Jeep problem and its 2D extension – the solution with an application of the reinforcement learning

Marcin Pluciński

Faculty of Computer Science and Information Technology,
Szczecin University of Technology, Żołnierska 49, 71-210 Szczecin, Poland,
mplucinski@wi.ps.pl

Abstract: The paper presents an application of the reinforcement learning in searching of a solution of the classic Jeep problem and its 2D extension. Our task here is to find a policy of a vehicle moving. A vehicle has some amount of fuel at its disposal and has to cover a given distance (in a classic 1D problem) or a given surface (in a 2D problem). Additional assumption is that some amounts of fuel can be left on the route for a future usage. The continuous problem in unrealistic so a main work was concentrated on the discrete Jeep problem. There is examined and described an influence of main learning parameters on a learning speed and there are presented some found exemplary policies for different problem conditions. The learning time was quite big for long action sequences, but solutions were found for all tested task conditions and finally the method can be considered as a uniform tool for the discrete problem policy searching.

Keywords: Jeep problem, exploration problem, reinforcement learning

1. Introduction

A Jeep problem, also commonly known as an exploration problem, is an important and interesting logistic task that can have a great weight, exemplary for planning of research expeditions or military operations. In the classic problem we have the given amount of fuel and we want to find such a vehicle movement policy that it travels the greatest distance. We assume that a vehicle tank contains only one unit of fuel and it allows for covering only one unit of the distance. The vehicle can leave on its way any amounts of fuel to use them in the future.

For the first time the problem was formulated over 1000 years ago in "*Propositiones ad acuendos iuvenes*" – one of the oldest mathematical puzzle collection in Latin [1, 13]. In 1947, N. Fine presented the problem in a modern mathematical literature and showed its exact solution [4]. From that time, the problem was searched in many different versions. First of all, many discrete versions (as more realistic) were analysed [2, 6, 13]. In other works there were presented solutions for many vehicles [5, 11], for a stochastic version of the task [7] and for its 2-D extension [8]. In all publications, there were presented solutions for given specific task conditions. A change of them would force a construction of a completely new policy on a base of a completely new algorithm. In the literature, there is no uniform method of a solution finding for any conditions of the exploration problem. A proposition of such a method we can find in [10], where an evolutionary algorithm was applied for optimal policy searching, but the task was solved there only in a limited scope.

In the paper, there will be presented an application of the reinforcement learning for searching of the optimal movement policy. Such method can be used for any given exploration problem conditions, but in our case the task will be defined in a different manner.

First, we'll assume that we know the distance to cover and our task is to find the proper policy. Next, the task will be extended into 2 dimensions. We'll assume that a certain surface is given and our task is to find a policy of covering it. Like in the classic task, one unit of fuel allows for covering only one unit of the distance and fuel can be left on the surface for a future usage. In both cases (1D and 2D) we assume that the amount of fuel for our disposal is unlimited.

2. Jeep problem – a continuous and discrete task and its 2D extension

In the classic, continuous task a vehicle can cover any distances (freely small) and leave any amounts of fuel. Such task can be solved with an application of a recurrent policy [4, 10]. In the case when fuel amount $n = 0$, the vehicle can't move so a covered distance is $d_0 = 0$. For $n = 1$, the only reasonable policy is to tank all fuel and drive one unit of a distance: $d_1 = 1$. But for $n = 2$, the vehicle can tank 1 unit of fuel, drive 1/3 unit of a distance, leave 1/3 unit of fuel and move back to the start point. There, it can tank again 1 unit of fuel, drive 1/3 unit of a distance, tank again 1/3 unit of fuel left there earlier and in the end drive 1 unit of a distance. The maximum distance covered in that way is equal $d_2 = 1$ and 1/3 units. Such policy we will write symbolically in the form [10]:

$$(\mathrm{T}1), (\mathrm{M}\tfrac{1}{3}), (\mathrm{T}-\tfrac{1}{3}), (\mathrm{M}0), (\mathrm{T}1), (\mathrm{M}\tfrac{1}{3}), (\mathrm{T}\tfrac{1}{3}), (\mathrm{M}1\tfrac{1}{3}),$$

where: M – means a movement to the given position, and T – tanking (when the value is positive) or fuel leaving (when the value is negative).

It can be proved [4], that the maximal distance a vehicle can drive in this way is equal:

$$d_n = \sum_{k=1}^{n} \frac{1}{2k-1}. \tag{1}$$

For $n \to \infty$, the distance will tend towards ∞ too, and a number of possible optimal policies will be also infinite. The fact of existence more than one optimal solutions (for $n > 2$) additionally complicates the task. In the Fig. 1, there are presented optimal solutions for $n = 3$.

Policy 1: $(\mathrm{T}1), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}-\tfrac{3}{5}), (\mathrm{M}0), (\mathrm{T}1), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}-\tfrac{3}{5}), (\mathrm{M}0), (\mathrm{T}1),$
$(\mathrm{M}\tfrac{1}{5}), (\mathrm{T}\tfrac{1}{5}), (\mathrm{M}\tfrac{8}{15}), (\mathrm{T}-\tfrac{1}{3}), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}1), (\mathrm{M}\tfrac{8}{15}), (\mathrm{T}\tfrac{1}{3}), (\mathrm{M}1\tfrac{8}{15})$

Policy 2: $(\mathrm{T}1), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}-\tfrac{3}{5}), (\mathrm{M}0), (\mathrm{T}1), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}\tfrac{1}{5}), (\mathrm{M}\tfrac{8}{15}), (\mathrm{T}-\tfrac{1}{3}),$
$(\mathrm{M}\tfrac{1}{5}), (\mathrm{T}\tfrac{1}{5}), (\mathrm{M}0), (\mathrm{T}1), (\mathrm{M}\tfrac{1}{5}), (\mathrm{T}\tfrac{1}{5}), (\mathrm{M}\tfrac{8}{15}), (\mathrm{T}\tfrac{1}{3}), (\mathrm{M}1\tfrac{8}{15})$

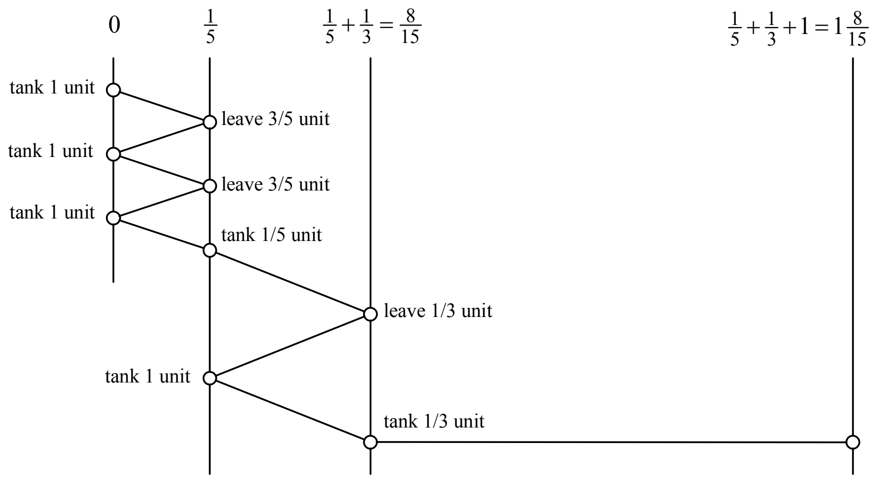


Figure 1. Two optimal policies for $n = 3$ (policy no 1 is illustrated in the figure).

The problem described above can become more difficult to solve when task conditions are changed. Moreover, it is unrealistic so next we pay the main attention to the discrete problem. Let's assume that a vehicle has unlimited amount of fuel at its disposal. Our task is to find a vehicle movement policy to cover the given distance *d* with the assumption that in one action a vehicle can drive any possible integer number of distance units or leave/tank any possible integer number of fuel units. A capacity of a vehicle tank equals *z*, and of course *z* and *d* are natural numbers.

Two-dimensional task will be considered also in a discrete version. We assume that on a surface that we want to cover there is a grid of uniformly located points, Fig. 2.

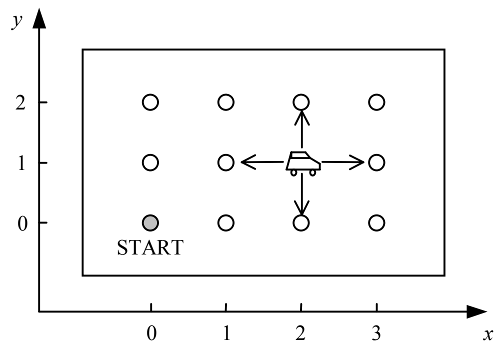


Figure 2. 2D Jeep problem – discrete grid of points and an auxiliary coordinate system.

The task will be realised if we find a policy of a vehicle movement that drives it through all points of the grid. In one step a vehicle can move only to one of its 4 neighbour points or it can leave/tank any possible integer number of fuel units. A distance between grid points equals one unit and the same as above the vehicle tank capacity equals z.

3. Reinforcement learning and TD algorithm

Generally, we can say that the main aim of the reinforcement learning is to find an optimal policy of solving a given task in an unknown environment. A learner can observe an environment state s_t and on its base it can choose an action a_t to do, according to its current policy. After each action, the learner observes the next state s_{t+1} and receives a reinforcement (a reward) r_t from the environment. The reward is a measure of an action quality. On the base of obtained observations, the learner tries to modify its policy to obtain better rewards in the future [9, 15].

The policy should depend only on the state so during learning we look for an optimal mapping [3]:

$$\pi : \quad S \to A, \tag{2}$$

where: π – a policy, S – a set of environment states and A – a set of all possible actions.

For the current learner policy π, we can define so called value function which is a mapping from a state s, to an entire amount of rewards the learner expects to receive in the future after policy starting from that state [3]:

$$V^{\pi}(s) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} \cdot r_t \mid s_0 = s\right], \tag{3}$$

where: E_π – expected (for the current policy π) sum of future rewards r_t, $\gamma \in (0,1]$ – discount rate, which determines that rewards received in the future are less worth for the state value.

During the reinforcement learning a learner looks for an optimal policy π^* – the policy for which it will always receives the best rewards from an environment. For such a policy, the value function $V^{\pi^*}(s)$ is always biggest or equal to the value function $V^{\pi}(s)$ for any policy π. If the optimal value function $V^{\pi^*}(s)$ is known, we can easily find the optimal action policy π^* as a greedy policy to $V^{\pi^*}(s)$. A greedy policy to $V(s)$ always chooses its action to maximize an immediate reward and a discounted next state value of a value function $V(s)$ [3, 12].

One of the most important reinforcement learning methods is a temporal difference (TD) algorithm [14, 16]. In this algorithm, we try to find the optimal value function $V^{\pi^*}(s)$.

At the beginning, the $V(s)$ function is initiated arbitrarily. During succeeding learning steps, the function is modified on the base of observed experiences $<s_t, a_t, r_t, s_{t+1}>$. The modification rule can be written as:

$$V_{t+1}(s_t) = V_t(s_t) + \eta \cdot \Delta , \tag{4}$$

where: η – step-size parameter and:

$$\Delta = r_t + \gamma \cdot V_t(s_{t+1}) - V_t(s_t) . \tag{5}$$

$V_t(s_t)$ – is the value of the current value function and $r_t + \gamma \cdot V_t(s_{t+1})$ – is a sum of a reward received in a step t and a discounted value of a next state – it is probably better evaluation of a real value of $V^{\pi}(s)$ (for a current policy π) than $V_t(s_t)$ [3, 12].

If actions are always chosen such that a next value of a sum of a reward and a function $V^{\pi}(s)$ is maximal, then our policy will be greedy to the function $V^{\pi}(s)$ and it can be proved [15] that it will be better or equal to the policy π. On the base of equations (4) i (5) we can formulate the complete formula for the function $V^{\pi}(s)$ adaptation:

$$V_{t+1}(s_t) = V_t(s_t) + \eta \cdot \left[\max_a \left(r_t + \gamma \cdot V_t(s_{t+1}) \right) - V_t(s_t) \right] . \tag{6}$$

A $\max_a$ operator means choosing the greedy policy. Iterative improving of the value function according to the formula (6) leads to a convergence to the optimal value function $V^{\pi^*}(s)$, for which the optimal policy can be easily found as greedy to it.

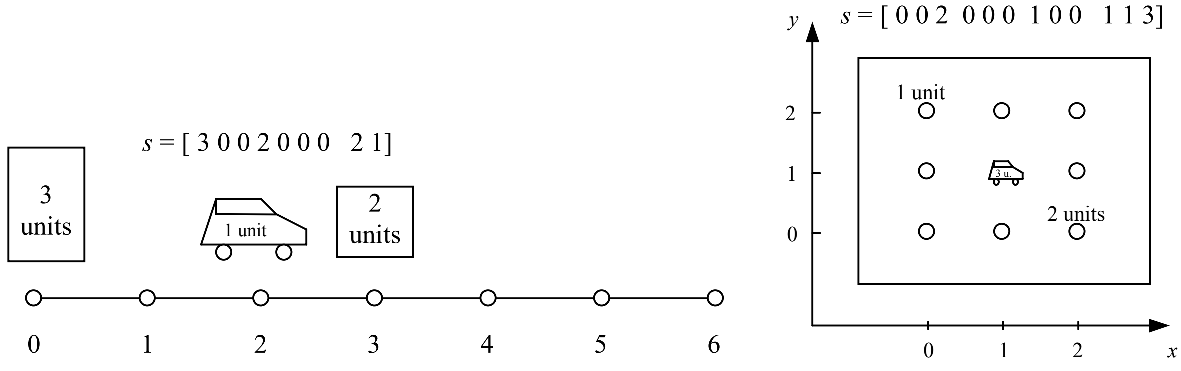
In practice, to ensure better searching of a possible solutions (policies) space the formula (6) isn't applied for improving $V(s)$ for all the time, but for most of the time. In each algorithm step, action can be taken randomly with a probability ε (of course after such action we don't modify the value function). So, we will tell about the ε-greedy policy, which will be compromise between better searching of possible policies space and a fast convergence to the optimal policy [12].

4. Searching for exploration policy

Now, we consider the discrete task described in the section 2. We assume that we have unlimited amount of fuel at our disposal and a fuel source is placed in the start point. In 1D version, in each step a vehicle position and a state of its tank can be changed by any possible integer value, and the capacity of a tank equals z. We want to find a policy which ensures a covering of a distance d, with the assumption that one unit of fuel is enough for driving one unit of a distance (z, d – are natural numbers). In 2D version, in each step a vehicle can move only to one of its 4 neighbours or the same as above the state of its tank can be changed by any possible integer value. In this case, we want to find a policy that ensures covering of all points of a discrete grid defined in the given surface.

If we want to take a decision about the action, we must know amounts of fuel in the start point and all midpoints and of course position of a vehicle and its tank content.

One of the most important tasks, during formulating the last version of an algorithm, is proper defining of a state. The environment state should have the Markov property – the decision about a taken action should depend only on the state value. If we assume that all points in which a vehicle can leave fuel are placed uniformly (and distance between them equals 1 unit) then first m+1 numbers in the state vector will describe available fuel amounts on the way (m is a natural number defining the maximal analysed range of a vehicle: $m \geq d$). 2 last values of the vector are numbers defining respectively a vehicle position and a content of its tank. Such defined state vector s is enough to take the decision about an action to do. Exemplary representation of the state vector is presented in Fig. 3.



$s = [\,3\;0\;0\;2\;0\;0\;0\quad 2\;1\,]$
3 units
1 unit
2 units
0
1
2
3
4
5
6
$s = [\,0\;0\;2\;\;0\;0\;0\;\;1\;0\;0\;\;1\;1\;3\,]$
y
x
1 unit
3 u.
2 units
0
1
2

Figure 3. Representations of the environment state in 1D and 2D Jeep problem.

In 2D task, the state vector will be defined similarly (Fig. 3). Let's assume that all points in a grid are placed uniformly and the distance between them equals one unit. If a grid size equals $n_x \times n_y$ (where: n_x – is the number of points in a horizontal direction and n_y – the number of points in a vertical direction) then $n_x \cdot n_y$ first values in the state vector will describe a fuel amount in grid points (taken by rows form lower to upper). The last 3 values in the vector will be: x-coordinate, y-coordinate of a vehicle and a vehicle tank content. Similarly as in 1D case, such defined state vector is enough to take the decision about an action to do.

The movement will always start in the point 0 or (0,0) with a full tank. Actions are taken according to an ε-greedy policy among all possible ones, with taking into account current vehicle position and its tank content. After each action, the environment goes to the next state and grants the reward. The reward value equals 0 when the task is not fulfilled or it equals 1 in other case. On the base of the taken greedy action, after moving to the next state and receiving the reward, the value function $V(s)$ is changed according to the formula (6).

The next important factor for the learning process is a way of a value function $V(s)$ representation. In experiments described in the next section, a table representation was applied. Each row of the table stored: the state vector value s and the value of the value function $V(s)$. The beginning value of the value function $V(s)$ for each state was taken as 0.5.

5. Results of experiments

5.1. 1D case

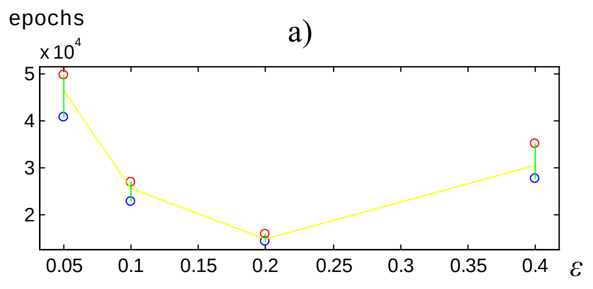
At the first stage of the research, an influence of method main parameters for its speed had to be found. In each experiment, there was counted a number of epochs needed for learning of a proper policy for covering a given distance. By the epoch, it was meant here a full learning cycle from leaving the start point to a moment when a further vehicle movement wasn't possible due to lack of fuel.

Parameters taken in experiments: given distance $d = 5$, tank capacity $z = 3$ and step-size parameter $\eta = 0.2$. Results are presented in the Fig. 4a and 4b. Plots show how learning speed (in epochs) depends on parameters ε and γ. Mean time of one epoch was equal about 1ms (research were made on a computer with a processor AMD Athlon 64 3000+, 2GHz clock).

The plot in the Fig. 4a presents an influence of ε coefficient ($\gamma = 0.9$). For each ε value there were made 10 trials. A continuous line runs through mean values found in the research. Additionally, ranges of variability are marked in the plot. The learning was fastest for $\varepsilon = 0.2$. Smaller values caused slower learning – it was a reason of a poor searching of a solutions space. For $\varepsilon = 0$, the learning method always used a greedy policy and the whole learning process became inefficient. Greater ε values caused greater learning time because a searching was too random.

The plot in the Fig. 4b presents an influence of γ coefficient ($\varepsilon = 0.1$). The same as above, for each γ value there were made 10 trials. The learning was fastest for $\gamma > 0.9$, but taking too great vales (for example $\gamma = 1$) caused that learning became inefficient.

Next experiments showed how learning time depends on a given distance d. Results of experiments are presented in Fig. 5 (parameters: $z = 3$, $\eta = 0.2$, $\varepsilon = 0.2$ and $\gamma = 0.9$). Additionally, there are presented some found policies. For the distance $d = 6$ learning time was very long – about 1300000 epochs (about 95 minutes). From the other side, the found policy consists of 79 actions, so its complication degree is quite big.



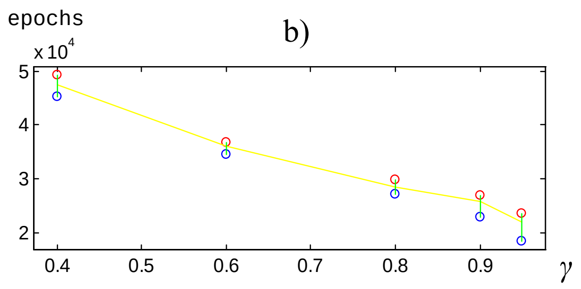
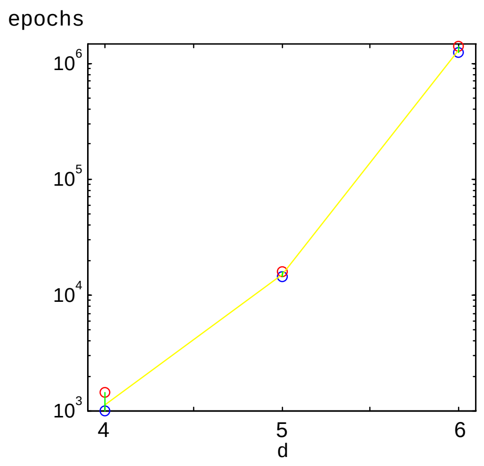


Figure 4. Influence of ε and γ parameters for learning time.



d	mean learning time (in epochs)	no. of actions in policy
4	1142	7
5	15058	25
6	1307977	79

$d = 4$:	M1 T-1 M0 T3 M1 T1 M4
$d = 5$:	M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T1 M2 T-1 M1 T1 M0 T3 M1 T1 M2 T1 M5
$d = 6$:	M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T-1 M0 T3 M1 T1 M2 T-1 M1 T3 M2 T-1 M1 T3 M2 T-1 M1 T3 M2 T-1 M1 T3 M2 T1 M3 T-1 M2 T3 M3 T1 M6

Figure 5. Learning time as a function of the given distance and exemplary found policies.

One of the greatest advantages of policy learning on the base of the reinforcement learning is possibility of its finding for different conditions taken in the task. Below, there are presented exemplary policies found for vehicles with different tank capacities. Parameters: $d = 6$, $\eta = 0.2$, $\varepsilon = 0.2$ and $\gamma = 0.9$.

z = 3:	policy is presented in Fig. 5
z = 4:	M1 T-2 M0 T4 M1 T1 M2 T-1 M0 T4 M1 T1 M2 T1 M6 mean learning time: 13800 epoch, policy length: 15 actions
z = 5:	M1 T-3 M0 T3 M1 T3 M6 mean learning time: 2620 epoch, policy length: 7 actions

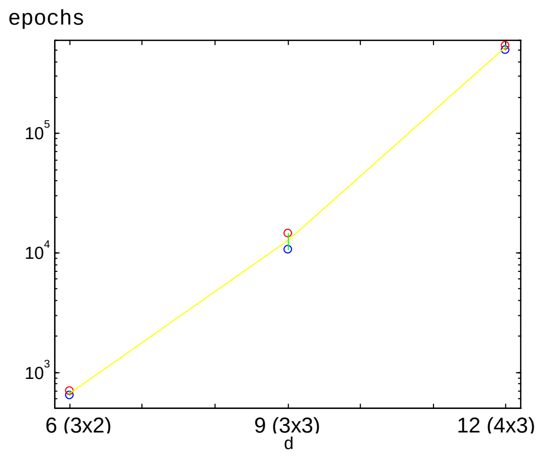
We can also find policies for environments in which fuel is initially set in some points of the route. Exemplary found policies are presented below. Polices were found for the given distance $d = 6$ and the tank capacity $z = 3$.

initial state x = [3 0 0 2 0 0 0 0 3]: M1 T-1 M0 T3 M1 T1 M3 T2 M6
mean learning time: 2400 epochs

initial state x = [3 0 1 0 1 0 0 0 3]: M1 T-1 M0 T3 M1 T1 M2 T1 M4 T1 M6
mean learning time: 3250 epochs

5.1. 2D case

In the second part of experiments, the exploration policy in 2 dimensions was searched. First experiments showed how learning time (in epochs) depends on the size of the covered surface. The size was represented by the grid points number. Results of experiments and exemplary found policies (for tank capacity $z = 4$) are presented in Fig. 6.



Size of environment: 3x2 grid points

M(0,1) T-1 M(0,0) T3 M(1,0) M(1,1) M(2,1) M(2,0)
policy length: 8 actions

Size of environment: 3x3 grid points

M(0,1) T-2 M(0,0) T4 M(1,0) T-2 M(0,0) T4 M(0,1) T1 M(0,2) T-1 M(0,1) M(0,0) T4 M(1,0) M(2,0) T-1 M(1,0) T1 M(0,0) T4 M(0,1) T1 M(1,1) M(1,2) M(2,2) M(2,1)
policy length: 28 actions

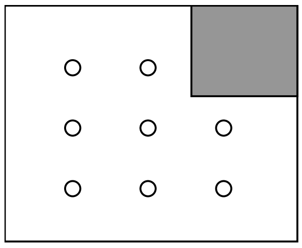
Figure 6. Learning time as a function of the given surface size and exemplary found policies.

The method based on the reinforcement learning is a very uniform tool and we can look for policies for surfaces with different shapes or with obstacles. Some examples of such environments with policies found are presented in Fig. 7.

6. Conclusions

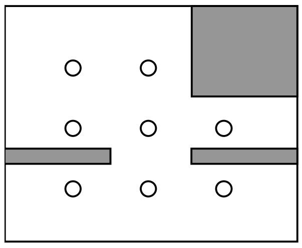
The presented method is a very effective tool in the exploration policy searching both for 1D and 2D case. Solutions can be found for different task conditions (different tank capacities, given distances or environments to cover). The learning time was quite big for long action sequences, but solutions were found for all tested task conditions and finally the method can be considered as a uniform tool for discrete problem policy searching.

The method has to deal with the curse of dimensionality – for the given distance $d = 7$ and the tank capacity $z = 3$ time of calculations would probably last about 10^9 epochs (extrapolation on the base of the plot from Fig. 5) and it means many days of calculations. However for simpler tasks, for which the solution has a form of a several dozen actions sequence, searching time is relatively short.



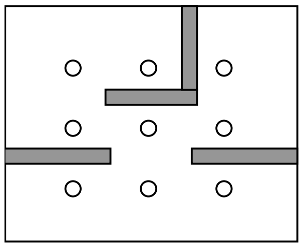
$z = 4$

21 actions:
M(0,1) T-2 M(0,0) T4 M(0,1) T1 M(0,2) T-1 M(0,1) M(0,0) T4 M(1,0) T-2 M(0,0) T4 M(1,0) T1 M(2,0) M(2,1) M(1,1) M(1,2)



$z = 4$

43 actions:
M(1,0) T-2 M(0,0) T4 M(1,0) T-1 M(0,0) T3 M(1,0) M(1,1) T-1 M(1,0) T1 M(0,0) T4 M(1,0) T-2 M(0,0) T4 M(1,0) T1 M(2,0) T-1 M(1,0) M(0,0) T4 M(1,0) T1 M(1,1) M(2,1) T-1 M(1,1) T1 M(1,0) T1 M(0,0) T4 M(1,0) T1 M(1,1) M(1,2) M(0,2) M(0,1)



$z = 5$

36 actions:
M(1,0) M(2,0) T-1 M(1,0) M(0,0) T5 M(1,0) T-3 M(0,0) T5 M(1,0) M(1,1) T-2 M(1,0) T1 M(0,0) T5 M(1,0) T1 M(1,1) T1 M(2,1) M(2,2) T-1 M(2,1) M(1,1) T1 M(1,0) T1 M(0,0) T5 M(1,0) M(1,1) M(0,1) M(0,2) M(1,2)

Figure 7. Exemplary environments and found policies.

Thanks to discount rate $\gamma < 1$ an algorithm looks for policies as short as possible because value function $V(x)$ is biggest in such cases. So, it can't be proved that policies found in experiments are optimal, but we can observe that they are short and in that way very close to best ones.

Bibliography

1. Ball W.W.R, Coxeter H.S.M.: *Mathematical Recreations and Essays*, 13edn, New York: Dover, 1987.
2. Brauer W., Brauer U.: *Reconsidering the Jeep Problem – or how to transport a birthday present to Salosauna*, Proceedings of the Colloquium in Honor of Arto Salomaa on Results and Trends in Theoretical Computer Science, pp. 30-33, 1994.
3. Cichosz P.: *Learning systems*, Wyd. Naukowo-Techniczne, Warszawa, 2000 [in Polish].
4. Fine N.J.: *The Jeep Problem*, The American Mathematical Monthly, v. 54, nr 1, pp. 24-31, 1947.
5. Franklin J.N.: *The range of a fleet of aircraft*, Journal of the Society for Industrial and Applied Mathematics, v. 8, nr 3, pp. 541-548, 1960.

6. Gale D.: *The Jeep Once More or Jeeper by the Dozen*, The American Mathematical Monthly, v. 77, nr 5, pp. 493-501, 1970.
7. Giffen W.J.: *Deterministic and stochastic extensions of the Jeep problem*, PhD Thesis, Purdue University, 2004.
8. Imada A.: *Can learning robot solve a 2-D jeep problem?* Proceedings of the International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE'2007), Buenos Aires, Argentina, 2007.
9. Kaelbling L.P., Littman M.L., Moore A.W.: *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research, v. 4, pp. 237-285, 1996.
10. Klęsk P.: *The Jeep Problem, searching for the best strategy with a genetic algorithm*, Information Processing and Security Systems, v. 3, Springer US, pp. 453-464, 2005.
11. Phipps C.G.: *The Jeep Problem: A more general solution*, The American Mathematical Monthly, v. 54, pp. 458-462, 1947.
12. Pluciński M.: *Application of the probabilistic RBF neural network in the reinforcement learning of a mobile robot*, Proceedings of the 14th International Multi-Conference on "Advanced Computer Systems", Międzyzdroje, Poland, 2007.
13. RoteG., Zhang G.: *Optimal logistics for expeditions – the jeep problem with complete refi-lling*, 'Spezialforschungsbereich F 003' Technical Report No. 71, TU Graz, Austria, 1996.
14. Sutton R.S.: *Learning to predict by the methods of temporal differences*, Machine Learning, v. 3, pp. 9-44, 1992.
15. Sutton R.S., Barto A.G.: *Reinforcement learning: An introduction*, The MIT Press, 1998.
16. Tesauro G.: *Practical issues in temporal differences learning*, Machine Learning, v. 8, pp. 257-277, 1992.