March 1992
LU TP 92-11

Neural Networks for Optimization Problems with Inequality Constraints - the Knapsack Problem

Mattias Ohlsson[1], Carsten Peterson[2] and Bo Söderberg[3]

Department of Theoretical Physics, University of Lund
Sölvegatan 14A, S-22362 Lund, Sweden

Submitted to **Neural Computation**

Abstract:

A strategy for finding approximate solutions to discrete optimization problems with inequality constraints using mean field neural networks is presented. The constraints $x \leq 0$ are encoded by $x\Theta(x)$ terms in the energy function. A careful treatment of the mean field approximation for the self-coupling parts of the energy is crucial, and results in an essentially parameter-free algorithm.

This methodology is extensively tested on the knapsack problem of size up to 10^3 items. The algorithm scales like NM for problems with N items and M constraints. Comparisons are made with an exact branch and bound algorithm when this is computationally possible ($N \leq 30$). The quality of the neural network solutions consistently lies above 95% of the optimal ones at a significantly lower CPU expense. For the larger problem sizes the algorithm is compared with simulated annealing and a modified linear programming approach. For "non-homogeneous" problems these produce good solutions, whereas for the more difficult "homogeneous" problems the neural approach is a winner with respect to solution quality and/or CPU time consumption.

The approach is of course also applicable to other problems of similar structure, like *set covering*.

[1] mattias@thep.lu.se
[2] carsten@thep.lu.se
[3] bs@thep.lu.se

Background

Feed-back artificial neural networks (ANN) have turned out to be powerful in finding good approximate solutions to difficult combinatorial optimization problems [1, 2, 3, 4]. The basic procedure is to map the problems onto neural networks of binary (Ising spin) or K-state (Potts spin) neurons with appropriate choice of energy functions, and then to find approximate minima of the energy using mean field theory (MFT) techniques. In this way essentially "black box" procedures emerge.

The application areas dealt with in refs. [1, 2, 3, 4] (traveling salesman, graph partition and scheduling) are characterized by global *equality* constraints, which can be implemented as quadratic penalty terms. These contain self-interaction parts (diagonal terms), which can be balanced by counterterms to assure reliable MFT dynamics.

However, in many real-world optimization problems, in particular those of resource allocation type, one has to deal with *inequalities.* The objective of this work is to develop a mapping and MFT method to deal with this kind of problem. As a typical resource allocation problem we choose the knapsack problem for our studies. Although artificial, we feel it is a realistic enough test bed. A crucial ingredient in our approach is to avoid self-couplings by a proper MFT implementation of the constraint terms.

The Knapsack Problem

In the knapsack problem one has a set of N *items* i with associated *utilities* c_i and *loads* a_{ki}. The goal is to fill a "knapsack" with a subset of the items such that their total utility,

$$U = \sum_{i=1}^{N} c_i s_i \tag{1}$$

is maximized, subject to a set of M load constraints,

$$\sum_{i=1}^{N} a_{ki} s_i \leq b_k \ , \ k = 1, M \tag{2}$$

defined by load *capacities* b_k. In eqs. (12) s_i are binary $\{0, 1\}$ decision variables, representing whether item i goes into the knapsack. The variables (c_i, a_{ki} and b_k) that define the problem are all real numbers.

We will consider a class of problems, where a_{ki} and c_i are independent uniform random numbers on the unit interval, while b_k are fixed to a common value b. With $b \approx N/2$, the problem becomes trivial – the solution will have almost all $s_i = 1$. Conversely, with

$b << N/4$, the number of allowed configurations will be small and an exact solution can easily be found. We pick the most difficult case, defined by $b = N/4$. The expected number of used items in an optimal solution will then be about $N/2$, and an exact solution becomes inaccessible for large N.

In the optimal solution to such a problem, there will be a strong correlation between the value of c_i and the probability for s_i to be 1. With a simple heuristic based on this observation, one can often obtain near-optimal solutions very fast. We will therefore also consider a class of harder problems with more **homogeneous** c_i distributions – the extreme case is when c_i are constant, and the utility proportional to the number of items used.

We note in passing that the *set covering* problem is a special case of the general problem, with random $a_{ki} \in \{0, 1\}$, and $b_k = 1$. This defines a comparatively simple problem class, according to the above discussion, and we will stick to the knapsack problem in what follows.

Neural Network Formulation and Solution Strategy

Neural Mapping

We start by mapping the problem defined in eqs. (1,2) onto a generic neural network energy function E

$$E = -\sum_{i=1}^{N} c_i s_i + \alpha \sum_{k=1}^{M} \Phi\left(\sum_{i=1}^{N} a_{ki} s_i - b_k\right) \tag{3}$$

where Φ is a penalty function to ensure that the constraint in eq. (2) is fulfilled. The coefficient α governs the relative strength between the utility and constraint terms. For equality constraints an appropriate choice of $\Phi(x)$ would be $\Phi(x) = x^2$. Having inequalities we need a $\Phi(x)$ that only penalizes configurations for which $x \geq 0$. One possibility is to use a sigmoid, $\Phi(x) = g(x; T)$ (see fig. 1a),

$$g(x; T) = \frac{1}{2}\left[1 + \tanh(x/T)\right] \tag{4}$$

This option has the potential disadvantage that the penalty is the same (=1) no matter how badly the constraints are violated. An alternative that gives penalty in proportion to the degree of violation is

$$\Phi(x) = x\Theta(x) \tag{5}$$

This function (see fig. 1b) has the additional advantage that no extra parameter like the temperature T in the sigmoid is needed. The slope of Φ is implicitly given by α in eq. (3). The $x\Theta(x)$ alternative consistently gives better performance and is used throughout this paper.

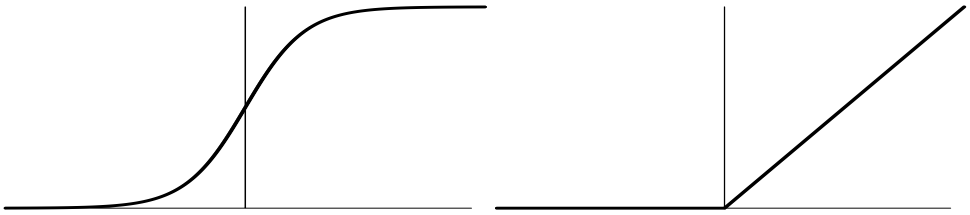

Figure 1: (a) The sigmoid g(x;T) of eq. (4). (b) The penalty function $x\Theta(x)$ of eq. (5).

Mean Field Dynamics

We want to minimize eq. (3) with the mean field approximation (MFT), which has turned out to be very powerful for other optimization problems [2, 3, 4, 5]. Due to the non-polynomial form of the constraint terms (eqs (4,5)) special care is needed when implementing the MFT approximation. Recall that the MFT approximation consists of replacing the binary variables s_i with mean field variables at temperature T, $v_i = < s_i >_T$ and solving the MFT equations

$$v_i = g(-\frac{\partial E}{\partial v_i}; T) \tag{6}$$

by iteration. In problems with equality constraints implemented by quadratic penalty terms the diagonal pieces are compensated for by adding appropriate self-coupling terms. Such a procedure is not trivial in this case of strongly non-linear constraint penalties. Instead, we avoid self-couplings altogether, by replacing $\partial E/\partial v_i$ with the difference in E computed at $v_i = 1$ and $v_i = 0$ respectively. One obtains

$$-\frac{\partial E}{\partial v_i} \rightarrow c_i - \alpha \sum_{k=1}^{M} \left[\Phi(\sum_{j=1}^{N} a_{kj} v_j - b_k)|_{v_i=1} - \Phi(\sum_{j=1}^{N} a_{kj} v_j - b_k)|_{v_i=0} \right] \tag{7}$$

Eqs. (6,7) are solved iteratively by annealing in T. To avoid small final constraint violations, we employ a progressive constraint term, $\alpha \propto 1/T$. This means that the slope of $x\Theta(x)$ (see fig. 1 b) increases during convergence. We will present a standardized scheme below when testing the algorithm numerically. The number of computational steps for solving eqs. (6, 7) scales like NMn_τ, where n_τ is the number of iterations needed for convergence which turns out to be problem size independent (as was observed in other MFT approaches to optimization problems [2, 4, 5]). A factor N has been gained by "recycling" the sums appearing in the argument of Φ.

High-T fixpoints and critical temperature

At a high temperature T, the system will approach a fixpoint with all v_i close to $1/2$ (see fig. 2). With random a_{ki} and c_i on $[0,1]$, and fixed $b_k = b$, two distinct types of high-T behaviour emerge.

- With b well above $b_{crit} \equiv N/4$, all constraints are safe at high T, and the system is stuck at a trivial fixpoint, $v_i = g(c_i; T)$.
- With b instead well below b_{crit}, all constraints are violated at high T, and the trivial fixpoint is instead $v_i = g(c_i - \alpha \sum_k a_{ki}; T)$.

In both cases a statistical analysis shows that v_i remain close to $1/2$ for

$$T >> 1 + \alpha M \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{b-N/8}{\sqrt{N/48}}} e^{-y^2/2} dy. \tag{8}$$

Thus, in the case at hand of $b = b_{crit}$, a suitable starting point for the annealing will be $T \approx 10$.

Other Approaches

In order to see how well our MFT algorithm works we need to compare it with other approaches. For reasonably small problem sizes it is feasible to use an exact algorithm, branch and bound, for comparison. For larger problem sizes, one is confined to other approximate methods, simulated annealing [6], greedy heuristics and linear programming based on the simplex method [7].

Branch and Bound (BB) The knapsack problem is NP-complete, and the effort to find *the* optimal solution by brute force grows like 2^N. Using a branch and bound tree search technique one can reduce the number of computational steps. This method consists in going down the search tree, checking bounds on the constraints or on the utility at each level, thereby avoiding unnecessary searching. In particular for non-homogeneous problems, this method is facilitated by ordering the c_i's according to magnitude:

$$c_1 > c_2 > .... > c_N \tag{9}$$

For problems where the constraints are "narrow" (b not too large) this method can require substantially lower computation needs. However, it is still based on exploration and it is only feasible for problem sizes less than $M = N \approx 30 - 40$.

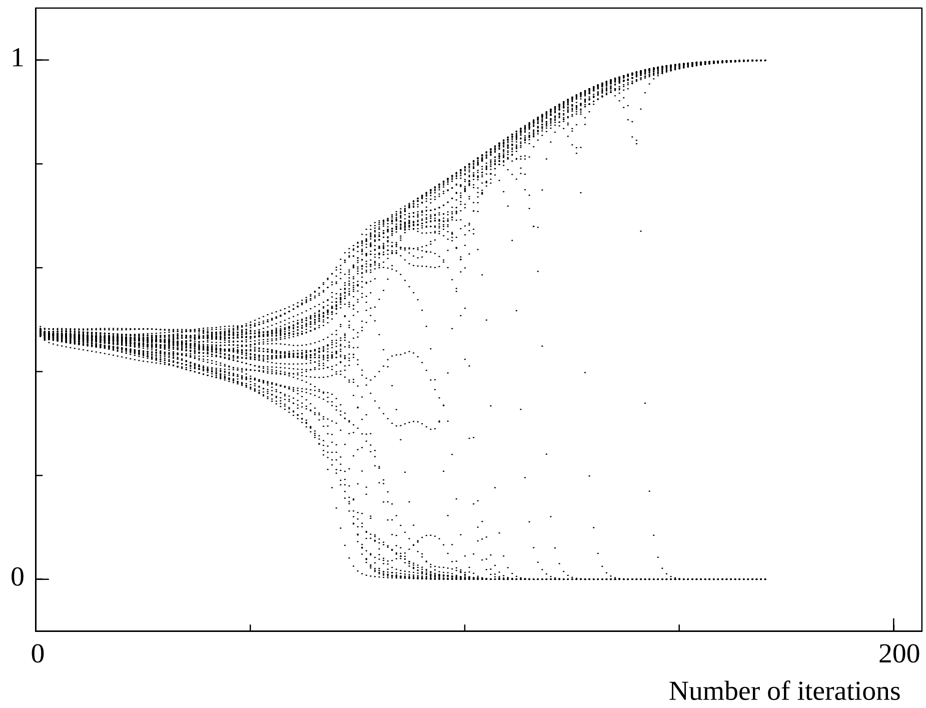

Figure 2: Development of v_i for a N=M=40 knapsack problem with c_i=rand[0.45,0.55].

Greedy Heuristics (GH) This is a simple and fast approximate method for a non-homogeneous problem. Proceeding from larger to smaller c_i (cf. eq. (9)), collect every item that does not violate any constraint. This method scales like NM.

Simulated Annealing (SA) Simulated annealing [6] is easily implemented in terms of attempted single-spin flips, subject to the constraints. Suitable annealing rates and other parameters are given below. This method also scales like NM times the number of iterations needed for thermalization.

Linear Programming with Greedy Heuristics (LP) Linear programming based on the simplex method [7] is not designed to solve discrete problems like the knapsack one. It does apply, however, to a modified problem with $s_i \in [0, 1]$. For the ordered (eq. (9)) non-homogeneous knapsack problem this gives solutions with a set of leading 1's and a set of trailing 0's, with a window in between containing real numbers. Augmented by greedy heuristics for the elements in this window, fairly good solutions emerge. The simplex method scales like (N^2M^2).

Algorithm	c_i=rand[0,1]		c_i=rand[0.45,0.55]		c_i=0.5	
	Perf.	CPU time	Perf.	CPU time	Perf.	CPU time
BB	1	16	1	1500	1	1500
NN	0.98	0.80	0.95	0.70	0.97	0.75
SA	0.98	0.80	0.95	0.80	0.96	0.80
LP	0.98	0.10	0.93	0.25	0.93	0.30
GH	0.97	0.02	0.88	0.02	0.85	0.02

Table 1: Comparison of performance and CPU time consumption for the different algorithms on a N=M=30 problem. The CPU consumption refers to a DEC3100 workstation.

Numerical Comparisons

Neural Network (NN). Convenient measures for monitoring the decision process are the saturation $\Sigma = \frac{4}{N}\sum_i (v_i - 0.5)^2$ and the evolution rate $\Delta = \frac{1}{N}\sum_i(\Delta v_i)^2$, where $\Delta v_i = v_i(t+\Delta t) - v_i(t)$. The saturation starts off around 0 at high temperature T, and increases to 1 in the $T \rightarrow 0$ limit. We have chosen an annealing schedule where T_0=10, $T_n = kT_{n-1}$, where $k = 0.985$ if $0.1 < \Sigma < \frac{N-1}{N}$ and 0.95 otherwise. At each temperature every neuron is updated once. We employ a progressive constraint coefficient, $\alpha = 0.1/T$, to avoid small final constraint violations. The annealing is terminated when $\Sigma > 0.999$ and $\Delta < 0.00001$. Should the final solution violate any constraint (which is very rare), the annealing is redone with a higher α. In fig. 2 we show a typical evolution of $\{v_i\}$ for an $N = M = 40$ problem.

Simulated Annealing (SA). The performance of this method depends on the annealing schedule. In order to compare the performance of this method with that of the neural network approach we have chosen these parameters such that the time consumption of the two methods is the same. This is accomplished with $T_0 = 15$, $T_{final} = 0.01$ and annealing factor $k = 0.99$. First we compare the NN, SA and LP approaches with the exact BB for a $N = M = 30$ problem. This is done both for non-homogeneous and homogeneous problems. The results are shown in table 1. As expected LP and in particular GH benefit from non-homogeneity both quality and CPU-wise, while for homogeneous problems the NN algorithm is the winner. For larger problem sizes it is not feasible to use the exact BB algorithm. The only thing we can do is to compare the different approximate approaches, NN, SA and LP. The conclusions from problem sizes ranging from 50 to 500 are same as above. The real strength in the NN approach is best exploited for more homogeneous problems. In figs. 3 and 4 we show the performance and CPU consumption for $N \in [50, 500]$ with $M = N$.

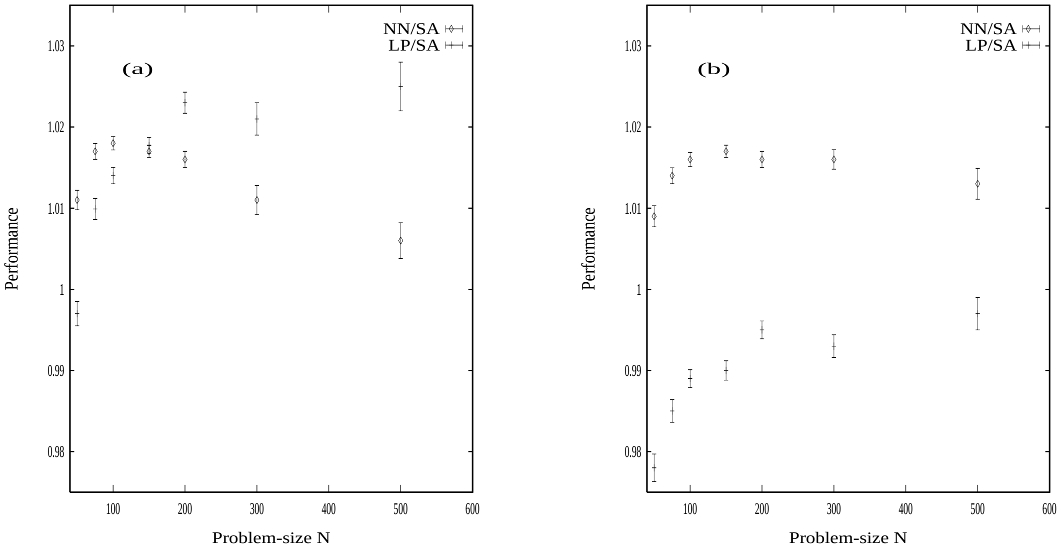

Figure 3: Performance of the neural network (NN) and linear programming approaches (LP) normalized to simulated annealing (SA) for problem sizes ranging from 50 to 500 with $M = N$. (a) c_i=rand[0.45,0.55] and (b) c_i=0.5.

Summary

We have developed a neural mapping and MFT solution method for finding good solutions to combinatorial optimization problems containing inequalities. The approach has been successfully applied to difficult knapsack problems, where it scales like NM. For the hard homogeneous problems the MFT approach is very competitive as compared to other approximate methods, both with respect to solution quality and time consumption. It also compares very well with exact solutions for problem sizes where these are accessible.

In addition, the MFT approach of course has the advantage of being highly parallelizable. This feature was not explored in this work.

In ref. [8] an ANN approach different from ours was applied to the knapsack problem. Since the difficult parameter region was not exploited there, a comparison would not be meaningful.

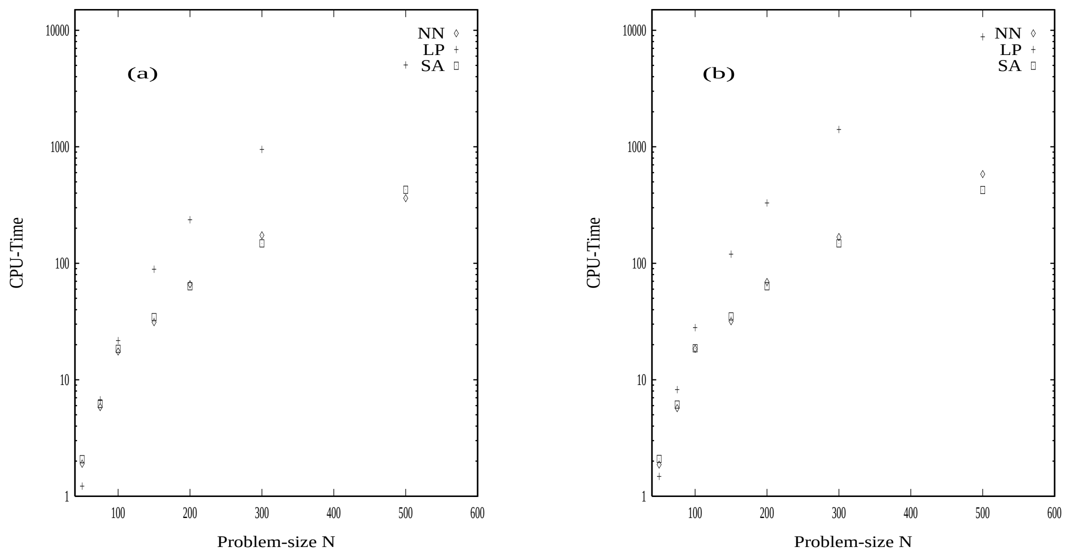

Figure 4: CPU consumption of the neural network (NN) and linear programming approaches (LP) normalized to simulated annealing (SA) for problem sizes ranging from 50 to 500 with $M = N$. (a) c_i=rand[0.45,0.55] and (b) c_i=0.5. The numbers refer to DEC3100 workstations.

References

[1] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics* **52**, 141 (1985).

[2] C. Peterson and B. Söderberg, "A New Method for Mapping Optimization Problems onto Neural Networks", *International Journal of Neural Systems* **1**, 3 (1989).

[3] C. Peterson, "Parallel Distributed Approaches to Combinatorial Optimization", *Neural Computation* **2**, 261 (1990).

[4] L. Gislén, B. Söderberg and C. Peterson, "Teachers and Classes with Neural Networks", *International Journal of Neural Systems* **1**, 167 (1989).

[5] L. Gislén, B. Söderberg and C. Peterson, "Scheduling High Schools with Neural Networks", Lund Preprint LU TP 91-9 (submitted to *Neural Computation*) (1991).

[6] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", *Science* **220**, 671 (1983).

[7] See e.g. W.P. Press, B.P Flannery, S.A. Teukolsky and W.T. Vettering, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge, (1986).

[8] V.V. Vinod, S. Ghose and P.P. Chakrabarti, "Resultant Projection Neural Networks for Optimization Under Inequality Constraints", Kharagpur Department of Computer Science Preprint (1990).