Can learning robot solve 2-D jeep problem?

Akira Imada

Brest State Technical University
Moskowskaja 267, Brest 224017 Republic of Belarus
akira@bstu.by

Summary. Although the topic of the *"robot navigation through learning"* has quite a long history, we still have many problems which are very hard to be solved, if not at all. In this paper, we propose, as a challenge, two such benchmarks in a world of $N \times N$ two-dimensional grid which is simulated inside the computer. The first task is to look for one unique location in the grid which was specifically specified in advance, and the robot has no *a priori* information of the location. A series of trials by random walk requires $O(N^2)$ steps for this task. Then question is, *"Whether a learning reduces the number of steps to reach the location, or not?"* The other task is a two-dimensional extension of a so-called *jeep-problem*. We modified the original one-dimensional version of the problem for a robot navigation. A robot navigates a jeep which can carry a limited amount of fuel, starting from its base where robot can return later to refill the fuel. The jeep has a container to put some of its fuels somewhere to use in future. The goal is to explore the desert as long as possible by repeating the procedure – (i) start the base; (ii) navigate the desert; (iii) put fuels somewhere; and (iv) return to the base. Or, here, to reach the exit of the desert far apart from the base. The task is extremely difficult and most robots less likely to survive in the desert.

Keywords: learning navigation robot, jeep problem, a needle in a haystack.

1 Introduction

Sometimes, we observe a random behavior of a computer system works better than a learned or intelligently designed system. Can we say "no," when we talk about this issue concerning a robot navigation? This is the topic of this paper. We propose, for the purpose, two benchmarks for a navigation robot which is supposed to elaborate its behavior by learning through a number of trials.

Both benchmarks can be accomplished more or less by random behavior. Question, however, is if a learning scheme can make the performance more

efficiently than those by random behavior via a series of experiences of multiple trials later.

In this paper, tasks are not in a real physical world but simulation inside computers. A navigation robot explores a fictitious world of $N \times N$ grid.

The first task is a kind of two-dimensional version of *a-needle-in-a-haystack problem.* The task is to look for a uniquely pre-determined location of the grid. The robot has no information of where the needle is. The navigation is, at the first trial, by random walk, in which the robot can eventually find the location unless N is very large. An average of steps required is $O(N^2)$. Question is whether the robot can minimize the path length by a learning algorithm later?

The other task is more demanding. We extended a kind of mathematical puzzle called a *jeep-problem* in which a jeep explores one-dimensional desert under a constraint. A robot here navigates a two-dimensional grid again. While in the first task we do not assume energy consumption of the robot, the jeep needs fuels to move from one location to the next.

We now take a look at the problems more in detail.

2 Two Challenging Benchmarks

2.1 Can learning robot search it more efficiently than before?

We assume here $N \times N$ grid as our world. Location in the lattice is given by an integer coordinate (i, j) where $i, j = 1, 2, \cdots, N$. The entrance of the world is (1,1) and the exit is (N, N), for example. Starting at the entrance, most robot can navigate itself eventually to find the exit even by just a *random walk*, unless the grid is very large. The expectation of number of steps of a robot to reach the exit is $O(N^2)$. The result of our experiment of seeking the average number of steps in 100 runs are plotted in Fig. 1 as a function of N^2. This might be called *a two-dimensional version of a-needle-in-a-hay-stack* problem. [1]

The question then will be, "A learning can enhance the efficiency?" That is to say, "If a robot try it multiple times under a learning scheme, then the number of steps of the robot to reach the exit becomes shorter than the previous trials, or hopefully minimized?"

[1] The problem in general from a computational context was firstly described in 1987 by Hinton & Nowlan [1] as *a needle being a unique configuration of 20-bit binary string while all other configurations being a haystack.*

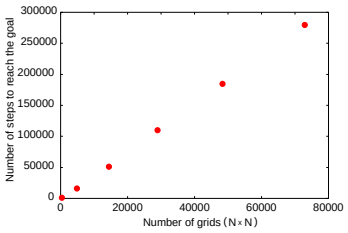

Fig. 1. Average number of steps during 100 runs until a robot who explores the $N \times N$ grid starting from $(0, 0)$ by a random walk eventually reaches (N, N) as a function of number of cells in the grid N^2.

Let us now summarize the problem asking readers to allow a little redundancy.

Challenge 1 (2-D Needle in a Haystack Problem)
Assuming $N \times N$ toroidal lattice each of whose cells is expressed by (i, j) where $i, j = -N, \cdots, 0, 1, 2, \cdots, N$, and the lattice has only one exit at (N, N). Starting from the base at (0,0) a robot navigate the gred from one cell to another. Then, can the robot learn how to take a path shorter than before through a learning of multiple trials later?

2.2 Can learning robot survive in a desert with a jeep?

Next, let us imagine that we are in a base located at the center of a desert. Again our world is a two-dimensional grid. This time, for some reasons which later become clear, size is 77×77, the coordinate of the bottom-left corner is $(-38, -38)$, and the top-right corner is $(38, 38)$ which is the only exit of the desert. The base is located at the origin $(0, 0)$. The grid is toroidal, that is, if the coordinate becomes $(N + 1)$ and $-(N + 1)$ then it is replaced with $-(N + 1)$ and $(N + 1)$, respectively.

A robot leave the base with a jeep. The jeep moves the desert of grid from one cell to the next, each time by consuming one unit of fuel. The jeep has a tank for fuel whose capacity is 30 units. The jeep also has a container with which the robot can store some amount of fuel in the tank to put at any location in the desert for the next time usage. Since the exit is 77 *Manhattan-distance* apart from the base, the tank full of 30 units are not enough to reach to the exit. The robot is allowed to go back to the base twice to refill the tank.

This is an extension of so-called a *jeep problem* where a jeep should maximize its penetration to one-dimensional desert under a condition. See, for example, the WWW page of *Wolfram MathWorld*

(`http://mathworld.wolfram.com/JeepProblem.html`).

It reads:

> *"Maximize the distance a Jeep can penetrate into the (one-dimensional) desert using a given quantity of fuel. The Jeep is allowed to go forward, unload some fuel, and then return to its base using the fuel remaining in its tank. At its base, it may refuel and set out again. When it reaches fuel it has previously stored, it may then use it to partially fill its tank. This problem is also called the exploration problem (Ball and Coxeter 1987)."*

As far as we know, this has never extended to a two-dimensional world. As you might find it already, the solution in our context is as follows. You now notice the reason why those parameters here are highly artificially devised. It is to fit the problem.

We now paraphrase the known solution of the original version.

1) Start with 30 units of fuel.
2) Go forward 10 distances, put 10 units, and then go back with the remaining 10 units of fuels.
3) Go forward 10 with 30 units refilled, spending 10 units, and get 10 units there.
4) Go forward 6 further, spending 6 units and put there 8 units.
5) Go back to the base spending remaining 16 units
6) With 30 units again, go forward 16 , spending 16 units, and get 8.
7) Go forward further until spending all the remaining fuel, and eventually reach the point which is 38 apart from the base.

This is how the jeep can penetrate to the desert with the maximum distance when allowed to go back to the base twice. Hence, in our context:

Challenge 2 (Jeep's survival in a desert) *Assume $N \times N$ toroidal lattice each of whose cells is expressed by (i,j) where $i, j = -38, -37 \cdots$, 0, 1, 2, $\cdots$, 37, 38. We call it a desert. The desert has only one exit at (38,38). Starting from (0,0) a robot navigates a jeep from one cell to the next. In order for the jeep to move one cell, it needs to spend one unit of fuel, and the jeep has the tank whose capacity is 30 units. The jeep also has a container with which the robot put some amount of fuel to any location of the desert for the next time usage. Allowing to go back to the base twice, can the robot learn how to reach the exit through a multiple times of failure experiences?*

3 Possible Approaches

Though, at this moment of writing this article, none of our experiments below or other attempts has not given us a satisfactory result, we show the followings just as strategies to start with, if any.

3.1 Messy Genetic Algorithm

One of the simplest approaches might be by using a genetic algorithm (GA) representing the path of the robot by what they call a *chromosome*. The problem, however, would be a length of the chromosome. Usually GA evolves a population of fixed length of chromosomes. The path of the robot, however, is different from navigation to navigation. Then one idea is to use a so-called *messy genetic algorithm* (mGA) in which flexible length of chromosomes can be exploited [2].

In mGA each gene is tagged with an additional number which expresses the location of the gene in its chromosome when created at random in the first generation. With the selection operation being similar with the standard GA, *crossover* is done with two operations called a *splice* and *cut*. The *cut* splits a string at a position chosen at random. Each of selected two parents are spliced into two parts. The first part of the first parent and the last part of the second parents, or vice versa, are concatenated and thus construct one offspring. So this reproduction works without taking care of parents' and offspring's length.

We now take a brief look at how we can apply mGA to a robot navigation. Assuming we have four different gene values, 1, 2, 3, 4 which means move right, left, up and down, respectively. As an example, assume now two parents are

$$((12)(22)(31)(42)(52)(63)(73)(83)(94)) \tag{1}$$

and

$$((13)(23)(34)(44)(52)(63)(73)(84)(93). \tag{2}$$

Then an extreme example from those possible offspring would be

$$((12)(22)(31)(42)(52)(63)(73)(83)(34)(44)(52)(63)(73)(84)(93)) \tag{3}$$

and

$$((13)(23)(94)). \tag{4}$$

See Fig. 2 below.

Fitness might be hamming distance of the closest point the path ever approached to (N, N).[2] Thus, starting with a population of those chromosomes this mGA creates chromosomes whose length ranges from 2 to 98 in the 2nd generation.

One thing to be added is how we interpreted the strings. First of all, we neglect the order of the genes in the chromosome. Then if we find the identical tag

[2] Though some readers might think this would be to evolve the length of chromosomes from short ones to long ones, it is rather opposit actually.

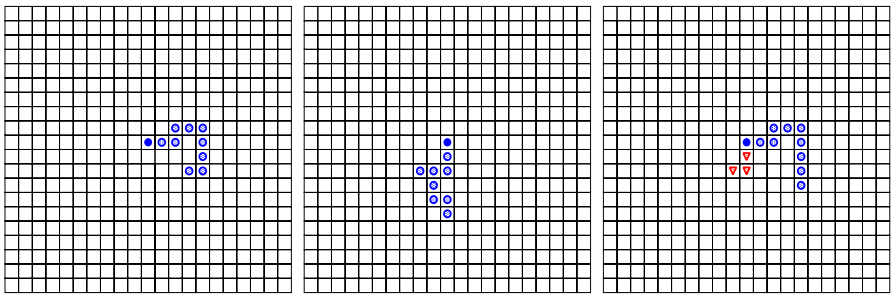

Fig. 2. Three fictitious paths represented in the text by the strings (1) (Left), (2) (Center), and (3)-(4) (Right).

more than once, we select one gene by a *first-come-first-serve* strategy. When we have no tag of the specific number we simply take no notice of the gene. Thus, for example,

$$((12)(43)(62)(83)(34)(82)(13)(63)) \tag{5}$$

is interpreted as

$$((12)(43)(62)(83)(34)) = ((12)(34)(43)(62)(82)). \tag{6}$$

In other words, evolution could yield tremendously long strings but in actual interpretation is that any strings is shorter than the length of those in the first generation.

Also see two examples of the path which are represented by such mGA chromosomes in Fig. 3.

3.2 Learning Navigation with Memory

It is natural to employ a *memory* for an effective navigation of robot, which allows the robot to learn from previous experiences. Hence we have had lots of such reports. See, for example, Remazeilles & Chaumette (2007) [3]. Most closely related, among others, is the work by Srinivasan (2006) the title reads *"... Memory-Based Learning Schemes for Robot Navigation in Discrete Grid-Worlds ..."* The author wrote, *"We attempt to find a solution to the problem of robot navigation using different memory-based learning schemes, and compare these schemes in terms of learning speed and efficiency."*

Our grid world, however, has no walls, no obstacles, or no corridors to be followed. It is really simple world. What makes a navigation difficult is no information about the destination. Under this simple situation our concern is to minimize Manhattan distance of the path. Hence robots must have a memory to store previous paths experienced.

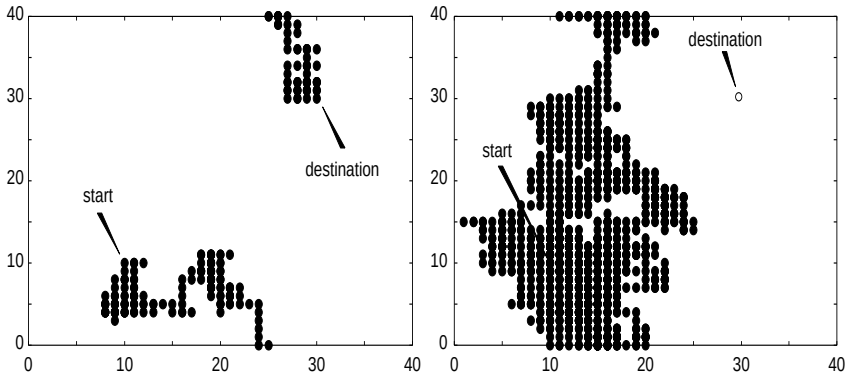

Fig. 3. Two examples of the path represented by a messy-GA chromosome starting from (10,10) with the destination being at (30,30) in 40 × 40 grid-world. The one succeeded to reach the destination (Left) and the one failed (Right).

3.3 Quantum Robot approach

Since Grover's [5] assertion in 1997 that quantum mechanics helps in searching for a needle in a haystack with $O(\sqrt{N})$ steps while classical computer requires $O(N)$ steps,[3] a fair amount of approaches exploiting a quantum random search has been proposed. See, for example, Shenvi (2003) et al. [6]. As for searching a space by a mobile robot, Beninof (2002) proposed a *quantum robot* [7]. It might be interesting to see what Beninof wrote:

> *"For this initial memory state all 2^N searches are carried out coherently. Since the path lengths range from 0 to 2^N, the quantum robot can search all sites of R and return to the origin in $O(N \log N)$ steps. Since this is less than the number of steps, $O(N^2 \log N)$, required by a classical robot, the question arises if Grover's algorithm can be used to process the final memory state to determine the location of S. If this is possible, the overall search and processing should require $O(N \log N)$ steps which is less than that required by a classical robot."*

At the same time, however, we must notice that search for a point in d-dimensional hypercube by a quantum random walk is $O(\sqrt{n})$ for $d \geq 3$ and $O(\sqrt{n \log^5 n})$ for $d = 2$ [8]. It would be more effective only in a dimension higher than 3.

4 Discussion

For the first benchmark, we have no walls or no obstacles. Hence, no dilemma of *"local information vs. knowledge of global geometry"* such described in

[3] As N is the number of points in search space in his equation, it is $O(N^2)$ in our context.

Salichs et al. [10]. If the robot starts navigation from (0,0) with goal being (N, N), for example, the task is just find a path which is made up of N movements to right and N movement to upwards. Nevertheless, it is hard for a robot to learn this after a couple of trials by random walk, if not at all.

As Knaden et al. wrote *"Desert ants use path integration as their predominant system of long-distance navigation"*, one of what we have in mind is to apply a nature's wisdom to this problem. The authors went on to write according to their observation of real ants in desert in Tunisia, "Ants had reset their home vector to zero state, and had therefore been able to reload their learned feeder vector, and consequently departed from the nest in the feeder direction."

A search by quantum random walk has already been mathematically proofed to be more efficient than the one with our currently available computers. So far, however, attention has been just on general search on hypercube. We think the second benchmark is a good one also as a test for the quantum computation, besides as a generall issue on a robot navigation, because we could design a *wave-function* quite easily which include an information on both geometry and the place where reserved fuels are. This is our motivation of this study too.

References

1. Hinton, G. E.,and S. J. Nowlan (1987) "How Learning Can Guide Evolution?" Complex Systems, Vol. 1, pp. 495–502.
2. Goldberg, D. E., B. Korb, and K. Deb (1989) "Messy genetic algorithms: motivation, analysis, and first results." Complex Systems, Vol. 3, pp. 493–530.
3. Remazeilles, A., and F. Chaumette (2007) "Image-based robot navigation from an image memory." Robotics and Autonomous Systems archive, Vol. 55(4), pp. 345-356.
4. Srinivasan, B. (2006) "Analysis of Memory-Based Learning Schemes for Robot Navigation in Discrete Grid-Worlds with Partial Observability." Available at author's WWW page but the author says this is just "Other paper." `(http://www.stanford.edu/~bharsrin/docs/papers/)`
5. Grover, L. (1997) "Quantum mechanics helps in searching for a needle in a haystack." Physical Review Letter, Vol. 79, pp. 325–328.
6. Shenvi, N., J. Kempe, and B. Whaley (2003) "A Quantum Random Walk Search Algorithm." Physical Review A, Vol. 67, pp. 052307–052318.
7. Benioff, P. (2002) "Space searches with a quantum robot" Contemporary Mathematics, Vol. 305, pp. 1–12.
8. Aaronson, S.,and A. Ambainis (2003) "Quantum search of spatial regions." Proceedings of IEEE Symposium on Foundations of Computer Science, pp. 200–209.
9. Knaden, M., and R. Wehner (2006) "Ant navigation: resetting the path integrator." Journal of Experimental Biology Vol. 209, pp. 26–31.
10. M.A. Salichs, M .A., and L. Moreno (2000) "Navigation of Mobile Robots: Open Questions." Robotica, Vol. 18, pp. 227-234.