Neural Networks for Automated Vehicle Dispatching

Yu Shen
Jean-Yves Potvin
Jean-Marc Rousseau

Centre de Recherche sur les Transports
Université de Montréal
C.P. 6128, Succ. "A"
Montréal (Québec)
Canada H3C 3J7

Scope and Purpose. Inspired by the findings about neurons, and especially by some effective training procedures found in the late 70's, neural networks have seen a resurgence in the research community. Its philosophy assumes simple but massive interconnected processing units as the basic structure for information processing. Problems that are difficult to tackle with conventional symbolic-based techniques, seem to be more readily solvable with a neural network approach (e.g. perceptual tasks, associative memory). This paper focuses on such a problem: the dispatching of vehicles and crews. This kind of problem is found, for example, in the context of parcel pick-up and delivery services or priority mail. The task of assigning a particular vehicle or driver to a new request is done by a human dispatcher, and his(her) decision process typically involves a lot of experience, judgment and expertise. We thus describe in this paper a neural network model that can learn to reproduce the decision process of an expert dispatcher by using examples of his(her) previous decisions.

Abstract. Decision making with respect to the dispatching of vehicles and crews is still mainly in the realm of human expertise. From our own experience, it seems very difficult to explicitly model that expertise via a symbolic approach. In this paper, we thus propose an alternative neural network model as a sub-symbolic and empirical alternative for modeling the decision process of expert dispatchers. Preliminary results about the ability of the network to reproduce various decision rules are reported.

1. INTRODUCTION

The Vehicle Dispatching Problem (or Demand Responsive Dial-A-Ride Problem) is concerned with the allocation of vehicles for servicing customers requesting immediate service. Typical examples would be parcel pick-up and delivery services, priority mail or ambulance services. The task of assigning a particular vehicle to a new request is done by a human dispatcher, and his(her) decision process typically involves a lot of experience, judgment and expertise. However, such capabilities are very difficult to formalize (e.g. explicit decision rules), since they require complex perceptual and cognitive abilities, like consideration of the current location of each vehicle with respect to the location of the origin and destination points of the new request. Given those premises, automated knowledge acquisition techniques look as a promising way to avoid the knowledge acquisition bottleneck in this area.

Inspired by the findings about neurons, and especially by some effective training procedures found in the late 70's, neural networks have seen a resurgence in the research community [Rumelhart and McClelland 86a]. Its philosophy assumes simple but massive interconnected processing units as the basic structure for information processing. Problems that are difficult to tackle with conventional symbolic-based techniques, seem to be more readily solvable with neural network models (e.g. perceptual tasks, associative memory).

This paper focuses on such a problem: the dynamic dispatching of vehicles and crews. We describe a neural network model that uses examples of expert dispatcher decisions for training. Given a set of attributes that describe typical dispatching situations and the decision that has been taken (i.e. the vehicle chosen for servicing the new request), a neural network is trained to assess the relative quality of each candidate vehicle and select the vehicle with highest quality.

In the rest of the paper, we divide our discussion along the following lines. Section 2 first defines more precisely the vehicle dispatching problem. Section 3 then describes various ways of encoding dispatching situations for a neural network model and discusses related network structures. Section 4 reports on experimental results in various decision contexts. Finally, concluding remarks are made in section 5.

2. THE VEHICLE DISPATCHING PROBLEM

In the Vehicle Dispatching or Dial-A-Ride Problem [Bodin et al. 83], customers call a dispatcher for service request. Each customer specifies a distinct pick-up and delivery location and, perhaps, a desired time for pick-up or delivery. If all customers demand immediate service, then routing and scheduling is done in real time and the problem is referred to as the dynamic Vehicle Dispatching Problem. In such a demand responsive system, the system is looked at when a new customer calls into the dispatch office. At that time, some of the earlier customers have been delivered to their destinations and, hence, are no longer considered. The remaining customers who have requested service have been assigned to a vehicle and are either on their way to their destination or are waiting for their pick-up. Moreover, at that time, the route and schedule for each vehicle is known. The problem is thus to determine the assignment of the new customer to a vehicle and the new route and schedule for the vehicle that the customer is assigned to.

Currently, this problem is still mainly solved by human dispatchers. Only a few algorithmic approaches have been designed for that class of problems, the best known approach being Wilson's heuristic [Wilson and Covin 77] for a dial-a-ride transportation system in Rochester, NY. A dynamic programming algorithm for the single vehicle case is also described in [Psaraftis 80].

Our approach is quite different, since we are interested in building an adaptive system that can learn to reproduce the decision process of the expert dispatcher it is trained with. In this way, such a system could easily adapt to various dispatching environments. Since the decision procedure can vary a lot from one organization to another, the system's adaptability looks here as a great asset.

Let us first assume that each one of the n available vehicles (drivers, candidates) involved in the dispatching situation can be characterized by a vector of m attributes $(x_{i1}, x_{i2},...,x_{im})$, i=1,2,...,n. Those attributes should reflect, for each vehicle, the actual dispatching situation with respect to their current route and the new request to be serviced, like: the extra-mileage required for servicing the new request, the pick-up time, the delivery time, additional delays incurred by the requests that are planned to be serviced by that vehicle, etc.

The set of all candidates, each with its m attribute values, is called a *context* for decision making. The objective of the dispatching process is thus, given a context, to select a suitable candidate. The decision process

can be modeled as a function F from the set of contexts C to the set of candidate vehicles V:

F: C -> V where:

$\{c_1,c_2,...,c_n\} \in C$

$c_i = (x_{i1},x_{i2},..., x_{im})$, i=1,2,...,n, and

x_{ij} = attribute value j of vehicle i.

The dispatching function F should encode the human expertise and it is obviously very difficult, if not impossible, to define F analytically. Here, we simply attempt to approximate it via a neural network model. In the next section, we will first describe ways of encoding a dispatching situation for a neural network. Then, the neural network's structure and training procedure will be discussed.

3. DATA ENCODING

A) Global encoding

The most obvious way to encode dispatching situations is to provide a context to the network as an input vector of length (m x n), where m is the number of attribute values for each vehicle and n is the number of alternative vehicles in the context. We would then expect a relative measure of quality for each vehicle as an output vector of length n.

We must however observe that a context is really an unordered set of n attribute vectors. Hence, the final outcome should not be affected by their ordering (e.g. providing the attribute values of vehicle i before or after the attribute values of vehicle j should have no effect on the final decision).

In order to do so, we define a lexicographic ordering $(c_{i_1},..., c_{i_n})$ of the n attribute vectors in each context, as follows:

put c_{i_j} before c_{i_k} if and only if

there is some t in the range $1 \le t \le m$ such that

$x_{ijs} = x_{iks}$, $1 \le s < t$ and $x_{ijt} < x_{ikt}$

As an example, Figure 1 shows a three layer network for m=3. As we can see, there is a hidden unit for each group of m input units (a group describes a single vehicle) and each hidden unit is only connected to one group of input units. The hidden layer is however completely connected

to the output layer and there is an output value o_{ij} for each group of input values (x_{ij1},..., x_{ijm}) that describe a vehicle, $1 \leq j \leq n$.

Given n groups of input units (I_{ij1},...,I_{ijm}) and the hidden units H_{ij}, $1 \leq j \leq n$, we would then like to constrain the weights on the connections between the input and hidden units in such a way that, for any given attribute k, $1 \leq k \leq m$, the weight value $W_{H_{ij},I_{ijk}}$ is the same for all j. This could be achieved via the "shared weights" approach of [Rumelhart et al. 86b].

In this way, it would be possible to encode the whole dispatching context. However, the length of the input vector is (m x n) and this value can quickly become very large for real life problems. It would thus be difficult to train the neural network without a huge training sample of contexts. In order to avoid this problem, the next section describes alternative ways of providing this information to the neural network.

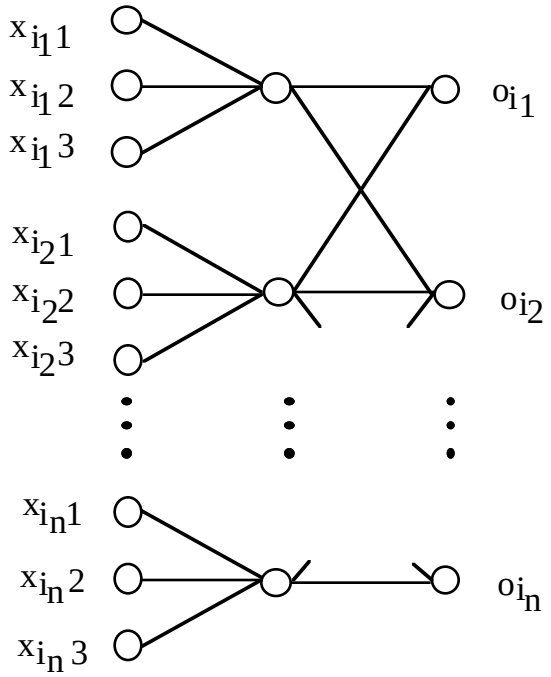

Fig. 1 Global encoding

B) Local Encoding

Obviously, each attribute value x_{ij} per se carries little meaning. For example, if the extra-mileage of vehicle i for servicing the new request is 5 miles, it does not mean much without knowing the extra-mileage of the other vehicles. Hence, the information that we need is disseminated over

the whole context. Since the vector representation of a context is of length (m x n), it would certainly be interesting to find a way to shift this global information into local information for each individual vehicle (so that the data associated with a given vehicle can be interpreted without direct reference to the other vehicles). By doing so, we could then train the network with input vectors describing each single vehicle, rather than input vectors describing the whole set of candidate vehicles. With only 25 distinct contexts and 10 alternative candidate vehicles in each context, it would already be possible to generate $25*10 = 250$ distinct input vectors for training. Moreover, reducing the length of the input vector by a factor of n allows us to train the network with much smaller training samples (since the complete set of all possible input patterns is greatly reduced).

In the following, where we assume that the attribute values are all greater or equal to zero, we describe two transformations of the data that allow us to shift a major part of the global information into a single vehicle's input vector.

Translation. Since we try to reproduce the preferences of the dispatcher (one vehicle is preferred over all other available vehicles), one good way of transforming the jth attribute value x_{ij} of vehicle i is to translate it with respect to the best jth attribute value over all vehicles in the same context, that is:

$$x_{ij}' = x_{ij} - (\min_i x_{ij}) \quad \text{or} \quad x_{ij}' = (\max_i x_{ij}) - x_{ij}\ , \quad i=1,...,n;\ \ j=1,...,\ m,$$

depending if the attribute value is a measure of cost (min) or profit (max). By doing so, $x_{i*j}' = 0$ for the best vehicle i* with respect to the jth attribute, and the remaining values x_{ij}' ($i \neq i^*$) in the context will rank relative to the gap with the best value. Geometrically, this transformation is simply a translation of each vehicle's input vector with respect to the minimum (maximum) value of each attribute in the context. It readily shows, for any given attribute, where a vehicle stands with respect to the best attribute value in its context.

Normalization. We can then transform furthermore the above values by normalizing over the maximum gap in a context, that is:

$$\begin{aligned} x_{ij}'' &= x_{ij}'/\max_i x_{ij}' \quad , \text{ if } \max_i x_{ij}' \neq 0 \\ &= x_{ij}' \quad , \text{ otherwise} \end{aligned}$$

$$i= 1,...,n;\ \ j= 1,...,\ m.$$

This normalization facilitates input pattern comparisons across distinct contexts. It should thus help the learning procedure in finding regularities over the set of training patterns.

It is obviously possible that some globally available information cannot be shifted to the local vehicle's input vector with the two techniques described above. However, it seems that during training, some patterns should now appear to be more clearly related with vehicles that are chosen by the dispatcher. For example, if the value of the jth attribute is very important in the decision process, vehicles whose input vectors show a jth attribute value close to zero are more likely to be chosen. Conversely, those with a value close to 1 are less likely to be chosen. We thus hypothesize that the above transformations capture essential information about the decision problem, and that they are sufficient for training a neural network with only individual vehicle's input patterns (x_{i1}",..., x_{im}").

Figure 2 shows a two layer and a three layer network based on such a local encoding of the data for m=3. The experiments described in the next section relate to those network structures. We used either the two layer or the three layer network depending on the linearity or non linearity of the decision procedure (to be discussed later).

We assume that the neural network will learn to compute a quality function F" based on the attribute values that describe a dispatching situation for a given vehicle. If vehicle i* is chosen over all other vehicles in a given context then, ideally, the activation of the output unit should be greater for the input pattern (x_{i*1}",..., x_{i*m}") of vehicle i* than for any other vehicle in the same context, that is $o_1(i^*) = F''(i^*) > F''(i) = o_1(i)$, for all $i \neq i^*$ in the context.

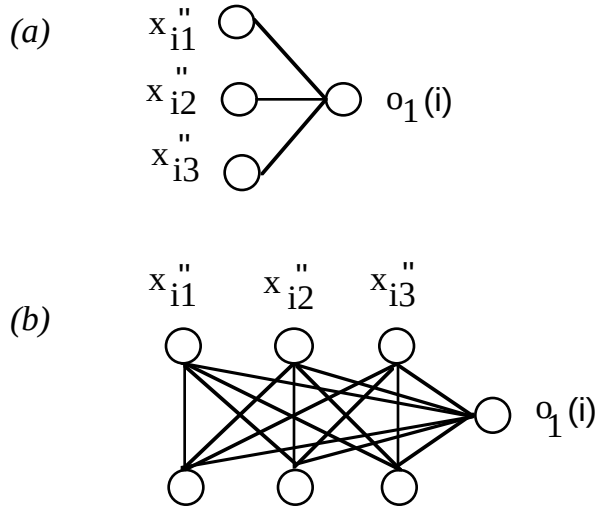

Fig. 2 Two layer and three layer networks

The neural network output can be related here to the relative selection frequency of a given pattern among all its occurrences in the training sample. If there is enough information within a pattern for decision making, then this pattern should be consistently selected (or not selected). If the information within the pattern does not include all the information for decision making, then this pattern will be chosen in some situations and not chosen in others. In such a case, the relative selection frequency for the pattern will be between 0% and 100%. However, it is expected that "good" and "bad" patterns should emerge with respectively high and low selection frequencies (i.e. frequencies respectively well over and under 50%), assuming that there is enough information in those input patterns to make suitable decisions in most situations.

Training. The training of the neural network was done via the back-propagation learning algorithm of Rumelhart et al., using the well known sigmoid activation function for each unit i [Rumelhart and McClelland 86b, Jones and Hoskins 87]:

$$1/\ 1 + e^{-x(i)} \quad \text{where} \quad x(i) = \theta_i + \sum_j w_{ij} * a_j,$$

θ_i = input bias of unit i.
a_i = activation value of unit i.
w_{ij} = connection weight between j and i.

The input patterns provided to the neural network were vectors of attribute values describing dispatching situations for individual vehicles, as defined above. For evaluating the error signal, the desired output was set to 1 when the vehicle was selected by the expert and it was set to 0 otherwise.

It should be noted that such a binary desired output can look a little bit artificial, since we do not necessarily want the neural network's response to be 1 for a selected vehicle and 0 for a non selected one. We simply want the neural network's output to be higher for the selected vehicle than for any other non selected vehicle in the same context. This is a typical case where it is easy to know if the neural network is right or wrong (by applying the neural network to all vehicles in a context and comparing the output for the vehicle chosen by the expert with the outputs for the other vehicles), but the desired or target output is not really known. We are thus currently investigating alternative approaches, such as reinforcement learning [Barto and Anandan 87], so as to try to improve the results that we obtained with the binary target outputs.

As an example, one alternative is to define the error signal as follows. For a *non selected* vehicle (i.e. not chosen by the expert), the error signal is set to the difference between the output for this vehicle's input pattern and the output for the selected vehicle in the same context, based on the current neural network's weights. If this value is negative, however (i.e. the neural network is right and the output for the current vehicle is lower than that of the selected vehicle), then the error is set to 0 and no training takes place. For a *selected* vehicle (i.e. chosen by the expert), the error signal is set to the difference between the largest output of all non selected vehicles in the same context and the output for the selected vehicle, based on the current neural network's weights. Once again, if this value is negative (i.e. the neural network is right and the output for the selected vehicle is higher than that of any other vehicle in the same context), the error signal is set to 0 and no training takes place.

With such an error signal, however, the error measure (mean squared error) is not anymore a differentiable function with respect to the set of weights. In order to overcome this difficulty, we first simply replaced the classical error signal "desired output - current output" by the above error signal into the backpropagation formulas. Unfortunately, the results were not as good as those obtained with the binary outputs. We are thus looking for a differentiable function that would approximate in a satisfactory way the new error measure, so as to preserve the gradient descent behavior of the backpropagation algorithm. This is one of the topics that we are currently investigating.

Section 4 will now present the results that we have obtained with the binary target outputs.

4. EXPERIMENTAL RESULTS

We ran three different experiments in order to get some insight about the ability of a neural network to reproduce systematic and human decision processes. In each case, the data were previously collected from an operations day of a courier company serving an urban area. The two first preliminary experiments were based on a priori decision rules. We felt that it would be useless to try to reproduce the decision making process of an expert, if the neural network model would fail to reproduce even simple systematic decision rules. In the last experiment, we asked an expert at the Centre de Recherche sur les Transports (CRT) of Montreal University to take the dispatching decisions.

In the first two experiments, we took 25 distinct contexts with 10 vehicles in each context for training and testing, that is, 25*10 = 250

training and testing patterns. We used m=3 distinct attribute values for characterizing a dispatching situation for a given vehicle, namely extra-mileage (expressed in time units), pick-up time and delivery time. The results are discussed below.

1) Experiment 1: The Minimum Value Rule.

The first rule that we tried to reproduce is quite simple: select the vehicle whose extra-mileage is minimum in a given context. Since the translated and normalized extra-mileage value of all selected vehicles is 0, it is easy to see that the classes of selected and non selected vehicles are linearly separable in the attribute or input space (assuming that in case of equality, all vehicles with the same minimum value in a context are selected). We thus easily reproduced this decision process with a two layer neural network model (cf. fig. 2a).

After training, we observed a very large negative weight for extra-mileage, and weights close to zero for the two remaining attributes. 100% accuracy was achieved both for training and testing contexts. It should however be noted that the same results could have been obtained with a classical linear regression model.

2) Experiment 2: The Neighborhood Rule.

The second rule is more complex. We first identify a subset of qualified vehicles in a given context, by considering all vehicles with an extra-mileage value within 5 minutes of the minimum value in the context (i.e. all vehicles with an extra-mileage value in the "neighborhood" of the best value). Then, for all vehicles satisfying the first condition, we select the vehicle with minimum pick-up time in the context. In case of equality, all vehicles with the same minimum value are chosen. It can be easily shown that the classes of selected and non selected vehicles are not anymore linearly separable.

At first, we used a standard three layer feedforward network for training, where units of layer i are linked to units of layer i-1 and i+1 (if any). During the course of experiments however, it just happened that we got better results by adding direct connections between the input units and the output unit (as depicted in figure 2b). Connecting the inputs directly to the output increases the number of free parameters and allows the network to use the direct connections for linearly separable features while reserving the hidden connections for the non linearly separable ones.

Figure 3 shows a typical learning curve for the neighborhood rule with a standard three layer feedforward neural network structure (backprop. net. 2) and a structure with direct links between input and output units (backprop. net. 1). At our first training attempt with the latter structure, we got 23 decisions right out of 25 for the training set and 21 out of 25 for the testing set. However, the magnitude of the error was very large for two erroneous contexts in the testing set (i.e. the output for the vehicle that should have been selected was quite low with respect to the vehicle with highest output). In both contexts, the vehicle to be selected by the neighborhood rule was a "borderline" case, in the sense that the gap between its extra-mileage value and the best value in its context was just slightly under 5 minutes.

Such results were an indication that the neural network did not clearly learn the decision rule and that our training set failed to provide some useful examples (such as "borderline" cases). We thus added the two borderline cases to the set of training contexts, and trained the neural network anew. At the end of that process, we got only 2 wrong decisions during testing and the magnitude of the error was greatly reduced. Moreover, the vehicle with highest output in both erroneous contexts related to a vehicle that should have ranked second according to the neighborhood rule and, as such, it was not really a bad choice.

Figure 4 summarizes the testing results for this experiment. For each one of the 25 contexts, we show the three highest ranked vehicles. Note that the vehicles satisfying the neighborhood rule are underlined and that the character "X" is shown at the end of the context line when the network is wrong (i.e. when the neural network's highest ranked vehicle is not underlined). In most contexts, only one vehicle satisfies the neighborhood rule, but in a few cases there is a tie between two or even three vehicles (contexts 1, 13, 18).

As we can see in the figure, an underlined vehicle is ranked first or second by the neural network in each context. Note also that for one erroneous context (context 22), the underlined vehicle is very close to the highest ranked vehicle, while in the other case (context 15) the error is somewhat more important.

Context #	Outputs	Wrong (X)		Context #	Outputs	Wrong (X)
1	0.996 0.980 0.787			14	0.874 0.807 0.794	
2	0.735 0.545 0.001			15	0.724 0.544 0.000	X
3	0.982 0.462 0.000			16	0.996 0.067 0.000	
4	0.992 0.421 0.029			17	0.739 0.605 0.287	
5	0.691 0.000 0.000			18	0.995 0.968 0.000	
6	0.995 0.052 0.000			19	0.972 0.643 0.000	
7	0.996 0.895 0.742			20	0.822 0.022 0.000	
8	0.996 0.213 0.000			21	0.807 0.048 0.014	
9	0.996 0.000 0.000			22	0.995 0.993 0.241	X
10	0.986 0.761 0.180			23	0.996 0.153 0.001	
11	0.963 0.337 0.000			24	0.996 0.872 0.000	
12	0.643 0.421 0.000			25	0.822 0.022 0.000	
13	0.997 0.995 0.840					

Fig. 4 Results of Experiment 2

By analyzing the final set of weights, some interesting findings emerged (c.f. Figure 5). First, we can see that the direct link from the extra-mileage input unit x_{i1}" to the output unit is largely negative. The same observation applies for the link leading to the lower hidden unit. The total effect is thus to strongly inhibit the output unit as extra-mileage increases.

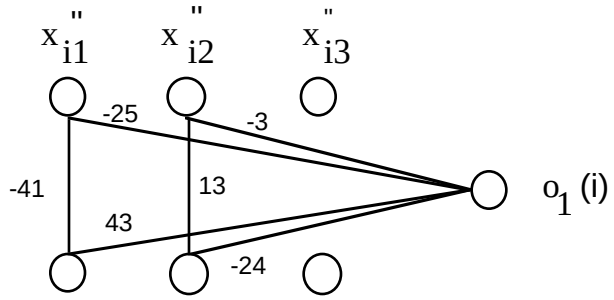

Fig. 5 Prominent weights

We can also observe that the link from the input unit for earliest pick-up time (x_{i2}") to the lower hidden unit is positive, but the link from that hidden unit to the output unit is largely negative. Hence, the total effect is to inhibit the output unit when pick-up time increases. Since those weights are not as large as for extra-mileage, the total inhibition is somewhat less important. It seems that this weight pattern is a way for the neural network to model the strongest effect of extra-mileage on vehicle selection (since extra-mileage is first used for filtering the set of candidate vehicles).

3) Experiment 3: Human Decision Process

Based on those preliminary results, we then decided to go on with a more "realistic" experiment. In this last setting, where a human was involved, we took 100 distinct contexts with 10 vehicles in each context for training and testing (i.e. 100∗10 = 1000 training and testing patterns). A larger set of attributes, modeling service quality preferences, was also used for describing the dispatching situations.

First, we introduced a limit on the delay between a customer's call and its pick-up and delivery. This limit was set to 30 minutes for pick-up and 90 minutes for delivery. Any service time exceeding those upper bounds was considered undesirable. Accordingly, we added the following four attributes to the three original ones: delay over 30 minutes for the pick-up at the new customer, delay over 90 minutes for the delivery at

the new customer, sum of additional delays for the pick-ups at the customers already included in the route (given that the new request is now in the route) and, finally, sum of additional delays for the deliveries at the customers already included in the route. Since it is advisable to equally share the service among vehicles, we also introduced the difference between the current number of customers serviced by a given vehicle and the average number of customers serviced by all vehicles in the same context. Hence, eight different attribute values were used for describing a dispatching situation for a given vehicle. Based on those values, a vehicle routing expert had to select the best vehicle in each one of the 100 training and testing contexts.

The neural network structure for this experiment is the one depicted in figure 2b, except for the fact that the number of input units was increased from three to eight.

Since the number of contexts is four times larger than in the first two experiments, we do not show here the detailed results of this experiment. Instead, we summarize those results in Figure 6 by showing the neural network's rankings of the vehicles selected by the expert for both the training and testing data after 10,000 and 40,000 training iterations. Figure 6 thus shows how many vehicles selected by the expert were ranked first in their context by the neural network, how many were ranked second, etc. Those results were obtained after respectively 4 hours and 16 hours of run time on a SPARC workstation by setting the learning rate to 0,01 and the momentum term to 0,9.

As we can see, the best results with respect to the training set were obtained after 40,000 training iterations. In such a case, the neural network provided the same response than the expert in 89% of the contexts, as compared with 79% after 10,000 iterations. For the testing set, all selected vehicles were ranked in the first four by the neural network after only 10,000 training iterations. Even if the correlation between the neural network's responses and the expert's responses did increase from 72% to 78% after 40,000 training iterations, it should be noted that, in the latter case, two selected vehicles are now ranked seventh and another one is ranked eighth! Such a result can be an indication that the neural network adjusted so well to the training set after 40,000 iterations that its generalization capabilities were slightly impaired.

Even if the neural network did not perfectly reproduce the decision process of the expert, we observed that its choices were never really bad (i.e. when the neural network's highest ranked vehicle is not the one selected by the expert, this vehicle is anyway a choice that makes sense)

and, in that respect, the trained network is behaving appropriately and do not provide "foolish" responses.

Rank		Training set 10,000 iter.	Testing set		Training set 40,000 iter.	Testing set
1st		79	72		89	78
2nd		12	18		5	12
3rd		6	7		4	7
4th		1	3		0	0
5th		1	0		1	0
6th		1	0		1	0
7th		0	0		0	2
8th		0	0		0	1
9th		0	0		0	0
10th		0	0		0	0

Fig. 6 Neural Network's Rankings of Vehicles chosen by the Expert

We immediately see the usefulness of such a network as an intelligent filter for extracting the best vehicles within a group of alternative vehicles. Such a restricted subgroup could then be presented to the human expert for the final selection. In that way, the neural network would greatly ease the decision process of the dispatcher, but would not remove the need for a final human intervention.

5. CONCLUSION

Overall, the results presented above indicate that the neural network paradigm looks quite interesting for solving dynamic Dial-a-Ride problems and that a neural network-based expert dispatcher system is

feasible. In that respect, we would like to make here a few additional remarks.

First, it is important to observe that the vehicle routing expert involved in the last experiment had to take decisions based on an *a priori* set of attributes. Since this expert was not a professional dispatcher, she accepted to take decisions based on such data. However, it is not clear if a real dispatcher would have accepted our metrics for describing dispatching situations. It is thus an interesting problem in itself to determine (with the expert dispatcher) the measures that should be used so as to allow good decisions to be taken. Since no professional dispatcher was available, we had to "skip" this phase and a priori determine those metrics, to the best of our knowledge.

The experiments have also shown the importance of providing the neural network with a training set that covers a range of dispatching situations that are representative of the whole set of possible situations. A way to identify "weaknesses" within a training set, is to generate alternative sets of data and use them to discover situations for which the trained neural network does not perform well. Those situations are then added to the original training set and the neural network is trained anew with the enlarged set. In a real setting, however, the neural network would be linked to a dispatcher taking decisions in real time. The neural network would thus use each new decision to incrementally adjust its weights. After training over a sufficiently long time, it would be possible to assume that the training data are representative of the current dispatching environment and include most (if not all) typical dispatching situations.

Finally, we would like to emphasize again the flexibility of the neural network approach. Such flexibility is very attractive in our context, because neural networks can be trained in various dispatching environments, and dynamically adapt to them during their training phase. The three distinct experiments that we performed were quite illustrative in that respect. This adaptive behavior is certainly a major asset, as compared with the classical expert system approach where the decision rules must be "a priori" determined and tailored to each specific context.

Acknowledgments. We would like to thank Lucie Bibeau for her collaboration during the computational experiments. Also, this research would not have been possible without the financial support of the Natural Sciences and Engineering Council of Canada (NSERC).

6. REFERENCES

[Barto and Anandan 85] A. G. Barto and P. Anandan, "Pattern Recognizing Stochastic Learning Automata", IEEE Transactions on Systems, Man and Cybernetics 15, pp. 360-375.

[Bodin et al. 83] L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and Scheduling of Vehicles and Crews: The State of the Art", Computers and Operations Research 10(2), pp. 63-211.

[Jones and Hoskins 87] W.P. Jones and J. Hoskins, "Back-Propagation: A Generalized Delta Learning Rule", Byte, October 1987, pp.155-162.

[Psaraftis 80] H. Psaraftis, "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem", Transportation Science 2, pp. 130-154.

[Rumelhart and McClelland 86a] D. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the MicroStructure of Cognition, vols. 1 and 2, The MIT Press.

[Rumelhart and McClelland 86b] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation", in Parallel Distributed Processing: Explorations in the MicroStructure of Cognition, vol. 1, The MIT Press, pp. 318-364.

[Wilson and Covin 77] N. Wilson and N. Colvin, "Computer Control of the Rochester Dial-a-Ride System", CTS Report Number 77-22, Cambridge, Mass., Center for Transportation Studies, MIT.