Learning and problem solving: The unexpected drawbacks of rationality, uncertainty reduction and learning on exploring a fitness landscape

J. Kasmire[a,*], Janne M. Korhonen[b], Igor Nikolic[a], Gerard Dijkema[c]

[a] *Faculty of Technology, Policy and Management, Section Energy and Industry, TU Delft, Postbus 5015, 2600 GA, Delft, The Netherlands*
[b] *Aalto University, School of Business, P.O. Box 21210, FI-00076, Aalto, Finland*
[c] *IVEM, Energy and Sustainability Research Institute Groningen (ESRIG), Nijenborgh 4, 9747 AG Groningen, The Netherlands*

Abstract

Learning, rationality and the reduction of uncertainty are generally understood to contribute to better problem solving, both by allowing more problems to be solved and by solving those problems more efficiently. Consequently, learning, rationality and uncertainty reduction are directly embedded in the scientific method and other problem solving frameworks. Nevertheless, some agent-based models of problem solving suggest that learning, rationality and uncertainty reduction can be detrimental to long term problem solving by increasing problem solving efficiency at the expense of the ability to solve more problems. This paper presents a new agent-based model that draws on previous models of problem solving to further explore the relationship between learning, rationality, uncertainty reduction and problem solving. The results show that uncertainty reduction improves problem solving efficiency but not the number of problems that are solved, and further suggest that learning can increase as well as decrease uncertainty with the key to complete and efficient problem solving hinging on the balance between uncertainty growth and reduction.

Keywords: problem solving, reducing uncertainty, learning, rationality,

*Corresponding author
Email addresses: janne.m.korhonen@aalto.fi (Janne M. Korhonen), i.nikolic@tudelft.nl (Igor Nikolic), g.p.j.dijkema@rug.nl (Gerard Dijkema)
URL: j.kasmire@tudelft.nl (J. Kasmire)

agent-based models, fitness landscapes

1. Introduction, background and specific question identification

Introduction. Modelling problem solving in complex, socio-technical systems often visualises the problems and their potential solutions as a metaphorical landscape. The elevated regions of the landscape represent societal needs, such as transport, communication or housing, while every point on the surface of the landscape represents a potential solution or technology, including devices, processes, behaviours, or other innovations [1]. The elevation of each point represents its 'fitness', or how well that solution or technology satisfies the nearest societal need. The highest points represent solutions or technologies that best meet a given societal need while nearby but lower points representing solutions or technologies that meet the same need but do so less well. In this metaphor, problem solving is a matter of landscape exploration to locate the elevated regions and reach the peaks.

Although it may not be immediately obvious, this view of problem solving as landscape exploration requires learning in some form. Learning is a matter of acquiring or synthesizing new information that adds to, modifies or reinforces existing knowledge, behaviours, skills, values or preferences [2], either by incorporating many and diverse sources of information at once or by accruing information over time. The acquisition or synthesis of new information increases understanding or reduces uncertainty about the landscape, allowing the learner to identify elevated regions and compare different landscape points. Further, without learning, any solutions could not be remembered or retained, meaning that the problems could not really be described as solved. Since problem-solving requires learning, then it might seem reasonable to expect that *more* learning means *better* problem solving. However, some modelling experiments [3, 1] cast doubt on the validity of this conclusion. At the very least, the relationship between learning and problem solving warrants more investigation.

Background. The metaphorical landscape view of problems and problem solving is generally based on the NK model [4] which consists of a solution space defined by the parameters N and K and agent(s) that move through that space searching for 'better' solutions. In management and innovation literature, N typically represents the number of decisions that have to be made and the number of ways different ways to make those decisions [1]. Each point in that space represents a unique combination of decisions, so N can be understood as defining how 'big' the landscape is. The K parameter controls how interconnected these decisions are and determines the topology of the landscape. When K is as low as possible, all decisions are totally independent of each other and each has its own optimum. The single point where all of those individual optimal decisions intersect is the global optimum, which appears as the highest point at the top of a single peak in the landscape. Points which have most but not all of the individually optimal decisions will be near to the global optimum but at slightly lower elevations on that single peak while the least optimal combinations of decisions have the lowest elevations possible. As K rises, the decisions become interdependent, meaning that there is no longer a single optimal solution or a single highest point in the landscape. Instead, the landscape grows more rugged, with multiple high points, each representing a combinations of decisions that is locally optimal. The ruggedness of the landscape also means that points near to the peaks have many of the same decisions but may or may not have similar elevations (Figure 1). Real-world problems are often part of complex socio-technical systems with multiple interdependencies and so are not often well represented by low K fitness landscapes.

Agents are typically incapable of learning or rational behaviour and instead move randomly across the landscape in search of the peaks through 'steps' (altering one decision at a time to move between adjacent points) or through 'jumps' (altering several decisions simultaneously to arrive at new, non-adjacent positions) [1]. Regardless of how they move, applying selection pressure drives the entire population up the peaks to produce seemingly rational, functional and order-driven behaviour [5]. The learning in this case takes place at the abstract

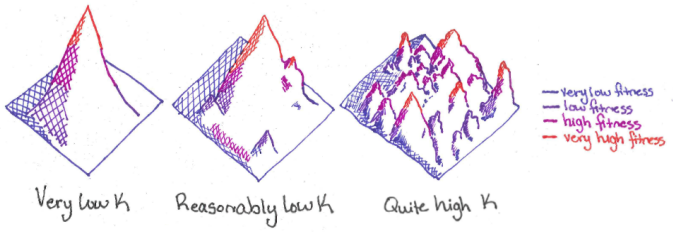

Figure 1: A set of fitness landscapes ranging from very low K (one large peak) to quite high K (many peaks of varying heights).

population level, is unlikely to discover the global optimum efficiently, and is most successful when agents are numerous and well scattered than when they are few or clustered. Relying entirely on population level learning highlights the contingency, agency and social construction of any solutions discovered while simultaneously downplaying or completely removing rationalism or functionalism for the individuals [1].

Unlike the modelled agents, humans trying to solve real-world problems in complex socio-technical systems and have reasons for what they do [6], are rational and can learn about problems and solutions. Human rationality is 'bounded' rather than perfect [7], meaning that searches for possible solutions are based on imperfect knowledge and 'satisficing' search strategies that stop the search when an option is found that seems like an improvement over the current state or that meets some minimum criteria [8]). Learning may not be able to change the satisficing search strategies, but if it increases understanding or reduces uncertainty [9] about the problems or the available means by which those problems might be solved, then learning improves the ability to plan and manage progress toward solving problems or developing technological solutions [10]. Populations of rational and learning agents should be as good or better at solving problems in high K fitness landscapes than populations of agents that are not capable of learning or rational behaviour.

However, testing that expectation produces mixed results. For example, Knudsen and Levinthal **(author?)** [3] experimented with populations of bound-

edly rational agents capable of learning about the fitness landscapes as they searched. They found that agents with perfect knowledge of the landscape and the options it presented quickly and efficiently reached a local optimum, but then behaved very conservatively and stopped moving rather than ever step downhill to find the global optimum. Their chances of finding the global optimum was just as contingent and subject to their initial position as the populations of irrational and learning-free agents guided entirely by selection pressure. Knudsen and Levinthal also found that agents with less perfect knowledge of the landscape and the options for moving across it behaved much less conservatively, so that they usually moved uphill but sometimes took downhill steps that allowed them to escape local optima. Although these less knowledgeable agents sometimes ended up at very low elevations, their risk-taking behaviours tended to robustly self-correct so that they eventually clustered near the global optimum. Essentially, the less knowledgeable agents were more likely to find the global optimum than the agents with perfect knowledge, but were much slower at finding their ultimate peak. What's more, hill climbers that start with poor landscape knowledge but improved that knowledge by learning from experience changed from being risk-takers to being conservatives, meaning that they gradually lost the ability to escape local optima.

Rapid and efficient discovery of local optima is only 'better' than slow and inefficient searches in low K landscapes where the local optimum is also the global optimum [3]. At the same time, finding the global optimum in a high K landscape is only 'better' if speed or efficiency is not important. Since real-world problems are likely to be better represented by high K landscapes, this rather counter-intuitively suggests that less knowledge, less perfect rationality and less learning are preferable for solving problems in complex, real-world, socio-technical systems. On the hand, the scientific method and other rational, learning-based approaches are usually advocated for problems identified as very urgent, such as unsustainability [9]. Unfortunately, problems like unsustainability are likely to be both complex and urgent, suggesting that they need to be approached with both more and less knowledge, rationality and learning.

The specific question identification. How can the conflicting demands to both increase and decrease knowledge, rationality and learning be reconciled? Perhaps the answer lies in the idea that a little learning or knowledge is more dangerous than much learning or knowledge would be [11]. Looking at the issue more closely, rational progress toward technological development or problem solving is suggested to require knowledge of the problems to solve and the means of solving those problems [10], but Knudsen and Levinthal's agents did not only knew about problems in the form of knowledge about the landscape. Rationality combined with limited landscape information drove the agents up the first hill they encountered, severely limiting their experiences and opportunities for learning. More importantly, the agents were incapable of knowing, learning about or controlling the means of problem solving, which can be interpreted as how to control movement across the landscape. Because they were only able to step, Knudsen and Levinthal's agents would be unable to reach the global optimum if it required some downhill steps, even if they had perfect knowledge about the entire landscape and could unambiguously identify the global optimum.

Of course, innovations can incorporate multiple changes at once. Further, the combinatorial nature of technological evolution means that even a very small, single change, such as replacing or duplicating a single component, can produce unexpected and far reaching results [12]. This suggests that problem solving through the use of innovations and technologies is not well represented by a landscape exploration where all steps are the same size. Adding a representation of combinatorial evolution allows the traditional fitness landscape model of problem solving to include steps of variable length [13]. Rationality and learning can then be added to create a new model of problem solving that allows agents to learn not only about the landscape but also the ability to move across it in different ways.

By adding more capacity to learn, rather than less, this new model of problem solving may allow rational, learning agents to solve problems that are both urgent and complex through learning-focussed approaches such as the scientific

method. On the other hand, if the agents are still incapable of improving both the speed and thoroughness of problem solving, then the roles of learning and rationality may need to be re-evaluated.

2. The model structure, operation, variable parameters and metrics

Structure. The model structure consists of a landscape, technologies and a rational and learning developer agent. The landscape is created by a peak density, valley density and maximum integer, all supplied by the modeller prior to initialisation. Peaks represent societal needs or problems to solve, such as transport, housing or communications. Valleys represent impossible or unworkable dead end technologies such as glass hammers, chocolate teapots or perpetual motion machines. When the model is initialised the peaks and valleys are randomly allocated according to their respective densities to integers between one and the maximum integer, which can be understood as the 'size' of the landscape. In this way, the peak density, valley density and maximum integer interact to determine how high or low K the landscape is. When a peak or valley is allocated to an integer within the landscape, that integer is added to a list so that the landscape is represented by two lists of integers, one for peaks and one for valleys. There are no restrictions on the numbers that can appear on the two lists, so that sequential numbers can both be peaks, valleys or one of each. Further, any given number can be both a peak and a valley, representing a perfect but unattainable solution to a societal need.

The technologies each have a structure composed of a fixed number of components, as set by the modeller before initialisation, separated by the operator +. The first technology, called the primitive, is created before the simulation starts and has one component defined as a '1' with all other components defined as 'null'. Summing over a technology's structure yields its product, which is compared to the lists of peaks to determine the nearest peak. The nearest peak represents the societal need that the technology satisfies and the distance between the technology's product and that peak integer represents how well that

need is satisfied by that technology. If a technology's product exactly equals the integer of its nearest peaks, then that peak is considered to be reached or exactly satisfied.

All non-primitive technologies are created during the simulation by developer agent who redefines some of the null components as the entire structure of a previously created technology. For example, at the first time step the only existing technology is the primitive, so the developer creates the first non-primitive technology and embeds the entire structure of the primitive technology as a component in place of at least one of the null components. In this way, technologies are built recursively out of previously existing technologies and can be used as components in the construction of later technologies. The product of these non-primitive technologies still sums over the entire structure, including all the recursive levels. In this way, small structural changes can potentially produce a large changes in the product, effectively 'jumping' across the landscape to approach entirely new peaks.

There is only one rational developer agent who learns about the landscape and the technologies available for use as components and who controls the construction of new technologies. This agent begins with a (possibly inaccurate) perception of at least one of the integers on the list of peaks and a variable grasp of the products of existing technologies. The developer agent uses this knowledge to rationally develop new technologies that exactly reach recognised peaks using as few components as possible and in so doing, improve his knowledge of the peaks and the existing technologies.

Operation. The model operation consists of several basic steps performed at every time step including selecting a problem to solve, testing out possible solutions, attempting one of those solutions, and learning from the attempt. First, a list of 'peaks to aim for' is compiled of all peaks that are known to the developer agent but that have not yet been reached. Any peak that is the nearest peak to any existing technology's product is known to the developer and so will feature on the list (unless it has already been reached). At the beginning

of the simulation, the only peak known to the developer is the peak closest to the primitive technology. When reached, peaks are removed from the list and replaced by the next closest peak to ensure that the list of peaks to aim for cannot be empty unless all peaks have been reached. New peaks can also be added to the list of peaks to aim for if the developer agent creates a technology whose product overshoots the intended peak to the point that it sits closer to an unidentified peak than to the one at which it was aimed. This means that the list of peaks to aim for is a two-dimensional space with the potential to grow non-monotonically, representing the way technological 'frontiers' advance over time as new technologies open up new combinatory possibilities and create new needs [14]. After the list of peaks to aim for is created, the developer agent selects one to aim for.

Next, the developer agent uses his knowledge of the existing technologies' products, again tempered by uncertainty as set by the modeller, to find the one that seems closest to the selected peak and which does not exceed the maximum number of components. The developer agent calculates the distance between the selected peak and the product of the selected technology according to the expression:

$$Peak \pm U_{peak} - Product_{Selected} \pm U_{product}$$

where $Peak$ is the the estimated integer of the selected peak, $Product_{Selected}$ is the estimated product of the selected technology, and U_{peak} and $U_{product}$ are uncertainties for the individual peak and product in question. The developer then clones the structure of the selected technology to create a new technology and experiments with temporarily adding the structure of up to ten technologies as components according to:

$$Peak \pm U_{peak} - Contribution_{Potential} \pm U_{contribution}$$

where $Contribution_{Potential}$ is the estimated product of the new technology after the component structures have been added and $U_{contribution}$ is the associated uncertainty for those component structures. The developer agent's

goal is to get as close as possible to the selected peak using as few components as possible. If components are found that seem to reduce the distance between the selected peak and the product of the new technology, then those components are permanently added to the structure of new technology so that $Product_{Modified} = ProductSelected + Contribution_{Potential}$.

The newly created technology is then compared to the list of valley integers by summing across the components, one at a time. At no point is that sum allowed to equal a valley. For example, if '3' is a valley integer but '4' is not, combinations 2+1 and 2+1+1 would not be viable, while a combination of 2+2 would 'leapfrog' the valley and would be viable. Viable technologies are added to the set of available and existing technologies and can be later used as a selected technologies or as potential components. The actual product of viable technologies is compared to the list of peak integers to see if a peak has been exactly satisfied so that the list of peaks to aim for can be adjusted as needed for the next time step.

As the developer agent selects peaks and tests components, he gains experience of both that reduce his uncertainty and allows him to learn. The uncertainty associated with the products of each technology and of each peak location diminishes according to a standard learning curve model,

$$U = U_i(x+1)^{log_2 b}$$

where U_i is the initial uncertainty, x the number of times a technology has been used as a component or a peak is selected, and b the relevant learning percentage as set by the modeller. With this learning curve built into the model, the initial uncertainty for a peak decreases each time the developer tries to satisfy it and the initial uncertainty for the product of a technology is reduced each time the developer tries to use it as a component. The higher the learning curve is set, the more the uncertainty is reduced with each attempt or use, so that an 80% learning curve reduces the initial uncertainty faster than a 20% learning curve. Thus, after the technologies are tested for viability and the list of possible peaks is adjusted, the relevant uncertainties are reduced and the time

step ends. Although it is a simplification, the developer agent has no knowledge of or capacity to learn about about valleys. This is meant to represent the way why some endeavour has failed is often not clear until success has been achieved through other means.

The variable parameters and metrics. Many parameters in this model can vary, but will be held constant. These parameters include the maximum integer for defining the landscape, the maximum number of components that can be added in the creation of a new technology, or the maximum number of components that a technology can have. This minimises the analytical complexity so that the parameters of most interest can vary. The parameters to vary fall into three categories:

K of the landscape The density of peaks and valleys are both individually tunable to create landscapes of varying K. The lowest possible K landscapes come from low densities of both peaks and valleys and the highest K landscapes come from high densities of both. Mid-K landscapes can be created through mid-range densities for both peaks and valleys or by having a high density for one and low density for the other. Preliminary testing suggests that problem solving is poorest when valleys outnumber peaks, but one such case is included for thoroughness.

Uncertainty The location of peaks and the product of technologies are both obscured by uncertainties, which can range from zero to 100% indicating that the developer agents can be completely certain, completely uncertain, or anything in between, before he begins to act. These uncertainties can be set individually, so that the developer agent can be very uncertain about the location of peaks but far less uncertain about the product of technologies, or vice versa. Varying them individually would make interpreting the results considerably more difficult, so they will instead vary in unison across the entire range.

Learning Like the uncertainties, the rate at which experience reduces the un-

certainty of both the peaks and the product of technologies can be set individually. This would allow the developer agent to learn more quickly about either the needs or the means of reaching those needs. Also like the uncertainties, the learning rates can range from zero to 100%, meaning that the agent could be completely incapable of learning or could become totally certain after a single learning experience. Just as with uncertainty, changing the rate of one kind of learning without a similar change to the other would complicate the analysis. To avoid this, both learning rates will change in lock step. However, the entire range will not be explored as the extremes are both unrealistic and unhelpful for understanding how learning interacts with initial uncertainty and landscape complexity.

An important point to note is that setting the initial uncertainty to 0% means that there no learning is possible at all, regardless of how the learning rates are set. This means that all cases with 0% initial uncertainty are equivalent.

The metrics. Although this model uses the fitness landscape metaphor, the way that the landscape is represented means that the usual concepts of local and global optima do not apply. The peaks can all be understood as being of equal height, although the way they are clustered or scattered across the entire landscape will mean that their slopes are not equally steep. Further, landscape explorers in classic fitness landscape models begin exploring from random points on the landscape while the explorers in this model all begin at the peak closest to the primitive technology after which each success introduces the next new peak to aim for. In light of this, the developer agent's problem solving ability is better judged on how much time is required to reach the summit of all peaks rather than on the ability to reach the summit of any specific peak. Thus, one possible metric might be the number of time steps needed to create a technology that exactly satisfies all peaks in the landscape.

However, preliminary tests suggest that this metric will not work. First, the presence of valleys means that some peaks will be much more difficult to reach than others. A peak sandwiched precisely between two valleys will, for

example, be much harder to reach by virtue of not allowing any near misses. In fact, a peak and valley can even share an integer, meaning that very rarely, an identified and imperfectly satisfied peak will be impossible to precisely satisfy. This suggests that the number of peaks that are identified and approached, even if never successfully reached, could be an important alternative measure of success alongside the more obvious number of exactly reached peaks. Second, the depth of recursion involved in this model is so resource intensive that it is completely infeasible to let the model run until all peaks are identified or satisfied appears to be completely infeasible. Even with very low K landscapes, the model did not manage to identify or satisfy all peaks within 100,000 time steps. This suggests that a less resource intensive metric would be the number of peaks satisfied (exactly or inexactly) within a fixed number of time steps. The preliminary tests showed that many peaks had been satisfied by the 25,000th time step and that letting the simulations run on beyond this did not alter the number of peaks satisfied. Thus, one metric will be the number of peaks approached and/or reached within the fairly arbitrary limit of 25,000 time steps.

Next, second metric is needed to cover the efficient creation of technologies. Developer agents with accurate information about the landscape and existing technologies' products should be well placed to create new technologies that exactly reach an identified peak. Developer agents with less accurate information are more likely to create technologies that might be viable but do not exactly satisfy the identified peaks. Thus, developer agents with poorer knowledge or learning are expected to create more technologies in total and to need more time steps to exactly reach those peaks. Therefore, a second metric will be the number of viable agents created during the 25,000 time steps. This metric can be further broken down into the number of viable agents created per peak in the landscape or per peak reached and/or approached, should the number of peaks reached and/or approached be very different to the number of peaks in the landscape. The two metrics together measure problem solving completeness and efficiency, which cannot be entirely separated when a time limit is imposed.

3. Experimental parameters, expectations and results

Parameters. The experimental parameters create 42 unique experimental cases (3*2*4*2=48-6 to remove the duplicate cases with 0% initial uncertainty). Each of these cases is repeated 10 times and the results averaged. For easier interpretation, results are presented from three distinct sets of cases, each representing a landscape category. The first landscape category covers the lowest K landscapes, where peak and valley densities are both 1%. With the maximum integer set to 1000, this produces 10 peaks and 10 valleys scattered across what might be called a sparse landscape. Zero K landscapes are trivially easy to explore, so this low K landscape still has multiple peaks and valleys. The second landscape category consists of the cases with the highest K landscapes, where peak density is set to 50% and valley density is set to either 1 or 20%. As the maximum integer does not change, this produces two very rugged landscapes with 500 peaks, although one is more rugged than the other by the inclusion of 190 additional valleys. The final category consists of case with a mid-K landscape, where peak density is set to 25% and valley density set to 1% for a total of 250 peaks and only 10 valleys. Other cases could be considered as mid-K landscapes and therefore could have been included in this final set, but the selected case was both representative and interesting without overcomplicating the analysis too much.

Each landscape category contains all the different cases created by varying initial uncertainty and rate of learning. Four levels of initial uncertainty cover the entire range possible, from zero initial uncertainty to 100% initial uncertainty. Two mid-range values are included, at 40% and 80% uncertainty, to better explore the range. The extreme ends of the learning rate were not included as these were neither realistic nor interesting for addressing how the rate of learning interacts with problem solving in complex systems. Thus, there are only two rates of learning, at 20% and 80%.

Expectations. The specific question addressed in this work is whether or not additional areas of learning improves rational problem solving in all kinds of

Parameter	Value	Justification
Time steps	25,000	Motivated by preliminary testing
Runs	10	Allows results of each case to be averaged over multiple runs
Max integer	1000	A nice, round number which produces easily calculable numbers of peaks and valleys
Density of peaks	1, 25, 50	A range of densities to capture landscapes of varying K
Density of valleys	1, 20	Allows landscapes of varying K without many cases with more valleys than peaks
Maximum components in structure	50	A nice, round number
Maximum components to be added	10	A nice, round number
Uncertainty of peak locations	0, 40, 80, 100	Both extremes and some in between
Uncertainty of technology products	0, 40, 80, 100	Varies in unison with peak uncertainty
Learning curve for peak locations	20, 80	Slow and rapid learning, while avoiding extremes
Learning curve for technology products	20, 80	Varies in unison with peak learning

Table 1: The parameter settings for the experimental cases

problem landscapes. Without the ability to make or learn about making variable length steps, a low K landscapes was best approached through better knowledge or more learning while a high K landscape demanded poorer knowledge and less learning. The introduced ability to move with variable length steps and to learn about those variable steps in order to control them may resolve the conflict by allowing both efficient and complete learning in landscapes of varying K. The expectations can be operationalised as null and alternative hypotheses.

The null hypothesis: the added features will produce no change. . This hypothesis suggests that adding combinatorial evolution to replicate variable length steps and allowing the rational developer agent to know and learn about both the problems and the means by which they might be solved will not change the problem solving behaviour seen in models without these added features. In effect, this hypothesis says that problem solving will continue to be either rapid or complete, but not both, for all fitness landscapes expect those that are very low K. Specifically regarding the two metrics, the null hypothesis foresees that experimental cases with no initial uncertainty will be the most efficient at creating technologies that exactly satisfy peaks, but the worst at satisfying all of the available peaks. This hypothesis also expects that the cases with low initial uncertainty and/or rapid learning will be efficient and so will produce relatively few technologies in total, but will only be able to satisfy a majority of peaks in very low K landscapes. Experimental cases with greater initial uncertainty and/or slower learning would be expected to be less efficient, and so produce more technologies, but to satisfy most peaks in both high and low K landscapes.

The alternative hypothesis: the added features will produce a meaningful change. . This hypothesis suggests that adding combinatorial evolution to replicate variable length steps and allowing the rational developer agent to know and/or learn about both the problems and the means by which they might be solved will improve problem. This hypothesis posits that the experimental cases with no initial uncertainty should be able to take advantage of the added capacity to make variable steps and should be excellent at reaching all or almost all of the

peaks and should also do so very efficiently. This hypothesis also expects that low initial uncertainty and/or rapid learning will create a search strategy that is both reasonably efficient and reasonably complete in all landscapes, although perhaps not as good as the cases with zero initial uncertainty. Specifically for the two metrics, this hypothesis suggests that the most peaks satisfied and most efficient number of technologies per peak will be found in the case with no initial uncertainty, followed by the cases with low initial uncertainty and/or rapid learning. The experimental cases with high uncertainty and/or slow learning are expected to satisfy fewer peaks and to produce far more technologies during the process.

3.1. Results

Low K landscapes. Both hypotheses agreed on many expectations in the low K landscape, such as the relatively efficient creation of technologies in cases with lower initial uncertainty and/or rapid learning. They also both expected that the majority of peaks would be satisfied for both rates of learning, but they did not agree on the expectations for the case with zero initial uncertainty. This case, with no need or possibility for learning, met all expectations regarding the efficiency of technology creation, but only managed to satisfy a minority of the peaks. Graphing the viable technologies as a network shows a chain-like structure (See Figure 2) without any branches or dead ends. Each agent appears to be successfully cloned precisely once as the developer agent makes totally rational progress toward each peak in turn until the developer reaches a point beyond which he can do nothing. By reaching each selected peak exactly, the developer never accidentally identifies any peaks and only becomes aware of one new peak at a time. The totally efficient technology creation also means that the developer has a very limited number of technologies to clone or use as components and may be unable to leapfrog a valley. In this case, it appears that rational efficiency is his greatest strength but also his greatest weakness by preventing him from making a complete exploration of the landscape, supporting the null hypothesis.

1% peak density, 1% valley density							
Learning	None	Slow			Rapid		
Initial uncertainty	0	40	80	100	40	80	100
Tech agents	7.1	70.5	1148.1	136.7	46.9	66.2	29.5
Tech agents/ peak	.7	7.1	114.8	13.7	4.7	6.6	3.0
Peaks satisfied	4.1	3	8.6	6.2	6	8.3	6.4
Reached/approached	3.1/1.0	1.6/1.4	4.4/4.2	1.9/4.3	4.6/1.4	5.4/2.9	4.3/2.1
Tech agents/ satisfied peak	1.7	23.5	133.5	22	7.8	8.0	4.6

Table 2: Mean values for the lowest possible K landscape with 10 peaks and 10 valleys.

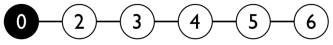

Figure 2: A typical network created in a low K landscape with zero initial uncertainty. The primitive agent is marked in black and all agents are labelled with their who number, indicating the order in which they were created.

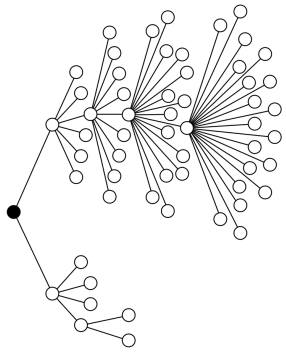

Figure 3: A typical network created in a low K landscape with non-zero initial uncertainty, with the primitive agent marked in black.

With some initial uncertainty, the developer agent made mistakes that lead to discovering additional peaks accidentally. As the learning reduces his uncertainty about identified peaks and existing technologies, he becomes aware of new peaks to be uncertain about. Thus, the mistakes allow the number of things about which he is uncertain to grow faster than the existing uncertainty reduces. In this low K landscape, there are very few peaks about which the developer agent can be uncertain, so it is perhaps not surprising that all but one of the experimental cases with rapid learning reached more peaks than the equivalent cases with slow learning. Furthermore, the cases with rapid learning created far fewer technologies, illustrating how a well informed or rapidly learning rational developer can apply his superior knowledge to reach the majority of peaks while also being efficient. For this landscape, rapid learning appear to better balance the growth in uncertainty against the reductions in uncertainty. Graphing the viable technologies as a network reveals a more organic, branching structure with some dead ends and with some technologies cloned multiple times (See Figure 3). The latter technologies are cloned very many times, although this is probably a modelling artefact stemming from the way that technologies can only be cloned if they have fewer than the maximum number of components. The overall shape of these networks does not change as uncertainty rises or learning slows, although the networks do grow more crowded, especially as the simulation hits the maximum number of components.

There was only one exception to the superior problem solving show by rapid learning in a low K landscape. Slow learning combined with 80% initial uncertainty narrowly edged out the equivalent case with rapid learning case by satisfying .3 more peaks. However, this success was achieved by approaching rather than exactly reaching the satisfied peaks and also showed the least efficient technology creation rates of all cases in the low K landscape. Both rates of learning show that the most complete exploration coincides with the least efficient technology creation, further supporting the null hypothesis. Interestingly, both learning rates also show that the most complete exploration occurs at 80% initial uncertainty rather than at the lowest non-zero level or at the highest

50% peak density, 1% valley density							
Learning	None	Slow			Rapid		
Initial uncertainty	0	40	80	100	40	80	100
Tech agents	38.6	3432.7	4663.6	2050.4	1480.1	1455.2	1013.4
Tech agents/ peak	0.1	13.7	9.3	4.1	3.0	2.9	2.0
Peaks satisfied	37.9	497.1	495.9	492.4	448.3	498.1	496.2
Reached/approached	36.9/1.0	490.7/6.4	489.5/6.4	483.5/8.9	446.4/1.9	495.7/2.4	493.4/2.8
Tech agents/ satisfied peak	1.0	6.9	94.1	4.2	3.3	2.9	2.0
50% peak density, 20% valley density							
Tech agents created	2.7	931.7	2195.8	1790.8	138.6	737.8	659.7
Tech agents/ peak	0.0	1.9	4.4	3.6	0.3	1.5	1.3
Peaks satisfied	2.8	136.4	272	362.8	47.8	271.9	319.6
Reached/approached	1.8/1.0	118.5/17.9	240.4/31.6	317.0/45.8	41.6/6.2	238.7/33.2	278.4/41.2
Tech agents/ satisfied peak	1.0	6.8	8.1	4.9	2.9	2.7	2.1

Table 3: Mean values for the two highest K landscapes with 500 peaks each and 10 or 200 valleys, respectively.

possible level, suggesting that there is a 'sweet spot' for how much initial uncertainty is most useful for beginning an exploration of a low K landscape. This sweet spot could be related to the balance between the growth of uncertainty through mistakes and the reduction of uncertainty through learning.

High K landscapes. The two hypotheses disagree about almost all of the expected behaviour in higher K landscapes, and so should prove more useful for accepting or rejecting the hypotheses. Regrettably, the networks created by graphing the technologies in these landscapes are so dense as to be totally impenetrable, so the results will only consist of a table full of the relevant metrics.

Matching the results of the low K landscape, the experimental cases with 0% initial uncertainty satisfied very few peaks but did so with very few superfluous

agents. The results were particularly dire for the highest possible K landscape with 200 valleys, where a mere 2.8 peaks were satisfied on average, further supporting the null hypothesis.

Also as with the low K landscape, more technologies are created and more peaks satisfied as initial uncertainty rises for both learning rates. When there are 500 peaks but only 10 valleys, rapid learning has the advantage by virtue of satisfying slightly more peaks for most initial uncertainties. The cases with rapid learning also created far fewer technologies than the cases with slow learning. This appears to give some support to the alternative hypothesis in that a rational developer agent can solve problems both completely and efficiently.

Interestingly, for both rates of learning the most complete exploration of this 500 peak and 10 valley landscape did not coincide with the most efficient technology creation. Further, the case with the lowest non-zero initial uncertainty was the worst of the cases with rapid learning but the best of the cases with slow learning. Although these results suggest that high K landscapes can be explored completely and efficiently, the results do not show a clear correlation between completeness and efficiency or between lower initial uncertainty, rate of learning and better problem solving, nor do they show a clear sweet spot. It would seem that there are multiple ways to balance uncertainty growth through making mistakes against uncertainty reduction through learning.

However, what stopped rational problem solvers from completely exploring high K landscapes in previous models was not the number of peaks to explore but the valleys that impeded their rational exploration. Thus, the real test is the highest possible K landscape, with 500 peaks and 200 valleys. In this landscape, slow learning satisfied more peaks than the equivalent cases with rapid learning. Both learning rates were more complete and more efficient in the cases with 100% initial uncertainty, suggesting that the increased number of valleys are better dealt with through greater uncertainty. Thus, the results from the highest possible K landscape support the null hypothesis by showing that more uncertainty rather than less leads to better landscape exploration. However, these same results provide some support for the alternative hypothesis by

25% peak density, 1% valley density							
Learning	None	Slow			Rapid		
Initial uncertainty	0	40	80	100	40	80	100
Tech agents created	26.2	3401.7	4097.2	1711.9	1112.6	1046.6	752.7
Tech agents/ peak	.1	13.6	16.4	6.8	4.5	4.2	3.0
Peaks satisfied	25.5	249.3	249	223.8	226	225.3	235.7
Reached/approached	24.4/1.0	243.0/6.3	240.1/8.9	214.6/9.2	223.8/2.2	223.1/2.2	231.3/4.4
Tech agents/ satisfied peak	1.1	13.7	16.5	7.6	4.9	9.1	3.2

Table 4:

suggesting that some experimental parameters improve both completeness and efficiency in very high K landscape exploration. These results also suggest that the balance between uncertainty growth and uncertainty reduction is influenced by the presence of barriers to problem solving as well as by the presence of additional problems to solve.

Mid-range landscape. Finally, the disagreements between the hypotheses centred around the expectations for high K landscapes, but they can be applied to mid-range and other non-low K landscapes as well. As with the other landscapes, the cases with zero initial uncertainty satisfied few peaks, but did so efficiently, further supporting the null hypothesis. The cases with non-zero initial uncertainty are much more interesting.

As with the highest possible K landscape, the cases with rapid learning efficiently satisfied the majority of peaks, but the cases with slow learning satisfied more peaks, albeit less efficiently. These results generally support the null hypothesis, although the cases with the most complete exploration are sometimes also the most efficient. Also mimicking the results of the highest K landscape, slow learning was best paired with low initial uncertainty while rapid learning was best paired with high initial uncertainty. This further supports the idea that the sweet spot, where uncertainty growth is balanced against uncertainty

reduction, depends on the level of initial uncertainty, the learning rate, and the features of the problem solving landscape.

4. Conclusions

Most of the results seem to either contradict the expectations of the alternative hypothesis or confirm the expectations of the null hypothesis. First, the very incomplete landscape exploration of the cases with zero initial uncertainty suggests that at least some initial uncertainty is required for effective problem solving in non-zero K landscapes. Complete certainty from the get-go prevents mistakes, but mistakes turn out to be beneficial. They lead to the discovery of alternative problems, which allows uncertainty to grow, and also produce imperfect technologies that expand the possibilities for leapfrogging valleys. Even the lowest K landscape was best explored by a surprisingly high 80% initial uncertainty while some combinations of rapid learning and higher K landscapes saw the best results from 100% initial uncertainty. This suggests that a complete exploration requires approaching certainty rather than starting with certainty.

Reducing uncertainty through learning kick starts a feedback loop by which the learner becomes better placed to further reduce his uncertainty. On the other hand, the initial uncertainty opens up the possibility for uncertainty to grow by discovering entirely new problems. This too starts a feedback loop, with the added uncertainty increasing the likelihood of discovering new problem that add yet uncertainty. However, every growth in uncertainty entails at least some reduction in uncertainty. Eventually, the uncertainty reduction overtakes the uncertainty growth, strengthening one feedback loop while weakening the other. Thus, not only does a complete exploration require some initial uncertainty, it also requires the capacity for uncertainty growth to temporarily outstrip uncertainty reduction.

Second, adding the capacity to learn about ways to move across a fitness landscape to the capacity to learn about that landscape did not resolve the conflict between the benefits of more knowledge, learning and rationality and

the benefit of less knowledge, learning and rationality. Complete landscape exploration demands mistakes and uncertainty growth, which necessarily entail inefficiency. Thus, no matter how much capacity for learning is added to a rational explorer, complex landscapes must still be explored either completely or efficiently. However, the trade off between completeness and efficiency is not simple. Zero initial uncertainty produced both the least complete exploration and the most efficient technology creation, but the cases with the most complete exploration did not always coincide with the particularly inefficient technology creation. Mistakes are not guaranteed to be useful in any way, so problem solvers that choose to prioritise exploratory, speculative landscape searches over efficient ones should be aware that inefficiency does not guarantee problem solving completeness.

Third, slower learning was as complete or more complete at landscape exploration than rapid learning for almost all K landscapes and was especially superior when the landscape included many valleys as well as many peaks. Although initially counter-intuitive, slow learning seems to provide a better balance between uncertainty growth and uncertainty reduction, particularly when the problems to address can reasonably be expected to be complex and to contain many barriers to progress. Rapid learning paired with high or total initial uncertainty produced nearly as good a balance between uncertainty growth and uncertainty reduction, especially if efficiency is a priority. Thus, choosing an appropriate approach to problem solving is not simply a matter of weighing urgency against the perceived landscape, but should also consider how much is already known.

Unfortunately, it is not clear how to determine the complexity of a problem, how many barriers lie in the way of the solution, or how much is really known about that problem before setting out to solve that problem, making the insights gained through this model difficult to apply. Additionally, there are many aspects of this model that could be changed and which might produce entirely different results. For example:

Dynamic landscape The NK model implies that agents solve exogenous problems in a fixed solution space and in a static environment, but allowing the landscape to shift or grow would mean that previously learnt information would go out of date. This would allow the developer to continue making mistakes, potentially resolving the conflict between completeness and efficiency.

Multiple, communicating developers Multiple problem solvers might be able to work on different problems and to communicate their successes and mistakes. This could reduce the chance that developer agent get permanently stuck on one problem and could improve both problem solving completeness and efficiency.

Generational turnover Mistakes reduced as learning moved the developer toward total certainty, but new, mistake-prone developers could be introduced to offset the removal of developers that have lost the ability to make mistakes. This could be especially useful in a dynamic landscape as generational turnover is key to the emergence of highly structured, consistent and adaptive languages among simulated agents [15].

The proposed changes might show the conflict between completeness and efficiency in a new light, but are don't seem like to change the way individual learning gradually makes the learner more conservative and averse to making risks. Further, these proposed changes appear to push learning and rational behaviour away from the individual and back up to the abstract, population level where it began with totally irrational and learning-free agents finding optimal solutions by wandering randomly in a landscape under the influence of selection pressure.

5. Acknowledgements

The research has been supported by the European Regional Development Fund and the Duurzame Greenport Westland-Oostland Task Force.

6. References

[1] J. M. Korhonen, J. Kasmire, Adder: a new model for simulating the evolution of technology, with observations on why perfectly knowledgeable agents cannot launch technological revolutions (Innovation and Competitiveness: Dynamics of Organizations, Industries, Systems and Regions).

[2] D. L. Schacter, D. T. Gilbert, D. M. Wegner, Introducing psychology, Macmillan, 2009.

[3] T. Knudsen, D. A. Levinthal, Two faces of search: Alternative generation and alternative evaluation, Organization Science 18 (1) (2007) 39–54.

[4] S. A. Kauffman, E. D. Weinberger, The NK model of rugged fitness landscapes and its application to maturation of the immune response, Journal of theoretical biology 141 (2) (1989) 211–245.

[5] M. R. Smith, L. Marx, Does technology drive history?: The dilemma of technological determinism, MIT Press, 1994.

[6] H. A. Simon, Rationality in psychology and economics, Journal of Business (1986) S209–S224.

[7] H. Simon, Models of Bounded Rationality, MIT Press, 1982.

[8] B. Schwartz, The tyranny of choice, SCIENTIFIC AMERICAN-AMERICAN EDITION- 290 (4) (2004) 70–75.

[9] J. Grin, J. Rotmans, J. Schot, Transitions to sustainable development, New Directions in the Study of Long Term Tansformative Change, New York.

[10] R. R. Nelson, Evolutionary Theories of Cultural Change: An Empirical Perspective, Tech. rep. (2004).

[11] A. Pope, An essay on criticism, Clarendon Press, 1909.

[12] W. B. Arthur, W. Polak, The evolution of technology within a simple computer model, Complexity 11 (5) (2006) 23–31.

[13] J. Kasmire, J. M. Korhonen, I. Nikolic, How radical is a radical innovation? An outline for a computational approach, Energy Procedia 20 (2012) 346–353.

[14] W. B. Arthur, The nature of technology: What it is and how it evolves, Simon and Schuster, 2009.

[15] S. Kirby, Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity, Evolutionary Computation, IEEE Transactions on 5 (2) (2001) 102–110.