A Neural Network Approach to the Crossword Puzzle Problem

Zafer Barutçuoğlu

Department of Mathematics
Boğaziçi University, 80815 Bebek, İstanbul, Turkey
barutcuoglu@usa.net

Abstract

This paper proposes a connectionist solution for the construction of crossword puzzles. Based on a word-based approach, the problem is expressed as a constraint satisfaction problem. Taking advantage of a double-update scheme, neural networks based on the Boltzmann and Cauchy machine models are implemented for the problem, and performances are compared.

1. Introduction

The construction of crossword puzzles poses a difficult problem to humans and computers alike. Construction by hand is a task often limited to professionals and dedicated enthusiasts, since it requires extensive experience.

The idea of using computers in crossword puzzle construction is not new; indeed research in this direction dates back to 1960. Since then, many diverse attempts have been made to tackle the problem. Most of these approaches depend on backtracking-based search techniques, which assume a centralized control, making a parallel implementation difficult.

When the problem is viewed as a constraint satisfaction problem, it is possible to apply neural network models that have proven suitable for such problems. The parallelism that comes with connectionist approaches makes distributed implementations possible. Also, the advances in VLSI allow for inexpensive hardware implementations of neural systems, and thus, dedicated hardware for crossword puzzle construction may be considered.

2. Previous Work

The research in crossword puzzle construction can be said to have started with E. S. Spiegelthal's program for constructing double-crostics, a crossword variant [1].

The computer construction of crossword puzzles as we know them has been first studied by L. J. Mazlack and O. Feger. Mazlack defined two basic approaches for construction: letter-by-letter and word-by-word. He himself chose the former approach, and constructed some simple puzzles using precedence relationships [2][3]. Feger used a larger word list and got better results [4].

P. D. Smith and S. D. Steen used a search scheme with an estimation-based heuristic. They were also the first to use the basic terminology of crossword puzzle research today [5].

A logic-based approach to the crossword puzzle was given by H. Berghel. It was shown that any grid and word list could be expressed as a set of Horn clauses [6]. Extending this work, Berghel and C. Yi developed a crossword-compiler-compiler, a program that given a word list and a grid, outputs a program that in turn produces a crossword puzzle with the given grid and list [7].

J. M. Wilson expressed the problem as pure 0-1 integer programming problems for both the letter-based and the word-based approaches [8]. We will further examine Wilson's models.

Ginsberg *et al.* applied a number of heuristics from artificial intelligence to the basic backtracking search, favoring different factors in choosing moves [9].

S. C. Jensen's thesis on crossword puzzles discusses and benchmarks existing word-based and letter-based approaches, and proposes a hybrid scheme [10].

As a variation of the basic problem, *unconstrained* crossword puzzles were also explored, ones where the grid structure is

determined dynamically by the program [11][12][13].

3. The Constraint Satisfaction Problem

The constraint satisfaction problem (CSP) consists of finding an assignment of values to a set of variables while satisfying a set of constraints [14].

Each variable X_i must be assigned a value from its domain D_i whose *l*-th element will be denoted as d_{il}. The assignments of a valid solution should comply with a set of given constraints.

Constraints in general are *n*-tuples that specify disallowed simultaneous assignments. However, since constraints of all dimensions can be represented in binary constraints by the inclusion of additional variables, and since simple neural network connections connect two units, we will assume that constraints are binary, i.e., each constraint is of the form (d_{im}, d_{jn}), meaning that the *m*-th value of D_i and *n*-th value of D_j cannot be simultaneously assigned to the variables X_i and X_j respectively.

4. Neural Networks for CSPs

Neural network solutions to CSPs are generally based on Hopfield-type networks [15][16][17][18]. Since simple Hopfield nets typically suffer from poor local minima, simulated annealing is used with the Boltzmann machine and its variants [19][20][21].

4.1. Hopfield-type Neural Networks

In the Hopfield-type network model, each network unit has a binary activation state, and symmetric weighted connections to the other units. The energy to be minimized corresponding to a network state is defined as the negative sum of the weights of connections between mutually active units [22]. Units are encouraged to be active through positive self-connection weights. In some conventions, self-connections are taken as zero, but threshold values are introduced, which in practice have the same effect. In our discussion we will prefer self-connections for compactness of notation.

In a binary Hopfield-type network with M units, the energy at the network state $\vec{y} = (y_1, \ldots, y_M)$ is:

$$E(\vec{y}) = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j w_{ij}$$

where each $y_i \in \{0,1\}$ denotes the activation state of the *i*-th unit, and w_{ij} is the weight of the connection between the *i*-th and *j*-th units.

The operation of the network is that at each *trial*, one unit is selected randomly, and the change in the network energy that will result if that unit changes state is calculated:

$$\partial E_i(\vec{y}) = (2y_i - 1)\left(\sum_{j=1}^{M} y_j w_{ji}\right).$$

In the actual Hopfield network, the change is accepted (an *update* is performed) if the energy will decrease. Thus the network performs simple gradient descent, and converges at a local –but possibly not global– minimum of the energy function. However, especially in our constraint satisfaction perspective, local minima occur frequently, while the objective is to attain the global minimum, so the pure Hopfield model is often inconvenient.

The Boltzmann machine is an improvement over the Hopfield network that is based on applying an annealing schedule to the energy [19]. Acceptance of new states is no more deterministic, but is based on a Boltzmann probability distribution. The probability of acceptance is:

$$P_B = \frac{1}{1+\exp\left(\frac{\Delta E}{T}\right)}$$

where T is the *temperature* parameter is introduced, which determines the tendency of the network to discriminate between transitions that increase energy and those that decrease it. Starting at an initial high temperature, the network is gradually "cooled" from unstable hence widely distributed states to stable minima. As the temperature tends to zero, the behavior more closely resembles the pure Hopfield network, but the probabilistic unstability allows the network to escape poor local minima while at high temperatures. The cooling schedule should be according to the formula

$$T_k = \frac{T_0}{\ln(1+k)}$$

where T_0 is the initial temperature, and k is the number of *epoch*s, each of which consists of M trials.

A modification to the Boltzmann machine is the use of the Cauchy probability distribution instead of the Boltzmann distribution, in which case the method is known as *fast simulated annealing*, or the Cauchy machine [21]. The Cauchy acceptance probability is:

$$P_C = \frac{1}{2} - \frac{1}{\pi}\arctan\left(\frac{\Delta E}{T}\right).$$

The infinite variance of the Cauchy distribution allows greater flexibility for large changes to avoid local minima. Another advantage of the Cauchy machine is that a faster cooling schedule than the Boltzmann machine can be used [23], and indeed is observed to be necessary to help the net stabilize [24]:

$$T_k = \frac{T_0}{1+k^{\alpha}}, \text{ where } 1 \le \alpha < 2.$$

4.2. A Neural Model for the CSP

In the typical Hopfield-type neural network solution to the CSP, each network unit represents a value of a variable. The network can be visualized in rows of units where each row corresponds to a variable, and the units in the row are the values that the variable can take.

Since a variable logically can have only one value, the intermediate network states where more than one node is active for a variable are obviously inconsistent with the model. To decrease the number of intermediate network states, a *double-update* model is used. Based on the *group-update* notion [25], this method ensures that the network moves only between consistent states.

The double-update scheme requires that the network is initially in a consistent state, i.e., exactly one unit is active for every variable. Then for each trial, two units for the same variable are chosen, one of which must be the active unit for the variable. The energy change is calculated for the state where both units are altered (the other unit becomes the active unit), so if the new state is accepted, the network is still in a consistent state. The double-update energy difference for row n can be computed as:

$$\Delta E_j^n(\vec{y}) = \sum_{m \neq n} w_{m'n'} - \sum_{m \neq n} w_{m'j}$$

where m' is the active unit in row m, n' is the active unit in row n, and j is an inactive unit in row n being tested for update.

This scheme has been shown to lead to a fast and efficient search of the problem state space [26].

5. The Crossword Puzzle Problem

The Crossword Puzzle Problem (CPP) is, given a list of words (the *dictionary*) and a fixed empty grid (with or without black squares), the job of filling the grid with letters such that every horizontal or vertical word formed is in the dictionary, and no word is used twice.

For a constraint satisfaction model for the CPP, let us examine Wilson's binary integer models.

For the word-based case, the grid is thought of as a set of fixed-length word-slots. The binary variables are defined as:

$$z_{nk} = \begin{cases} 1 & \text{if word } k \text{ is in slot } n \\ 0 & \text{otherwise} \end{cases}$$

The constraints are:

(1) $\sum_n z_{nk} \le 1$ for each k

(2) $\sum_k z_{nk} = 1$ for each n

(3) $\sum_{k \in S_{jnp}} z_{nk} = \sum_{k \in S_{jpn}} z_{pk}$

where S_{jnp} is the set of words that fit in the vertical slot n which have the letter j in the cell where the vertical slot n intersects with the horizontal slot p, and S_{jpn} denotes the set of words allocable in the horizontal slot p which have the letter j in the cell where the horizontal slot p intersects with the vertical slot n [8]. This translates as the natural constraint that words assigned to intersecting slots must have the same letter at the cell of intersection.

Wilson's other integer model is based on the letter-based approach to the problem. While the letter-based model has the advantage of a small number of variables (hence small network size) independent of word list size,

Wilson's is indeed not a true model for the CPP, in that it does not disallow words to be used more than once. Actually the model is logically less constrained than the problem, and includes invalid solutions. Since we are in search of a correct foundation, this model does not meet our needs.

A true letter-based solution to the problem cannot be achieved without sacrificing the convenience of binary constraints, which will in turn call for additional network nodes, or nodes with multiple link-areas [27], which again increase network complexity in effect similar to the word-based case. Therefore we will use the word-based model in our constraint satisfaction approach.

Wilson's second constraint indicates the nature of a CSP, in that the slots become the variables, and the words become the values. The first and third constraints will form the set of constraints for the CSP.

Now we can formulate the CPP as a CSP as follows:

The variables X_i are defined as the word-slots in some arbitrary ordering. The values in the domain D_i of X_i correspond to all dictionary words of the same length as the i-th slot. The set of constraints C is defined such that $(d_{in},d_{jm})\in C$ if:

(a) d_{in} and d_{jm} correspond to the same word in the dictionary, or
(b) the i-th and j-th slots intersect, but the letters of the words corresponding to d_{in} and d_{jm} at the point of intersection are not the same.

In the light of this formulation, we can now apply a neural framework to the problem.

6. Neural Networks for the CPP

Since we expressed the CPP as a binary CSP, the neural network models previously discussed for CSPs can be applied. However, a closer inspection of the problem provides some case-specific information for improving the design and performance of the network.

With the use of the double-update scheme, it is ensured that each row contains exactly one active unit at any given time. Therefore, the self-connections introduced to encourage units to be active, and the connections among units of the same row are obsolete, and can be taken as zero. Thus, all remaining non-zero connections will serve as either invalid-intersection or multiple-use-of-word penalties, and can have constant negative values for each case.

In an $n\times m$ grid with b black cells, the number of word slots bounded above and coarsely approximated by $m+n+2b$ [13]. For a minimal American-type puzzle of commercial quality, let us assume a 10×10 grid with 15 black cells, and a dictionary of words up to 10 letters. If there are about 1000 words of each length, we have a network of $(10+10+2\times15)\times1000 = 50000$ units. The weight matrix for this network would be huge. However, all non-zero weights can be determined on-line by looking up the words from the dictionary when necessary and checking slot intersections, so we need not store a weight matrix.

Furthermore, since horizontal words are only associated with vertical words and vice versa, they need not have connections among themselves. We can then visualize the network as a bipartite of horizontal and vertical units. This allows us to approximately halve the cost of calculating the energy change.

A minor variation to the standard CPP is when the initial grid may have existing letters. This is called the *protected* CPP [10] and is used especially in commercial applications. Our model also accommodates the protected CPP, simply by excluding from the network the units which represent word assignments that disagree with the existing letters.

7. Experimental Results

The proposed crossword puzzle network scheme was implemented as both a Boltzmann machine and a Cauchy machine.

Taking advantage of the double-update scheme, the networks were stored as integers for each row, representing the index of the active unit of the row.

Both invalid-intersection and multiple-use-of-word penalty weights were taken as -1.0, and calculated on-line through the grid and the dictionary. Since there are no self-connections, the global minimum of zero represents a valid solution.

The initial temperatures were set so that for both networks the initial probability of accepting an energy change of 1.0 was around 0.4. For the Cauchy machine, the parameter α was taken as 1.

A 5×5 complete puzzle (one without black cells) was chosen for testing, using different dictionary sizes. It was ensured by full search that each dictionary had exactly one solution (and naturally its transpose).

Both networks were tested for each scenario until a solution was found. 20 experiments were conducted for each case, and the numbers of epochs that took each network to find a solution are summarized in Table 1.

Words	Boltzmann		Cauchy	
	avg	stddev	avg	stddev
10	25	8	14	6
100	244	132	98	61
1000	1423	962	810	316

Table 1: Epochs until a solution was found

The Cauchy machine required fewer epochs as expected. Most epochs of the Boltzmann machine was spent in poor local minima.

The actual durations are dependent on the particular hardware and software. On a system with a single 200 MHz processor, despite the low-performance high-level language used, none of the experiments exceeded an hour.

8. Conclusion

Expressing the crossword puzzle problem as a constraint satisfaction problem, existing Hopfield-type neural networks suitable for such problems were considered. Using the benefits of a double-update method, Boltzmann and Cauchy machines were constructed for the proposed solution scheme.

In the experiments conducted, a solution was found in all cases. Performances were within practical time limits, in spite of the serial and unoptimized implementation.

With a neural network solution to the problem, parallel implementations may be produced, which would have considerably superior performance. Looking at the recent advances in neural network circuitry, even low-cost dedicated hardware may be possible for crossword puzzles.

However, neural networks can converge only to one solution, so cannot return all solutions associated with a given scenario.

Also, compared to traditional backtracking search techniques, the neural network solution lacks complete reliability, due to its stochastic nature. We cannot guarantee convergence to a valid solution. This aspect seriously reduces the chances of commercial popularity, but can be remedied by a very efficient (possibly parallel) implementation, in which the time cost of re-running the network when converged to a poor local minimum will be practically insignificant.

9. References

[1] E. S. Spiegelthal, "Redundancy Exploitation in the Computer Construction of Double-crostics", Proceedings of the EJCC, pp. 39-56, 1960.

[2] L. J. Mazlack, "Computer Construction of Crossword Puzzles Using Precedence Relationships", Artificial Intelligence, Vol. 7, No. 1, pp. 1-19, 1976.

[3] L. J. Mazlack, "Machine Selection of Elements in Crossword Puzzles", *SIAM Journal of Computing*, Vol. 5, No. 1, pp. 51-72, 1976.

[4] O. Feger, "A Program for the Construction of Crossword Puzzles", *Angewandte Informatik*, Vol. 17, No. 5, pp. 189-195, 1975.

[5] P. D. Smith and S. Y. Steen, "A Prototype Crossword Compiler", *The Computer Journal*, Vol. 24, No. 2, pp. 107-111, 1981.

[6] H. Berghel, "Crossword Compilation with Horn Clauses", The Computer Journal, Vol. 30, No. 2, pp. 183-188, 1987.

[7] H. Berghel and C. Yi, "Crossword Compiler-Compilation", *The Computer Journal*, Vol. 32, No. 3, pp. 276-280, 1989.

[8] J. M. Wilson, "Crossword Compilation Using Integer Programming", *The Computer Journal*, Vol. 32, No. 3, pp. 273-275, 1989.

[9] M. L. Ginsberg *et al.*, "Search Lessons Learned from Crossword Puzzles", *Proceedings of AAAI 90*, 1990.

[10] S. C. Jensen, "Design and Implementation of Crossword Compilation Programs Using Sequential Approaches", M. S. Thesis, Odense University, Denmark, 1997.

[11] G. H. Harris, "Generation of Solution Sets for Unconstrained Crossword Puzzles", *Symposium on Applied Computry*, TH037-9/90/0000/0214 Copyright 1990 IEEE.

[12] İ. Berker and C. Say, "A Crossword Puzzle Generator for Turkish", *Proceedings of the 8th International Symposium on Computer and Information Sciences*, 1993.

[13] Z. Barutçuoğlu, "Automated Generation of Unconstrained Crossword Puzzles and an Estimate of Their Solution Space", *Proceedings of the 12th International Symposium on Computer and Information Sciences*, 1997.

[14] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, 1993.

[15] J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, No. 52, pp. 141-152, 1985.

[16] G. Tagliarini and E. Page, "Solving Constraint Satisfaction Problems with Neural Networks", *Proceedings of the International Conference on Neural Networks*, San Diego, CA, 1987, Vol. 3, pp. 741-747.

[17] H. Adorf and M. Johnston, "A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems", *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, 1990, Vol. 3, pp. 917-924.

[18] P. Bourret and C. Gaspin, "A Neural Based Approach of Constraints Satisfaction Problems", *Proceedings of IJCNN92*, Baltimore, 1992.

[19] G. E. Hinton, T. J. Sejnowski and D. H. Ackley, "Boltzmann Machines: Constraint Satisfaction Machines That Learn", Technical Report CMU-CS-84-119, Carnegie-Mellon University, 1984.

[20] Y. Akiyama *et al.*, "The Gaussian Machine: A Stochastic Neural Network for Solving Assignment Problems", *Journal of Neural Network Computing*, pp. 43-51, Winter 1991.

[21] H. H. Szu and R. Hartley, "Fast Simulated Annealing", *Physics Letters*, pp. 157-162, 1987.

[22] J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences USA*, No. 79, 1982, pp. 2554-2558.

[23] H. Jeong and J. H. Park, "Lower Bounds of Annealing Schedule for Boltzmann and Cauchy Machines", *International Joint Conference on Neural Networks*, Washington, DC, 1989, Vol. 1, pp. 581-586.

[24] H. H. Szu, "Colored Noise Annealing Benchmark by Exhaustive Solutions of TSP", *International Joint Conference on Neural Networks*, Washington, DC, 1990, Vol. 1, pp. 317-320.

[25] A. Likas and A. Stafylopatis, "Group Updates and Multiscaling: An Efficient Neural Network Approach to Combinatorial Optimization", *IEEE Transactions on Systems, Man and Cybernetics*, 1996.

[26] A. Likas, G. Papageorgiou and A. Stafylopatis, "A Connectionist Approach for Solving Large Constraint Satisfaction Problems", *Applied Intelligence*, Vol. 7, pp. 215-225, 1997.

[27] C. Gaspin, "Automatic Translation of Constraints for Solving Optimization Problems by Neural Networks", *Proceedings of IJCNN90*, San Diego, 1990.