Variational Methods in Combinatorial Optimization and Phylogeny Reconstruction

Henrik Jönsson

Department of Theoretical Physics
Lund University, Sweden

Thesis for the degree of Doctor of Philosophy

Thesis Advisor: Bo Söderberg

Faculty Opponent: Rémi Monasson

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of Lund University, for public criticism in Lecture Hall F of the Department of Theoretical Physics on Friday, the 14th of December 2001, at 10.15 a.m.

DOKUMENTDATABLAD
enl SIS 61 41 21

Organization
LUND UNIVERSITY
Department of Theoretical Physics
Sölvegatan 14A
223 62 LUND

Document Name
DOCTORAL DISSERTATION

Date of issue
November 2001

CODEN:

Author(s)
Henrik Jönsson

Sponsoring organization

Title and subtitle
Variational Methods in Combinatorial Optimization and Phylogeny Reconstruction

Abstract

Algorithms based on the variational approach, as used in statistical physics, are developed.

For constraint satisfaction problems a novel cost function, based on information-theoretic arguments, is introduced, and an algorithm similar to the mean-field annealing algorithm is proposed. It outperforms the conventional mean-field algorithm, and its performance is comparable to good problem-dedicated heuristics for KSAT and graph coloring.

For nonlinear assignment problems, improvements to mean-field annealing algorithms based on Potts spins are suggested, and confirmed for TSP. Also a more proper variational approach to assignment problems is proposed and analysed.

A novel variational approximation to maximum likelihood is introduced and applied to phylogeny reconstruction. In tests on artificial and real DNA-sequences, the performance is seen to be comparable to that of standard maximum likelihood for reasonably similar sequnces.

Key words
variational, mean-field, annealing, combinatorial optimization, constraint satisfaction, phylogeny, maximum likelihood.

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title

ISBN
91-7874-161-1

Recipient's notes

Number of pages 128

Price

Security classification

Distribution by (name and address)
Henrik Jönsson, Dept. of Theoretical Physics,
Sölvegatan 14 A, S-223 62 LUND

Signature ______________________ **Date** 2001-11-01

To Annika

This thesis is based on the following publications:

I Henrik Jönsson and Bo Söderberg,
An Information-Based Neural Approach to Constraint Satisfaction
Neural Computation **13**, 1827-1838 (2001).

II Henrik Jönsson and Bo Söderberg,
An Information-Based Neural Approach to Generic Constraint Satisfaction
LU TP 00-40, submitted to *Artificial Intelligence*.

III Henrik Jönsson and Bo Söderberg,
Deterministic Annealing and Nonlinear Assignment
LU TP 01-16.

IV Henrik Jönsson and Bo Söderberg,
An Approximate Maximum Likelihood Approach, applied to Phylogenetic Trees
LU TP 01-31, submitted to *Journal of Computational Biology*.

Contents

Introduction

Methods developed in statistical physics have proved to be useful in research fields outside physics such as computer science, statistics, economy and molecular biology. In this thesis, the variational method, developed in statistical physics, is used to approach combinatorial optimization and phylogeny reconstruction.

In statistical physics the aim is to describe global behavior of a system, by using models for the behavior of the parts that are the building blocks of the system. This can for example be to average over atom behavior to predict global features of a gas, or a solid compound. The variational approach is used to approximate the behavior of the included parts of a problem. The approximation is optimized with respect to its parameters to describe an actual behavior as close as possible.

The variational approximation has been proven to be useful when approaching hard combinatorial optimization (CO) problems. A CO problem is defined by a number of discrete variables and a cost is associated with each combination of values (a state) of these variables. The goal is to find the state with minimum cost. It is suitable to use computers to solve CO problems, and most of the research have been performed after the introduction of the computers in the middle of the last century. To some of the CO problems, efficient computer algorithms have been introduced to find optimal solutions, while for others, efficient algorithms are still lacking, despite the immense amount of research that has been directed towards these problems. The question of whether it is possible to find efficient algorithms is still open, and the activity of research in the field is substantial. For some purposes it suffices to have a fast algorithm that finds a cost close to the optimal. In this thesis an algorithm of this type, deterministic annealing (or mean-field annealing), is described and developed, and it is applied to a number of combinatorial optimization problems.

Another problem where the variational method can be used is in phylogeny

reconstruction. When building a phylogenetic tree, the goal is to infer the history of evolution that leads to present species. The underlying assumption is that all species have evolved from one ancestor and the differentiation is due to some kind of branching (e.g. an animal group has been geographyly divided). DNA sequences from different species can be used along with a model for how sequences change by mutation. It is then possible to infer the most likely history for the data according to the model. This approach is called maximum likelihood and a novel approximation to this approach, based on the variational method, is presented in this thesis.

In the next section a short introduction to spin systems is given and the statistical physics description of these systems is presented. The variational approach is introduced, leading to the mean-field approximation of the spin variables, which is the central part of the algorithms in the thesis. After that, the problems studied are described, and the methods used in this research are introduced. First combinatorial optimization is described and then phylogeny reconstruction. The introduction concludes with a short presentation of the papers that the thesis is based on.

Spin Systems

The magnetic properties of particles such as electrons and atoms are often described using spin variables. In statistical physics lots of work has been done on investigating spin systems, as simple models of single spins generate interesting global behavior of the system [10].

A simple model of a spin is a variable, s, with a discrete number of possible states, indicating the direction of the spin. An Ising spin can be in only two states ($s = \pm 1$) and will be used in the following discussion as an example. In a spin system a number of spins interact according to some simple connections. The energy of the system can be formulated as a Hamiltonian function

$$H(\mathbf{s}) = \sum_{ij} J_{ij} s_i s_j, \tag{1}$$

where a nonzero J_{ij} indicates the interaction between spin i and j. If all nonzero connections J are negative we have a ferromagnetic system and it is energetically favorable for the spins to align so that connected spins have the same value (results in a negative contribution to the energy). If all nonzero J are positive we have the opposite situation (anti-ferromagnetic) and connected spins prefer to be in different states. In both these situations it is often no problem to find the lowest energy state, but if there are connections of both signs the system may be frustrated, and spins receive ambiguous information

of what state is energetically favorable. This type of system is called a spin glass and it has the characteristics of having lots of local energy minima with values close to the global minimum.

There is more to a spin system than the spin interactions. A more accurate description of a spin system results if the system is placed in a heat bath, such that a constant temperature, T is obtained and energy is allowed to fluctuate between the system and the heat bath. The probability that a system is in a state $\mathbf{s}$ is then assumed to follow the Boltzmann distribution, $p(\mathbf{s}) = e^{-H(\mathbf{s})/T}/Z$, where $Z = \sum_{\mathbf{s}} e^{-H(\mathbf{s})/T}$ is the normalizing partition function and the summation is over all possible states. A low energy state always has a higher probability to be present, but the temperature regulates to what extent this is obeyed. A high temperature results in a more equal probability for states with different energy, while a low T results in low energy states being much more probable. In the limit of zero temperature, only the lowest energy states have a finite probability.

The Boltzmann distribution is the distribution that minimizes the free energy, F, defined by

$$F = -T\log(Z) =< H > -TS. \tag{2}$$

Here Z is the partition function and $< H >= \sum_{\mathbf{s}} H(\mathbf{s})\exp(-H(\mathbf{s})/T)/Z$ is the average energy. S is the entropy and is defined by

$$S = -\sum_{\mathbf{s}} p(\mathbf{s})\log(p(\mathbf{s})), \tag{3}$$

and it can be seen as a measure of the disorder of the system. As there is a factor T in front of the entropy term, this is dominating for higher temperatures, while it is less important for lower T resulting in the characteristics described above of more energy sensitivity for low T.

The variational approach

As the distribution that minimizes F for a given Hamiltonian H is the Boltzmann distribution, $p(\mathbf{s}) = \exp(-H(\mathbf{s})/T)/Z$, a distribution built from another Hamiltonian H_0, then yields a larger value for the free energy, i.e.

$$F \leq F_v =< H >_0 -TS_0 = F_0 + < H - H_0 >_0, \tag{4}$$

where $< H >_0$, S_0 and F_0 are connected to the distribution from H_0. For complicated systems computational complications may occur when using the correct Boltzmann distribution. Then a simpler distribution (from H_0) can be

used and optimized with respect to parameters in H_0 to minimize the gap between F and F_v. This is exactly the variational method that will be frequently used in this thesis [3].

After some manipulating it is possible to show that

$$F_v - F = T \sum_{\mathbf{s}} p_0(\mathbf{s}) \log \left(\frac{p_0(\mathbf{s})}{p(\mathbf{s})} \right) \tag{5}$$

where p, p_0 are the respective probability distributions and the sum is over all possible states. This is a nonnegative convex function of the p_0 variables. It is zero only when there is equality between p and p_0.

The Mean-Field Approximation

A particularly simple form of H_0 is a linear function of the spin variables,

$$H_0 = \sum_i c_i s_i. \tag{6}$$

Such an additive H_0 is convenient because it leads to a factorized probability distribution $p = \prod_i p_i$, where each spin has an independent distribution. As every Hamiltonian, in the case of Ising spins, can be written as a multilinear function of the spins, a factorized distribution makes it possible to use $< H(\mathbf{s}) >_0 = H(< \mathbf{s} >_0)$, where

$$v_i \equiv < s_i >_0 = \frac{e^{-c_i/T} - e^{c_i/T}}{e^{-c_i/T} + e^{c_i/T}} = \tanh\left(-\frac{c_i}{T}\right). \tag{7}$$

An optimal $F_v = H(\mathbf{v}) - TS_v$ with respect to v_i is found by

$$0 = \frac{\partial F_v}{\partial v_i} = \frac{\partial H}{\partial v_i} + T \frac{\partial}{\partial v_i} \left(\sum_i \left[\frac{1+v_i}{2} \log\left(\frac{1+v_i}{2}\right) + \frac{1-v_i}{2} \log\left(\frac{1-v_i}{2}\right) \right] \right) \tag{8}$$

leading to the mean-field equations ([10])

$$v_i = \tanh\left(-\frac{c_i}{T}\right) \tag{9}$$

$$c_i = \frac{\partial H(v)}{\partial v_i}. \tag{10}$$

This is the main result of this section as the mean-field equations are the corner stones of the algorithms presented in this thesis, where they are used in an iterative manner. The rest of the section is devoted to other types of spin variables, used for problems where the variables can assume more than two values.

Potts spins

In problems where a variable (spin) can be in more than two states, Potts spin encoding is suitable [13]. Each spin state can be encoded as a principal vector (e.g. $s = (0, 1, 0, 0)$, where the spin is in state two out of four). Any function of Potts spins can be formulated as a multi-linear function of the spins, as in the case of Ising spins. For a linear variational Hamiltonian $H_0 = \sum_{ia} c_{ia} s_{ia}$, the thermal average of a spin component is

$$v_{ia} \equiv < s_{ia} >_0 = \frac{e^{-\frac{c_{ia}}{T}}}{\sum_b e^{\frac{c_{ib}}{T}}}, \tag{11}$$

and the normalization $\sum_a v_{ia} = 1$ is obvious, leading to the interpretation of v_{ia} as the probability of spin i being in state a. The corresponding variational free energy is defined by

$$F_v = H(\mathbf{v}) - TS_v = H(\mathbf{v}) + T \sum_{ia} v_{ia} \log(v_{ia}). \tag{12}$$

An extremal value is now defined by $\partial F_v / \partial v_{ia} = \lambda_i$, where λ_i is a Lagrange multiplier for the constraint $\sum_a v_{ia} = 1$. This yields

$$\frac{\partial F_v}{\partial v_{ia}} = \frac{\partial H}{\partial v_{ia}} + T + T \log(v_{ia}) = \lambda_i, \tag{13}$$

and the resulting mean-field equations

$$v_{ia} = \frac{e^{-c_{ia}/T}}{\sum_b e^{-c_{ib}/T}} \tag{14}$$

$$c_{ia} = \frac{\partial H(v)}{\partial v_{ia}}. \tag{15}$$

Assignment Spins

An assignment is a matching between two sets and can be encoded as an assignment (or permutation) matrix. This is a matrix with (0,1) elements and where each row and each column have exactly one element equal to one. An element of value one then represents a matching between a row and a column representing the two sets. In a mean-field formulation there will be weighted averages of more than one spin, and this is represented by doubly stochastic matrices defined by having nonnegative elements constrained to form row sums and column sums equal to one.

A number of difficulties compared to Ising and Potts spins emerge in the case of assignment spins. First, the most general function of the spins is not multilinear in the spin components and hence it is not evident that the best variational Hamiltonian to be used is a linear one. Also there is no one-to-one mapping from doubly stochastic matrices to a weighted sum of permutation matrices. Consider for example the 3×3 matrix where all elements are equal to $\frac{1}{3}$, which is obviously a doubly stochastic matrix. It can represent the uniform average of the symmetric permutation matrices, $\frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, but equally well the uniform average of the antisymmetric permutation matrices, $\frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$, or the uniform average of all possible permutations.

There is of course the possibility to encode assignments by Potts spins, where each possible assignment represents an element. Then we are back with a more computationally convenient formulation, but this is not very useful for large N as there are $N!$ possible assignments, compared to the N^2 elements used in a permutation matrix encoding.

Assuming a linear Hamiltonian in the spin elements,

$$H = \sum_{ia} c_{ia} s_{ia} \tag{16}$$

the thermal averages can be written as

$$< s_{ia} >= v_{ia} = \frac{M_{ia} P_{ia}}{P}, \tag{17}$$

where $M_{ia} = \exp(-c_{ia}/T)$ and P (P_{ia}) is the permanent (sub-permanent) of the matrix M. The permanent [6] of a matrix M is defined by

$$P = \sum_{\pi}^{N!} \prod_{i} M_{i\pi(i)}, \tag{18}$$

where π is a permutation and $\pi(i)$ is the column matched to row i in permutation π. The sub-permanent P_{ia} is the permanent of the matrix where row i and column a are removed.

This does not lead to simple mean-field equations for a nonlinear Hamiltonian, and the peculiarities associated with assignment spins are discussed in paper 3.

Combinatorial Optimization

Now it is time to introduce the group of problems to which the deterministic annealing algorithm can be applied. One of the main contributions of this section is the description of how these problems can be encoded as spin systems. But first we start with an example to get a feeling for these kinds of problems.

Imagine a pile of (jigsaw) puzzle-pieces, and that the pieces may belong to different puzzles. Your task is to decide which pieces to use and try to make a complete puzzle. If you succeed it is easy to show as there is a puzzle in front of you, but if you fail you cannot really tell whether there is no complete puzzle among the pieces, or if you were just not able to find it.

This is the essence of the hard optimization problems studied in this thesis. A combinatorial optimization (CO) problem [9] is defined by a number of discrete variables and a cost associated with each possible state the variables can be in. The goal is then to find the state with the lowest possible cost.

In some CO problems the cost consists of penalties from constraints on a subset of variables that are not allowed to attain certain values at the same time. In these problems the question is whether there are values for the variables that fulfill all constraints. A positive answer to this question is easily proven by showing the state that fulfills the constraints, but a negative answer is usually much harder to prove. In other CO problems a cost is defined as a function of the variable state and the goal is to find the lowest cost. Without loss of generality the problem can be reformulated as a question of whether there is a state with a cost lower than a certain value. Then again it is easy to prove solvability by showing the state (and the cost connected to it), while non-solvability usually is harder to prove.

Computer scientists have grouped different types of problems into different classes depending on the difficulty of the problems [9]. Problems are more tractable if there exists an algorithm that solves the problem in a time polynomial in the size of the input, N (the number of variables, constraints,...). This in contrast to problems where algorithms use exponential time in the problem size. Problems with polynomial algorithms are in a class called P, while problems with exponential algorithms having the feature of an easy (polynomial time) proof of a solution (as described above) are in a class called NP. As the classes are defined by if there is an algorithm that solves the problem in a polynomial time the question whether $P = NP$ is still open, although most people believe that there are problems in NP that are not in P. Further there is a class of problems within NP that is called NP-complete. Any problem in NP can be transformed in polynomial time to these problems. Hence NP-complete problems are at least as hard as every other NP-problem, and more

important: if any NP-complete problem is solved in an efficient way (for example in polynomial time) all problems in NP can be solved using the algorithm. NP-complete problems have hence attracted much attention by researchers and the problems studied in this thesis are such problems. The practical use of NP-complete problems is huge, and they appear in e.g. scheduling problems, routing problems and VLSI construction.

Constraint Satisfaction Problems

In a constraint satisfaction problem the cost is connected to constraints of subgroups of the variables not allowed to be in certain states simultaneously and the aim is to find out whether there is a state where no constraint is broken. Two NP-complete problems of this type are studied in this thesis. The first problem is the SAT problem, which was the first problem that was proven to be NP-complete. Actually we will discuss a form of SAT written on a special form (conjunctive normal form) called a $KSAT$ problem. There is no limitation in studying only $KSAT$ problems since any SAT instance can be written in this form. The other problem studied is the graph coloring problem, where discrete variables with more than two values are used.

KSAT

A KSAT problem [1] consists of N boolean (two-state) variables $x_i = (\text{True}, \text{False})$ and the constraints are built up from clauses of the type (a_i or a_j or a_k), where the literals a represent a boolean variable or its negation. A clause is broken if all included literals are in the wrong state and fulfilled otherwise. A problem consists of a number, M, of these clauses and all of them must be fulfilled for a solution. The number of variables in each clause can be different and if it is constant we name the problem 2SAT, 3SAT and so forth. An interesting feature is that 2SAT is in P, while if there are three or more variables in each clause, the problem is NP-complete. Two problems that appear quite similar have totally different characteristics. While one of them is quite easy to solve, the other one is notoriously hard to solve.

A suitable cost function can be formulated for each clause m, using Ising spins ($s_i = \pm 1$) for the variables. It is defined as

$$H_m = \prod_{i \in m}^{K_m} \frac{1}{2}(1 - C_{mi} s_i), \tag{19}$$

where K_m is the number of literals in the clause and the matrix C defines

which variables appear in the clause. C_{mi} is 1 if x_i is present in clause m, -1 if its negation is, and 0 otherwise. The factor $\frac{1}{2}(1 - C_{mi}s_i)$ becomes zero if the correct version of a variable is present yielding $H_m = 0$, but if all variables are wrong, H_m becomes one. The total cost is formulated as

$$H(s) = \sum_{m}^{M} \prod_{i \in m}^{K_m} \frac{1}{2}(1 - C_{mi}s_i), \tag{20}$$

and it counts the number of broken clauses. This cost function can be written as

$$H(s) = \text{const} + \sum_{i}^{N} h_i s_i + \sum_{i}^{N} \sum_{j}^{N} J_{ij} s_i s_j + \ldots, \tag{21}$$

and the resemblance of a spin system (eq. 1) is evident.

Graph Coloring

A special version of the graph coloring problem [9] is if you have a map of the world and want to color the countries in such a way that two countries with a common border have different colors. Of course this is a simple task if there is an infinite palette of colors, but if there are only a small number of colors available it might not be so trivial. A definition of the more general problem is that there are N nodes with discrete variables, x_i, that can have C values, where C is the number of colors that can be used. Then nodes are connected by L links and the variables on the connected nodes cannot have the same value. If there are two connected nodes with the same variable value a constraint is broken. The main difference from KSAT is that here the variables are allowed to have more than two values, while the constraints are built up by two variables. A graph coloring problem where two colors may be used is special as it is solvable in polynomial time.

The penalty for a constraint between node i and node j can be defined, using Potts spins $\mathbf{s}$ of size C, as

$$H_{ij} = \sum_{c}^{C} s_{ic} s_{jc}, \tag{22}$$

and the total cost function is then

$$H(\mathbf{s}) = \frac{1}{2} \sum_{ij} J_{ij} \sum_{c}^{C} s_{ic} s_{jc} = \frac{1}{2} \sum_{ij} J_{ij} \mathbf{s}_i \mathbf{s}_j, \tag{23}$$

where J_{ij} is one if nodes i and j are connected and zero otherwise. This cost again counts the number of constraints broken and is nothing but an anti-ferromagnetic Potts spin system.

Finding Hard Instances

As discussed above the problem groupings are defined from the time complexity of exact algorithms. It is always possible to find instances of NP-complete problems that are easily solved and hence the definitions are for worst case scenarios. It is then interesting to investigate problem instances that are particularly hard. Of course we want to use NP-complete problems, and thus this thesis deals with 3SAT and 3-coloring. But also depending on the number of constraints in comparison with the number of variables, differences in difficulty are apparent [4, 7]. A common way to test an algorithm is to create a testbed of random instances from a defined rule of generating problems. For example random instances of graph coloring can be defined by generating L distinct links connecting two random nodes out of the N available. The average difficulty of the problems can be related to the parameter $\gamma = \frac{2L}{N}$. The possibility of a graph being 3-colorable decreases with the number of links added, but the decision problem, where we ask whether a graph is 3-colorable, is most difficult around a value of γ where about half of the generated graphs are colorable. If the number of links are few it is easy to show that a graph is 3-colorable, and if the number of links are large it is easy to find a contradiction among the constraints.

This kind of behavior is apparent for a number of combinatorial optimization problems, and in this thesis the behavior of the algorithms are explored in the region where the hardest problems appear.

Nonlinear Assignment

In an assignment problem two sets of entities are to be mapped onto each other in an one-to-one fashion. An example is if there are N taxi cars that are given N customers and the distance between each car and each customer is known. The goal is to assign a customer to each car in such a way that the total distance traveled by the cars to the customers is as small as possible. This is a linear assignment problem as the cost (total distance) is a sum of the distances each individual car has to travel. A linear assignment problem is solvable in polynomial time. A classic example of a nonlinear assignment problem is the traveling salesman problem (TSP) [9], where a person is to visit

a number of cities and come back to the starting point in such a way that each city is visited exactly once and the total distance is to be minimized. Here each city can be mapped onto the position in the route. The cost is the total distance of the route and the cost of a city being in a certain place in the route depends on which cities that come before and after in the route. This is a quadratic assignment problem, which is NP-complete. Using an assignment spin $\{s_{ia}\}$, where i is the position in the route and a is the city, a cost function can be defined as

$$H(\mathbf{s}) = \sum_{iab}^{N} D_{ab} s_{ia} s_{i+1b}, \tag{24}$$

where D is the distance matrix between cities. If city a is chosen at position i in the route and city b is chosen at position $i+1$, D_{ab} is added to the cost, and the total cost will be the sum of the distances in the route. Position $N+1$ in the route is the same position 1 (i.e. periodic boundaries).

Deterministic Annealing

In the previous section cost functions were defined as Hamiltonians of spin systems, and the corresponding optimization problem is solved by finding the global minimum of these functions, but as these spin systems all have a very rugged energy landscape this is not an easy task. If an algorithm just tries to minimize the energy in each step (by e.g. gradient descent in each spin variable), it will most probably get stuck in a local minimum. Some kind of approach to avoid local minima as much as possible must be used.

An algorithm that is often used is simulated annealing [5], where an artificial temperature is introduced to simulate that the system is in a heat bath. In this algorithm the spin system evolves by simple random spin flips which are accepted/rejected in such a way that the Boltzmann distribution is generated. The algorithm starts at a high temperature, where almost all states are equally probable and hence most flips are accepted and the number of visited states is large. Then the temperature is slowly lowered (annealed) and the algorithm favors low energy states more and more. If the annealing is performed slowly enough, the algorithm will find the global minimum with a probability one. However this is too slow to be practically useful, and simulated annealing is most often used as a heuristic algorithm, i.e. an algorithm that finds a low energy state but does not prove that it is the global minimum.

The deterministic annealing (mean-field annealing) [11] is somewhat related to simulated annealing, but it differs in that instead of a stochastic update with rejection, it has a deterministic update scheme for the spin averages v, based on

the mean-field equations. It starts at a high temperature, where the mean-field equations have a fixed point for all spin values equally probable, then the spin variables, v, are updated while annealing the temperature, and energetically more favorable states become more probable. At low temperatures the spins saturate, $v \rightarrow s$, and freeze in a locally stable state which is proposed as a solution. At high temperature the energy landscape can be seen to be more smooth and local minima are avoided. The feature of starting in "all" states at a high temperature, and then feeling its way down to more energetically favorable states is appealing in contrast to most optimization algorithms that starts in one state and then searches the neighborhood of this state. The mean-field annealing algorithm is described in figure 1. For problems encoded by assignment spins, some complications appear, as discussed in paper 3.

- Initiate the mean-field spins v to a value close to uniform (i.e. 0 for Ising spins and $1/K$ for K-state Potts spins) with a small random noise, and T to a high value.
- Repeat the following (a sweep), until the mean-field variables have *saturated*, i.e. v has become close to s:
  - Update each spin according to the mean-field equations.
  - Decrease T slightly (typically by a few percent).
- Extract the resulting solution candidate, using $v_i \rightarrow s_i$, where s_i is the spin value closest to v_i.

Figure 1: A mean-field annealing algorithm.

Information-Based Hamiltonian

For constraint satisfaction problems, the Hamiltonian is built up only from penalties from broken constraints. As a single broken constraint is enough for the problem to be unsolved it might be useful to use a nonlinear penalty so that constraints that are nearly broken in the mean-field spin formulation receives more attention. This is the inspiration for formulating a nonlinear cost function based on ideas from information theory.

The probabilities of different spin states can be formulated by the mean-field variables. In the case of Potts spins the probability, $p(s_a)$ of a spin $\mathbf{s}$ being in the state a is given by the corresponding component v_a. For Ising spins the

probabilities can be encoded as $p_{s=\pm 1} = \frac{1}{2}(1 \pm v)$. The average information resource residing in a spin is given by

$$S = -\sum_i p_i \log(p_i), \tag{25}$$

where p_i is the probability of the spin being in state i. In the case of an Ising spin, a completely random spin ($p_{s=\pm 1} = \frac{1}{2}$) yields $S = \log(2)$, representing an unused resource of one bit of information. A definite value of the spin ($s = \pm 1$) yields $S = 0$, and no information is available. The average information available in all spins is then equivalent to the entropy for a spin system.

As the probabilities of different spins are independent in the mean-field formulation, the expected amount of information needed to satisfy a constraint can be formulated as a simple function of the mean-field spins

$$I_m = -\log(P^{(m)}_{sat}(\mathbf{v})) = -\log(1 - P^{(m)}_{unsat}(\mathbf{v})). \tag{26}$$

The probability, $P^{(m)}_{unsat}(\mathbf{v})$, of a constraint m being broken is given by $H_m(\mathbf{v})$ as defined in equations (19,22) for KSAT and graph coloring, and hence

$$I_m = -\log(1 - H_m). \tag{27}$$

In analogy with the free energy for a system with a cost function defined by a Hamiltonian a new variational information-based "free energy" can be formulated as

$$F = I(\mathbf{v}) - TS(\mathbf{v}), \tag{28}$$

and the deterministic annealing algorithm can be adjusted for this F. This free energy can be seen as a balance between information residing in the spins and information "needed" to satisfy the constraints, and the artificial temperature, T, is a factor determining the weight between the two terms. At a high temperature the spins are holding as much information as possible, while at low temperatures the clauses become more and more important to satisfy. As I is built up from logarithms, the value tends to infinity when an argument tends to zero (the constraint is broken). Hence some care has to be taken to yield smooth dynamics of the spin during update. Instead of using the derivatives of I with respect to a spin v, a difference update is used [8], which for Ising spins yields

$$\frac{\partial H}{\partial v_i} \to \frac{\Delta I}{\Delta v_i} = \frac{1}{2}\left(I_{|v_i=1} - I_{|v_i=-1}\right), \tag{29}$$

and for Potts spins,

$$\frac{\partial H}{\partial v_{ia}} \to \frac{\Delta I}{\Delta v_{ia}} = \left(I_{|v_{ia}=1}\right) + \text{const}, \tag{30}$$

where the constant is a-independent and hence does not affect the update of the spin. With these adjustments, the mean-field annealing algorithm can be used as described above in figure 1.

Phylogeny Reconstruction

We now leave the world of combinatorial optimization and introduce another problem where the variational approach is practicable.

It is a common assumption that existing species of today all descend from a single ancestor, and that differences have emerged via evolution. An important problem is then to reconstruct this phylogenetic history.

The study of molecular sequences such as DNA has become the main tool used to reconstruct evolutionary history [12]. The key feature of the algorithms is to use the information found in the differences in homologous sequences from different species. This information is then used when trying to infer the phylogenetic relationships between the species. When an evolutionary tree is defined, one assumes that branching events occur where populations of a species are split into two or more groups that no longer interacts. In between branch points the different species evolve independently and in the end we have the differentiated species now present. The goal in phylogeny reconstruction is to find the topology (branching pattern) and the geometry (evolutionary distances between the branch points) for the most probable tree. Several algorithms have been developed to infer evolutionary trees from DNA sequences [12]. In this thesis a variational approximation to one of the most theoretically appealing methods, maximum likelihood, is presented.

Maximum Likelihood

A common way of describing the world is to introduce a model that predicts the behavior of nature, and usually experimental data is used to tune the model. The aim is to maximize the probability, $P(M|S)$, of a model, M, given the data, S, with respect to all possible models. This probability is often impossible to find out, but using Bayes' theorem it can be seen to be related to the likelihood, $P(S|M)$, which is computable for a given model. Bayes' theorem is defined by

$$P(M|S) = \frac{P(M)P(S|M)}{P(S)}, \tag{31}$$

where $P(S)$ can be seen as a model independent normalization factor and does not interfer with the model optimization. $P(M)$ is the prior, and it is often

suppressed when optimizing the model. This is justified if the data set is large, as the effect of the prior then is small compared to the likelihood, or if all models are considered equally probable, yielding a constant prior. The likelihood is then used as a measure of performance for the model. It is maximized by adjusting the parameters of the model to find the best possible fit to the data.

In this thesis maximum likelihood will be used to fit a model for phylogenetic trees to DNA-sequence data [2]. An evolutionary model consists of a model for sequences mutation and for the branching pattern. The model for branching is often suppressed by optimizing with respect to the mutation model for a fixed topology, and then comparing the result for all possible topologies.

Mutation Models

A DNA sequence is built up of the four nucleotides adenine (A), cytosine (C), guanine (G) and thymine (T). In the models studied here, each site in the sequence is assumed to evolve independently from the others, but with a common stochastic model. The model used here defines probabilities of substitutions between nucleotides, while insertions-deletions are not considered. It is assumed that the substitutions follow a stationary stochastic process and hence the frequencies of different nucleotides are constant. It is also assumed that the model is time reversible, and then no actual time direction is present leading to an unrooted tree with evolutionary "distances" between the nodes (see figure 2).

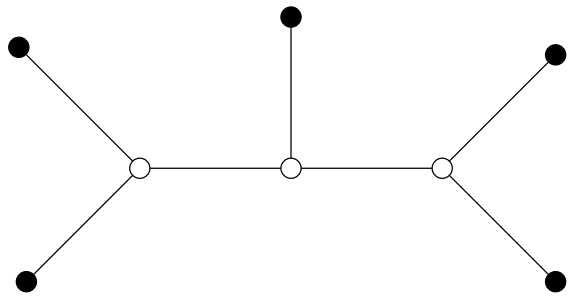

Figure 2: An unrooted phylogenetic tree with 5 leaves (species) and 3 ancestor nodes.

Among such models, the Jukes-Cantor (JC) model is the simplest. It assumes that each nucleotide is equally probable and substitutions between nucleotides are symmetric. It defines the substitution probability between nucleotides i

and j as

$$T_{ij} = \frac{1 - e^{-t}}{K} + e^{-t}\delta_{ij} = \frac{1 - a}{K} + a\delta_{ij}, \tag{32}$$

where K is the number of different nucleotides (four for DNA), t is the evolutionary distance, and δ_{ij} equals 1 if $i = j$ and 0 otherwise. There is a single parameter $a = e^{-t}$ for a link between two nodes. If $t \to 0$, T_{ij} becomes the identity matrix and no substitutions appear, while when $t \to \infty$ the substitution to any nucleotide (including itself) is equally probable.

Calculating the Likelihood

Assume that a tree topology and aligned homologous sequences (figure 3) for N species are given. At each site in the sequences there is an observed combination of nucleotides $S = (s_1, s_2, \ldots, s_N)$ for the species, and an unknown combination of nucleotides $I = (i_1, ...i_{N-2})$ for the internal (ancestor) nodes. The single-site probability for the combination of nucleotides (SI) can be calculated for a given model. It can be done by starting from a proper nucleotide probability at one node ($1/K$ for JC), defined as a root, and then propagate along the links using the substitution probabilities, T, until the observed nucleotides at the leaves are reached. For JC the probability of the nucleotide combination SI becomes

$$P_{SI} = \frac{1}{K} \prod_{[kl]} \left(\frac{(1 - a_{[kl]})}{K} + a_{[kl]}\delta_{ij} \right), \tag{33}$$

where $[kl]$ denotes the link between node k and l. The total single-site likelihood, P_S, is given by summing over the unknown internal nucleotides

$$P_S = \sum_I P_{SI}. \tag{34}$$

The total likelihood is then the product of the P_S for the independent sites of the sequences. Two sites where exactly the same nucleotides are observed at the leaves will contribute equally to the likelihood and hence a grouping over distinct S can be used. An alignment of N sequences of length M can be seen as M independent experiments that results in a distribution of the outcome as

$$Q_S = \frac{M_S}{M} \tag{35}$$

where M_S is the number of times S is present in the aligned sequences. The probability of the observed multiplicities M_S is given by $\prod_S P_S^{M_S}$ multiplied

by the combinatorial factor $(M!/\prod_S M_S!)$ that for long sequences can be approximated by $\prod_S Q_S^{-M_S}$, yielding a normalized likelihood of

$$L = \prod_S \left(\frac{P_S}{Q_S}\right)^{MQ_S}. \tag{36}$$

Note that the normalization is model independent and that for this definition, L equals one if the model fits perfectly to data ($P = Q$), and a value less than one otherwise. It is more convenient to work with the logarithm of the likelihood and hence the free energy per site, F, is defined as

$$F = \frac{-\log(L)}{M} = \sum_S Q_S \log\left(\frac{Q_S}{P_S}\right). \tag{37}$$

The maximization of L corresponds to a minimization of F, and F is a convex nonnegative function which equals zero only if $P = Q$.

H ...CCCATGGTGGGAGGAGAGACTGAG...
C ...CCCATGGTGGCAGGAGAGACTGAG...
G ...CCCATGGTAGGAGGAGAGACTGAG...
O ...CCTATGGTGGGAGGAGAGACGGAG...

Figure 3: Part of aligned sequences from human (H), chimpanzee (C), gorilla (G) and orangutan (O). At each site a combination $S = (s_H, s_C, s_G, s_O)$ is observed.

An optimal F with respect to a single link parameter a is found at $\partial F/\partial a = 0$, which yields an expression of a solvable by using some iterative scheme, e.g. the Newton-Raphson method.

The link parameters are updated one at a time until convergence, and an (locally) optimal F is found along with the optimal geometry (link parameters). Then free energies obtained from different topologies are compared and the one with lowest F is chosen as the most likely topology.

A Variational Approach to Phylogeny Reconstruction

The optimization of the link parameters in the maximum likelihood approach requires an iterative procedure. In this section a variational nucleotide distribution for the unknown ancestors is introduced and a scheme for optimizing

the parameters of the model and the variational distribution is described. For JC, the parameters are all updated using explicit equations.

If the sequences of the ancestors were known we would use

$$\hat{F} = \sum_{IS} Q_{IS} \log\left(\frac{Q_{IS}}{P_{IS}}\right) \tag{38}$$

to optimize the model parameters. Using $Q_{IS} = Q_S Q_{I|S}$ we can rewrite this as

$$\hat{F} = F + \sum_{S} Q_S \sum_{I} Q_{I|S} \log\left(\frac{Q_{I|S}}{P_{I|S}}\right) = F + \sum_{S} Q_S G_S, \tag{39}$$

where G_S is defined by

$$G_S = \sum_{I} Q_{I|S} \log\left(\frac{Q_{I|S}}{P_{I|S}}\right), \tag{40}$$

and it is of the form of equation (5) describing the relation between the variational and the Boltzmann distributions for a system in a heat bath. $G(S)$ can be interpreted as the variational free energy for the approximation of $P_{I|S}$ by $Q_{I|S}$. A perfect fit yields $G_S = 0$ and then $\hat{F} = F$.

A possibility is to assume that $Q_{I|S}$ is factorized over the $N-2$ internal nodes

$$Q_{I|S} = \prod_{k}^{N-2} v_{ki_k|S}, \tag{41}$$

where k is the label for internal nodes and i_k is the nucleotide at the node. This is equivalent to a mean-field approximation that is optimized by the mean-field equations for each $\mathbf{v}$. Using this formulation, $\hat{F}$ evaluates to

$$\hat{F} = \text{const} + \sum_{S} Q_S \sum_{k} \sum_{i} v_{ki|S} \log v_{ki|S} - \sum_{S} Q_S \sum_{[kl]} \sum_{ij} v_{ki|S} v_{lj|S} \log T_{ij}^{[kl]}, \tag{42}$$

and it is to be optimized with respect to the link parameters (included in $T^{[kl]}$) for the model and with respect to each $v_{ki_k|S}$. For each S, the variational parameters, $\mathbf{v}_{k|S}$, are used to optimize the relevant part of $\hat{F}$,

$$\sum_{k} \sum_{i} v_{ki|S} \log(v_{ki}|S) - \sum_{[kl]} v_{ki|S} v_{lj|S} \log\left(T_{ij}^{[kl]}\right), \tag{43}$$

which is done using mean-field equations. Optimality with respect to a single link parameter $a_{[kl]}$is given by $\partial\hat{F}/\partial a_{[kl]} = 0$ and the relevant part of $\hat{F}$ is

$$-\sum_{S} Q_S \sum_{ij} v_{ki|S} v_{lj|S} \log T_{ij}^{[kl]} = -\sum_{ij} < v_{ki|S} v_{lj|S} >_Q \log T_{ij}^{[kl]}, \tag{44}$$

where $<>_Q$ is a weighted average in the Q_S distribution.

For the Jukes-Cantor model, this results in explicit updating equations for the variational parameters as well as for the model parameters. The link parameters a are updated according to

$$a_{[kl]} = \frac{K < \mathbf{v}_k \mathbf{v}_l >_Q -1}{K-1}, \tag{45}$$

for a link between nodes l and k, and the variational parameters for internal nodes are updated using the mean-field equation

$$v_{ki} \propto \prod_{l \in NN} \left(\frac{1 + (K-1)a_{[kl]}}{1 - a_{[kl]}} \right)^{v_{jk}}, \tag{46}$$

where NN indicates the neighboring nodes. The equations (45,46) are the core in a variational maximum likelihood algorithm to infer the geometry (link lengths) for a given topology and assuming a JC model. It updates link parameters and variational parameters iteratively, until convergance, and an optimal free energy F is found (along with an optimal geometry). Again free energies of different topologies are compared to find the most likely topology.

The Papers

In this section a short introduction is given to the papers presented in the second part of this thesis. The first three papers deal with the deterministic annealing algorithm. In the first two, constraint satisfaction problems are approached, and in the third, nonlinear assignment. In paper IV, the variational approximation to maximum likelihood is introduced.

Paper I

In this paper the information-based deterministic annealing algorithm for Ising spins is introduced. Differences compared to the conventional mean-field approach are discussed. It is numerically explored on a testbed of K-SAT problems and it is shown to outperform the conventional mean-field annealing algorithm. The performance of the information-based annealing algorithm is seen to be comparable to the KSAT-dedicated heuristic Gsat+Walk.

Paper II

The information-based deterministic annealing algorithm is extended to be able to deal with constraint satisfaction problems where the variables are non-Boolean. The Potts spin encoding is employed and the numerical experiments are on a testbed of graph coloring problems. The performance is compared to those of a biased simulated annealing algorithm, a problem-dedicated simple heuristic, DSATUR, and the conventional mean-field annealing algorithm. The information-based deterministic annealing algorithm and the biased simulated annealing algorithm are shown to perform best.

Paper III

This paper deals with nonlinear assignment problems, and a proper variational approach, where the problem cost function is approximated by a cost linear in the assignment matrices, is investigated. It leads to an iterative scheme to minimize the variational free energy, which can be used in a deterministic annealing algorithm. It uses a time exponential in the size of the assignments, and its practical usefulness is then limited (at least for single-assignment problems). Also improvements to existing Potts-based mean-field-inspired heuristics are proposed, and the traveling salesman problem is used to numerically confirm this.

Paper IV

In this paper the variational approximation scheme for maximum likelihood is introduced. It is applied to phylogeny reconstruction from DNA-sequences. It is tested on artificial problems of different sequence lengths (including infinite length). The performance is comparable to that of the original maximum likelihood approach when the sequences are relatively similar. For dissimilar sequences the variational method deteriorates somewhat. The method is also applied to real DNA-sequences from primates and is shown to yield a result consistent with those obtained by standard algorithms.

Acknowledgments

My appreciation of the completely new universe of algorithms, models and analytical methods that has been introduced to me by my supervisor Bo Söderberg is beyond words. The always enthusiastic way of explaining problems and presenting methods has been invaluable to me.

I also would like to thank the people at the department for contribution to the atmosphere at work. A special thought goes to the golf team, Stefan and Thomas, especially for the fruitful collaborations during the midnight hours.

Thanks also to my family and friends who have helped me living a pleasant life beyond the outer door of the department, and to Fredrik Sjunnesson and Stephen Burby, for proofreading this introduction.

Finally I would like to thank you Annika, for loving and supporting me the way you do, also at times when I hardly leave the office and my mind never does.

References

[1] D. Du, J. Gu, and P. M. Pardalos, editors. *Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[2] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[3] R. Feynman. *Statistical Mechanics, Frontiers in Physics*. W. A. Benjamin, Inc., Reading, MA, 1972.

[4] T. Hogg, B. A. Hubermann, and C. P. Williams. Special volume on frontiers in problem solving: Phase transitions and complexity. *Artificial Intelligence*, 81(1,2), 1996.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] H. Minc. *Permanents*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1978.

[7] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137, 1999.

[8] M. Ohlsson, C. Peterson, and B. Söderberg. Neural networks for optimization problems with inequality constraints - the knapsack problem. *Neural Computation*, 5(2):331–339, 1993.

[9] C. H. Papadimitriou. *Combinatorial Optimization*. Dover Publications, Inc., Mineola, New York, 1998.

[10] G. Parisi. *Statistical Field Theory*. Addison-Wesley Publishing Company, Reading, MA, 1988.

[11] C. Peterson and B. Söderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1:3–22, 1989.

[12] D. L. Swafford and G. J Olsen. Phylogeny reconstruction. In C. Moritz D. M. Hillis and B. K. Mable, editors, *Molecular Systematics*. Sinauer Associates, Sunderland, 1996.

[13] F. Y. Wu. The Potts model. *Review of Modern Physics*, 54(1):235–268, 1982.

An Information-Based Neural Approach to Constraint Satisfaction

Paper I

An Information-Based Neural Approach to Constraint Satisfaction

Henrik Jönsson and Bo Söderberg

Complex Systems Division, Department of Theoretical Physics
Lund University, Sölvegatan 14A, S-223 62 Lund, Sweden
http://www.thep.lu.se/complex/

Neural Computation **13**, 1827 – 1838 (2001)

A novel artificial neural network approach to constraint satisfaction problems is presented. Based on information-theoretical considerations, it differs from a conventional mean-field approach in the form of the resulting free energy. The method, implemented as an annealing algorithm, is numerically explored on a testbed of K-SAT problems. The performance shows a dramatic improvement to that of a conventional mean-field approach, and is comparable to that of a state-of-the-art dedicated heuristic (Gsat+Walk). The real strength of the method, however, lies in its generality – with minor modifications it is applicable to arbitrary types of discrete constraint satisfaction problems.

1.1 Introduction

In the context of difficult optimization problems, artificial neural networks (*ANN*) based on the mean-field approximation provides a powerful and versatile alternative to problem-specific heuristic methods, and have been successfully applied to a number of different problem types [5, 9].

In this paper, an alternative ANN approach to combinatorial constraint satisfaction problems (*CSP*) is presented. It is derived from a very general information-theoretical idea, which leads to a modified cost function as compared to the conventional mean-field based neural approach.

A particular class of binary CSP that has attracted recent attention is *K-SAT* [8, 2]; many combinatorial optimization problems can be cast in K-SAT form. We will demonstrate in detail how to apply the information-based ANN approach, to be referred to as *INN*, to K-SAT as a modified mean-field annealing algorithm.

The method is evaluated by means of extensive numerical explorations on suitable testbeds of random K-SAT instances. The resulting performance shows a substantial improvement as compared to that of the conventional ANN approach, and is comparable to that of a good dedicated heuristic – *Gsat+Walk* [10, 3].

The real strength of the INN approach lies in its generality – the basic idea can easily be applied to arbitrary types of constraint satisfaction problems, not necessarily binary.

1.2 K-SAT

A CSP amounts to determining whether a given set of simple constraints over a set of discrete variables can be simultaneously fulfilled.

Most heuristic approaches to a CSP attempt to find a *solution*, i.e. an assignment of values to the variables consistent with the constraints, and are hence *incomplete* in the sense that they cannot prove unsatisfiability. If the heuristic succeeds in finding a solution, satisfiability is proven; a failure, however, does not imply unsatisfiability.

A commonly studied class of binary CSP is *K-SAT*. A K-SAT instance is defined as follows: For a set of N Boolean variables x_i, determine whether an assignment can be found such that a given Boolean function U evaluates to

True, where U has the form

$$U = (a_{11}\texttt{OR}a_{12}\texttt{OR}\ldots a_{1K})\,\texttt{AND}\,(a_{21}\texttt{OR}\ldots a_{2K})\,\texttt{AND}\ldots\texttt{AND}\,(a_{M1}\texttt{OR}\ldots a_{MK})\ , \quad (1.1)$$

i.e. U is the Boolean disjunction of M *clauses*, indexed by $m = 1 \ldots M$, each defined as the Boolean conjunction of K simple statements (literals) $a_{mk}, k = 1 \ldots K$. Each literal represents one of the elementary Boolean variables x_i or its negation $\neg x_i$.

For $K = 2$ we have a 2-SAT problem; for $K = 3$ a 3-SAT problem, etc. If the clauses are not restricted to have equal length the problem is referred to as a *satisfiability* problem (*SAT*). There is a fundamental difference between K-SAT problems for different values of K. While a 2-SAT instance can be exactly solved in a time polynomial in N, K-SAT with $K \geq 3$ is NP-complete. Every K-SAT instance with $K > 3$ can be transformed in polynomial time into a 3-SAT instance [8]. In this paper we will focus on 3-SAT.

1.3 Conventional ANN Approach

1.3.1 ANN Approach to CSP in General

In order to apply the conventional mean-field based ANN approach as a heuristic to a Boolean CSP problem, the latter is encoded in terms of a non-negative cost function $H(\mathbf{s})$ in terms of a set of N binary (± 1) spin variables, $\mathbf{s} = \{s_i,\ i = 1, \ldots, N\}$, such that a solution corresponds to a combination of spin values that makes the cost function vanish.

The cost function can be extended to continuous arguments in a unique way, by demanding it to be a *multi-linear* polynomial in the spins (i.e. containing no squared spins). Assuming a multi-linear cost function $H(\mathbf{s})$, one considers mean-field variables (or *neurons*) $v_i \in [-1, 1]$, approximating the thermal spin averages $\langle s_i \rangle$ in a Boltzmann distribution $P(\mathbf{s}) \propto exp(-H(\mathbf{s})/T)$. They are defined by the mean-field equations

$$v_i = \tanh(u_i/T) \quad (1.2)$$

$$u_i = -\frac{\partial H(\mathbf{v})}{\partial v_i}\ , \quad (1.3)$$

where u_i is referred to as the *local field* for spin i. Here, T is an artificial temperature and $\mathbf{v}$ denotes the collection of mean-field variables.

The equations (1.2,1.3) can be seen as conditions for a local minimum of the

mean-field *free energy* $F(\mathbf{v})$,

$$F(\mathbf{v}) = H(\mathbf{v}) - TS(\mathbf{v}) \ , \tag{1.4}$$

where $S(\mathbf{v})$ is the spin entropy,

$$S(\mathbf{v}) = -\sum_i \frac{1+v_i}{2} \log\left(\frac{1+v_i}{2}\right) - \frac{1-v_i}{2} \log\left(\frac{1-v_i}{2}\right) \ . \tag{1.5}$$

The conventional ANN algorithm consists in solving the mean-field equations (1.2, 1.3) iteratively, combined with annealing in the temperature. A typical algorithm is described in figure 1.1.

- Initiate the mean-field spins v_i to random values close to zero, and T to a high value.
- Repeat the following (a sweep), until the mean-field variables have *saturated* (i.e. become close to ± 1):
  - For each spin, calculate its local field from (1.3), and update the spin according to (1.2).
  - Decrease T slightly (typically by a few percent).
- Extract the resulting solution candidate, using $s_i = \mathrm{sign}(v_i)$.

Figure 1.1: A mean-field annealing ANN algorithm.

1.3.2 Application to K-SAT

When applying the ANN approach to K-SAT the Boolean variables are encoded using ± 1-valued spin variables s_i, $i = 1 \ldots N$, with $s_i = +1$ representing True, and $s_i = -1$ False. In terms of the spins, a suitable multi-linear cost function $H(\mathbf{s})$ is given by the following expression,

$$H(\mathbf{s}) = \sum_{m=1}^{M} \prod_{i \in \mathcal{M}_m} \frac{1}{2} \left(1 - C_{mi} s_i\right) \ , \tag{1.6}$$

where $\mathcal{M}_m$ denotes the set of spins involved in the mth clause. $H(\mathbf{s})$ evaluates to the number of broken clauses, and vanishes iff $\mathbf{s}$ represents a solution. The $M \times N$ matrix C defines the K-SAT instance: An element C_{mi} equals $+1$

(or -1) if the mth clause contains the ith Boolean variable as is (or negated); otherwise $C_{mi} = 0$.

The cost function (1.6) defines a problem-specific set of mean-field equations, (1.2,1.3), in terms of mean-field variables $v_i \in [-1, 1]$. In the mean-field annealing approach (figure 1.1), the temperature T is initiated at a high value, and then slowly decreased (annealing), while a solution to (1.2,1.3) is tracked iteratively. At high temperatures there will be a stable fixed point with all neurons close to zero, while at a low temperature they will approach ± 1 (the neurons have *saturated*) and an assignment can be extracted.

For the K-SAT cost function (1.6) the local field u_i in (1.3) is given by

$$u_i = \sum_m \frac{1}{2} C_{mi} \prod_{\substack{j \in \mathcal{M}_m \\ j \neq i}} \frac{1}{2} \left(1 - C_{mj} v_j\right) \ , \tag{1.7}$$

which, due to the multi-linearity of H does not depend on v_i; this lack of self-coupling is beneficial for the stability of the dynamics.

1.4 Information-Based ANN Approach: INN

1.4.1 The Basic Idea

For problems of the CSP type, we suggest an information-based neural network approach, based on the idea of balance of information, considering the variables as *sources* of information, and the constraints as *consumers* thereof.

This suggests constructing an *objective function* (or free energy) F of the general form

$$F = \text{const.} \times (\text{information demand}) - \text{const.} \times (\text{available information}) \ , \tag{1.8}$$

that is to be minimized. The meaning of the two terms can be made precise in a mean-field-like setting, where a factorized artificial Boltzmann distibution is assumed, with each Boolean variable having an independent probability to be assigned the value True. We will give a detailed derivation below for K-SAT. Other problem types can be treated in an analogous way. We will refer to this type of approach as *INN*.

1.4.2 INN Approach to K-SAT

Here we describe in detail how to apply the general ideas above to the specific case of K-SAT.

The average information resource residing in a spin is given by its entropy,

$$S(s_i) = -P_{s_i=1} \log P_{s_i=1} - P_{s_i=-1} \log P_{s_i=-1} \ , \tag{1.9}$$

where P are probabilities. If the spin is completely random, $P_{s_i=1} = P_{s_i=-1} = \frac{1}{2}$ and $S(s_i) = log(2)$, representing an unused resource of one bit of information. If the spin is set to a definite value ($s_i = \pm 1$), no more information is available and $S(s_i) = 0$.

For a clause the interesting property is the expected amount of information needed to satisfy it. For the mth clause, this can be estimated as

$$I_m = -\log P_m^{\text{sat}} = -\log\left(1 - P_m^{\text{unsat}}\right) \ , \tag{1.10}$$

in terms of the probability P_m^{sat} for the clause to be satisfied in a given probability distribution for the spins.

Of the 2^K distinct states available to the K spins appearing in the clause, only one corresponds to the clause being unsatisfied. Then, for a totally undetermined clause (all K spins having random values), we have $P_m^{\text{unsat}} = 2^{-K}$, yielding $I_m = -\log\left(1 - 2^{-K}\right)$. For a definitely satisfied clause, on the other hand, we must have $P_m^{\text{unsat}} = 0$, giving $I_m = 0$. Finally, a broken clause corresponds to $P_m^{\text{unsat}} = 1$, leading to $I_m \to \infty$.

Assuming a mean-field-like probability distribution, with each spin obeying independent probabilities

$$P_{s_i=\pm 1} = \frac{1 \pm v_i}{2} \ , \tag{1.11}$$

in terms of mean-field variables $v_i = \langle s_i \rangle \in [-1, 1]$, the probabilities used above for the clauses become

$$P_m^{\text{unsat}} = \prod_{i \in \mathcal{M}_m} \frac{1}{2} \left(1 - C_{mi} v_i\right) \ . \tag{1.12}$$

The unused spin information is given by the entropy S of the spins (eq. (1.5)) and the information I needed by the clauses is

$$I(\mathbf{v}) = \sum_{m=1}^{M} -\log\left(1 - \prod_{i \in \mathcal{M}_m} \frac{1}{2} \left(1 - C_{mi} v_i\right)\right) \ . \tag{1.13}$$

We now have the necessary prerequisites to define an information-based free energy, which we choose as $F(\mathbf{v}) = I(\mathbf{v}) - TS(\mathbf{v})$ (in analogy with ANN), which is to be minimized. Demanding that F have a local minimum with respect to the mean-field variables yields equations similar to the mean-field equations (1.2,1.3), but with $H(\mathbf{v})$ replaced by $I(\mathbf{v})$:

$$u_i = -\frac{\partial I}{\partial v_i} \ . \tag{1.14}$$

Note that for discrete arguments, $v_i = \pm 1$, the infomation demand I will be infinite for any non-solving assignment.

1.4.3 Algorithmic Details

Based on the analysis above, we propose an information-based annealing algorithm similar to mean-field annealing, but with the multi-linear cost function H (1.6) replaced by the clause information I (1.13).

Note that the contribution I_m to I from a single clause m is a simple function of the corresponding contribution H_m to H,

$$I_m = -\log\left(1 - H_m\right) \ . \tag{1.15}$$

As a result, the effective cost function I is not multilinear, and measures have to be taken to ensure stability of the dynamics. The resulting self-couplings can be avoided by instead of the derivative in (1.14) using the difference,

$$u_i = -\frac{1}{2}\left(I|_{v_i=1} - I|_{v_i=-1} \right) \ , \tag{1.16}$$

which coincides with the derivative for a multilinear I [7].

The resulting INN annealing algorithm is summarized in figure 1.2. At high temperatures, information is expensive, and the neurons stay fuzzy, $v_i \approx 0$. As T is decreased, information becomes cheaper and the more useful neurons begin to saturate. As $T \to 0$, all neurons are eventually forced to saturate, yielding a definite spin state, $v_i \approx s_i = \pm 1$.

1. Choose a suitable high initial temperature T, such that the equilibrium neurons are close to zero.
2. Do a sweep: Update all neurons according to (1.2,1.16).
3. Lower the temperature T by a fixed factor μ.
4. If the stop-criteria are not met, repeat from 2.
5. Extract a solution by means of $s_i = \text{sign}(v_i)$.

A typical μ value is 0.95 - 0.99, and suitable stop-criteria are that all neurons are either saturated ($|v_i| > 0.99$) or redundant ($|v_i| < 0.01$).

Figure 1.2: The INN annealing algorithm for K-SAT.

1.5 Numerical Explorations

1.5.1 Testbeds

For performance investigations, we have considered two distinct testbeds. One consists of uniform random K-SAT problems with N and $\alpha = M/N$ fixed ([1]). For every problem instance, each of the M clauses is independently generated by chosing at random a set of K distinct variables (among the N available). Each selected variable is negated with probability $\frac{1}{2}$.

For this ensemble of problems, the fraction unsatifiable problems increases with the parameter α. In the thermodynamic limit ($N \to \infty$) there is a sharp satisfiability transition at a K-dependent critical α-value $\alpha_c^{(K)}$ [4, 6]. For problems where $\alpha < \alpha_c^{(K)}$ almost all generated problems are satisfiable and for $\alpha > \alpha_c^{(K)}$ almost all are unsatisfiable. For 3-SAT, $\alpha_c \approx 4.25$ [1, 6].

We have used a set of N-values between 100 and 2000, and for each N a set of α-values between 3.7 and 4.3. For each N and α, 200 problem instances are generated.

In addition, testbeds consisting purely of satisfiable instances are useful to gauge the efficiency of a heuristic. Such a testbed can be generated by filtering out unsatisfiable instances (using a *complete* (exact) algorithm) from the uniform random distribution described above.

For a second testbed, we have collected a set of instances of this type from

SATLIB[1], consisting in satisfiable random problems for different N between 20 and 250, with α fixed close to α_c. For natural reasons, this testbed does not include very large N.

1.5.2 Comparison Algorithms

To gauge the performance of the INN algorithm, we have in addition to the conventional ANN algorithm also applied a state-of-the-art dedicated heuristic to our testbeds. A wealth of algorithms has been tested on SAT problems. For a survey, see e.g. [3]. A local search method proven to be competitive is the *gsat+walk* algorithm which we will use as a second reference algorithm.

Gsat+walk starts with a random assignment and then uses two types of local moves to proceed. A local move consists in flipping the state of a single variable between True/False. The first type of move is greedy; the flip that increases the number of satisfied clauses the most is chosen. The second type of move is a restricted random walk move. A clause among those that are unsatisfied is chosen at random, and then a randomly chosen variable in this clause is flipped.

1.5.3 Implementations details

In order to have a fair comparison of performances, we have chosen the parameter values such, that the three algorithms use approximately equal CPU time for each problem size.

ANN

For ANN a preliminary initial temperature of 3.0 is used, which is dynamically adjusted upwards until the neurons are close to zero ($\sum_i v_i^2 < 0.1N$), in order to ensure a start close to the high-T fixed point.

The annealing rate is set to 0.99. At each temperature up to 10 sweeps are allowed in order for the neurons to converge, as signalled by the maximal change in value for a single neuron being less than 0.01. At every tenth temperature value, the cost function is evaluated using the signs of the mean-field variables, $s_i = \text{sign}(v_i)$; if this vanishes, a solution is found and the algorithm exits. If no solution has been found when the temperature reaches a certain lower

[1] `http://www.informatik.tu-darmstadt.de/AI/SATLIB`

bound (set to 0.1), the algorithm also exits; at that temparature, most neurons typically will have stabilized close to ± 1 (or occasionally 0). Neurons that wind up at zero are those that are not needed at all or equally needed as ± 1.

INN

For the INN approach, the same temperature parameters as in ANN are used except for the low T bound, which is set to 0.5. Because of the divergent nature of the cost function I (1.13) and the local field u_i (1.16), extra precaution has to be taken when updating the neurons – infinities appear when all the neurons in a clause are ± 1 with the wrong sign: $v_i = -C_{mi}$. When calculating u_i, the infinite clause contributions are counted separately. If the positive (negative) infinities are more (less) numerous, v_i is set to +1 (-1); otherwise, v_i is randomly set to ± 1 if infinities exist but in equal numbers, else the finite part of u_i is used.

This introduces randomness in the low temperature region if a solution has not been found; the algorithm then acquires a local search behaviour increasing its ability to find a solution. In this mode the neurons do not change smoothly and the maximum number of updates per temperature sweep (set to 10) is frequently used, which explains why INN needs more time than the conventional ANN for difficult problem instances. Performance can be improved, at the cost of increasing the CPU time used, with a slower annealing rate and/or a lower low-T bound. Restarts of the algorithm also improves performance.

gsat+walk

The source code for gsat+walk can be found at SATLIB [2]. We have attempted to follow the recommendations in the enclosed documentation for parameter settings. The probability at each flip of choosing a greedy move instead of a restricted random walk move is set to 0.5. We have chosen to use a single run with $200 \times N$ flips per problem, instead of several runs with less flips per try, since this appears to improve overall performance. Making several runs or using more flips per run will improve performance at the cost of an increased CPU consumption.

[2] `http://www.informatik.tu-darmstadt.de/AI/SATLIB`

1.5.4 Results

Here follow the results from our numerical investigations for the two testbeds. All explorations have been made on a 600 MHz AMD Athlon computer running Linux.

The results from INN, ANN and Gsat+Walk for the uniform random testbed are summarized in figures 1.3, 1.4, and 1.5.

In figure 1.3 the fraction of the problems not satisfied by the separate algorithms (f_U) is shown as a function of α for different problem sizes N. The three algorithms show different transitions in α above which they fail to find solutions. For INN and gsat+walk the transition appears slightly beneath the real α_c, while for ANN the transition is situated below $\alpha = 3.7$.

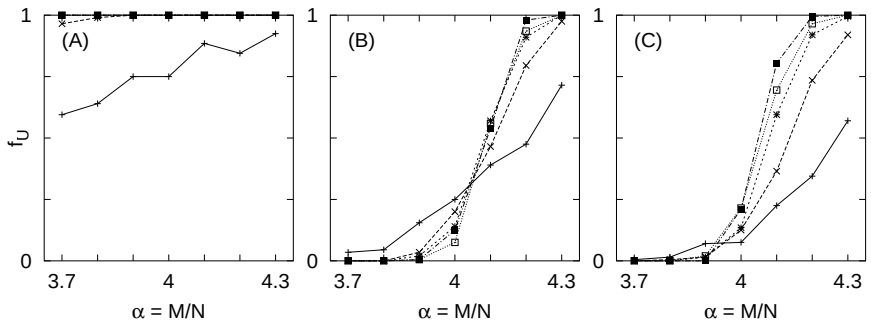

Figure 1.3: Fraction unsatisfied problems (f_U) versus α for ANN (**A**), INN (**B**) and gsat+walk (**C**), for $N = 100$ (+), 500 ($\times$), 1000 ($*$), 1500 ($\square$) and 2000 ($\blacksquare$). The fractions are calculated from 200 instances; the error in each point is less than 0.035.

The average number of unsatisfied clauses per problem instance (H) is presented in figure 1.4 for the three algorithms. H is shown as a function of α for different N. This can be used as a performance measure also when an algorithm fails to find solutions [3].

The average CPU-time consumption (t) is shown in figure 1.5 for all algorithms. The CPU-time is presented as a function of N for different α in order to show how the algorithms scale with problem size.

[3]Finding a maximal number of satisfied clauses for a SAT instance is referred to as MAXSAT [8].

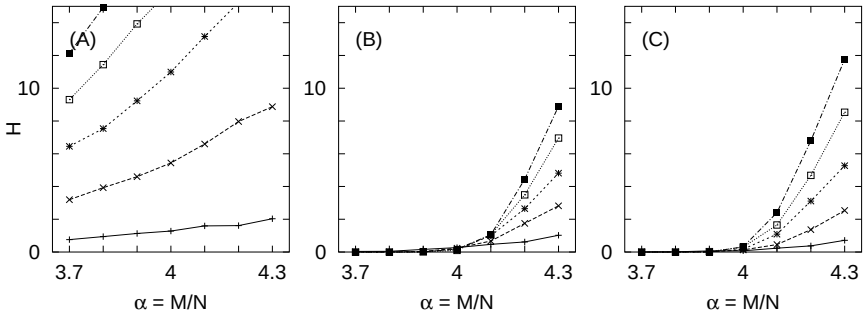

Figure 1.4: Number of unsatisfied clauses H per instance versus α, for ANN (**A**), INN (**B**) and gsat+walk (**C**), for $N = 100$ (+), 500 ($\times$), 1000 ($*$), 1500 ($\square$) and 2000 ($\blacksquare$). Average over 200 instances.

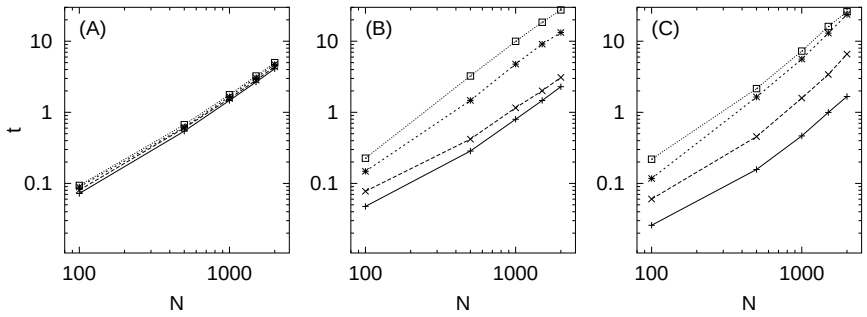

Figure 1.5: Log-log plot of used CPU-time t (given in seconds) versus N, for ANN (**A**), INN (**B**) and gsat+walk (**C**), for $\alpha = 3.7$ (+), 3.9 ($\times$), 4.1 ($*$) and 4.3 ($\square$). N ranges from 100 to 2000. Averaged over 200 instances.

The results (f_U, H, t) for the solvable testbed for all three algorithms are summarized in table 1.1.

1.5.5 Discussion

The first point to be made is the dramatic performance improvement in INN as compared to ANN. This is partly due to the divergent nature of the INN cost function I, leading to a progressively increased focus on the neurons in-

		num	ANN			INN			gsat+walk		
N	M	inst.	f_U	H	t	f_U	H	t	f_U	H	t
20	91	1000	0.23	0.25	0.01	0.08	0.08	0.01	0.00	0.00	0.01
50	218	1000	0.61	0.76	0.04	0.19	0.21	0.05	0.01	0.01	0.02
75	325	100	0.84	1.30	0.07	0.41	0.44	0.11	0.05	0.05	0.05
100	430	1000	0.84	1.49	0.09	0.32	0.36	0.13	0.07	0.07	0.09
125	538	100	0.88	1.72	0.11	0.39	0.41	0.18	0.10	0.10	0.13
150	645	100	0.89	2.07	0.14	0.34	0.40	0.23	0.16	0.17	0.19
175	753	100	0.98	2.60	0.17	0.51	0.61	0.39	0.27	0.28	0.33
200	860	100	1.00	3.06	0.20	0.60	0.81	0.52	0.32	0.34	0.39
225	960	100	0.97	3.15	0.22	0.52	0.67	0.51	0.35	0.37	0.46
250	1075	100	0.99	3.53	0.25	0.58	0.77	0.65	0.39	0.44	0.53

Table 1.1: Results for the solvable 3-SAT problems close to α_c. f_U is the fraction of problems not satisfied by the algorithm, H is the average number of unsatisfied clauses (1.6) and t is the average CPU-time used (given in seconds). The third column (num inst.) is the number of instances in the problem set.

volved in the relatively few critical clauses on the virge of becoming unsatisfied. This improves the revision capability which is beneficial for the performance. The choice of randomizing v_i to ± 1 (which appears very natural) in cases of balancing infinities in u_i contributes to this effect.

A performance comparison of INN and gsat+walk indicates that the latter appears to have the upper hand for small N. For larger N however, INN seems to be quite comparable to gsat+walk.

1.6 Summary and Outlook

We have presented a heuristic algorithm, INN, for binary satisfiability problems. It is a modification of the conventional mean-field based ANN annealing algorithm, and differs from this mainly by a replacement of the usual multilinear cost function by one derived from an information-theoretical argument.

This modification is shown empirically to dramatically enhance the performance on a testbed of random K-SAT problem instances; the resulting performance is for large problem sizes comparable to that of a good dedicated heuristic, tailored to K-SAT.

An important advantage of the INN approach is its generality. The basic philosophy – the balance of information – can be applied to a host of different

types of binary as well as non-binary problems; work in this direction is in progress.

Acknowledgement

Thanks are due to C. Peterson for inspiring discussions. This work was in part supported by the Swedish Foundation for Strategic Research.

References

[1] S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In Du et al. [2], pages 1–18.

[2] D. Du, J. Gu, and P. M. Pardalos, editors. *Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[3] J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. Algorithms for the satisfiability (sat) problem: A survey. In Du et al. [2], pages 19–152.

[4] T. Hogg, B. A. Hubermann, and C. P. Williams. Special volume on frontiers in problem solving: Phase transitions and complexity. *Artificial Intelligence*, 81(1,2), 1996.

[5] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[6] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137, 1999.

[7] M. Ohlsson, C. Peterson, and B. Söderberg. Neural networks for optimization problems with inequality constraints - the knapsack problem. *Neural Computation*, 5(2):331–339, 1993.

[8] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

[9] C. Peterson and B. Söderberg. Neural optimization. In M. A. Arbib, editor, *The Handbook of Brain Research and Neural Networks, (2nd edition)*, pages 617–622. Bradford Books/The MIT Press, 1998.

[10] B. Selman, H. A. Kautz, and B.Cohen. Noise strategies for improving local search. In *Proceedings of AAAI-94*, pages 46–51, Menlo Park, California, 1994. AAAI Press.

An Information-Based Neural Approach to Generic Constraint Satisfaction

Paper II

An Information-Based Neural Approach to Generic Constraint Satisfaction

Henrik Jönsson and Bo Söderberg

Complex Systems Division, Department of Theoretical Physics
Lund University, Sölvegatan 14A, S-223 62 Lund, Sweden
http://www.thep.lu.se/complex/

LU TP 00-40, submitted to *Artificial Intelligence*

A novel artificial neural network heuristic (INN) for general constraint satisfaction problems is presented, extending a recently suggested method for binary problems. It employs a particular non-polynomial cost function, based on the information balance between multi-state Potts variables and constraints. Implemented as an annealing algorithm, the method is numerically explored on a testbed of Graph Coloring problems. The performance is comparable to that of dedicated heuristics, and clearly superior to that of a conventional mean-field ANN approach. An appealing feature of the method is its versatility, inherited from ANN; it is applicable to a wide range of discrete constraint satisfaction problems.

2.1 Introduction

Artificial Neural Networks (**ANN**) have provided a versatile heuristic approach to combinatorial optimization and constraint satisfaction [7, 14, 4].

In a recent paper [9], an improved ANN approach (**INN**) to binary constraint satisfaction problems (**CSP**) was presented, based on information-theoretical considerations. Numerical explorations on a testbed of K-sat instances showed a substantially improved performance as compared to a conventional ANN method. This improvement can be understood, at least partly, as being due to a progressively increased sensitivity to the breaking of constraints, stemming from the nonlinearity of the cost function of INN.

In this paper we provide an extension of the binary INN approach to general constraint satisfaction by utilizing an encoding in terms of Potts spins, and present an annealing algorithm based on this approach. Like the binary method, it is derived from an analysis of the information balance between variables and constraints. The result is a powerful general-purpose heuristic method, that can be applied to a large class of constraint satisfaction problems.

As a specific application example we have chosen *Graph Coloring* (**GC**) [13], and we provide a detailed discussion of the implementation of INN for this problem type.

The method is numerically explored for GC with three colors on a large testbed of random graphs with varying edge densities.

Like for the binary K-sat problems, a remarkably improved performance is observed, as compared to a conventional ANN approach. To gauge the performance, we have applied two dedicated heuristics to the testbed; a biased simulated annealing approach, *SAU* [8], and a simple search heuristic, *DSATUR* [1, 3].

The structure of the paper is as follows: In Section 2, we present a general derivation of the method for a generic CSP, based on information analysis. In Section 3, we show in detail the specific application of the method to GC, and discuss algorithmic details. Section 4 contains a description of the numerical explorations and the testbeds, as well as a presentation and a discussion of the results. Finally, Section 5 contains a brief summary and our conclusions.

2.2 INN for Non-binary (Potts) Systems - General Derivation

In this section we will, in the context of a general CSP, explain the ideas behind the INN method, which are based on analyzing the information content in the constraints to be satisfied.

In a CSP a set of N discrete variables is given; we assume each variable to have C possible states (defining a C-state Potts variable [15]), yielding a state space of size C^N. Further, a set of M constraints is given, restricting the state space to a smaller set of admissible states (solutions). We assume that each constraint involves a function of a subset of size K of the N variables.

We have limited the discussion to fixed C and K for simplicity only – the method can easily be adapted to problem types with varying C as well as K.

The important question is whether such a CSP is solvable, i.e. whether the state space contains any solutions. Finding a solution suffices to prove solvability, while proving non-solvability is in general harder.

In heuristic approaches to CSP one uses some method to attempt to find a solution. ANN and its modified version INN are such methods.

In what follows, we will label each variable by an integer $i \in \{1, 2, \ldots N\}$. The state of each variable (Potts spin) is encoded in terms of a C-dimensional vector $\mathbf{s}_i = \{s_{i1}, s_{i2}, \ldots s_{iC}\}$, with the cth state given by the cth principal vector, $s_{ic} = 1; s_{ic'} = 0, c' \neq c$. The state of the entire set of variables is denoted by $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_N\}$.

2.2.1 Conventional MF ANN Approach

In the conventional ANN approach, the problem at hand is encoded in terms of a non-negative cost function E over the state space, such that $E(\mathbf{s})$ vanishes iff $\mathbf{s}$ is a solution. Normally, E is chosen as a polynomial in $\mathbf{s}$, such that each term is the product of components from distinct Potts spins (*multilinearity*).

The state space is placed in a virtual heat bath represented by a Boltzmann distribution over the state space, such that the probability of a state $\mathbf{s}$ is proportional to $\exp(-E(\mathbf{s})/T)$, where T is an artificial temperature. At high T all states are about equally probable, while as $T \to 0$, the support of the distribution shrinks to contain only the states with the lowest cost (the solutions for a solvable problem), all equally probable.

The Mean Field Approximation

One then considers the mean field (**MF**) approximation $\mathbf{v}_i$ to the thermal averages $\langle \mathbf{s}_i \rangle$ of the state vectors $\mathbf{s}_i$, defined by a self-consistent set of equations for the **MF spins** $\mathbf{v}_i$, the **MF equations**, given by

$$v_{ic} = \frac{\exp(u_{ic})}{\sum_d \exp(u_{id})}, \tag{2.1}$$

$$u_{ic} = -\frac{1}{T}\partial E/\partial v_{ic}. \tag{2.2}$$

Note that an MF spin $\mathbf{v}_i$ is constrained to the convex hull of the set of allowed states for the corresponding Potts spin $\mathbf{s}_i$. In particular, $\sum_c v_{ic}$ is automatically equal to 1, and v_{ic} can be interpreted as the probability for the ith Potts spin to be in its cth state.

Variational MF Derivation

The MF approximation can be derived from a variational principle, where the true Boltzmann distribution $\propto \exp(-E/T)$ is approximated by one with independent probabilities for each spin, defined by the components of $\mathbf{v}_i$. These are optimized by minimizing a free energy, given by

$$F(\mathbf{v}) = T\sum_{ic} v_{ic}\log(v_{ic}) + E(\mathbf{v}), \tag{2.3}$$

with the restriction that $v_{ic} \geq 0$ and $\sum_c v_{ic} = 1$. The first term amounts to $-TS(\mathbf{v})$, where S is the entropy of $\mathbf{v}$. At an extremal value we must have $\partial F/\partial v_{ic} = \lambda_i$, where λ_i is a Lagrange multiplier for the unit-sum constraint on $\mathbf{v}_i$. This yields

$$T\log(v_{ic}) + T + \partial E/\partial v_{ic} = \lambda_i, \tag{2.4}$$

which leads to the MF equations (2.1,2.2).

MF Annealing

The MF equations (2.1,2.2) are solved iteratively, updating one spin at a time, while the temperature is slowly decreased from an initial high value (with all $v_{ic} \approx 1/C$). As T becomes low enough, the fuzzy MF variables will eventually converge towards a sharp state, $\mathbf{v}_i \rightarrow \mathbf{s}_i$, which defines the output of the algorithm. If this defines a solution, the CSP is proven solvable.

2.2.2 INN Approach

INN can be seen as an information-based modification of ANN, where a very specific cost function is chosen, based on an information-theoretical analysis of the constraints, which for each constraint typically leads to the negative logarithm of a polynomial function yielding unity if the constraint is satisfied, and zero if not. This yields a divergent cost for a non-solution, which in an MF setting leads to a sensitivity to softly broken constraints that progressively increases with severity.

INN cost function

To construct such a cost function, we make the assumption (inherent in the MF approximation) that the Boltzmann distribution at each temperature factorizes into a product of single-spin distributions,

$$P(\mathbf{s}) = \prod_i p_i(\mathbf{s}_i). \tag{2.5}$$

Each single-spin distribution p_i is completely defined by an MF spin $\mathbf{v}_i$, with the probability for spin i to be in state c given by v_{ic}. Thus, $\mathbf{v}$ can be seen as a parameterization of P.

Now, assume the mth constraint is defined by a function f_m of a subset Ω_m of the spins as

$$f_m\left(\mathbf{s}_i, i \in \Omega_m\right) = 0. \tag{2.6}$$

Then, the probability distribution P, parameterized by $\mathbf{v}$, gives a well-defined probability $U_m(\mathbf{v})$ that the constraint be unsatisfied; this will be a polynomial in the involved MF spins $\mathbf{v}_i, i \in \Omega_m$, with the degree given by the number K of involved spins. Specifically, it amounts to

$$U_m(\mathbf{v}) = \sum_{c_1,\ldots,c_K} v_{i_1,c_1} \ldots v_{i_K,c_K} \Theta_{c_1 \ldots c_K}, \tag{2.7}$$

where Θ is zero if the state combination $(c_1, \ldots, c_K)$ for the K spins $(i_1, \ldots, i_K)$ in Ω_m solves the constraint, and unity otherwise.

The amount of information required to force a constraint to be satisfied can be estimated as $-\log(1 - U_m)$, and as a cost function we take the sum of this over the set of constraints, i.e.

$$I(\mathbf{v}) = -\sum_m \log(1 - U_m(\mathbf{v})), \tag{2.8}$$

which is thus an estimate of the total amount of information needed for solving the problem.

As a comparison, the polynomial $E(\mathbf{v}) = \sum_m U_m(\mathbf{v})$ defines a possible conventional ANN cost function, corresponding to a Boltzmann distribution over the state space given by $P_E(\mathbf{s}) \propto \prod_m \exp(-U_m(\mathbf{s})/T)$ with a penalty factor of $\exp(-1/T)$ for each broken constraint.

The INN cost function I formally corresponds to the more radical Boltzmann distribution

$$P_I(\mathbf{s}) \propto \prod_m (1 - U_m(\mathbf{s}))^{\frac{1}{T}} , \tag{2.9}$$

which, since $1 - U_m(\mathbf{s})$ yields 1 for a solution and 0 for a non-solution, vanishes for all non-solutions and yields a uniform non-zero weight for all solutions, independently of T (the T dependence appears only in the MF equations).

INN MF Annealing

When using the INN cost function in place of the conventional ANN one in the MF equations (2.1,2.2), a slight modification is needed due to its non-polynomial nature, in order to avoid instabilities [12]. Eq. (2.2), which is relevant for a multilinear polynomial cost function, has to be replaced by

$$u_{ic} = -\frac{I(\mathbf{v})|_{\mathbf{v}_i=\hat{c}}}{T} \tag{2.10}$$

(plus an unimportant constant), where $\hat{c}$ denotes the state where $v_{ic} = 1$ while the other components vanish. This amounts to using cost *differences* rather than derivatives, and is equivalent to (2.2) for a multilinear cost function.

Due to the singular behavior of the logarithm in eq. (2.8) and thus in eq. (2.10) for small arguments, it may happen that one or more components of $\mathbf{u}_i$ become divergent ($-\infty$) within the numerical resolution, which is the case when a constraint is close to becoming fully broken for a particular choice of state $\hat{c}$, which typically happens at a low T.

If not all components are divergent, there is no problem: The corresponding components of $\mathbf{v}_i$ will become zero. However, in cases where all components would become divergent, a regularization method has to be devised.

To that end, a counter n_{ic} is assigned to each state c of the variable, initialized to zero, and incremented for each constraint that would make a singular contribution (which is skipped) to u_{ic}. In cases where all the counters are non-zero,

only the states with the lowest count are considered, and the corresponding components are set to equal values summing up to one, while the others are set to zero. Thus, the finite contributions are ignored in this case.

This seems to be a reasonable choice, preserving the deterministic property of the MF equations, as opposed to the stochastic choice made in [9]. The deterministic variant appears to perform slightly worse, but is considerably faster, due to better convergence properties.

In practice, when evaluating the INN cost function in eq. (2.10) it is faster to take the logarithm of a product than to sum the logarithms. It may happen that the product vanishes numerically, yielding a divergent logarithm; this is treated as an extra divergent contribution, and the corresponding counter is incremented.

With these modifications, INN annealing proceeds just like conventional ANN annealing: The modified MF equations are solved iteratively in a serial manner, with annealing in T, etc. However, due to the nonlinearity of the logarithm in the INN cost function (2.8), constraints on their way to becoming unsatisfied are detected on an earlier stage, leading to a better revision capability for INN as compared to ANN, and a substantially improved performance.

Variational Interpretation

Formally, the modified MF approximation can be seen as the result of the minimization of a variational free energy,

$$F(\mathbf{v}) = -TS(\mathbf{v}) + I(\mathbf{v}), \tag{2.11}$$

where the first term can be interpreted as a measure of the unused information resources associated with the MF spins. At high T, all states are about equally probable, and $v_{ic} \approx 1/C$. The information content $-S$ is then high, amounting to $\log(C)$ per spin (one bit for $C = 2$). As T is lowered, a specific state is chosen for each spin, and the information resources are used up, $-S \to 0$.

Thus, in the INN approach, the spins can be seen as information *resources*, that are gradually used to satisfy the constraints, seen as information *consumers*.[1] At high T (typically above a well-defined critical temperature, T_c), the resources are intact, but as T is decreased, an increasing pressure is applied towards spending them.

[1] This might seem as an abuse of the notion of information, but we think the point is clear.

2.3 Application to Graph Coloring

Now we leave the general discussion in favor of an application to a specific problem type, for which we have chosen *Graph Coloring* (**GC**).

In GC, a graph of N nodes and M edges is given, and a set of C colors. To each node a specific color is to be assigned, such that each edge connects nodes of distinct colors. Thus, there is one constraint for each edge, involving $K = 2$ variables.

2.3.1 INN for Graph Coloring

We assign a Potts spin $\mathbf{s}_i$ to encode the coloring of each node i, $i = 1, \ldots, N$, and employ the (modified) MF approximation, yielding MF spins $\mathbf{v}_i$.

The scalar product $\mathbf{s}_i \cdot \mathbf{s}_j$ yields unity if the nodes i, j are in the same state, and zero if not. The analogous expression with MF spins, $\mathbf{v}_i \cdot \mathbf{v}_j$, measures the probability that the nodes are in the same state, which is precisely what we need for U_m, so let

$$U_m(\mathbf{v}) = \mathbf{v}_{i_m} \cdot \mathbf{v}_{j_m}, \tag{2.12}$$

where i_m, j_m label the two nodes connected by the edge m.

Let J denote the connection matrix for the graph, i.e. $J_{ij} = 1$ if i, j are connected, zero otherwise. Then the INN cost function $I = -\sum_m \log(1 - U_m)$ can be expressed as

$$I = -\frac{1}{2} \sum_{i,j} J_{ij} \log\left(1 - \mathbf{v}_i \cdot \mathbf{v}_j\right). \tag{2.13}$$

The INN MF equations become $v_{ic} = exp(u_{ic}) / \sum_d \exp(u_{id})$, with

$$u_{ic} = \frac{1}{T} \sum_j J_{ij} \log\left(1 - v_{jc}\right), \tag{2.14}$$

where care has to be taken to regularize singular contributions, as discussed in Sec. 2.2.2. Note that a component of $\mathbf{u}_i$ gets a singular contribution from an edge only in the limit of the corresponding component of $\mathbf{v}_j$ being unity (within the numerical resolution), which is more likely to happen at low T.

Critical T Discussion

The MF equations have a trivial solution for

$$v_{ic} = \frac{1}{C}, \tag{2.15}$$

which is a fixed point of the dynamics. By means of a linearization around this fixed point, the dynamics for infinitesimal deviations $\epsilon_{ic} = v_{ic} - 1/C$ becomes

$$\epsilon_{ic} = -\frac{1}{T(C-1)} \sum_j J_{ij} \left(\epsilon_{jc} - \frac{1}{C} \sum_d \epsilon_{jd} \right), \tag{2.16}$$

which is stable at high T, and becomes unstable at a critical temperature T_c given by $-\frac{\lambda}{C-1}$, where λ is the largest (in absolute value) negative eigenvalue of J. T_c serves as a suitable initial temperature in the annealing algorithm, and can easily be estimated. For a graph with at least one edge, $1/(C-1)$ is a strict lower bound for T_c.

2.4 Numerical Explorations

2-coloring ($C = 2$) is known to be of polynomial complexity, and we will focus on 3-coloring ($C = 3$), which is NP-complete [13].

Problem difficulty depends on the edge density $\gamma = 2M/N$. For 3-coloring, there exists a critical edge density $\gamma_c \approx 4.6$ [6], such that in the large-N limit all problems are solvable below γ_c, and unsolvable above. (Such phase transitions are known to exist in many classes of CSP [5, 11].)

Although the problem of finding a solution becomes increasingly difficult with increasing γ, the decidability problem is most difficult around γ_c; at low γ a solution is easy to find, while at higher γ, it is easier to prove unsolvability.

2.4.1 Testbeds

To probe the performance of INN annealing as applied to 3-coloring, we have chosen a testbed of random graphs, with edge densities in the neighborhood of γ_c. To be specific, we have used γ values between 3.4 and 4.6 in steps of 0.2, and in addition 4.1 and 4.3. Problem sizes probed are $N = 250, 500, 1000, 2000$. For a given edge density and problem size, the edge count is given as $M = \gamma N/2$, and a set of 200 random problem instances are generated by for each instance choosing at random M of the $N(N-1)/2$ possible edges.

Preprocessing

A simple preprocessing is made on the graphs before they are handed to the algorithms. Nodes with less than $C = 3$ neighbors are removed from the graph, since they can be trivially colored. This is done recursively until the remaining nodes have at least C neighbors. In table 2.1 the average sizes of the graphs after preprocessing is shown for some original values of N and γ. The preprocessing can be expected to improve speed about equally much for

	$\gamma = 3.6$			$\gamma = 4.2$			$\gamma = 4.6$		
N	M	N'	M'	M	N'	M'	M	N'	M'
250	450	130	259	525	181	414	575	201	496
500	900	253	506	1050	361	827	1150	401	990
1000	1800	509	1020	2100	720	1649	2300	799	1973
2000	3600	1025	2055	4200	1437	3293	4600	1596	3943

Table 2.1: Average problem size reduction due to preprocessing. N, M and γ are the original problem parameters, while N' and M' denote the reduced node and edge counts.

the different algorithms, which is confirmed by empirical test runs. These also indicate a comparable performance improvement in the form of a small change (slightly larger for ANN) in the position in γ of the (algorithm-dependent) apparent phase transition.

2.4.2 Algorithmic Details for INN

The temperature is initially set close to an estimate of the critical temperature T_c, obtained by iterating $\mathbf{x} \rightarrow (const \times \mathbf{1} - J)\,\mathbf{x}$ a few dozen times, with a suitable random initial vector $\mathbf{x}$. A schematic description of the algorithm is given in fig. 2.1. For the annealing factor we have made experiments with different values. Slower annealing tends to improve the quality for large and difficult problems, but at the cost of more CPU time used. There is also a limit where slower annealing does not improve the quality anymore, and a restart of the algorithm gives a better payoff. The presented results are consistently based on an annealing rate of 0.99, and we have allowed for a maximum of ten updates of all spins per temperature, to allow the spins to converge ($\max_{ic} |\Delta v_{ic}| < 0.1$).

At every tenth temperature we check for two stop criteria. First, a temporary sharp configuration $\mathbf{s}$ is extracted from $\mathbf{v}$, and if $\mathbf{s}$ is a solution the algorithm stops. The second criterion applies if the spins are saturated ($\sum_i \sum_c v_{ic}^2 >$

1. Set $T \approx T_c$, and initialize $\mathbf{v}$ close to the high-T fixed point, $v_{ic} = 1/C$, with small random deviations (a few %).
2. For each node i in turn:
   (a) Compute $\mathbf{u}_i$ using eq. (2.10).
   (b) Update $\mathbf{v}_i$ according to eq. (2.1). (In case of singular components in $\mathbf{u}_i$, this pair of steps is modified as discussed in Sec. 2.2.2).
3. If any of the stop criteria is met, go to 5.
4. Lower T by a fixed annealing factor, and go to 2
5. Extract a candidate solution by for each node i choosing a color corresponding to the largest component of $\mathbf{v}_i$.

Figure 2.1: The INN annealing algorithm for GC.

$0.9N$) and stable ($\max_{ic} |\Delta v_{ic}| < 0.01$). In addition, if no solution has been found until $T < 0.3$ the algorithm aborts.

2.4.3 Comparison Algorithms

The performance of INN annealing has been compared to that of three other algorithms: **1)** conventional ANN annealing, **2)** a biased simulated annealing algorithm, SAU [8], and **3)** a heuristic, DSATUR [1, 3], all applied to the same testbed.

Comparing different algorithms is quite difficult, as they have different time scales at which they are most efficient. Also, each algorithm has different optimal parameter settings for different N and γ. Despite this, we have for each algorithm for simplicity used the same parameter settings on the entire testbed.

We have chosen a set of suitable parameters (without attempting to fine-tune them) for the INN algorithm first, and then tried to roughly optimize the settings for the other algorithms, given that they are allowed to use about the same time as INN.

Conventional ANN annealing

A suitable ANN cost function, corresponding to (2.13) is

$$E = \frac{1}{2} \sum_{i,j} J_{ij} \mathbf{v}_{i_m} \cdot \mathbf{v}_{j_m}, \tag{2.17}$$

yielding the MF equations (2.1) with

$$u_{ic} = -\frac{1}{T} \sum_{j} J_{ij} \mathbf{v}_{j_m}. \tag{2.18}$$

T is initialized close to the critical temperature, given by $(C-1)/C$ times that for INN. We have used an annealing rate of 0.99, and the same stop criteria as in INN, except for the stop temperature, which is set to 0.1 instead of 0.3 (lower than for INN, where the non-linear cost function makes the spins saturate faster).

SAU

A number of simulated annealing [10] approaches have been devised for graph coloring problems [8, 2]. For 3-coloring it is appropriate to use one with a fixed number of colors where the goal is to minimize the number of broken edges. In such an approach each node in the graph is assigned a variable, $c_i = 1, .., C$ representing the color of the node. A cost analogous to eq. 2.17, defined as

$$E = \frac{1}{2} \sum_{i,j} J_{ij} \delta_{c_i c_j}, \tag{2.19}$$

can be used. Local moves are selected by choosing a node and a new random color from the $C-1$ available. To guide the search into low cost states a virtual temperature parameter, T, is introduced and moves are accepted with a probability $p = \min\left(\exp\left(-\Delta E/T\right), 1\right)$, where $\Delta E = E_{new} - E_{old}$. If the temperature is high all moves are accepted, while at low temperature uphill moves (increased cost) are rejected.

In [8], an algorithm of this type is described, which we will refer to as SAU. There, a restricted move class is used, where only nodes contributing to the cost (as opposed to all nodes) are allowed as candidates for a color change; this is empirically more efficient.

This move class is not symmetric, and corresponds to a kind of biased simulated annealing, which does not necessarily yield an emulation of a Boltzmann

distribution. Also, ergodicity seems to be broken: All possible states can not be reached from any initial state; this does not have to be a disadvantage.

The algorithm starts in a random state and at a high temperature to allow for uphill moves. A certain number ($2N$) of attempted moves (accepted or not) define an iteration; between iterations the temperature is decreased by a fixed factor (0.97). This is repeated until a solution is found, or the cost has not changed over a certain number (10) of iterations. The annealing is beneficial to avoid local minima.

DSATUR

The DSATUR algorithm is designed to answer the question how many colors are needed to color a graph; it always succeeds, and returns the number of colors used. If this is less than or equal to C, a solution to C-coloring is implied.

DSATUR starts with all nodes uncolored, and selects nodes to color one by one. The order in which nodes are selected is dynamically determined by the number of colors that can not be used because of already colored neighbor nodes. In each step the node with the smallest number of available colors is selected; if several, a random selection is made.

This algorithm was presented by D. Brélaz [1], and we have used a program made by J. Culberson [3] available from the World-Wide Web [2]. In [3] a set of similar algorithms was presented, with varying rules for ordering the nodes. Our choice of DSATUR among these was based on preliminary experiments indicating that DSATUR performed best on the class of problems used in our testbed.

2.4.4 Results

Here follows a discussion and evaluation of the testbed results for INN and the comparison algorithms.

In figures 2.2 to 2.5 we present, for each algorithm, (A) the fraction unsolved problems, f_U, as a function of edge density γ for different problem sizes, and (B) the average CPU time used as a function of problem size N, for different edge densities.

The time presented is the total time used, including reruns, until a solution is

[2] `http://web.cs.ualberta.ca/~joe/Coloring/`

found or the maximum allowed number of reruns is done. The time resolution is a mere 1/100 of a second, which means that the shorter times tend to be somewhat underestimated.

Parameter settings are as described above, and up to ten restarts are allowed for each algorithm on a problem, except for the faster DSATUR algorithm where up to 80 restarts are allowed. All experiments have been made on a 600 MHz AMD Athlon computer running Linux.

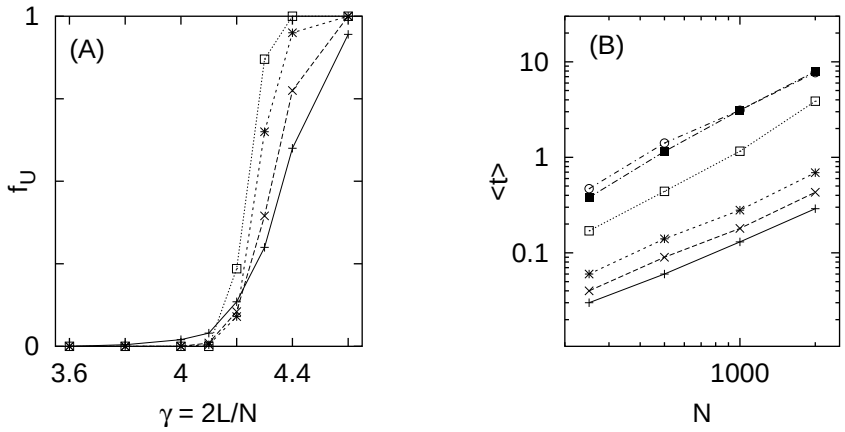

Figure 2.2: INN results from 200 instances for each N and γ. (**A**) Fraction unsolved problems (f_U) versus γ, for $N = 250$ (+), 500 ($\times$), 1000 ($*$), 2000 ($\square$). The statistical error in each point is less than 0.035. (**B**) Average CPU time (in seconds) used versus N, for $\gamma = 3.6$ (+), 3.8 ($\times$), 4.0 ($*$), 4.2 ($\square$), 4.4 ($\blacksquare$) and 4.6 ($\circ$).

Discussion

As can be seen in figures 2.2 - 2.5, each algorithm seems to show a more or less distinct critical γ, above which it fails to find solutions. In all cases it is situated below the established value of $\gamma_c \approx 4.6$, and can be used as a measure of the performance. This indicates that INN and SAU are the best algorithms for difficult problems (for lower γ where all problems are solved by either method, DSATUR wins on speed).

A closer look at the INN and SAU results reveals a tendency for SAU to perform slightly better in the lower γ range, while INN seems to have the upper hand

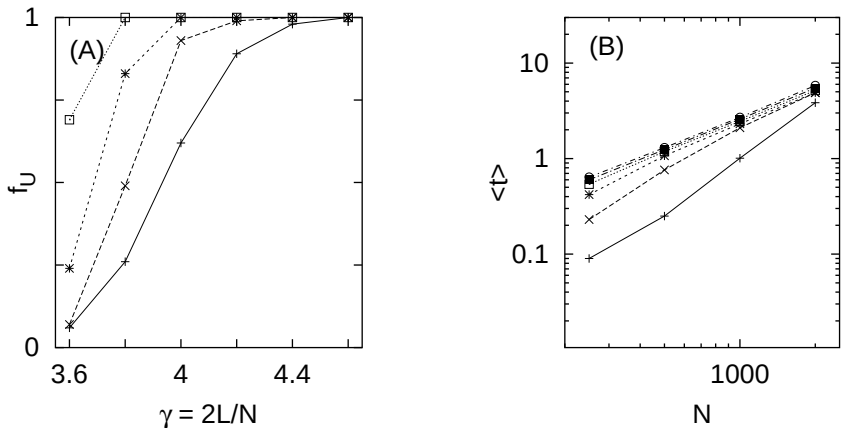

Figure 2.3: Conventional ANN results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.2.

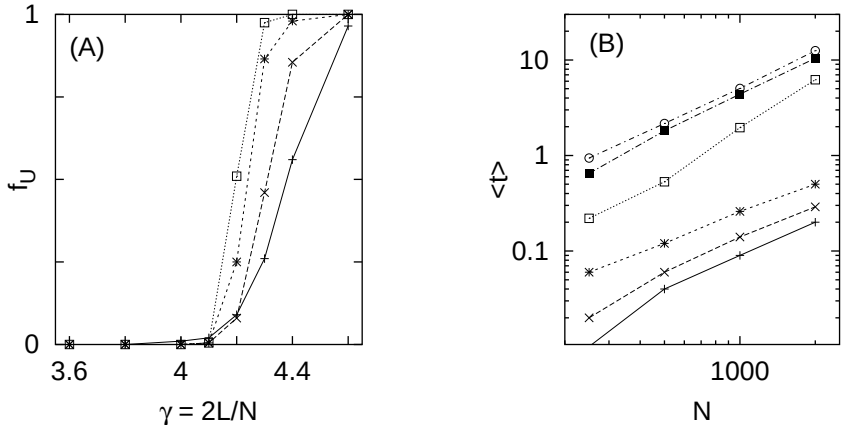

Figure 2.4: SAU results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.2.

for higher γ. To some extent, this is an effect of the chosen amount of CPU time allowed, and a different choice might slightly change the outcome; both INN and SAU would benefit from a slower annealing rate, requiring more time.

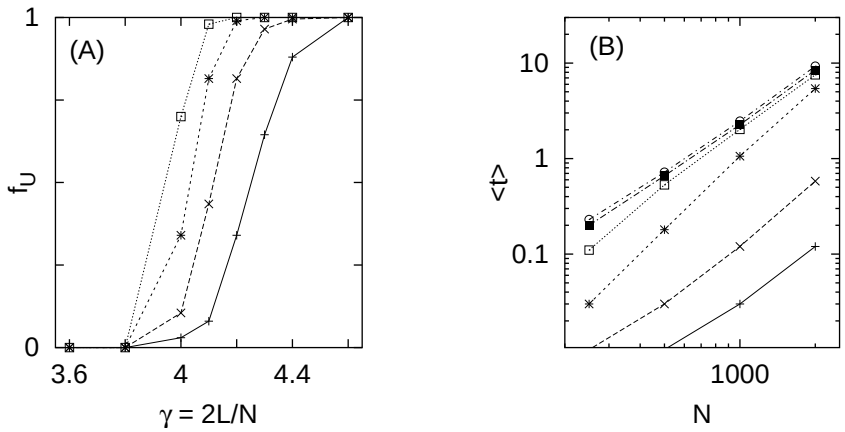

Figure 2.5: DSATUR results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.2.

As for CPU time consumption, DSATUR seems to be very fast in solving the easier problems for small N, although the differences in time may be somewhat overestimated due to the finite time resolution. On the other hand, the DSATUR time rises faster with increasing N, and for large N the time consumption is comparable to that of the other algorithms.

Our overall conclusion is that INN and SAU have comparable performances and are the overall winners. They are consistently superior to ANN, and beat DSATUR for the large/difficult problems, where it matters the most.

Solution rates

When comparing performances for heuristics for finding a single solution to a problem, a single simple quality measure is desirable.

A possible candidate would be the *instantaneous solution rate*, $r(t) = -\dot{U}(t)/U(t)$, where $U(t)$ is the fraction of the problem instances in an ensemble that are not solved within time t (in case of reruns, this is the total accumulated time). Suitably binned in time, $r(t)$ provides information on the time-scales at which a heuristic is most efficient. For a random search (where U decreases exponentially), $r(t)$ will be a constant. However, for a typical heuristic, and with limited statistics, it will fluctuate too much to be useful for producing a stable number to be used for comparing performances between algorithms.

Instead, we consider the *average solution rate* $R(t)$, which will have a smoother time-dependence. It is defined as

$$R(t) = -\frac{\log U(t)}{t}, \tag{2.20}$$

and can be interpreted roughly as the fraction solved problems per unit of time, averaged between 0 and t. It will also yield a constant for a random search, and can be expected to be a slowly varying function of t, after an initial transient, for a typical heuristic.

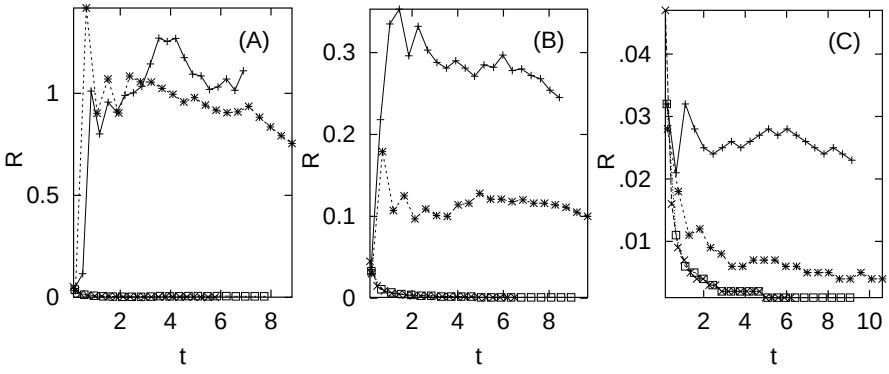

Figure 2.6: Average solution rates for $N = 2000$, for INN (+), ANN ($\times$), SAU ($*$), and DSATUR ($\square$), at (A) $\gamma = 4.1$, (B) $\gamma = 4.2$, (C) $\gamma = 4.3$.

Figure 2.6 shows the average solution rates $R(t)$ for the testbed problems with $N = 2000$ and $\gamma = 4.1, 4.2, 4.3$, for the four algorithms. The rates are based on an expected $U(t)$ for the respective problem ensemble, computed as $U(t) = (n_U(t)+1)/(n+2)$, where $n_U(t)$ is the number of unsolved problems remaining at t, of a total of $n = 200$ for each γ value.

For all three γ values, INN and SAU appear much superior to ANN and DSATUR. For $\gamma = 4.1$, the rates for INN and SAU are about equal, after the initial transients have died out. For $\gamma = 4.2$, the INN rate settles to about a factor 2.5 higher than the SAU rate; for $\gamma = 4.3$, there appears to be a factor of about 4 in favor of INN.

The precise ratios should not be taken too seriously; they will depend on annealing rates and precise stop criteria, etc. However, figure 2.6 does confirm very clearly the qualitative conclusion from figs. 2.2 and 2.4, that INN per-

forms somewhat better than SAU for the higher γ values, and that both beat ANN and DSATUR clearly.

2.5 Summary and Conclusions

We have presented a modified ANN annealing heuristic, INN, for generic CSP, generalizing a previously described method for binary CSP. It is based on an analysis of the balance of information between constraints and mean-field variables, yielding a very specific non-polynomial cost function.

The method was applied to a testbed of random graph 3-coloring problems, and the performance was shown to be in parity with a good dedicated heuristic, SAU, and much superior to that of a conventional ANN approach with a polynomial cost function.

The improvement compared to traditional ANN can be attributed to the strong non-linearity of the particular cost function used in INN, which boosts the ability to recognize and avoid bad solutions on an early stage, and yields an improved revision capability.

The method shares with ANN the appealing feature of not being tailored for a specific application, but can be applied to a large class of constraint satisfaction problems.

For constrained optimization problems, we suggest a hybrid method, where the constraints are handled by a non-linear information-based cost function, while the proper object function is treated with a traditional polynomial ANN cost function. Work to explore this avenue is in progress.

Acknowledgements

Thanks to D. Blancenbecler, M. Ohlsson and C. Peterson for stimulating discussions. This work was in part supported by the Swedish Foundation for Strategic Research.

References

[1] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[2] M. Chams, A. Hertz, and D. de Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2):260–266, 1987.

[3] J. C. Culberson and F. Luo. Exploring the k-colorable landscape with iterated greedy. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challange, 1993 DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 245–284. American Mathematical Society, 1996.

[4] L. Gislén, B. Söderberg, and C. Peterson. Complex scheduling with Potts neural networks. *Neural Computation*, 4(6):805–831, 1992.

[5] T. Hogg, B. A. Hubermann, and C. P. Williams. Special volume on frontiers in problem solving: Phase transitions and complexity. *Artificial Intelligence*, 81(1,2), 1996.

[6] Tad Hogg. Applications of statistical mechanics to combinatorial search problems. In D. Stauffer, editor, *Annual Reviews of Computational Physics*, volume 2, pages 357–406. World Scientific, 1995.

[7] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[8] D. S. Johnson, C. R. Aragon, and L. A. McGeoch. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partition. *Operations Research*, 39(3):378–406, 1991.

[9] H. Jönsson and B. Söderberg. An information-based neural approach to constraint satisfaction. *Neural Computation*, 13:1827–1838, 2001.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[11] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137, 1999.

[12] M. Ohlsson, C. Peterson, and B. Söderberg. Neural networks for optimization problems with inequality constraints - the knapsack problem. *Neural Computation*, 5(2):331–339, 1993.

[13] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

[14] C. Peterson and B. Söderberg. Neural optimization. In M. A. Arbib, editor, *The Handbook of Brain Research and Neural Networks, (2nd edition)*, pages 617–622. Bradford Books/The MIT Press, Cambridge, Massachusetts, 1998.

[15] F. Y. Wu. The Potts model. *Review of Modern Physics*, 54(1):235–268, 1982.

Deterministic Annealing and Nonlinear Assignment

Paper III

Deterministic Annealing and Nonlinear Assignment

Henrik Jönsson and Bo Söderberg

Complex Systems Division, Department of Theoretical Physics
Lund University, Sölvegatan 14A, S-223 62 Lund, Sweden
http://www.thep.lu.se/complex/

LU TP 01-16.

For combinatorial optimization problems that can be formulated as Ising or Potts spin systems, the Mean Field (MF) approximation yields a versatile and simple ANN heuristic, Deterministic Annealing.
For assignment problems the situation is more complex – the natural analog of the MF approximation lacks the simplicity present in the Potts and Ising cases.
In this article the difficulties associated with this issue are investigated, and the options for solving them discussed. Improvements to existing Potts-based MF-inspired heuristics are suggested, and the possibilities for defining a proper variational approach are scrutinized.

3.1 Introduction

Many mathematical methods originating from theoretical physics have found use in completely different contexts, among them the variational approach to the thermodynamics of complicated systems, lying at the basis of e.g. the mean field approximation to spin systems. This has been successfully used in heuristic methods in the context of combinatorial optimization, for problems that allow a simple formulation in terms of Ising or Potts spins. For other kinds of combinatorial optimization problems, in particular assignment problems, a similar approach is more difficult to achieve; the related difficulties is the focus of this paper.

In an instance of a combinatorial optimization problem, a cost function is defined in terms of a set of discrete variables, and the object is to find an optimal state – a particular state of the variables that minimizes the cost function; in other words the ground state, if the cost function is interpreted as a Hamiltonian. In cases where the variables are of a binary nature, such a problem thus amounts to finding the ground state of an Ising spin system (spin glass) with a given Hamiltonian.

For a small problem instance, an exact method can be used to solve it exactly. In addition to more problem-specific methods, Branch-and-Bound [9] provides a generic class of exact methods, where an intelligent (as opposed to exhaustive) tree-search of the phase-space is performed, disregarding parts that can be ruled out beforehand. Another interesting approach is Simulated Annealing [5], where a standard Monte-Carlo method is used to simulate the immersion of the system in a heat bath, starting at a high temperature, which is slowly lowered (annealing) in the course of the simulation. In the limit of very slow annealing, this stochastic method is guaranteed to yield the ground state as $T \to 0$ [1].

For a large system, however, finding the exact ground state can be a very time-consuming task. For a large class of problems (NP-hard), the expected time required scales worse than any polynomial in the system size, and the quest for the exact ground state must be given up. Instead, one has to resort to more or less dedicated heuristics, to meet the more modest goal of finding states with as low a cost as possible.

Problems that can be formulated in terms of Potts or Ising spins admit a versatile heuristic method, *Deterministic Annealing*, based on the iterative solution of the equations associated with the *mean field* (MF) approximation of the system at hand, combined with a slow decrease in temperature. With the MF variables interpreted as neuron activities, the resulting dynamics at

each temperature is that of a generalized Hopfield (or connectionist) network [3]. Deterministic (MF) annealing has been successfully applied to a range of problem types, see e.g. [12, 2, 4, 7].

The MF approximation is most conveniently derived from a variational approach, where the proper Boltzmann distribution based on the true Hamiltonian is approximated by a factorized distribution, constrained to be the product of individual single-spin distributions, each of which can be parameterized by the corresponding single spin average. The optimal parameters of the approximating distribution minimize an associated free energy.

Thus, the minimization of the cost function in a discrete phase space is replaced by the minimization of an effective cost function in a continuous parameter space, which in suitable coordinates (the spin averages) interpolates between the discrete states of the original phase space. This gives an advantage as compared to a local optimization method confined to the discrete space, due to the possibility of taking shortcuts.

A somewhat different type of optimization problems is given by *assignment* problems, where an optimal matching (assignment) between two sets of objects is desired, as defined by a given cost function. While certain subclasses of assignment problems, like e.g. *linear assignment* where the cost is linear in the assignment matrix, can be solved exactly in polynomial time, the generic assignment problem is a non-polynomial one.

For nonlinear assignment problems, an obvious generalization of the MF-based deterministic annealing approach is lacking, mainly due to the absence of a simple and natural analog of the MF approximation. While a linear cost appears to be the most sensible choice for a variational Ansatz, it does not lead to the simplicity usually associated with the MF approximation. Nevertheless, it is possible to exploit the linear Ansatz to define dedicated deterministic annealing schemes for non-linear assignment, and we will investigate the difficulties and peculiarities involved in connection with this. A major drawback with this approach, however, is that the time required is exponential in the problem size, and so its practical usefulness is limited.

A popular alternative, to avoid the complexity of such an approach, is to tweak MF annealing as defined for Potts systems to make it apply to assignment problems. We will discuss two common methods of this type, Potts-plus-Penalty [11] and SoftAssign [13], point out their strong and weak points, and where appropriate suggest improvements to the existing state of the art.

To illustrate the implementation on a specific problem type, and to gauge the effect of the suggested improvements, a suitable subset of the methods will be

applied to a small testbed of simple applications.

The article is structured as follows: In Sec. II, the basic idea of variational methods in general is described. In Sec. III, MF Annealing for a Potts system is derived from a variational MF approximation, and briefly described. Sec. IV contains a general discussion of assignment problems, and defines some notation. The polynomial problem of Linear Assignment is briefly discussed there. In Sec. V, we discuss the definition of proper deterministic annealing methods dedicated to assignment problems. Sec. VI contains a discussion of existing tweaked Potts-based MF approaches, and suggestions for improvements. In Sec. VII we compare some of the suggested methods on a few simple test problems. Finally, Sec. VIII contains our conclusions.

3.2 MF in General – Variational Approach

The MF approximation, as it is used in MF annealing for binary and Potts systems, is most conveniently derived from a general variational principle.

Given a complicated cost function $H(s)$ of the variables of interest, s, the idea is to approximate its associated Boltzmann distribution $\propto \exp(-H/T)$ (at a fixed artificial temperature T) with one derived from a simpler cost function $H_V(s,\lambda)$ (e.g. a linear one), with a set of free parameters λ (the coefficients in the linear case). The parameter values are then determined by minimization of the associated free energy $F_V(\lambda)$,

$$F_V(\lambda) = \langle H \rangle - TS \equiv -T \log Z + \langle H - H_V \rangle \,, \tag{3.1}$$

where $\langle \cdot \rangle$ stands for an expectation value in the approximating distribution, and Z denotes the corresponding partition function $\sum_s \exp(-H_V(s)/T)$. S is the associated entropy, given by $-\sum_s p(s) \log p(s)$, with $p(s)$ the probability of state s, $p(s) \equiv \exp(-H_V(s)/T)/Z$.

The variational free energy is bounded from below by the true free energy, $F = -T \log(Z_0) = -T \log \sum_s \exp(-H(s)/T)$. The condition for an extremum of F_V with respect to parameter variations $\delta\lambda$ is

$$\delta F_V \equiv \langle \delta H_V \, (H - H_V) \rangle_c = 0, \tag{3.2}$$

where $\langle ab \rangle_c$ stands for the *connected* expectation value (or cumulant) $\langle ab \rangle - \langle a \rangle \langle b \rangle$. Thus, for each parameter λ_a, we must have

$$\left\langle \frac{\partial H_V}{\partial \lambda_a} (H - H_V) \right\rangle_c = 0. \tag{3.3}$$

Although (3.3) may permit multiple solutions, including saddle-points or local minima, it is commonly used in the search for an optimal set of parameter values.

Note that since the expectation values involve the variational Boltzmann distribution $\propto \exp(-H_V/T)$ and hence depend on the temperature T, so will the optimal parameter values.

A particularly simple special case results when the variational cost function depends **linearly** on the parameters,

$$H_V(s;\lambda) \equiv \sum_a \lambda_a E_a(s). \tag{3.4}$$

Then, eq. (3.3) for an extremum takes the simple form

$$\sum_b \langle E_a E_b \rangle_c \lambda_b = \langle E_a H \rangle_c . \tag{3.5}$$

This has the form of a matrix equation, $\langle EE \rangle_c \lambda = \langle EH \rangle_c$, and a straightforward strategy for finding a solution is given by iteratively updating the parameters according to

$$\lambda \to \langle EE \rangle_c^{-1} \langle EH \rangle_c , \tag{3.6}$$

followed by the corresponding updates of the expectation values $\langle EE \rangle_c , \langle EH \rangle_c$, which depend on the parameters via the Boltzmann distribution.

3.3 MF Annealing for Potts systems

In order to understand the problems associated with defining a deterministic annealing approach to assignment-based problems, it is instructive to first review how simpler types of systems are treated.

A simple q-state multiple-choice variable (Potts spin) is conveniently represented by a q-dimensional vector $\mathbf{s}$, with the allowed states represented by the q principal vectors $(1,0,0,\ldots)$, $(0,1,0,\ldots)$, etc., with the position of the single nonvanishing component indicating which state is "on". These vectors are linearly independent, and point to the corners of a regular q-simplex.

As a result, any single-spin cost function can be written as a linear function in $\mathbf{s}$, $H(\mathbf{s}) = \mathbf{C} \cdot \mathbf{s}$, where C_a is the cost associated with state a. For a system

of N Potts spins, it follows that an arbitrary cost function can be written in a *multilinear* form,

$$H(s) = a + \sum_{i,a} b_{ia} s_{ia} + \frac{1}{2} \sum_{i,a} \sum_{j \neq i,b} c_{ia,jb} s_{ia} s_{jb} + \ldots, \tag{3.7}$$

where s_{ia} denotes the a-th component of the i-th spin. Solving the associated optimization problem corresponds to finding the ground state of the system, i.e. the combination of states of the N Potts spins that minimizes $H(s)$.

The MF approximation to such a system results from a variational approach, corresponding to an optimal approximation of the non-linear cost $H(s)$ in terms of a *linear* one,

$$H_V(s) = \sum_i \mathbf{C}_i \cdot \mathbf{s}_i = \sum_{ia} C_{ia} s_{ia}. \tag{3.8}$$

The coefficients C_{ia} constitute the parameters, and are to be chosen so as to minimize the variational free energy F_V. It is convenient to express F_V in terms of the spin averages in the variational distribution, the (*mean field spins*) $\mathbf{v}_i$, with components in $[0, 1]$ amounting to

$$v_{ia}(C) \equiv \langle s_{ia} \rangle_V = \frac{\exp\left(-C_{ia}/T\right)}{\sum_b \exp\left(-C_{ib}/T\right)}. \tag{3.9}$$

In the MF approximation, v_{ia} corresponds to the probability for spin i to be in state a, consistently with the identity $\sum_a v_{ia} = 1$. The MF spins thus interpolate between the discrete states of the original spins; in terms of them the variational free energy evaluates to

$$F_V(v) = T \sum_{ia} v_{ia} \log v_{ia} + H(v), \tag{3.10}$$

which can be minimized with respect to the normalized MF spins by adding a Lagrange parameter λ_i for the normalization of each MF spin $\mathbf{v}_i$. The condition for a extremum, equivalent to eq. (3.3) amounts to

$$dH(v)/dv_{ia} + T(1 + \log v_{ia}) = \lambda_i, \tag{3.11}$$

which, together with the normalization that fixes the λ_i values, gives the variational coefficients C up to an unimportant constant in terms of v as

$$C_{ia}(v) = \frac{\partial H(v)}{\partial v_{ia}}. \tag{3.12}$$

Eqs. (3.9) and (3.12) define the *MF equations*.

The MF approximation corresponds to neglecting the correlations between the different spins, since the linear variational Ansatz used is the most general factorized distribution $P_V(s) = \prod_i p_i(s_i)$, where the different Potts spins obey independent distributions.

MF annealing corresponds to solving the MF equations iteratively, starting with a high T, where a fixed point with $v_{ia} \approx 1/N$ will dominate, and slowly lowering T. At low enough T, the MF spins will be forced *on shell*, i.e. for $v_{ia} \approx s_{ia} \in 0, 1$, and a suggested solution can be extracted.

3.4 Assignment Problems – General Discussion

3.4.1 Notation

When it comes to permutation/assignment problems, we have to distinguish between *single assignment* problems and *multiple assignment* problems, the latter being based on several assignments.

To begin with, we will consider the simpler case of a single assignment, where an optimal matching between two sets of N objects is desired, i.e. for each object i in the first set, an object a in the second set is to be chosen, such that different i are assigned to different a. There are obviously $N!$ ways to accomplish this.

A compact way to describe an assignment is by means of the associated *assignment matrix*, i.e. an $N \times N$ doubly stochastic matrix s with elements in $\{0, 1\}$, such that $s_{ia} = 1$ if i is assigned to a, and 0 otherwise (for a somewhat different encoding of assignment problems as used in deterministic annealing, see [7]). Obviously, we must have precisely one unit element in each row as well as in each column of s, consistently with

$$\sum_a s_{ia} = 1, \ \sum_i s_{ia} = 1. \tag{3.13}$$

Then, a given single assignment problem can be described in terms of a cost function $H(s)$, which is to be minimized.

Note, however, that, in contrast to e.g. the Potts case, the most general cost function for single assignment (for $N > 2$) is **not** linear in s (see Appendix); in the worst case a polynomial of degree $N - 1$ is needed.

Alternatively, the cost function can be viewed as an explicit function over the permutation group P_N: Each group element g is associated with an individual

cost C_g, defining an element of an $N!$-dimensional *cost vector* $\vec{C}$. The relation to the formulation in terms of the assignment matrices s is $C_g = H(s_g)$, where s_g is the particular assignment matrix representing g.

3.4.2 Thermodynamics for a single nonlinear assignment

The thermodynamics af a system consisting of a single $N \times N$ assignment with an arbitrary cost function is not difficult to express, when formulated in terms of a cost vector $\vec{C}$ over group space. The assignment then acts as a single $N!$-state Potts spin, that can be described by an $N!$-dimensional vector $\vec{S}$ with precisely one unit component, while the rest is zero: $S_g \in \{0, 1\}$, $\sum_g S_g = 1$.

At an artificial temperature T, the probability of a particular state g amounts to

$$V_g = \langle S_g \rangle = \frac{\exp\left(-C_g/T\right)}{\sum_h \exp\left(-C_h/T\right)}. \tag{3.14}$$

As $T \to 0$, the distribution gets increasingly concentrated at the state (group element) with the lowest cost.

When viewed this way, the difficulty lies entirely in the huge number of states involved if N is large. In order to compute one component of $\vec{V}$, one has to know the costs for all the $N!$ states. For a generic cost function, the associated computational complexity is non-polynomial in the size N of the system.

Thus, for a generic large assignment problem, one has to make do with some kind of heuristic.

3.4.3 Linear assignment

Certain classes of assignment problems can be solved exactly in polynomial time. One such class is linear assignment, where the cost function is constrained to be linear in the assignment matrix s,

$$H(s) = \sum_{ij} c_{ij} s_{ij}, \tag{3.15}$$

defining an $N \times N$ *cost matrix* c.

This problem corresponds essentially to a linear programming one, and can be solved in polynomial time, using e.g. the so called *Hungarian algorithm* [6, 9], based on the fact that the addition of terms to c depending on row or

column alone, $c_{ij} \to c_{ij} + a_i + b_j$, is equivalent to adding a constant to the cost function, $H(s) \to H(s) + \sum_i a_i + \sum_j b_j$. This is used to iteratively modify the cost matrix until it takes a form where it has zeros on a set of elements corresponding to the optimal assignment, and non-negative values elsewhere.

Unfortunately, this doesn't help when computing thermal averages at a finite T; this is still a non-polynomial task. Thus, e.g., the expectation value of s_{ij} is given in terms of a matrix M, obtained from c by elementwise exponentiation,

$$M_{ij} = \exp(-c_{ij}/T), \tag{3.16}$$

as

$$v_{ij} \equiv \langle s_{ij} \rangle = \frac{M_{ij} P_{ij}(M)}{P(M)}. \tag{3.17}$$

Here, $P(M)$ is the *permanent* [8] of M; it has some similarities to the determinant, being the sum of all the possible products of N elements in M, one in each row and one in each column, but with *no minus signs* in contrast to the case for the determinant. Similarly, $P_{ij}(M)$ is the *subpermanent* of M, obtained by removing row i and column j from M, and computing the permanent of the remaining $(N-1) \times (N-1)$ matrix.

The expression for v_{ij} in eq. (3.17) can be derived from

$$\langle s_{ij} \rangle = -\frac{T}{Z} \frac{\partial Z}{\partial c_{ij}}, \tag{3.18}$$

where Z is the partition function,

$$\begin{aligned} Z &= \sum_g \exp\left(-\sum_{ij} c_{ij} s_{ij}(g)/T \right) \\ &= \sum_g \exp\left(-\sum_{ij \in g} c_{ij}/T \right) \\ &= \sum_g \prod_{ij \in g} \exp\left(-c_{ij}/T\right) = \sum_g \prod_{ij \in g} M_{ij} = P(M), \end{aligned} \tag{3.19}$$

where the restriction $ij \in g$ in the sum over ij means that row i and column j are matched in $s(g)$ (so $s_{ij}(g) = 1$). The derivative of $Z = P(M)$ with respect to M_{ij} yields $P_{ij}(M)$, which completes the proof of eq. (3.17).

The combination $M_{ij} P_{ij}(M)$, appearing in eq. (3.17), gives the sum of those terms in the permanent that contain the element M_{ij}. Summing this over i or j yields $P(M)$, ensuring that eq. (3.17) yields a doubly stochastic matrix v.

Similarly, the expectation value of the product of two elements of s becomes

$$\langle s_{ij} s_{kl} \rangle = \delta_{ik}\delta_{jl} \frac{M_{ij} P_{ij}(M)}{P(M)} + \frac{M_{ij} M_{kl} P_{ik,jl}(M)}{P(M)}, \tag{3.20}$$

where $P_{ik,jl}(M)$ is a subpermanent obtained as the permanent of the submatrix where rows i, k and columns j, l are removed, if $i \neq k$ and $j \neq l$; else it is zero. Thus $M_{ij} M_{kl} P_{ik,jl}(M)$ sums up the terms in $P(M)$ that contain $M_{ij} M_{kl}$.

As an aside, replacing the permanents by determinants in the expression (3.17) for v would lead to the combination $D_{ij}(M)/D(M)$, exactly corresponding to the j, i element of the matrix inverse of M. The elementwise product with M would yield a doubly (quasi-)stochastic v, where row and column sums are equal to one, albeit with elements of both signs.

However, while the determinant of a matrix can be calculated in polynomial time, the permanent in general can not, in spite of their similarity – the computational time required to compute a generic $N \times N$ permanent is exponential in N (roughly $\propto 2^N$ using e.g. Ryser's method [15, 8]).

3.5 Proper Variational Method for a Single Nonlinear Assignment

For a large generic single assignment problem, an exact solution is out of reach, and one has to make do with heuristic methods. One possibility then is to consider a deterministic annealing approach based on approximating the true cost function by a variational one that is simpler.

3.5.1 Linear Ansatz for H_V

The most natural Ansatz for the variational cost function H_V is a linear one,

$$H_V(s) = \sum_{ij} c_{ij} s_{ij}, \tag{3.21}$$

with the coefficients c_{ij} as free parameters.

The equation (3.3) for a minimum of the variational free energy then yields:

$$\sum_{kl} \langle s_{ij} s_{kl} \rangle_c \, c_{kl} = \langle s_{ij} H \rangle_c \, . \tag{3.22}$$

In analogy with eq. (3.6), this is a matrix equation, from which c can be formally extracted as

$$c = \langle ss \rangle_c^{-1} \langle sH \rangle_c \,. \tag{3.23}$$

Note that the $N^2 \times N^2$ matrix $\langle ss \rangle_c$ is not fully invertible; it always has $2N-1$ zero-modes corresponding to the addition of redundant terms to c depending on row or column index alone. These merely yield row and column factors in the exponentiated matrix M, and are of no importance for expectation values.

If H is a low-order polynomial in s, a solution to eq. (3.22) can in principle be computed iteratively in analogy to the iterative solution of the Potts MF eqs. (3.9, 3.12), by repeating the two steps

1. Calculate the expectation values appearing in eq. (3.22); they depend on the present c via M and its permanent (and subpermanents), where $M_{ij} = \exp(-\beta c_{ij})$, in analogy to eqs. (3.17,3.20).
2. Obtain an updated cost matrix c by means of eq. (3.23), suitably regularized with respect to the zero-modes of $\langle ss \rangle_c$. [1]

This can be turned into an annealing approach, by starting with a high T (low β), and decreasing T slightly after every step, until the "MF variables" $v_{ij} \equiv \langle s_{ij} \rangle$ have stabilized sufficiently close to zero or one.

A major drawback of this approach is that it is only feasible if N is not too large, since the computation of expectation values involves the computation of permanents, which requires a time exponential in N.

3.5.2 Quadratic H

The simplest non-linear function of s is quadratic, so assume H to be a given quadratic plus linear function in s,

$$H(s) = \frac{1}{2} \sum_{ijkl} A_{ijkl} s_{ij} s_{kl} + \sum_{ij} B_{ij} s_{ij}, \tag{3.24}$$

involving a symmetric tensor A, $A_{ijkl} = A_{klij}$, which can be assumed to vanish for $i = k$ or $j = l$.

The variational eq. (3.22) corresponding to a linear $H_V(s)$ becomes

$$\sum_{kl} \langle s_{ij} s_{kl} \rangle_c \, c_{kl} = \frac{1}{2} \sum_{klmn} \langle s_{ij} \, (s_{kl} s_{mn}) \rangle_c \, A_{klmn}$$

[1] In cases of instability, the change in c can be decreased by e.g. a factor of 1/2.

$$+\sum_{kl}\langle s_{ij}s_{kl}\rangle_c\left(\sum_{mn}A_{klmn}\langle s_{mn}\rangle+B_{kl}\right). \tag{3.25}$$

Denoting the first term on the RHS of eq. (3.25) by F_{ij}, the effective cost matrix c can be formally extracted as $c = B + A\langle s\rangle + \langle ss\rangle_c^{-1} F$, using a suitably regularized matrix inverse.

3.5.3 Group theoretical aspects

It is instructive to view an assignment problem from a group-theoretical point of view, where the relevant group of course is the permutation group of N elements, denoted by P_N.

Like any functions over group space, H and H_V can be expressed in a unique way as linear combinations of the matrix elements of the irreducible representation matrices of P_N (see Appendix for details).

Requiring H_V to be linear in s means that its expansion is constrained to contain only elements from the trivial and the fundamental irreducible representations, **e** and **f**; thus, it can be written in a unique way in the form

$$H_V(g) = A + \sum_{ij} B_{ij} u^f_{ij}(g), \tag{3.26}$$

where f stands for the fundamental representation. This leads to the probability $V_g \propto \exp(-H_V(g)/T)$, such that $\sum_g V_g = 1$, for a particular group element g. The corresponding version of the variational eq. (3.3) becomes

$$A + \sum_{kl} B_{kl} \sum_g V_g u^f_{kl}(g) = \sum_g V_g H(g), \tag{3.27}$$

$$A\sum_g V_g u^f_{ij}(g) + \sum_{kl} B_{kl} \sum_g V_g u^f_{ij}(g) u^f_{kl}(g) = \sum_g V_g u^f_{ij}(g) H(g), \tag{3.28}$$

corresponding exactly to respectively the trivial and the nontrivial parts of eq. (3.22). The trivial part A can be eliminated from (3.27) and inserted into (3.28), yielding

$$\sum_{kl}\left(\sum_g V_g u^f_{ij}(g)u^f_{kl}(g) - \sum_g V_g u^f_{ij}(g)\sum_h V_h u^f_{kl}(h)\right) B_{kl} = \sum_g V_g u^f_{ij}(g)H(g) - \sum_g V_g u^f_{ij}(g)\sum_h V_h H(h), \tag{3.29}$$

which is nothing but a disguised version of eq. (3.22); the sums over g with V_g as a weight correspond to averages.

For the special case (3.24) of a quadratic $H(s)$, the corresponding $H(g)$ is constrained to include elements from **e**, **f** and two additional representations, **a** and **s** (see Appendix), in its irrep expansion, with dimensions $d_a = (N-1)(N-2)/2$ (if $N > 2$), and $d_s = N(N-3)/2$ (if $N > 3$), respectively.

Although the above analysis illuminates the linear variational approach from a group-theoretical point of view, the resulting formulation is not of an immediate practical interest for large N, since it (at least formally) requires the complete enumeration of the $N!$-dimensional group space.

3.5.4 Proper variational methods for multiple nonlinear assignment

Here we will briefly discuss the possibilities for treating systems of several distinct assignments in a variational approach.

General additive Ansatz

In the case of a generic cost function of several assignments, the most natural choice is to consider a generic *additive* Ansatz for a variational cost function, corresponding to a factorized Boltzmann distribution. In principle, this corresponds to the MF approximation to a system of $N!$-state Potts variables, and can be treated as such. This can be useful, even for large systems (many distinct assignments), as long as the individual assignments are of low dimensionalities.

Linear Ansatz

A further simplification results from requiring that the variational cost function not only be additive in the different assignments, but also that the contribution from each assignment be linear.

A possible strategy then is to update the cost matrix for one assignment at a time according to eq. (3.23), considering the single-assignment averages associated with other assignments as constant, and recomputing the single-assignment averages associated with the updated cost matrix before moving on to update the next assignment.

Multilinear cost, linear Ansatz

A particularly simple special case of the above is when the exact cost function is a *multilinear* function of the assignments matrices s. For the case of two assignments, $s^{(1)}$ and $s^{(2)}$, this means a cost function H of the form

$$\begin{aligned} H(s^{(1)}, s^{(2)}) &= \sum_{ij} \sum_{kl} A_{ij,kl} s_{ij}^{(1)} s_{kl}^{(2)} \\ &+ \sum_{ij} B_{ij}^{(1)} s_{ij}^{(1)} + \sum_{ij} B_{ij}^{(2)} s_{ij}^{(2)}. \end{aligned} \tag{3.30}$$

Then an additive variational cost function automatically will be linear

$$H_V(s^{(1)}, s^{(2)}) = \sum_{ij} C_{ij}^{(1)} s_{ij}^{(1)} + \sum_{ij} C_{ij}^{(2)} s_{ij}^{(2)}. \tag{3.31}$$

The resulting updates then amount simply to

$$\begin{aligned} C_{ij}^{(1)} &= \sum_{kl} A_{ij,kl} \left\langle s_{kl}^{(2)} \right\rangle + B_{ij}^{(1)}, \\ C_{kl}^{(2)} &= \sum_{ij} A_{ij,kl} \left\langle s_{ij}^{(1)} \right\rangle + B_{kl}^{(2)}, \end{aligned} \tag{3.32}$$

where variational expectation values are understood, computable in terms of permanents and sub-permanents of the respective cost matrices.

The generalization to several assignments is straightforward.

3.6 Potts-based MF Heuristics for Nonlinear Assignment

Although a proper variational approach as described above appears to be the natural choice for constructing a deterministic annealing approach for assignment problems, a major problem is the computational complexity involved in the required computation of permanents and subpermanents. Indeed, for certain problem classes an instance can be solved exactly in the time it takes to compute a single permanent. This implies that these methods have a rather limited applicability.

Instead, alternative methods based on Potts spins have been used to construct faster deterministic annealing methods for non-linear assignment problems.

3.6.1 Potts plus Penalty

One such method is based on viewing the assignment matrix s as an array of N-state Potts spins, one for each row. Then the row condition, $\sum_a s_{ia} = 1$, is automatically fulfilled. One then adds to H a penalty term for breaking of the column normalization, and treats the result using Potts MF annealing, based on the modified cost function

$$H(s) + \frac{\alpha}{2} \sum_a \left(1 - \sum_i s_{ia} \right)^2 , \tag{3.33}$$

with the penalty strength α suitable adjusted. (Of course, one might equally well consider the columns as Potts spins and add penalties for the rows.)

This approach, in what follows referred to as **PPP** for Potts plus Penalty, has been successfully applied to a number of different problems [12]. The problem with a soft penalty is that it involves a delicate tuning of the coefficient α; too small, and improper assignments result, where two rows are mapped to the same column; too large, and the cost is too dominated by the penalty term, with a consequent deterioration in performance.

In PPP, the MF spins are preferably updated in a serial manner, one row at a time. This leads to the most stable MF dynamics, provided H is put in multilinear form. It is easy to see then that the Potts free energy (3.10) becomes a Lyapunov function of the dynamics at a fixed temperature. This ensures that the MF dynamics is well-behaved in the high-T domain, with the trivial high-T fixed point losing stability in a controlled manner. It also guarantees that PPP turns into a form of local optimization in the low-T limit, ensuring the stability of an optimal assignment.

3.6.2 SoftAssign

An obviously ugly feature of the PPP approach is the asymmetry in the treatment of rows and columns of s. This can be cured in a slightly more advanced Potts-inspired method, the *SoftAssign* (or "Double Potts") approach [13], which can be derived as a somewhat *ad hoc* improvement to PPP as follows.

In PPP, the resulting MF average is given by $v_{ij} = a_i M_{ij} b_j$, where M_{ij} is given by $\exp(-(\partial H/\partial v_{ij})/T)$ and the column factor b_j comes from the penalty term, $b_j = \exp(\alpha(1 - \sum_i v_{ij})/T)$. In contrast, the row factor is the usual automatic Potts normalization factor, ensuring the exact normalization of rows.

The idea in SoftAssign is to skip the penalty, and freely choose positive row and column factors so as to force the exact normalization of both rows and columns. This leads to the following problem: Given a matrix M with non-negative elements, find vectors of row and column factors, a and b, such that the result,

$$v_{ij} \equiv a_i M_{ij} b_j, \tag{3.34}$$

is a doubly stochastic matrix.

This in fact determines v uniquely, which can be seen by defining $x_i = \log a_i$, $y_j = \log b_j$ (a_i, b_j are assumed positive), and noting that the correct x, y minimize the strictly convex function $f(x,y) \equiv \sum_{ij} e^{x_i} M_{ij} e^{y_j} - \sum_i (x_i + y_i)$.

However, the proper row and column factors can not be obtained as simple, closed expressions in the matrix elements of M. Instead, the desired doubly stochastic matrix v is usually obtained by iteratively modifying M, alternatingly normalizing rows and columns until the resulting matrix is sufficiently close to being correctly normalized:

i) $\quad M_{ij} \to \dfrac{M_{ij}}{\sum_k M_{ik}},$

ii) $\quad M_{ij} \to \dfrac{M_{ij}}{\sum_k M_{kj}},$

iii) $\quad$ Go to **i)**.

This procedure, which is due to Sinkhorn [16], is guaranteed to yield convergence to a unique doubly stochastic matrix v.

For a nonlinear problem, we can obviously identify the derivatives $\partial H / \partial v_{ij}$ with the elements of an estimated *effective cost matrix*, obtained by linearizing the cost-landscape in the vicinity of the present point. Note that, for a *linear* assignment problem, SoftAssign leads to exactly the same initial M as in eq. (3.17); but that a different doubly stochastic matrix v is derived from it in eq. (3.34).

As an aside, the SoftAssign approach can formally be associated with the minimization of an entity reminiscent of a free energy,

$$F(v) = T \sum_{ij} v_{ij} \log v_{ij} + H(v), \tag{3.35}$$

with v constrained to be doubly stochastic, e.g. by means of adding suitable Lagrange terms (with Lagrange multipliers associated with the row and column factors). Although $F(v)$ is superficially highly analogous to the Potts free energy in eq. (3.10), SoftAssign does *not* correspond to a proper variational

approch to approximating the true $H(s)$, mainly because the first term is not $-T$ times the proper entropy. Nor does v correspond to the expectation value of s in an approximating Boltzmann distribution.

Note that for SoftAssign (unlike the case with a proper variational approach), the resulting dynamics is sensitive to the precise formulation of $H(v)$ as an extrapolation of $H(s)$ to continuous arguments v.

Nevertheless, SoftAssign seems theoretically more appealing than PPP, in treating rows and columns in a symmetric manner, and in guaranteeing a doubly stochastic v, and it has also been successfully applied to various types of problems [2], although the method does have some weak points as will be discussed below.

Weak points with SoftAssign

In the SoftAssign approach, the iterative Sinkhorn procedure for normalizing v is problematic at low T. This can be seen by assuming one has reached a stage where the matrix is close to being doubly stochastic:

$$M_{ij} = (1 + \alpha_i) v_{ij} (1 + \beta_j), \tag{3.36}$$

where v is the desired doubly stochastic matrix, while α_i, β_j are assumed small. To linear order in α, β, one step of row normalization corresponds to $\alpha \to -v\beta$ in matrix notation, while one step of column normalization yields $\beta \to -v^\top \alpha$. Together, this gives $\beta \to v^\top v \beta$.

Hence, the asymptotic rate of convergence is determined by the eigenvalues of the positive-definite matrix $U = v^\top v$. At a high temperature, U will be close to a uniform matrix with $1/N$ everywhere, while at a low T, it will be close to the identity matrix. Consider a simple Ansatz for U: $U_{ij} = (1 - a)\delta_{ij} + a/N$, with $0 \leq a \leq 1$. The limit $a \to 1$ corresponds to high T, while $a \to 0$ emulates low T. U can also be written as $U = (1 - a)\mathbf{1} + aP$, where $\mathbf{1}$ is the identity matrix, and P is the projection matrix onto the uniform vector.

U has two distinct eigenvalues: a single unit eigenvalue with a uniform eigenvector $(1, 1, \ldots, 1)$, and an $(N-1)$-fold degenerate value $1-a$ with eigenvectors in the transverse space. The unit eigenvalue is to be expected, reflecting the irrelevance of shuffling a global factor between a and b.

What is worse is that as $T \to 0$ ($a \to 0$), also the other eigenvalue, $1 - a$, tends towards unity. This is not a special feature of this particular Ansatz for v, but a generic phenomenon: As v approaches a proper assignment matrix, $v^\top v$

approaches the identity matrix. This means that the normalization procedure inevitably runs into convergence problems in the limit of low T.

Another drawback as compared to PPP is that in SoftAssign, the elements of the assignment matrix v have to be updated in *synchrony*, and there is no obvious simple way to update it, say, one row at a time. As a result, stability is not guaranteed at low T even for a good solution, unless the cost function H is carefully tuned.

Speeding up the iterative normalization

In order to improve convergence for the normalization procedure in SoftAssign at low T, it appears important to initialize the multiplicative row and column vectors a and b carefully, so as to leave as little as possible for the slow iterative procedure to do.

Obviously, to avoid overflow on the computer upon implementation of eq. (3.16), the effective cost matrix will have to be modified with suitable row and column additions, $c_{ij} \to c_{ij} + \alpha_i + \beta_j$, to ensure that the smallest elements be zero and the rest positive. This can be done, e.g., by first subtracting from the elements in each row the lowest element in that row, and then subtracting from the elements in each column the lowest element in that column.

This measure does not suffice, however, to guarantee a proper starting point. Consider $N = 3$ and the cost matrix $c = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$. At zero T this yields $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$, and the normalization procedure will be caught in an eternal loop, alternating forever between the two states $\begin{pmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \end{pmatrix}$. In fact, this M is not normalizable with finite row and column factors!

Obviously, it is not enough to ensure at least one zero in each row and each column of c. The zeros must be arranged such that at least one combination of them will correspond to a one-to-one matching between rows and columns. Finding such a modification with otherwise non-negative elements corresponds precisely to solving the associated (effective) linear assignment problem.

Thus, we suggest using e.g. the Hungarian method to transform c to the proper form, in order to guarantee a normalizable M for $T \to 0$.

Even this step will not guarantee a fast convergence. As a second example, consider $N = 2$ and an effective cost matrix $c = \begin{pmatrix} 0 & 0 \\ 2 & 0 \end{pmatrix}$, obviously in the

proper form. The corresponding M at $T = 0$ will be $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$. If this is handed to the normalization procedure at a low T, the approach to the final doubly stochastic matrix, $v = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, will be merely harmonic.

The situation is considerably improved, by in addition carefully *balancing* the cost matrix c, to ensure also a maximal number of non-zero elements (while maintaining a sufficient set of zeros), with the smallest of these being as large as possible. This can be done in polynomial time, using a recursive procedure. For the $N = 2$ case above, the balancing step will yield $c = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, giving $M = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ at zero T, and convergence is immediate.

Still, there will be cases for $N > 2$ where the procedure will be caught in a slowly converging sequence – this is inevitable, due to the unit eigenvalues at zero T –, but in this way the most obvious traps are avoided.

An additional improvement is possible when using the Hungarian algorithm for preprocessing the cost matrix, by exploiting that it gives a preferred matching. This can be used to modify the normalization process to improve the convergence rate by updating of matched row-column pairs simultaneously. Especially in the low T region, where the matched elements unambiguously define a selected assignment, and the corresponding elements of v will be close to unity and the rest small, this noticeably speeds up the normalization.

Thus, the normalization constraints for a coupled row-column pair i, j of a modified M read

$$\begin{aligned} a_i \left(\sum_{k \neq j} M_{ik} + M_{ij} b_j \right) &= 1, \\ b_j \left(\sum_{k \neq i} M_{kj} + a_i M_{ij} \right) &= 1, \end{aligned} \tag{3.37}$$

and are simultaneously satisfied by $a_i = x/A_i$ and $b_j = x/B_j$, where $A_i = \sum_{k \neq j} M_{ik}$, $B_j = \sum_{k \neq i} M_{kj}$ and

$$x = \frac{\sqrt{A_i B_j \left(4 M_{ij} + A_i B_j\right)} - A_i B_j}{2 M_{ij}}. \tag{3.38}$$

Each matched row-column pair of M in turn is updated in this manner and the process is repeated until the result is close to being a doubly stochastic matrix. We will refer to this normalization scheme as *coupled normalization*.

Ensuring a stable dynamics

The second major problem with SoftAssign is the lack of a guarantee for the stability of a solution in the low-T limit; this problem never appears e.g. in a properly handled system of Potts MF spins with serial updating.

To remedy this, we will have to exploit the freedom of adding terms to H which are trivial on shell, but nevertheless affect the MF dynamics. One would at least like to ensure that in the low-T limit, SoftAssign turns into some form of local optimization.

One possibility then is to demand that $H(v)$ be transformed into a *concave* function of v (in the subspace consistent with v being doubly stochastic). This guarantees that the energy in eq. (3.35) for $T \to 0$ will have a local minimum in the corner corresponding to an optimal assignment.

A crude way to ensure concavity is to add a negative quadratic diagonal term $-(\alpha/2)\sum_{ij} s_{ij}^2$ to H, with a large enough coefficient α. For a quadratic H in particular, concavity also ensures that the SoftAssign free energy (3.35) becomes a Lyapunov function of the dynamics at a fixed temperature [14]. A disadvantage with this method is that also non-solutions will be stabilized. Empirically, a smaller diagonal addition often suffices to stabilize the dynamics.

A more advanced possibility is to employ problem-specific modifications of H, adding suitable terms to ensure the stability of a good solution. It is therefore of interest to analyze what kinds of generic additions are possible without altering the on-shell costs (or merely adding a constant). We will refer to such terms as being **redundant**.

Linear redundant terms: Based on decomposing the defining representation of the permutation group P_N, given by s, into the direct sum of the trivial 1-dimensional irrep $\mathbf{e}$ and the fundamental $(N-1)$-dimensional one $\mathbf{f}$ (see Appendix), the only possible linear redundant additions are given by combinations of row or column sums of s; in SoftAssign, such additions merely lead to a modification of the initial row or column factors a, b in eq. (3.34), and have no effect on the resulting doubly stochastic matrix v.

Combinations of quadratic and linear terms: The possibilities here stem from the decomposition of the reducible representation of P_N defined by the symmetric direct product of the defining representation ($\Leftrightarrow \mathbf{e} + \mathbf{f}$) with itself, as discussed in the Appendix. The contributions stemming from the trivial direct product of the $\mathbf{e}$ part with itself or with $\mathbf{f}$ give nothing useful, corresponding to quadratic terms involving row or column sums of s, completely equivalent to the corresponding linear terms with the row or column sums replaced by 1.

The possibly interesting terms come from $\mathbf{f}\times\mathbf{f} = \mathbf{e}+\mathbf{f}+\mathbf{s}+\mathbf{a}$. As discussed in the Appendix, the symmetric part of this consists of a part $\mathbf{a}$, antisymmetric in both row and column indices, yielding no redundant terms, and a part containing $\mathbf{e}+\mathbf{f}+\mathbf{s}$, symmetric in both row and column indices, which yields quadratic terms that vanish on shell, or equal a constant, or a linear combination of the elements of s. In a slighly disguised form, they correspond to the on-shell identities

$$\begin{aligned} s_{ij}^2 - s_{ij} &= 0, \\ s_{ij}s_{il} &= 0, \quad \text{for } j \neq l, \\ s_{ij}s_{kj} &= 0, \quad \text{for } i \neq k. \end{aligned} \tag{3.39}$$

Group theory certifies that these identities suffice to generate all possibly useful redundant additions to $H(s)$, that are at most quadratic in s. Although such additions to H are identically zero on shell ($v \to s$), as additions to $H(v)$ they will alter the properties of the dynamics in SoftAssign, in terms of a modification of the expression for the effective cost matrix $c = \partial H(v)/\partial v$.

Thus, the addition of a term proportional to the square of an element v_{ij} minus the element itself, modifies only the corresponding element of c, by an addition proportional to $2v_{ij} - 1$. Adding the product of two elements of s in the same row i but in different columns j, l affects the corresponding two elements of the effective cost matrix, with an addition to c_{ij} proportional to v_{il}, and vice versa. Analogously, adding to H a term involving the product of two elements in the same column j but in different rows i, k yields the addition to c_{ij} of a term proportional to v_{kj} and vice versa.

TSP-specific modifications

In certain quadratic problems, such as the *travelling salesman problem* (TSP), where a set of N *sites* is to be cyclically visited in an optimal order, the cost function has the particular structure

$$H(s) = \frac{1}{2}\mathrm{Tr}\left(sDs^\top X\right), \tag{3.40}$$

with D, X a pair of symmetric $N \times N$ matrices, vanishing on the diagonal.

For TSP, D is the pair-distance matrix, with D_{ab} defining the distance between sites a, b, while X defines the cyclic tour sequence neighborhood, $X_{ij} = \delta_{i,j+1} + \delta_{i,j-1}$, such that H measures the total *tour length*.

Concavity in the subspace consistent with s being doubly stochastic, of the direct product matrix $A = D \times X$, then corresponds to one of D or X being

positive-semidefinite, and the other negative-semidefinite, each in the subspace orthogonal to $e = (1, 1, 1 \ldots 1)$. This can be ensured by suitable diagonal additions to D and X separately,

$$D \to D + \alpha \mathbf{1},\ X \to X + \gamma \mathbf{1}, \tag{3.41}$$

with α and γ of opposite signs. The diagonal additions to D and X implies adding terms to H of the form discussed in the previous subsection. All vanish on shell except for the $\alpha \times \gamma$ term, which evaluates to a simple constant.

For the case of TSP, often D is already negative-definite in the transverse subspace, and a suitable addition to X suffices. X is easily diagonalized by means of a discrete Fourier transform, with the spectrum given by $\lambda_k = 2\cos(2\pi k/N)$, for $k = 1, \ldots, N-1$. Thus, $\gamma = 2$ is required to make the modified X positive-semidefinite. In practice, however, $\gamma = 1$ will suffice to stabilize the dynamics in most cases; in the low-T limit this is just enough to secure the stability of assignments locally optimal with respect to local changes in the ordering of visited sites.

3.7 Tests on Simple Applications

In order to illustrate the ideas discussed in the previous section, we will here test the various improvements to the SoftAssign algorithm, as applied to a set of small single-assignment problems.

The effects of the improvements to the normalization algorithm at low temperatures are illustrated using the linear assignment problem. For TSP, the use of a problem-specific stabilizing term is compared to employing a negative quadratic term $-(\alpha/2)\sum_{ij} s_{ij}^2$.

The SoftAssign algorithm used is described in Fig. 3.1.[2] All experiments have been performed on an 800MHz PentiumIII computer running Linux.

3.7.1 Speeding up the iterative normalization

As discussed in section 3.6.2, the Sinkhorn normalization of M, eq. (3.35), runs into trouble when the temperature is low and the corresponding v is close to an on-shell assignment matrix.

[2] For a more thorough description of SoftAssign in general, we refer to [13]

- Initiate the elements of v to random values close to $1/N$, and T to a high value.
- Repeat the following (a sweep), until the v matrix has *saturated* (i.e. become close to a (0,1)-matrix):
  - Calculate the effective cost matrix by means of $c_{ij} = \partial H/\partial v_{ij}$, possibly modify it with suitable row/column additions, and let $M_{ij} = exp(-c_{ij}/T)$.
  - Normalize M with the proper row and column factors to yield a doubly stochastic matrix, defining an updated v.
  - Decrease T slightly (typically by a few percent).
- Extract the resulting solution candidate.

Figure 3.1: A SoftAssign algorithm

To probe the efficiency of each normalization scheme, we use random linear assignment, eq. (3.15), where the costs $c_{ij} \in [0, 1]$ are uniform random numbers. We investigate the number of iteration steps needed before row and column sums of the modified M matrix are in the range $1 \pm \Delta_{max}$ with Δ_{max} a small number, and measure the time used by the normalization scheme. This is done at a set of decreasing temperatures such that $v_{ij} \approx 1/N, \forall i, j$ for the higher values of T, while v is nearly on shell for the lower T values.

We compare the following schemes described in section 3.6.2.

1. Plain Sinkhorn. Preprocess c by first for each row subtracting the smallest element, and then doing the same for each column. Then the Sinkhorn row and column normalization (eq. (3.35)) is applied on the resulting M.
2. Hungarian+Sinkhorn. Preprocess c using the Hungarian algorithm. Then normalize M using Sinkhorn.
3. Hungarian+Balancing+Sinkhorn. Preprocess c using the Hungarian and the balancing algorithms. Sinkhorn normalization of M.
4. Hungarian+CoupledNorm. Preprocess c as in 2, then apply the coupled row-column normalization described in section 3.6.2.
5. Hungarian+Balancing+CoupledNorm. Same preprocessing of c as in 3, then coupled row-column normalization.

Figures 3.2 and 3.3 show statistics from 100 linear assignment problems of size $N = 100$. The data is binned for different values of the *saturation*, $\Sigma = (1/N)\sum_{ij} v_{ij}^2$, representing different temperature regions. At high temperatures the saturation is close to $1/N$, while it approaches one in the low temperature region. The annealing is continued until the saturation becomes larger than 0.999, but is also aborted when the number of normalization steps exceeds a maximal value of 20000 iteration steps for three consecutive temperatures (which only happened for the plain Sinkhorn approach as discussed below); these data points are not included when calculating the averages in Figs. 3.2 and 3.3.

The results illustrate the efficiency of the different normalization methods used on M, and the effect of preprocessing of the cost matrix c. As can be seen in Fig.

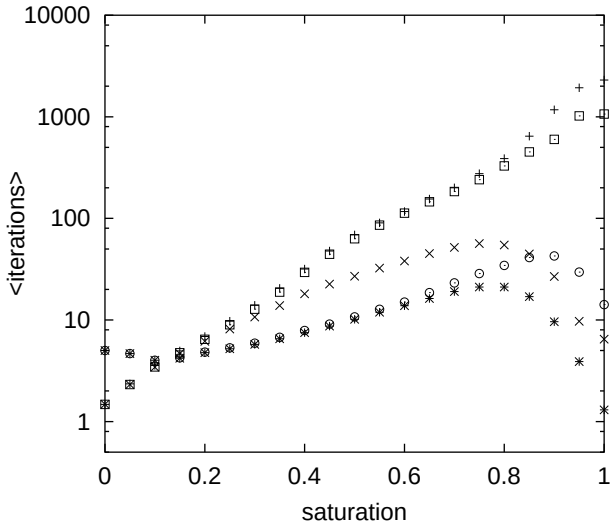

Figure 3.2: Number of normalization iterations used versus saturation. The data consist of averages from runs on 100 random linear assignment problems, binned into different values of the saturation. The normalization procedure is continued until all row and column sums are within $1 \pm \Delta_{max}$ with $\Delta_{max} = 0.01$. The plot shows Plain Sinkhorn (+), Hungarian+Sinkhorn ($\Box$), Hungarian+CoupledNorm ($\odot$), Hungarian+Balancing+Sinkhorn ($\times$), and Hungarian+Balancing+CoupledNorm (*).

3.2 the number of iterations needed with plain Sinkhorn grows by several orders of magnitude in the low temperature region, where the saturation approaches

1. What is not revealed in the figure is that the plain Sinkhorn scheme failed to normalize M at low temperatures in *all* of the tested problem instances – because of the limited resolution on the computer, small elements in M become zero where the corresponding elements in v should be one. The Sinkhorn scheme then gets stuck in an eternal loop, failing to produce a doubly stochastic v. The failure occurred at high enough temperatures that the saturation was below our limit of 0.999, and the algorithm had to be aborted as described above.

This problem can be avoided by either interrupting the annealing at an earlier stage, and extracting a solution from the unsaturated v matrix, or by adding additional redundant terms to the Hamiltonian, at the cost of a lower performance. However, to guarantee that an arbitrary cost matrix yields a doubly stochastic v, preprocessing of the cost matrix is essential.

Using a Hungarian preprocessor on the cost matrix ensures that the initial M has element values equal to one on a permutation, and values between zero and one on the other elements. This guarantees that at least the selected permutation survives in the normalizing process, and a doubly stochastic matrix v will always result. Though this sounds appealing, our tests reveal that this does not substantially decrease the number of Sinkhorn iterations as compared to the plain Sinkhorn approach, as seen in Fig. 3.2.

To avoid the extreme increase of the number of normalization iterations in the low temperature region one can apply balancing of the cost matrix c, or use the coupled normalization approach, both described in section 3.6.2. Applying either one (or both) of the methods decreases the number of normalization iterations. This is evident in Fig. 3.2, especially in the low temperature region where the saturation approaches one.

Applying the Hungarian and balancing methods to the cost matrix c leads to an increased time used to produce an initial M, which is revealed in Fig. 3.3. The total time for the algorithm is dominated by the time spent on Hungarian and balancing. This time is nevertheless far exceeded by the time required with the plain Sinkhorn approach in the low temperature region.

The Hungarian method is known to scale in time with problem size as $O(N^3)$ [9], and the balancing routine empirically appears to behave similarly. This is to be compared to the time it takes to calculate the effective cost matrix, which e.g. for TSP is also an $O(N^3)$ procedure, while for generic quadratic assignment it scales as $O(N^4)$.

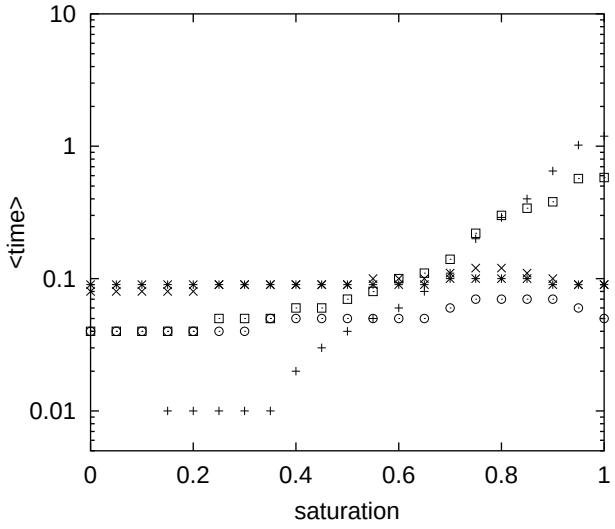

Figure 3.3: Time used for normalization, including preprocessing of the cost matrix, versus saturation. The data consists of averages from runs on 100 random linear assignment problems, binned into different values of the saturation. The normalization procedure is continued until all row and column sums are within $1 \pm \Delta_{max}$ with $\Delta_{max} = 0.01$. The plot shows Plain Sinkhorn (+), Hungarian+Sinkhorn (□), Hungarian+CoupledNorm (⊙), Hungarian+Balancing+Sinkhorn (×), and Hungarian+Balancing+CoupledNorm (*). All times are measured in seconds.

3.7.2 Stable dynamics in TSP

One of the most studied combinatorial optimization problems is TSP. Deterministic annealing has been applied to it using both PPP and SoftAssign [10, 2] and we refer to these articles for a more thorough description of the implementation on TSP. Here we will use TSP as an example where the choice of stabilizing term needed by SoftAssign indeed influences the performance.

The standard assignment-matrix Hamiltonian for TSP is given in equation eq. (3.40). In addition to this an extra stabilizing term is needed. We have compared the addition of a *generic stabilizer* in the form of a diagonal quadratic term,

$$H_A = -\frac{\alpha}{2} \sum_{ia} s_{ia}^2, \tag{3.42}$$

as proposed in the literature [2], with a *problem-specific stabilizer* as discussed in section 3.6.2($X \to X + \gamma \mathbf{1}$),

$$H_B = \frac{\gamma}{2} \sum_{iab} s_{ia} s_{ib} D_{ab}. \tag{3.43}$$

Throughout our experiments we have used the values 1.0 for both α and γ. The α value is slightly smaller than the value 1.4 used in [2]. A γ value of 1.0 ensures the stability with respect to local changes of the ordering of visited sites as discussed in section 3.6.2, but does not always suffice to produce a proper assignment. When this happens (in about 1% of the tested problems) the algorithm is restarted, initialized with a new v. Typically, one restart is sufficient to find a proper assignment matrix.

We studied random TSP problems where the sites were uniformly generated in the two-dimensional unit square. In Fig. 3.4 the tour lengths from 500 problems of size 100 are shown.

The generic stabilizer (eq. (3.42)) works by enhancing already large spin elements. Due to this, v saturates faster (towards an assignment matrix) in the course of the annealing. Since this effect is not as pronounced with the problem-specific stabilizer (eq. (3.43)), equal annealing parameters will not lead to equal time used by the algorithms. Instead, parameters are chosen such that the respective performances are not too far from optimal and the times used are comparable. We have used a slower annealing, $T \to T/1.01$, for the generic stabilizer, and also allowed it to use up to 5 sweeps per temperature if the maximal change in a spin components is larger than 0.01. An even slower annealing would not lead to a considerably better performance, in spite of the increase in time. With the problem-specific stabilizer, we have used the annealing rate $T \to T/1.05$, with one sweep per temperature.

With the generic stabilizer, the average tour length was 9.53 and the average time used 1.53 s. With the problem-specific one, corresponding values where 8.39 and 3.74 s (including possible restarts). Thus, performance-wise, the problem-specific stabilizer is superior, while the increase in time used can be attributed to the slower saturation – this might be avoided by adding a small generic stabilizer term.

3.8 Conclusions

We have investigated the possibilities of defining a deterministic annealing approach to nonlinear assignment problems, in anology to existing algorithms for

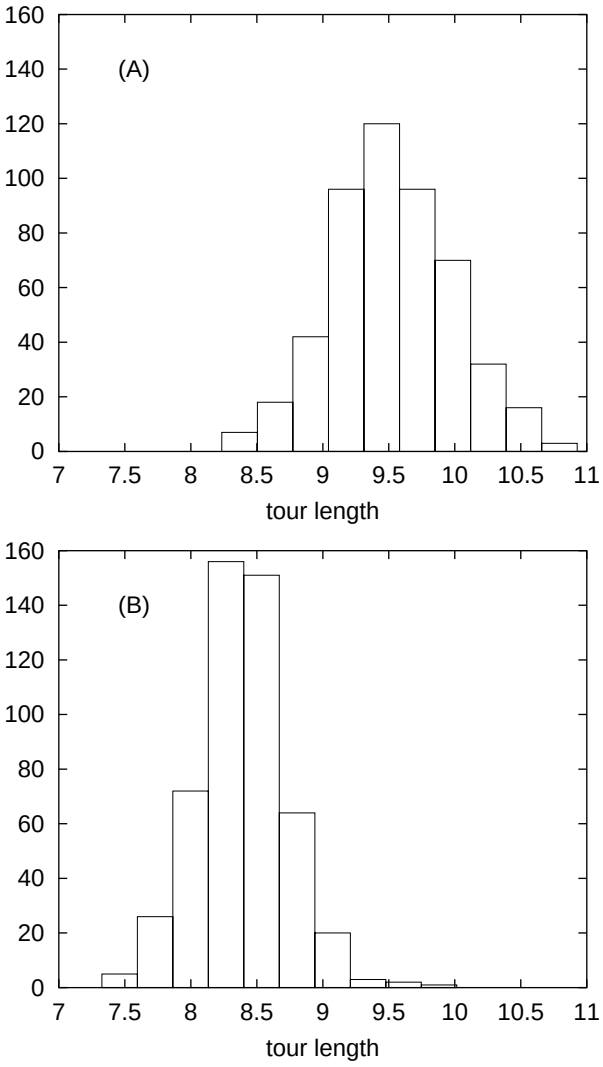

Figure 3.4: Tour lengths for 500 random two-dimensional Euclidean TSP problems of size 100, using (**A**) the generic stabilizer (3.42) with $\alpha = 1.0$, (**B**) the problem-specific stabilizer (3.43) with $\gamma = 1.0$.

Ising and Potts systems.

We have analyzed a proper variational approach, where the problem cost function is approximated by a variational cost, linear in the assignment matrices. For a single assignment problem this allows for an iterative scheme to minimize the variational free energy at a given temperature. Combined with annealing, this can be used as a deterministic annealing algorithm.

As an aside, the generalization to multiple assignment problems is straightforward. Assuming additive linear contributions to the variational cost from the different assignments leads to a mean-field-like approximation with a factorized Boltzmann distribution, and the variational parameters for the individual assignments can be updated in a serial manner.

A major problem with the proper variational approach, however, is that it requires the calculation of permanents, which needs exponential time (in problem size). This implies that considering this as a general heuristic for large nonlinear assignment problems is not feasible.

Abandoning the quest for a proper variational method for nonlinear assignment, we have also studied Potts-based methods as a more promising alternative, although *per se* not tailored for assignment problems. The currently most appealing method of this type is the SoftAssign algorithm, and we have proposed some improvements to it.

The Sinkhorn normalization procedure used in SoftAssign runs into convergence problems at low temperatures. We present arguments why this is unavoidable, and propose proper adjustments of the effective cost matrix to reduce the effect. The application of a Hungarian preprocessing to the effective cost matrix guarantees that the Sinkhorn procedure always produces a doubly stochastic matrix. An additional balancing of the cost matrix decreases the number of iteration needed by the normalization. In addition we devise an alternative normalization procedure which is easily implemented when a Hungarian preprocessing is used. It is superior to the Sinkhorn procedure at low temperatures. We have experimentally confirmed these statements by implementation of SoftAssign on random instances of linear assignment. With other problem ensembles, however, we have experienced varying effects of the improvements, in some cases they appear essential, while in others they are more or less superfluous.

Another problem with the SoftAssign approach is the lack of guarantee for stability in the low temperature region: Solutions may not be stable. This problem can be resolved by adding to the Hamiltonian a stabilizer – a redundant term that affects the dynamics without altering the on-shell cost. We have used arguments from group theory to determine the possible types of redundant additions that are at most quadratic in the spin components. As an example

we discuss how such redundant terms can be used for the travelling salesman problem, and propose a TSP-specific stabilizing term different from the generic one normally used with SoftAssign. In numerical experiments we show that this enhances the performance.

Acknowledgements

This work was in part supported by the Swedish Foundation for Strategic Research.

References

[1] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[2] S. Gold and A. Rangarajan. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4):381–399, 1996.

[3] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[4] H. Jönsson and B. Söderberg. An information-based neural approach to constraint satisfaction. *Neural Computation*, 13:1827–1838, 2001.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[7] M. Lagerholm, C. Peterson, and B. Söderberg. Airline crew scheduling using Potts mean field techniques. *European Journal of Operations Research*, 120:81–96, 2000.

[8] H. Minc. *Permanents*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1978.

[9] C. H. Papadimitriou. *Combinatorial Optimization*. Dover Publications, Inc., Mineola, New York, 1998.

[10] C. Peterson. Parallel distributed approaches to combinatorial optimization - benchmark studies on travelling salesman problem. *Neural Computation*, 2(3):261–269, 1990.

[11] C. Peterson and B. Söderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1:3–22, 1989.

[12] C. Peterson and B. Söderberg. Neural optimization. In M. A. Arbib, editor, *The Handbook of Brain Research and Neural Networks, (2nd edition)*, pages 617–622. Bradford Books/The MIT Press, Cambridge, Massachusetts, 1998.

[13] A. Rangarajan, S. Gold, and E. Mjolsness. A novel optimizing network architecture with applications. *Neural Computation*, 8(5):1041–1060, 1996.

[14] Anand Rangarajan, Alan Yuille, Steven Gold, and Eric Mjolsness. A convergence proof for the softassign quadratic assignment algorithm. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 620–626. The MIT Press, 1997.

[15] H. J. Ryser. *Combinatorial Mathematics*. Mathematical Association of America, Washington DC, 1963.

[16] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35:876–879, 1964.

3.A Basic Group Theory for P_N

Here we will briefly review elements of the basic group theory for the permutation group of N elements, $G \equiv P_N$.

3.A.1 Representations and irreps

G has a finite number of inequivalent irreducible representations, or irreps; the squares of their dimensions sum up to the size V of the group, $V \equiv N!$. If r labels an irrep, let d_r be its dimension, and let the associated $d_r \times d_r$ matrices be denoted $u^r(g)$. We have $\sum_r d_r^2 = V$.

A D-dimensional reducible representation $\{U(g)\}$ can be decomposed into the direct sum of (not necessarily distinct) irreps $\{r_\mu\}$ as

$$U_{ij}(g) = \sum_\mu \sum_{k,l} P^\mu_{ik} P^\mu_{jl} u^{r_\mu}_{kl}(g), \tag{3.44}$$

or, in matrix form, $U(g) = \sum_\mu P^\mu u^{r_\mu}(g) P^{\mu\top}$, where P^μ is a (g-independent) $D \times d_{r_\mu}$ matrix that projects out the part of a vector that belongs to the associated irrep $r = r_\mu$. It can be seen as a submatrix of an orthogonal $D \times D$ matrix V, used to similarity transform U to an explicitly blocked form U_B, $U(g) = V U_B(g) V^\top$.

The ortogonality of V implies the following properties for the matrices P^μ:

$$\sum_\mu P^\mu P^{\mu\top} = \mathbf{1}_D, \tag{3.45}$$

$$P^{\mu\top} P^\nu = \delta_{\mu\nu} \mathbf{1}_{d_{r_\mu}}, \tag{3.46}$$

where $\mathbf{1}_d$ denotes the $d \times d$ identity matrix. Inverting the similarity transform, eq. (3.44) is equivalent to

$$\sum_{ij} P^\mu_{ik} P^\nu_{jl} U_{ij}(g) = \delta_{\mu\nu} u^{r_\mu}_{kl}(g), \tag{3.47}$$

or, in matrix form, $P^{\mu\top} U(g) P^\nu = \delta_{\mu\nu} u^{r_\mu}(g)$, expressing the similarity transform of U to blocked form, with μ, ν labeling respectively the row and column block.

The assignment matrices s define a particular N-dimensional representation (the defining representation) of P_N. It is *reducible* if $N > 1$, being the direct

sum $\mathbf{e}+\mathbf{f}$ of two irreps, where $\mathbf{e}$ is the trivial one-dimensional irrep with $u^{\mathbf{e}} \equiv 1$, while $\mathbf{f}$ is a non-trivial $(N-1)$-dimensional irrep, the *fundamental* representation. For the $\mathbf{e}$ part, e.g., the corresponding $N \times 1$-dimensional projection matrix is given by $P^{\mathbf{e}}_{ik} = 1/\sqrt{N}$.

3.A.2 Irrep Expansion

Due to the identity $\sum_r d_r^2 = V$, there are as many distinct matrix elements in the inequivalent irreps as there are elements in the group. As is well known, these elements form a complete orthogonal basis in group space, as expressed by

$$\sum_{g \in G} u^r_{ij}(g) u^s_{kl}(g) = \frac{V}{d_r} \delta_{r,s} \delta_{i,k} \delta_{j,l} \qquad \text{(orthogonality)}, \tag{3.48}$$

$$\sum_r \sum_{i,j=1}^{d_r} d_r u^r_{ij}(g) u^r_{ij}(h) = V \delta_{g,h} \qquad \text{(completeness)}. \tag{3.49}$$

Thus, any function F over the group can be expressed in a unique way as a linear combination of the irrep elements,

$$F(g) = \sum_r \sum_{ij} C^r_{ij} u^r_{ij}(g), \tag{3.50}$$

where C are coefficients, and $u^r(g)$ is the orthogonal matrix representing the group element g in the irrep r. Due to the completeness, eq. (3.50) can be inverted to yield the coefficients uniquely as

$$C^r_{ij} = \frac{d_r}{N!} \sum_g u^r_{ij}(g) F(g). \tag{3.51}$$

Linear expressions in s

eq. (3.47) can be interpreted as follows: Independently of the group element g, certain linear combinations of the elements of the matrix $U(g)$ representing g in a reducible representation R, will be identical to the elements of the orthogonal matrix $u^r(g)$ corresponding to an irrep r that appears in R; certain other linear combinations (corresponding to $\mu \neq \nu$) will vanish. Together, these span a complete basis in the space of all possible linear combinations. This can be used to identify the redundant linear or quadratic expressions in the assignment matrix s.

A *linear* function of the assignment matrix s can have non-vanishing coefficients only for $r = \mathbf{e}$ and $r = \mathbf{f}$ in its expansion (3.50).[3] Separating the $\mathbf{e}$ and $\mathbf{f}$ parts of s yields

$$s_{ij} = 1/N + \sum_{kl} P^{\mathbf{f}}_{ik} u^{\mathbf{f}}_{kl} P^{\mathbf{f}}_{jl}. \tag{3.52}$$

The different versions of eq. (3.47) then yield a set of identities,

$$\sum_{ij} s_{ij}(g) = N, \tag{3.53}$$

$$\sum_{ij} s_{ij}(g) P^{\mathbf{f}}_{jl} = 0, \tag{3.54}$$

$$\sum_{ij} P^{\mathbf{f}}_{ik} s_{ij}(g) = 0, \tag{3.55}$$

$$\sum_{ij} P^{\mathbf{f}}_{ik} s_{ij}(g) P^{\mathbf{f}}_{jl} = u^{\mathbf{f}}_{kl}(g). \tag{3.56}$$

The first three of these express in a slightly disguised form the constraints of unit row and column sums in s.

Quadratic expressions in s.

A quadratic expression in s means a linear combination of products $U_{ik,jl} \equiv s_{ij} s_{kl}$, defining the direct product representation $(\mathbf{e}+\mathbf{f}) \times (\mathbf{e}+\mathbf{f})$, considering the pair of row indices $\{i,k\}$ as a composite row index, and the pair of column indices $\{j,l\}$ likewise as a composite column index.

It is reducible, and the corresponding version of eq. (3.47) reads

$$\sum_{ik,jl} Q^{\mu}_{ik,m} s_{ij}(g) s_{kl}(g) Q^{\nu}_{jl,n} = \delta_{\mu\nu} u^{r_\mu}_{mn}(g). \tag{3.57}$$

The non-trivial part comes from $\mathbf{f} \times \mathbf{f}$, which reduces to $\mathbf{e} + \mathbf{f} + \mathbf{s} + \mathbf{a}$ (for $N > 2$), where $\mathbf{s}$ and $\mathbf{a}$ are two new irreps, of dimensionalities $d_{\mathbf{s}} = N(N-3)/2$ and $d_{\mathbf{a}} = (N-1)(N-2)/2$.

Due to the obvious symmetry of U, $U_{ik,jl} \equiv U_{ki,lj}$, it is natural to divide the resulting irreps in two sets, according to the symmetry with respect to swapping the index pair ik in $Q^{\mu}_{ik,m}$. Thus, the *symmetric* part of $\mathbf{f} \times \mathbf{f}$ contains $\mathbf{e}+\mathbf{f}+\mathbf{s}$, while the *antisymmetric* part contains $\mathbf{a}$ alone.

[3] Which shows that the most general cost function is not linear in s.

With opposite symmetry type in the row and column parts, the LHS of eq. (3.57) vanishes identically. Thus, the interesting parts require μ, ν to correspond to the same type of symmetry. The antisymmetric part contains only the non-trivial part $\mu = \nu = \mathbf{a}$, yielding $u^{\mathbf{a}}$. In the symmetric part, the $\mu = \nu = \mathbf{s}$ part similarly yields $u^{\mathbf{s}}$, while the remaining combinations yield products of elements of s for which the RHS will either vanish ($\mu \neq \nu$), or yield a constant ($\mu = \nu = \mathbf{e}$), or a linear combination of the elements of s ($\mu = \nu = \mathbf{f}$); the LHS then defines candidates for redundant quadratic additions to a cost function.

An Approximated Maximum Likelihood Approach, applied to Phylogenetic Trees

Paper IV

An Approximate Maximum Likelihood Approach, applied to Phylogenetic Trees

Henrik Jönsson and Bo Söderberg

Complex Systems Division, Department of Theoretical Physics
Lund University, Sölvegatan 14A, S-223 62 Lund, Sweden
http://www.thep.lu.se/complex/

LU TP 01-31, submitted to *Journal of Computational Biology.*

A novel type of approximation scheme to the maximum likelihood (ML) approach is presented and discussed in the context of phylogenetic tree reconstruction from aligned DNA sequences. It is based on a parameterized approximation to the conditional distribution of hidden variables (related e.g. to the sequences of unobserved branch point ancestors) given the observed data. A modified likelihood, based on the extended data, is then maximized with respect to the parameters of the model as well as to those involved in the approximation. With a suitable Ansatz the proposed method allows for simpler updating of the parameters, at the cost of an increased parameter count and a slight decrease in performance. The method is tested on phylogenetic tree reconstruction from artificially generated sequences, and its performance is compared to that of ML, showing that the approach is competitive for reasonably similar sequences. The method is also applied to real DNA-sequences from primates, yielding a result consistent with those obtained by other standard algorithms.

4.1 Introduction

Several different types of algorithms have been proposed for inferring phylogenetic trees from sequence data of species (see e.g. [10, 8]). The theoretically most appealing of these are based on the maximum likelihood (ML) approach, introduced in this context by Felsenstein [1], and implemented into standard computer packages [2, 13]. In ML, a more or less simple, stochastic model is assumed for sequence branching and independent site evolution, resulting in a tree graph expressing the phylogenetic relationship between the observed species. The topology (structure) and geometry (arc lengths) of the tree, and possible additional model parameters, are to be chosen so as to maximize the *likelihood*, defined to be proportional to the probability of the model to produce the observed sequences.

In practice, the maximization of likelihood with respect to the model parameters is a complicated task. Typically, it is done by optimizing one parameter at a time while keeping the others fixed. This is repeated until some criterion for convergence is met. However, even with the simplest evolution models, each single-parameter optimization step requires an iterative procedure, and the method can be quite time-consuming.

In this article, an alternative approach is proposed, where the observed data is extended by means of a simple Ansatz for the conditional probability distributions of the unobserved branch-node sequences. The conventional likelihood L is replaced by an alternative likelihood $\hat{L} < L$, associated with the extended data. The maximization of L with respect to the model parameters is then replaced by the maximization of $\hat{L}$ with respect to these, as well as to the parameters of the factorized Ansatz.

Depending on the complexity of the Ansatz, this will decrease performance to some degree, as measured by the achieved value of L, and introduce additional parameters to optimize. This is compensated for by allowing for a simpler updating of model parameters. With a suitable Ansatz, also its parameters allow for a simple updating scheme.

The method contains elements both from ML and from the variational approach [3], as used in statistical physics, and will be referred to as a *variational maximum likelihood* approach (*VML*).

4.2 Background

Before presenting the VML approach in some detail in the nect section, we will here introduce notation, and give some theoretical background. We will briefly discuss independent-site Markov models for the evolution and the associated maximum likelihood approach. We will also briefly review variational methods as used in statistical physics, in particular the specific example given by the mean-field (**MF**) approximation [9].

4.2.1 Stochastic Independent-Site Mutation Models

We will consider a class of simple stochastic models for the evolution of DNA sequences, where individual sites in the sequence mutate at random according to an identical model, but independently from each other; deletions and insertions are neglected. Speciation is assumed to occur in the form of branching events, where a species bifurcates in two,[1] which continue to evolve independently.

In such a model, a specific set of observed species is assumed to have evolved from a common ancestor, by means of successive bifurcations, and subsequent periods of independent random mutations. Their phylogenetic relationship takes the form of a phylogenetic tree, with the observed species at the leaves, while each branch point corresponds to an unobserved ancestor subject to a speciation event.

There is a priori no reason to assume that the mutation rate is the same in different branches, nor constant in time on a single branch. If it were, one would expect constraints between the evolutionary distances along the different branches in the tree. This also makes the position in the tree of the common ancestor ambiguous, in particular if the assumed model is invariant under time reversal.

Thus, one is led to consider unrooted trees without a definite temporal ordering along branches. Assuming interior nodes of order three, a tree with N leaves will have $N-2$ internal nodes and $2N-3$ links, and allow for $(2N-5)!!$ distinct leaf-labeled topologies. Figure 4.1 shows a schematic phylogenetic tree for five observed species, connected via three unobservable ancestors.

Consider a set of N aligned homologous DNA sequences, corresponding to a set of N species. Then at each site in the sequences, a combination $S = (s_1, \ldots, s_N)$ is observed, where each s_i is one of the four symbols A, C, G, or

[1] Multifurcations can be seen as sequences of several bifurcations

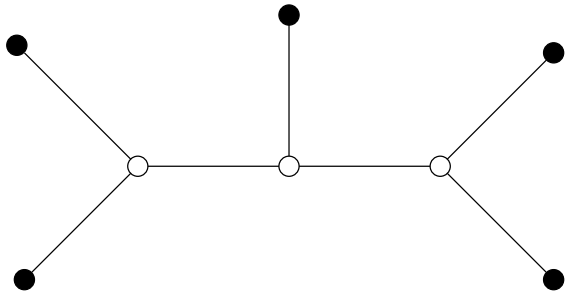

Figure 4.1: A phylogenetic tree.

T, corresponding to an alphabet of size $K = 4$. For a given tree topology, the set of $N - 2$ branch point species corresponds to an unknown combination $I = (i_1, \ldots, i_{N-2})$ at the same site.

In a model of the above described type, the probability for a definite aggregate symbol combination $(SI) = (x_1, \ldots, x_{2N-2})$ is given by a product of single link factors,

$$P_{SI} = \prod_{[kl]} T^{[kl]}_{x_k, x_l}, \tag{4.1}$$

with $[kl]$ denoting a link between nodes k and l. The entries in a link factor $T^{[kl]}$ depend on the specific mutation model, as well as on the link-specific parameters, such as evolutionary distance. Different models put more or less severe constraints on the link factors.

With a given topology, and with given link factors defining P_{SI}, the probability of the observed single-site combination S becomes

$$P_S = \sum_I P_{SI}. \tag{4.2}$$

4.2.2 Maximum Likelihood

In a situation where aligned homologous sequences from a set of N species are given, each individual site can be viewed as an independent experiment, the outcome of which is the ordered set S of observed symbols of the considered species at that particular site. The observation of the full sequences, assumed to have a common length M, thus corresponds to repeating the same experiment M times and gathering statistics of the outcome. The statistical information

can be collected into an *observed distribution* Q over the single-site observations S,

$$Q_S = \frac{M_S}{M}, \tag{4.3}$$

where M_S is the number of sites where the single-site combination S appears. Obviously, we have $\sum_S Q_S = 1$.

To these experimental data, the parameters of a model is to be fitted. Assuming the model to yield the probability P_S for a single-site combination S, the probability that the model produce the observed multiplicities M_S is given by $\prod_S P_S^{M_S}$, multiplied by the proper combinatorial factor $M!/\prod_S M_S!$, that takes into account the number of distinct ways to realize the observed multiplicities (by permuting sites). In the limit of very long sequences the combinatorial factor is dominated by $\prod_S Q_S^{-M_S}$, which we use to define a suitably normalized likelihood as

$$L = \prod_S \left(\frac{P_S}{Q_S}\right)^{MQ_S}, \tag{4.4}$$

associated with a model yielding the probabilities P, given the observed distribution Q.

Note that the approximation of the combinatorial factor only affects the normalization of the likelihood in a way that is independent of P, and so is harmless. Although somewhat unconventional, this normalization is very natural since it gives a unit likelihood for a perfect fit, i.e. for $P = Q$.

Maximizing L corresponds to minimizing its *negative logarithm* divided by the sequence length M, to be referred to as the *free energy* per site, F,

$$F = \sum_S Q_S \log\left(\frac{Q_S}{P_S}\right) \geq 0. \tag{4.5}$$

In terms of F, the likelihood is given by $L = \exp(-MF)$. The free energy is related to the *mutual entropy* between Q and P; it is a *strictly convex, non-negative* function of the probabilities P_S, and hence of P_{SI}, vanishing only if P and Q are identical. Thus, if a model is capable of producing a P that exactly matches Q it will yield a vanishing F; otherwise a strictly positive F will result.

4.2.3 Variational Method - General Principles

Consider a situation where a complicated theoretical probability distribution Q over a set of variables S is given, and for simplicity one wishes to approximate it in an optimal way by a simpler, parameterized Ansatz P of a certain form.

In the *variational approach* [3], one considers an associated *variational free energy* G, defined by

$$G = \sum_S P_S \log\left(\frac{P_S}{Q_S}\right), \tag{4.6}$$

which is to be minimized with respect to the parameters in P. Note that G is a non-negative convex function of P, with a unique vanishing minimum for $P = Q$.

G has a very similar appearance to the free energy F of the ML approach, eq. (4.5). Note however, that while both are to be minimized with respect to P for fixed Q, the roles of P and Q are interchanged in the expression for G.

4.2.4 The Mean-Field Approximation

The distribution over symbols considered in the reconstruction of phylogenetic trees is highly analogous to the thermal distributions encountered in the statistical physics of spin systems.

In the context of spin systems, a common application of the variational approach is the *Mean-Field* (**MF**) approximation [9], where a given spin distribution is approximated by one that is factorized over the distinct spin variables. Thus, for a set of K-state spins $S = (s_1 \dots s_N)$, a given distribution Q_S is to be approximated in an optimal way by a factorized distribution $P_S = \prod_{i=1}^N v^i_{s_i}$, as defined by the minimization of the free energy

$$G(v) = \sum_i \sum_s v^i_s \log v^i_s - \sum_S \log Q_S \prod_i v^i_{s_i}. \tag{4.7}$$

The expression $-\log Q_S$ can be interpreted as a cost function (or Hamiltonian) H_S, in terms of which G can be written as

$$G(v) = \sum_i \sum_s v^i_s \log v^i_s + \langle H_S \rangle_v, \tag{4.8}$$

where the first term is the negative of the entropy, while the second expresses the average cost. Minimization of G with respect to each single-spin distribution v^i yields the *MF equations*,

$$v^i_s \propto \exp\left(-\partial \langle H \rangle_v / \partial v^i_s\right), \tag{4.9}$$

where the constant of proportionality is fixed by the normalization, $\sum_s v^i_s = 1$. (Locally) optimal distributions can be found by an iterative updating scheme based on eq. (4.9).

4.3 Variational Maximum Likelihood

We are now prepared to formulate the hybrid approach VML, where an approximation to the likelihood is maximized. It contains elements both of ML and of the variational approach.

4.3.1 General Idea

When applying ML to a stochastic mutation model as described in section 4.2.1, the factorization in P_{SI} cannot be fully exploited, since it is lost in P_S due to the summation over the hidden symbols I.

If, in addition to the observed sequences at the leaves of the tree, also the corresponding sequences of the unknown ancestors at branchpoints were known, one could apply ML to the extended distribution Q_{SI}, corresponding to minimizing a modified free energy $\hat{F}$,

$$\hat{F} = \sum_{SI} Q_{SI} \log \left(\frac{Q_{SI}}{P_{SI}} \right). \tag{4.10}$$

This would simplify the maximization of likelihood considerably due to the factorization of P, corresponding to a decomposition of $\hat{F}$ as the sum of terms, each associated with a single link. As a result, the parameters at a single link would be determined by the minimization of an expression like

$$\sum_{ij} q_{ij} \log \frac{q_{ij}}{p_{ij}}, \tag{4.11}$$

where i, j correspond to the respective symbols at the two nodes connected by the link, while q and p denote their joint marginal distribution as derived from Q_{SI} and P_{SI} respectively.

To take advantage of the factorization property of P_{SI}, we now propose an approximative approach, generically described as follows.

VML:

- Choose a parameterized Ansatz for the conditional distribution $Q_{I|S}$ of the hidden symbols, given the observed ones.
- Q_{SI} is now determined by Q_S and $Q_{I|S}$. Consider the corresponding free energy $\hat{F}$, as given by eq. (4.10).

- The minimization of $\hat{F}$ with respect to the model parameters in P is straightforward, yielding an optimal value of $\hat{F}$, associated with this particular Ansatz.
- Minimize the resulting free energy also with respect to the Ansatz parameters in $Q_{I|S}$.

Thus, in VML, $\hat{F}$ is to be minimized both with respect to the parameters of the model defining P_{SI}, and the parameters in the Ansatz defining $Q_{I|S}$.

Note that $\hat{F}$ approximates F from above, which can be seen by rewriting it as

$$\hat{F} = F + \sum_S Q_S \left[\sum_I Q_{I|S} \log \left(\frac{Q_{I|S}}{P_{I|S}} \right) \right] \geq F. \tag{4.12}$$

It has the obvious form

$$\hat{F} = F + \sum_S Q_S G_S, \tag{4.13}$$

where for each S, G_S can be interpreted as a *variational free energy* for the approximation of $P_{I|S}$ by $Q_{I|S}$.

With a general enough Ansatz for $Q_{I|S}$, it would match $P_{I|S}$ at optimality, making the second term above vanish. Thus, in such a case, $\min \hat{F} = \min F$. The resulting approach would be an exact reformulation of conventional ML.

4.3.2 Factorized Ansatz

Here we will consider another possibility as an example, by employing a particular Ansatz for $Q_{I|S}$, constrained to be *factorized* over the internal nodes,

$$Q_{I|S} = \prod_k v^k_{i_k|S}, \tag{4.14}$$

where k runs over the $N-2$ internal nodes. Then, the modified free energy $\hat{F}$ can be simplified to read

$$\hat{F} = \sum_S Q_S \log Q_S + \sum_S Q_S \sum_k \sum_i v^k_{i|S} \log v^k_{i|S} - \sum_S Q_S \sum_{[kl]} \sum_{ij} v^k_{i|S} v^l_{j|S} \log T^{[kl]}_{ij}, \tag{4.15}$$

where $[kl]$ labels a link connecting two nodes k, l, while $T^{[kl]}$ is the corresponding link factor. If k refers to an external node, $v^k_{i|S}$ is to be interpreted as δ_{i,s_k}.

For each S, the corresponding Ansatz probabilities v minimize

$$\sum_k \sum_i v_i^k \log v_i^k - \sum_{[kl]} \sum_{ij} v_i^k v_j^l \log T_{ij}^{[kl]}, \tag{4.16}$$

where the "$|S$" has been stripped off for clarity. This has the precise form of a variational free energy for the MF approximation, cf. eq. (4.8). The condition for a local minimum of $\hat{F}$ with respect to v^k yields the MF equations (4.9), which in this case can be written as

$$v_{i|S}^k \propto \prod_{l \in \mathcal{N}_k} \prod_j \left(T_{ij}^{[kl]} \right)^{v_{j|S}^l}, \tag{4.17}$$

normalized such that $\sum_i v_i^k = 1$. Here, $\mathcal{N}_k$ stands for the set of nodes that are neighbors to k. Eq. (4.17) can be used for iteratively updating v.

There is an obvious ambiguity in the link factors, associated with factors depending on a single internal symbol – such factors can be exchanged between the link factors associated with the three links surrounding it, without affecting their product. For a fixed link $[kl]$, we can use this freedom to force the corresponding link factor $T^{[kl]}$ to equal the marginal two-symbol distribution p^{kl}, derived from P_{IS} by summing over the remaining nodes.

Then the part of $\hat{F}$ relevant for a link factor p^{kl} can be written in the form of eq. (4.11). This means that p^{kl} should be chosen to optimally fit (in the ML sense) the corresponding marginal two-symbol distribution q^{kl}, as determined by Q_{IS}. For a parameter a in p^{kl}, optimality thus implies

$$\frac{\partial \hat{F}}{\partial a} \equiv \sum_{ij} \frac{q_{ij}^{kl} \partial p_{ij}^{kl} / \partial a}{p_{ij}^{kl}} = 0, \tag{4.18}$$

if the optimal value of a is in the interior of its allowed interval.

One might fear that the optimization of the link factors in the form of pair-distributions might yield inconsistent results for two neighboring links, since both pair-distributions determine the marginal distribution p^k for their common node k. This is no problem – both are consistent with an identical p^k, minimizing (if not fixed by the model) its own relevant part of $\hat{F}$, given by

$$\sum_i q_i^k \log \left(\frac{q_i^k}{p_i^k} \right). \tag{4.19}$$

4.4 Application to the JC model

While the above considerations are somewhat abstract, we will in this section be very concrete and give a detailed description for one of the simplest mutation models, the Jukes-Cantor **JC** model [6]. It is highly constrained: For an arbitrary alphabet size K, a link factor (as above taken as the corresponding marginal pair probability) must take the form

$$p_{ij} = \frac{1}{K^2}\left(1 - a + Ka\delta_{ij}\right), \tag{4.20}$$

with a single free parameter $a \in [0,1]$ per link, given by $a = \exp(-t)$, with t an associated evolutionary distance. For each single node k, this yields a uniform marginal single-symbol distribution, $p_i^k = 1/K$. For a given topology, the full P_{SI} becomes

$$P_{SI} = \frac{1}{K}\prod_{[kl]}\left(\frac{(1-a_{[kl]})}{K} + a_{[kl]}\delta_{ij}\right). \tag{4.21}$$

4.4.1 ML approach for JC

In the ML approach one would for the JC model consider the free energy F of eq. (4.5), with P_S as given by summing P_{SI} in eq. (4.21) over I. F is to be minimized both with respect to the topology, as given by the structure of the tree, and with respect to its geometry, as defined by the link parameters a.

Thus, for a fixed topology, the link parameters a are chosen so as to minimize

$$F = \sum_S Q_S \log(Q_S) - \sum_S Q_S \log\left(\sum_I P_{SI}(a)\right). \tag{4.22}$$

For the update of a single link parameter $a = a_{[kl]}$, it is advantageous to write F as

$$F = \text{const.} - \sum_S Q_S \log\left(\sum_{ij} u_i^S \left(\frac{1-a}{K} + a\delta_{ij}\right) v_j^S\right) \tag{4.23}$$

$$= \text{const.} - \sum_S Q_S \log\left(\frac{1-a}{K}\sum_i u_i^S \sum_j v_j^S + a\sum_i u_i^S v_i^S\right), \tag{4.24}$$

where u_i^S and v_j^S represent probabilities associated with the observed symbols S in the two subtrees joined by link $[kl]$, and fixed symbols i, j at the nodes k, l

attached to the link. They do not depend on a, and differentiation of F with respect to a yields

$$\frac{\partial F}{\partial a} = -\sum_S \frac{Q_S}{a - z_S}, \tag{4.25}$$

which should vanish at minimum. This expression has singularities (simple poles) at S-dependent positions z_S, given by

$$z_S = -\frac{\sum_i u_i^S \sum_j v_j^S}{K \sum_i u_i^S v_i^S - \sum_i u_i^S \sum_j v_j^S}, \tag{4.26}$$

guaranteed to lie outside the interval $[-1/(K-1), 1]$.

Thus, for a in the physical interval $[0, 1]$, $\partial F/\partial a$ is strictly increasing; if it has a zero in this interval, this defines the optimal value of a (for fixed values of the other parameters); otherwise it is positive on the whole interval $[0, 1]$, in which case $a = 0$ is the optimal value, or negative, in which case $a = 1$ is optimal. An optimum in the interior of $[0, 1]$ has to be found by some iterative method, such as Newton-Raphson, or binary search.

In this way, one parameter at a time can be locally optimized for fixed values of the others, eventually leading to a (local) minimum of F for the chosen topology; this value is taken as a measure of F for that topology.

The optimization with respect to tree topology can be done in different ways. To strictly ensure that the best topology is found, a search of all possible topologies must be performed. Often, though, one settles for a neighborhood search, where an initial topology is chosen at random, or better, by means of some heuristic, and an optimization if performed with respect to link parameters, yielding a value of F for the chosen topology. Then, neighboring topologies, obtained e.g. by rearranging the tree around one of the shorter links (i.e. one with a large a), are investigated. If one of these yields a lower F, it is chosen as the new present topology, and its neighbours are checked. When no more improvements can be made in this way, the present topology is considered optimal. For a more detailed discussion of heuristic topology optimization, see e.g. [10].

4.4.2 VML approach for JC

For the JC model, the VML method becomes particularly simple. At optimality, each internal single-node distribution v^k must satisfy

$$v_i^k \propto \prod_{l \in \mathcal{N}_k} \left(\frac{1 + (K-1)a_{[kl]}}{1 - a_{[kl]}} \right)^{v_i^l}, \tag{4.27}$$

with normalization such that $\sum_i v_i^k = 1$. Similarly, for fixed values of the other parameters, the optimal value for a link parameter $a_{[kl]}$ is given by

$$a_{[kl]} = \frac{K \sum_S Q_S \sum_i v_{i|S}^k v_{i|S}^l - 1}{K-1} = \frac{K \left\langle \mathbf{v}^k \cdot \mathbf{v}^l \right\rangle_Q - 1}{K-1}, \tag{4.28}$$

which is automatically in the interval $[-\frac{1}{(K-1)}, 1]$, since $\mathbf{v}^k \cdot \mathbf{v}^l \in [0,1]$. A negative value can be corrected to yield zero.

In this way, one parameter at a time can be locally optimized for fixed values of the others, eventually leading to a (local) minimum of $\hat{F}$ for the chosen topology; this value is taken as a measure of $\hat{F}$ for that topology. The optimization with respect to topology can be performed in the same way as in ML.

4.4.3 Note on more general models

The JC model is the simplest random independent site mutation model. It has been extended by allowing for differentiation in transition rates, as well as for different single-node probabilities [7, 11, 1, 5]. Also more general models generated from arbitrary rate matrices (reversible or not) have been studied ([12, 4]).

In analogy to the proper ML approach, VML can be applied equally well to any of these models. The actual updating equations will of course change, but will still be based on eqs. (4.17, 4.18).

4.5 Numerical Explorations

In this section we present the results of some simple computer experiments to gauge the performance of VML by comparing it to standard ML. We have consistently used JC for transition probabilities, both when generating sequences

for the testbed problems and as the underlying model in the algorithms used to infer the trees.

First, we have probed the method for a homogenous (all a equal) tree with infinite sequence length, and varying a. In the next test we used randomly generated trees, where we compared the achieved values for F obtained by the methods when given the correct (generated) tree topology, and also checked whether the tree with correct topology has the lowest F. Finally we used DNA-sequences from primates to check whether the preferred topology from VML is the one achieved by standard methods.

4.5.1 Infinite Sequence Test

If random sequences are generated, e.g. according to the JC model, for a tree with given topology and geometry, the single-site distributions Q_S will approach the corresponding model probabilities P_S in the limit of infinitely long sequences. Thus, for each of the K^N possible S, the JC model yields

$$Q_S = \sum_I \frac{1}{K} \prod_{[kl]} \left(\frac{1 - a_{[kl]}}{K} + a_{[kl]} \delta_{i_k, i_l} \right). \tag{4.29}$$

Given these data and the correct topology, we expect the ML algorithm to be able to produce the correct link lengths and a vanishing F within numerical limitations. Being based on maximizing an approximation to the real likelihood, the VML algorithm can be expected to perform slightly worse on such data.

We have probed VML and ML with infinite-sequence data for four species[2], based on a tree where the five link parameters were all set equal to a common value a_0. Figure 4.2 A shows the resulting values of F as a function of the input parameter a_0. When applied to the correct topology, ML clearly gives an essentially perfect fit as expected. VML is seen to perform well for short links ($a_0 \approx 1$), and slightly worse for longer links (smaller a_0), though still not far from optimal. For an incorrect topology both algorithms produced F values well above the ones achieved for the correct topology, showing that either algorithm identifies the correct topology. The resulting individual link parameters were essentially exactly a_0 for the ML algorithm when probed on the correct topology. For VML the resulting link parameters tend to deviate somewhat for cases with smaller a_0 as shown in figure 4.2 B.

[2]For three species the mean-field approximation becomes exact.

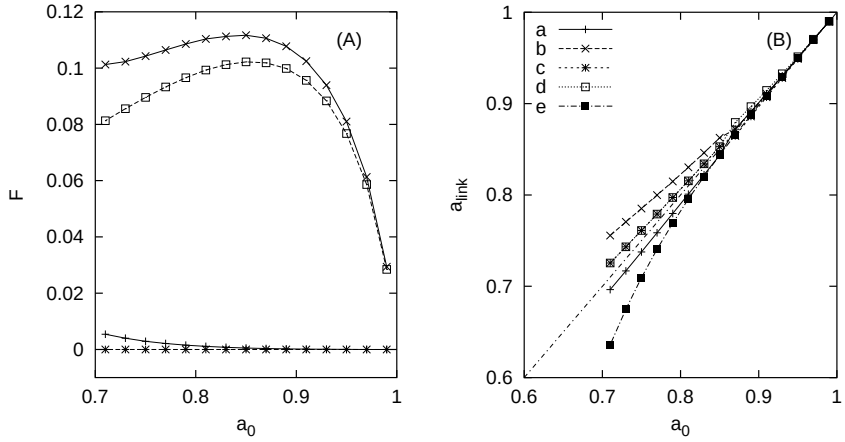

Figure 4.2: Infinite sequence results for N=4. A) The free energy per site (F) versus link lengths of generated tree (a_0) plotted for ML (correct topology ($*$), wrong topology ($\square$)) and VML (correct topology ($+$), wrong topology ($\times$)). B) Link lengths resulting from the VML algorithm applied on the correct topology (a_{link}) versus link lengths of generated tree (a_0). Links a and b connects one pair of species, c and d the other pair, and e is the link between the internal nodes.

4.5.2 Tests with Artificial Sequences

We have used testbeds with artificial sequences generated according to the JC model on random trees for varying numbers of species, N, and sequence lengths, M. The random trees were defined by first generating a random topology, then setting link parameters according to a specific probability distribution. Finally, sequences were generated based on JC on this tree.

A random tree topology is defined by starting with two nodes connected by a single link. Then a new node is connected to an existing link chosen at random; this is repeated until there are N external nodes. This results in an equal probability for each possible leaf-labeled topology. Link parameters are independently generated as $a := R^{t_0}$, with R a distinct uniform random number in $[0, 1]$, while t_0 is a common parameter, setting the temporal scale of the generated links. The motivation for using this probability distribution is that the time associated with a link will follow an exponential distribution, corresponding to a constant branching rate. For t_0 we have used values between 0.05 and 0.3, corresponding to an average a between 0.95 and 0.77. The a-distributions are shown in figure 4.3. Finally, sequences are generated randomly

according to the JC model.

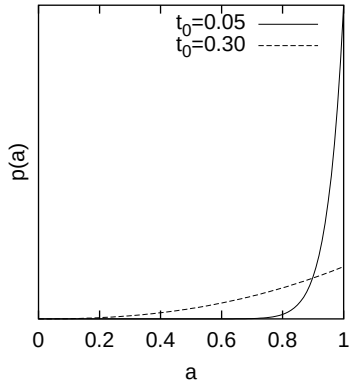

Figure 4.3: Link parameter distributions ($p(a)$) for the studied test sets.

Inferring the Geometry

The ML and VML algorithms were used to infer the geometry of each tree given the correct topology and the external sequences, and the free energy, F, was used as a measure of the quality of the obtained geometry. We have probed different numbers of species, $N = \{4, 8, 12, 16\}$, link distributions, $t_0 = \{0.05, 0.1, 0.3\}$, and sequence lengths, $M = \{250, 500, 1000\}$, and constantly used the alphabet size $K = 4$ for the sequences.

In figure 4.4 the achieved F values are plotted versus N for different values of M and t_0. As can be seen in the figure, the difference between ML and VML is hardly noticable. However there is a small difference, as shown in figure 4.5.

As in the case with infinite sequence lengths we can again see that the difference increases for trees where longer links are used. A larger M results in lower values of F for both algorithms, and the difference has a quite small M-dependence.

Topology Inference

We also tested the ability to identify the correct topology, in the sense that it yields the lowest free energy compared to other topologies. The tests were

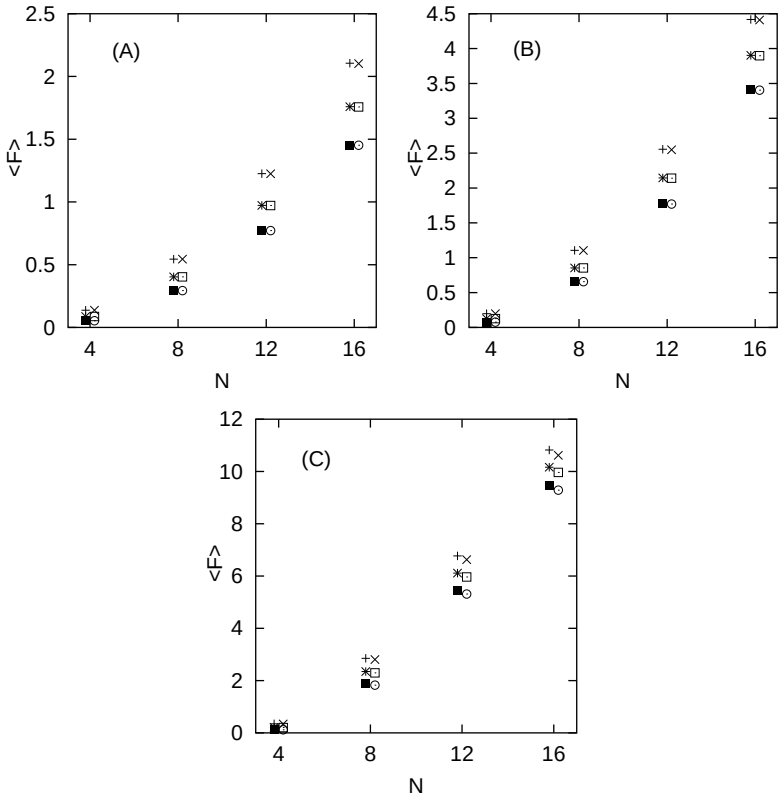

Figure 4.4: Average of free energy per site $< F >$ versus number of species N for ML and VML. Each data point is the average of 100 randomly generated trees. $K = 4$ for all runs. The plots shows $M = 250$ (ML($\times$) and VML($+$)), $M = 500$ (ML($\square$),VML($*$)) and $M = 1000$ (ML($\circ$),VML($\blacksquare$)). Link parameters are generated using $a := R^{t_0}$ where R is a uniform random number in [0,1]. A) $t_0 = 0.05$, B) $t_0 = 0.1$ and C) $t_0 = 0.3$.

performed on trees with four species, and link parameters were generated as above, using t_0 between 0.05 and 0.3. Sequences of length $M = \{250, 500, 1000\}$ and $K = 4$ were used.

The fraction, f_u, of trees where the correct topology does not yield the lowest free energy was measured for ML and VML, and the result is shown in table 4.1. For both algorithms, longer sequences result in a lower f_u, as expected. VML seems to perform slightly worse than ML, and the tendency that the

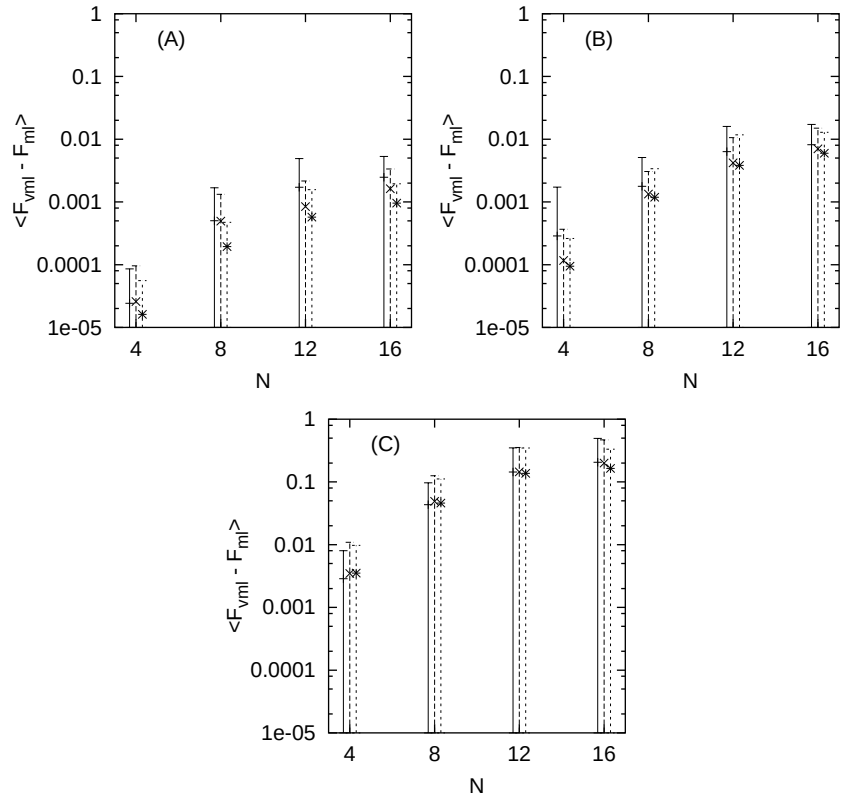

Figure 4.5: The average of the difference in F versus N for the same data set as in figure 4.4. The plots show the difference for $M = 250$ (+), $M = 500$ ($\times$) and $M = 1000$ ($*$). The errorbars indicate the standard deviation. A) $t_0 = 0.05$, B) $t_0 = 0.1$ and C) $t_0 = 0.3$.

difference between the algorithms increase with link length is again present.

4.5.3 Primate Sequence Test

VML was also applied to real DNA data, using five homologous primate sequences obtained from the Silver Project[3]. The aligned sequences are from the aromatic L-amino acid decarboxylase (AADC) gene, and 711 nucleotide posi-

[3] `http://sayer.lab.nig.ac.jp/~silver/`

	M=250		M=500		M=1000	
t_0	f_u^{vml}	f_u^{ml}	f_u^{vml}	f_u^{ml}	f_u^{vml}	f_u^{ml}
0.05	0.09	0.09	0.05	0.04	0.03	0.03
0.075	0.07	0.07	0.04	0.04	0.03	0.02
0.1	0.06	0.06	0.05	0.05	0.03	0.03
0.15	0.07	0.07	0.03	0.02	0.03	0.03
0.2	0.09	0.08	0.06	0.04	0.04	0.03
0.25	0.12	0.10	0.09	0.04	0.07	0.04
0.3	0.11	0.07	0.09	0.07	0.07	0.04

Table 4.1: Fraction of trees where the correct (generated) tree does not have the lowest free energy (f_u). 500 trees for each link parameter (t_0) and sequence length (M) are tested and standard ML and VML are compared. All tests are performed with $N = 4$ and $K = 4$.

tions are used from two humans, a chimpanzee, a gorilla and an orangutan.

As there are only five sequences, all possible (15) topologies have been investigated. VML and ML both favor the tree shown in Figure 4.6 as the most likely tree. The topology is identical to the one proposed along with the data, obtained with a neighbour-joining method. Both ML and VML yielded a free energy per site of $F = 0.04867$, and the link lengths inferred are those given in the figure. Also local search implementations of both the ML and VML

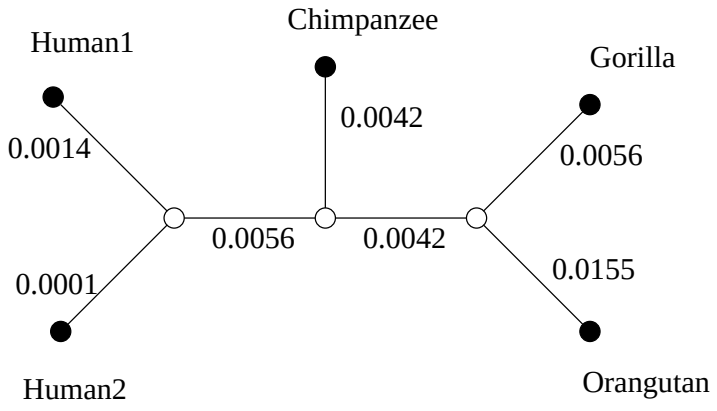

Figure 4.6: Preferred topology for ML and VML. The inferred link lengths given as expected number of substitutions per site, $h = \frac{K-1}{K}(1-a)$, are the same for the two methods.

methods were tested. A local search implementation starts from a random topology, checks the two neighbouring topologies around the shortest internal link, and changes topology if any of the new topologies result in a lower F. This is continued until no improvement can be found. For each choice of an initial topology, the local search variants of ML and VML found the topology shown in figure 4.6.

4.6 Summary and Conclusions

We have proposed and explored a novel hybrid approach, VML, where the variational mean-field approximation is combined with the maximum likelihood principle for the reconstruction of phylogenetic trees based on DNA sequences.

The method consists in maximizing an approximation from below to the actual likelihood L. Results for cases with reasonably similar sequences are comparable to those of a standard ML approach, while for dissimilar sequences the method deteriorates somewhat.

The advantage of VML is that it may allow for simpler update equations for the link parameters as compared to standard ML. This is especially apparent with the Jukes-Cantor model, where the link parameters can be directly updated, whereas in standard ML an iterative procedure has to be used.

The VML approach defines a generic class of methods, and is applicable to all possible models where ML can be used. This includes different mutation models as well as models with differentiated rates at different sites in the sequences.

Acknowledgements

This work was in part supported by the Swedish Foundation for Strategic Research.

References

[1] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[2] J. Felsenstein. *PHYLIP: Phylogenetic Inference Package*. Seattle, WA: University of Washington, 1993.

[3] R. Feynman. *Statistical Mechanics, Frontiers in Physics*. W. A. Benjamin, Inc., Reading, MA, 1972.

[4] X. Gu and W. Li. A general additive distance with time-reversability and rate variation among nucleotide sites. *Proceedings of the National Academy of Sciences of the United States of America*, 93:4671–4676, 1996.

[5] M. Hasegawa, H. Kishino, and T. Yano. Dating the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of Molecular Evolution*, 22:160–174, 1985.

[6] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic, New York, 1969.

[7] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980.

[8] M. Nei. Phylogenetic analysis in molecular evolutionary genetics. *Annual Review of Genetics*, 30:371–403, 1996.

[9] G. Parisi. *Statistical Field Theory*. Addison-Wesley Publishing Company, Reading, MA, 1988.

[10] D. L. Swafford and G. J Olsen. Phylogeny reconstruction. In C. Moritz D. M. Hillis and B. K. Mable, editors, *Molecular Systematics*. Sinauer Associates, Sunderland, 1996.

[11] K. Tamura and M. Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular Biology and Evolution*, 10:512–526, 1993.

[12] Z. Yang. Estimating the pattern of nucleotide substitution. *Journal of Molecular Evolution*, 39:105–111, 1994.

[13] Z. Yang. Paml: A program package for phylogenetic analysis by maximum likelihood (`http://abacus.gene.ucl.ac.uk/software/paml.html`). *CABIOS*, 13:555–556, 1997.