Pursuing Social Progress: The Question of Orientation
Technical Appendix

Contents

Note: A replication package, including NetLogo code and simulation data, is available from the authors upon request.

1 State Space Generation Procedure

Our model represents the universe of possible social states as a 33×33 torus consisting of 1089 unique patches whose locations are specified by integer valued x and y coordinates in the interval [0,32]. To represent the all things considered value of social states each patch is also associated with a *pheight* that takes on a real numbered value determined by the stochastic terrain generation algorithm described below. The toroidal structure of the state space is a standard assumption in agent-based modeling, used to eliminate boundary effects and capture the idea that the state space is unbounded. The dimensions, 33×33, were chosen so that the universe of possible social states is large enough to exhibit a wide variety of "topographies" while still being computationally tractable. The toroidal space can alternately be characterized as a 33×33 wrapped grid overlaid with "hills." Characterized this way, the height of the hills (and every other patch in the space) represents the all-things-considered value of a state from the perspective of the agents exploring the space, and agents traveling off one edge reappear on the opposite edge.

(It is natural, although not necessary, to think of the state space as a "landscape" or "terrain." We use this metaphor to simplify our exposition.)

1.1 Initial Parameters

The landscape generation process begins by setting the *hillcount* parameter, which determines the maximum number of peaks to be distributed across the terrain. This value is exogenously specified and controls the complexity of the resulting optimization landscape. Each simulation trial generates a unique terrain configuration through the following stochastic process.

1.2 Baseline Terrain Initialization

Prior to hill generation, the algorithm establishes a baseline terrain by assigning each patch a small initial height $pheight_{initial} = 1 + \varepsilon$, where $\varepsilon \sim U(0, 0.01)$. This baseline stabilizes the process of calculating performance metrics by ensuring that the metrics can be easily normalized to the height of any patch on the landscape.

1.3 Gaussian Hill Generation

Landscape construction then proceeds by iteratively generating Gaussian hills on the landscape, where the number of hills is determined by the *hillcount* parameter described above. Visually, a Gaussian hill has the shape of a classic bell curve. The size and location of these hills is stochastically determined by the algorithm, and the hills can (and often do) overlap so that the ultimate height of each patch on the landscape is determined by adding the contributions of each hill covering a patch to the initial height of the patch. More specifically, the hill generation procedure has the following steps:

Step 1: Peak Location Assignment. For each hill $i \in \{1, 2, \ldots, hillcount\}$, the algorithm independently draws random coordinates:

$hill\text{-}x \sim U(0, world\text{-}width)$
$hill\text{-}y \sim U(0, world\text{-}height)$
These coordinates (*hill-x*, *hill-y*) serve as the center point for each Gaussian function.

Step 2: Hill Parameter Generation. Each hill is characterized by two stochastic parameters:
$hill\text{-}amplitude \sim U(1, 26)$
$hill\text{-}spread \sim U(1, 5)$
The amplitude parameter determines the height of the peak of each hill (and so its vertical scale), while the spread parameter determines the spatial extent and steepness of the Gaussian function by governing how height decreases as a function of distance from the peak. Together the two parameters dictate how much the hill contributes to the heights of each patch encompassed by the hill.

Step 3: Height Calculation. For each patch at coordinates (*pxcor*, *pycor*), the algorithm calculates the height contribution from hill *i* using the Gaussian function:

$$height\_contribution_i = amplitude_i \times \exp\left(-0.5 \times \left(\frac{distance_i}{spread_i}\right)^2\right) \tag{1}$$

where

$$distance_i = \sqrt{(\min(|hill\text{-}x_i - pxcor|, 32 - |hill\text{-}x_i - pxcor|))^2 + (\min(|hill\text{-}y_i - pycor|, 32 - |hill\text{-}y_i - pycor|))^2}$$

which represents the Euclidean distance measured along the torus surface from the patch to the center of the hill.

Step 4: Additive Height Assignment. The final height of each patch is computed as the sum of contributions from all hills plus the baseline height:

$$pheight = pheight_{initial} + \sum_{i=1}^{hillcount} height\_contribution\_i \tag{2}$$

This additive approach ensures that overlapping hills create higher peaks where their Gaussian functions intersect, while maintaining the smooth character of the terrain. Specifically, hill construction procedure ensures C^∞ continuity across the terrain, with no discontinuous jumps between adjacent patches. This property reflects the assumption that transitions from one social state to another involve gradual rather than dramatic normative changes. (We take this as a starting point; future research could relax this assumption.) Note, however, that the additive procedure for generating the landscape that iteratively layers hills on top of one another entails that the hillcount parameter does not provide a measure of the number of peaks on a landscape. Instead, it provides an upper bound on that number, while the actual number of peaks on a landscape is a function of the location and size of the hills that are layered on the landscape. For instance, when hills with relatively small amplitudes and/or large spreads are located near hills with relatively larger

amplitudes and smaller spreads, the contribution of the taller/steeper hill will typically swamp the contribution of the shorter/broader hill so that the peak of the shorter/broader hill no longer stands out on the landscape.

1.4 Global Optimum Enforcement

Finally, to guarantee the existence of a unique global maximum, the algorithm implements a post-processing step. After all hills are generated, it identifies the patch (or patches) with the maximum *pheight* using the *max-one-of* operation. If multiple patches share the maximum height the algorithm selects one arbitrarily and increases its height by 1 unit. This ensures a single, well-defined global optimum that serves as the "ideal" in the simulation framework.

1.5 Infeasibility Extension

In the infeasibility extension of the model, we operationalize the idea that there are some social states that might be unrealizable, or that are otherwise inaccessible for agents, by removing some patches from the landscape. The procedure operates as follows:

Step 1: Parameter Selection. First, specify a value for the parameter *percent-infeasible* that determines the percentage of the landscape that will be rendered inaccessible, then calculate the number of patches to be marked infeasible/removed from the landscape as follows: $\textit{num-blocked} = (\textit{percent-infeasible}/100) \times \textit{total-patches}$

Step 2: Patch Assignment. Randomly select *num-blocked* patches using uniform sampling without replacement

Step 3: Height Adjustment. Set the *pheight* of *infeasible-patches* to -10 and visually identify them by setting the color of the patch to black.[1]

2 Agent Movement

Below is a description of how our NetLogo code operationalizes the movement rules for the four agent types described in the text.

[1] Note that the decision to reset the *pheight* of infeasible patches to -10 has no significance for the analysis or interpretation of the model described here. This is because the agents in our model neither start on nor enter infeasible patches (nor do the heights of those patches bear on their movement in other ways). The mechanism for changing the *pheight* of infeasible patches was simply introduced to the algorithm to facilitate possible extensions of the model in which infeasibility functionally impacts the value of a patch.

2.1 Ascenders

These are the agents that we think of as operationalizing the problem-solving approach to pursuing progress. They implement a strategy that our NetLogo code labels *stepwise-ascent* that we describe in the text as simply "Ascent", which can be alternately characterized as myopic hill-climbing, .

Technical Details:

Search Space: All 8 neighboring patches

Exclusion Criteria: Ignore infeasible patches (i.e., *set neighbor-patches neighbors with [infeasible = false]*)

Movement Rule: Identify the non-excluded neighboring patch with the best-fitness (*max-one-of neighbor-patches [pheight]*). If that patch is better than the *current-patch*, move there (i.e., if *best-fitness* $\geq$ *old-fitness*).

Termination: If no neighboring patches are better than the current patch, stop.

2.2 ProxMax (Pros)

These are agents that operationalize a naïve ideal-oriented approach to pursuing progress. They implement a strategy we call "ProxMax" that directs the agent to myopically pursue the ideal by moving directly towards the patch that is the global maximum with every step. Note that, in our code and dataset, we used the label "Pros" to refer to this type of agent (hence, its appearance in this appendix).[2]

Technical Details:

Search Space: *Face ideal-patch* (i.e., orient towards the ideal). Consider *current-patch* and *patch-ahead 1*, where *patch-ahead 1* is the neighboring patch that now lies directly ahead (i.e., directly on the path to the ideal).

Movement Rule: If *patch-here = ideal patch*, stop. Otherwise, *face ideal-patch*, and if *[infeasible] of patch-ahead 1 = false*, then move forward to that patch. Repeat. If *[infeasible] of patch-ahead 1 = true*, then stop. In other words, if current patch is not the ideal, and the neighboring patch on path to the ideal is feasible, then move there.

Termination: Stop when the *ideal-patch* is reached, or when facing *ideal-patch* and *patch-ahead 1* is infeasible.

---

[2] As we explain in the main text, we think this strategy is too myopic to fairly capture what ideal theorists have in mind, so we provide minimal discussion of the strategy there. Because we collected data about the strategy and its analysis is still useful as a point of comparison, though, we describe the strategy here and the data tables and summary statistics included with this appendix include this data.

2.3 Flexible ProxMax (FPM or FlexPros)

These are the agents that operationalize a more flexible and intuitively plausible version of an ideal-oriented strategy for pursuing progress. They implement a strategy that our NetLogo code labels *flexproxmax* that we refer to in the text as "Flexible ProxMax" (FPM) that balances ideal orientation with local optimization by continually orienting themselves towards the ideal and moving to the best neighboring patch within an ideal-oriented cone. Note that, in our code and data, we used the label "FlexPro" to refer to this type of agent (hence, its appearance in this appendix).

Technical Details:

Search Space: *Face ideal-patch*. Consider *current-patch* and the *optionset* of neighboring patches that is defined by looking at *patch-ahead 1* (where this is the neighboring patch that lies directly on the path to the ideal), *patch-left-and-ahead 45 1* (where this is the neighboring patch that lies 45 degrees to the left of the direct path to the ideal-patch), and *patch-right-and-ahead 45 1* (where this the neighboring patch that lies 45 degrees to the right of the direct path to the ideal-patch).[3]

Exclusion Criteria: *Set optionset with [infeasible = false]*, (i.e., remove infeasible patches from the *optionset*).

Movement Rule: If *patch-here = ideal patch*, stop. Otherwise, *move-to max-one-of options [pheight]*, (i.e., move to the (feasible) patch in the *optionset* that has the maximum *pheight*). Repeat. If the *optionset* is empty (e.g., because all patches in the ideal-oriented cone are infeasible), stop.

Termination: Stop when the *ideal-patch* is reached, or when no feasible patches exist in *optionset*.

2.4 Random (Randos)

These are agents that use a stochastic exploration strategy with probabilistic termination. We include as a baseline to show that some strategy is better than none. (We use the label "randos" in our code and data to refer to this type of agent.)

Technical Details:

Search Space: All 8 neighboring patches

[3]Note that because the landscape consists of discrete patches, while orientation on the landscape can take on continuous values within an agent's 360-degree field of view, there will be cases where *patch-ahead 1* will be identical to one of *patch-left-and-ahead 45* or *patch-right-and-ahead 45*. In these cases, the *optionset* will only contain two patches. This aspect of how the strategy is defined has some implications for how often FlexPros climb down and how often they get stuck short of the ideal, but it does not substantively impact the nature of the lottery that FlexPros confront (nor does it substantially impact how they fare relative to Ascenders). Of course, the optionset that FlexPros consider could have been defined differently (e.g., it could be defined to include all neighboring patches that lie within the 90 degree ideal oriented cone, or to include the three neighboring patches closest to the ideal), but note that none of those other methods of defining the *optionset* are obviously more plausible ways of operationalizing the notion of flexible ideal oriented search.

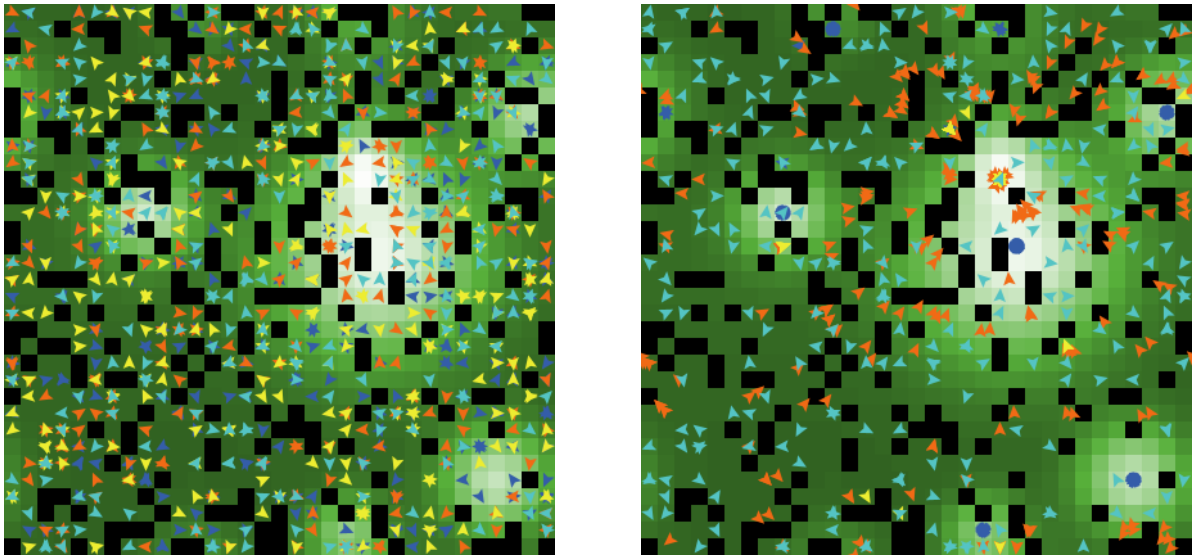
Exclusion Criterion: Ignore infeasible patches (i.e., *set neighbor-patches neighbors with [infeasible = false]*)

Movement Rule: If *patch-here* = *maxpeak*, stop. Otherwise, with 90% probability randomly move to a patch that is uniformly drawn from the feasible neighbors, and with 10% probability stop (i.e., if *random 100 > 90 move-to one-of neighbor-patches*).

Termination: Stop when *max-peak* is reached or when there are no feasible neighbors, otherwise 10% probability of stopping each step.

3 Data Collection

To analyze the model, data was collected using a BehaviorSpace simulation consisting of 70,000 trials. In each trial or "run," a new landscape was generated using the procedure described above, and 250 agents of each type were randomly placed at (feasible) patches on the landscape. 500 runs were performed for each combination of *hillcount* $\in \{1,3,5,\ldots,19\}$ and *percent-infeasible* $\in \{0,3,6,\ldots,39\}$. The figures below provide a representative picture of what a landscape looks like both before and after a run (in this case the landscape is parameterized with *hillcount* $= 11$ and *percent-infeasible* $= 21$). In the model visualization *pheight* is illustrated with colors ranging from dark green for low lying areas to white for the highest areas. Infeasible patches are black. Ascenders are blue, FlexPros are yellow, ProxMax agents are orange, and Randos are teal.



With Agents in Initial Locations | After Agents Conclude Search

Figure 1. Sample Maps

For each run, the performance of agent-types was measured by collecting the mean values and

IQRs for the following metrics averaged across the 250 agents of each type. These metrics include those we describe in the main text, as well as several others that we do not discuss there, but whose values are recorded in the dataset included in our replication package. The list below includes in bold the label of the relevant metric as it appears in the NetLogo code and data spreadsheet, along with a brief description of what the metric describes, and in italics (where applicable) the label we've given the metric in the main text.

max-peak-height The *pheight* of the global maximum patch on a landscape

optimum-percent — *Percent Ideal (PI)* Percent of agents reaching the global maximum (i.e. the patch with $pheight = max-peak-height$)

end-below-start-pct — *Percent Stop Below (PSB)* Percent of agents that conclude search at a patch with a lower height than the height at which it started (i.e. where $pheight_{end} < pheight_{start}$)

avg-endpoint-normalized — *Expected Stop Value (ESV)* Average *pheight* of agents at the end of their search normalized by the max-peak-height

avg-pathlength — *Expected Path Length* Average number of steps taken by agents during their search process

avg-pathvalue-normalized-max — *Expected Path Value (EPAV)* Average *pheight* of all patches visited during an agent's path normalized by the max-peak-height

avg-pathvalue-normalized-start Average *pheight* of all patches visited during an agent's path normalized by the $pheight_{start}$ of the patch at which the individual agent started their search

avg-increase-normalized Average increase in *pheight* from start to end position normalized by each agent's starting value (i.e., $(pheight_{end} - pheight_{start})/pheight_{start}$)

avg-shortfall-normalized Average shortfall from the global maximum of the ending position normalized by the max-peak-height (i.e., $(max\text{-}peak\text{-}height - pheight_{end})/max\text{-}peak\text{-}height$)

avg-climb-down-percent — *Expected Percent Climb Down (EPCD)* Average percentage of steps in an agent's path where the agent moved to a patch with a *pheight* than the previous patch it occupied

avg-path-below-start-percent — *Expected Percent Path Sacrifice (EPPS)* Average percentage of steps in an agent's path where the agent was at a patch with lower height than its starting position (i.e., $\sum_{i=start}^{i=end} pheight_{i+1} < pheight_i$)

startmean Average $pheight_{start}$ of agents on a landscape

start-norm Average $pheight_{start}$ of agents on a landscape normalized by the max-peak-height

max-pathlength Maximum number of steps taken by any agent of a given type for a given simulation run

As we indicated above, in addition to reporting the mean value for each of the performance metrics just described, we also report the inter-quartile range (IQR) of those metrics (where applicable)[4] for the 250 agents of each type on a given landscape. This provides a measure of the within-run variability of the metrics. We chose to report IQRs rather than alternative measures of variation, such as standard deviation, because many of the landscapes generate data that is not normally distributed and that is often censored by the normalized metrics. Because most of the metrics are normalized the IQRs are also relatively easy to interpret (e.g. an IQR of .3 indicates that for the middle 50% of agents the metric in question ranges over 30% of the possible values it can take on). In addition to the within-run IQRs, our summary statistics tables also report the average of IQRs across runs. This provides a measure of the average within-run variability for each metric under the specified landscape parameters.

4 Summary Statistics and Plots

This section presents box-and-whisker plots to illustrate how agents performance with respect to various metrics depends on the *hillcount* and *infeasibility* parameters.

4.1 Baseline Experiment

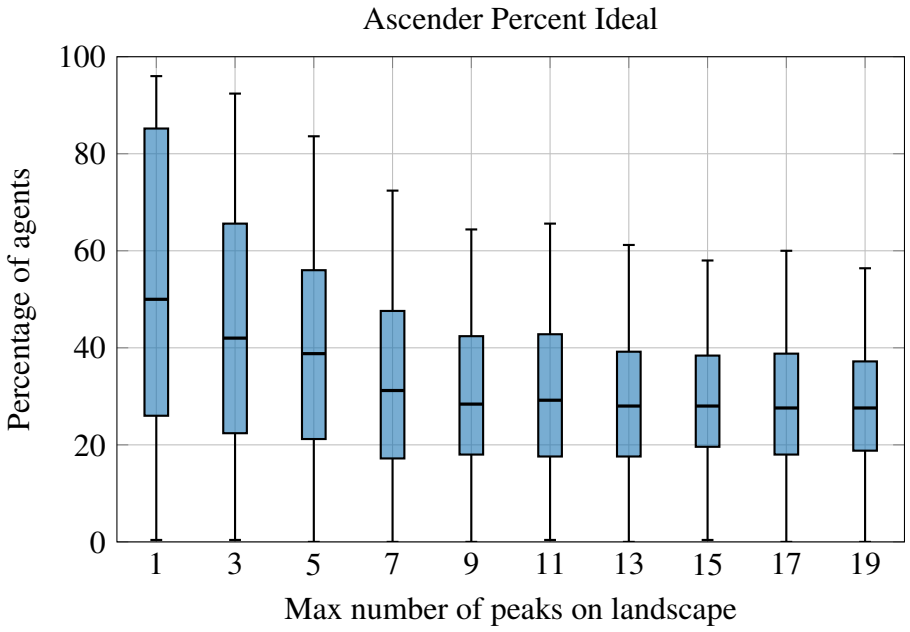


Figure 2. Ascender Percent-Ideal interacted with hillcount

[4]We do not report IQRs for Percent Ideal or Percent Stop Below because these metrics simply report the percentage of agents in a given run that respectively reach the ideal or stop below their starting point. The box-plots for these metrics that are presented below illustrate the between-run IQR for these metrics.

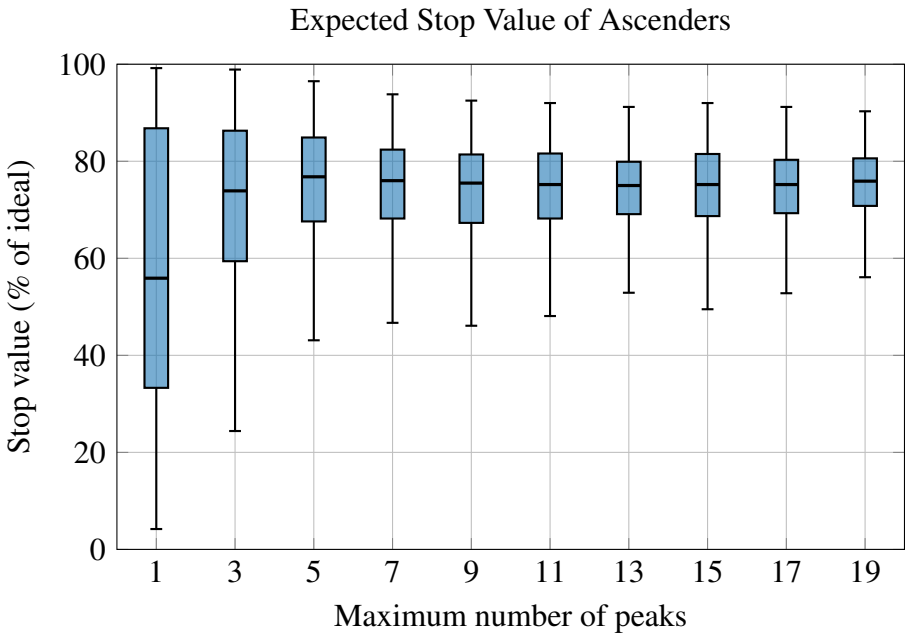


Figure 3. Ascender Expected Stop Value interacted with hillcount

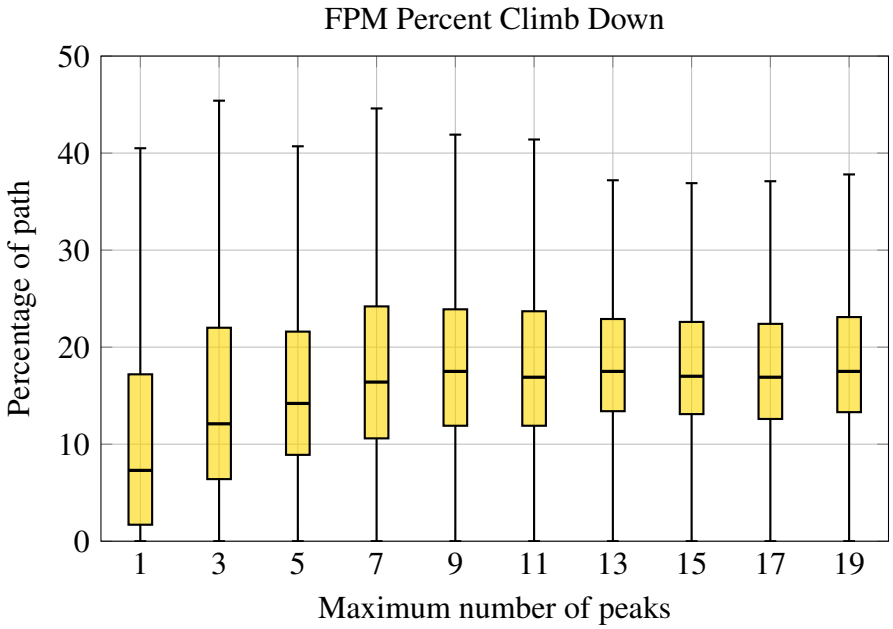


Figure 4. Percent Climb Down of FPM agents interacted with hillcount

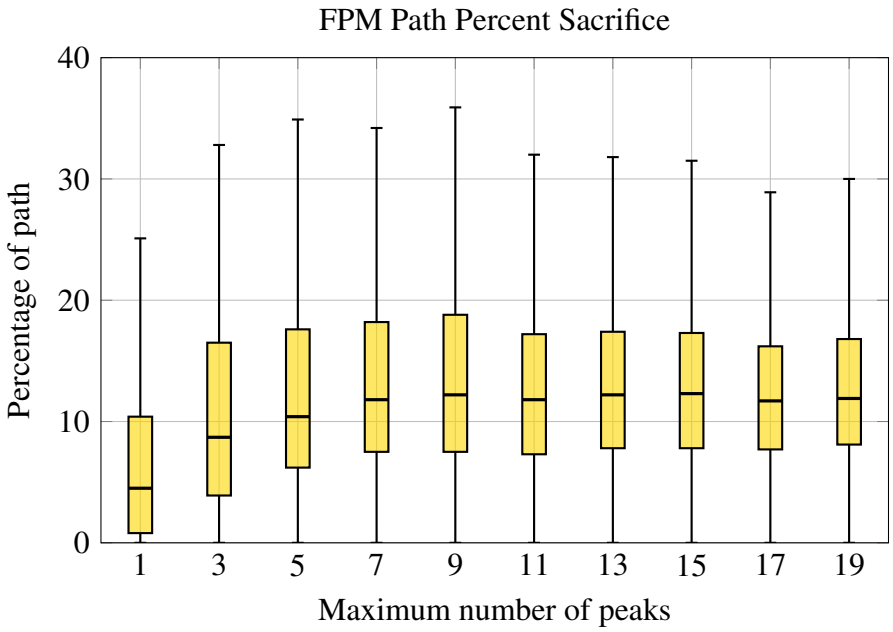


Figure 5. Percent Path Sacrifice of FPM agents interacted with hillcount

4.2 Infeasibility Extension

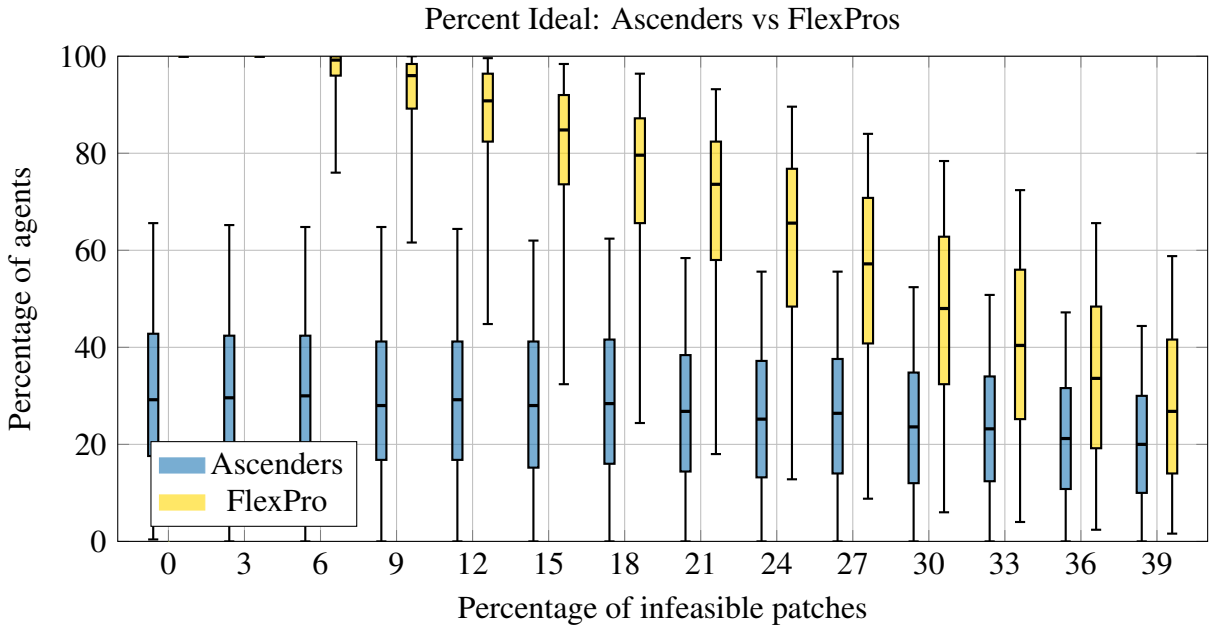


Figure 6. Percent-Ideal for Ascender and FlexPro agents interacted with percent-infeasible

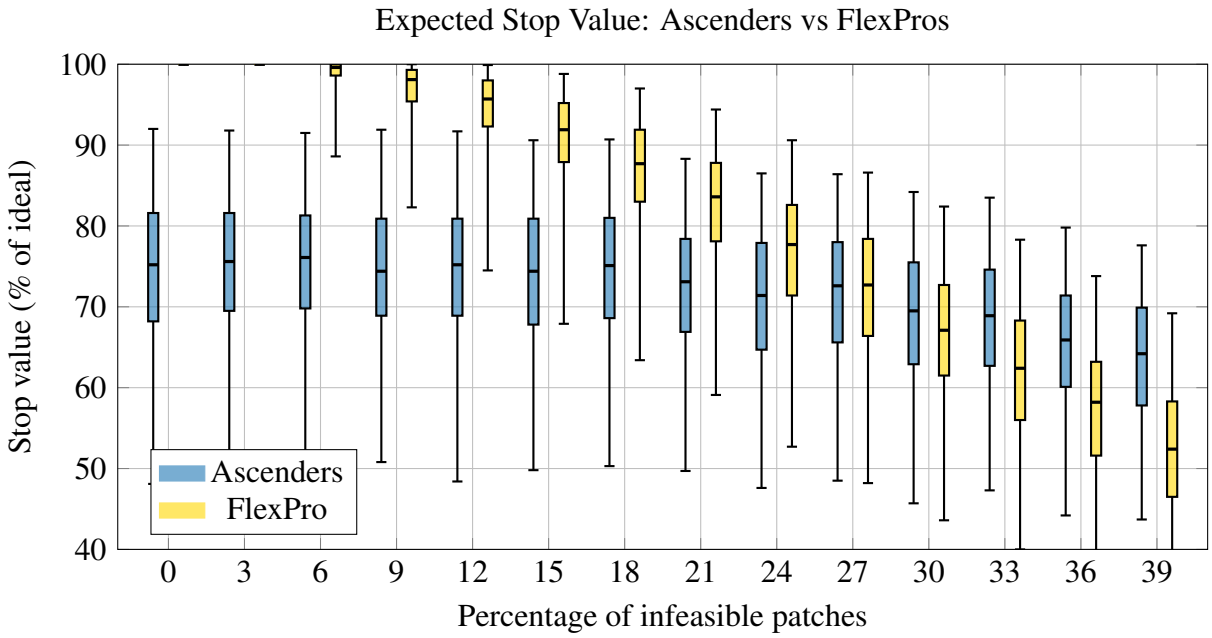


Figure 7. Expected Stop Value for Ascenders and FlexPros interacted with percent-infeasible

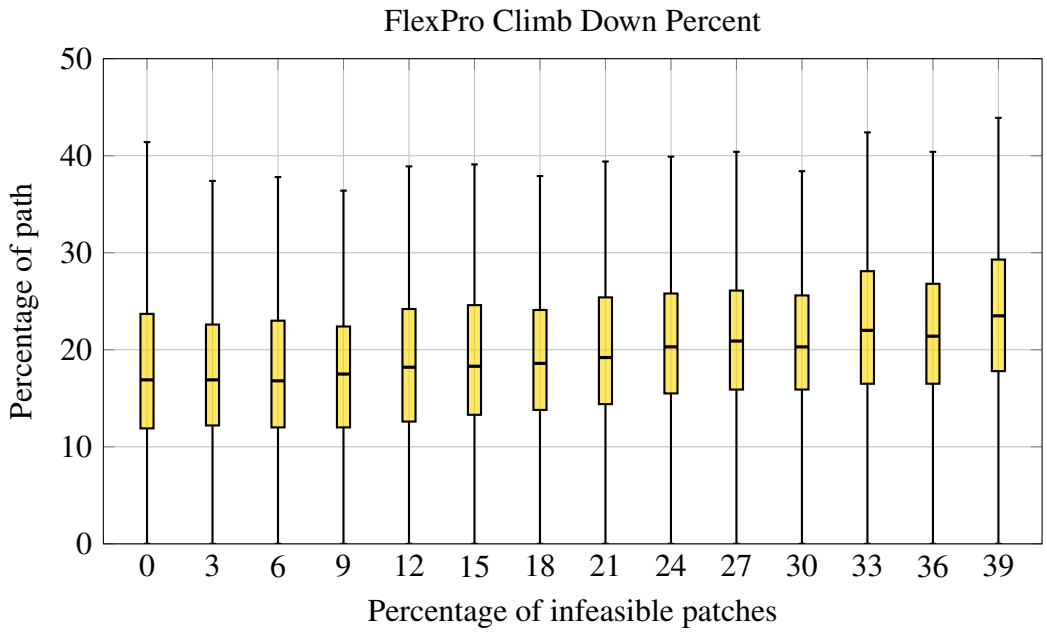


Figure 8. Percent Climb Down of FlexPro paths interacted with percent-infeasible

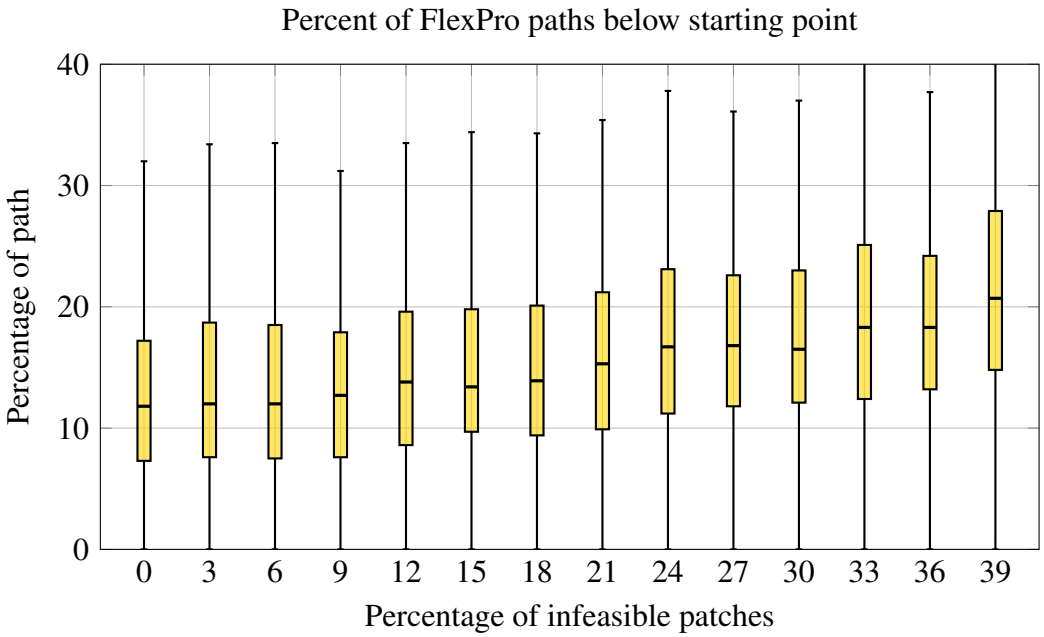


Figure 9. Percent Path Sacrifice of FlexPro interacted with percent-infeasible

4.3 FlexPro vs. ProxMax Comparison

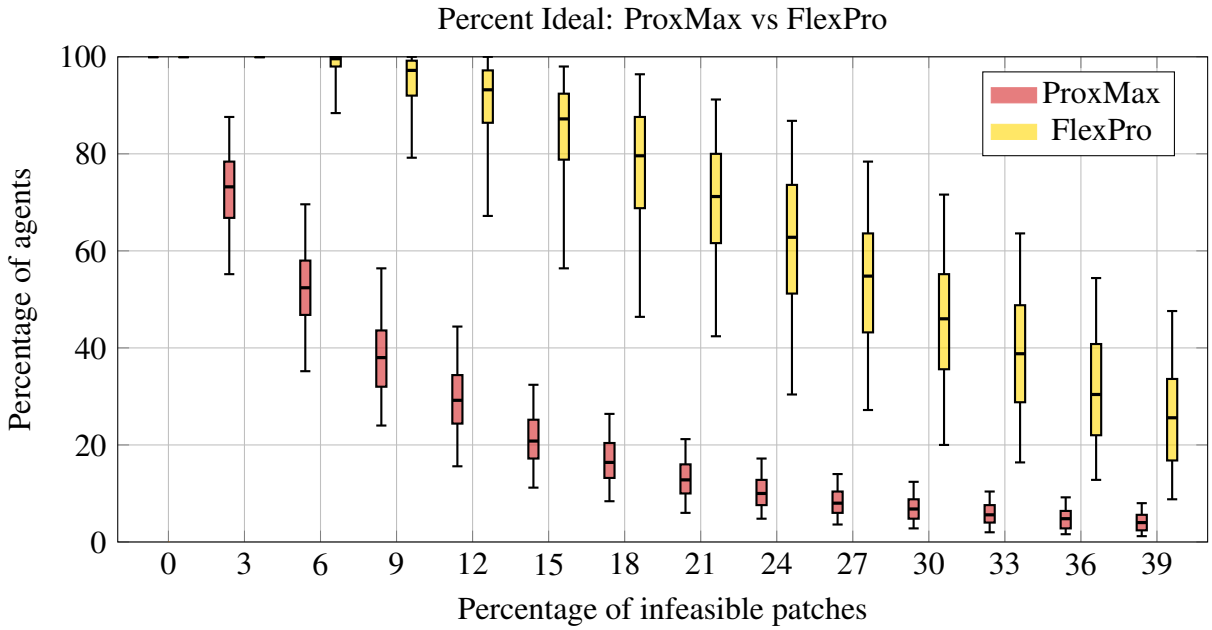


Figure 10. Percent Ideal of ProxMax and FlexPro agents across infeasibility levels

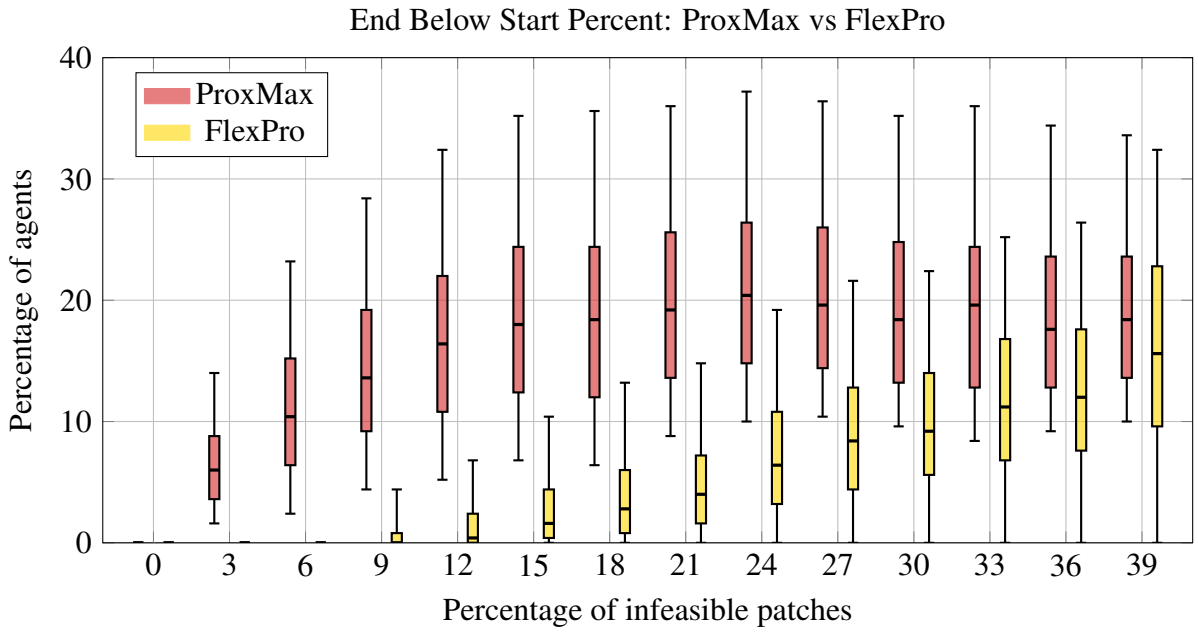


Figure 11. Percent Stop Below of ProxMax and FlexPro agents across infeasibiilty levels

5 Representative Paths

In Section 4 of the main text, we suggest that we can think of the two strategies we examine as lotteries, using Figures 1 and 2 to make this idea concrete. Each figure plots a sample of 100 randomly selected paths for each of our two strategies, Ascend and FPM. In this section, we discuss our sampling procedure and the construction of the "representative" or "average" paths, and compare the sample to the full population.

The first step in constructing our sample was to randomly select four pairs of parameter specifications from the middle ranges of our *hillcount* and *percent-infeasible* variables. Specifically, we randomly drew four pairs from the product set $\{5,7,9,11,13\} \times \{9,12,15,18,21,24\}$, which yielded the pairs (7,12), (5,15), (9,21), and (11,9) (the *hillcount* is the first number in each pair). Then, for each pair, we ran a trial with 250 agents per type and collected the path data (the number of steps and the patch value at each step) for 25 randomly selected agents per type. After collecting the data, we normalized the patch values to be percentages of the ideal value for each trial, following our procedure in the full analysis. We then calculated the performance averages for the two samples to check how they compared to the performance averages for the full population. These comparisons are presented in Tables 1 and 2. Our samples do not perfectly represent the full populations of the two types, as the tables show. But they're not wildly divergent either, so we believe they are useful as illustrations.

Figures 1 and 2 in the main text plot the (normalized) paths for each agent in our samples. We split these into two subgroups, Success Paths (those that reach the ideal) and Failure Paths (those that do not reach the ideal); we further subdivide the FPM Success Paths into "Efficient" Success

Statistic	Successes	Failures	Combined	Full Pop.
Expected Length	8.97	8.77	8.83	6.31
Percent Ideal	1.00	0.00	0.30	0.31
Percent Stop Below	0.00	0.00	0.00	0.00
Expected Stop Value	1.00	0.76	0.83	0.70
Exp. Percent Climb Down	0.00	0.00	0.00	0.00
Exp. Percent Path Sacrifice	0.00	0.00	0.00	0.00
Exp. Path Average Value	0.68	0.51	0.56	0.46
Average Start Value	0.35	0.26	0.29	0.24
Sample size (n)	30	70	100	16.25 mil

Table 1. Statistical Comparison of Ascend Dataset Subgroups

Statistic	Successes	Failures	Combined	Full Pop.
Expected Length	14.38	9.14	13.23	10.64
Percent Ideal	1.00	0.00	0.78	0.67
Percent Stop Below	0.00	0.14	0.03	0.06
Expected Stop Value	1.00	0.43	0.88	0.79
Exp. Percent Climb Down	0.22	0.33	0.24	0.19
Exp. Percent Path Sacrifice	0.21	0.21	0.21	0.15
Exp. Path Average Value	0.51	0.32	0.47	0.43
Average Start Value	0.29	0.26	0.28	0.24
Sample size (n)	78	22	100	16.25 mil

Table 2. Statistical Comparison of FPM Dataset Subgroups

Paths (those that have 3 or fewer downward steps) and "Costly" Success Paths (those with more than 3 downward steps). For each of these subgroups, we used Claude AI (model: Sonnet 4.5, https://claude.ai) to construct a "representative" or "average" path from our sample data. These depict the central performance tendencies for a subgroup in a single illustrative path: they start at the average starting point for the subgroup, they take the average number of steps, they take the average number of downward steps and average number of steps below the starting point, and the average of their path values is equal to the Expected PAV for the subgroup.

Figures 1 and 2 offer an instructive way of visualizing what we mean when we characterize the two strategies we consider as embodying distinct lotteries. As we say in the text, FPM consists of a lottery between efficient success, costly success, and failure, while Ascent consists of a lottery between efficient success and (non-costly) failure.[5] What these figures do not illustrate is how the

[5]Failure for Ascenders is "non-costly" in the sense that, although they fail to reach the ideal, they never step down, nor do they spend time below their starting point. There are also instances in which FlexPros fail and the failure paths

structure of these lotteries depends on the structure of the landscape an agent finds herself in, and the analysis of our simulations also allows us to say something about that. In general, whether the FPM strategy results in efficient success, costly success, or failure depends on where in a landscape the agent begins her search (and we can say something similar for agents implementing the Ascend strategy). How likely an FPM agent is to start her search from a patch that will lead to efficient success as opposed to costly success or failure varies significantly from landscape to landscape, though. In fact, there are landscapes where FlexPros will have efficient success from most starting locations, others where costly success is most likely, and still others where FlexPros will fail from a majority of starting locations. As we emphasize in the text, an advantage of the kind of computational modeling we describe here is that it allows us to analyze the extent of this variation.

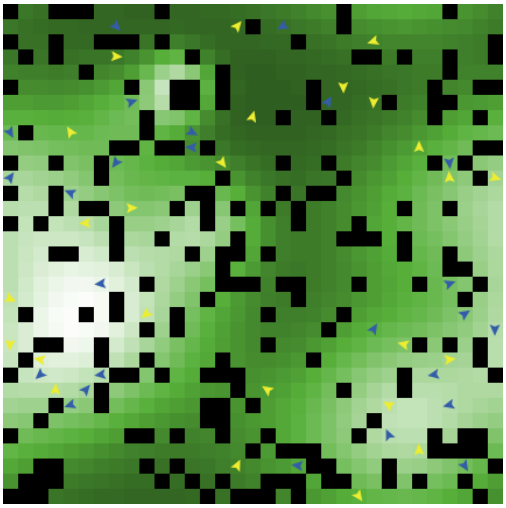
To illustrate this point concretely, we present maps of three different landscapes along with graphs illustrating the paths taken by 25 Ascenders and 25 FlexPros placed on those maps at random locations. Each landscape was generated with a *hillcount* $= 11$ and *percent-infeasible* $= 21$. In the maps the *pheight* of patches is depicted visually (with white representing the highest-valued patches, dark green the lowest-valued patches, and black the infeasible patches). *Ascenders* are blue, *FlexPros* are yellow. As an additional point of comparison, the path graphs also include an average representative path that was constructed using the mean normalized start value and expected path length, expected stop value, expected path value, expected climb down percent, and expected path percent sacrifice that were calculated for the 500 complete runs of our simulations with *hillcount* $= 11$ and *percent-infeasible* $= 21$. As the path graphs illustrate, the first map lends itself to efficient success for FPM, while the second makes costly success more likely, and the third makes failure more likely.

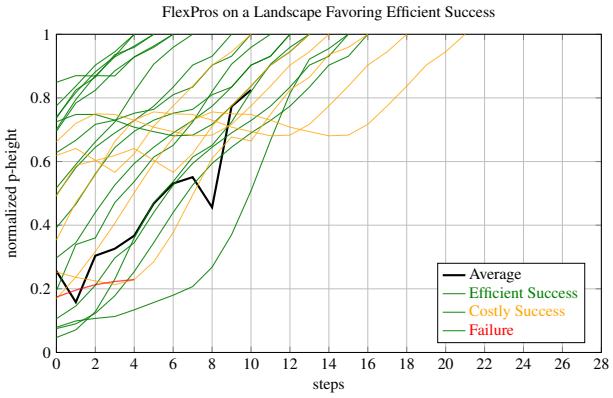
Two further observations are warranted here. First, landscapes where implementing the FPM strategy is likely to result in costly success also tend to have relatively higher failure rates for agents implementing Ascent. This is because the landscape features that explain costly success for FlexPros — namely, the existence of many hills that are geographically distinct such that there are valleys between them — also explain failure for Ascenders. In other words, Ascenders tend to fail when they terminate search at a local optimum, whereas, for FlexPros, costly success is explained by starting on (or having to navigate along and around) small hills on the way to the global optimum. Note, however, that whether landscapes with high failure rates for FlexPros tend to also have relatively higher failure rates for Ascenders is harder to say because, unlike costly success, failure for FlexPros is driven by something — namely the presence of clusters of infeasible patches — that has relatively less impact on Ascenders. The second thing to note is that it can be hard to distinguish landscape types simply by visually inspecting them. We take it that this latter fact, especially when combined with the fact that we're not generally in a position to know what type of landscape we find ourselves in, provides further reason to think that computational modeling can lend useful insight to this debate.

---

are non-costly in this way, but notice that FPM failures are typically also accompanied by costly steps down, and, in any case, from an analytical standpoint the contrasts between efficient success, costly success, and failure is starker than the contrast between costly and non-costly failure.

Figure 12. Landscape that favors efficient success for FlexPros.





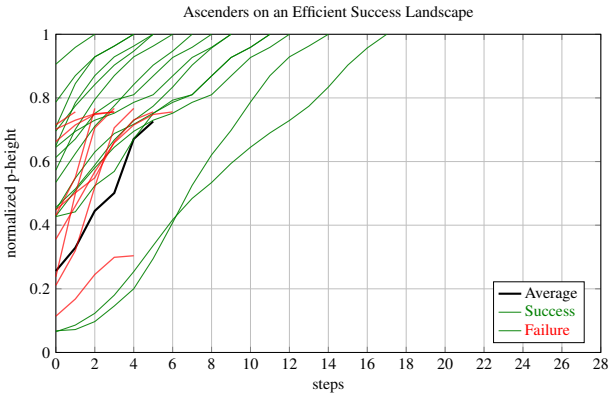
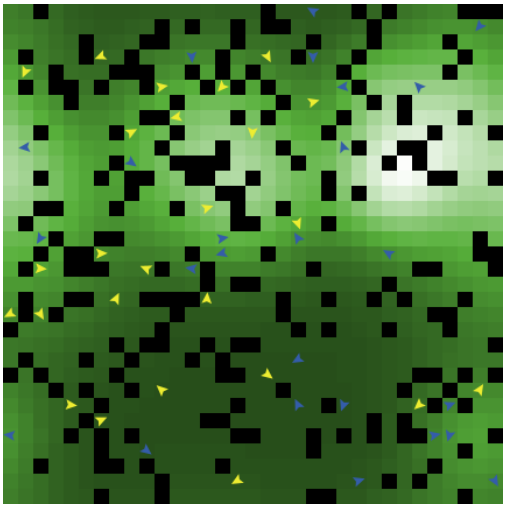


Figure 13. Landscape that favors costly success for FlexPros.



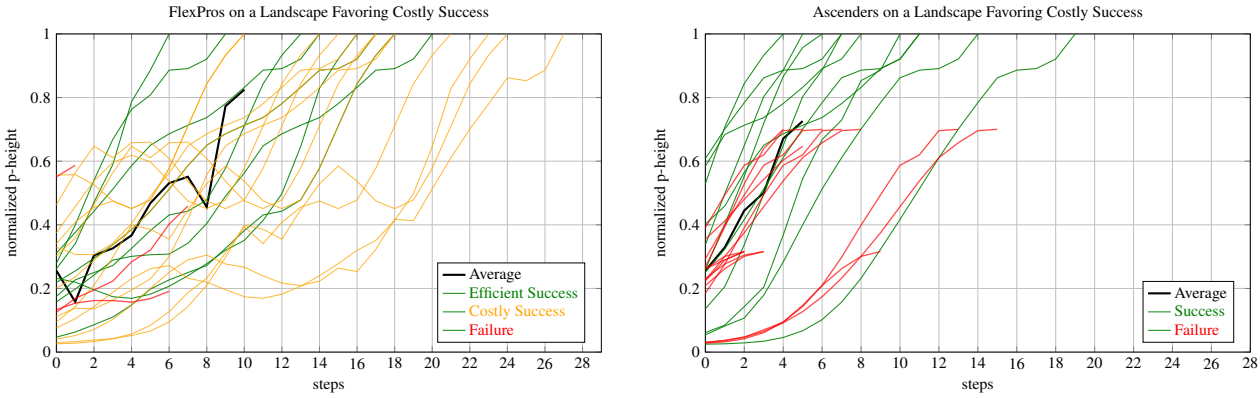
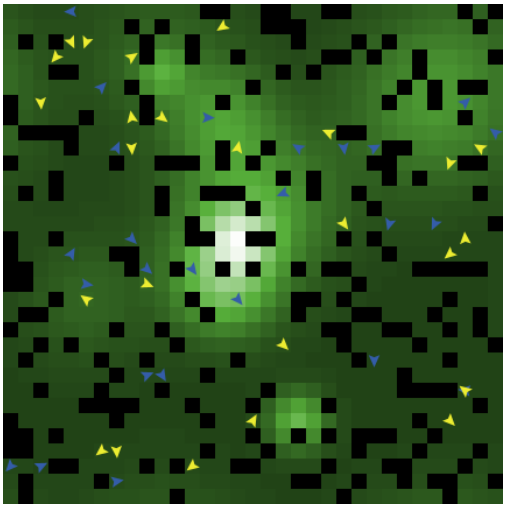
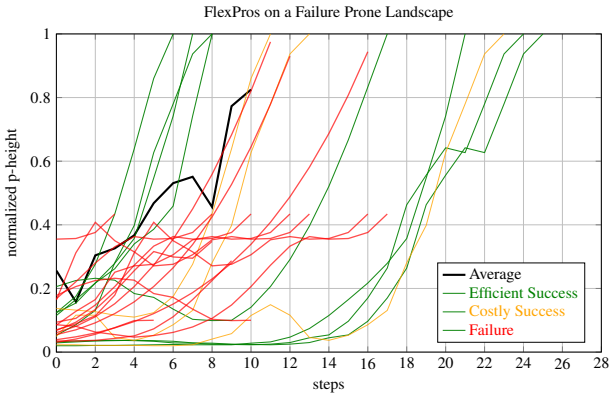
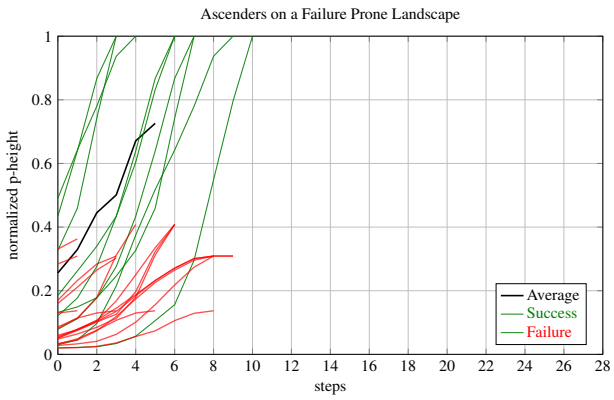


Figure 14. Landscape that favors failures for FlexPros.







6 Alternative Landscapes

To confirm that the results our characterization of the trade-offs is robust to alternative ways of generating landscapes and characterizing the state-space, we evaluated numerous ways of parameterizing the landscape generation algorithm. In particular, we ran simulations with different values for the *hill-amplitude* and *hill-spread* parameters that determine how steep hills are and how much of the landscape they cover. We also considered various ways of distributing infeasible patches on the landscape, e.g., methods that prevented patches close to peaks from being infeasible, or methods that clustered infeasible patches. None of these variations had a significant qualitative impact on the results we report.

In addition to the alternatives described above we also considered two more fundamental differences in ways of generating and conceptualizing the landscape. One such change (mentioned in fn. 24 of the main text) involved guaranteeing the existence of a gradient on the landscape. To do this the landscape generation algorithm was modified so that the Gaussian hill generation process would begin by creating an initial hill with:

$$base - amplitude \sim U(6,20) \text{ and } base - spread = 16.$$

These parameters guarantee that the hill makes a meaningful contribution to the *pheight* of every patch on the landscape. In our 33×33 landscape the farthest any two patches can ever be from one another is approximately 23 units away, and with a spread parameter of 16 the base hill contributes approximately 25% of its base-amplitude to the *pheight* of such patches. Tables 3 and 4 provide the summary statistics for 70,000 runs of a BehaviorSpace simulation of such a model using the same parameter sweeps as the simulation discussed in the main text. Landscapes with gradients improve the performance of both Ascenders and FlexPros, with both strategies significantly improving their expected stop values and expected path values, and FlexPros climbing down significantly less often. Overall, however, the trade-offs between the two strategies is similar to what we found for the model discussed in the main text.

The second substantively different type of landscape we considered was an unwrapped grid. This landscape was parameterized in the same way as the baseline model, but the edges of the landscape were not wrapped, creating a 33×33 grid with 1089 patches where the maximum distance between two patches was approximately 47 units. Once again, the performance of Ascenders relative to FlexPros remains consistent with the results reported for the model discussed in the main text, with both strategies faring slightly worse than they do in the wrapped landscape, The only noteworthy difference is that the pathlength of FlexPros increases significantly in the bounded landscape. See Tables 5 and 6 for summary statistics describing a 70,000 run BehaviorSpace simulation (using the same parameter sweeps).

Statistic	Ascend	ProxMax	FlexPro	Random
Optimum Percent	0.42	1.00	1.00	0.01
Endpoint Normalized	0.91	1.00	1.00	0.48
	(0.05)	(0.00)	(0.00)	(0.15)
Path Length	7.77	12.52	13.44	8.94
	(5.83)	(6.85)	(8.49)	(10.39)
Path Value Normalized Max	0.64	0.63	0.65	0.48
	(0.10)	(0.10)	(0.10)	(0.15)
Path Value Normalized Start	1.83	1.83	1.83	1.16
	(0.82)	(0.79)	(0.80)	(0.67)
Increase Normalized	0.91	1.60	1.54	0.00
	(0.83)	(0.88)	(0.93)	(0.12)
Shortfall Normalized	0.06	0.00	0.00	0.58
	(0.05)	(0.00)	(0.00)	(0.16)
Climb Down Percent	0.00	0.06	0.06	0.45
	(0.00)	(0.14)	(0.16)	(0.34)
Path Below Start Percent	0.00	0.06	0.06	0.43
	(0.00)	(0.18)	(0.09)	(0.82)
End Below Start Pct	0.00	0.00	0.00	0.43
Max Peak	48.83	—	—	—

We report interquartile ranges in parentheses where available.

Table 3. Summary Statistics for Baseline Experiment on Gradient Landscape

Statistic	Ascend	ProxMax	FlexPro	Random
Optimum Percent	0.36	0.22	0.67	0.01
Endpoint Normalized	0.85	0.62	0.92	0.48
	(0.07)	(0.26)	(0.04)	(0.16)
Path Length	7.37	4.44	10.62	8.93
	(5.90)	(5.24)	(8.77)	(10.38)
Path Value Normalized Max	0.63	0.53	0.61	0.48
	(0.11)	(0.22)	(0.13)	(0.15)
Path Value Normalized Start	1.73	1.33	1.66	1.15
	(0.82)	(0.75)	(0.81)	(0.68)
Increase Normalized	0.88	0.22	1.03	0.00
	(0.85)	(0.32)	(1.00)	(0.11)
Shortfall Normalized	0.08	0.38	0.05	0.58
	(0.08)	(0.26)	(0.04)	(0.16)
Climb Down Percent	0.00	0.06	0.06	0.45
	(0.00)	(0.22)	(0.21)	(0.33)
Path Below Start Percent	0.00	0.07	0.06	0.42
	(0.00)	(0.27)	(0.13)	(0.80)
End Below Start Pct	0.00	0.15	0.06	0.43
Max Peak	48.45	—	—	—

We report interquartile ranges in parentheses where available.

Table 4. Summary Statistics for Infeasibility Extension on Gradient Landscape

Statistic	Ascend	ProxMax	FlexPro	Random
Optimum Percent	0.28	1.00	1.00	0.01
Endpoint Normalized	0.68	1.00	1.00	0.14
	(0.28)	(0.00)	(0.00)	(0.11)
Path Length	6.21	16.77	18.55	8.93
	(5.47)	(11.69)	(14.56)	(10.40)
Path Value Normalized Max	0.43	0.45	0.49	0.13
	(0.30)	(0.13)	(0.11)	(0.11)
Path Value Normalized Start	5.08	5.43	5.61	2.35
	(4.24)	(5.63)	(6.08)	(0.80)
Increase Normalized	5.78	8.29	8.80	-0.02
	(4.28)	(6.91)	(6.64)	(0.10)
Shortfall Normalized	0.35	0.00	0.00	0.79
	(0.28)	(0.00)	(0.00)	(0.10)
Climb Down Percent	0.00	0.09	0.12	0.45
	(0.00)	(0.20)	(0.26)	(0.33)
Path Below Start Percent	0.00	0.09	0.07	0.43
	(0.00)	(0.21)	(0.10)	(0.79)
End Below Start Pct	0.00	0.00	0.00	0.42
Max Peak	39.12	—	—	—

We report interquartile ranges in parentheses where available.

Table 5. Summary Statistics for Baseline Experiment on Landscape w/ Edges

Statistic	Ascend	ProxMax	FlexPro	Random
Optimum Percent	0.24	0.18	0.62	0.01
Endpoint Normalized	0.64	0.42	0.88	0.14
	(0.34)	(0.31)	(0.08)	(0.10)
Path Length	5.85	4.83	13.48	8.92
	(5.39)	(5.72)	(12.63)	(10.38)
Path Value Normalized Max	0.41	0.23	0.41	0.13
	(0.29)	(0.16)	(0.18)	(0.11)
Path Value Normalized Start	4.94	3.35	4.90	2.45
	(4.07)	(2.09)	(5.15)	(0.75)
Increase Normalized	5.53	1.93	6.51	-0.02
	(4.11)	(1.27)	(6.78)	(0.09)
Shortfall Normalized	0.42	0.65	0.08	0.78
	(0.34)	(0.31)	(0.07)	(0.10)
Climb Down Percent	0.00	0.12	0.14	0.45
	(0.00)	(0.26)	(0.33)	(0.31)
Path Below Start Percent	0.00	0.11	0.08	0.42
	(0.00)	(0.22)	(0.15)	(0.78)
End Below Start Pct	0.00	0.17	0.07	0.42
Max Peak	37.75	—	—	—

We report interquartile ranges in parentheses where available.

Table 6. Summary Statistics for Infeasibility Extension on Landscape w/ Edges