Neuro-Genetic Truck Backer-Upper Controller

Marc Schoenauer, Edmund Ronald

Neuro-Genetic Truck Backer-Upper Controller

Marc Schoenauer and Edmund Ronald

Abstract— **The precise docking of a truck at a loading dock has been proposed in [Nguyen & Widrow 90] as a benchmark problem for non-linear control by neural-nets. The main difficulty is that back-propagation is not a priori suitable as a learning paradigm, because no set of training vectors is available: It is non-trivial to find solution trajectories that dock the truck from anywhere in the loading yard.**
In this paper we show how a genetic algorithm can evolve the weights of a feedforward 3-layer neural net that solves the control problem for a given starting state, achieving a short trajectory from starting point to goal. The fitness of a net in the population is a function of both the nearest position from the goal and the distance travelled. The influence of input data renormalisation on trajectory precision is also discussed.

I. PLANT DESCRIPTION

The plant to be controlled here is a truck with trailer, which should be backed up to a loading dock; the truck is initially located at some random position in the parking lot. The truck backs up by small uniform steps, denoted by r, while steering is effected by re-orienting the front wheels before each step. There is thus only one control variable, the angle of the steering wheels with respect to the cab, denoted u.

Our truck is the same as that described in the pioneering studies of [Nguyen & Widrow 90], who devised back-propagation through time to investigate the system. We have followed [Nguyen & Widrow 90] in chosing the observable state variables which together define a state: The coordinates of the center back of the trailer, and the orientation of the cab and the trailer with repect to the x axis. The constraints on the steering range and the sizing of the truck have been retained.

We have however adopted a different set of dynamic equations, which allow us to resolve more easily the uncertainty on the angles θ_S and θ_C occasioned by the use of the arctangent in the paper cited above. The reader may verify that both derivations are equivalent in the first order, and thus define similar trajectories.

Both authors can be reached at Centre de Mathématiques Appliquées, URA CNRS 756, Ecole Polytechnique, 91128 Palaiseau, France. Email should be adressed to marc@cmapx.polytechnique.fr

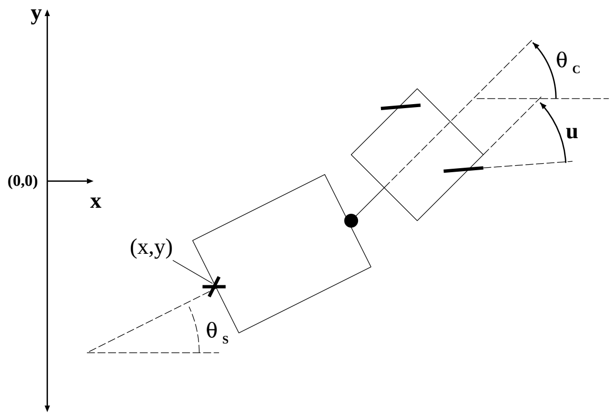

Figure 1: Truck-Trailer System Description

DETAILS OF PLANT

State	x, y	coordinates of the center rear of the trailer
	θ_S	angle of the trailer with x-axis
	θ_C	angle of the cab with x-axis
Control	u	steering angle of the front wheels relative to cab orientation
Constraints	$x > 0$	the loading dock is at $x = 0$
	$\lvert\theta_S - \theta_C\rvert \leq 90°$	the angle between the cab and trailer can't exceed 90°
	$-70° \leq u \leq 70°$	limit of the steering of the front wheel

Equations of Motion

$$A = r * \cos(u)$$

$$B = A * \cos(\theta_C[t] - \theta_S[t])$$

$$x[t+1] = x[t] - B * \cos(\theta_S[t])$$

$$y[t+1] = y[t] - B * \sin(\theta_S[t])$$

$$\theta_S[t+1] = \theta_S[t] - \arcsin\left(\frac{A * \sin(\theta_C[t] - \theta_S[t])}{L_S}\right)$$

$$\theta_C[t+1] = \theta_C[t] + \arcsin\left(\frac{r * \sin(u)}{L_S + L_C}\right)$$

$\theta_C[t+1]$ is then adjusted to respect the constraint on $\theta_S - \theta_C$

Parameters	r	$3m$	distance front wheel moves per time step.
	L_S	$14m$	length of the trailer, from rear to pivot.
	L_C	$6m$	length of the cab, from pivot to front axle.

II. EXPERIMENTAL SETUP

The problem we set out to investigate was the precise docking of the truck, with the center back of the trailer at the origin, and the trailer perfectly straight, ie. aligned with the x axis and thus with angle $\theta_S = 0$. We first concerned ourselves with docking from some random starting point such that $x > 0$, with the trailer arbitrarily jack-knifed with repect to the cab.

In order to solve this problem by a genetic algorithm, we decided not to search for short trajectories directly, but rather to postulate an intermediate agent, a neural net controller. Such a neural net acts as truck driver. Thus in the first instance we applied genetic search to the task of evolving a neural net able to dock the truck, from some single starting point. We then attempted to extend the method to finding nets capable of docking the truck from various starting configurations.

The experiments reported in this paper were done with a homebrew general-purpose genetic algorithm package embodying the principles described in [Holland, Goldberg] as well as in [Radcliffe, Michalewicz 92]. Our software subjects a small population of nets to crude parody of natural evolution. This artificial evolutionary process aims to achieve nets which display a large *fitness.* This is a positive real value which denotes how well a net does at its assigned task of driving the car. In other words, the fitness gets large if the net docks the car, larger yet if the trajectory it traverses is short. The details of the calculation of the fitness are found below.

For the purpose of applying the genetic algorithm, a given net is canonically represented by the chromosome consisting of its weights, i.e. as a vector of real numbers. Note that our genetic engine differs from the first methods presented by John Holland in that it does not use bit-strings; it is hybridised in that it directly represents floating-point numbers with the usual arithmetic precision achieved with 8 byte double-precision reals on workstations. In these experiments the genetic engine was not customised to reflect the structure of neural nets - a net structure-specific adaptation of the crossover operator may be undertaken in future research.

A stated in the introduction, a classical 3-layered feed-forward net architecture was employed [PDP], with complete interconnection between layers 1 and 2, and 2 and 3. The neural transfer function was the usual sigmoid logistic $f(x) = 1/(1+exp(-x))$. Regarding the crucial question of the number of neurons in the middle layer, the Kolmogorov model [Hecht-Nielsen] for such a net with n inputs and 1 output requires at least 2n+1 intermediate neurons and this paradigm was adopted.

The fitness calculation is central to solving a problem by a genetic algorithm. In our case, in the initial stages of this research we were considering a truck without trailer ie. a car, and did not take into account the length of a trajectory, but only whether it approached the $(0,0,0)$ docking position. To this end, each net was allowed to drive the car for 300 steps, and, at each step, the distance d to $(0,0,0)$ (in 3d-space) was computed, by the formula

$$d^2 = x^2 + y^2 + min({\theta_S}^2, (\theta_S - 2\pi)^2, (\theta_S + 2\pi)^2)$$

where the expression $min({\theta_S}^2, (\theta_S - 2\pi)^2, (\theta_S + 2\pi)^2)$ was employed to allow an offset of one complete turn either way. However, genetic algorithms maximise, rather than minimise, so the above distance was rescaled by taking an inverse, yielding the following fitness function:

$$Perf(x, y, \theta_S) = \frac{1}{\varepsilon + \min_{trajectory} d^2}$$

With the above fitness $Perf(x, y, \theta)$, a set of 10 numerical experiments was made: In each experiment, the car was started from (20,10 -2), and a different random seed was used to generate an initial population of 30 nets with random weights, with $-1 < w_i^j < 1$. The genetic algorithm was then allowed to run for 100 generations, and the best-performing net consistently achieved $d^2 < 5$, which translates to a maximum angular error of 0,1 radians (i.e. 6 degrees) and $\sqrt{5}$ linear units in x, y. Note that with a car move-step size of 3 a better result was hardly to be expected!
However, as trajectory length was not taken into account in the performance function, the nets took a circuitous, roundabout route in getting to the goal, a bit like a taxi driver who takes his guest for an expensive ride but ends up delivering him safe and sound. More results concerning the car are reported in [Schoenauer & Ronald 93]. The figure [1]below shows such a roundabout trajectory in the case of the truck.

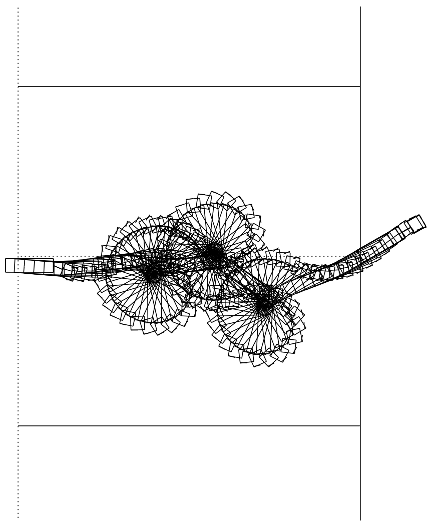

Figure 2: 2-Input net docking trajectory, goal is at center-left

[1]In all figures the region $[0, 100] \times [-50, 50]$ is shown, with the goal at $(0, 00)$

In order to take into account the length of the trajectory, the fitness function was amended to read:

$$Perf(x, y, \theta_S) = \frac{1}{\varepsilon + \min_{trajectory} d^2 + \gamma * \ell}$$

where ℓ is the number of time steps needed to dock the truck to $(0, 0, 0)$. The values $\varepsilon = 0.1$ and $gamma = 0.001$ were adopted as ensuring a good compromise between trajectory accuracy and length.

III. RESULTS

The results presented below fall into two categories that reflect the chronology of our experimentation: Single-point learning and multi-point learning.

A. SINGLE POINT LEARNING

For the single starting-point docking problem, successive experimental runs for the truck-with-trailer were made with neural nets differing by the type and number of inputs. These are detailed below.

At first, the same input-output architecture as in [Nguyen & Widrow 90] was essayed. Here the controller net takes the 4 variables $(x, y, \theta_S, \theta_C)$ as inputs and, by the Kolmogorov paradigm cited above, consists of a 4-9-1 fully connected neural net; this means that the chromosome of weights and biases is formed of 55 floating point numbers.

In this case, genetic learning yielded adequately short and precise trajectories, as in Fig 1. However convergence to a satisfying performance was only achieved by setting a fairly large value $r = 3$ to the step-length by which the truck backs up. the figure shows 500 generations (100 without improvement) with a population size of 100 individuals.

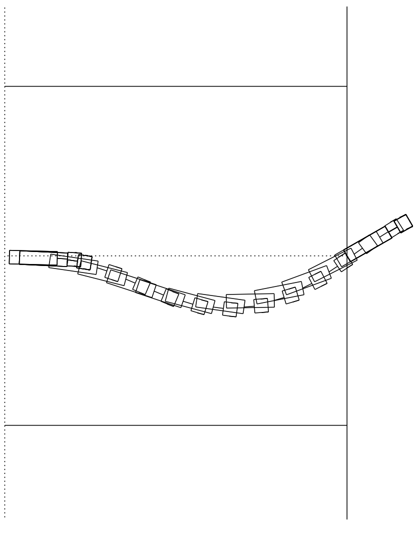
Figure 3: 4-Input net trajectory with best approach at $(1.35, -0.37, -1.52)$

The single-starting point problem allows an interesting optimisation: Recall that all we require of a controller is that it be able to dock the car from the given unique position. This means generating a set of steering angles $u[t]$ for each time-step t. It is reasonable to wonder whether a net really needs much information for this simple task, or whether it can be taught to "fly by instruments". We investigated this question by reducing the number of inputs to 3, by rendering the state variable θ_C unobservable. Thus the net does not "know" the angle of the cab. In practice this optimisation corresponds to a 3-7-1 net, ie. chromosomes of 36 reals. These 3-input nets could be trained at a lesser computational cost, yet provided excellent results, as can be seen in the figure below. We are still investigating just how far we can pursue this type of optimisation by removal of excess information.

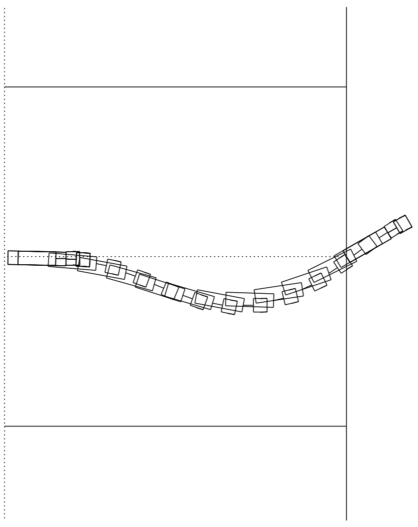
Figure 4: 3-Input net trajectory with best approach at $(0.99, -0.32, -1.11)$

B. MULTI-POINT LEARNING

In order to teach the nets to be able to dock from several points, we chose to compute a global fitness by taking the aritmetic mean, or the minimum of the single-point fitness mentioned earlier. Both methods gave acceptable results for a training ensemble of 15 points consisting of 5 angular orientations (-90, -30, 0, 30, 90) degrees clustered at the 3 locations of coordinates (100,0), (80,50), (80,-50). This yielded accptable accuracy after about 1500 generatiions of the GA.

In an attempt to improve the accuracy, which had stabilised, we enhanced the net structure, going from 4 inputs to 8. The additional inputs reproduced the original ones, but were rescaled by a factotr of 10. This was done in order to give the net the more precise perception necessary for fine control. The resulting 8-17-1 nets needed fewer generations (800) to learn, and were more precise. Actual computation time went up however because of the increase of the number of weights from 55 to 171.

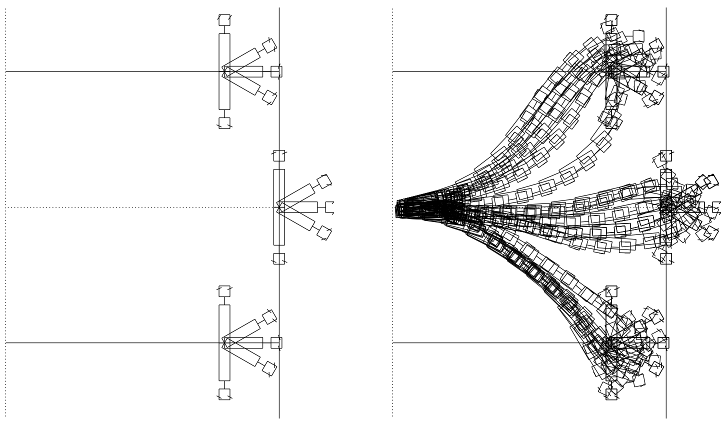

Figure 5: Multi-point learning ensemble and ensuing traffic

The remarkable generalization properties of neural nets are demonstrated by the fact that the net performed trained on the above described ensemble performed reasonably well when launched from any point in the 3-d space $[50-100] \times [-50, 50] \times [-180, 180]$, whatever the jacknife angle θ_C. In the figure below one the trajectories from 10 randomly chosen starting points are presented.

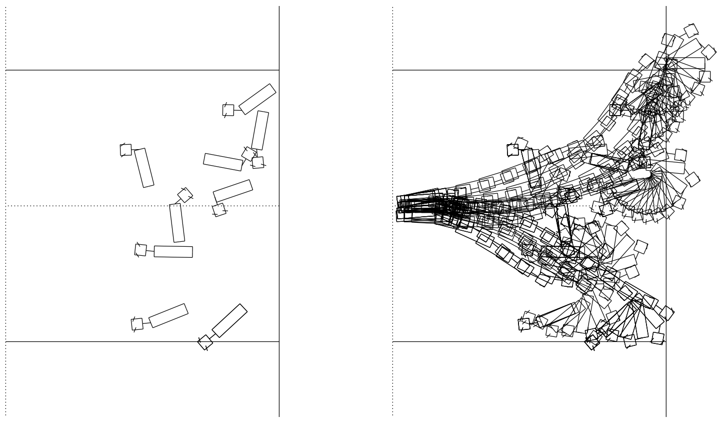

Figure 6: 8 random starting points and ensuing traffic

IV. DISCUSSION AND FUTURE WORK

Our study of the truck-and-trailer problem is complementary to, rather than competitive with the results in [Nguyen &Widrow], because our results apply to travel starting far from the goal, while their study yielded complex close-up manoeuvering behaviour which we have yet failed to reproduce.

This foray into genetic learning in neural nets has established the basic feasibility of the appproach, howewver it has also raised a number of specifc issues:

- Sizing the nets is always a problem , however we have not seen it posed in the context of supplying a net with additional, redundant, renormalized copies of some inputs. In our experiments with the truck this has proven useful. We have employed the same technique with success in fine-tuning the controller of a lunar-lander simulation [Ronald &Schoenauer 93].
- We have also seen that in some cases, in particular for the single-start trajectories, the number of necessary input variables can be reduced. We went from 4 to 3 inputs without performance degradation, ultimately perhaps all precise topographic inputs might be eliminated. The resultant blind net would still receive some time or elapsed distance input, or even embody a looped recurrent architecture with no input at all.
- The computation of the fitness in the multi-point learning utilizes the fitness employed for the single-point case, in our study. However, the fitness values computed for each starting point in the training ensemble must still be summated or amalgamated. The exact form of the summation needs fine-tuning. Also, the trajectory computations become very time consuming; runs can take hours on a workstation.
- If it were feasible not to represent all the starting points in the training ensemble, then the genetic search procedure could be notably speeded up. In [Mitchell, Haber & Crutchfield], in the context of evolving cellular automata, a different random sampling of the training ensemble is presented at each generation; the genetic algorithm embodies elitism to the extent of passing 20% of the population unchanged from generation to generation. We will investigate this strategy in future work.

References

[Goldberg 89] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, 1989.

[Hecht-Nieslen 90] R. Hecht-Nielsen, *Neurocomputing*, Addison Wesley, 1990.

[Hecht-Nieslen 92] R. Hecht-Nielsen, The Munificence of High Dimensionality, in *Artificial Neural Networks, 2*, I. Aleksander and J. Taylor editors, Elsever Science Publishers, pp 1017-1030, 1992

[Holland 75] J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Harbor, 1975.

[Michalewicz 92] Z. Michalewicz, Genetic Algorithms+Data Structures=Evolution Programs, Springer Verlag 1992.

[Mitchell, Haber & Crutchfield] N. J. Radcliffe, Revisiting the Edge of Chaos: Evolving Cellular automata to perform Computations, in *Complex Systems*,1993 in Press.

[Nguyen & Widrow 90] D. Nguyen, B. Widrow, The truck Backer Upper: An example of self learning in neural networks, in *Neural networks for Control*, W. T. Miller III, R. S. Sutton, P. J. Werbos eds, The MIT Press, Cambridge MA, 1990.

[Radcliffe 91] N. J. Radcliffe, Equivalence Class Analysis of Genetic Algorithms, in *Complex Systems* **5**, pp 183-205, 1991.

[Ronald & Schoenauer 93] E. Ronald, M. Schoenauer Genetic Lander, Tech Report 295, Centre de Mathematiques Appliquees, Ecole Polytechnique, Dec 1993.

[PDP] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing - Exploration in the micro structure of cognition*, MIT Press, Cambridge MA, 1986.

[Schoenauer & Ronald 93] M. Schoenauer, E.Ronald S.Damour, Evolving Networks for Control, in *Neuronimes 93* , EC2, Paris 1993.