Comparing Exploration Approaches in Deep Reinforcement Learning for Traffic Light Control

Yaniv Oren, Rolf Starre, and Frans Oliehoek

Delft University of Technology

Y.Oren@student.tudelft.nl, R.A.N.Starre@tudelft.nl, F.A.Oliehoek@tudelft.nl

Abstract

Identifying the most efficient exploration approach for deep reinforcement learning in traffic light control is not a trivial task, and can be a critical step in the development of reinforcement learning solutions that can effectively reduce traffic congestion. It is common to use baseline dithering methods such as ϵ-greedy. However, the value of more evolved exploration approaches in this setting has not yet been determined. This paper addresses this concern by comparing the performance of the popular deep Q-learning algorithm using one baseline and two state of the art exploration approaches, and their combination. Specifically, ϵ-greedy is used as a baseline, and compared to the exploration approaches Bootstrapped DQN, randomized prior functions, and their combination. This is done in three different traffic scenarios, capturing different traffic profiles. The results obtained suggest that the higher the complexity of the traffic scenario, and the larger the size of the observation space of the agent, the larger the gain from efficient exploration. This is illustrated by the improved performance observed in the agents using efficient exploration and enjoying a larger observation space in the complex traffic scenarios.

1 Introduction

Traffic congestion is a global pandemic. For instance, in the EU alone its cost is estimated to be 1% of the EU's GDP [1]. One approach for reducing this cost is optimization of traffic flows by improving traffic light control policies. To find such policies, reinforcement learning (RL) has a strong appeal as a paradigm that is able to find high performance solutions to sophisticated problems. Research has been done into the application of RL to the problem of traffic light control optimization in the past [2]–[6], often specifically concerning application of deep RL algorithms [2], [5], [6].

In RL, an agent operates in an environment, observes a state, executes actions, receives rewards, and ultimately, attempts to find an optimal policy for maximized cumulative reward over time. Often, the state-action space is too large for the agent to be able to keep a tabular representation of its learned policy. This led to the development of deep RL [7]. In deep RL, the agent makes use of neural networks as nonlinear function estimators. The agent can then use this estimation to prioritize between the different actions available in each state of the environment.

A fundamental principle of RL is exploration, and the balance between exploration and exploitation. Namely, how much does the agent explore its environment, versus how much it opts for actions that it expects to return the most cumulative reward. Many different exploration approaches and algorithms exist for reinforcement learning, from the common baseline ϵ-greedy [7], to addition of random noise [8], optimism in the face of uncertainty [9], Bootstrapped DQN [10], randomized prior functions [11], and others. These approaches often perform differently in different settings, in addition to having different computational costs [10], [11]. It has been shown that for some RL settings, simple exploration approaches such as ϵ-greedy are insufficient for RL to be able to perform well, or at all [10],[11], which can be caused by the reward function used and the complexity of the specific problem tackled. This illustrates the importance of identifying effective exploration techniques for specific RL settings. To the best of our knowledge, there has not been an attempt to investigate the importance of efficient exploration in deep RL in the setting of traffic light control.

This paper investigates a comparison between different exploration approaches in deep RL for traffic light control. This, to facilitate better deep RL by identifying the value of evolved exploration approaches in this setting, such as higher sample efficiency, or higher final policy score. For that purpose, this paper compares the performance of the popular deep Q-learning algorithm (DQN) [7] using one baseline and two state of the art exploration approaches, and their combination. Specifically, ϵ-greedy is used as a baseline, and compared to the exploration approaches Bootstrapped DQN, randomized prior functions, and their combination. This is done in three different traffic scenarios, ranging from simplified to simulating real traffic, in order to investigate the effect of exploration in different traffic profiles.

This paper first introduces a theoretical background for deep RL and exploration. Second, a description of the exploration techniques compared, along with the modeling of traffic light control as an RL problem, are given. This is then followed by an explanation of the research methodology and the experimental setup, leading to a presentation of the results obtained and their analysis. Last, ethical and epistemic concerns are considered, implications of the work are discussed and conclusions are drawn.

Altogether, the results obtained suggest a link between the complexity of the traffic scenario, the amount of information accessible to the agent, and the gain from efficient exploration. This is illustrated by the improved performance observed in the agents using efficient exploration and enjoying a large observation space, in the complex traffic scenarios.

2 Background

This section introduces background information relevant to the work presented in this paper. First, an overview of reinforcement learning (RL), is given, laying the basis for an introduction to deep RL and a description of the deep Q-learning (DQN) algorithm [7] that follows. Last, the principle of exploration is explained, followed by an overview of dithering, deep and directed exploration.

2.1 Reinforcement learning

In RL, an agent operates in an environment. The environment provides information regarding the state the agent is in and what actions it can execute. The environment is usually described in the form of a Markov decision process (MDP), a stochastic control process often used to model decision making in partially stochastic domains. A Markov decision process

is represented as a four-tuple, $M = (S, A, P, R)$, where S represents the state space, A the action space, P the transition function and R the reward function.

The agent interacts with the environment by observing a state $s_t \in S$, executing an action $a_t \in A$, and receiving a reward $r_t \in R$ for the action executed. The agent is attempting to learn a policy π, such that the expected reward over time $E[R_t] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$ is maximised. This, where $0 \leq \gamma \leq 1$, is a discount factor, r_{t+1} is the reward received for the action a_t, the action chosen at time t, and r_{t+k+1} is the reward expected to be received for executing the action at moment $t + k$, chosen based on the policy π. There are different RL approaches to finding such a policy. The algorithms investigated in this paper use an approach that focuses on the agent learning the expected reward over time from taking specific actions in specific states. The value of this state-action pair in terms of its expected reward is termed the Q-value, and the RL algorithm based on learning it is called Q-learning [12].

In the Q-learning algorithm, the value of state-action pairs is estimated by the agent, using iterative Bellman updates: $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[y_t - Q_t(s_t, a_t)]$, where α is the learning rate, and the target $y_t = r_t + \gamma \max_a Q_t(s_{t+1}, a)$. s_{t+1} denotes the new state arrived at after choosing action a_t in state s_t, and a any action available at state s_{t+1}. In many scenarios however, the state-action pair space is too large for the computation or memorisation of each value $Q(s, a)$ to be tractable. To avoid this problem, function estimators such as neural networks can be used to estimate the Q value. This gives rise to the use of neural networks in reinforcement learning, and specifically the deep Q-Learning (or deep Q-networks) algorithm, commonly referred to as DQN [7], [13].

2.2 Deep reinforcement learning with DQN

The DQN algorithm uses deep neural networks to estimate a mapping from states to Q-values [7]. Instead of saving or computing each $Q(s, a)$ value separately, the algorithm learns a parameterized value function $Q(s, a; \theta_t)$. As a result, rather than learning the Q-values directly, the algorithm learns the parameter set θ of the Q-function. The previous Q-learning update then becomes:

$$\theta_{t+1} = \theta_t + \alpha(y_t - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t)$$

Here, $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t)$. To prevent instabilities, the DQN algorithm uses an additional network, termed the target network, θ^- [7]. The target network is the same as the regular, or online, network. However, it only updates every certain τ time-steps, by copying the parameters θ of the online network. This target network is used by the DQN algorithm in the target term, which becomes instead: $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t^-)$. An additional mechanism to further improve the performance and stability of the DQN algorithm is experience replay. Experience replay saves previous experiences, in order to decouple learning from action taking [7]. In the learning process, a batch of experiences are sampled uniformly from the replay, and used for the iterative learning updates.

Since its publication in 2015, several modifications, extensions, and improvements have been proposed to the DQN algorithm, to further improve stability, learning rate, and overall performance. Among them are prioritized experience replay [14], dueling networks [15], noisy-networks [8] and double-DQN [16]. Prioritized experience replay attempts to utilize the observation that some experiences can be more valuable then others for learning. Dueling networks (Dueling DQN) suggests a different network architecture to the original DQN architecture. The architecture suggested consists of a network that learns two components

of the $Q(s,a)$ values separately: the value function of states $V(s)$ and the advantage function $A(s,a)$, defined as $A(s,a) = Q(s,a) - V(s)$. Noisy networks (NoisyNet) attempts to achieve better exploration by introducing parametric noise to the agent's network's weights, instead of using the common ϵ-greedy strategy.

Double DQN aims to reduce overoptimism that DQN is prone to due to estimation errors, by decoupling the selection of an action from its evaluation [16]. In vanilla DQN, in the term $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t^-)$, the agent uses the same network θ^- for both selecting and evaluating an action. This is illustrated by expanding the target term presented above to:

$$y_t = r_{t+1} + \gamma Q_t(s_{t+1}, \arg\max_a Q_t(s_{t+1}, a; \theta_t^-); \theta_t^-)$$

This term is equivalent to the original target term, but here the implicit use of two networks is apparent. The double-DQN algorithm proposes using the online network θ to choose the action, and the target network θ^- to evaluate the choice. The target term y_t used in the double DQN update then becomes:

$$y_t = r_{t+1} + \gamma Q_t(s_{t+1}, \arg\max_a Q_t(s_{t+1}, a; \theta_t); \theta_t^-)$$

This successfully achieves reduction of over-optimism of the Q-values [16], and is a commonly used modification in implementations of DQN.

2.3 Exploration in reinforcement and deep reinforcement learning

In order to find an optimal policy through experience alone, which is the general premise of RL, the agent must encounter the rewards that are part of an optimal policy at least once. This leads directly to a necessity to explore the environment - if the agent does not explore, how will it encounter valuable rewards that do not lie over its existing policy's path? However, the agent is also expected to efficiently converge into an optimal policy, and not only explore its environment. This leads to one of the fundamental principles of RL - exploration vs exploitation. Different approaches have been developed - ranging from dithering, random action choosing exploration [7], to more evolved notions such as deep and directed exploration [10], [11], and more. These approaches each attempt to achieve efficient exploration through different means - from simplicity of computation to effective analysis of the agent's knowledge and uncertainty.

Dithering exploration

The common baseline exploration strategy used in DQN is a dithering exploration method, or ϵ-greedy. In ϵ-greedy, the agent takes a random action (i.e., explores) with probability ϵ, and with probability $1 - \epsilon$ the agent takes the best action according to its current Q value estimation. ϵ-greedy achieves state of the art performance against many popular benchmarks [10], [16]. However, as discussed in [10], in environments where rewards are scarce and far away from each other in the state-action space, and their values have a large spread, the dithering exploration of ϵ-greedy can take an exponentially long time to arrive at high-valued rewards. This raises the necessity for a more advanced type of exploration, that can be directed over over multiple time steps. These concepts have been coined 'directed exploration' and 'deep exploration' [10].

Deep & directed exploration

The concept of directed exploration attempts to improve the efficiency of the agent's exploration by directing it. For example to previously unexplored or under-explored areas of the state-action space. To achieve a measure of directed exploration, an uncertainty measure can be used - the more uncertain the agent about the value of some state or action, the more it can prioritize exploration. However, in many environments, it does not suffice for exploration to be directed, it must be directed over multiple time steps, and thus termed 'deep exploration'.

As an illustration of the importance of deep exploration, consider the problem presented in figure 1, used by [10] for this purpose. In this RL problem, the agent starts each episode in state s_2, and can step one step left or one step right at each timestep. Each episode lasts $N + 9$ steps. Upon stepping left from s_2, the agent encounters a reward of $\frac{1}{N}$, N being the number of cells in the environment. In position s_N there is a reward of size 1. All other actions reward zero. An agent using ϵ-greedy is expected to identify the reward in position s_1 very quickly, and require an exponential number of steps to identify the much more valuable reward at position s_N, as is illustrated empirically in [10]. With deep exploration, the agent can direct its exploration towards the yet-unexplored areas of the environment, thus finding the reward at position s_N, and overcoming the problem.

Figure 1: A toy, chain environment used by [10] to illustrate the importance of deep exploration.

3 Exploration In Deep Reinforcement Learning for Traffic Light Control

This section will first motivate the choice to evaluate exploration techniques that focus on deep and directed exploration. Following that, the agent used in the experiments is outlined, and the exploration approaches it implements are described. Last, the modelling of traffic light control as an RL problem used in this paper is presented.

3.1 Motivation

This paper opts to specifically identify the value of exploration approaches that focus on achieving deep exploration, in the setting of traffic light control. These approaches have been chosen not only for being state of the art in this field, but also for their potential in this setting. In heavy traffic scenarios, suboptimal actions may carry a long term effect. They may immediately cause congestion, and once there, it may be difficult to return to less congested states [2]. Deep exploration may be able to improve the agent's ability to escape such scenarios, by directing its exploration along a specific path. While this path will not necessarily pay in the short run, it may allow the agent to recover from congestion in the

longer run. Additionally, the ability of the agent to more efficiently explore areas of the state action space that lie beyond areas plagued by negative rewards, as a result of employing deep exploration, may allow the agent to learn optimal policies that will otherwise be unlikely for a dithering agent to ever achieve.

3.2 The agent

The agent used in the experiments presented in this paper is a DQN agent, using the double-DQN [16] modification. This is the only modification implemented from the ones listen in section 2.2, for two related reasons. First, it is frequently used in state of the art implementations of DQN, because of the simplicity of its implementation and performance gains it provides. Second, the differences from regular DQN are not significant, which allow the conclusions drawn in this paper to relate more directly to a general DQN agent.

The agent implements the following exploration mechanisms: ϵ-greedy, Bootstrapped DQN (BDQN) [10] and randomized prior functions [11]. The implementation used in this paper, based on [17] and modified for the setting of traffic light control, allows the agent to use any combination of the three different mechanisms listed above. ϵ-greedy has been described in section 2.3, and has been chosen as it is the common baseline exploration used in DQN [7], [16]. BDQN and randomized prior functions have been chosen as state of the art exploration approaches that aim to achieve efficient deep exploration, and in addition, for their ability to elegantly combine for even better exploration. BDQN and randomized prior functions are described below.

Bootstrapped DQN

BDQN has been developed in an attempt to achieve a measure of deep exploration, discussed in section 2.3. In order to achieve deep exploration, BDQN approximates a distribution over Q-values, using a bootstrap. Bootstrapping is a technique used to approximate a population distribution from a sample distribution, using random sampling with replacement [18].

The bootstrap is implemented efficiently by using a shared neural network with several heads. The shared network's role is to learn a feature representation, while each head is providing an independent Q-values estimation. A visualization can be found in figure 2. In learning, the algorithm randomly samples an estimator (a 'head') out of the bootstrap, and follows the policy which is optimal for that estimator for some number of steps greedily or ϵ-greedily. In the experiments done in this paper ϵ-greedy is used. The resulting experiences are gathered in a buffer, and are available for all estimators to learn from, under some probability that decides which experiences will be available to which estimator. Each estimator is trained against its own target network / target network head.

In evaluation, an ensemble voting policy is used to evaluate which action has been chosen by most heads. If there is no majority vote, an arbitrary choice is made between the actions chosen by the most heads. The action is then chosen and executed.

BDQN attempts to achieve a measure of deep exploration by following the policy of one of the estimators for some number of steps. For this to be effective, the agent must guarantee that in areas of uncertainty (under-explored areas of the state action space), the different estimators will have different estimations. However, in BDQN this uncertainty, or variety in the Q-value estimations, is only based on the observed data [11]. This can be problematic, because in environments where rewards are very scarce, the agent may learn to believe that there is no reward, and lose all uncertainty, rather than direct its exploration to remote, unexplored areas of the state action space in the hope that they may contain rewards. Such

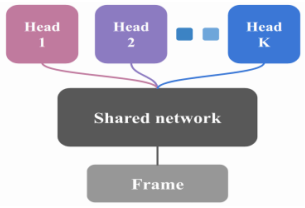


Figure 2: The BDQN architecture proposed in [10].

'prior' drive for exploration, that is independent from the data, is proposed in [11] in the form of randomized prior functions.

Randomized prior functions

While usable with a regular DQN agent, the randomized prior functions algorithm is designed to be combined with the BDQN model. To achieve independent uncertainty, the randomized prior functions model consists of one additional neural network for each Q-value estimator, or one shared neural network with one head for each estimator. This additional network or head p is combined with the original estimator f to form the final output Q, through a scaling factor β: $Q = f + \beta p$ [11]. Q is then used in the learning process to minimize the training loss. This results in uncertainty that is independent from the data: the Q value estimation always includes a neural network initialized with random parameters. No matter the uniformity of experiences the agent encounters (for example, only similar, negative rewards), it will still consider some additional prior 'assumption', in the form of the prior function, in regard to previously un-encountered states. As a result, each estimator will always approximate the Q value of as yet un-encountered states differently. This allows the agent to better direct its exploration, by guaranteeing diversity between the estimations of the different bootstrap heads for previously un-encountered states.

3.3 The model

The modeling of the traffic light control problem as an RL problem used in this paper is follows work done in [2]. The problem is modelled as an MDP $M = (S, A, P, R)$, where S is the state space, A is the action space, P the transition function and R the reward function. The open source traffic simulator SUMO [19] is used to generate the environment.

State space, action space & transition function

The state provided to the agent is represented as a set of stacked frames of size $x \in \mathbb{Z}^+$. Each frame is a matrix containing current locations of vehicles in the agent's observation space, and the current traffic light configuration. The observation space of the agent is a square centered at the intersection controlled. Each location of a vehicle is marked with a 1, and empty locations with a 0. The traffic lights configuration is presented in the matrix as numbers between 0 and 1 chosen arbitrarily. Stacking several frames allows the agent to extrapolate vehicle speed from the state representation. $x = 4$ was used in the experiments, to achieve a balance between too much information (complicating the learning process), and too little information (hindering the capacity of the agent to learn effective policies).

The actions available to the agent at each time step are one of two traffic light configurations, representing which lanes receive a green light. The transition function is defined by SUMO.

Reward function

The reward function used is a modified version of the reward function developed in [2]. At each time-step t, the agent receives a reward r_t, computed by iterating over all vehicles currently in the agent's observation space, and summing different penalties:

$$r_t = -1.5c - 0.2\sum_{i=1}^{N} e_i - 0.3\sum_{i=1}^{N} d_i - 0.3\sum_{i=1}^{N} w_i$$

This, where i represents the vehicle index. c is a penalty for switching the light configuration, to prevent flickering. e_i is a penalty for sharp decelerations, to penalize emergency stops. d_i is a penalty for the 'delay' of a vehicle, defined as $1 - \frac{vehicle\ speed}{allowed\ speed}$. Finally, w_i is a waiting penalty, defined as 0.5 for the first step of a car standing still, and 1 for any consecutive step.

The modification included removal of a term that punishes teleportation of vehicles (used in SUMO to mark traffic collisions) due to implementation challenges. Additionally, the coefficient of the term c was increased from 0.1 to 1.5, after observation that otherwise the penalty for light switching is barely noticeable with almost any number of cars.

4 Research Methodology

This section first describes and motivates the methodology used to evaluate the exploration approaches in the setting of traffic light control. This is followed by a description of the three traffic scenarios used to evaluate the exploration approaches.

4.1 The method

To evaluate the impact of exploration in the setting of traffic light control, this paper compares the performance of agents using ϵ-greedy, BDQN [10], randomized prior functions [11] and a combination of the above for exploration, in three different scenarios. The performance is evaluated using three different metrics, by averaging the results achieved by the agents over several of experiments.

The compared agents

As all three exploration approaches investigated are designed to be combined, the performance of the following six agents is compared: a regular DQN agent ad regular DQN agent with a randomized prior function, both employing ϵ-greedy; Two BDQN agents with increasing bootstrap size: 4 & 10 bootstrap samples (or neural network 'heads'); Last, two similar BDQN agents, combining randomized prior functions in their bootstrap mechanism. The number of bootstrap samples has been chosen based on a relation between computational complexity (the larger the bootstrap, the larger the complexity), and gain from the bootstrap (the larger the bootstrap, the better the average performance). As shown empirically in [10], the relative gain from sizes larger than 10 becomes insignificant very quickly.

The experimentation with different combinations of those techniques allows to evaluate, in essence, even more exploration techniques, and is the reason it is done in this paper.

As an additional baseline, two more 'agents' are included in each experiment conforming to the above experimenting methodology (termed 'the baseline agents'). These are a random agent and a constant agent. The random agent chooses a random action each time step, while the constant agent chooses a different action (switches the traffic light configuration) every set number of time steps c. $c = 20$ was chosen for the experiments as a balance between too similar to the random agent and too infrequent to have a sufficient measure of efficiency. These agents are included in the results presented as 'sanity checks', to illustrate both the overall quality of the policies learned by the agents, and the type of policies that score well in each environment.

The parameters of the agents investigated are not tuned for the specific setting of traffic light control. For the purpose of full reproducibility, a full list of the experiment parameters and agents' hyper-parameters used is available with the code base used in the experiments.

The evaluation

The evaluation is done as follows: an experiment is done, for each agent in each scenario and each intersection considered. Each experiment consists of N learning episodes. Every $evaluation\_frequency$ learning episodes, an evaluation phase is ran. In order to reduce sensitivity to stochastic noise from the random nature of the traffic used in the experiments, in the evaluation phase the agent's policy is evaluated over $number\_evaluations$ evaluation episodes, with $\epsilon = 0$. The average episodic reward is then used for evaluation. The exact parameters $evaluation\_frequency$, $number\_evaluations$ used in each experiment are detailed in section 5. To further reduce the impact of stochasticity, the entire experiment is repeated X times for each agent and the results averaged. Finally, the performances of the different agents are plotted against each other, in the form of their averaged evaluations' rewards. The results are presented in section 5.

The different parameters mentioned above were chosen in the following way: N was chosen from experimentation, as the range within which the agents' learning starts to plateau, in order to present the differences between the evaluations of the agents' in the clearest way. The $evaluation\_frequency$ used for each experiment set is chosen to achieve balance between the number of episodes each experiment is ran for, and the number of total evaluation episodes in the experiment. The $number\_evaluations$ parameter has been chosen as a balance between the total number of evaluation phases and the total number of episodes in each experiment. The larger the number of learning episodes, and lower the evaluation frequency, the larger the $number\_evaluations$ parameter. To balance computational costs and time constraints with reliability of results, each experiment was chosen to be averaged over $X = 10$ repetitions. This, except for the baseline agents which achieve a very similar performance every experiment. Their experiments repeat 5 times.

For the purpose of full reproducibility, all random generators used are fully seeded, and the seeds logged. Each experiment is initiated with a different random seed, to guarantee random initialization of the agents' neural networks' weights. The environment's traffic generator however is seeded with the same set of seeds for all experiments. This is done to guarantee that while all agents experimented with are different, they are tested against the same traffic simulations.

The metrics

When analyzing the comparison between the agents, this paper considers three fundamental metrics as measures for performance: the learning rate, the stability of the learning and the score of the policy achieved after certain final number of episodes. The main concerns for an RL agent are: how good is the solution it acquires for the problem, how long did it take it to arrive at that solution, how likely it is to converge to the solution and how stable the convergence. These concerns are covered by the metrics listed above: quality of solution by policy score, time to convergence by learning rate, and stability and likelihood to converge by stability of the learning. In the analysis of the results, these metrics are considered independently. However, when all three appear to coincide, they are simply referred to as the agent's 'performance'.

4.2 The traffic scenarios

The impact of the exploration approaches investigated in this paper is evaluated using the traffic simulator SUMO [19], in three different traffic scenarios. In the first scenario, one set of experiments is done. In the second and third scenarios, two separate experiment sets are done, evaluating the agent against traffic of slower and faster average speed.

Scenario 1: The grid

The first scenario is a basic grid-like road network, with one intersection in the center, and four roads going one in each direction from the intersection: north, east, south and west. A visualization of the grid scenario is presented in figure 3 a. The first scenario is meant to capture a simple, independent intersection profile, that does not consider or experience the behavior of other neighboring intersections. The traffic in this scenario is generated randomly, based on a set number of vehicles over a set spawning time.

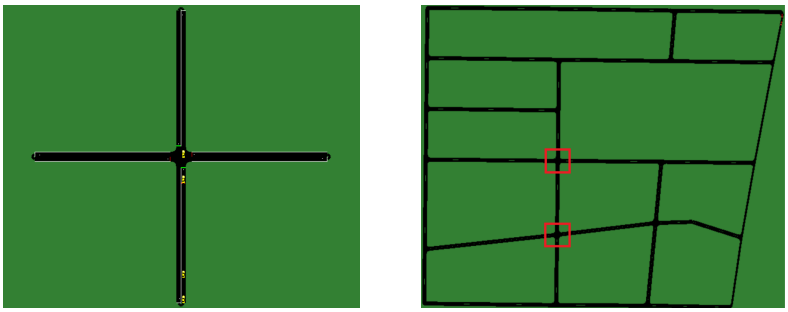
Scenario 2: Real traffic simulation in Manhattan, New York

The second scenario, visualized in figure 3 b, is based on a section of the road map of Manhattan, New York, and is a more complex network containing several interconnected intersections. The specific road map used in our experiments is a 700 m^2 section centered around the corner of Waring and Woodhull Avenues. The map has been imported using SUMO's web-wizard. This scenario means to evaluate the effect of efficient exploration in a more complex traffic profile, where the agent may observe and consider the behavior of neighboring intersections. Manhattan enjoys a grid road-map design, that for the purpose of this work serves as both realistic and practical to use.

The red squares marked in figure 3 point to the two intersections given to the agent to control, as two separate experiments sets in this scenario. These intersections were chosen due to their encapsulation of different traffic profiles. The top intersection, Waring-Woodhull, presents a gentler form of traffic with significantly lower vehicle speed average. The bottom intersection, Eastchester-Waring, experiences much higher average vehicle speed, and more strongly resembles a central road. The throughput of both intersections is rather similar, with a slightly heavier load going through Eastchester-Waring. In every experiment, all intersections in the network except the one controlled by the agent are controlled by SUMO. To generate traffic for the Manhattan scenario, a random routes generator is used, based on real population distributions in the area, for the time of day 08:30 AM to 09:00 AM. The traffic data is imported using SUMO's web-wizard as well.

Scenario 3: Custom traffic simulation in Manhattan, New York

A third scenario is introduced in order to evaluate exploration under heavier traffic settings. This scenario uses the same map, and experiments with the same intersections illustrated in figure 3 b. However, in this scenario a random traffic generator is used for traffic generation, in order to introduce much heavier traffic loads than the ones generated to simulate real traffic. Again, two different sets of experiments are done in this scenario, one on each of the two marked intersections in figure 3 b. The difference of the traffic profiles between the two intersections is similar to the one in the second scenario: the top intersection enjoys slower average speed, and the bottom faster.



(a) The basic grid scenario.

(b) The Manhattan scenario. The top red box marks the intersection Waring-Woodhull, and the bottom the intersection Eastchester-Waring.

Figure 3: The road maps used in the traffic scenarios.

Observation space specification

An important difference between the scenarios is the observation space provided to the agent in each one. The observation space provided in the experiments done with the grid scenario and the Eastchester-Waring intersection in either scenario, was $50\,\mathrm{m}^2$. This is done because in all of the three a significantly larger observation space would be outside the bounds of the environment. The observation space provided in the experiments done with the Waring-Woodhull intersection in either scenario was $84\,\mathrm{m}^2$. This is done to allow the agent access to more information, which (1) contains the adjacent intersections, enabling the agent to take their behavior into account, and (2) providing the agent with the ability to react to incoming traffic earlier, as a result of the larger observation space.

5 Results

This section presents and analyzes the results obtained in the experiments described in section 4, divided between the different scenarios investigated and intersections controlled. For each set of experiments, the results presented are the episodic rewards attained in the evaluations, as described in section 4.1. This is presented alongside a plot of the 95% confidence interval of the mean, computed using the standard error of the mean (SEM) [20] of the different experiments for two sample agents. These agents are chosen independently for each experiment: the one that performed, on average, the best, and the one that performed the worst. This is done to illustrate how significant are the differences observed between

the evaluations of the different agents in each experiment. Only two agents are presented in order to reduce clatter in the plot. A simple moving average (SMA) of window size 5 is applied to the data presented in order to smoothen the stochastic effect, to facilitate visual analysis of the results.

5.1 Scenario 1: The grid

The results of evaluating the performance of eight different agents, six learners and two baselines, against the grid scenario can be found in figure 4. The agents' policies are evaluated every five learning episodes, and averaged over three evaluation episodes. Additionally, the 95% confidence intervals of the means of the two sample agents are presented.

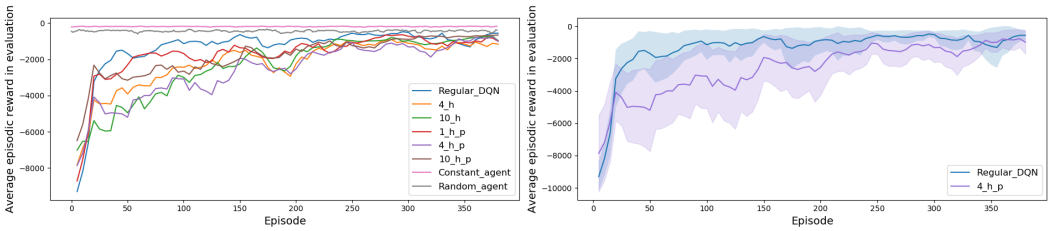


Figure 4: Evaluating the agents against the grid scenario. The left figure presents the evaluations. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same for two chosen agents, including the 95% confidence interval of the mean.

Figure 4 illustrates that while all agents learn, the agents that appear to have the sharpest learning rates and higher averaged evaluation scores are the regular DQN with and without prior function applied. However, as can be seen in the right plot in figure 4, there is overlap in the confidence intervals of the means. As a result, the differences observed cannot be considered significant.

A situation where the regular DQN performs better than the more evolved agents can be explained with the fact that the regular learns from all experiences attained in learning, while the larger bootstraps each learn from a different set of those experiences. While this keeps the different bootstrap heads varied, it can slow down learning of solution to simple problems.

Additionally, none of the agents appear to overcome either of the two baselines, in the number of learning episodes used in the experiment. This effect is attributed in part to the moving average, which while hiding the instability of the results, also hides the peaks, which achieve similar score to the random agent. These results can be found in figure 8a.

5.2 Scenario 2: Real traffic simulation in Manhattan, New York

The results of evaluating the agents against the Manhattan scenario simulating real traffic can be found below, separately for each set of experiments, controlling each of the two intersections marked in figure 3 b. The agents' policies are evaluated every training episode, over two evaluation episodes, and averaged.

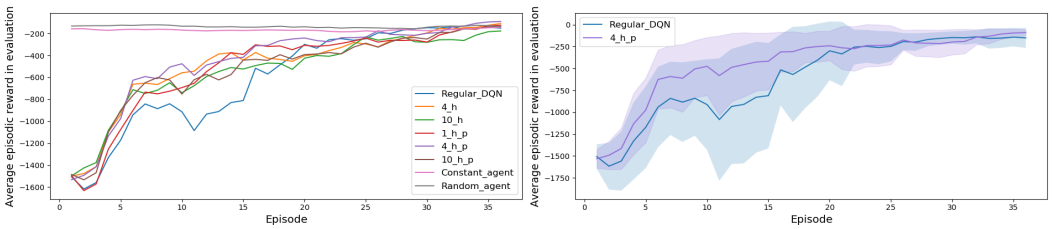


Figure 5: Evaluating the different agents against the Waring-Woodhull intersection, simulating real traffic. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

Traffic of low average speed

Figure 5 presents the results of experimenting control of the intersection Waring-Woodhull, the top of the two intersections in figure 3 b, capturing a traffic profile of slower average speed.

As observable, the difference between the evaluation scores in relation to the confidence interval of the means, is mostly negligible, with the exception that the regular DQN achieves inferior scores prior to episode 20. However, these scores are still well within the confidence interval of the means of the other agent's, and thus cannot be considered significant. This behavior is attributed to the simplicity of the traffic profile, in relation to the amount of information accessible to the agent in this scenario. As mentioned in section 4.2, the observation space of the agents in this experiment is $84\,\mathrm{m}^2$. While the scenario can be viewed as complex (several interconnected intersections, whose behaviors directly influence each other's traffic), which can translate to the learning process being more difficulty, the volume of the traffic, including the average speed, is rather low, and thus the policy required is not complex.

Traffic of high average speed

Figure 6 presents the results of experimenting control of the second intersection, Eastchester-Waring. Eastchester-Waring enjoys both traffic of higher average speed, as well as higher traffic loads. The main observation that can be made here is the mostly negligible difference in performance between the different agents, with the exception of the regular DQN employing a prior, between episodes 20 and 30.

A related observation is that the two agents with the largest average evaluation score difference, appear to be two of the arguably most similar agents - regular DQN and regular DQN incorporating a prior. Taking into account the confidence interval of their means presented in figure 6, the differences are attributed to stochastic behavior. A possible explanation of this observation is that due to the small observation space and high traffic speed, the agent cannot react to the traffic in time, which results in a problem that is not very complex, and thus all exploration approaches achieve similar scores over time.

5.3 Scenario 3: Custom traffic simulation in Manhattan, New York

The results of evaluating the agents against the Manhattan scenario simulating random, heavy traffic can be found below. The agents' policies are evaluated every second training

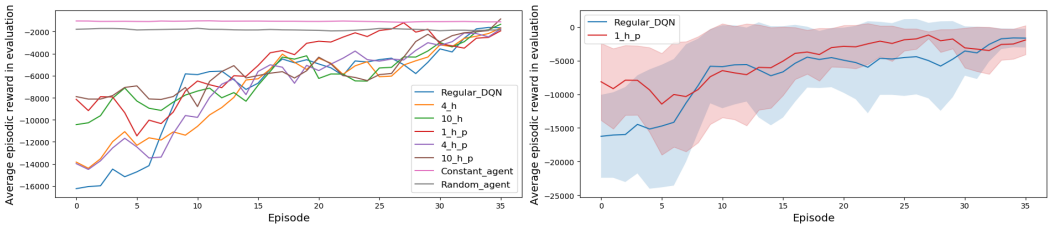


Figure 6: Evaluating the different agents against the Eastchester-Waring intersection, simulating real traffic. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

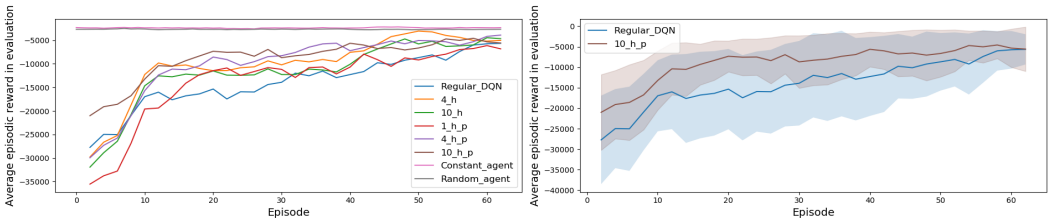


Figure 7: Evaluating the different agents against the Waring-Woodhull intersection in the custom scenario. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

episodes, and averaged over two evaluation episodes.

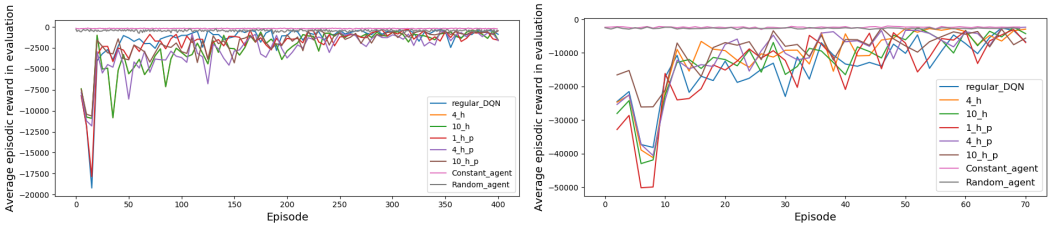
Traffic of low average speed

The results for the intersection Waring Woodhull are presented in figure 7. The variance between the performances of the different agents appears slightly higher than in previous experiments. In this setting specifically, the advanced agents (the agents incorporating a prior and / or a larger bootstrap) appear to achieve a higher score than both the regular DQN agent and the regular DQN incorporating a prior, throughout most of the evaluations, and in the order of their relative complexity.

In other words, the bootstraps of sizes 10 and 4 with priors achieve the highest evaluation scores throughout most of the experiment, while the bootstraps of size 10 and 4 without priors achieve very similar scores to each other, secondary to the above, throughout most of the experiment.

It is important to consider here the specific parameters of this experiment - traffic of slow average speed, that the agent can react to. Larger observation space ($84\,\mathrm{m}^2$) and traffic volumes, such that the agent must consider more information in its learning and evaluation. While the differences are not significant, and still within the confidence interval of the means, this may still imply that in scenarios that are (1) more complex, (2) the agent has a better chance to react to the changes in traffic, (3) the amount of information accessible to the agent is high (complicating the learning process), efficient, or specifically deep exploration approaches show advantage.

Last, the different agents do not appear to converge to policies that score as high as either of the baselines. This is attributed to the application of the moving average. Figure



(a) The grid scenario.

(b) The Waring-Woodhull intersection in the custom scenario.

Figure 8: Evaluating the different agents, without application of a moving average. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated.

8b presents the average episodic evaluation results without the moving average applied. As can be observed, the agent achieves evaluation scores that are similar to the baseline agents, but the policies have yet to stabilize.

Traffic of high average speed

The results of experimenting control over the intersection Eastchester Waring are presented in figure 9. There does not seem to be any significant difference between the evaluations of the different agents, especially considering the confidence interval of the means, presented in the right plot in figure 9. This is explained similarly to the findings of the experiment simulating real traffic with same intersection, presented above.

5.4 Analysis

Many different factors can influence the results obtained by the different agents, ranging from the reward function, the observation space, the state abstraction, the complexity of the traffic scenario to plain stochasticity. This in addition to the specific configuration of the hyper parameters. The type and complexity of the traffic scenario, along with the observation space, are the main parameters that have not been kept static between the experiments. For this reason, they are the main parameter considered. Here, the complexity of the scenario, while not weighted exactly, considers the volume and variety in the traffic and the number of surrounding interconnected intersections.

It appears that when the agent is given sufficient information of sufficient complexity, deep exploration approaches are able to outperform the ϵ-greedy approach, by achieving faster learning. Information complexity is considered as both the amount of information (size of the observation space) and traffic complexity (measuring both diversity and intensity). This implies the following: (1) it is important to consider efficient exploration approaches when applying RL to complex traffic scenarios, provided a sufficiently large observation space is used. (2) Optimization of hyper-parameters, including choosing a fitting reward function and state abstraction, are essential parameters to the relevance of a certain exploration method. However, it is important to mention that the differences observed were generally small and within a 95% confidence interval of the means, and as such, the strength of this implication is limited.

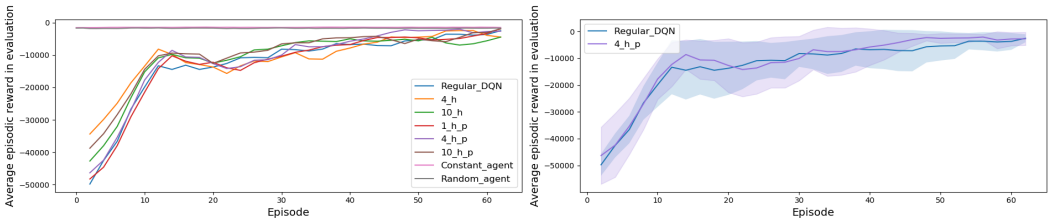


Figure 9: Evaluating the different agents against the Eastchester-Waring intersection in the custom scenario. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

6 Responsible Research

This section discusses the ethical and epistemic concerns of the research presented in this paper, and its possible implications. The discussion is divided between the two concerns.

In terms of ethical concerns, the experiments done in the work presented were all virtual simulations, based on data that is fully anonymized. The only information considered about the vehicles is their location and speed. Additionally, conclusions made based on the work done in this paper are not expected to directly influence real-life application of RL to traffic light control. As such, the authors of this paper do not believe there are any concerns of safety or privacy, or any other significant ethical concerns warranting specific consideration.

In terms of epistemic concerns, reproducibility, data credibility, and cherry-picking are considered as the main items of discussion. In regard to reproducibility, the code used to generate and run the experiments is accessible upon request from TUDelft, and is based on algorithms presented in papers that are publicly accessible [10] [11], and an implementation from a publicly accessible repository [17]. All the parameters used in the experiments are logged and accessible with the codebase, and all the experiments are seeded based on randomized and logged seeds and should therefore be fully reproducible.

In regard to data credibility, the simulation of real traffic was generated in the following way: population statistics were collected from sources such as the United States Census Bureau, the New York State Legislature, and the New York City Department of Transportation, publicly accessible at *data.census.gov*. These were used to generate a traffic profile that is realistic as the statistics allow.

In regard to cherry picking, the experiments presented are a representative sample of all experiments done as part of this research. None of the results of the other experiments done imply conclusions conflicting with those drawn in this paper.

7 Discussion

The results obtained suggest a link between the complexity of the scenario and the information accessible to the agent, and the gain from deep exploration, as discussed in section 5.4. However, the advanced exploration approaches investigated are expected to have significant gain especially when utilized in environments with scarcity of significant positive rewards, and abundance of smaller negative rewards [10], [11]. This implies that the reward function can directly dictate the results - reward shaping that successfully leads the agent

progressively towards optimal policies may completely negate the need for effective deep exploration, while a harsher reward function may require deep exploration for any real chance at success. The reward function used was not designed for any specific exploration approach. It is therefore very plausible that under a tailored reward function the gain from deep exploration would have been much higher. Future work could therefor investigate for performance difference between different exploration approaches under different reward functions in this setting. This can be especially relevant in the case of reward functions that are known for promoting good traffic light control policies, while hindering learning.

Additionally, the size and complexity of the parameter space is important to consider. Other parameter configurations than the one used might have produced different results. Future work could attempt to optimize these parameters for each of the different agents or approaches studied, and measure performance differences under optimized parameters.

8 Conclusions and Future Work

This paper investigated the value of efficient, and specifically deep exploration in the setting of traffic light control, by comparing agents using different exploration approaches in three different traffic scenarios of rising complexity. Specifically, the state of the art approaches Bootstrapped DQN [10] and randomized prior functions [11] were compared to a baseline ϵ-greedy. This, to facilitate better deep RL in traffic light control, by identifying the value of evolved exploration approaches in this setting, such as higher sample efficiency or higher final policy score.

The results presented in this paper suggest a link between the complexity of the traffic scenario, the size of the observation space of the agent, and the gain from efficient exploration, achieved with Bootstrapped DQN and randomized prior functions, under a specific parameters configuration. Specifically, the more complex the scenario, and the larger the observation space, the larger the gain observed from efficient exploration.

The results presented leave the following open questions, however. What is the exact relation between the gain from efficient exploration and the complexity of the scenario? How sensitive is this relation to specific parameter configurations, and specifically under parameters optimized for the specific setting? Does this conclusion apply for other exploration approaches? These questions are left for future work.

Acknowledgements

We would like to thank the students in the research-project group, Pepijn Tersmette, Cian Jansen, Emanuel Kuhn & Chris van der Werf, for their ready assistance and availability for discussion, along with their contributions to the creation of the traffic scenarios and the development of the reward function used. Additionally, we would like to thank the TUDelft ILDM research group for their helpful comments and support. An additional thanks goes to the TUDelft for granting us access to the INSY cluster, which was a crucial computational resource used in this research.

References

[1] H. R. TO and M. M. Barker, "White paper european transport policy for 2010: time to decide," 2001.

[2] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[3] M. Rezzai, W. Dachry, F. Mouataouakkil, and H. Medromi, "Reinforcement learning for traffic control system: Study of exploration methods using q-learning," *International Research Journal of Engineering and Technology*, vol. 4, no. 10, pp. 1838–1848, 2017.

[4] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Information Systems*. Springer, 2010, pp. 475–510.

[5] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[6] M. Coşkun, A. Baggag, and S. Chawla, "Deep reinforcement learning for traffic light optimization," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 564–571.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[9] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann, "Better exploration with optimistic actor critic," in *Advances in Neural Information Processing Systems*, 2019, pp. 1785–1796.

[10] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Advances in neural information processing systems*, 2016, pp. 4026–4034.

[11] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 8617–8629.

[12] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[13] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560*, 2018.

[14] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[15] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1995–2003.

[16] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[17] J. Hansen, "bootstrap_dqn," *GitHub repository*, 2019.

[18] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[19] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[20] M. P. Barde and P. J. Barde, "What to use to express the variability of data: Standard deviation or standard error of mean?" *Perspectives in clinical research*, vol. 3, no. 3, p. 113, 2012.