Stochastic and Deterministic Approaches to the Number Partitioning Problem

Lana Awad,[1,*] Erik Bauch,[2,†] Tamara Dordevic,[2,‡] and Gregory Phelps[2,§]

[1] *Harvard TH Chan School of Public Health, Boston, MA 02115, USA*
[2] *Department of Physics, Harvard University, Cambridge, Massachussetts 02138, USA*

We explore different stochastic optimization algorithms applied to solving the number partitioning problem, and compare them against well-known deterministic benchmarks - the Greedy algorithm and the Karmarker and Karp (KK) heuristic. In the partitioning problem, we are given a set of numbers and are tasked with dividing them into two partitions such that the difference between the sum of elements in each partition is minimized. The algorithms we consider include stochastic algorithms, such as simulated annealing (SA) and modifications to SA such as parallel tempering and stochastic tunneling. We also consider a genetic algorithm. We consider these different approaches and try to characterize the performance of each for this task and distinct data sets. Finally, we discuss the use of quantum annealing in finding a solution to the number partitioning problem and the significant speed-up it might provide.

I. INTRODUCTION

NP-complete problems are of central interest in Computer Science. Their solutions cannot be verified in polynomial time, but often require an exhaustive search through the entire solution space. Many combinatorial problems fall into this category, such as the Traveling Salesman problem and the Knapsack problem. The two approaches to finding approximate solutions to NP-complete problems are through deterministic and stochastic algorithms. Stochastic algorithms often include a Monte-Carlo-style search through the solution space, while deterministic algorithms include heuristics tailored to particular problems.

In this paper we explore different approaches to finding an optimal solution for a famous NP-hard problem, the Number Partitioning problem. The Number Partitioning problem can be stated as follows: given a set of N numbers, split them into two partitions so that the difference between the sums of the two partitions is minimal. This problem can be also generalized to M partitions, in which case it has M^N possible solutions. Formally, it amounts to minimizing the energy functional:

$$E[P] = \sum_{m=1}^{M} \left| \sum_{i \in P_m} a_i - \frac{1}{M} \sum_{i=1}^{N} a_i \right|^2 \qquad (1)$$

where P_m is the partition containing the indices of numbers in our data set $A = \{a_1, a_2 ... a_N\}$. The minimal theoretical value of $E[P]$ is zero, but its existence is not guaranteed in general. This is especially obvious if we take small data sets. For example, if we want to partition $\{2, 3, 4\}$ into two partitions, the best solution is $P_1 = \{2, 3\}$,$P_2 = \{4\}$, and the energy of this solution is 0.5.

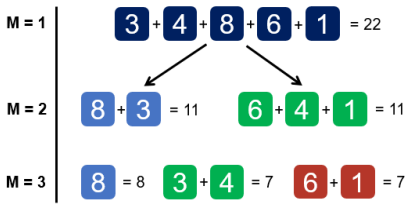

FIG. 1. Example of a partitioning problem for $N = 5$. Here we show the optimal partitions for $M = 1, 2, 3$. Notice that we find a perfect solution for $M = 2$ where the energy function evaluates to 0.

The number partitioning problem is applicable to many fields including scheduling tasks in parallel computing. Moreover, NPP is essential in many everyday and business scheduling activities, such as splitting people into optimal teams or splitting the indivisible assets in court cases [3].

We look for solutions to the problem with $M = 2$ partitions with six different algorithms. Our main focus are stochastic algorithms which give approximate solutions. We implement four of these algorithms - Simulated Annealing, Stochastic Tunneling, Parallel Tempering and the Genetic Algorithm. In order to have a benchmark for comparison, we also implement two deterministic algorithms known to perform well for this task - the Greedy algorithm and the Karmarkar-Karp Heuristic.

This paper is organized as follows: in Section II

* lana.awad90@gmail.com
† ebauch@physics.harvard.edu
‡ tamaradordevic@g.harvard.edu
§ gphelps@physics.harvard.edu

and III we introduce the deterministic and stochastic algorithms used in detail and discuss how they can be applied to the number partitioning problem. In section IV we discuss the results we found when comparing the algorithms and which algorithm generally performs best. Finally, in section V, we briefly describe how solving this problem on a quantum computer can provide a significant speed-up over classical algorithms.

II. DETERMINISTIC ALGORITHMS

Deterministic algorithms produce results completely dictated by the set and no other random factors. There are several algorithms suited to this problem, which, relatively consistently, produce near optimal results. Here we explore two common algorithms: the greedy algorithm and the Karmarkar-Karp Heuristic [4]. These algorithms are used as a benchmark for the stochastic algorithms. In general, the deterministic algorithms will scale more favorably with set size - resulting in complexities with polynomial scaling.

A. Greedy

The Greedy algorithm is the simplest of the deterministic algorithms - it is extremely fast and gives reliable results for large set sizes. The algorithm works by placing the largest number into the bin with the least total value. This continues until all elements in the set are exhausted. An example of this process for $N = 5$ and $M = 2$ is shown in Fig. 2. The example begins with the set $S = \{5, 8, 4, 7, 6\}$ and orders the set $S = \{8, 7, 6, 5, 4\}$. The partitioning starts by placing the largest value, 8, into the first bin. According to the recipe previously outlined, it then places 7 into the second bin. It continues until it finds the partitions $P_1 = \{8, 5\}$ and $P_2 = \{7, 6, 4\}$. Clearly the algorithm's complexity is dictated by the sorting, which is of order $\mathcal{O}(N \log N)$. The remainder of the algorithm has complexity N. As is exemplified by the example in Fig. 2, the algorithm does not always find the optimal solution, especially for smaller sets. Luckily, there are more sophisticated deterministic algorithms that typically yield better partitions.

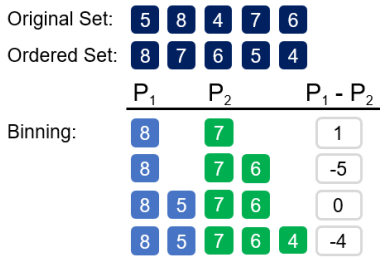

FIG. 2. An example of the greedy algorithm for $N = 5$ and $M = 2$ partitioning problem. In each iteration, the numbers are sorted and the next largest number is placed in the bin with the smaller sum until all numbers are exhausted

B. Karmarkar-Karp Heuristic

KKH is a heuristic designed for the partition problem and has the best performance among the deterministic methods. For the same set above $S = \{5, 8, 4, 7, 6\}$, KKH begins with ordering the set $S = \{8, 7, 6, 5, 4\}$, Fig. 3. It then replaces the largest two items in the set by their difference, effectively positioning them in different partitions, except the decision of where to place the numbers is left till later. So in the first iteration, 8 and 7 are replaced by a 1. The algorithm keeps track of where the difference came from in a tree data structure. The iterations end with the final difference left as the only number of the list. That is the cost/error of this partition. The partitions of the solution are then reconstructed by propagating back from the final difference up the tree, replacing each difference by the values that gave rise to it.

III. STOCHASTIC ALGORITHMS

A. Simulated Annealing

Simulated Annealing (SA) is a very efficient technique for finding an approximate ground state of a potential energy landscape by making stochastic Monte Carlo steps. The main idea in SA is that we gradually improve our current solution by suggesting a slightly changed new solution, and accepting it with probability:

- $p = 1$, if $E_{new} > E_{old}$
- $p = \exp[-(E_{new} - E_{old})/T]$, otherwise

where we need to choose current "annealing temperature", T. The main difficulty in performing an

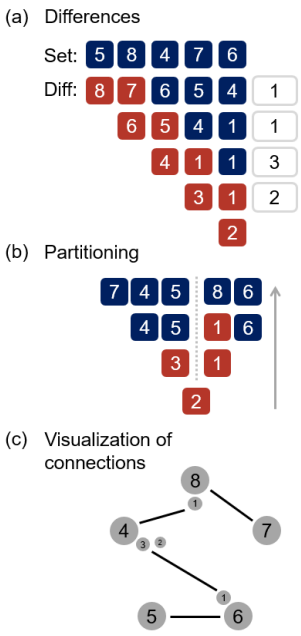

FIG. 3. An example of KKH algorithm applied to $N = 5$ and $M = 2$ partitioning problem. The largest two numbers in each iteration are replaced by their difference. The optimal solution is reconstructed by propagating back up the tree.

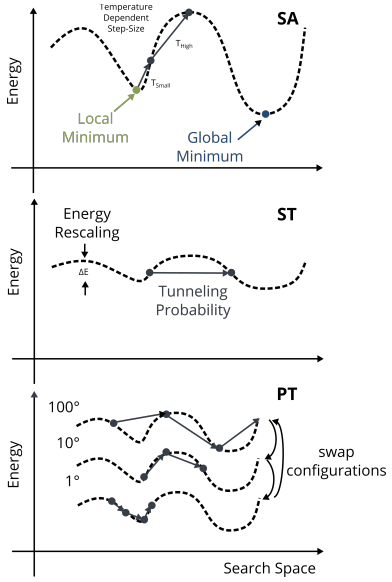

FIG. 4. Comparison of different variations of Simulated Annealing. Top: Regular SA. Middle: Stochastic Tunneling, where the energy landscape is rescaled so that we can tunnel through the high energy areas faster. Bottom: Parallel Tempering, where we run multiple copies of SA at the same time at different temperatures, and then swap the configurations.

efficient SA is in choosing and adjusting T as we progress through the simulation. Usual SA does this in the following way:

1. Choose starting temperature, T_0
2. After every n iterations, set $T_i = \alpha T_{i-1}$, where α is a cooling coefficient smaller than 1
3. If $T_i < T_{min}$, set $T_i = T_{set}$

The cooling schedule is therefore determined by 5 parameters, T_0, n, α,T_{min} and T_{set}. However, neither one of these parameters depends on the current state of our search, which may get us trapped in a local minimum or may lead us to jump over it.

B. Stochastic Tunneling

Stochastic Tunneling (ST) algorithm is an attempt at solving this problem - it modifies the cooling schedule of SA so that it adapts to the current state of our search. It is especially fitted for energy landscapes with large sudden jumps. ST was first suggested by Wenzel et al. in [5]. In ST one calculates the transition probability p by applying the SA transition formula to the mapped energies. An energy E_1 is mapped to

$$f(E_1, E_0) = 1 - \exp[-\gamma((E_1 - E_0))] \qquad (2)$$

where E_0 is the minimal energy found thus far.

Suppose our current energy is E_1, the minimal energy found thus far is E_0 and the next suggested energy is E_2. The probability of accepting it in ST is given by:

$$p = \exp[-(f(E_2, E_0) - f(E_1, E_0))/T] \qquad (3)$$

where $f(E_1, E_2)$ is given by equation 2. This expression can be rewritten as

$$p = \exp[-\frac{1}{T} e^{-\gamma(E_1 - E_0)} \left(1 - e^{-\gamma\Delta}\right)] \qquad (4)$$

where $\Delta = E_2 - E_1$. We can analyze equation 4 in two limits:

1. **Propose a small step $\Delta << 1/\gamma$**: We can expand the second part of equation 4 to get

$$p \approx \exp[-\frac{\gamma\Delta}{T} e^{-\gamma(E_1 - E_0)}]$$

The probability of accepting has the same form as in Simulated Annealing, but the temperature is rescaled to be

$$T_{eff} = \frac{T}{\gamma} e^{\gamma(E_1 - E_0)}$$

Therefore, we effectively warm up our system if $E_1 - E_0 > \ln(\gamma)/\gamma$, and cool it down otherwise. In this way, we explore low energy regions with higher precision (since the temperature is low if $E_1 \approx E_0$) and we go through the high energy region faster.

2. **Propose a large step $\Delta >> 1/\gamma$**: This case corresponds to climbing up a large potential energy hill (or tunneling through it, hence the name Stochastic Tunneling). Here $\exp[-\gamma\Delta] \to 0$, and equation 4 becomes

$$p \approx \exp[-\frac{1}{T} e^{-\gamma(E_1 - E_0)}]$$

The upward moving probability now does not depend on the step size, but only depends on whether we are close to E_0, Fig. 7.

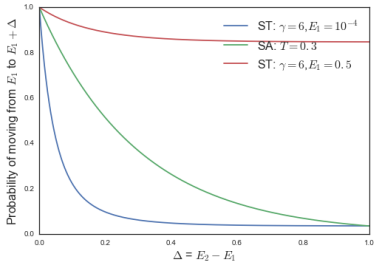

FIG. 5. Comparison of transition probabilities for $T = 0.3$ and $E_0 = 0$. SA has a transition probability that only depends on the energy difference $\Delta = E_2 - E_1$. ST also depends on the difference between the current energy and best energy found so far, E_0 - it moves freely if the current energy is far from E_0, while it cools down if the current energy is close to it.

Stochastic Tunneling is most efficient if we explore both low and high energy regions. In order not to get stuck in a low energy region, we can track the moving average for the last couple of steps for $e^{-\gamma(E_1 - E_0)}$. If we put $a =< e^{-\gamma(E_1 - E_0)} >$, the tunneling probability is given by $e^{-a/T}$. We can now adjust T so that the tunneling probability through a tall barrier never drops under a certain value p_{min}. Our dynamical cooling schedule would then be:

- If $e^{-a/T} < p_{min}$, $T = T + dT$
- Otherwise $T = \alpha T$

where dT is a similar parameter to T_{set} from SA (the "reannealing temperature") and α is the cooling coefficient. Therefore, the cooling schedule in ST is controlled by γ, T_0, α, p_{min} and dT, same number of parameters as SA.

C. Parallel Tempering

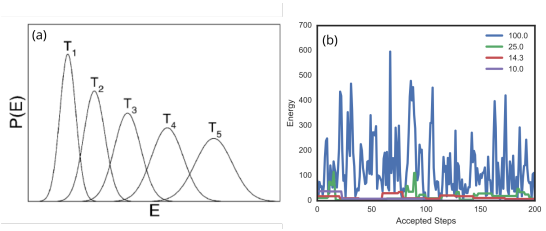

FIG. 6. a) - Schematic

Parallel Tempering (PT) is an extension of Simulated Annealing in which multiple simulations, each at its own fixed temperature, are run in parallel as depicted in Fig. 4. The key idea is that simulations run at higher temperatures can explore a much larger phase space, while simulations at lower temperatures explore the phase space more meticulously. By swapping configurations - or equivalently temperatures - at regular intervals, configurations at high temperatures are made available at low temperatures and vice versa.

Since no deterministic cooling schedule exist, contrary to standard SA, the algorithm is very sensitive to the initial temperatures chosen. Depending on the specific problem, temperatures should be spaced in a way such that the energy histograms of each corresponding simulation have a finite overlap with its neighboring simulations. This is depicted in Fig. 6. At the same time, the temperatures should cover the complete energy spectrum.

In between swap intervals, each individual simulation needs to be given a sufficient amount of time to evolve, making the swap intervals another important input parameter. In addition, a swap between two neighboring simulations i, j with energies E_i, E_j and temperatures T_i, T_j is accepted with probability

$$p = \exp((E_i - E_j)(\frac{1}{kT_i} - \frac{1}{kT_j}))$$

e.g., only when their overlap is finite, and rejected otherwise. This Monte Carlo criterion guar-

anties that a process at higher temperature explores a spaces that is interesting to the neighboring process at lower temperature. In terms of overall performance, it should be noted that for specific problems the significant overhead that comes with running N simulations in parallel is compensated by a convergence that scales even faster than 1/N. In addition, this algorithm is extremely well suited to be run on large clusters with multiple CPUs [2].

D. Genetic Algorithm

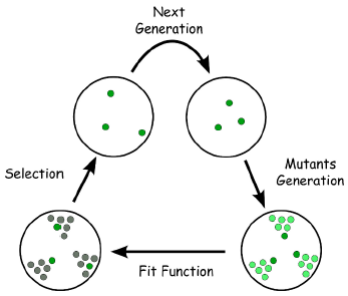

FIG. 7. Overview of genetic algorithm workflow. Every new generation is bred from cross overs and mutation events that take place in the previous generation in a manner dependent on each individual solution's fitness

Genetic algorithms adopt concepts from Darwinian evolution to find solutions for optimisation problems. The algorithm deals with a 'population' of solutions, each associated with a different cost. In each iteration of the algorithm, the population evolves new, better solutions by relying on evolutionary mechanisms like:

- elitism: best solutions in the current population automatically move on to the next generation.
- crossover: best solutions are selected to become parents of new solutions with a probability proportional to their fit, where fit here is the inverse of cost. Two parents then undergo cross over exchanging subsets.
- mutations: random events of inserting and deleting numbers into a subset take place adding stochasticity to the population of solutions.

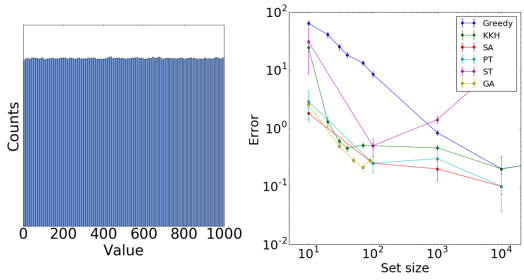

FIG. 8. Performance of different algorithms with uniformly distributed synthetic data of different sizes

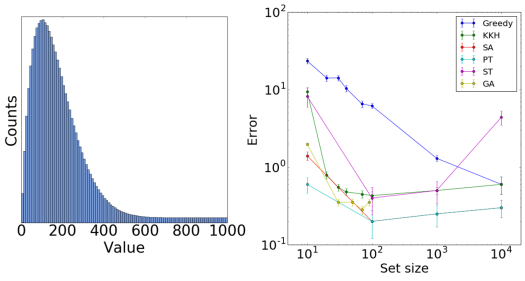

FIG. 9. Performance of different algorithms with skewed distribution of synthetic data of different sizes

IV. RESULTS AND DISCUSSION

We found that parallel tempering performed nearly identical to standard SA and did not provide any improvement, neither in execution time nor in the reduction of error for different set sizes (Fig. 8, Fig. 10,Fig.11). For a fairer comparison, we kept the total number of MC steps constant. After a detailed investigation, we found that the energy diagrams for simulations at different temperatures showed only small variations and significant overlaps. In that case the MC criterion for swapping of temperatures is almost always accepted. It therefore appears that the parallel tempering algorithm reverts to a parallel run of multiple SA with similar temperature, without any enhanced convergence (though we expect a speed up due to the aforementioned parallelization).

As outlined in [5], stochastic tunneling is most well-suited for energy landscapes with abrupt jumps. The energy mapping, outlined in the equation 2, effectively flattens out the high energy part so that we can freely move around it and emphasizes low energy wells. The energies in our problem are, however, quite continuous and do not have sudden jumps. That is why we do not see an improvement in ef-

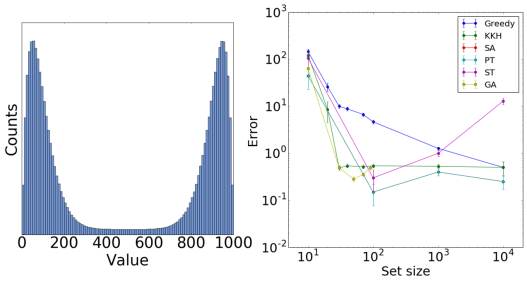

FIG. 10. Performance of different algorithms with synthetic data skewed toward extreme values of different sizes

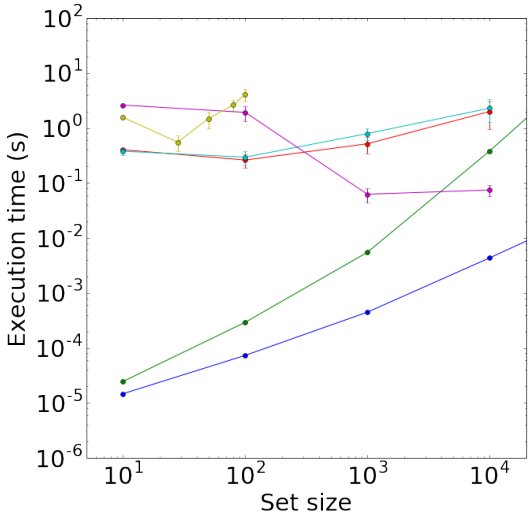

FIG. 11. All code was executed on the same machine using an Intel I7-4790K processor. We find, as expected, that the Greedy and KKH perform the best in execution time. The scaling of the two deterministic algorithms are nowhere near the optimal due to inefficient implementation. Ideally these two should always scale more favorably than the other algorithms.

ficiency or precision when we apply ST to our problem. Moreover, since we check after each proposal if we need to lower or increase the temperature to meet the criterion for p_{min} for tunneling, ST may warm up too much for the exploration of the low energy landscape, which we see as an upward trend in errors as the set size increases. Overall, both ST and PT offer little improvement for problems with slowly varying energy space, and are likely to be better suited for problems where the solution space is bi- or multimodal.

When it comes to the execution time, deterministic algorithms clearly win for small sample sizes, but then gradually meet stochastic algorithms for $N \approx 10^4$. Stochastic algorithms have an almost flat scaling in the execution time with the set size since they never go through all the data points to do branching or comparisons like KKH and Greedy do. Therefore, we can see that stochastic algorithms start outplaying deterministic ones for problems with very high dimensionality, Fig. 11. Overall, stochastic algorithms explored in this paper did not significantly outperform deterministic algorithms we used as a benchmark - Greedy and KKH. However, one should take into consideration that Greedy and KKH presented here were particularly developed to solve the Number Partitioning Problem efficiently, so no wonder they outperform the general stochastic optimization techniques. The power of the four algorithms presented here, SA, ST, PT and GA, is in their wide applicability and good scaling with the problem dimensionality, where they gradually outperform deterministic algorithms in the execution time. Potential improvements to this problem include even more detailed optimization of the parameters used for SA, ST and PT, as well as a more detailed exploration of the energy landscape so that we could deduce where our stochastic search slows down. However, another more exciting improvement in efficiency of solving this problem, as well as many other NP-hard problems, may be achieved with the use of quantum computers. That is why in the next section we briefly outlay how this problem can be mapped to a quantum algorithm.

V. BEYOND CLASSICAL ALGORITHMS

Quantum computation has the potential of solving a multitude of classically difficult problems. These range from prime factorization to general optimization problems. (Citation needed) The two partition NPP is also particularly well suited for a quantum computer due to the bit-wise nature of the solution and the size of the state space. The current state of the art is D-Wave System's D-WAVE 2X quantum annealer, which can potentially handle a set of 1024 numbers for partitioning. The idea of quantum annealing is similar to simulated annealing in that the energy landscape (Hamiltonian in quantum case) is changed in a way to allow minimum solutions of the energy functional to be found. However, a quantum annealer actually does the physical annealing procedure (i.e. it cools down a physical system to its lowest energy state), and reads off the solution to

the optimization problem from the final state of the physical system, as opposed to simulating the annealing procedure. Moreover, the annealing procedure is "parallelized" when it is performed on quantum systems due to the extension of the classical bit to the qubit (quantum bit), where the state is no longer 1 and 0, but a complex mixture of both. This extension allows for a simultaneous exploration of a much higher state space. Looking for a ground state of a quantum system, as opposed to a classical physical system, also allows for the tunneling through potential barriers (a quantum effect where a particle can move through a potential wall that is classically forbidden). Overall, quantum annealing can be thought of as simulated annealing performed by nature on multiple processors at the same time.

In order to solve the Number Partitioning Problem on a quantum computer, we need to map this problem to a quantum Hamiltonian. To do so, we define the signed energy functional,

$$F[\mathbf{P}] = \sum_{\mathbf{i}\in\mathbf{P}} \mathbf{a_i} - \sum_{\mathbf{i}\notin\mathbf{P}} \mathbf{a_i} \tag{5}$$

If we assume the existence of an element of set $\mathbf{A}$ in partition P_1 is given by $s_i = -1$ and its association with partition P_2 is given by $s_i = 1$, then we may write the signed energy functional as

$$F[\mathbf{P}] = \sum_{\mathbf{i}} \mathbf{s_i a_i}. \tag{6}$$

The s_i are equivalent to spin in physics. Squaring this, we find a classical Hamiltonian of the Ising form

$$H_{\mathrm{cl}}[\mathbf{P}] = \sum_{\mathbf{i,j}} (\mathbf{a_i a_j}) \mathbf{s_i s_j} = \sum_{\mathbf{i,j}} \mathbf{J_{i,j} s_i s_j}. \tag{7}$$

The extension of this to a quantum Hamiltonian comes from replacing the spins by operators. The Ising Hamiltonian can be simulated on D-Wave Systems D-WAVE 2X using quantum annealing. This was done by Denchev et al. in [1], where they showed a 10^8 faster execution time of a Quantum Annealer than the Simulated Annealing implemented on a classical computer.

[1] Denchev, V., arXiv:1512.02206v4 (2016).

[2] Earl, D. J. and Deem, M. W., Physical chemistry chemical physics : PCCP **7**, 3910 (2005), arXiv:0508111 [physics].

[3] Hayes, B., American Scientist , 484 (2002).

[4] Korf,, Artificial Intelligence , 181203 (1998).

[5] Wenzel, H., Phys. Rev. Lett. 82(1999)3003-3007 (1999).