LU TP 00-40
October 3, 2000

An Information-Based Neural Approach to Generic Constraint Satisfaction

Henrik Jönsson[1] and Bo Söderberg[2]

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

Submitted to Artificial Intelligence

Abstract

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.

[1]henrik@thep.lu.se
[2]bs@thep.lu.se

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 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 $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_N\}$.

2.1 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.

2.1.1 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

$$\begin{aligned} v_{ic} &= \frac{\exp(u_{ic})}{\sum_d \exp(u_{id})}, &(1)\\ u_{ic} &= -\frac{1}{T}\partial E/\partial v_{ic}. &(2)\end{aligned}$$

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.

2.1.2 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

$$F(\mathbf{v}) = T\sum_{ic} v_{ic}\log(v_{ic}) + E(\mathbf{v}), \tag{3}$$

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

$$T\log(v_{ic}) + T + \partial E/\partial v_{ic} = \lambda_i, \tag{4}$$

which leads to the MF equations (1,2).

2.1.3 MF Annealing

The MF equations (1,2) 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, $\mathbf{v}_i \to \mathbf{s}_i$, which defines the output of the algorithm. If this defines a solution, the CSP is proven solvable.

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.

2.2.1 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,

$$P(\mathbf{s}) = \prod_i p_i(\mathbf{s}_i). \tag{5}$$

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

$$f_m\left(\mathbf{s}_i, i \in \Omega_m\right) = 0. \tag{6}$$

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

$$U_m(\mathbf{v}) = \sum_{c_1,\ldots,c_K} v_{i_1,c_1} \ldots v_{i_K,c_K} \Theta_{c_1 \ldots c_K}, \tag{7}$$

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.

$$I(\mathbf{v}) = -\sum_m \log(1 - U_m(\mathbf{v})), \tag{8}$$

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

$$P_I(\mathbf{s}) \propto \prod_m (1 - U_m(\mathbf{s}))^{\frac{1}{T}}, \tag{9}$$

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).

2.2.2 INN MF Annealing

When using the INN cost function in place of the conventional ANN one in the MF equations (1,2), a slight modification is needed due to its non-polynomial nature, in order to avoid instabilities [12]. Eq. (2), which is relevant for a multilinear polynomial cost function, has to be replaced by

$$u_{ic} = -\frac{I(\mathbf{v})|_{\mathbf{v}_i=\hat{c}}}{T} \tag{10}$$

(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) for a multilinear cost function.

Due to the singular behavior of the logarithm in eq. (8) and thus in eq. (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. (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 (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.

2.2.3 Variational Interpretation

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

$$F(\mathbf{v}) = -TS(\mathbf{v}) + I(\mathbf{v}), \tag{11}$$

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*.[3] 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.

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**).

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

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.

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

$$U_m(\mathbf{v}) = \mathbf{v}_{i_m} \cdot \mathbf{v}_{j_m}, \tag{12}$$

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

$$I = -\frac{1}{2} \sum_{i,j} J_{ij} \log\left(1 - \mathbf{v}_i \cdot \mathbf{v}_j\right). \tag{13}$$

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

$$u_{ic} = \frac{1}{T} \sum_j J_{ij} \log\left(1 - v_{jc}\right), \tag{14}$$

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.

3.1.1 Critical T Discussion

The MF equations have a trivial solution for

$$v_{ic} = \frac{1}{C}, \tag{15}$$

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

$$\epsilon_{ic} = -\frac{1}{T(C-1)} \sum_j J_{ij} \left(\epsilon_{jc} - \frac{1}{C} \sum_d \epsilon_{jd} \right), \tag{16}$$

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.

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.

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.

4.1.1 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 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

	$\gamma = 3.6$			$\gamma = 4.2$			$\gamma = 4.6$		
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 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.

speed about equally much for 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.

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. 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 ($\sum_i \sum_c v_{ic}^2 > 0.9N$) and stable ($\max_{ic} |\Delta v_{ic}| < 0.01$). In addition, if no solution has been found until $T < 0.3$ the algorithm aborts.

```
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. (10).
   (b) Update $\mathbf{v}_i$ according to eq. (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 1: The INN annealing algorithm for GC.

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.

4.3.1 Conventional ANN annealing

A suitable ANN cost function, corresponding to (13) is

$$E = \frac{1}{2} \sum_{i,j} J_{ij} \mathbf{v}_{i_m} \cdot \mathbf{v}_{j_m}, \tag{17}$$

yielding the MF equations (1) with

$$u_{ic} = -\frac{1}{T}\sum_{j} J_{ij}\mathbf{v}_{j_m}. \tag{18}$$

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).

4.3.2 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. 17, defined as

$$E = \frac{1}{2}\sum_{i,j} J_{ij}\delta_{c_i c_j}, \tag{19}$$

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.

4.3.3 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 [4]. 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.

4.4 Results

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

In figures 2 to 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 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.

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

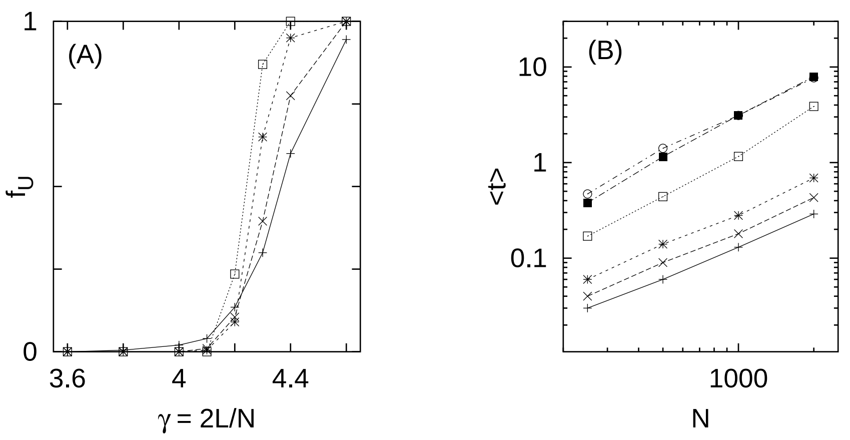

Figure 2: INN results from 200 instances for each N and γ. (**A**) Fraction unsolved problems (f_U) versus γ, for $N = 250$ (+), 500 (×), 1000 (∗), 2000 (□). The statistical error in each point is less than 0.035. (**B**) Average CPU time (in seconds) used versus N, for $\gamma = 3.6$ (+), 3.8 (×), 4.0 (∗), 4.2 (□), 4.4 (■) and 4.6 (○).

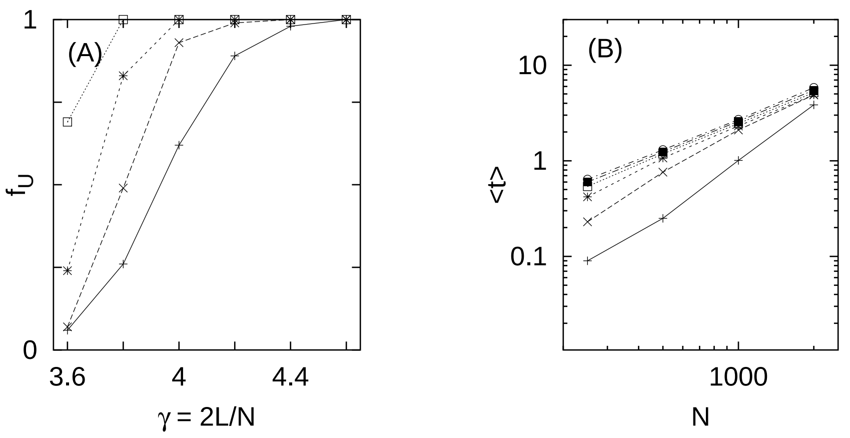

Figure 3: Conventional ANN results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.

4.4.1 Discussion

As can be seen in figures 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).

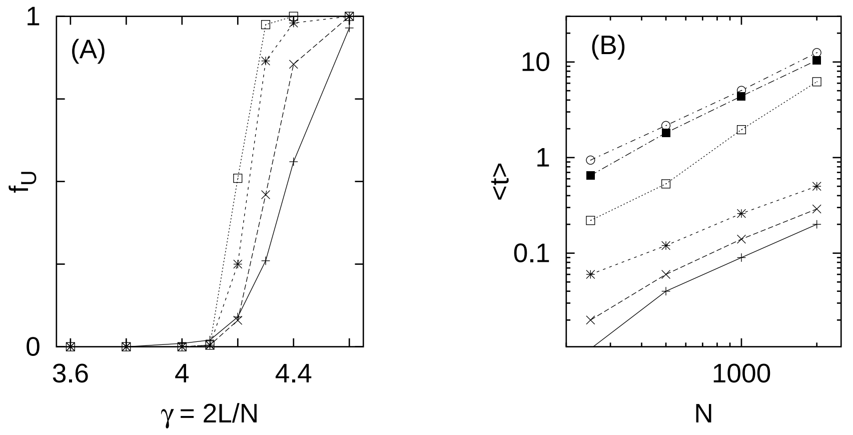

Figure 4: SAU results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.

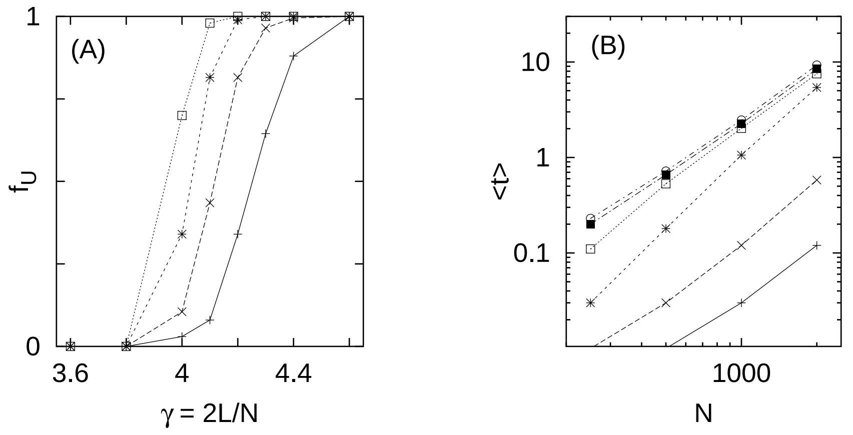

Figure 5: DSATUR results. (**A**) Fraction unsolved problems (f_U) versus γ. (**B**) Average CPU time (in seconds) used versus N. Notation as in figure 2.

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 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.

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.

4.4.2 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

$$R(t) = -\frac{\log U(t)}{t}, \tag{20}$$

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 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 6 does confirm very clearly the qualitative conclusion from figs. 2 and 4, that INN performs somewhat better than SAU for the

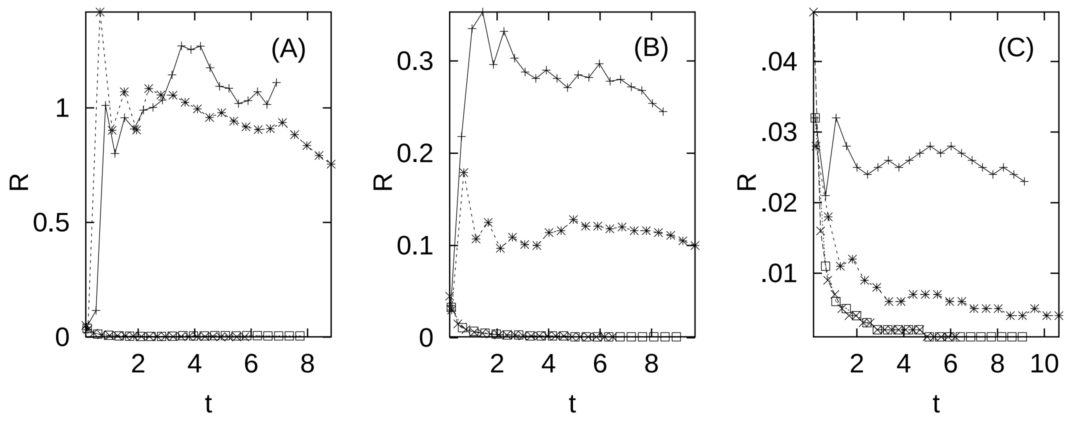

Figure 6: Average solution rates for $N = 2000$, for INN (+), ANN ($\times$), SAU ($*$), and DSATUR ($\square$), at (A) $\gamma = 4.1$, (B) $\gamma = 4.2$, (C) $\gamma = 4.3$.

higher γ values, and that both beat ANN and DSATUR clearly.

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. Technical Report LU-TP 00-19, Department of Theoretical Physics, Lund University, 2000. to appear in *Neural Computation.*

[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.