NASA Contractor Report 181798

ICASE REPORT NO. 89-12

ICASE

FAULT-TOLERANCE OF A NEURAL NETWORK SOLVING THE TRAVELING SALESMAN PROBLEM

P. Protzel

D. Palumbo

M. Arras

Contract No. NAS1-18605
February 1989

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

FAULT-TOLERANCE OF A NEURAL NETWORK SOLVING THE TRAVELING SALESMAN PROBLEM

P. Protzel *
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23665

D. Palumbo
Mail Stop 130, NASA Langley Research Center
Hampton, VA 23665

M. Arras*
Dept. of Computer Science, The College of William and Mary
Williamsburg, VA 23185

Abstract

This study presents the results of a fault-injection experiment that simulates a neural network solving the Traveling Salesman Problem (TSP). The network is based on a modified version of Hopfield's and Tank's original method. We define a performance characteristic for the TSP that allows an overall assessment of the solution quality for different city-distributions and problem sizes. Five different 10-, 20-, and 30-city cases are used for the injection of up to 13 simultaneous stuck-at-"0" and stuck-at-"1" faults. The results of more than 4000 simulation-runs show the extreme fault-tolerance of the network, especially with respect to stuck-at-"0" faults. One possible explanation for the overall surprising result is the redundancy of the problem representation.

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665.

1. Introduction

One of the most intriguing characteristics of biological as well as artificial neural networks is their apparently *inherent* fault-tolerance. The fault-tolerance of most conventional systems requires some form of hardware-, software-, and/or time-redundancy, which increases the complexity of the system. That is, it is possible to construct a simpler system, which is not fault-tolerant and has the same performance under fault-free conditions as the fault-tolerant system. The fault-tolerance of neural networks, however, seems to be inseparable from their functional characteristics, which means it is not possible to have a simpler, not fault-tolerant network perform the same task.

Unlike in conventional systems where the fault-tolerance is a carefully planned and calculated design goal, the cause and the degree of the inherent fault-tolerance of different neural networks basically is still an unknown factor. There have been only a few investigations on the fault-tolerance of artificial neural networks (ANNs) that demonstrate the performance degradation under the presence of faults [1,2,3,4]. For example, Anderson (1983) shows the effect of removing connections from a brain-state-in-a-box model with and without intermittent learning [1]. Sejnowski and Rosenberg (1986) and Hinton and Sejnowski (1986) report on relearning and recovery after random perturbations of the weights for a Backpropagation network [4] and the Boltzmann machine [2], respectively. Random weight perturbations were also used by Hutchinson and Koch (1986) to demonstrate the robustness of their resistive network [3].

We also focus our investigations on the fault-tolerance aspect of ANNs with the goal of getting more insight into the correlation between fault-tolerant and functional characteristics of ANNs. In this paper, we describe first results of a fault-injection experiment that simulates an ANN solving the Traveling Salesman Problem (TSP) based on the method of Hopfield and Tank (H&T) (1985) [5]. In order to produce consistently valid tours in the fault-free cases, we had to use a modification of H&T's original equations. Since the performance varies for different city-distributions and initializations, we defined a performance characteristic that can be averaged over different distributions to allow a better assessment of the

solution quality. This definition and simulation results of the fault-free case are described in the next section. The results of the fault-injection experiment are presented in section 3, followed by an interpretation and discussion in section 4, and a summary with conclusions in section 5.

2. Performance Characteristic for the TSP

Hopfield's and Tank's method of 'programming' an ANN to solve the TSP created a new class of applications for ANNs in the area of combinatorial optimization. H&T's TSP-Solver is also an interesting candidate for the investigation of fault-tolerance characteristics because of the way the problem is represented by the network. To describe a closed tour for n cities, n^2 'neurons' are needed and each represents the probability V_{Xi} of city X being in the i-th position of the tour. Thus, the same tour can be mapped onto the network in $2n$ different ways because the starting point and the tour direction are free variables. The considerable redundancy produced by the $2n$-fold degeneracy of the problem representation raises the question how this might affect the performance of the system under the presence of faults.

Since H&T's method to solve the TSP has become widely known and is well documented, we will not describe the details of the original approach; all the necessary information can be found in [5]. While working with H&T's method it became apparent that the performance varies considerably for different city-distributions. These fluctuations make it impossible to prove a point by looking at just one or two city-distributions. But if the performance has to be averaged over many different city-distributions, we need to define a normalized performance index that reflects the quality of a solution. Each city-distributions has two distinct characteristics, the optimal tour length l_{opt} and the average tour length l_{ave} of a sufficient number of randomly generated tours. The quality of a given solution l for a specific city-distribution can be characterized by its relation to the optimal and the average tour length of this city-distribution. We define the quality of a given solution or *tour quality* q as

$$q = \frac{l_{ave} - l}{l_{ave} - l_{opt}} . \qquad (1)$$

Therefore, we get a value $q = 1$ if a given solution is optimal ($l = l_{opt}$), and $q = 0$ if the answer corresponds to an average random tour ($l = l_{ave}$). This definition requires the knowledge of the optimal tour length for each city-distribution. We generated a test set consisting of 10 different city-distributions for each problem size of 10-, 20-, and 30-cities, and 5 different city-distributions for a 50-city and 100-city case. To obtain the optimal tour length for each distribution, we used several runs of the heuristic algorithm of Lin and Kernigham (1973) as described in [6]. Although this is one of the best algorithms for solving the TSP, it does not guarantee to produce *the* optimal answer. If the reference value l_{opt} is not optimal and the network produces an even better tour with $l < l_{opt}$, it results in a value $q > 1$. If, on the other hand, the solution of the network is worse than the average random tour, we get $q < 0$. The average values l_{ave} were obtained by generating 10^4 random tours for each city-distribution.

Another difficulty in assessing the performance is the need for a random initialization [5] and the dependency of the solution on this initialization. We used H&T's method for determining the initial values, but we chose the symmetry-breaking noise δu_{Xi} uniformly in the interval $-10^{-7} \leq \delta u_{Xi} \leq +10^{-7}$. These small values for the noise were sufficient to break the initial symmetry and produced on the average better tours than higher values. In order to take the variation of the solution for different random initializations into account, we performed several initializations for each city-distribution and computed the average tour quality.

Table 1 shows the performance results of the original H&T equations [5] for our test set of city-distributions in comparison to our modification described below. The values in parentheses show the proportion of valid solutions after convergence. Values shown for the 10, 20, and 30 city sizes are averages over 10 different city-distributions and 10 different random initializations each, that is, each value is averaged over 100 simulations. The 50 city size shows averages of 5 different city-distributions and 10 different random initializations, while the single 100 city case shows averages over 5 different cities and 5 different initializations.

We could confirm the observations of Wilson and Pawley (1988) [7] and others [8,9,10] that H&T's original equations do not consistently produce valid tours. The

TABLE 1

Fault-free performance of the methods in terms of the defined tour-quality q and the proportion of valid solutions (in parentheses).

	number of cities				
Method	10	20	30	50	100
Original Hopfield	0.905	0.903	0.851	0.000	-/-
& Tank Method	(0.15)	(0.11)	(0.02)	(0.00)	-/-
Modified	0.836	0.844	0.808	0.817	0.860
Method	(1.00)	(1.00)	(0.99)	(1.00)	(1.00)

proportion of valid solutions in Table 1 declines rapidly as the problem size increases. Since none of the 50 city cases converged to a valid solution, we did not investigate the performance for 100 cities. H&T use the following equations that describe the time evolution of the system that 'solves' the TSP by converging to a local minimum corresponding to a (locally) optimal solution to the TSP [5] :

$$C_{Xi}\frac{du_{Xi}}{dt} = -\frac{u_{Xi}}{R_{Xi}} - A\sum_{j\neq i} V_{Xj} - B\sum_{Y\neq X} V_{Yi} - C\left(\sum_{Y}\sum_{j} V_{Yj} - n\right) - D\sum_{Y\neq X} d_{XY}\left(V_{Y,i+1} + V_{Y,i-1}\right) . \quad (2)$$

These equations frequently produce invalid tours with more than one neuron in a row or column converging to '1'. Adding the term

$$-E\left(\sum_{X} V_{Xi} + \sum_{i} V_{Xi} - 1\right) \quad (3)$$

to the right side of equation (2) introduces an additional inhibition that penalizes the occurrence of those events. With the parameter values A=B=C=E=200 and D=90 we were able to produce consistently valid tours over a wide range of problem sizes. Although Table 1 shows a slightly lower tour quality for the modified version, we observed only one single 30-city case that did not converge to a valid tour. A comparison to other modifications recently reported in the literature (e.g.

[8,10]) is planned, and we are especially interested in investigating a possible correlation between fault-tolerance characteristics and different ways to enforce the constraints.

3. Fault-Injection

Since this type of ANN is readily implementable in hardware, it seems reasonable to consider the components with the highest failure rate as the primary candidates for a fault-injection experiment. These are the operational amplifiers as the active elements of the network, and in our simulations we distinguished between two different kinds of faults, called stuck-at-"0" and stuck-at-"1". As the self-explanatory notation suggests, a stuck-at-"0" fault occurs when the output of an amplifier is permanently pulled to ground potential, and a stuck-at-"1" fault corresponds to an amplifier output permanently stuck at its highest output potential.

The fault-locations were randomly selected with one important exception: *Two simultaneous faults in the same row or column are prohibited.* Obviously, two stuck-at-"1" faults in the same row or column would preclude a valid solution and represent the total system failure. Thus, it is important to keep this exception in mind when interpreting the results because this failure mode of the system is explicitly excluded from our figures. The same fault-locations were used for the stuck-at-"0" and stuck-at-"1" faults in order to compare the effect of different fault types in the same location. Therefore, we also excluded two stuck-at-"0" faults in the same row or column, although this would not affect the system in the same way as in the stuck-at-"1" case.

We used a subset of 5 different 10-, 20-, and 30-city-distributions and performed 5 different random initializations for each distribution. These overall 75 simulation-runs were repeated for up to 13 injected faults of both types. The 50 and 100 city cases were not considered for this experiment simply because the simulations are computationally too expensive. The results in terms of the defined tour quality are shown in Figure 1 to 4. Figure 1a-c shows the individual results for three out of five 10-city-distributions with up to seven injected faults. The Figures 2a-c and 3a-c do

the same for the 20- and 30-city case, respectively, but with up to 11 injected faults for the 20-city cases, and up to 13 faults for the 30-city cases. The average values for all five 10-, 20-, and 30-cities are shown in Figure 4a-c.

4. Discussion

At a first glance it is hard to believe that the injection of multiple, especially stuck-at-"0" faults seems to have almost no negative effect on the performance and sometimes even results in better tours than in the fault-free cases. In fact, we had to convince ourselves of the absence of serious software errors in our batch-simulator by using a second, interactive simulator to monitor the evolution of the network solution. For the interpretation of the results it is important to realize that the two different fault-types have different effects on the network. While stuck-at-"0" faults preclude certain configurations or tours, stuck-at-"1" faults impose the order in which the corresponding cities are visited on the tour. The latter case has obviously a more serious effect on the performance which is reflected in the results. In the extreme case of n stuck-at-"1" faults for an n-city-distribution we would simply impose a random tour, given our constraint of having only one fault in a row or column. This effect can be seen in Figure 1a and 1c after injecting 7 stuck-at-"1" faults for a 10-city case.

The performance variations for the individual city-distributions are not surprising since the fault effect depends on the actual locations of the cities in a given distribution. A more general trend of the performance degradation can be seen in Figure 4 after averaging over all five city-distributions for each problem size. These results indicate that stuck-at-"0" faults are no problem at all, and that the network is well capable 'of working around' multiple stuck-at-"1" faults as long as the number of faults does not reach the number of cities and as long as there is only one fault in each row or column.

Another very important performance characteristic not reflected in Figures 1-4 is the number of valid tours after fault-injection. Surprisingly, we observed only 3 cases (two 30-city and one 20-city case) out of over 4000 simulations in which the

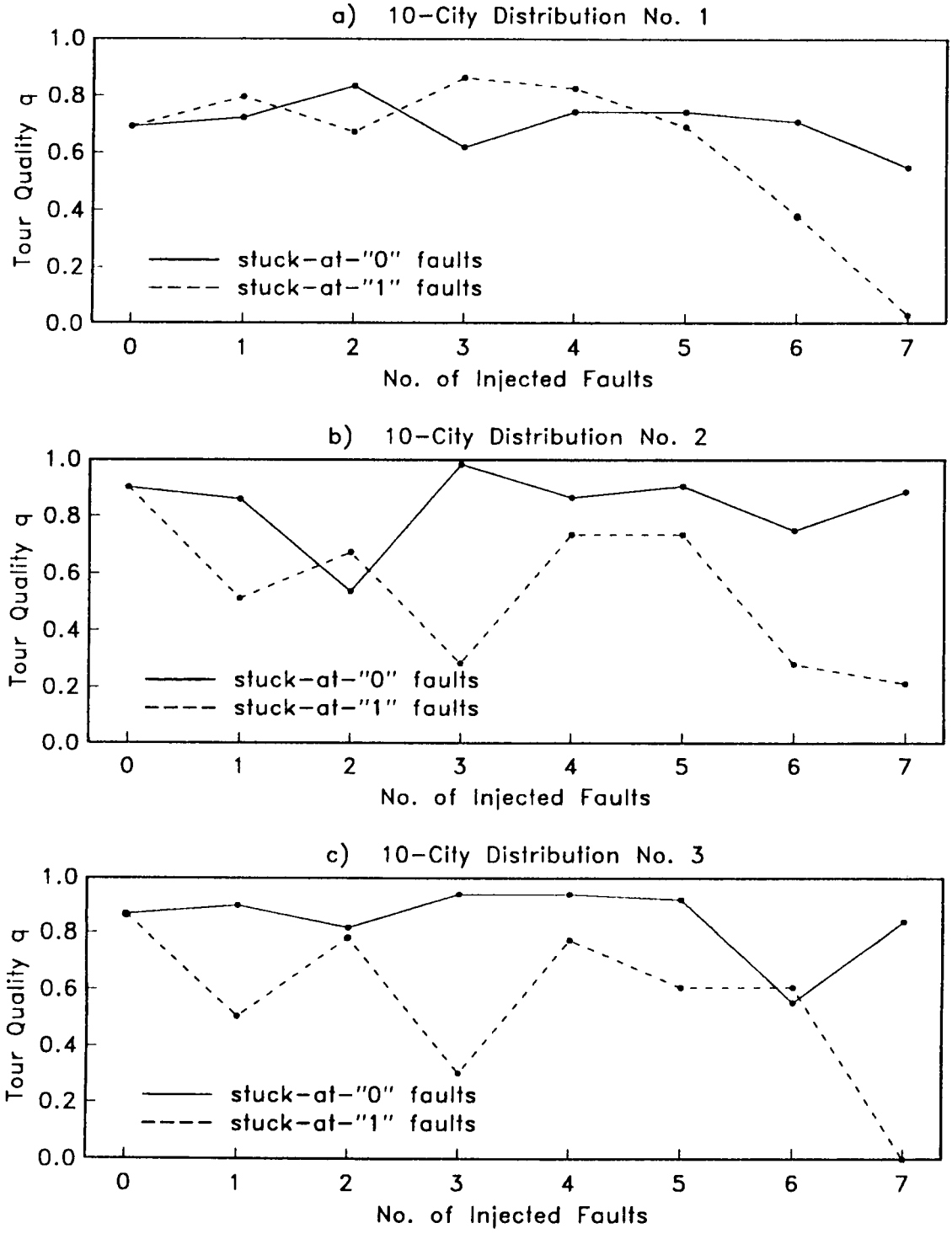

Figure 1. Performance after fault-injection for three 10-city-distributions.

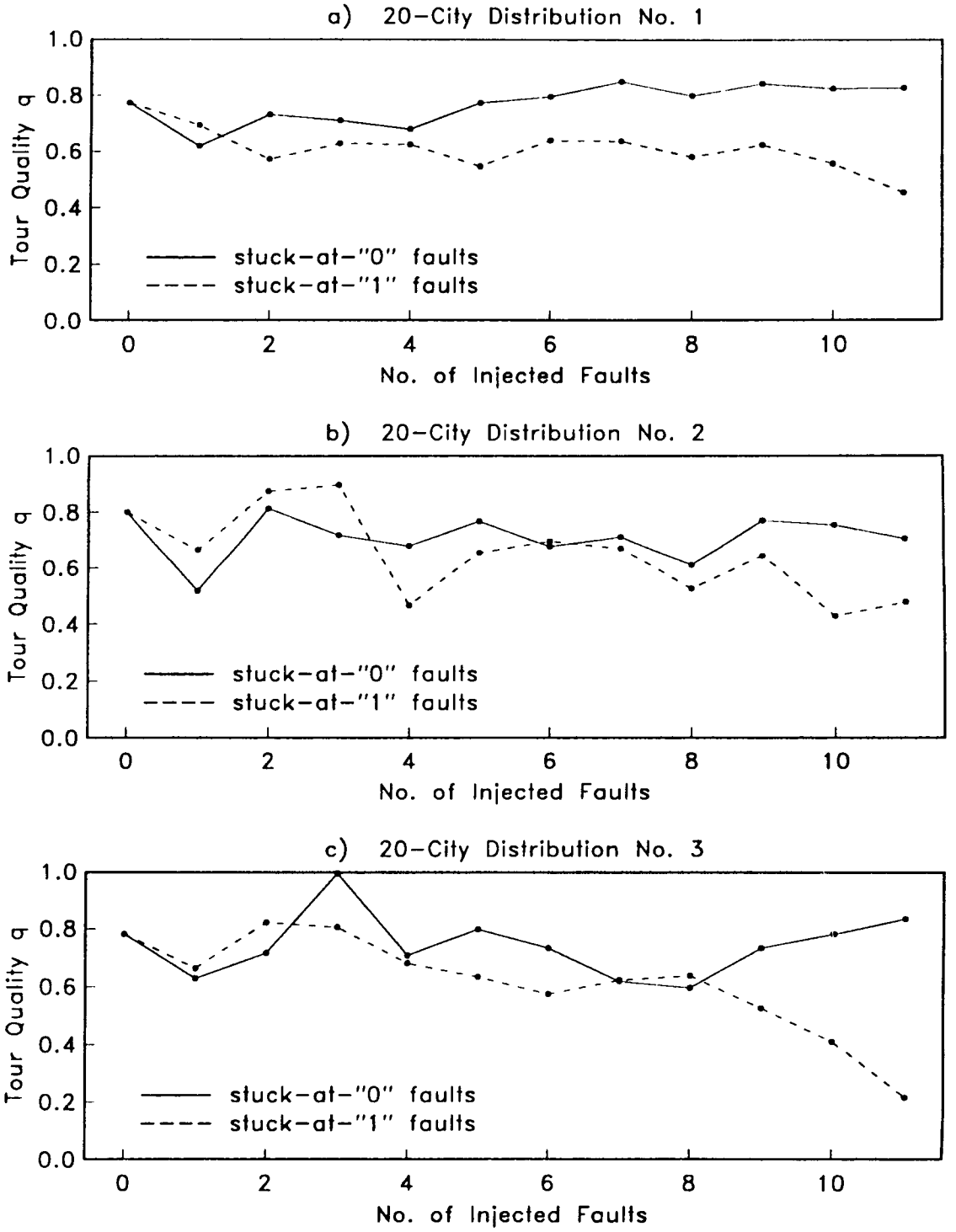

Figure 2. Performance after fault-injection for three 20-city-distributions.

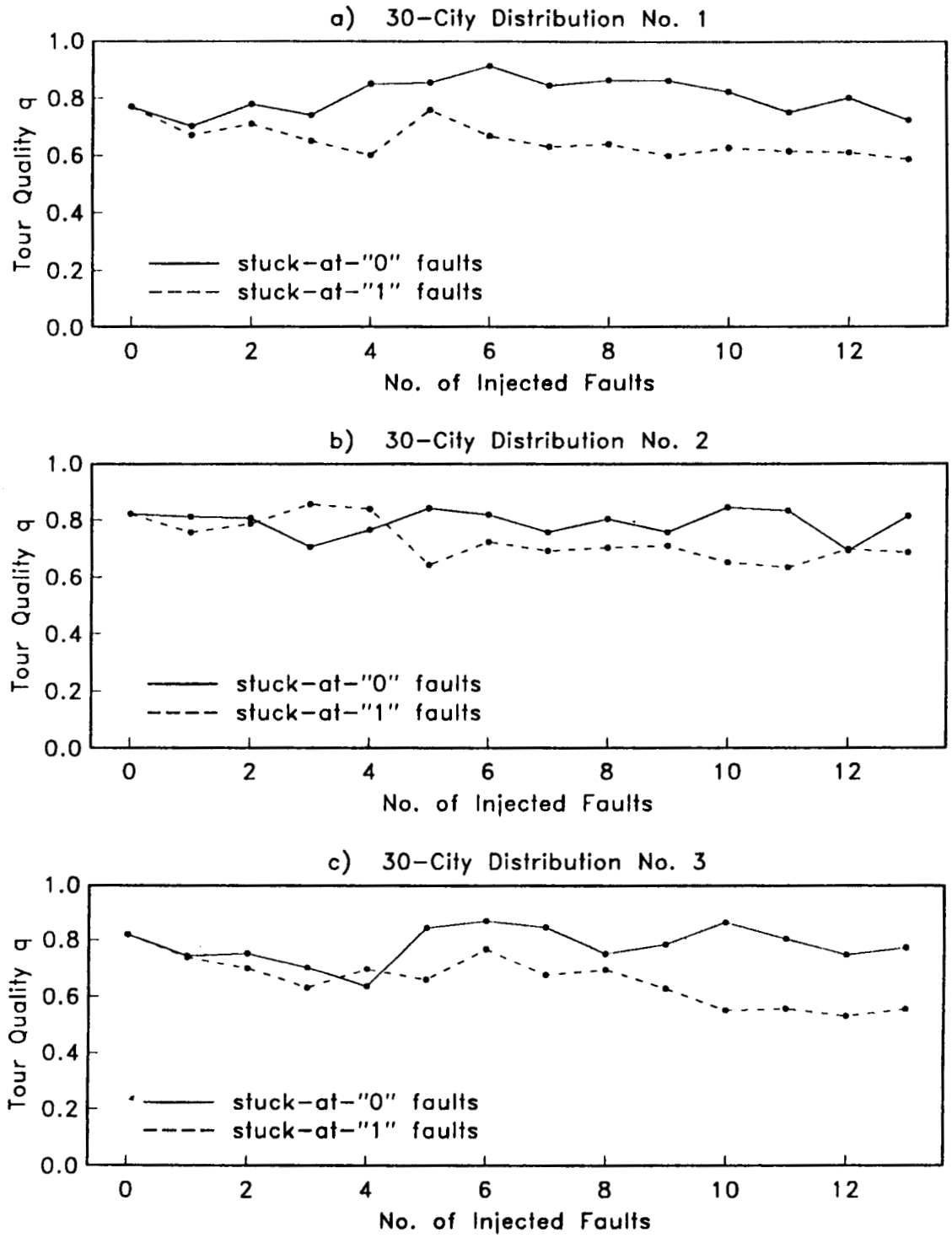

Figure 3. Performance after fault-injection for three 30-city-distributions.

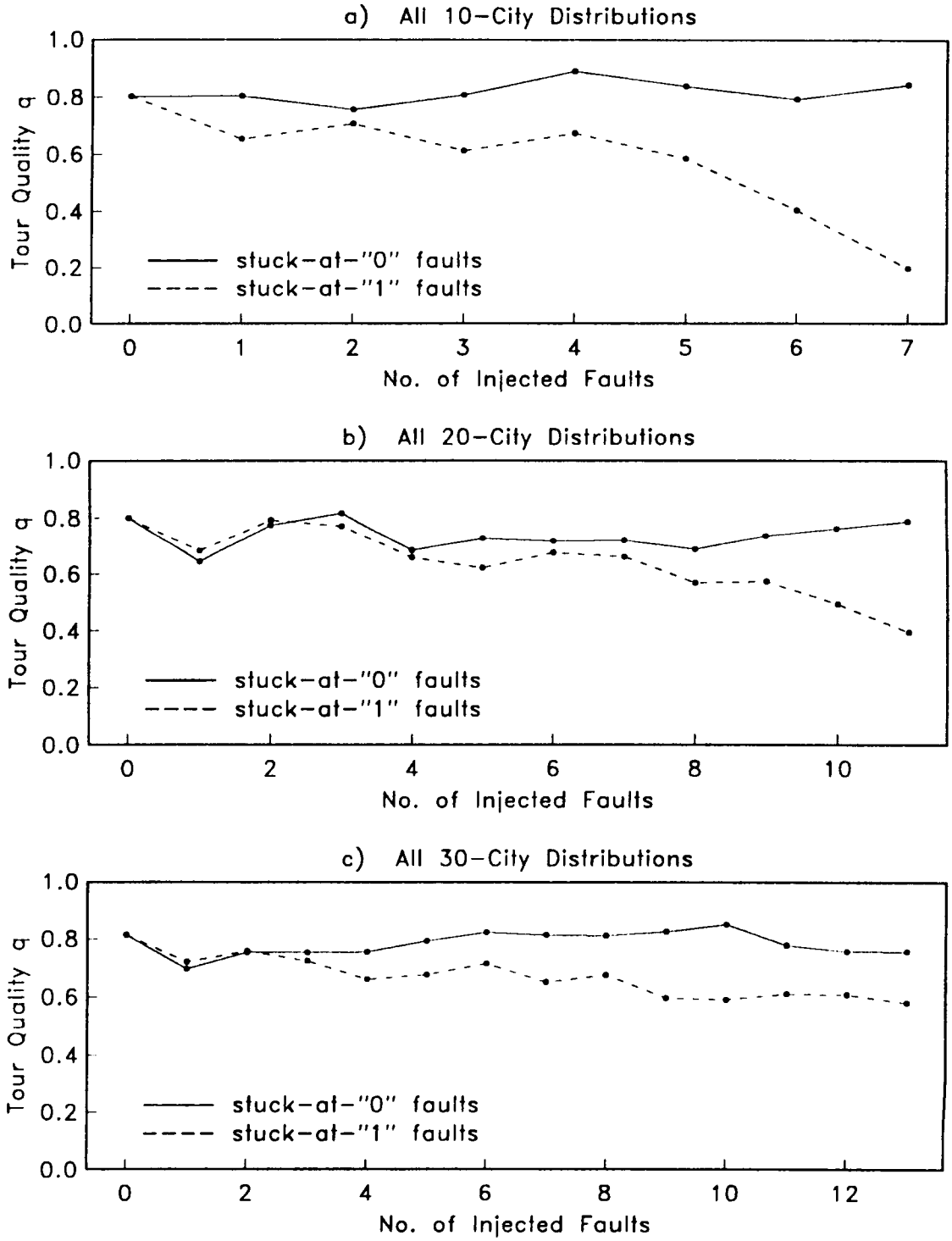

Figure 4. Performance after fault-injection averaged over five city-distributions for each problem size.

network did not converge to a valid solution. Another interesting phenomenon is that two or more injected faults of any type 'overwrite' the random initialization, that is, we get identical tours for different random initializations.

5. Summary and Conclusion

The goal of this study was to investigate the effect of injecting faults into a simulated ANN solving the TSP. The network is based on a modified version of the original method of Hopfield and Tank. The modified method produces consistently valid tours and represents the basis for the fault-injection experiment. Since the quality of a solution varies for different city-distributions and different random initializations, we defined a characteristic measure for the tour quality in relation to the optimal and random tour length for a given city-distribution. This allows the assessment of the overall performance for different city-distributions and problem sizes.

Five different 10-, 20-, and 30-city-distributions were used for the injection of up to 13 stuck-at-"0" and stuck-at-"1" faults. The results of more than 4000 simulation-runs reveal a surprisingly high fault-tolerance. Stuck-at-"0" faults seem to have almost no or even a positive effect on the performance. A sufficiently small number of stuck-at-"1" faults also has little effect, as long as there is only one fault in a row or column. Only three cases produced invalid solutions under the presence of faults.

The results of this experiment show that the only dominant failure mode for the network is the simultaneous occurrence of two stuck-at-"1" in the same row and column, precluding a valid solution. One important factor that contributes to the extreme fault-tolerance exhibited by the network is probably the redundancy inherent in the problem representation with its 2n-fold degeneracy. Unfortunately, there does not yet exist another, less redundant problem representation for the TSP. Further experiments with different model problems are necessary to draw more general conclusions and to gain more insight into the relation between the redundancy of the problem representation and the fault-tolerance of the network.

References

[1] J.A. Anderson, "Cognitive and Psychological Computation with Neural Models", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 5, Sep-Oct 1983, pp. 799-815.

[2] G.E. Hinton and T.J. Sejnowski, "Learning and Relearning in Boltzmann Machines", in *Parallel Distributed Processing, Vol. 1*, D.E. Rumelhart and J.L. McClelland, Bradford Books/MIT Press, Cambridge 1986, pp. 282-317.

[3] J.M. Hutchinson and C. Koch, "Simple Analog and Hybrid Networks for Surface Interpolation", in *Neural Networks for Computing*, J.S. Denker, ed., American Institute of Physics, New York, 1986, pp. 235-239.

[4] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: a parallel network that learns to read aloud", John Hopkins Univ. Technical Report JHU/EECS-86/01, 1986, 32 pp.

[5] J.J. Hopfield and D.W. Tank, " 'Neural' Computation of Decisions in Optimization Problems", *Biological Cybernetics*, Vol. 52, 1985, pp. 141-152.

[6] S. Lin and B.W. Kernigham, "An Effective Heuristic Algorithm for the Traveling Salesman Problem", *Operations Research*, Vol. 21, 1973, pp. 498-516.

[7] G.V. Wilson and G.S. Pawley, "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank", *Biological Cybernetics*, Vol. 58, 1988, pp. 63-70.

[8] R.D. Brandt, Y. Wang, A.J. Laub, and S.K. Mitra, "Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem", *Proc. of the IEEE Int. Conf. on Neural Networks*, San Diego, CA, July 1988, pp. II-333 - II-340.

[9] S.U. Hedge, J.L. Sweet, and W.B. Levy, "Determination of Parameters in a Hopfield/Tank Computational Network", *Proc. of the IEEE Int. Conf. on Neural Networks*, San Diego, CA, July 1988, pp. II-291 - II-298.

[10] D.E. Van den Bout and T.K. Miller, "A Traveling Salesman Objective Function that Works", *Proc. of the IEEE Int. Conf. on Neural Networks*, San Diego, CA, July 1988, pp. II-299 - II-304.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181798 ICASE Report No. 89-12		

4. Title and Subtitle	5. Report Date
FAULT-TOLERANCE OF A NEURAL NETWORK SOLVING THE TRAVELING SALESMAN PROBLEM	February 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
P. Protzel D. Palumbo M. Arras	89-12
	10. Work Unit No. 505-90-21-01

9. Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	NAS1-18605
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Final Report

Submitted to Inter. Joint Conference on Neural Networks

16. Abstract

This study presents the results of a fault-injection experiment that stimulates a neural network solving the Traveling Salesman Problem (TSP). The network is based on a modified version of Hopfield's and Tank's original method. We define a performance characteristic for the TSP that allows an overall assessment of the solution quality for different city-distributions and problem sizes. Five different 10-, 20-, and 30-city cases are sued for the injection of up to 13 simultaneous stuck-at-"0" and stuck-at-"1" faults. The results of more than 4000 simulation-runs show the extreme fault-tolerance of the network, especially with respect to stuck-at-"0" faults. One possible explanation for the overall surprising result is the redundancy of the problem representation.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
fault-tolerance, fault-injection neural networks, optimization	63 - Cybernetics 62 - Computer Systems Unclassified - Unlimited

19 Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	14	A03