# A Minimal River Crossing Task to Aid the Explainability of Evolutionary Agents

Abida Ghouri[1], Chloe M. Barnes[1] and Peter R. Lewis[1]

[1]Aston University, Birmingham, B4 7ET, UK
{ghouria — barnecm1 — p.lewis}@aston.ac.uk

**Abstract**

Evolving agents to learn how to solve complex, multi-stage tasks to achieve a goal is a challenging problem. Problems such as the River Crossing Task are used to explore how these agents evolve and what they learn, but it is still often difficult to explain why agents behave in the way they do. We present the Minimal River Crossing (RC-) Task testbed, designed to reduce the complexity of the original River Crossing Task while keeping its essential components, such that the fundamental learning challenges it presents can be understood in more detail. Specifically to illustrate this, we demonstrate that the RC- environment can be used to investigate the effect that a cost to movement has on agent evolution and learning, and more importantly that the findings obtained as a result can be generalised back to the original River Crossing Task.

## Introduction

Understanding, explaining and learning about complex natural phenomena and living systems are some of the aims of artificial life research (Aguilar et al., 2014; Bedau, 2007). This often involves creating simple models that reduce the complexity of the original scenario, such that the phenomena can be studied in detail. Tasks that individuals face in both natural and artificial life can be complex, involving many stages (Brutschy et al., 2014); studying how artificial agents learn to solve these tasks can therefore be beneficial for explaining or predicting more complex artificial scenarios, or phenomena in natural life. Stanton and Channon (2015) have explored how three-dimensional virtual agents can evolve to learn sub-tasks in order to solve increasingly difficult problems. Nicolay et al. (2014) have explored the effect that the order that conflicting tasks are learnt in has on performance in virtual robots. Additionally, Brutschy et al. (2014) show in simulation that interdependent tasks can be broken down and allocated to individuals in a swarm in simulation; further, they show that the approach in simulation can be transferred to physical robots, thus demonstrating the value of performing the simulation experiments. Representing the problem with minimal complexity thus aids explainability of the problem itself.

The River Crossing Task (RCT) was originally developed to explore how agents learn to solve increasingly complex tasks (Robinson et al., 2007); initially this was done using a novel neural network architecture that separated reactive from deliberative processes. Since then, the testbed has been extended to explore the effect of social learning strategies (Jolley et al., 2016), learning by imitation (Borg et al., 2011), and social action (Barnes et al., 2019), as well as how agents learn in a 3D world (Stanton and Channon, 2015). One of the difficulties of the RCT and its extensions is that agents must learn sub-tasks in order to achieve their goal - specifically they must learn to build a bridge to cross a river safely to access their reward object, without any prior knowledge of the task or environment. Understanding why learning this sub-task is so difficult for agents may aid the explainability of the problem, and enable predictions to be made about how or why agents behave in the way that they do.

We therefore propose the Minimal River Crossing (RC-) Task, designed to reduce the complexity of the original River Crossing Task (Robinson et al., 2007) while keeping its core properties, such that agent evolution can be studied in closer detail. This is intended to increase the explainability of agent evolution by analysing the fundamental learning tasks in a simpler environment. We explore the effect that a cost to movement has on agent evolution using the RC- environment, as just one example of the concepts that can be explored in greater detail. We also ask the questions of what the impact will be on evolution when varying population sizes are used when evolving agents, and what the impact will be on evolution when Random Immigrants (Cobb and Grefenstette, 1993) are introduced in an attempt to diversify the population. We hypothesise that experimentation – and in particular the analysis that simpler problems such as this lend themselves to – using the RC- will explain and predict the outcome of the same experiments in the original River Crossing Task, thus increasing explainability of the original problem.

## Related Work

The original River Crossing Task (RCT) was developed as a testbed to investigate how agents can evolve to solve complex tasks using both reactive and deliberative behaviours

in a dynamic environment (Robinson et al., 2007). The RCT environment is a 2D grid, with a river of Water separating an agent from its target 'Resource'; the agent must learn appropriate behaviours to cross the river safely, in order to collect this Resource object and receive a reward. Various other objects exist within the environment: Stones can be picked up and placed in the river to build a bridge; Traps are dangerous items that will kill the agent if stepped upon, giving a very negative fitness; all 'empty' cells represent Grass, which is safe to pass over. Agents must first learn to avoid the river, otherwise they will drown and receive a highly negative fitness. They can then explore a neutral fitness landscape until they learn that Stones can be used to build bridges; building a bridge itself does not provide any reward, but it does enable agents to reach their target object within the environment - the Resource.

Whilst the RCT in design is simple, the task involved is complex for agents to learn to solve; a contributing factor to this is that agents are evaluated on a series of environments that increase in complexity. This inspired extensions to the testbed, such as the RC+ task, designed to increase in complexity such that the final environment cannot be solved by incremental evolution on its own (Borg et al., 2011). Current strategies to solve the RC+ task include learning by imitation through transcription errors and cultural transmission in the RC+ environment (Borg et al., 2011), and by employing teacher-learner social learning strategies (Jolley et al., 2016). Other extensions include the 3D River Crossing (3D RC) Task used to evolve 3D virtual agents in a 3D world (Stanton and Channon, 2015), and the River Crossing Dilemma (RCD), to explore how agents coevolve to pursue individual goals in a shared environment (Barnes et al., 2019).

A common factor in the RCT and its variants is that the seemingly simple act of building a bridge involves learning two implicit things: adapting learnt knowledge that the river is 'safe' *if* a stone is being carried, and that the neutral behaviour of putting a stone in the river is beneficial in the long-term. An agent therefore depends on learning this subtask of building a bridge first in order to achieve its goal, and must endure a period of low fitness during evolution in order to find the optimal solution.

Task decomposition has been shown to be powerful when using a neuroevolutionary approach for learning sequential sub-tasks (Jain et al., 2012), and when using modular connectionist architectures (Jacobs et al., 1991). Nicolay et al. (2014) conclude that, when training a robot to learn two conflicting tasks, learning the 'harder' task before learning both simultaneously is more beneficial than learning both together initially. One issue with current approaches to solving the RCT is that agents use neuroevolution to evolve a combination of two neural networks for reactive and deliberative behaviours (Robinson et al., 2007). As complexity of the task and the sub-tasks involved increases, so does the learning architecture required; this means that the search space also increases, and evolving optimal solutions is more difficult (Federici and Downing, 2006).

Whilst approaches such as task decomposition and evolving modularity in the learning architecture may be beneficial for designing agents to solve the RCT and its variants, our focus is rather exploring *why* the task is difficult for agents to learn at its core. To study this in more detail, an appropriate testbed is required with reduced complexity, paired with a simplified agent representation and learning architecture.

## The Minimal River Crossing Task Testbed

We propose the Minimal River Crossing (RC-) Task testbed - an abstraction of the original River Crossing Task (RCT) and its extensions. The RC- is designed to reduce the complexity of the environment and its representation, such that the core task of building a bridge can be studied in closer detail.

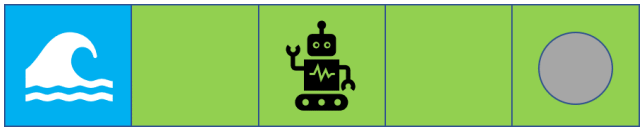

Figure 1: The Minimal River Crossing Task Testbed: a $1 \times 5$ grid-world environment with water in the leftmost cell, and grass in all others. There is a stone in the rightmost cell, and an agent in the center cell.

The RC- testbed consists of a $1 \times 5$ grid, containing Water in the leftmost cell, a Stone in the rightmost cell, and Grass in all others (including underneath the Stone). The goal of the agent, starting in the centre cell, is to build a bridge by first picking up the Stone, and then heading to the Water to build a bridge. In the original RCT environment, it is this act of building a bridge that allows an agent safe passage across the river in order to collect the reward object; thus the act of collecting the reward object is not accessible unless a bridge is built. The RC- therefore simplifies the agent's reward state by only awarding a highly positive fitness when it has built a bridge. An agent dies if it steps into the Water without carrying a Stone, and receives a highly negative fitness. Another simplification present in the RC- is a reduction in the existing objects in the environment to only those that are necessary to achieve the goal; excess Stones, other objects such as Traps, and Resource objects, which can all be seen in other variations of the RCT, are therefore not used in the RC- environment.

### Agent Representation

An agent's genome is represented using a pair of integers $(R, L)$, each in the range $[0, 4]$, where $R$ is the number of moves to the right it will take, and $L$ is the subsequent number of moves to the left. Further, a strategy of $(1R, 2L, 3R)$


Figure 2: The fitness landscape of all 25 possible agent solutions in the RC- environment, where there is no cost to movement. The 13 invalid solutions are indicated with an 'x', whereas the 12 valid solutions show the fitness received if the agent moves the specified amounts to the right then left. A fitness of $+10$ is received if an agent makes 2 moves to the right to pick up a Stone, then left 4 to build a bridge in the water. If an agent moves into the Water without first picking up a Stone, it will receive a fitness of $-10$.

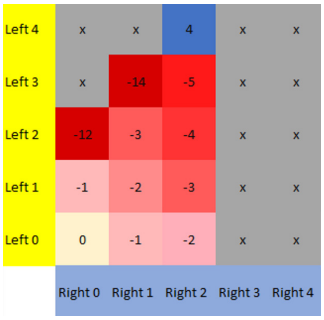


Figure 3: The fitness landscape of all 25 possible agent solutions in the RC- environment, where there is a cost of $-1$ per move. The 13 invalid solutions are indicated with an 'x', whereas the 12 valid solutions show the fitness received if the agent moves the specified amounts to the right then left (i.e. 2 moves right then 4 left gives a fitness of $+4$, after a movement penalty of $-6$). Rewards and penalties are the same as in Figure 2, with an additional cost to movement.

for example can be simplified to $(2, 0)$; this removes unnecessary complexity from the genome representation, but also means that the genome represents the outcomes of more complex movement strategies in a simple manner. As a result, there are a total of 25 possible agent solutions between $(0, 0)$ and $(4, 4)$, with 12 being valid (e.g. $(1, 2)$) and 13 being invalid (e.g. $(3, 2)$); from the starting position, an agent can take a maximum of two moves to the right, then four to the left, resulting in the optimal solution of $(2, 4)$.

The fitness $f$ for agent $A$ is calculated with Equation 1:

$$A_f = Is - Iw - (c \times m) \tag{1}$$

where $s = 1$ if a bridge is built successfully with a Stone and 0 otherwise, $w = 1$ if the agent falls into the Water and 0 otherwise, $c = 1$ if there is a cost to movement or 0 otherwise, and $m$ is the number of moves taken by the agent. $I$ is a constant where $I = 10$, indicating the magnitude of the reward or penalty. Agents can thus be characterised as the Builders (achieve the goal), the Dead (receive a penalty for drowning), and the Neophobic (do nothing).

## Landscape Analysis

As an agent's genome is represented using a pair of integers, each in the range $[0, 4]$, the fitness landscape for the RC- environment can be fully mapped with a $5 \times 5$ grid. Figures 2 and 3 show the fitness landscapes for the RC- environment, without and with a cost for movement respectively. Out of the 25 possible agent solutions, 12 are valid and 13 are invalid (e.g. agents can make at most two moves to the right). As the agent has no prior knowledge of the environment or its location within it, restricting the search space to $3 \times 5$ would provide additional information and bias the problem.

Without a cost to movement, Figure 2 shows that a neutral landscape is created; agents can explore without penalty, and there is no indication of proximity to the optimal solution. However, when a cost to movement is introduced, Figure 3 shows that a trap function is created in the landscape (Nijssen and Bäck, 2003). This means that agents receive a lower fitness as they get closer to the optimum, making it difficult for evolution to find.

## Experimental Design

All experiments evolve agents using an evolutionary algorithm (EA) with the following common parameters. Agents evolve for 50 generations, and experiments are repeated 30 times. At each generation, the worst agent is selected and evaluated as the 'parent'. All experiments are repeated with and without a cost to movement, to explore how this impacts evolution and learning. Mutation occurs by adding 1 to a random element in the genome unless otherwise specified. Agents have no prior knowledge of the task or environment, and are randomly initialised at the start of evolution.

The first set of experiments explore the effect a cost to movement has on evolution and learning, using a 1+1 evolutionary algorithm (EA). The 1+1 EA has been shown to be simple yet effective for learning simple tasks (Droste et al., 2002; Nijssen and Bäck, 2003); here, an EA evolves a population of size 1 with small mutations and no crossover. The created offspring replaces the parent only if its received fitness is better than or equal to the fitness of the parent.

The second set of experiments increase the population size in the EA to 5, to explore the effect of this change on evolution and learning. The offspring is a mutation of the parent, which it always replaces.

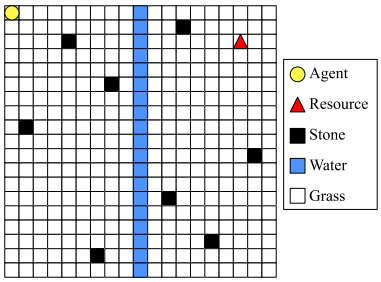


Figure 4: The base RCT environment inspired by Robinson et al. (2007), adapted from Barnes et al. (2019). The goal of the agent is to collect the Resource object on the opposite side of the river; to do this, it must learn to build a bridge with a Stone by placing it in the Water.

The third set of experiments also use a population size of 5, to explore the effect of Random Immigrants on evolution and learning. Random Immigrants inject randomness into the population, leading to greater diversity (Cobb and Grefenstette, 1993). In these experiments, the parent is the agent with the worst fitness in the population at each generation; if the offspring has a better or equal fitness to the parent then the parent is replaced with the offspring, otherwise the parent is replaced with a randomly-initialised solution.

The final set of experiments explore whether the results obtained using the RC- environment generalise back to the original RCT environment. The $19 \times 19$ environment has one Resource/Reward object, eight Stones and a one-cell deep river of Water. Agents use a neural network learning architecture inspired by previous work (Barnes et al., 2019), where weights of the network are mutated by a random value from a Gaussian distribution with $\mu = weight$ and $\sigma = 0.01$; crossover does not occur. Parents are replaced in line with the corresponding experiments above. The fitness $f$ for agent $A$ is calculated with Equation 2:

$$A_f = Ir - Iw - (c \times \frac{m}{2T}) \quad (2)$$

where $r = 1$ if the Resource is collected and 0 otherwise, $w = 1$ if the agent falls into the Water and 0 otherwise, $c = 1$ if there is a cost to movement and 0 otherwise, and $m$ is the number of moves the agent has made. $I$ and $M$ are constants, where $I = 1$ and $T = 500$ (the total number of moves allowed by the agent). Agents must always make a move at each time step - they cannot remain in the same place. By dividing the amount of moves by double the total number of allowed moves for the cost to movement, agents that use the maximum amount of moves whilst still achieving the goal ($A_f = 0.5$) will receive a better fitness than those that are Neophobic and do not achieve the goal ($A_f <= 0.0$); this therefore still rewards successful agents that move excessively.

# Experimentation

## Cost to Movement and Evolution

When there is no cost to movement, agents encounter a neutral landscape (Figure 2) in which they can explore without penalty; there is no explicit incentive to exploration, but more importantly there is no inherent *disincentive* to exploration. As such, agents across all 30 runs are able to evolve a Builder solution within the first 12 generations of 50 (Figure 5). Once found, these solutions are maintained throughout the rest of the evolution.

In contrast, adding a cost of $-1$ per move has a dramatic effect on the ability for agents to evolve, resulting in the majority of agents across the 30 runs evolving to be Neophobic; some agents do not move at all, whilst others venture at most four moves away from the starting point which gives them a negative fitness. A trap is created in the fitness landscape (Figure 3), as agents must endure negative fitness in order to evolve a solution that is capable of achieving the goal. Agents are therefore found to be discouraged from exploring when a cost to movement is present, and are encouraged to be Neophobic.

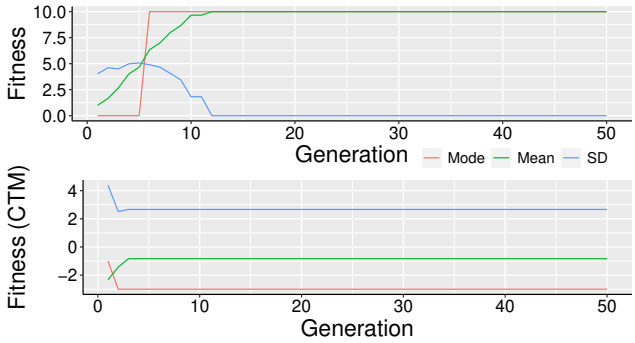


Figure 5: The mean, mode and standard deviation in fitness for 30 runs of a 1+1 EA in the RC-, without and with a cost to movement (CTM) respectively. A cost to movement impacts the chance that successful solutions are evolved.

## Population Size and Evolution

Increasing the population size in the evolutionary algorithm means that there is more diversity in the population, and thus more ability to traverse the search space. As with the previous set of experiments, no cost to movement means that agents are able to explore without penalty. Agents across all 30 runs evolve a Builder solution by generation 27 out of 50.

Interestingly, the addition of a cost to movement with an increased population size appears to enable agents to evolve a Builder solution faster than without, when looking at the best-in-population fitness (Figure 6); all 30 runs evolve a Builder solution, which is an improvement compared to the same experiment with a population size of 1.

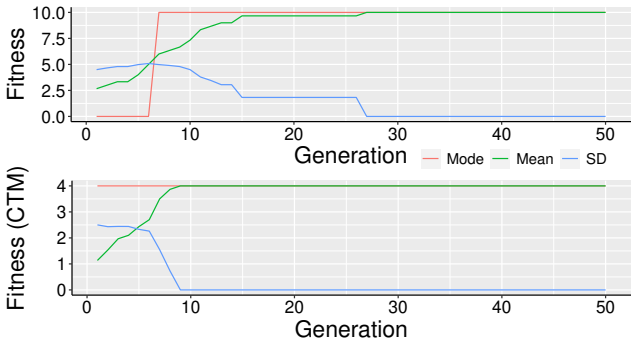


Figure 6: The mean, mode and standard deviation of the *best*-in-population fitness for 30 runs of an EA with a population size of 5 in the RC-, without and with a cost to movement (CTM) respectively.

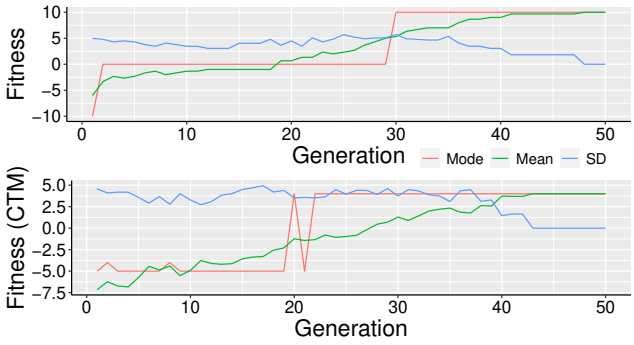


Figure 7: The mean, mode and standard deviation of the *worst*-in-population fitness for 30 runs of an EA with a population size of 5 in the RC-, without and with a cost to movement (CTM) respectively.

Compared to the best-in-population fitness (Figure 6), the worst-in-population fitness (Figure 7) shows the diversity in the population and how quickly the population converges on the goal-achieving Builder solutions. When agents are not subjected to a cost for movement, high-fitness solutions propagate quicker through the population once found than when there is a cost for movement.

## Random Immigrants and Evolution

Injecting randomness into a population via Random Immigrants during evolution increases the diversity and ability of the population to traverse the search space (Cobb and Grefenstette, 1993). Adding Random Immigrants to the 1+1 EA used in the previous experiments would resemble random search, as the population size is 1. As such, we explore the effect of Random Immigrants when agents learn with an evolutionary algorithm with a population size of 5, in line with previous experiments.

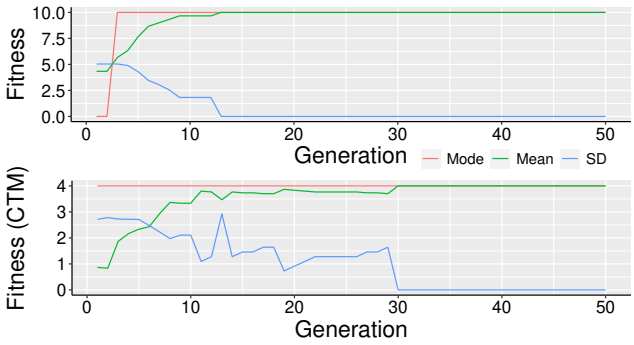


Figure 8: The mean, mode and standard deviation of the *best*-in-population fitness fitness for 30 runs of an EA with Random Immigrants and a population size of 5 in the RC-, without and with a cost to movement (CTM) respectively.

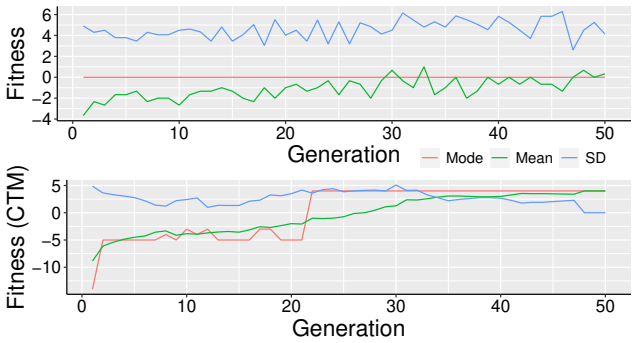


Figure 9: The mean, mode and standard deviation of the *worst*-in-population fitness fitness for 30 runs of an EA with Random Immigrants and a population size of 5 in the RC-, without and with a cost to movement (CTM) respectively.

Agents are on average able to evolve Builder solutions more quickly when they encounter a neutral fitness landscape as a result of no cost to movement, compared to when there is a trap in the fitness landscape caused by a cost to movement (Figure 8).

When looking at the worst-in-population fitness (Figure 9), the additions of a cost to movement and Random Immigrants appear to facilitate convergence towards the optimal Builder solution, compared to no cost to movement. This may be because the worst solution is always replaced by an offspring (either by a mutation of the parent or a random solution). As adding a cost to movement creates a trap function in the fitness landscape (Figure 3), agents receive a lower fitness as they get closer to the optimum; therefore, the higher-fitness optimal solution appears to become more accessible to agents with Random Immigrants, bridging the gap that the trap function creates and promoting convergence.

This is not as prominent when there is no cost to movement; agents experience a neutral fitness landscape, and as the worst agent is replaced by its mutated offspring if the fitness of the latter is better than or equal to the former, Random Immigrants are less likely to be injected into the population than when there is a cost to movement (as there is more variation in the fitnesses in the population). This results in a cost to movement appearing to facilitate convergence of the population towards the optimal solution, and not when there is no cost to movement.

## Generalising to the Original RCT Problem

The previous experiments were repeated in the base RCT environment (Figure 4), in order to ascertain whether the findings hold true when the complexity of the task, environment and agent design is increased.

**Addition of Cost to Movement** The increased complexity in the RCT environment and agent representation, compared to the RC-, means that it is more difficult for agents to evolve successful Builder solutions both with *and* without a cost to movement; only 9 out of 30 runs with a 1+1 EA in the RCT environment evolve Builder solutions (Table 1, both without and with a cost to movement, rows 7 and 8 respectively), compared to 30 without and 6 with a cost to movement in the RC- (Table 1, rows 1 and 2 respectively).

Despite both experiments (with and without a cost, 1+1 EA) evolving the same number of Builder solutions in the RCT, Figure 10 shows that the average fitness in the RCT is lower with a cost to movement. This is to be expected because agents that are successful receive a lower fitness when subjected to a cost to movement, than when they are not. This is in line with the findings of the same experiments conducted in the RC- environment, depicted in Figure 5.

**Increase in Population Size** When the population size is increased from 1 to 5, the increased diversity in the population enables more agents across each of the 30 runs to evolve a successful Builder solution; 24 runs out of 30 are successful by 50 generations when there is no cost to movement, and 12 with a cost (Table 1, rows 9 and 10 respectively). This also corroborates the results from the RC- environment, in line with our hypothesis, as more agents evolve to be successful when there is an increased population size (Table 1, rows 1-4 compared to 7-10).

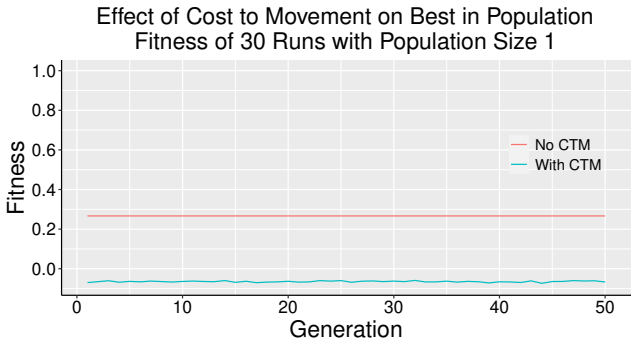


Figure 10: The mean fitness for 30 runs of a 1+1 EA in the base RCT, without and with a cost to movement (CTM) respectively.

Table 1: The number of runs out of 30 in the RC- or base RCT in which a Builder solution is found in the population after 50 generations, where $p$ is the population size, $CTM$ is the cost to movement ($CTM = 0$ is no CTM, $CTM = 1$ is a CTM) and $RI$ is that Random Immigrants are introduced.

| ID | Experiment | Builders |
|---|---|---|
| 1 | (RC-) $p = 1$, $CTM = 0$ | 30 |
| 2 | (RC-) $p = 1$, $CTM = 1$ | 6 |
| 3 | (RC-) $p = 5$, $CTM = 0$ | 30 |
| 4 | (RC-) $p = 5$, $CTM = 1$ | 30 |
| 5 | (RC-) $p = 5$, $CTM = 0$, $RI$ | 30 |
| 6 | (RC-) $p = 5$, $CTM = 0$, $RI$ | 30 |
| 7 | (RCT) $p = 1$, $CTM = 0$ | 9 |
| 8 | (RCT) $p = 1$, $CTM = 1$ | 9 |
| 9 | (RCT) $p = 5$, $CTM = 0$ | 24 |
| 10 | (RCT) $p = 5$, $CTM = 1$ | 12 |
| 11 | (RCT) $p = 5$, $CTM = 0$, $RI$ | 28 |
| 12 | (RCT) $p = 5$, $CTM = 0$, $RI$ | 12 |

Figure 11 shows the impact that introducing a cost to movement can have on evolution and fitness. The risk of receiving a lower fitness when moving acts as a deterrent to exploration; learning the sub-task of building a bridge is more difficult as a period of low fitness must first be endured, meaning that these low-fitness but potentially successful solutions are more likely to be replaced during evolution.

When agents evolve in the RC- environment with a population size of 5, no cost to movement allows successful solutions to propagate quicker through the population once found, compared to when there is a cost to movement (Figures 6 and 7). This is corroborated in the base RCT environment, as less agents evolve successful solutions when there is a cost to movement than without; finding a successful solution is more difficult when there is an incentive to be neophobic, through a penalty for exploration (Figure 11).

**Introduction of Random Immigrants** The introduction of Random Immigrants into a population of size 5 means that parts of the search space become accessible, that may not be accessible via incremental evolution alone. 28 out of the 30 runs were able to find successful solutions after 50 generations with no cost to movement, compared to 12 with a cost to movement (Table 1, rows 11 and 12 respectively).

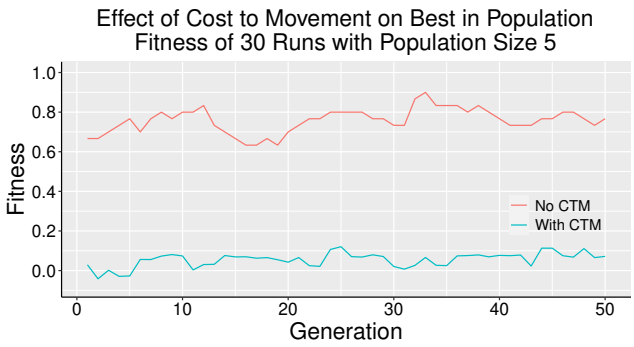


Figure 11: The mean *best*-in-population fitness for 30 runs of an EA with a population size of 5 in the base RCT, without and with a cost to movement (CTM) respectively.

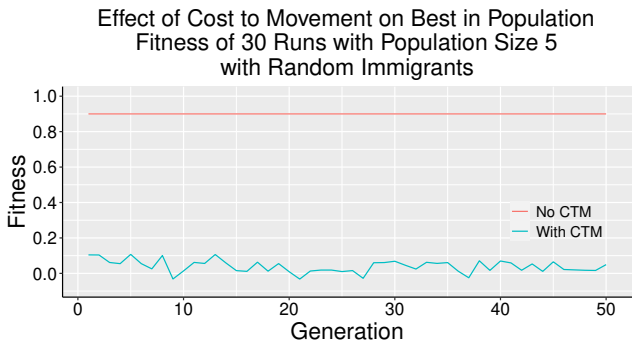


Figure 12: The mean *best*-in-population fitness for 30 runs of an EA with Random Immigrants (RI) and a population size of 5 in the base RCT, without and with a cost to movement (CTM) respectively.

Random Immigrants have a positive effect on the amount of Builder solutions found during evolution compared to no Random Immigrants (Table 1, rows 9 and 11), and no effect when there is a cost for movement (Table 1, rows 10 and 12).

The effect of Random Immigrants in the RCT environment is demonstrated in Figure 12. Compared to the RC- environment (Figure 8), a cost to movement appears to have a more negative effect on agent evolution when there is more complexity in the environment, than in a simpler environment. In the RCT environment, agents that do not find the Resource (i.e. by first building a bridge) within the maximum amount of moves will receive the same cost to movement penalty (Using Equation 2, $c = 1$, $m = 500$ and $T = 500$, resulting in a penalty of $0.5$). This creates a neutral fitness landscape that is not seen when there is a cost to movement in the RC- environment; it is therefore harder for agents to evolve successful solutions.

As the evolutionary algorithm only introduces Random Immigrants if the mutated offspring has a worse fitness than the parent, the experiments in the RCT both with and without a cost to movement create neutral fitness landscapes; this means that the chance for Random Immigrants to be introduced is low (only when the offspring's fitness is worse than the parent). This provides supporting evidence for the findings in the RC- environment with a population size of 5, regarding the effect of Random Immigrants on populations where there is a neutral fitness landscape.

## Discussion

An issue highlighted when using a 1+1 evolutionary algorithm in the RC- is that the introduction of a cost to movement creates a trap function in the fitness landscape; the received fitness gets lower as agents get closer to the optimum. This makes it difficult for successful Builder solutions to evolve because selection pressure in the evolutionary algorithm favours higher-fitness solutions. With a 1+1 evolutionary algorithm, unless agents are initialised with the optimal solution (or within one mutation), incremental evolution will be unable to find the optimum.

By increasing the population size from 1 to 5, the diversity of the population in the RC- increases; more of the search space can therefore be traversed or accessed during evolution. This results in an increase in the number of agents that evolve successful Builder solutions - both with and without a cost to movement. It is worth noting that the search space for the RC- environment consists of 12 valid solutions, so a population size of 5 can cover nearly half of the overall search space if the solutions are unique. This lends itself to explain why all agents both with and without a cost to movement are able to evolve successful Builder solutions. Whilst an increase in population size may help more agents to evolve successful solutions than a 1+1 EA, incremental evolution may still be problematic for agents that are far from the optimum depending on where in the search space they begin.

The introduction of Random Immigrants in the RC- is shown to be beneficial when agents face a fitness landscape with a trap, and less useful when there is a neutral landscape. This is because the evolutionary algorithm only replaces the parent if the offspring is better than or equal to the parent; therefore, a neutral landscape means there is less chance for Random Immigrants to be introduced. As a result, populations of agents subjected to a cost to movement appear to converge to the optimum quicker than those that are not.

Increasing the complexity of the task itself from the RC- to the RCT demonstrates a number of things. A 1+1 EA is not as effective for evolving Builder solutions as an increased population size of 5 - both with and without a cost to movement. This is because agents have a more complex learning architecture, creating a larger search space. It is also observed that adding a cost to movement creates a neutral (negative) fitness landscape, as agents that do nothing

will use the maximum amount of moves and therefore incur the maximum cost to movement. By referring to the results in the RC-, we can expect to see Random Immigrants be less effective when there is a neutral landscape in the RCT environment. This is observed in Figure 12. Another point to mention when analysing Figure 12 is that the number of agents that are successful Builders is constant over time; this means that Random Immigrants does not benefit the agents, but crucially does not harm them. This is in line with the findings from the same experiments introducing Random Immigrants conducted in the RC- environment.

As the agent representation in the RCT is more complex than in the RC-, we hypothesise that we would see similar convergence to optimal solutions to those found in the RC- if agents were evolved for longer than 50 generations.

## Conclusions

Previous work using the River Crossing Task and its extensions have identified that learning the sub-task of building a bridge is a difficult problem. We have proposed the Minimal River Crossing (RC-) Task as a simple yet sufficient testbed to explore why this fundamental learning task is difficult in closer detail, as an example of the concepts that can be explored. Analysis of the RC- fitness landscape, as well as a series of experiments with a 1+1 evolutionary algorithm, an increase in population size and the introduction of Random Immigrants have shown that adding a cost to movement in the fitness function makes learning this already difficult task less likely.

The results obtained by repeating these experiments in the base River Crossing Task support those obtained from the RC- environment. We have therefore demonstrated the value of using the RC- for increasing the explainability of the original problem, by showing that the fundamental learning tasks can be studied in detail using an environment with minimal complexity.

## References


Aguilar, W., Santamaría-Bonfil, G., Froese, T., and Gershenson, C. (2014). The Past, Present, and Future of Artificial Life. *Frontiers in Robotics and AI*.

Barnes, C. M., Ekárt, A., and Lewis, P. R. (2019). Social action in socially situated agents. In *Proceedings of the IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 97–106.

Bedau, M. A. (2007). Artificial life. In Matthen, M. and Stephens, C., editors, *Philosophy of Biology*, Handbook of the Philosophy of Science, pages 585 – 603. North-Holland, Amsterdam.

Borg, J. M., Channon, A., and Day, C. (2011). Discovering and maintaining behaviours inaccessible to incremental genetic evolution through transcription errors and cultural transmission. In *Proceedings of the European Conference on Artificial Life 2011*, pages 102–109. MIT Press.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., and Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*.

Cobb, H. G. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 523–530, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1 + 1) evolutionary algorithm. *Theoretical Computer Science*.

Federici, D. and Downing, K. (2006). Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. *Artificial Life*.

Jacobs, R. A., Jordan, M. I., and Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*.

Jain, A., Subramoney, A., and Miikulainen, R. (2012). Task decomposition with neuroevolution in extended predator-prey domain. In *Artificial Life 13: Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems, ALIFE 2012*.

Jolley, B. P., Borg, J. M., and Channon, A. (2016). Analysis of social learning strategies when discovering and maintaining behaviours inaccessible to incremental genetic evolution. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9825 LNCS, pages 293–304.

Nicolay, D., Roli, A., and Carletti, T. (2014). Learning multiple conflicting tasks with artificial evolution. In *Communications in Computer and Information Science*.

Nijssen, S. and Bäck, T. (2003). An analysis of the behavior of simplified evolutionary algorithms on trap functions. *IEEE Transactions on Evolutionary Computation*.

Robinson, E., Ellis, T., and Channon, A. (2007). Neuroevolution of agents capable of reactive and deliberative behaviours in novel and dynamic environments. In *Advances in Artificial Life*, pages 1–10. Springer.

Stanton, A. and Channon, A. (2015). Incremental Neuroevolution of Reactive and Deliberative 3D Agents. *European Conference on Artificial Life (ECAL)*, pages 341–348.