Supplemental Materials

Making the Crowd Wiser: (Re)combination through Teaming in Crowdsourcing

Appendix A: The *NK* Fitness Landscape Model

The creation of new technologies, products and services involves solving complex problems that require configuring a large number of interdependent elements (Simon 1962). The attempt to find high quality configurations of interdependent elements can be illustrated using the metaphor of a "rugged landscape" (Levinthal 1997). Each location on the landscape represents one possible configuration of elements, and its height represents the performance (or fitness) of that configuration. Finding a high quality configuration is thus akin to reaching a high peak on the rugged landscape. The *NK* fitness landscapes model provides a formal framework for constructing such landscapes computationally and for modeling agents' behavioral rules aimed at reaching high peaks.

A.1. Modeling the Problem Space

In our context, the fitness landscape represents the problem space that solvers must search for high quality solutions to configuring interdependent elements. The problem space involves N elements $d_1, d_2, \ldots, d_N$, and K interdependencies between each element and other elements. A configuration is thus represented by a N-element vector $\mathbf{d} =< d_1, d_2, \ldots, d_N >$, where d_i can take the value of 0 or 1. The interdependencies among N elements can be represented by an $N \times N$ influence matrix (*INF*), where $INF_{i,j} = 1$ if the fitness contribution of the i^{th} element c_i is influenced by the j^{th} element, or 0 otherwise. Figure A.1.1 shows an example of an influence matrix when $N = 12$ and $K = 3$. The fitness contribution of each element is influenced by its own specification (i.e., all diagonal cells (i.e., $i = j$) in Figure A.1.1 have the value of 1) and the specification of three other elements. For instance, the fitness contribution of the first element c_1 is influenced by the specifications of the second, fifth, and last elements according to Figure A.1.1.

Given an influence matrix, fitness values are randomly generated. Specifically, for each possible specification of element d_i and K other interdependent elements, c_i is randomly drawn from a uniform distribution $U(0, 1)$. For instance, the first element d_1 in Figure A.1.1 will have a total of 16 possible fitness contributions randomly drawn from the uniform distribution (i.e., $d_1 \in \{0, 1\}$, $d_2 \in \{0, 1\}$, $d_5 \in \{0, 1\}$, $d_{12} \in \{0, 1\}$). The overall fitness value of a configuration $\mathbf{d}$ is the average of all N fitness contributions c_i, namely $F(\mathbf{d}) = \frac{1}{N}\sum_i c_i$ where $c_i = c_i(d_i|k \text{ other } d_j\text{'s})$. Given the influence matrix as per Figure A.1.1, a configuration $\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1\rangle$ has a fitness value:

1	1	0	0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	1	0	1	0	0
0	1	0	1	0	0	1	0	0	0	1	0
0	0	0	0	1	1	0	0	1	1	0	0
1	0	1	0	0	1	0	1	0	0	0	0
0	0	1	1	0	0	1	0	0	0	1	0
1	0	0	1	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	1
0	0	1	0	1	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	1

Figure A.1.1 Example of an Influence Matrix (*INF*) when *N=12, K=3*

$$
\begin{aligned}
F(\mathbf{d}) = \frac{1}{12}[&c_1(d_1 = 0|d_2 = 1, d_5 = 0, d_{12} = 1) + c_2(d_2 = 1|d_4 = 1, d_8 = 1, d_9 = 0) + \\
&c_3(d_3 = 0|d_6 = 1, d_8 = 1, d_{10} = 1) + c_4(d_4 = 1|d_2 = 1, d_7 = 0, d_{11} = 0) + \\
&c_5(d_5 = 0|d_6 = 1, d_9 = 0, d_{10} = 1) + c_6(d_6 = 1|d_1 = 0, d_3 = 0, d_8 = 1) + \\
&c_7(d_7 = 0|d_3 = 0, d_4 = 0, d_{11} = 0) + c_8(d_8 = 1|d_1 = 0, d_4 = 1, d_6 = 1) + \\
&c_9(d_9 = 0|d_2 = 1, d_6 = 1, d_{12} = 1) + c_{10}(d_{10} = 1|d_3 = 0, d_5 = 0, d_7 = 0) + \\
&c_{11}(d_{11} = 0|d_2 = 1, d_6 = 1, d_9 = 0) + c_{12}(d_{12} = 1|d_1 = 0, d_4 = 1, d_7 = 0)]
\end{aligned}
$$

This procedure of assigning fitness values to each configuration enables us to tune the ruggedness of the generated landscapes by modifying the degree (i.e., K) and pattern (i.e., *INF*) of interdependencies.[1] When there are few interdependencies among elements (i.e., low K), the generated landscape is more smooth; conversely, when elements are highly interdependent (i.e., high K), the resulting landscape is more rugged. Figures A.1.2a and A.1.2b illustrate a smooth landscape with a single peak (i.e., the global optimum) and a rugged landscape with multiple peaks (i.e., a global optimum and numerous local optima), respectively. The ruggedness of the fitness landscape influences the difficulty of finding the global optimum. As shown by Figure A.1.2a, when the landscape is smooth, incremental local search enables agents to eventually reach the global optimum. However, when the landscape is rugged (see Figure A.1.2b), incremental local search tends to lead agents to local optima. What's worse, agents are likely to get stuck at those local optima because incrementally changing their configurations will always yield fitness values lower than the status quo.

Figure A.1.3 shows the average number of peaks on the fitness landscape across different levels of complexity (i.e., K) when $N = 12$ (i.e., the setup in our main analysis). Not surprisingly, when $K = 1$, there

[1] Whilst it is possible to specify the influence matrix to explore the impact of various interdependency structures (e.g., Rivkin and Siggelkow 2003, 2007), this study adopts a more general approach by randomizing these effects – i.e., we randomly generate influence matrices according to predefined parameters N and K and then use the stochastic procedure described above to generate fitness values for all possible configurations (i.e., the fitness landscape).

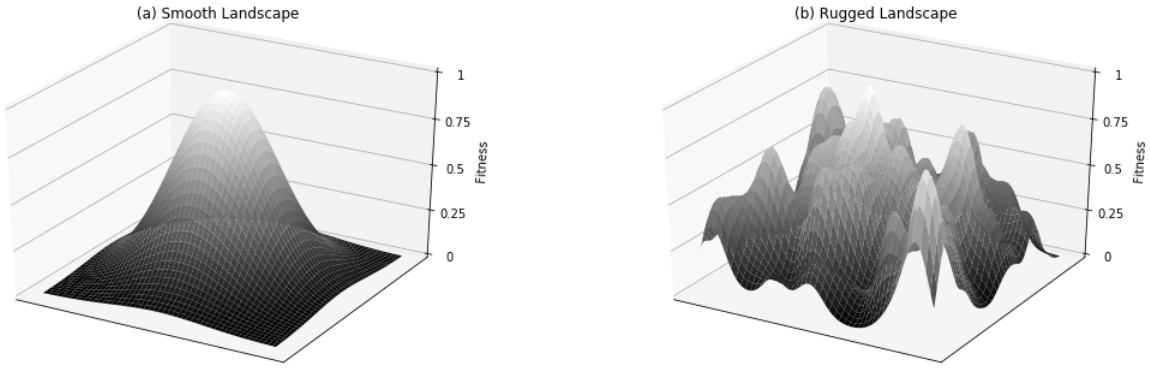

Figure A.1.2 Illustrative Examples of Fitness Landscapes

Notes: The real fitness landscape is a $N+1$ dimensional space (N dimensions for N elements and 1 dimension for fitness values). For illustrative purpose, Figures A.1.2a and A.1.2b are 3-D simplifications of the real fitness landscape. The x-axis and y-axis have no specific meaning other than to indicate the proximity of configurations.

are only a few peaks on the landscape. As K increases, the number of peaks on the landscape significantly increases. When $K = 11$, there are around 300 peaks on the landscape. In other words, around 7.3% ($=\frac{300}{2^{12}}$) configurations are local peaks that hinder agents from reaching the global optimum. As such, the degree of interdependencies (i.e., K) translates into the complexity of searching for high fitness configurations on the landscape.

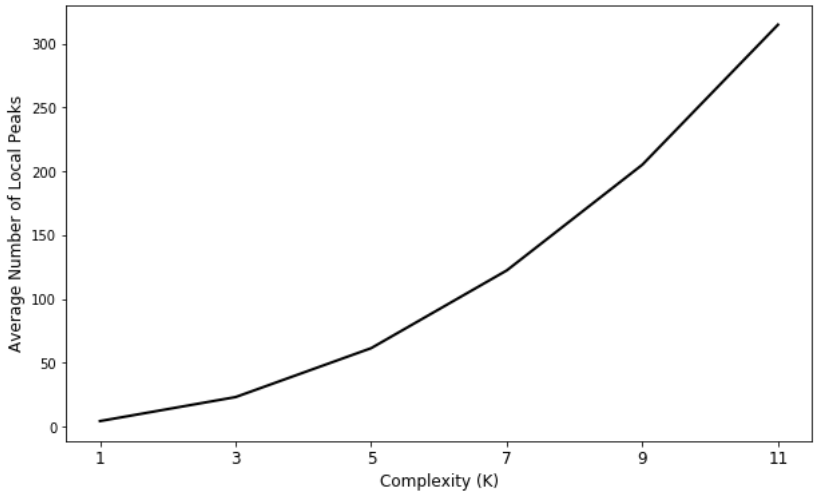

Figure A.1.3 Number of Local Peaks when *N=12*

A.2. Modeling Agents with Diverse Knowledge

As discussed, when the crowdsourced problem becomes broad in scope and requires the integration of knowledge from many different domains, there may be no individual solver who understands all aspects of the problem. Imperfect knowledge possessed by solvers can hinder solvers from fully understanding the problem,

leading to incomplete mental representations of the problem. Imperfect knowledge also prevents solvers from developing complete solutions beyond the scope of their individual knowledge. To faithfully model these characteristics, we need to model agents' knowledge in a way that the lack of knowledge about elements outside their knowledge domain would hamper a focal agent's search on the landscape. To do so, we adopt Gavetti and Levinthal's model (2000) of cognitive simplification, which allows agents to have a simple, low-dimensional representation of a more complex, high-dimensional fitness landscape based on their imperfect knowledge. The actual modeling approach is to set an agent's perceived fitness value $\tilde{F}(\mathbf{d})$ of solution $\mathbf{d}$ by averaging the fitness values across all possible configurations of the elements outside of its knowledge domain D. Table A.2.1 provides a numeric example of how cognitive simplification is modeled. When there are 8 design elements (i.e., $N = 8$), an agent with knowledge about the first 6 elements would perceive a solution $\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 1\rangle$ as $\langle 0, 1, 0, 1, 0, 1, *, *\rangle$, where $*$ indicates unknown elements. The perceived value equals to the average fitness across all the possible configurations of the unknown elements – i.e., all possible combinations of the last two elements in the third column of Table A.2.1.

Table A.2.1 An Illustrative Example of Cognitive Simplification

Agent's Knowledge	Configuration	Fitness across Unknowns	Perceived Fitness
$D = \{1, 2, 3, 4, 5, 6\}$	$\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, *, *\rangle$	$F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 0\rangle) = 0.82$ $F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 1\rangle) = 0.75$ $F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 1, 0\rangle) = 0.94$ $F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 1, 1\rangle) = 0.68$	$\tilde{F}(\mathbf{d}) = 0.7975$

This modeling approach does not mean that an agent actually knows the fitness values of all solutions and is able to calculate the average fitness values across configurations of unknown elements. Instead, the average value is simply an unbiased expected value that an agent would perceive a solution configuration given its imperfect knowledge. As illustrated in the third and fourth columns of Table A.2.1, the perceived fitness values of different configurations may be higher or lower than the actual fitness values due to a lack of knowledge. Yet, how the agent perceives all configurations on the landscape as a whole would be largely correlated with the actual fitness landscape.

Figure A.2.1 shows how the generated cognitive simplification correlates with the actual fitness landscape when $N = 12$ and $|D| = 6$ (i.e., the setup in our main analysis). Whilst the correlation between randomly generated fitness landscapes (the dashed line) is always around 0, the correlation between the cognitive simplification and the actual fitness landscape is above 0.6 when problem complexity is low (i.e., the leftmost point on the solid line). As the problem complexity increases, their correlation declines but remains significant and positive. These results are consistent with conventional wisdom. That is, as elements become tightly interdependent, a lack of knowledge of certain elements would exacerbate the imperfect evaluation of solution configurations.

To capture solvers' diverse knowledge, in each replication of the simulations, we always randomly set each agent's decision set D. Each design element has a equal probability of being selected into agents' decision sets. It is possible that certain elements being more commonly represented in the population of agents and others being less so. Therefore, we also experimented with alternative setups in which elements are chosen

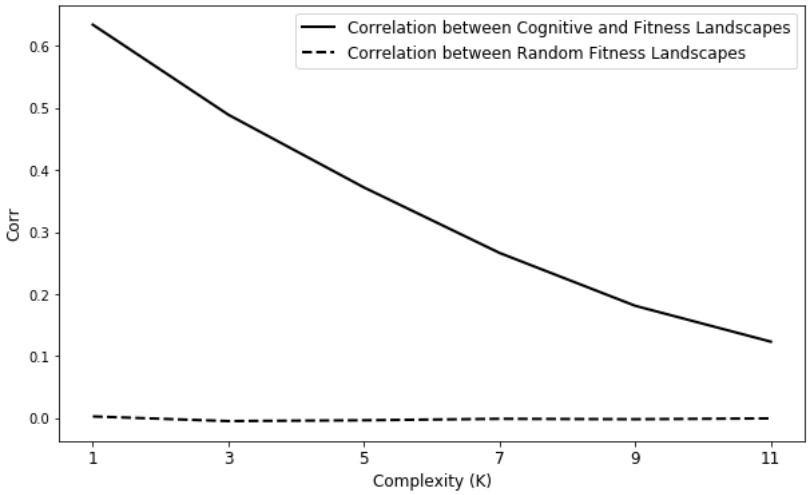

Figure A.2.1 Correlation between Cognitive Landscapes and Fitness Landscapes when *N=12, |D|=6*

according to a nearly normal distribution (i.e., $Beta(2,2)$) and found that the alternative distribution only affects the size of the proposed effects without qualitatively altering our conclusions.

With imperfect evaluation, agents search the problem landscape in an iterative and incremental manner. Continuing with the example above, let's suppose that an agent's current solution configuration is $\mathbf{d} = \langle 0,1,0,1,0,1,*,* \rangle$, with knowledge about how to configure the first 6 design elements (i.e., $D = \{1,2,3,4,5,6\}$). During search, the agent would randomly change one design element within its decision set D and adopt the change if the new configuration is perceived as being better than the status quo based on the imperfect evaluation. In other words, a new configuration $\mathbf{d}'$ would be selected from the 6 neighboring configurations:

$$\{\langle \underline{1},1,0,1,0,1,*,* \rangle, \langle 0,\underline{0},0,1,0,1,*,* \rangle, \langle 0,1,\underline{1},1,0,1,*,* \rangle,$$
$$\langle 0,1,0,\underline{0},0,1,*,* \rangle, \langle 0,1,0,1,\underline{1},1,*,* \rangle, \langle 0,1,0,1,0,\underline{0},*,* \rangle,\}$$

The new configuration $\mathbf{d}'$ would be adopted to replace the current configuration $\mathbf{d} = \langle 0,1,0,1,0,1,*,* \rangle$ if $\tilde{F}(\mathbf{d}') > \tilde{F}(\mathbf{d})$. Otherwise, the current configuration $\mathbf{d}$ is retained.

In the extended model, we also examine the scenario where agents exhibit heterogeneous search capability and are able to consider more than one alternative configuration in each iteration. Continuing with the $N = 8$ and $D = \{1,2,3,4,5,6\}$ example, suppose that the agent can consider 8 alternative configurations and the current configuration is $\mathbf{d} = \langle 0,1,0,1,0,1,*,* \rangle$. The agent would first consider all 6 neighboring configurations listed above and 2 additional configurations that involve changing two decisions – e.g., $\{\langle \underline{1},\underline{0},0,1,0,1,*,* \rangle, \langle 0,\underline{0},\underline{1},1,0,1,*,* \rangle\}$. Among all considered configurations, the one with the highest perceived value $\tilde{F}(\mathbf{d}')$ will be eventually selected. The selected configuration $\mathbf{d}'$ will be adopted to replace the current configuration $\mathbf{d}$ if $\tilde{F}(\mathbf{d}') > \tilde{F}(\mathbf{d})$. In this way, the number of alternative configurations the agent can consider reflects its problem-solving capability. An agent with a higher level of problem-solving capability is able to consider more alternatives and assess the ramifications of changing more design elements at once within a given time frame (Rivkin and Siggelkow 2003).

In the extended model, we randomly vary the number of alternatives an agent can consider from 1 to $2 \times N_D$ (i.e., 12 in our main analysis). It is worth noting that the upper bound is set to sufficiently capture the advantages of agents with higher levels of search capability in altering and evaluating multiple design elements simultaneously. Further increasing the upper bound would only add to the computational burden without qualitatively changing the results.

A.3. Modeling Teaming

At the time of teaming t, each agent has a probability (p_{teamup}) to initiate a team-up invitation. If an agent u decides to initiate a team-up invitation as per p_{teamup}, it sorts other agents according to the quality signal on which it relies and considers possible teammates one by one. The invitation process stops if an agent on the ordered list accepts the invitation (i.e., a team is formed) or if no agents accept the invitation (i.e., no team is formed). The recipients v of team-up invitations also sort other agents according to the public quality signal and have a probability (p_{accept}) to accept the team-up invitation. When a team is formed, agents in the team are excluded from subsequent matches.

In our main analysis, we set p_{teamup} as a constant equal to 0.2 to approximate the proportion of teams on Kaggle. As for p_{accept}, we specify it as an exponential decay function of the extent to which the sender is favored according to the public quality signal. It takes the form of $\alpha \cdot (e^{-\beta x} + \gamma)$, where x is the relative order of the sender agent according to different quality signals. When rank information is used, x represents the relative order of agents' performance at the timing of teaming (i.e., $F(\mathbf{d})$; x takes the value of 1 if the sender agent is with the highest $F(\mathbf{d})$ among all solvers, so on and so forth). When knowledge-based information is used, x represents the relative order of the knowledge overlap between the sender u and recipient v (i.e., $|\boldsymbol{D}_u \cap \boldsymbol{D}_v|$; x takes the value of 1 if the sender agent is with a modest level of knowledge overlap when teaming takes place according to moderate knowledge overlap).[2] When capability information is used, x represents the relative order of the sender's search capability (i.e., the number of alternatives it can consider; x takes the value of 1 if the sender agent has the greatest search capability among all solvers).

To calibrate hyperparameters α, β, and γ, we conducted a grid search on the parameter space (i.e., $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\beta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\gamma = \{0.001, 0.003, 0.005, 0.007, 0.009\}$) with 200 populations of 100 agents to minimize the mean squared error (MSE) of the following terms: 1) the distribution of team member's relative order is as closed as to our empirical observation (i.e., Figure 2a); 2) the likelihood of teaming probability across x closely matches Figure 2b. We specify $\alpha = 0.5$, $\beta = 0.1$, and $\gamma = 0.001$ for our main analysis.

Unlike rank- and capability-based information, the extent of knowledge overlap varies among agents. An agent with an unfavored extent of knowledge overlap for a focal agent might be perceived as having a favored extent of knowledge overlap for another agent (i.e., a greater value of p_{accept}). As a result, agents are more likely to team up when they rely on the extent of knowledge overlap to determine possible teammates. α thus is rescaled by multiplying it by 0.14 when agents team up according to knowledge overlap. In a similar

[2] In our primary analyses, agents are sorted by the extent of knowledge overlap (i.e., $\langle 6, 5, 4, 3, 2, 1, 0 \rangle$ for maximal knowledge overlap; $\langle 3, 2, 4, 1, 5, 0, 6 \rangle$ for modest knowledge overlap; and $\langle 0, 1, 2, 3, 4, 5, 6 \rangle$ for minimal knowledge overlap).

vein, when we raise the number of agents in the crowd, the proportion of teams increases because agents each will receive more invitations. As such, we rescale α by $\frac{100}{CrowdSize}$.

It is important to note that the specification of the probability of teaming might vary on different crowdsourcing platforms for different contests. As such, our main thesis, namely how teaming impacts the overall crowdsourcing effectiveness through affecting the interplay between the parallel path effect and the *identification*, *integration*, and *teamwork effects*, is built upon random team formation as a counterfactual baseline. Different specifications of the probability of teaming would only alter the size of different effects without qualitatively changing our conclusions (see, for instance, §5.3).

Once a team is formed, solvers integrate their individually developed solutions into a joint one for subsequent teamwork. In our model, the solution developed by the solver with the most favorable signals serves as the baseline for integration. The remaining solvers are then ordered and iterated according to the favorability of their signals. If a design element is not configured in the baseline solution (i.e., marked as $*$) but is configured by a solver, the solver contributes the corresponding configuration of that element to the baseline solution. The integration process ends once all solvers in the team have been iterated. After the integration of solutions, agents in the team stick together for the remainder of the simulations. The next section explains how agents in teams jointly search the landscape after the timing of teaming.

A.4. Modeling Teamwork

Prior research on the *NK* fitness landscape model has documented various ways to model how a complex problem is decomposed and how subsets of design elements are allocated to two or more agents to collectively search the landscape (e.g., Rivkin and Siggelkow 2003, Siggelkow and Rivkin 2005). A key modeling decision is to determine the extent of specialization and integration (Baumann et al. 2018). Specialization refers to the labor division among agents. A team's specialization is jointly determined by agents' decision sets D, which are randomly generated as per section A.2, and how agents with diverse decision sets are teamed together (i.e., the teaming process in section A.3). Integration refers to the process through which agents with different labor divisions collaborate. Unlike groups in traditional organizations where hierarchies play an important role in collaboration, self-organized teams in crowdsourcing are often small and relatively more decentralized and as a result, all team members are more likely to share authority over accepting and rejecting what should be done. Therefore, we adopt the "liaison archetype" as per Siggelkow and Rivkin (2005). That is, a team member agrees to a new solution configuration only if, based on its evaluation, a payoff at least as high as that of the status quo is yielded.

To illustrate how agents in a team jointly search the problem landscape, let us suppose again that the solution configuration is $\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 1\rangle$. Suppose that agent u with decision set $D_u = \{1, 2, 3, 4, 5, 6\}$ and agent v with decision set $D_v = \{3, 4, 5, 6, 7, 8\}$ team up. (Please note that the label u and v are used for illustrative purposes only; the order of search is randomized in the model.) In this case, agent u would perceive the solution as $\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, *, *\rangle$, whilst agent v would perceive the same solution as $\mathbf{d} = \langle *, *, 0, 1, 0, 1, 0, 1\rangle$. In each iteration, agent u would select an alternative solution configuration $\mathbf{d}_u^{'}$ following the rule described in section A.2 (i.e., changing one or two of the first 6 design elements). If the alternative

is perceived as better than agent u's current solution $\mathbf{d}_u$ (i.e., $\tilde{F}_u(\mathbf{d}_u^{'}) > \tilde{F}_u(\mathbf{d}_u)$), the alternative would be proposed to agent v; otherwise, the status quo $\mathbf{d}_u$ would be proposed to agent v.

Once agent v receives the proposal, it would evaluate whether the proposed configuration leads to a perceived value at least as high as its current configuration. The proposal would be approved if $\tilde{F}_v(\mathbf{d}_u^{'}) \geq \tilde{F}_v(\mathbf{d}_v)$. If the proposal is approved by agent u, the solution configuration $\mathbf{d}$ is updated according to agent v's proposal. In order to keep the number of alternatives the team can consider identical before and after teaming, we assume that agent u and agent v each has a chance to propose and evaluate the other's proposal. Continuing the example above, after updating the solution configuration, agent v takes a turn to search for an alternative configuration according to its imperfect evaluation and propose it for agent u's evaluation. See Table A.4.1 for a numerical example of the whole procedure when agent v is with higher search capability (i.e., agent v is able to change two elements at once) and agent u is with lower search capability (i.e., agent u is only able to change one element at once).

Table A.4.1 An Illustrative Example of Joint Search

Input	Solution Configuration $\mathbf{d} = \langle 0,1,0,1,0,1,0,1\rangle$	
	Agent u $D_u = \{1,2,3,4,5,6\}$, $\mathbf{d}_u = \langle 0,1,0,1,0,1,*,*\rangle$	Agent v $D_v = \{3,4,5,6,7,8\}$, $\mathbf{d}_v = \langle *,*,0,1,0,1,0,1\rangle$
Step 1.1	Agent u proposes a new solution configuration based on its evaluation $\mathbf{d}_u^{'} = \langle 0,\underline{0},\underline{1},1,0,1,*,*\rangle$	
Step 1.2		Agent v accepts the proposed solution configuration because $\tilde{F}_v(\langle *,*,\underline{1},1,0,1,0,1\rangle) \geq \tilde{F}_v(\langle *,*,0,1,0,1,0,1\rangle)$
Step 2.0	Solution Configuration $\mathbf{d} = \langle 0,0,1,1,0,1,0,1\rangle$	
	Agent u $\mathbf{d}_u = \langle 0,0,1,1,0,1,*,*\rangle$	Agent v $\mathbf{d}_v = \langle *,*,1,1,0,1,0,1\rangle$
Step 2.1		Agent v proposes a new solution configuration based on its evaluation $\mathbf{d}_v^{'} = \langle *,*,1,1,0,\underline{0},0,1\rangle$
Step 2.2	Agent u rejects the proposed solution configuration because $\tilde{F}_u(\langle 0,0,1,1,0,\underline{0},*,*\rangle) < \tilde{F}_u(\langle 0,0,1,1,0,1,*,*\rangle)$	
Outcome	Solution Configuration $\mathbf{d} = \langle 0,0,1,1,0,1,0,1\rangle$	
	Agent u $\mathbf{d}_u = \langle 0,0,1,1,0,1,*,*\rangle$	Agent v $\mathbf{d}_v = \langle *,*,1,1,0,1,0,1\rangle$

We also assume that during joint search, agents build and update their meta-knowledge of knowledge possessed by others, which serves as a guide for identifying the knowledge needed (Wegner 1987). Meanwhile, team members will gradually become better aware of their teammates' knowledge and construct a shared understanding of the problem (Alavi and Tiwana 2002, Robert et al. 2008, 2018), which helps to align each other's search behaviors. Agents' evaluation of solution configurations gradually becomes more coherent because shared team cognition unifies the divergent cognitions towards a coherent understanding of the problem. The extent to which such a coherent understanding can be built depends on the degree of knowledge overlap. Greater common ground among team members' respective knowledge bases (i.e., more knowledge overlap) facilitates intra-group communication and coordination and reduces inconsistencies and conflicts (Espinosa et al. 2007, Cramton 2001). In the above example (i.e., Table A.4.1), greater common ground increases the likelihood of agents u and v working around the same design elements. When inconsistencies

arise on these design elements, agents are more likely to be aware of the configurations of those design elements outside their knowledge domains that lead to the inconsistencies. Conversely, teams formed by members with no overlapping knowledge might miss the opportunity to build a shared understanding because each of them independently works on a subset of design elements.

As such, in our simulation model, an agent u has a probability of $p_{learn} \times |\boldsymbol{D}_u \cap \boldsymbol{D}_v|$ to incorporate the configuration of one element from its teammate v's decision set in evaluating alternatives $\mathbf{d}$. Continuing the example in Table A.2.1, if an agent u with decision set $D_u = \{1, 2, 3, 4, 5, 6\}$ would perceive a solution $\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, 0, 1\rangle$ as $\langle 0, 1, 0, 1, 0, 1, *, *\rangle$. Now if agent u knows its teammate v has configured the seventh element as 1, the perceived value would equal to the average fitness across the possible configurations of the last element. Table A.4.2 provides a numeric example.

Table A.4.2 Cognitive Simplification Given Knowledge Overlap

Agent's Knowledge	Configuration	Fitness across Unknowns	Perceived Fitness
$D = \{1, 2, 3, 4, 5, 6\}$	$\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, \underline{1}, * \rangle$	$F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, \underline{1}, 0\rangle) = 0.94$ $F(\mathbf{d} = \langle 0, 1, 0, 1, 0, 1, \underline{1}, 1\rangle) = 0.68$	$\tilde{F}(\mathbf{d}) = 0.81$

A.5. Model Validation

In this validation experiment, we investigate the likelihood of finding the global optimum of the problem landscape only through parallel search (i.e., agents search independently) across varying levels of problem complexity. We vary the number of agents in the simulation from 0 to 500 so as to benchmark our model to the classical *NK* model and validate our conceptualization of crowdsourcing as landscape search. Figure A.5.1 shows the results.

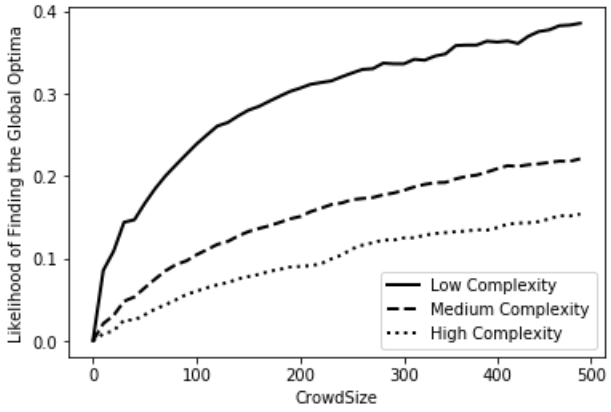

Figure A.5.1 Likelihood of Finding the Global Optima in Parallel Search

As the classical *NK* model would predict (Levinthal 1997), individual agents suffer a performance decline as problem complexity increases (see the performance gap between the solid line (low complexity) and the dotted line(high complexity)). A crowd of 500 solvers has a likelihood of 40% of generating the optimal solution for the solution seeker when the complexity is low, but the probability of finding the optimal solution

reduces to about 10% for high-complexity problems. Moreover, we find that the likelihood of finding the global optima increases as the number of agents, suggesting the efficiency of the parallel path effect for crowdsourcing. Consistent with the crowdsourcing literature (Boudreau et al. 2011, Afuah and Tucci 2012), the simulation results also suggest that the parallel path effect would be more critical for problems with high complexity. The marginal return of increasing the number of solvers keeps decreasing when the complexity is low (see the slope of the solid line). Doubling the number of solvers from 200 to 400 only increases the likelihood of finding the best solution by around 30% ($=(0.39-0.3)/0.3$) when problem complexity is low. Conversely, the marginal return of increasing the number of solvers almost remains the same for problems with high complexity (see the slope of the dotted line). Doubling the number of solvers from 200 to 400 can almost double the likelihood of find the best solution when problem complexity is high.

References

Afuah A, Tucci CL (2012) Crowdsourcing as a Solution to Distant Search. *Academy of Management Review* 37(3):355–375.

Alavi M, Tiwana A (2002) Knowledge Integration in Virtual Teams: The Potential Role of KMS. *Journal of the American Society for Information Science and Technology* 53(12):1029–1037.

Baumann O, Schmidt J, Stieglitz N (2018) Effective Search in Rugged Performance Landscapes: A Review and Outlook. *Journal of Management* 45(1):285–318.

Boudreau KJ, Lacetera N, Lakhani KR (2011) Incentives and Problem Uncertainty in Innovation Contests: An Empirical Analysis. *Management Science* 57(5):843–863.

Cramton CD (2001) The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. *Organization Science* 12(3):346–371.

Espinosa JA, Slaughter SA, Kraut RE, Herbsleb JD (2007) Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems* 24(1):135–169.

Gavetti G, Levinthal D (2000) Looking Forward and Looking Backward: Cognitive and Experiential Search. *Administrative Science Quarterly* 45(1):113–137.

Levinthal DA (1997) Adaptation on Rugged Landscapes. *Management Science* 43(7):934–950.

Rivkin JW, Siggelkow N (2003) Balancing Search and Stability: Interdependencies among Elements of Organizational Design. *Management Science* 49(3):290–311.

Rivkin JW, Siggelkow N (2007) Patterned Interactions in Complex Systems: Implications for Exploration. *Management Science* 53(7):1068–1085.

Robert LP, Dennis AR, Ahuja MK (2008) Social Capital and Knowledge Integration in Digitally Enabled Teams. *Information Systems Research* 19(3):314–334.

Robert LP, Dennis AR, Ahuja MK (2018) Differences are Different: Examining the Effects of Communication Media on the Impacts of Racial and Gender Diversity in Decision-Making Teams. *Information Systems Research* 29(3):525–545.

Siggelkow N, Rivkin JW (2005) Speed and Search: Designing Organizations for Turbulence and Complexity. *Organization Science* 16(2):101–122.

Simon A Herbert (1962) The Architecture of Complexity. *Proceedings of the American Philosophical Society* 106(6):467–482.

Wegner DM (1987) Transaction Memory: A Contemporary Analysis of the Group Mind. Mullen B, Goethals GR, eds., *Theories of Group Behavior*, 185–208 (New York: Springer-Verlag).