A NEW DYNAMICAL APPROACH TO THE TRAVELLING SALESMAN PROBLEM

W. BANZHAF
Institut für Theoretische Physik und Synergetik, Universität Stuttgart, Pfaffenwaldring 57/IV, D-7000 Stuttgart 80, FRG

Received 17 May 1988; revised manuscript received 22 November 1988; accepted for publication 18 January 1989
Communicated by A.P. Fordy

We present a new dynamical method to solve problems of combinatorial optimization. The basis units (artificial neurons) of a network generate a competitive dynamics by their two-dimensional interactions. Based on that specific dynamics the system uses much the same information as human beings to reduce the solution space of the problem. The method is applied to the TSP and compared to conventional approaches.

1. Introduction

Recently, a couple of physical methods were found to solve optimization problems. These include simulated annealing [1], evolutionary search strategies [2–4] and neural network algorithms [5]. The neural network approach is of special interest, since it simultaneously has the potential to be realized in cheap hardware.

Our own work on this problem was triggered by a paper of Wilson and Pawley [6] who reported on considerable difficulties in applying the neural net algorithm of Hopfield and Tank [5] on the travelling salesman problem. Although this algorithm may suffer from serious deficiencies, its authors nevertheless pointed to one decisive issue in the game, namely the usage of a dynamical system to solve optimization problems.

In the following Letter we want to follow this general intention. We use, however, another dynamical system which was proposed recently in the context of pattern recognition and associative memory [7].

By introducing a clear separation between coding the problem and the dynamics "solving" it we avoid some of the sources of trouble one encounters in the algorithm of Hopfield and Tank. As in ref. [5], the dynamics is driven by minimization of a potential or energy function. In ref. [6], the authors have shown – at least in our interpretation – that an additive mixture of "energies" coming from the problem domain, i.e. distances between cities, and energies coming from the intrinsic dynamics domain, i.e. constraints necessary for the system to converge to one of the desired states, is problematic.

Moreover, one may ask quite generally, whether a gradient method as that of Hopfield and Tank is, will ever give good results, if the only choice one has at hand is that of initial conditions (of uncertainty). Presumably, there should exist numerous local minima in the energy landscape complicating the search considerably. The standard argument is that at the beginning of the dynamical process to solve the TSP no path is selected definitively. Starting from a fuzzy state, the system chooses under the influence of "distance energies" one of the more favourable tours. As is reported in ref. [6], however, in a large portion of cases no tour at all results from the dynamics under consideration. This precisely reflects the fact of "antagonistic" forces being present due to distance energies on the one hand and constraint violation energies on the other.

Besides this potential non-convergence of the algorithm there remains another serious problem: Loading connections (the synaptic matrix) with corresponding data into the neural net is very time consuming (of the order $O(N^4)$) if it has to be done for every city configuration separately [8].

Here, we take another point of view in clearly separating dynamics and problem conditions. We map the travelling salesman problem onto a two-dimen-

0375-9601/89/$ 03.50

sional grid of units performing a competitive dynamics in two dimensions. As order parameters which will allow us to encode the problem appropriately we use something like inverse distances between cities. These inverse distances will determine the initial state of a dynamical system which then relaxes to its equilibrium state. The consequence of this relaxation is that the dynamical system extracts short paths or neighborhood relations between cities.

After relaxation, the problem conditions or constraints are checked. If the proposed solution is not feasible, i.e. no unique tour results, a new run of the dynamics is prepared in such a way as to approach the constraints successively. A recursive application of this algorithm will finally lead to a feasible solution. The decoding of the result will give us in general a good solution to the TSP problem.

Thus, the *overall structure* of the problem to be solved is mapped onto appropriate *interactions* between components of a dynamical system – in this case to amplify certain quantities – whereas the *specifically posed* problem is used as *initial condition* for the time development of that system. This is a general method which can be applied to several optimization problems given an appropriate coding.

The proposed procedure is suitable for parallel computers as well as for special hardware. Our simulations, however, were obtained on a serial machine with a discretized version of the algorithm.

The paper is organized as follows: In section 2, a brief sketch of the one-dimensional competitive system is followed by the two-dimensional generalization. Section 3 discusses the precautions to be taken if the system is applied to the travelling salesman problem. In section 4 we report on simulations of 100-city-problems. Section 5 compares the results to other methods and comments on it.

2. The algorithm

We first recall the one-dimensional case. The problem is to find the greatest component of a vector c_k, $k=1,...,N$. We use the vector c_k as initial conditions $d_k(0)$ of a dynamical system described by a vector $d_k(t)$, $k=1,...,N$. For definiteness, let $0 \leqslant c_k,\ d_k(t) \leqslant 1$. A dynamical system which amplifies the greatest $d_k(0)$, $d_{k_{max}}(0) = \max\{d_k(0), k=1,..., N\}$ until it saturates at $d_{k_{max}}(t\to\infty)=1$ and suppresses all others ($d_k(t\to\infty)=0$, $k\neq k_{max}$) (cf. fig. 1a) was formulated in ref. [7]. The interaction between different d_k's is competitive and a single $d_k \neq 0$ will survive in the long run. Laws of this kind are very common in nature [9]. Recently, they were applied to pattern recognition tasks with considerable success [7,10].

A natural extension to competition in two dimensions is written in the following way,

$$\begin{aligned}\dot{d}_{i,j}(t) &= d_{i,j}(t)\Big(1-2\sum_{i'\neq i} d^2_{i',j}(t) \\ &\quad -2\sum_{j'\neq j} d^2_{i,j'}(t) - d^2_{i,j}(t)\Big) \\ &= d_{i,j}(t)\Big(1-\sum_{i'\neq i} d^2_{i',j}(t) - \sum_{i'} d^2_{i',j}(t) \\ &\quad -\sum_{j'\neq j} d^2_{i,j'}(t) - \sum_{j'} d^2_{i,j'}(t) + d^2_{i,j}(t)\Big) \\ &= d_{i,j}(t)\Big(1-2\sum_{i'} d^2_{i',j}(t) - 2\sum_{j'} d^2_{i,j'}(t) \\ &\quad +3d^2_{i,j}(t)\Big). \end{aligned} \tag{1}$$

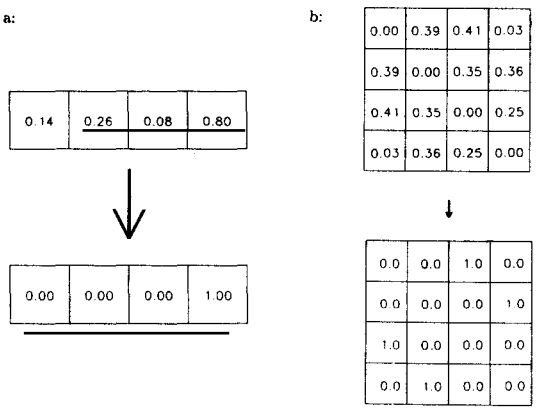

Fig. 1. (a) Behavior of a one-dimensional system amplifying the greatest component of a vector. (b) Schematical sketch of a two-dimensional system. The resulting matrix of states has a single "1" in every row and column.

Equilibrium is reached for

(a) $d_{i,j}=0$,

(b) $d_{i,j}=1$, $d_{i',j}=d_{i,j'}=0$ $\forall i'\neq i, j'\neq j$.

Eq. (1) follows as a gradient dynamics from the potential

$$V(d_{i,j})=-\tfrac{1}{4}\sum_{i,j} d_{i,j}^2+\tfrac{1}{4}\sum_{i,i',j} d_{i,j}^2 d_{i',j}^2$$

$$+\tfrac{1}{4}\sum_{i,j,j'} d_{i,j}^2 d_{i,j'}^2-\tfrac{1}{8}\sum_{i,j} d_{i,j}^4,$$

$$0\leqslant d_{i,j}(t)\leqslant 1\,. \qquad (2)$$

Here, we had to introduce two different indices for rows and columns, since competition has to take place only in the corresponding row and column. Thus, the competitive dynamical system performs operations on a grid of the $d_{i,j}$'s and changes their values depending on other $d_{i,j}$'s.

The dynamics ensures that in every row and column only one element dominates suppressing the others (cf. fig. 1b). It has a similar function as the neuron dynamics in Hopfield and Tank's solution to the task assignment problem [12]. Not in all cases the greatest initial value survives because all cells are mutually connected and inhibit each other. Accordingly, a change in one cell's state has consequences for all others. Fig. 2 shows a typical example of the process.

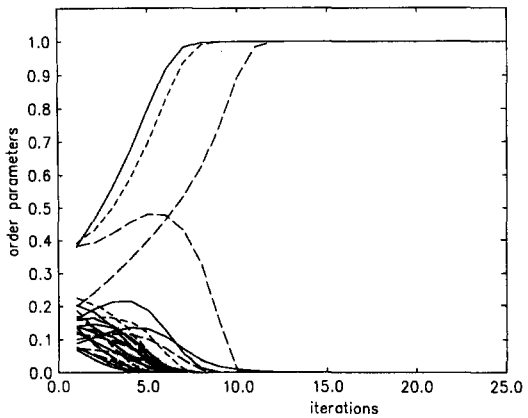

Fig. 2. Typical behavior of the dynamics (only 3 rows shown): In each row and column one component gets amplified. Due to interactions sometimes a smaller component dominates.

3. Application to the TSP

The N-dimensional travelling salesman problem consists of finding the shortest path through N cities visiting each city once and returning to the starting point. The two-dimensional competitive dynamics introduced above will serve us now to reduce the solution space of the N-dimensional TSP drastically. Note first that a $N\times N$ matrix A (the adjacency matrix) is able to represent the $N!$ possible tours of a travelling salesman visiting N cities [11]. We could use the dynamical system to generate an adjacency matrix if an appropriate coding could be found. To this end, we consider the heuristic search procedure humans are applying if confronted with the same problem. Roughly, then we perform the following operations

(i) Connect cities with small distances,

(ii) fulfil the constraint of a closed tour,

(iii) minimize tour length by trial-and-error.

We shall follow this heuristic closely. Fig. 3 shows the overall system composed of a coding layer, the central dynamic layer and a decoding layer. We start with the city-coordinates. The coding layer transforms this information in two steps into initial states of a two-dimensional competitive dynamical system with N rows and N columns.

First, a table of distances between all cities is generated. Since only a relative measure is needed, the distances are normalized by dividing through the minimal distance in the whole problem,

$$e_{i,j}=\frac{\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}{\min_{i',j'}\sqrt{(x_{i'}-x_{j'})^2+(y_{i'}-y_{j'})^2}}\,. \qquad (3)$$

Secondly, a transformation to the inverse normalized distances is performed,

$$d_{i,j}=\frac{1}{e_{i,j}+c}, \quad \text{if } e_{i,j}\neq 0,$$

$$=0, \qquad \text{else}, \qquad (4)$$

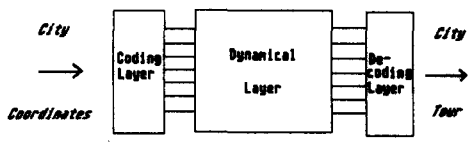

Fig. 3. The overall system. City coordinates are transformed by a coding layer into initial conditions of a dynamical layer. After relaxation, a decoding of the stationary state results in a city tour.

which will be given as the starting state $d_{i,j}(0)$ to our dynamical system. c is a cut-off parameter regulating the maximally allowed value [#1] of $d_{i,j}(0)$.

The inverse distances $d_{i,j}$ quantify the proximity of cities to each other. Indeed, they are used by human problem solvers to largely reduce alternative choices among tours. By $d_{i,j}$, in principle, every city is connected to every other one, thus we can interpret them as possibly realized (fuzzy) paths in a solution to the TSP at hand and allow its time development. The dynamical system is constructed to isolate and "amplify" small distances among cities.

Due to the symmetric character of the matrix $(D)_{i,j}=d_{i,j}$ two cities must agree in choosing each other as of shortest distance. Beginning with $d_{i,j}(0)$, at the end of the so-called phase I of the dynamics every city is then connected to its nearest neighbor if this was not inhibited by another cell. But we want each city to be connected to two neighbors. Consequently, we have to go through the dynamical stage twice. In a second phase, one starts again with the original initial values of inverse distances where, however, the nearest connections resulting from phase I are destroyed. This ensures that after running dynamics a second time in phase II each city is connected to the next nearest neighbor, too, if this was not inhibited by other cells.

In the decoding layer results of phases I and II are considered simultaneously. Now, every city is connected to two other cities whereby one or more closed paths are generated. The latter constitutes a difficulty since more than one closed path (i.e. appearance of loops) is not a feasible solution to the travelling salesman problem. Although the solution in terms of overall length is quite good, it is only feasible for the assignment problem.

A closer look at the criteria for non-feasible solutions gives the following picture:

To result in more than one closed loop two conditions must be fulfilled:

(i) The number of participating cities in each loop N_i, $i=1,\ldots, n_{\text{Loop}}$, must be even and greater than 2, i.e. $4 \leqslant N_i \leqslant N \ \forall i$.

(ii) Each city may participate only in one loop,

$$\sum_{i=1}^{n_{\text{Loop}}} N_i = N .$$

An unfavourable city-distribution will then lead in the second dynamical phase to two or more loops.

A method to cure this problem is to disconnect in every closed loop the largest connection between cities, to reload the $d_{i,j}(0)$ left for another run and to allow all but the disconnected paths. Applying this procedure recursively one finally ends up with a feasible tour. Together with a subsequent local improvement of the solution, we call this postprocessing.

4. Simulation results

The simulations presented here are done on 100-city TSPs. The cities were distributed randomly on a unit square. Data were accumulated with 100 different city-configurations. To solve eq. (1) time was discretized and a Newton approximation was applied.

Fig. 4 sketches a selection of typical results for two configurations. After obtaining the unfeasible solutions (fig. 4a), a recursive application of the algorithm generates a feasible tour with obvious shortcomings (fig. 4b). The postprocessing by a local search strategy (2-opt), however, eliminates these flaws rather effectively (fig. 4c).

Table 1 shows the outcome of the dynamical phases I and II in terms of resulting loops, added for all configurations. Fig. 5 displays all runs compared to a purely local search by 2-opt and a simple greedy algorithm starting with an arbitrary city. Table 2 summarizes these results and compares them to the theoretical limit of $t=7.49$ [13]. It turns out that the algorithm converges in general to a tour slightly longer than the two simpler algorithms. The local part of the postprocessing, however, improves the results considerably. The latter might be identified with the trial-and-error phase of a human problem solver. We expect the postprocessing to be necessary in nearly all cases due to the local nature of the dynamics applied.

An interesting aspect is the saturation behavior of variables $d_{i,j}(t)$. If one plots the number of variables not saturated over the number of iterations a phase transition seems to be visible. It may be said that the system undergoes a phase transition from its *unde-*

[#1] This value is $d_{i,j}=1/(1+c)$.

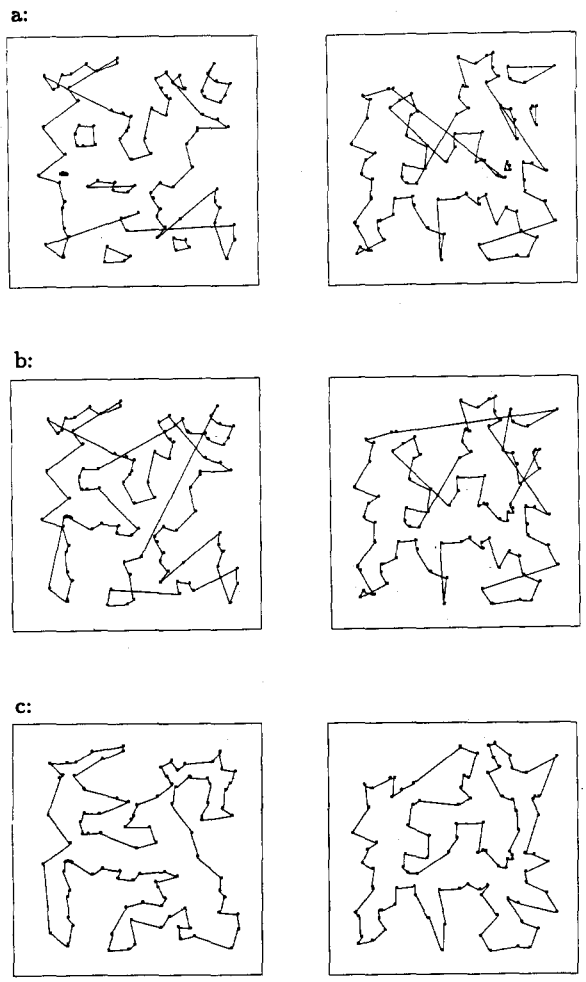

Fig. 4. Two samples of the results. (a) After phases I and II. (b) After application of the recursive algorithm. (c) After an additional local search by 2-opt.

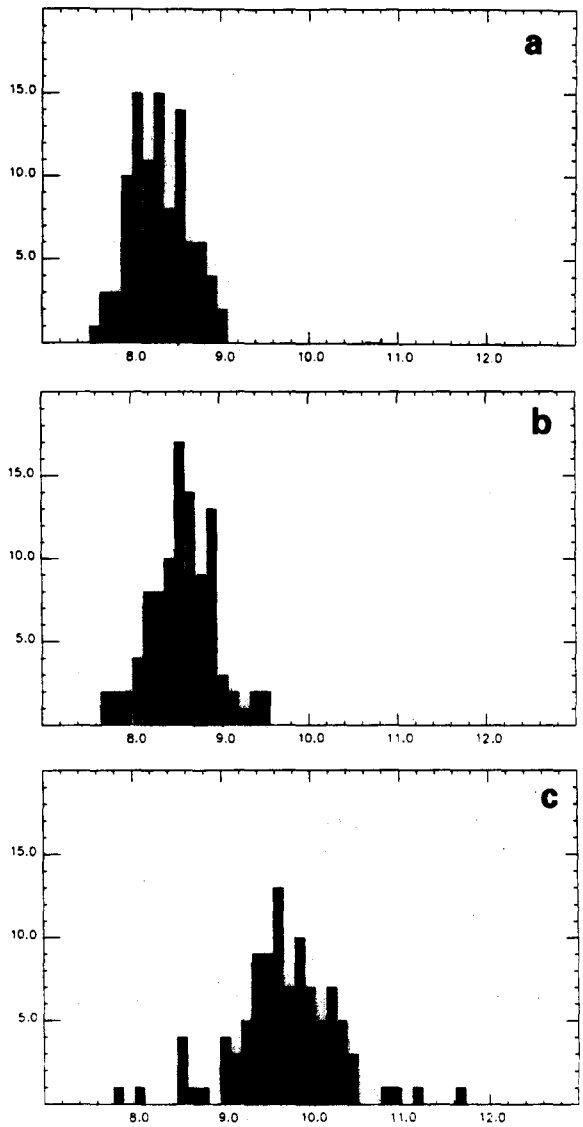

Fig. 5. Resulting tour length for 100 100-city-TSPs. (a) Two-dimensional competitive dynamics including postprocessing. (b) 2-opt. (c) Greedy algorithm.

Table 1
Loops of N_i cities appearing in 100 100-city-TSPs.

N_i	n	N_i	n
4	323	20	3
6	89	22	4
8	45	24	6
10	19	26	4
12	14	28	1
14	12	30	2
16	9	34	2
18	5	42	1

cided state (no variable saturated) to its *decided* state (all variables saturated). Fig. 6 exemplifies this behavior using a sample of 100 random 10-city configurations. The earlier competition sets in, the faster the problem is solved. It follows that one can accelerate the process by starting the system near the onset of competition.

Table 2
Comparison of 100 runs with identical city configurations.

	Comp. dyn.	2-opt	Greedy
Configuration 1 [a]	7.98	8.71	9.44
Configuration 2 [a]	8.27	8.80	9.26
Average [b]	8.32	8.59	9.71
Better solutions [c]	84	16	0

[a] Cf. fig. 5. [b] Over 100 configurations.
[c] Compared to the other two methods.

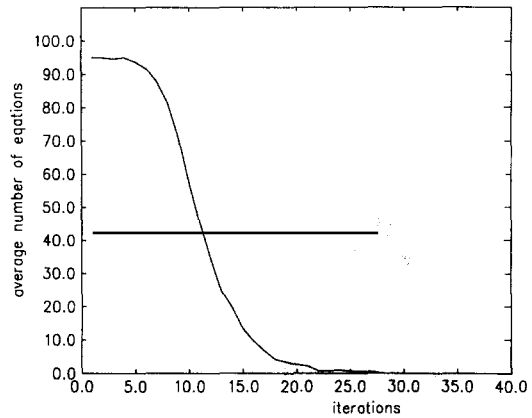

Fig. 6. Saturation behavior in random city configurations. The sharp transition in the number of equations iterated (=unsaturated variables) indicates a phase transition.

5. Discussion

Other dynamical laws are likewise reasonable. One particular candidate would be

$$\dot{d}_{i,j}(t)=d_{i,j}(t)\left[\alpha\left(1-\sum_{i'\neq i}d^2_{i',j}(t)-\sum_{j'\neq j}d^2_{i,j'}(t)\right)\right.$$

$$\left.+\beta\left(1-\sum_{i'\neq i}d^2_{i',j}(t)\right)+\gamma\left(1-\sum_{j'\neq j}d^2_{i,j'}(t)\right)\right], \quad (5)$$

$$\alpha=\frac{1}{N}\sum_{i',j'}d^2_{i',j'}(t)\,, \quad (6)$$

$$\beta=\gamma=1-\frac{1}{N}\sum_{i',j'}d^2_{i',j'}(t)\,. \quad (7)$$

Parameters α, β, γ are varying dependent on the global field which is a measure of the overall decision state of the system. Towards the end of the relaxation process this forces the system more and more to the desired state.

The comparison with other algorithms shows an average upgrade in solution quality of at least 3 percent. This result should be seen in the context of the natural parallelizability of the dynamics considered. A special purpose hardware could lead to decision times of the order of *milliseconds* having most of the processing time saved for postprocessing and readout. The computational resources of N^2 processing units could be limited (to say 10^4 simple processors) by processing local regions of the total problem serially. Besides these modifications, a single dynamical run may be useful to solve graph matching problems, and the dynamics as a whole, though without postprocessing, may be used to solve assignment problems (cf. ref. [11]).

Thus, the algorithm demonstrated is not a general solution to the TSP problem. Rather it is another dynamical system capable of collective computation. The difference to other systems used for optimization is that only the general structure of the problem is mapped onto interactions between collective variables whereas the concrete problem itself is presented as initial condition to the network.

This procedure resembles in some aspects the "intelligence amplifier" designed by Ashby in the fifties [14]. Indeed, a system is constructed which has the potential to give a nearly infinite amount of solutions. The overwhelming majority of possible solutions is destabilized under influence of the initial conditions by the particular interaction chosen. In this interpretation, an enormous range of models may be candidates for a solution of various optimization problems.

Acknowledgement

The author is very grateful to Professor H. Haken for his valuable suggestions and for creating in his institute the research atmosphere without which this work would never have been completed.

References

[1] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Science 220 (1983) 671.

[2] T. Boseniuk, W. Ebeling and A. Engel, Phys. Lett. A 125 (1987) 307.
[3] R.M. Brady, Nature 317 (1985) 804.
[4] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer, Parallel Computing 7 (1985) 65.
[5] J.J. Hopfield and D.W. Tank, Biol. Cyb. 52 (1985) 141.
[6] G.V. Wilson and G.S. Pawley, Biol. Cyb. 58 (1988) 63.
[7] H. Haken, in: Computational systems – Natural and artificial, Proc. Elmau Int. Symp. on Synergetics, 1987, ed. H. Haken.
[8] M. Takeda and J.W. Goodman, Appl. Opt., 25 (1986) 3033.
[9] H. Haken, Synergetics, an introduction, 3rd Ed. (Springer, Berlin, 1983).
[10] H. Haken, in: Neural and synergetic computers, Proc. Elmau Int. Workshop on Synergetics, 1988, ed. H. Haken (Springer, Berlin, 1988).
[11] E. Lawler, Combinatorial optimization, networks and matroids, (Holt, Rinehart and Winston, New York, 1976).
[12] J.J. Hopfield and D.W. Tank, Sci. Am. 266 (Dec. 1987) 62.
[13] E. Bonomi and J. Lutton, SIAM Rev. 26 (1984) 551.
[14] R.W. Ashby, in: Automata studies, eds. C.E. Shannon and J.McCarthy (Princeton Univ. Press, Princeton, 1956).