langle a^2\rangle = \tfrac{1}{3}2^{2\kappa N}(1 - \mathcal{O}(2^{-\kappa N})), \quad (17)$$

we get

$$\log_2\Omega(0) = N(q-1)(\kappa_c - \kappa), \quad (18)$$

with

$$\kappa_c = \frac{\log_2 q}{q-1} - \frac{1}{2N}\log_2\left(\frac{2\pi N}{3q^{q/(q-1)}}\right). \quad (19)$$

The leading term in this expression can be intuitively understood by the following heuristic argument [7]: Given the values of the a_i are κN-bit integers, the workloads defined by Eq. (2) are (neglecting for the moment carry bits) also κN-bit integers. The probability that a randomly chosen schedule $s(i)$ realizes a particular value of A_1 is therefore $2^{-\kappa N}$. Neglecting correlations, the chance to realize all the workloads defined in Eq. (2) is, hence, $2^{-(q-1)\kappa N}$ (A_q is fixed implicitly). Since there are q^N different schedules, the number of perfect ones is roughly given by $q^N 2^{-(q-1)\kappa N}$. This number is large for small κ but becomes exponentially small for $\kappa > \kappa_c = \log_2(q)/q - 1$. The second term in Eq. (19) represents finite size corrections to this leading behavior. For $q = 2$, Eqs. (18) and (19) reduce to the known results for number partitioning [14,15].

According to Eq. (18), the ground state entropy $\log_2\Omega(0)$ is a linear function of κ for large N. In fact, this linearity already holds for rather small values of N, as can be seen from numerical enumerations (Fig. 2). Linear regression on the data for $\log_2\Omega(0)$ gives numerical values for $\kappa_c(N)$. These values in turn agree well with the predictions of Eq. (19) for larger values of N (Fig. 3).

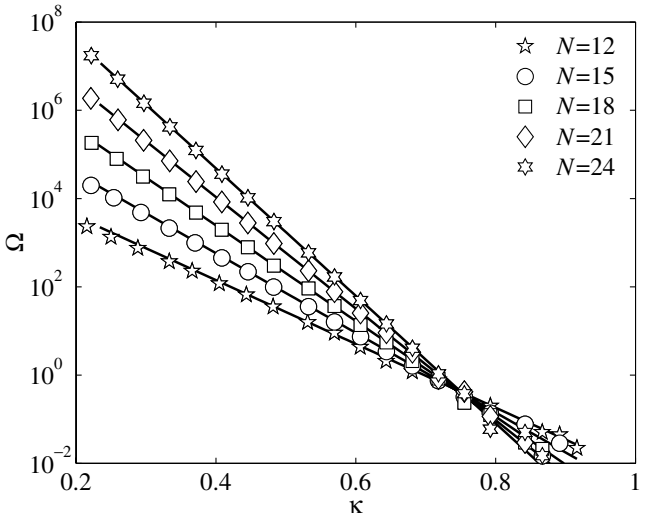

FIG. 2. Numerical measurements of Ω for $q = 3$. Symbols are averages over 10^3 random instances with $\sum_j a_j \bmod q = 0$; lines are given by Eq. (18). The error bars of the enumeration data are smaller than the symbols.

For $\kappa > \kappa_c$, Eq. (16) predicts that the expected number of target vectors decays exponentially with N for any *fixed* $\vec{E}$. The integrated density is finite, however. From numerical simulations we know that in this regime the ground state is unique up to permutations, the average ground state energy E_0 is, hence, implicitly given by $\int^{E_0} d^{q-1}E\ \Omega(\vec{E}) = q!$, or

$$E_0 = \sqrt{\frac{2N}{3(q-1)}}\sqrt[q-1]{q!\Gamma\left(\frac{q+1}{2}\right)}2^{N(\kappa-\kappa_c)}. \quad (20)$$

This equation agrees very well with the numerics [16] and with the known results for $q = 2$ [14,15].

Until now, we have discussed *static* properties of the random MSP. How do they affect the *dynamical* behavior of search algorithms? An obvious algorithm is to sort the

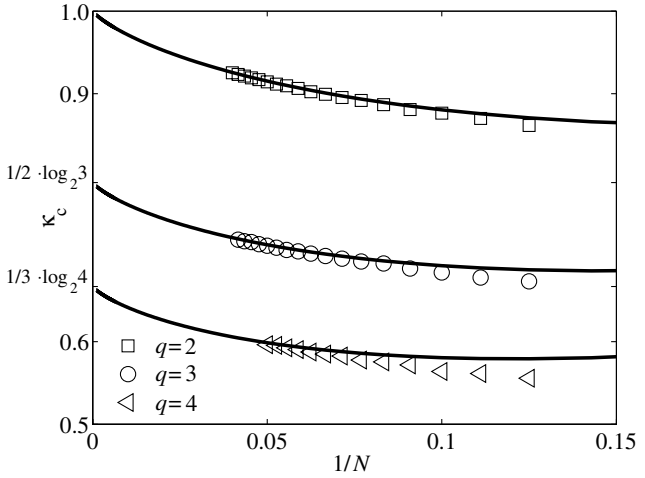

FIG. 3. κ_c from linear regression of the numerical data for Ω (symbols) compared to Eq. (19) (curves).

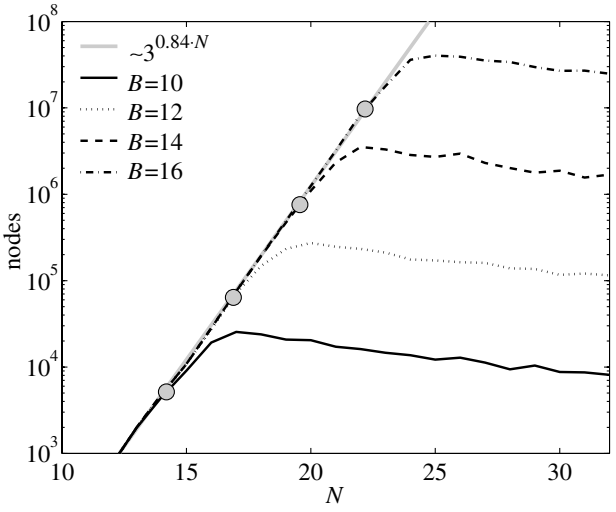

FIG. 4. Number of nodes traversed by the complete greedy algorithm for $q = 3$ and fixed number of bits B. The circles mark the critical system size N_c given by Eq. (21).

tasks a_i in decreasing order and to assign the first (and largest) task to processor 1. The next tasks are each assigned to the processor with the smallest total workload thus far. Proceed until all tasks are assigned. Ties are broken by selecting the processor with the lower rank. This so-called greedy heuristics usually produces poor schedules, but it can be extended to an algorithm that yields the optimum schedule. Instead of assigning a task to one processor, the extended algorithm branches: In the first branch it follows the heuristic rule and assigns the task to the processor with the lowest workload; in the second branch it selects the processor with the second lowest workload, and so on. Ignoring ties this algorithm generates the complete search tree with its q^N leaves and, hence, will eventually find the true optimum. This algorithm is known as complete greedy algorithm (CGA) [17]. Of course it is exponential in the worst case but with pruning we can hope to achieve a speedup for the typical case. The most efficient pruning rule is to simply stop the moment one hits upon a perfect schedule. Another pruning rule applies if the sum of the unassigned tasks is smaller than the difference between the current maximal and minimal workloads. In this case, one can simply assign all remaining tasks to the processor with the minimum workload.

In our simulations we fix the number B of bits in the a_j and measure the number of nodes traversed by CGA as a function of N. In this case κ_c translates into a critical value N_c for the system size, N_c being the solution of

$$\frac{B}{N_c} = \frac{\log_2 q}{q-1} - \frac{1}{2N_c}\log_2\left(\frac{2\pi N_c}{3q^{q/(q-1)}}\right). \qquad (21)$$

Figure 4 shows a typical result for $q = 3$. For $N < N_c$, the number of nodes traversed by CGA increases like 3^{xN} with $x \approx 0.84$. The fact that $x < 1$ is due to the pruning. As soon as $N > N_c$, the pruning by perfect solutions takes effect and the growth slows down significantly. Eventually the search costs even decrease with increasing N. This indicates that the algorithm takes advantage of the growing number of perfect solutions, although their *relative* number is still exponentially small. There are algorithms that outperform simple CGA, but the differences show up only for $N > N_c$: With better algorithms, the subexponential growth and the decrease of the search costs set in at values of N closer to N_c [17].

In conclusion, we have shown that multiprocessor scheduling has a phase transition controlled by the variance in size of the individual tasks. The "easy" phase is characterized by an exponential number of perfect schedules, the "hard" phase by the absence of perfect schedules. Compared to the analysis of the special case $q = 2$ in [14], our "microcanonical" approach is more transparent and requires no *ad hoc* measures to cope with a negative ground state entropy.

Discussions with Ido Kanter and Robert Urbanczik are gratefully acknowledged. S. M. enjoyed the hospitality of the ICTP, Trieste. All numerical simulations have been done on our Beowulf cluster TINA [18].

*Email address: heiko.bauke@physik.uni-magdeburg.de
†Email address: stephan.mertens@physik.uni-magdeburg.de
‡Email address: andreas.engel@physik.uni-magdeburg.de

[1] Special issue on "Phase Transitions in Combinatorial Problems," edited by O. Dubois, R. Monasson, B. Selman, and R. Zecchina [Theor. Comput. Sci. **265** (2001)].
[2] P.W. Anderson, Nature (London) **400**, 115 (1999).
[3] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyanksy, Nature (London) **400**, 133 (1999).
[4] M. Mézard, G. Parisi, and R. Zecchina, Science **297**, 812 (2002).
[5] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness* (Freeman, New York, 1997).
[6] S. Mertens, Comput. Sci. Eng. **4**, 31 (2002).
[7] I. P. Gent and T. Walsh, in *Proceedings of ECAI-96*, edited by W. Wahlster (Wiley, New York, 1996), pp. 170–174.
[8] F. Wu, Rev. Mod. Phys. **54**, 235 (1982).
[9] I. Kanter and H. Sompolinsky, J. Phys. A **20**, L673 (1987).
[10] In L_2 norm, the minima of $\vec{E}$ correspond to the ground states of a mean-field Potts antiferromagnet with Mattis-like couplings [11,12].
[11] D. Mattis, Phys. Lett. A **56**, 421 (1976).
[12] J. Provost and G. Vallee, Phys. Rev. Lett. **50**, 598 (1983).
[13] A. Lima and M. de Menezes, Phys. Rev. E **63**, 020106(R) (2001).
[14] S. Mertens, Phys. Rev. Lett. **81**, 4281 (1998).
[15] C. Borgs, J. Chayes, and B. Pittel, Rand. Struct. Alg. **19**, 247 (2001).
[16] H. Bauke and S. Mertens (to be published).
[17] R. E. Korf, Artificial Intelligence **106**, 181 (1998).
[18] URL: http://tina.nat.uni-magdeburg.de.