Applying Reward Shaping and Optimal Reward Search to a Novel Grid-World Setting

Ishank Juneja - 16D070012 Rahul Chanduka - 160070003
Gopinath Balamurugan - 160010053 Anirudh Arputham - 160010056

1. Introduction

Making learning tractable in a reasonable computation time is one of the challenges of Reinforcement Learning. A good learning agent is characterised by its ability to learn a near optimal policy without extensive exploration. However sparse rewards from the external environment make learning a challenge and careful design of additional shaping rewards becomes a practical necessity.

In our project we explore two methods of designing the reward schemes of learning agents that are present in the literature, namely - Optimal Reward Search and Reward Shaping. We apply these two approaches to hasten the learning of a near optimal policy by our agent in a novel Grid World Setting called the **Balanced Diet Problem**.

Through experimentation described in our report we find that it is indeed possible to design additive and potential like Shaping Reward functions that match the performance of an Optimal Reward Search.

2. Related Work

While designing an intelligent agent, an agent designer has certain expectations from the agent's behaviour. These broader design objectives can be modeled as an extrinsic reward function. The qualifier extrinsic is appropriate since these rewards are obtained from the agents environment. For a majority of tasks, this extrinsic reward function is too sparse for the learning of an optimal policy to be tractable in a reasonable number of iterations. To overcome this problem, multiple solutions have been proposed in the literature. Two of these solutions are Optimal Reward Search (Singh et al., 2009) and Policy Invariant Reward Shaping (Ng et al., 1999).

The Optimal Reward Search framework performs a search for an optimal intrinsic reward function over the space of all possible reward functions. Here the optimal reward function, as defined in (Singh et al., 2009) is the one that maximizes the expected extrinsic reward thereby complying with the objectives of the designer.

Reward Shaping, on the other hand, is a method in which a shaping function is added to the sparse extrinsic reward function and the new reward scheme is used to train the learning agent.

We employ these frameworks for training an agent to maximize its fitness (a kind of extrinsic reward) in the Balanced diet setup described in detail under section 3

3. The Balanced Diet Problem

The balanced diet setup consists of a grid-world of dimensions 6 x 6 with two kinds of food sources - Fat and Protein - available for consumption by an agent residing in it. The action space of the agent consists of five possible actions lying in $\mathcal{A} = \{$up, down, right, left, eat$\}$. The first four actions make the agent move in the specified direction whenever possible and remain in place when blocked by a boundary. The eat action allows the agent to consume food when it is on a food source square.

By consuming any of the two food groups, the agent moves into a state where it is satiated with respect to that food group. When the agent is satiated with respect to both the food groups, we say that its diet is **balanced**.

Under the problem setup, the sparse extrinsic reward for the agent is an incremental fitness. The agents environment confers much larger fitness levels for a balanced diet as compared to an unbalanced diet consisting of a single food group.

Quantitatively speaking, for every time step that the agent is satiated in a single food group, its fitness level is incremented by 0.01 and for every step when it is satiated in both food groups, its fitness level is incremented by 1.0. In case the agent is not satiated in either of the food groups, its cumulative fitness level remains unchanged.

After entering a satiated state in a certain food group, the agent becomes un-satiated with respect to the food group with probability 0.1 in every subsequent time step. This transition happens independent of its state with respect to the other food group and aslo independent of it's position.

The task is modelled as a continuing task with a discount factor of $\gamma = 0.99$. Across all runs, food sources in the grid are located at diagonally opposite corners and the agent is always born in the 6^{th} row and 6^{th} column. See figure 1 for a visualization.

Given the parameters specific to this setup, to maximize

its cumulative fitness, the optimal policy for the agent is to alternate between the fat and protein food sources. To make the agent learn such a policy, we use Q-learning coupled with ϵ-greedy exploration as our control algorithm of choice.

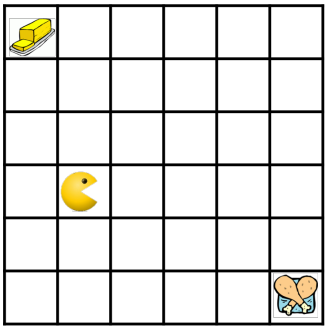

Figure 1. The Balanced Diet Problem

4. Reward scheme strategies for learning

4.1. Optimal Reward Search

As described briefly in section 1, reward search performs an exhaustive search over a reward space under some reasonable constraints on the reward function. The **optimal reward scheme** is said to be the scheme that leads to the highest expected cumulative fitness (extrinsic reward) for a fixed finite horizon, when the agent learns under it.

We define the hunger state of the agent to be $s = 00$ if it is un-satiated in both food groups, $s = 01/s = 10$ if it satiated in only fat/protein and $s = 11$ if it is satiated in both food groups. In our experiments we only consider reward functions which reward the agent based solely on the hunger state it transitions to.

Hence on taking any action, if the agent moves to a Hunger state s, then the agent can receive one of four possible rewards given by

$$R(s) = \begin{cases} r_{00} & \text{if } s = 00 \\ r_{01} = r_{10} & \text{if } s = 01 \text{ or } s = 10 \\ r_{11} & \text{if } s = 11 \end{cases}$$

To compare the results of learning under the optimal reward scheme learned using reward search, we use the incremental fitness levels as the baseline reward function. That is, we use,

$$R_{base}(s) = \begin{cases} 0 & \text{if } s = 00 \\ 0.01 & \text{if } s = 01 \text{ or } s = 10 \\ 1.0 & \text{if } s = 11 \end{cases}$$

Under our scheme the reward space is three dimensional, so to search for the optimal reward scheme we must discretize the parameters $\{r_{00}, r_{01}, r_{11}\}$ in a range of choice which we take to be $[-1, 1]$.

Precisely, we discretize r_{01} over the range $[-1, 1]$ with a step size of 0.2. However for r_{11} and r_{00} we use slightly modified schemes for efficiency.

For r_{11} we use the range $[0, 1]$ since it is easy to see that a negative reward for consuming a balanced diet is guaranteed to be sub-optimal.

For r_{00} we use the search space,

$$r_{00} \in \{-1, -0.5, -0.25, 0, 0.01, 0.02\}$$

This is a coarser and in-exhaustive search space. The coarseness of the search space is justified since, a low satiatedness decay probability of 0.1 means that the number of time steps spent in $s = 00$ are quite few. The maximum value of 0.02 is chosen based on the intuition that rewarding a completely un-satiated state by a positive reward larger than 0.02 is likely to be sub-optimal.

In addition to the above discretization, we also simulate the results for the baseline $R_{base}(s)$.

Results from Reward Search

Based on the above choice of discretization, the size of the search space $\mathcal{R}$ becomes $|\mathcal{R}| = 330$. Within $\mathcal{R}$ we define a subclass which we call Fitness Based Rewards, a term taken from (Singh et al., 2009). Fitness Based Rewards are further constrained in that they can only assign non-negative rewards to events that increment fitness and zero rewards to all other events. We would like to point out that the baseline reward function $R_{base}(s)$ also lies in this subclass.

The cumulative fitness of the agent for a horizon of $75,000$ time steps was calculated for each reward scheme $R(s) \in \mathcal{R}$ after averaging over a total of 10 runs. These results have been plotted in figure 2.

From figure 2 we see that the optimal reward function $R^*(s)$ performs significantly better than the baseline reward scheme in terms of cumulative fitness. An interesting observation from figure 8 is that the Best-Fitness based reward seems to coincide with the Baseline reward scheme, the reasons for this are discussed later in this section. The optimal reward scheme in $\mathcal{R}$ is given by,

$$R^*(s) = \begin{cases} -0.25 & \text{if } s = 00 \\ -0.2 & \text{if } s = 01 \text{ or } s = 10 \\ 1.0 & \text{if } s = 11 \end{cases}$$

Characteristics of a Good Reward Scheme

Being satiated in a single food group is an event that increments fitness, despite this, the optimal reward function $R^*(s)$ reveals that associating a negative reward with an

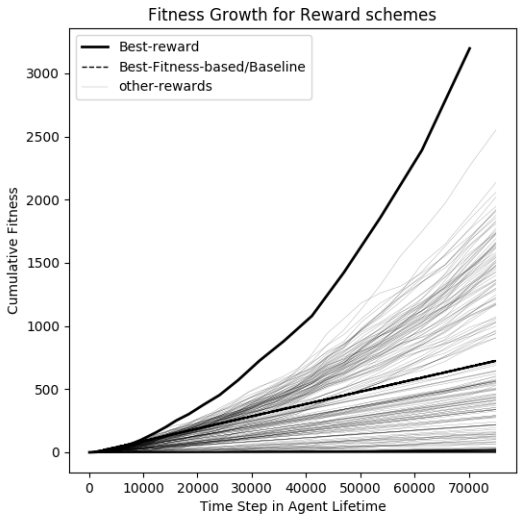

Figure 2. Comparison of the learning rate across reward schemes

unbalanced diet is desirable. The reason for this is explored further through figure 3

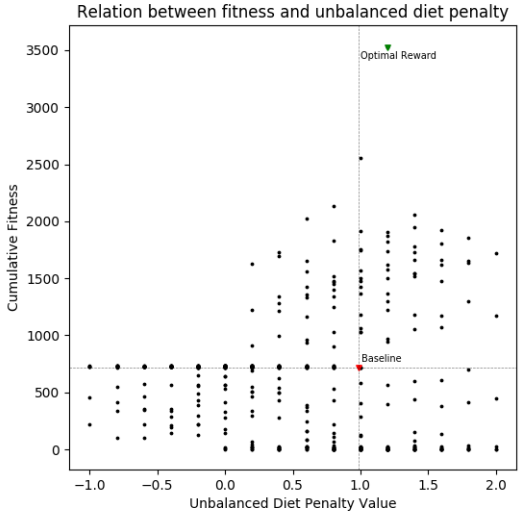

Figure 3. Reward schemes and their Relative Penalties

In figure 3, we plot **relative penalty** on the horizontal axis. Here relative penalty is defined as $r_{11} - r_{01}$, that is the difference in the reward for transitioning to a balanced state vs. consuming a single food group. From the figure 3 we see that reward schemes which have a large positive relative penalty tend to perform better than other schemes.

Under fitness based rewards, we assign only non-negative rewards to fitness increasing events. Although, a fitness based scheme seems like a natural choice for a good reward scheme, it means that the maximum relative penalty can be $r_{11} - r_{01} = 1.0 - 0.0 = 1.0$, However, we must be able to move to larger relative penalty values if we wish to improve cumulative fitness as can be seen from figure 3 and the fact that the optimal reward scheme has a relative penalty of 1.2. So, under constraints imposed on reward schemes $R(s) \in \mathcal{R}$, the notion of a good reward scheme can be understood as one which penalises the agent for consuming an unbalanced diet up to a point. Beyond this, a greater relative penalty begins to harm the agent. This happens since the agent learns to put too much emphasis on not being in hunger states 01 and 10 which in turn jeopardizes its ability to transition to hunger state 11.

Lastly it is interesting to note that the optimal reward scheme makes the agent spend a much greater fraction of time in a hungry state of 00 then the baseline. This is seen through figure 4 that plots the fraction of time the agent is un-satiated on both food groups vs. the reward r_{01}.

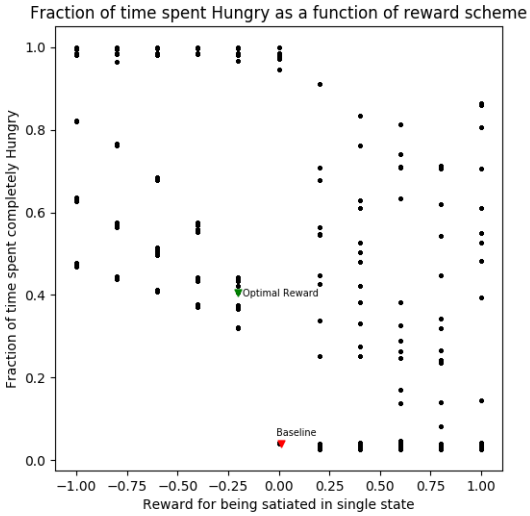

Figure 4. Fraction of time spent completely hungry

POLICIES LEARNED

Here we visually present the policy being followed by the agent at the end of a particular run. We plot the best action found using Q-learning in 3 different hunger states of 00, 01 and 11. The case of 10 is analogous to that of 01. For the state 00, we can see that the optimal action tends to take it to the nearest food source, which is what one should expect it to learn. There are some obvious errors in some cells, and we attribute this to insufficient exploration in those states (since the optimal policy does not include those states). Also note that the agent eats at both the protein source's location and that of the fat source, as it should.

For the state 01, we see that the agent at most points is trying to go towards the fat source, as it should. The mistakes are again due to insufficient exploration along non-optimal paths. Notice that it doesn't eat at the protein source, which makes sense since it gains nothing by eating protein if it is already satiated in protein.
For the state 11, we see that there is no obvious pattern in what the agent does. It tries to seek out a balance between staying close to both food sources(in the middle of the grid) and close to one food source in case it loses that first.

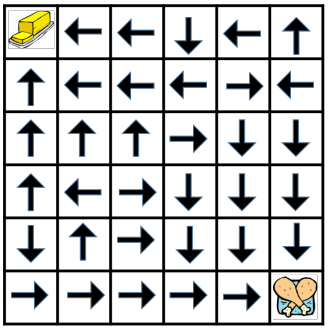

Figure 5. Policy being followed when $s = 00$

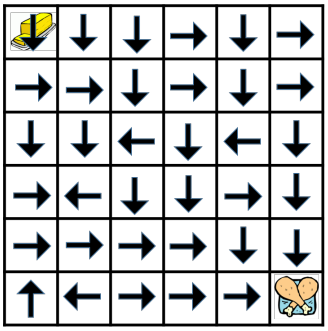

Figure 6. Policy being followed when satiated only in fat: $s = 01$

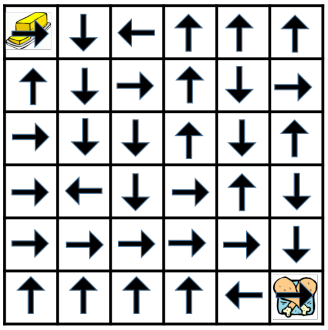

Figure 7. Policy being followed when $s = 11$

4.2. Reward Shaping

The second part of our project was based on the approach given by (Ng et al., 1999). In this paper the authors introduce the notion of potential functions for reward shaping, which are rewards for transition between states that are expressible as the difference of some arbitrary potential function applied to those states.
The basic idea is that for the MDP M = (S,A,T,R,γ), we "shape" the reward functions in a way that guides the learning algorithm to an optimal policy faster. Formally, we tweak R to R' where $R' = R + F$ and run our learning algorithm on this modified MDP. Here, $F : S \times A \times S \to \mathbb{R}$ is a bounded real valued function called the **shaping function**. Using expert knowledge of the environment, we can design an F that enables faster learning.
The essence of this paper is in the type of shaping functions they introduce. A shaping function F is called a **potential-based** shaping function if there exists a function $\phi : S \to \mathbb{R}$ such that for all non-terminal states $s \in S$, $a \in A$, $s' \in S$,

$$F(s, a, s') = \gamma\phi(s') - \phi(s)$$

In (Ng et al., 1999), the authors show that the above is a necessary and sufficient condition for **policy invariance**, meaning that any other family of shaping functions can lead to sub-optimal policies in the original MDP becoming optimal under the modified MDP. Further, it also means that for a control algorithm such as Q-learning any potential based shaping will converge to a policy that is optimal under both search and shaping.
The theorem advocates that we look for potential functions of the form $F(s, a, s') = \gamma\phi(s') - \phi(s)$ using expert knowledge of the environment of the MDP. One example would be to reward the agent for going to "good" states, and penalising it for going to "bad" states. If this is to be modelled, the $\phi(s)$ can be kept positive for the good states and negative for the bad ones.
Another interesting idea derived from the paper is using $\phi(s) = V_M * (s)$, that is keeping the values equal to an estimate of the optimal value functions. This estimate is based on our understanding of the expected return from an MDP starting from a certain state. We try both these approaches on the Balanced Diet problem and have compiled the specifics and the results in this section.

VALUE FUNCTION ESTIMATION

Our first attempt was to guess a potential function based on a crude estimate of the value function of each state. Rather than guessing the value function, we tried to guess another quantity, namely the expected number of steps needed to reach full satiation. Since a higher expected number steps suggests a lower value function, we use the negative of this quantity as a substitute for the value function, and use a

potential based reward scheme based on this.
Since this quantity is hard to compute in closed form, we further simplified it to computing a quantity that we call 'steps to satiation assuming no decay'. For each state (grid-position, hunger state), we define the quantities d_1 =Manhattan distance of protein source from current position and d_2 =Manhattan distance of fat source from current position. We further define d_{src} =Manhattan distance between the two food sources. Let 0 correspond to the agent being hungry in both sources, 1 to satiated in fat only, 2 to satiation in protein only and 3 correspond to full satiation. We define ϕ as:

$$\phi(\text{pos}, 0) = -\min(d_1, d_2) - \frac{d_{src}}{\mathbb{P}(\text{No decay in } \mathrm{d_{src}} \text{ steps})}$$
$$\phi(\text{pos}, 1) = -\frac{d_1}{\mathbb{P}(\text{No decay in } \mathrm{d_1} \text{ steps})}$$
$$\phi(\text{pos}, 2) = -\frac{d_2}{\mathbb{P}(\text{No decay in } \mathrm{d_2} \text{ steps})}$$
$$\phi(\text{pos}, 3) = 0$$

Notion of Good States

Our second attempt was to categorise the states of the MDP as good or bad based on their position on the grid, irrespective of the satiation state in the positions. By doing this, we wanted to encourage the agent to take a particular path on the grid, and thus converge to an optimal policy faster. This was achieved by encouraging the agent to move along the diagonal so that it does not need to wander around and can learn an optimal with limited exploration. The potential function we used for this is as follows (same for all hunger states):

$$\phi_{diag}(\text{x,y}, a) = \begin{cases} r, & \text{if } x + y \in [m_1, m_2] \; \forall a \in \{0, 1, 2, 3\} \\ 0, & \text{otherwise} \end{cases}$$

By constraining x+y between two values, we make sure that the agent learns to stay in some diagonal patch along the diagonal, which indeed corresponds to an optimal path from a balanced diet viewpoint.
By using this shaping function, we narrow down the agents search to one of the optimal paths, rather than learning several optimal paths and converging to one among them via random events. For this setup, we tried various values of m_1, m_2 and r and their differences are highlighted in the graphs below.
Another idea we had along similar lines was forcing the agent to stick to a couple of edges of the grid since we aren't allowing diagonal movements. Similar to the above scheme, this also narrows down the optimal paths available to the agent and learning should improve with this potential function as well. Formally, we set:

$$\phi_{edge}(\text{x,y}, a) = \begin{cases} r, & \text{if } x = 0 \; \forall a \in \{0, 1, 2, 3\} \\ r, & \text{if } y = 0 \; \forall a \in \{0, 1, 2, 3\} \\ 0, & \text{otherwise} \end{cases}$$

The results for all of these as well as the value function estimation approach are depicted below.

Shaping Results

We have simulated 4 different shaping methodologies based on the ideas discussed above. For each of these, the cumulative fitness of the agent was computed over a horizon of 125,000 time steps. The graphs have been averaged across 10 such random runs.

- **Baseline:** The learning curve with no shaping, only the rewards given by the environment.
- **Value Based Potential:** Here, we have used the potential function (based on both the position and state) as described in section 4.2.1
- **Broad Diagonal Potential:** Here, we have used the function ϕ_{diag} with $m_1 = 4$, $m_2 = 6$ and $r = 10$. We have scaled the rewards (while learning) appropriately to keep the range of rewards that the agent receives same in all cases.
- **Thin Diagonal Potential:** This uses the function ϕ_{diag} with $m_1 = 5$, $m_2 = 6$ and $r = 10$. Again, scaling has been done appropriately.
- **Edge Potential:** The function ϕ_{edge} has been used here with $r = 10$. All rewards have been scaled.
- **Optimal Reward Search:** This refers to the best reward search scheme from section 4.1, included to enable learning comparison between shaping and searching.

As can be seen from figure 8, certain versions of shaping outperform not just the baseline but also the optimal reward search scheme. This is expected in general because during reward search we are allowing only 4 degrees of freedom (the rewards corresponding to the 4 satiation states) while in shaping we have allowed potential functions based on position as well as state. However, the interesting thing to note is that while we had no hand in how search was done, the shaping scheme was entirely our own creation based on an accurate understanding of the environment. As for the value based potential not performing as well as the baseline, we attribute it to a poor guess of the value function. Moreover, the idea that expected number of time to reach full satiation may not accurately represent the value

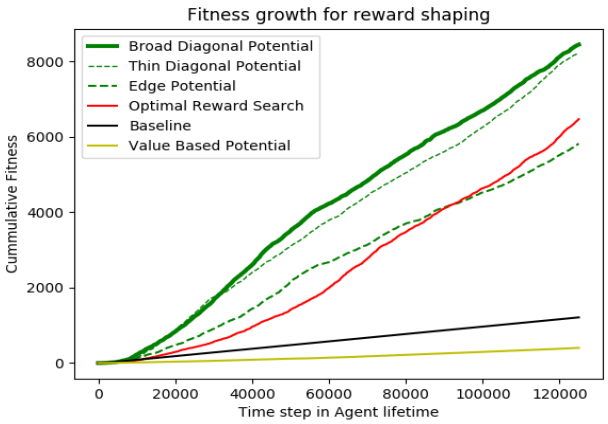

Figure 8. Comparison of the learning rate across different schemes

function for the MDP. We believe better estimates of the value function should in fact make learning of an optimal policy faster.

5. Shaping v/s Searching

Structurally, the two ideas are very different and we believe it is not appropriate to compare the results of the two approaches directly. If reward search is done on the entire reward space then it is bound to give better results than any shaping policy, simply because it is designed to obtain the global optimal over the search space of reward functions: $\mathcal{R}$. However, the computational expense in performing such a search would be immense. On appropriately reducing the search space (as we did by limiting it to only the hunger states irrespective of position) we can get convergence rates substantially faster than the baseline.

Shaping, on the other hand, is a much more intuitive approach where we guide the agent based on our understanding of the environment. The shaping functions can be as detailed as we want, and generally the computational expense corresponding to an intuitive potential function is minimal. If our guiding policies are correct, the agent indeed does outperform the baseline learning rates as can be seen from the graphs.

At the same time, shaping also provides a theoretical advantage in the sense that it guarantees convergence to an optimal policy in the limit, irrespective of the choice of the potential function. However, this approach is limited by our knowledge of the environment. In case we know little about the environment and/or are unable to formulate "good" guiding principles, the result is often worse than one without shaping.

6. Robustness of Shaping

We know that reward shaping is based on the designer's knowledge of the environment which is passed on the agent in the form of shaping functions. As is expected, when this knowledge is flawed, the shaping functions are of little help. To demonstrate this, we changed the environment a little bit and assumed that the designer is unaware of this modification and thus doesn't change the shaping functions. By running the shaping schemes in this modified environment, we show that the same schemes that worked in the Balanced Diet problem don't improve learning rates in the modified setting.

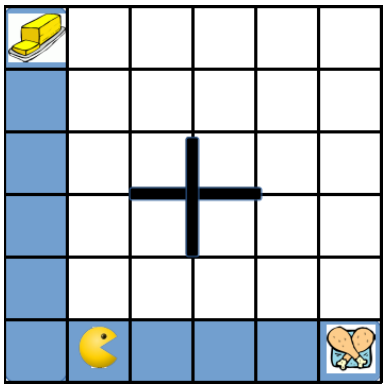

Figure 9. Modified Environment

Modification: In order to come up with an environment where the `thin diagonal potential` and `broad diagonal potential` will not speed up learning, we added some hard boundaries in the middle of the grid. These boundaries are similar to walls in that the agent can't cross them.

Also, we modified the environment such that the agent is given additional rewards for walking along the edges of the grid, which we think will make the policy of sticking to the diagonals sub-optimal. As shown in figure 9, the cross represents hard boundaries and the blue portion represents regions of additional rewards (for just being there). Specific parameters for this scheme can be found in the appendix.

In this modified setting, we run the same potential functions along with the baseline to obtain some plots, as shown in figure 10. Since the rewards have been modified and are no longer in the same range as the original problem, the scale on the y-axis isn't comparable to the original environment.

Observations: As is apparent from figure 10, the potential functions designed for the Balanced Diet problem don't improve learning if the environment has been modified. This solidifies the need of proper understanding of the environment when using potential based reward schemes and confirms that shaping (in terms of learning speed improvement) is certainly not robust to changes in the environment.

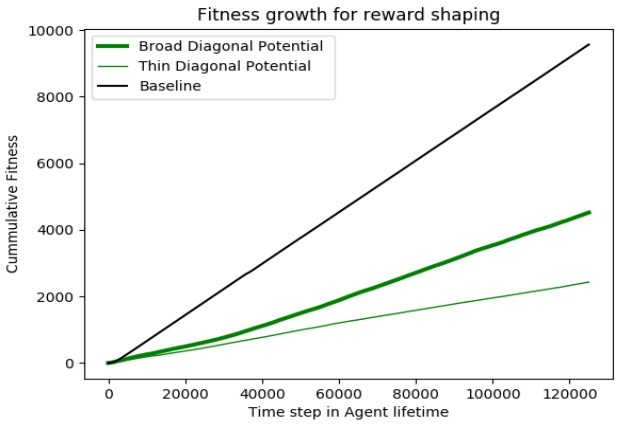

Figure 10. Comparison of the learning rate across different schemes

7. Conclusion

We have described a novel Balanced Diet problem and have discussed the implications of reward search and reward shaping in this MDP. We started with an analysis of reward search in this environment. Because of the heavy computational resources needed for exhaustive reward search, we restrict our search space to rewards based on hunger states only. Within this search space, we find an optimal reward scheme and characterise the policy obtained by the learning algorithm (Q-learning) when using this reward scheme. We discuss traits of a good reward scheme and examine why the optimal reward scheme found using reward search acts like one.

We then take a step towards reward shaping and provide details about how shaping functions work mathematically. We discuss qualities of a good shaping function and highlight methods of guessing potential based shaping functions. Next we take two different routes for guessing good shaping functions in our Balanced Diet problem, value function based potentials and potentials based on "goodness" of states. We run Q-learning using all these reward functions, and show that reward shaping does much better than the baseline when we guide the agent(via shaping functions) appropriately.

Finally, we scrutinise the robustness of shaping policies, and discover that there exist variations of the environment that if unknown to the designer may result in poor shaping functions. This shows the importance of environment knowledge when designing shaping functions.

Further work can include increasing the reward search space over positions on the grid as well as the hunger state in searching schemes.

For shaping policies, coming up with a better estimate for the optimal value function is also worth investigating, since our estimate yielded sub-par results. We believe better estimates of the quantity 'expected number of steps to full satiation' may be a good estimate, which can be computed using MDP knowledge by keeping track of position and state along any shortest path between the two food sources.

References

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99. Morgan Kaufmann Publishers Inc., 1999.

Singh, S., Lewis, R. L., and Barto, A. G. Where do rewards come from? In *Proceedings of the Annual Conference of the Cognitive Science Society*, 2009.

Singh, S. P., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Trans. Autonomous Mental Development*.

Appendix

This appendix includes parameters used in our experiments that are not explicitly mentioned in the main paper:
Under Q-learning we use:

- An exploration rate of the epsilon greedy scheme as $\epsilon = 0.1$
- A constant learning rate $\alpha = 0.5$

Parameters used in the modified environment:

- Extra fitness increment for going along edges: 40
- The agent can't move up from positions (2,2) and (3,2), can't go down from positions (2,3) and (3,3), can't go right from (2,2) and (3,2), and can't go left from (2,3) and (3,3). If it tries to it just stays there for another iteration and is penalised 0.2 fitness points.

The codes used for running experiments can be found in the github repo link https://github.com/ishank-juneja/reward-search-shaping