Supplementary Information for

Multi-disciplinary learning through collective performance favors decentralization

John Meluso and Laurent Hébert-Dufresne

Corresponding Author: John Meluso
E-mail: john.meluso@uvm.edu

This PDF file includes:
Supplementary text
Figs. S1 to S20
Tables S1 to S4
Legends for Dataset S1 to S10
SI References

Other supplementary materials for this manuscript include the following:
Datasets S1 to S10
Supporting Information Text

1. Task definitions

As we describe in the main text, we assume that each agent $i$ works to advance the team’s task $h(\vec{x})$ through their decisions $x_i \in [0, 1]$. Each agent receives feedback from a neighborhood function $g_i(x_i, \vec{x})$ that describes how well the agent performs on the team’s task given the mediated influences of their neighbors $j$. We assume that agents work toward the same goal as the team and so use the same task for $h(\vec{x})$ and each $g_i(x_i, \vec{x})$ during each simulation run of the model.

We constructed a series of tasks $y(\vec{x}_m)$, where $\vec{x}_m$ is a vector of $d$ decision variables $x_m$ such that $m \in M$. Each task is scalable, normalized, and commutative. By scalable, we mean that we can describe the task $y$ with the same equation form regardless of the number of inputs $d$. Consequently, $y$ is said to be $d$-dimensional. This scalability is true with many optimization test functions (1), like the Ackley function (see task ah), and enables us to make use of the function as a task regardless of decision variables we provide it as inputs while retaining the same task form.

Next, the tasks are normalized in that $y \in [0, 1]$ regardless of the number of inputs $d$. We did this to ensure that the outputs of the tasks were comparable to one another regardless of the number of inputs. Scaling in this way produces self-consistent outcomes when decision variables are applied to different neighborhoods and different levels (say when the operand of an agent $i$ exists in a neighborhood subtask $g_i$ versus a team task $h$). This occurs because scaling does not affect the quality of any value of $x_i$ relative to alternatives, only its magnitude. Thus, agents would make the same decisions regardless of scaling.

Then, the tasks are commutative in that the order of the inputs $m \in M$ does not matter and any decision variable $x_m$ can take any position in the task equation (with four exceptions, discussed later in this section). An average function of a normalized set of values is a common example of such a function that can accept any number of inputs, always returns outputs in the range of $[0, 1]$, and for which input order does not matter. Making tasks commutative ensures that agents do not have disproportionate influence compared to one another, thereby biasing results by favoring nodes that take certain positions in functions (we mitigate this for the four exceptions, discussed later in this section). Consequently, the functions describing each task at the team-level are functions $h(\vec{x}) = y(\vec{x})$ composed of all agents’ actions so that $\vec{x}_m = \vec{x}$. Then, the functions describing a task from the perspective of an agent’s neighborhood are $g_i(x_i, \vec{x}_j) = y(x_i, \vec{x})$ where $x_m = [x_i, x_{j=1}, \ldots, x_{j=m}]$.

Each of the following subsections describes one of the 34 tasks that we used in our model. Where present, $d$ is the total number of decision variables used as inputs for the task and $k_m$ is the number of network neighbors of agent $m$ (also called the degree). Other variables are defined in context.

Task functions that are unweighted are simple averages of the agents’ values and so divided by $d$. Functions that are degree-weighted are weighted averages of the agents’ values divided by $\sum_m k_m + 1$ in place of $\sum_m k_m$. We do this primarily because it avoids dividing by zero when an agent has no interactions with other agents, as with the empty graph and some random graphs. But as described in one of the footnotes in the main work, agents technically affect themselves and so have a directed self-edge, further justifying this increase in the number of edges by $d$ in the denominator. Functions with a uniform frequency use a consistent $7\pi/2$, while those with degree frequencies use $(1 + 2k_m)\pi/2$. Functions with uniform exponents use the degree of the first decision variable throughout the calculations, and with degree exponents use the degree of the respective agents.

Of note, the uniform exponent versions of the low-degree square high-degree root tasks (tasks $s \& u$), and the uniform exponent versions of the high-degree root, low-degree square tasks (tasks $w \& y$) are the exceptions to decision variable commutativity. Each of these four tasks includes a term anchored to the first index of $\vec{x}_m$. To counteract any biases introduced by adding tasks with a fixed term that favors either high or low degrees, these tasks are opposites of each other by design, with task $s$ opposite $w$, and task $u$ opposite to $y$. We also added a difficulty measure that accounts for the resulting divergence between agent interests which we discuss in Sec. 2.

\[ y(\vec{x}_m) = \frac{1}{d} \sum_m x_m \]  
\[ y(\vec{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) x_m \]  
\[ y(\vec{x}_m) = \frac{1}{d} \sum_m x_m^2 \]  
\[ y(\vec{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) x_m^2 \]
e. **Average of square roots function (unweighted).** See Fig. S1e on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{d} \sum_m x_m^{1/2}
\]  

[5]

f. **Average of square roots function (degree-weighted).** See Fig. S1f on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) x_m^{1/2}
\]  

[6]

g. **Sin^2 function (unweighted, uniform frequency).** See Fig. S1g on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{d} \sum_m \sin^2 \left( \frac{7\pi}{2} x_m \right)
\]  

[7]

h. **Sin^2 function (unweighted, degree frequency).** See Fig. S1h on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) \sin^2 \left( \frac{7\pi}{2} x_m \right)
\]  

[8]

i. **Sin^2 function (degree-weighted, uniform frequency).** See Fig. S1i on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) \sin^2 \left( \frac{1 + 2k_m}{2} \pi x_m \right)
\]  

[9]

j. **Sin^2 function (degree-weighted, degree frequency).** See Fig. S1j on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{\sum_m (k_m + 1)} \sum_m (k_m + 1) \sin^2 \left( \frac{1 + 2k_m}{2} \pi x_m \right)
\]  

[10]

k. **Sin^2 + square function (unweighted, uniform frequency).** See Fig. S1k on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2d} \sum_m \sin^2 \left( \frac{7\pi}{2} x_m \right) + x_m^2
\]  

[11]

l. **Sin^2 + square function (unweighted, degree frequency).** See Fig. S1l on page 6 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2d} \sum_m \sin^2 \left( \frac{1 + 2k_m}{2} \pi x_m \right) + x_m^2
\]  

[12]

m. **Sin^2 + square function (degree-weighted, uniform frequency).** See Fig. S2m on page 7 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2 \sum_m (k_m + 1)} \sum_m (k_m + 1) \left( \sin^2 \left( \frac{7\pi}{2} x_m \right) + x_m^2 \right)
\]  

[13]

n. **Sin^2 + square function (degree-weighted, degree frequency).** See Fig. S2n on page 7 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2 \sum_m (k_m + 1)} \sum_m (k_m + 1) \left( \sin^2 \left( \frac{1 + 2k_m}{2} \pi x_m \right) + x_m^2 \right)
\]  

[14]

o. **Sin^2 + square root function (unweighted, uniform frequency).** See Fig. S2o on page 7 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2d} \sum_m \sin^2 \left( \frac{7\pi}{2} x_m \right) + x_m^{1/2}
\]  

[15]

p. **Sin^2 + square root function (unweighted, degree frequency).** See Fig. S2p on page 7 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\bar{x}_m) = \frac{1}{2d} \sum_m \sin^2 \left( \frac{1 + 2k_m}{2} \pi x_m \right) + x_m^{1/2}
\]  

[16]
q. Sin²+ square root function (degree-weighted, uniform frequency). See Fig. S2q on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{2\sum_m (k_m + 1)} \sum_m (k_m + 1) \left( \sin^2 \left( \frac{7\pi}{2} x_m \right) + x_m^{1/2} \right)$$ [17]

r. Sin²+ square root function (degree-weighted, degree frequency). See Fig. S2r on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{2\sum_m (k_m + 1)} \sum_m (k_m + 1) \left( \sin^2 \left( \frac{(1 + 2k_m) \pi}{2} x_m \right) + x_m^{1/2} \right)$$ [18]

s. Low-degree square, high-degree root function (unweighted, uniform exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2s on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{d} \sum_m x_m^{c_m}$$ [19]

t. Low-degree square, high-degree root function (unweighted, degree exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2t on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{d} \sum_m x_m^{c_m}$$ [20]

u. Low-degree square, high-degree root function (degree-weighted, uniform exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2u on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \sum_m (k_m + 1) \sum_m (k_m + 1) x_m^{2c_m}$$ [21]

v. Low-degree square, high-degree root function (degree-weighted, degree exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2v on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \sum_m (k_m + 1) \sum_m (k_m + 1) x_m^{c_m}$$ [22]

w. High-degree square, low-degree root function (unweighted, uniform exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2w on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{d} \sum_m x_m^{-c_m}$$ [23]

x. High-degree square, low-degree root function (unweighted, degree exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S2x on page 7 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \frac{1}{d} \sum_m x_m^{-c_m}$$ [24]

y. High-degree square, low-degree root function (degree-weighted, uniform exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S3y on page 8 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \sum_m (k_m + 1) \sum_m (k_m + 1) x_m^{-c_m}$$ [25]

z. High-degree square, low-degree root function (degree-weighted, degree exponents). Here, $c_m = 1 - \frac{2k_m}{1 + \sum_m k_m}$. See Fig. S3z on page 8 for a 2-variable example where $k_1 = 1$ and $k_2 = 5$.

$$y(\bar{x}_m) = \sum_m (k_m + 1) \sum_m (k_m + 1) x_m^{-c_m}$$ [26]
aa. **Maximum function.** See Fig. S3aa on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \max (x_m) \tag{27}
\]

\[y(\vec{x}_m) = \text{max} (x_m)\]

ab. **Minimum function.** See Fig. S3ab on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \min (x_m) \tag{28}
\]

ac. **Median function.** See Fig. S3ac on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \text{median} (x_m) \tag{29}
\]

ad. **\(K+1\) power function (unweighted).** See Fig. S3ad on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \frac{1}{d} \sum_m x_m^{k_m+1} \tag{30}
\]

ae. **\(K+1\) power function (degree-weighted).** See Fig. S3ae on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \frac{1}{d} \sum_m (k_m + 1) x_m^{k_m+1} \tag{31}
\]

af. **\(K+1\) Root function (unweighted).** See Fig. S3af on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \frac{1}{d} \sum_m x_m^{\frac{1}{k_m+1}} \tag{32}
\]

ag. **\(K+1\) Root function (degree-weighted).** See Fig. S3ag on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = \frac{1}{d} \sum_m (k_m + 1) x_m^{\frac{1}{k_m+1}} \tag{33}
\]

ah. **Ackley function.** For this implementation, we invert the Ackley function from its traditional concave form (for minimizing) to a convex form (for maximizing). Here, \( c_1 = 20 \), \( c_2 = 0.2 \), and \( c_3 = 7\pi \). See Fig. S3ah on page 8 for a 2-variable example where \( k_1 = 1 \) and \( k_2 = 5 \).

\[
y(\vec{x}_m) = 1 - \left( -c_1 \exp \left( -c_2 \sqrt{\frac{1}{k_i+1} \sum_m x_m^2} \right) - \exp \left( \frac{1}{k_i+1} \sum_m \cos (c_3 x_m) \right) + \exp (1) + c_1 \right) \frac{c_1 (1 - \exp (-c_2)) + (\exp (1) - \exp (-1))}{c_1 (1 - \exp (-c_2)) + (\exp (1) - \exp (-1))} \tag{34}
\]
Fig. S1. Tasks (a–l).
(m) \( \sin^2 + \text{square function} \) (degree-weighted, uniform frequency)

(n) \( \sin^2 + \text{square function} \) (degree-weighted, degree frequency)

(o) \( \sin^2 + \text{square root function} \) (unweighted, uniform frequency)

(p) \( \sin^2 + \text{square root function} \) (unweighted, degree frequency)

(q) \( \sin^2 + \text{square root function} \) (degree-weighted, uniform frequency)

(r) \( \sin^2 + \text{square root function} \) (degree-weighted, degree frequency)

(s) \( \text{Low-degree square, high-degree root function} \) (unweighted, uniform exponents)

(t) \( \text{Low-degree square, high-degree root function} \) (unweighted, degree exponents)

(u) \( \text{Low-degree square, high-degree root function} \) (degree-weighted, uniform exponents)

(v) \( \text{Low-degree square, high-degree root function} \) (degree-weighted, degree exponents)

(w) \( \text{High-degree square, low-degree root function} \) (unweighted, uniform exponents)

(x) \( \text{High-degree square, low-degree root function} \) (unweighted, degree exponents)

Fig. S2. Tasks (m-x).
(y) High-degree square, low-degree root function (degree-weighted, uniform exponents)

(z) High-degree square, low-degree root function (degree-weighted, degree exponents)

(aa) Maximum function

(ab) Minimum function

(ac) Median function

(ad) K+1 power function (unweighted)

(af) K+1 Root function (unweighted)

(age) K+1 Root function (degree-weighted)

(ah) Ackley function

Fig. S3. Tasks (y-ah).
2. Task measures

We constructed four measures to control for different qualities that describe tasks. These qualities are grounded in the different types of tasks identified by McGrath (2), which underlie studies of collective intelligence (3, 4) and are recommended for the study of transactive memory systems (5, 6). The four qualities are exploration difficulty, exploitation difficulty, neighborhood interdependence, and neighborhood alignment. An exploration difficulty (1 – the integral of the task) measures the probability of finding better-performing solutions over the entire decision space. An exploitation difficulty (the number of peaks in the task) measures the likelihood that the team will randomly land on a peak in the task landscape with the best value. An interdependence difficulty (the average correlation between agents’ neighborhood performances) measures how likely a change in one agent’s neighborhood is to affect the neighborhood performance of another agent. Last, an alignment difficulty (the average difference between agents’ neighborhood performances) measures how likely agents’ changes are to produce commensurate changes for other agents. The follow sections motivate and define these measures.

a. Exploration difficulty: 1 - integral of the task. Exploration difficulty resembles the difficulty of generating new ideas or plans creatively (2). In our formulation, when agents can explore the entire decision space (search radius \( r = 1.0 \)), the relative position of each \( x_i \) does not matter. Random selection of each \( x_i \) effectively yields a probability distribution for different performance values. As a result, tasks that are more difficult for exploration are those for which there is a lower probability of finding a good performance value. In this work, performance values are “better” the closer they are to 1. So, we can measure the exploration difficulty of each task by calculating its expected value (through integration) over every \( x_i \in X \) on the domain [0, 1] and subtracting it from 1 as follows:

\[
q_1(h(\vec{x})) = 1 - \int_X h(\vec{x})d\vec{x}.
\]

We calculated the integral for each task through Monte Carlo integration, that is, by generating 100 random vectors \( \vec{x} \), calculating \( h(\vec{x}) \) for each, and averaging their results. Subtracting the resulting value from one gave us the exploration difficulty.

b. Exploitation difficulty: number of peaks in task. This quality derives from choose tasks (2). Tasks with more peaks are often described as making it more difficult or complex to select the best answer (7–9), particularly when individuals are limited to moving to adjacent locations (as with a search radius \( r = 0.01 \)). So the second difficulty measure we define is the number of peaks in each task. The tasks we defined above are simple enough that we can analytically determine the number of peaks in each task by their shapes and frequencies. Many tasks only have one peak, or have a continuous ridge of best values (as with the maximum function, task aa) for which we set the number of peaks to 1 as well. On the other hand, those with sinusoidal forms can have many peaks in each dimension, which can yield \( 10^{30} \) of more peaks for networks with large numbers of connections and degree frequencies. So our second measure of difficulty is:

\[
q_2(h(\vec{x})) = \text{number of peaks.}
\]

c. Interdependence between task neighborhoods. Execution tasks involve “coordination between members” (2, p.65). Tasks in which individuals actions are more interdependent require greater coordination, while those in which individuals actions are less interdependent require less coordination. Thus, we also measure how interdependent agents are by calculating the Spearman correlation coefficient \( r_s \) (10) between every pair of agents and averaging those coefficients. This captures how much agents affect each other when they make a change, and therefore measures how much control each agent has over their own outcomes given the influences of the others they interact with through mediation. This gives:

\[
q_3(h(\vec{x})) = \frac{1}{n^2} \sum_{i,j} r_s(g_i, g_j)
\]

Again, we generated 100 random vectors \( \vec{x} \) and entered these vectors into the neighborhood functions \( g_i \) and \( g_j \) to generate \( r_s \) for each pair of nodes.

d. Alignment Between Task Neighborhoods. Last, agents actions need not be aligned with one another. Many tasks involve negotiation to resolve conflicts or advance mixed motives (2). In our formulations, a great change for one agent might produce only small gains for their neighbor. Or in a few cases, an improvement for one agent could result in degraded performance for their neighbor, potentially leading to adaptive dynamics where agents compete (only the non-commutative tasks—that is, the high- and low-degree swinging tasks \( s, u, w, \) & \( y \)—see this among the tasks we constructed above). While the intent of this work was not primarily to explore these dynamics, we controlled for misalignment between agent interests by measuring how different their neighborhood performances are from one another on average:

\[
q_4(h(\vec{x})) = \frac{1}{n^2} \sum_{i,j} \left( 1 - \frac{1}{v} \sum_{v} |g_i - g_j| \right)
\]

where agents that are completely aligned have paired values of 1, and completely misaligned have values of 0. Here, too, we generated \( v = 100 \) random vectors \( \vec{x} \) and entered these vectors into the neighborhood functions \( g_i \) and \( g_j \) to generate the mean term for each pair of nodes.
3. Network measures

We measured common properties of network structure (11) for each team using algorithms provided by the python package NetworkX (12). Degree centrality (mean and standard deviation) measures how connected individuals are to one another (11). Next, eigenvector centrality and nearest neighbor degree (mean and standard deviation of each) measure how connected individuals are to individuals who are themselves highly-connected (11). For eigenvector centrality specifically, high means indicate evenly distributed connectedness and decentralized networks, while high standard deviations indicate varied connectedness and centralization. The mean of the shortest path length, betweenness centrality (mean and standard deviation), and diameter measure how efficiently information can flow from each node to each other node. And last, the clustering coefficient and assortativity measure how often individuals are connected to the same individuals.

Below, we specify the NetworkX algorithm we used and any details for each measure and group them by aspects of the network that they measure. We used each algorithm’s default values unless otherwise specified. Averages and standard deviations were calculated by taking the mean and standard deviation over all nodes in the team for that run of the model. All measures are valid for any network except assortativity, shortest path length, and diameter, which are only defined on completely connected graphs.

- Measures of individual connectedness
  - Degree centrality (mean and standard deviation)
    * Mean describes how well-connected individuals are on average
    * Standard deviation describes how much variation there is in individuals’ connectedness
    * Measured with `algorithms.centrality.degree_centrality`
    * Valid on connected and disconnected graphs
  - Eigenvector centrality (mean and standard deviation)
    * Describes how centralized a network is (13)
    * Mean eigenvector centrality describes how decentralized a network is because higher mean values indicate that most or all individuals are nearly as important as one another.
    * Higher standard deviations represent how centralized a network is because higher values correspond to a fewer individuals holding more importance than most others
    * Measured with `algorithms.centrality.eigenvector_centrality` (tolerance: 0.001, max iterations: 1000)
    * Valid on connected and disconnected graphs
  - Nearest neighbor degree (mean and standard deviation)
    * Mean describes how well-connected an individual’s neighbors are on average
    * Standard deviation describes how much variation there is in how well-connected an individual’s neighbors are
    * Measured with `algorithms.assortativity.average_neighbor_degree`
    * Valid on connected and disconnected graphs

- Measures of network efficiency
  - Betweenness centrality (mean and standard deviation)
    * Mean indicates how often individuals are intermediaries on the shortest path between two individuals
    * Standard deviation indicates variation in the number of intermediaries between individuals in the network
    * Measured with `algorithms.centrality.betweenness_centrality`
    * Valid on connected and disconnected graphs
  - Shortest path length
    * Measured with `algorithms.shortest_paths.generic.average_shortest_path_length`
    * Only valid on connected graphs
  - Diameter
    * Measured with `algorithms.distance_measures.diameter`
    * Only valid on connected graphs

- Measures of shared connections
  - Degree assortativity
    * Measured with `algorithms.assortativity.degree_assortativity_coefficient`
    * Only valid on connected graphs
  - Clustering coefficient
    * Measured with `algorithms.cluster.average_clustering`
    * Valid on connected and disconnected graphs
4. Network definitions

In this work, we constructed 12 different types of networks (three of which included probabilistic variants) of four different sizes, \( n \in \{4,9,16,25\} \). To select networks to include in this study, we began by compiling a list of networks widely used in previous works on teams and organizations. These included the complete graph, the empty graph, preferential attachment graphs, random graphs, small world graphs, the star graph, the tree graph, and the ring of cliques \((7–9, 14–17)\). To introduce greater variation in network properties, we supplemented this list with networks provided in the python package NetworkX (12) and suggested by colleagues, adding the ring graph, the rook’s graph, the wheel graph, and the windmill graph. Below, we briefly describe the constructions of these networks. See Fig. S4 on page 12 for a graphical representation of each network and each size team.

**Complete graph.** A network in which every node is connected to every other node.

**Empty graph.** A network in which no nodes are connected to any other nodes. In this work, an empty graph represents individual learning.

**Random graph.** A randomly-generated network in which a probability \( p \) specifies the likelihood that an edge exists between any two nodes.

**Ring graph.** A network in which each node is connected to two other nodes in a cycle, roughly forming the shape of a ring for large \( n \).

**Small world graph.** A ring network in which each node is connected to its \( k \) nearest neighbors. Then, each edge is rewired to another random node with probability \( p \). For our study, we created small world graphs of \( k = 2 \), but with three different probabilities of rewiring \( p \in \{0.1, 0.5, 0.9\} \).

**Preferential attachment graph.** A preferential attachment network generated using the Holme-Kim algorithm (18). This algorithm places new nodes \( i \) in the graph, and connects each \( i \) to existing nodes \( j \) by placing \( m \) edges following traditional preferential attachment. Then, for each neighbor \( j \), there is a probability \( p \) that \( i \) will be connected to one of \( j \)’s neighbors by adding a new edge to form a triangle. For this work, we used \( m = 2 \) and \( p \in \{0.1, 0.5, 0.9\} \).

**Ring of cliques graph.** This network is a set of completely-connected groups or “cliques” where two members of each clique have an additional edge connecting them to another clique. The cliques are connected in a ring-like pattern. Our implementation places \( \sqrt{n} \) nodes in each of \( \sqrt{n} \) cliques for each team size \( n \), which partially motivated our selection of the values of \( n \) we used. At \( n = 4 \), the ring of cliques is identical to a ring graph with \( n = 4 \), but the distinction becomes meaningful thereafter. Teams of \( n = 9 \) produce 3 connected cliques of 3; \( n = 16 \) produces 4 cliques of 4, and \( n = 25 \) produces 5 cliques of 5.

**Rook’s graph.** A network in which nodes are placed in a square grid. Every node is connected to every other node in its column with an edge, and likewise every node is connected to every other node in its row with an edge. Traditionally, each node metaphorically represents a location on a chess board and each edge represents the possible moves that a rook could make from each position on the board, though the board can be of arbitrary size (19). In this work, we created rook’s graphs for each value of \( n \) where the length of each side is \( \sqrt{n} \), corresponding to teams placed in \( 2 \times 2 \) grids (again, a simple ring), \( 3 \times 3 \) grids, \( 4 \times 4 \) grids, and a \( 5 \times 5 \) grids, again with every node connected to all other nodes in their same row and column.

**Star graph.** A network of \( n \) nodes in which \( n - 1 \) nodes are each connected to one central node, and only to that central node.

**Tree graph.** A network in which new nodes are randomly connected to an existing node when placed, creating a source and “branches” or “roots.”

**Wheel graph.** This network is a combination of both the ring graph and a star graph. One central node is connected to \( n - 1 \) nodes, and those \( n - 1 \) nodes are connected to one another in a ring.

**Windmill graph.** A network in which \( c \) completely-connected cliques, each of \( k \) nodes, are all connected to one another via one node which is a member of every clique. This creates an effect where networks look like sails connected to a shaft, like a windmill. We constructed windmills of \( c = \sqrt{n} + 1 \) cliques where each clique has \( k = \sqrt{n} \) nodes. This created teams of \( n = 4 \) with 3 cliques of 2 nodes; \( n = 9 \) with 4 cliques of 3 nodes; \( n = 16 \) with 5 cliques of 4 nodes; and \( n = 25 \) with 6 cliques of 5 nodes.
Fig. S4. Networks for each team size used in the model.
5. Performance by task and network

The heatmaps and bar graphs in Figs. S5-S8 (pages 14-17) show performance results for teams of \( n = 9 \) agents. Each figure shows results for one search radius \( r \). Within each figure, the heatmap shows the performance of teams with each network on each task, averaged over all 250 runs and all timesteps. The top bar graphs shows the mean and standard deviation of team performance for each network type compared to the performance averaged over all networks (with the same number of agents and the same search radius). The right bars show the mean and standard deviation of team performance for each task compared to the performance averaged over all tasks.

6. Average performances by network and search radius

As with the average performance figure in the main text, Fig. S9 contains the average performance of teams of \( n \in \{4, 9, 16, 25\} \). The figures corroborate that dense teams tend to perform better on local “exploitation” searches while sparse teams tend to perform better on global “exploration” searches. This trend is less clear for small teams (\( n = 4 \)), but network structures are less distinct from one another at this scale (see Fig. S4 for examples). Otherwise, the trend appears to hold across sizes. Here, we also include results for \( r = 0.001 \). This radius is much smaller than the smallest feature of any task and so functionally acts as a random starting point with slight perturbations. As expected, this search radius yielded random results for small teams, and mirrored the results of \( r = 0.01 \) for larger teams where the greater potential of interaction gradually increases the odds of affecting performance.
Fig. SS. Network task performance, teams of $n = 9$, search radius of $r = 0.001$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S6. Network task performance, teams of $n = 9$, search radius of $r = 0.01$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S7. Network task performance, teams of $n = 9$, search radius of $r = 0.1$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S8. Network task performance, teams of \( n = 9 \), search radius of \( r = 1.0 \). (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S9. The relative average performances compared to an empty graph baseline for networks of $n \in \{4, 9, 16, 25\}$. 
7. Random forest analysis details

We conducted two random forest regression analyses using `ensemble.RandomForestRegressor` from Scikit-learn (20) with the default parameters. One random forest included completely-connected networks and all network measures, and the other included all networks and network measures that are still valid on disconnected graphs.

We measured the contribution of the task measures defined in Sec. 2 and included a fixed effect variable for each task. Then, we normalized each of the 12 network measures described in Sec. 3 to the range $[0, 1]$. We added one variable for each measure to estimate how much different qualities affect performance. The random forests also measured the effects of several control variables. We measured team progress over time through a variable for the time step of each run. Then, while the tasks themselves mostly control for team size through their normalized ranges, we added a variable for team size to control for any additional effects (which were negligible in both random forests). We also included a variable for the search radius to control for exploration abilities not attributable to network properties.

See Fig. S10 on page 20 and Tab. S1 on page 21 for the results of each.

8. Regression analysis details

We estimated the effects that different task and network measures have on team performance through multivariate ordinary least squares (OLS) regressions. Many of these measures are correlated (Fig. S11 on page 22), a known challenge in networks research (21). We overcame this by performing six regressions with different controls. We performed three regressions using only completely-connected networks and all network measures (Tab. S2 on page 23), and another three with all networks and network measures that are still valid on disconnected graphs (Tab. S3 on page 24). Within each set of three, we ran one regression with only task measures, one with only task fixed effects (that is, with $w - 1 = 33$ dummy variables for the $w = 34$ tasks), and one with both task measures and task fixed effects.

For these regressions, we used the python package Statsmodels (22). Each regression used robust standard errors (type HC2). We controlled for team progress over time through a variable for the $\log_{10}$ of the time step of each run, adding one to the time step to avoid the undefined value at $t = 0$. Then, while the tasks themselves mostly control for team size through their normalizations, we added variables for team size and the square of team size to ensure we controlled for any additional effects (which were negligible in all regressions). We included a variable for the $\log_{10}$ of the search radius to control for exploration abilities not attributable to network properties. The regressions included all four search radii for thoroughness.

After including these controls, we normalized each of the 12 network measures described in Sec. 3 to the range $[0, 1]$. We added one variable for each measure to estimate how much different qualities affect performance (as described in the Results section of the main text). In preliminary regressions, we included dummy variables for the different search radii. However, the statistically significant network measures varied approximately linearly with the $\log_{10}$ of the search radius. Consequently, the regressions we show here included one interaction term between each measure and the $\log_{10}$ of the search radius to measure how much the search radius (and exploration-exploitation) moderates the effect of each network measure on performance.

9. Network statistics

Tab. S4 on page 25 displays the average value of each network measure for each type of network used in our study for teams of $n = 9$ individuals.
Fig. S10. Random forest feature importances. Error bars display the range of one standard deviation. The most important measures for team performance were the task difficulty measures. This was followed by the inefficiency of the network (measured by the mean and standard deviation of betweenness centrality) which is within one standard deviation of decentralization (measured by mean eigenvector centrality), then by variation in the connectedness of one's neighbors (via the standard deviation of nearest neighbor degree), and shared connections with neighbors (assortativity).
## Table S1. Random forest feature importances with connected graphs and all graphs. Parenthetical values are standard deviations.

|                  | Connected graphs | All graphs |
|------------------|------------------|------------|
| Observations     | 55,342,898       | 63,648,000 |
| Controls         |                  |            |
| Time step        | 0.104252 (0.000075) | 0.105389 (0.000070) |
| Team size        | 0.000354 (0.000020) | 0.000752 (0.000069) |
| Search radius    | 0.361612 (0.000096) | 0.369159 (0.000092) |
| Task measures    |                  |            |
| Exploration difficulty (1 - Task integral) | 0.432379 (0.000146) | 0.427343 (0.000133) |
| Exploitation difficulty (Number of peaks) | 0.022157 (0.000162) | 0.019545 (0.000118) |
| Neighborhood alignment | 0.025309 (0.000061) | 0.023335 (0.000302) |
| Neighborhood interdependence | 0.025691 (0.000056) | 0.023416 (0.000044) |
| Network measures |                  |            |
| Degree Cent. (Mean) | 0.000540 (0.000009) | 0.001536 (0.000322) |
| Degree Cent. (St. Dev.) | 0.001223 (0.000013) | 0.001569 (0.000017) |
| Eigenvector Cent. (Mean) | 0.002512 (0.000028) | 0.003149 (0.000043) |
| Eigenvector Cent. (St. Dev.) | 0.001959 (0.000013) | 0.002480 (0.000021) |
| Betweenness Cent. (Mean) | 0.003067 (0.000041) | 0.002665 (0.000033) |
| Betweenness Cent. (St. Dev.) | 0.002599 (0.000018) | 0.002793 (0.000017) |
| Nearest Neighbor Degree (Mean) | 0.001779 (0.000015) | 0.0022476 (0.000038) |
| Nearest Neighbor Degree (St. Dev.) | 0.002269 (0.000016) | 0.002827 (0.000016) |
| Clustering Coeff. | 0.001688 (0.000032) | 0.002020 (0.000029) |
| Degree Assortativity | 0.002522 (0.000016) |            |
| Shortest Path Length (Mean) | 0.001277 (0.000013) |            |
| Diameter         | 0.000493 (0.000006) |            |
Fig. S11. Correlation coefficients for each combination of network and task measures.
| Measure groups | Task controls | Network measures | Measures only | Fixed effects only | Meas. & Fixed eff. |
|----------------|--------------|------------------|---------------|-------------------|-------------------|
| **Controls**   | Intercept    |                   | 2.37* (0.00121) | 0.806* (0.00355) | 1.51* (0.00135)  |
|                | log_{10}(Time step + 1) |   | 0.166* (5.26e-05) | 0.166* (5.3e-05) | 0.166* (5.24e-05) |
|                | Team size    |                   | 0.000915* (4.03e-05) | -0.00145* (3.9e-05) | -0.000871* (3.9e-05) |
|                | Team size^2  |                   | -4.59e-05* (8.95e-07) | 2.09e-05* (8.74e-07) | 5.44e-06* (8.63e-07) |
|                | log_{10}(Search radius) | | 0.43* (0.000622) | 0.211* (0.000489) | 0.43* (0.000602) |
| **Network measures** | Degree Cent. (Mean) | | 0.145* (0.000367) | -0.0283* (0.000327) | 0.0429* (0.000362) |
|                | Degree Cent. (St. Dev.) | | -0.02296* (0.000286) | 0.00711* (0.000251) | 0.00494* (0.000250) |
|                | Eigenvector Cent. (Mean) | | 0.0399* (0.000484) | 0.0183* (0.000471) | 0.0215* (0.000465) |
|                | Eigenvector Cent. (St. Dev.) | | -0.0173* (0.000541) | 0.00778* (0.000523) | 0.00378* (0.000521) |
|                | Betweenness Cent. (Mean) | | -0.067* (0.000292) | -0.014* (0.000274) | -0.0342* (0.000278) |
|                | Betweenness Cent. (St. Dev.) | | 0.0541* (0.000315) | 0.00633* (0.000293) | 0.0257* (0.000305) |
|                | Nearest Neighbor Degree (Mean) | | -0.12* (0.000312) | 0.00813* (0.000272) | 0.044* (0.000302) |
|                | Clustering Coeff. | | 0.0491* (0.000183) | -0.00109* (0.000135) | 0.0195* (0.000174) |
|                | Degree Assortativity | | 0.0338* (0.000441) | 0.00777* (0.000426) | 0.0103* (0.000427) |
|                | Diameter | | 0.0728* (0.000933) | -0.00214 (0.000888) | 0.0303* (0.000893) |
| **Task measures** | Exploration difficulty (1 - Task integral) | | -0.571* (0.000161) | -0.533* (0.000535) | -0.571* (0.000161) |
|                | log_{10}(Task number of peaks) | | 0.00379* (3.99e-06) | 0.00119* (5.07e-06) | 0.00119* (5.07e-06) |
|                | Neighborhood alignment | | -1.68* (0.00131) | -0.676* (0.00146) | -0.676* (0.00146) |
|                | Neighborhood interdependence | | 0.16* (0.00028) | 0.0637* (0.000258) | 0.0637* (0.000258) |
| **Interactions** | log_{10}(Search radius) × Degree Cent. (Mean) | | 0.0369* (0.000204) | 0.0106* (0.000199) | 0.0369* (0.000204) |
|                | log_{10}(Search radius) × Degree Cent. (St. Dev.) | | -0.0183* (0.000147) | -0.00274* (0.000147) | -0.00181* (0.000145) |
|                | log_{10}(Search radius) × Eigenvector Cent. (Mean) | | 0.0102* (0.000114) | 0.0112* (0.000115) | 0.0103* (0.000112) |
|                | log_{10}(Search radius) × Eigenvector Cent. (St. Dev.) | | -0.0063* (0.000283) | -0.00111* (0.000286) | -0.00647* (0.000283) |
|                | log_{10}(Search radius) × Betweenness Cent. (Mean) | | -0.0196* (0.000161) | -0.0126* (0.00016) | -0.0197* (0.00016) |
|                | log_{10}(Search radius) × Betweenness Cent. (St. Dev.) | | 0.017* (0.000176) | 0.01* (0.000175) | 0.0172* (0.000176) |
|                | log_{10}(Search radius) × Nearest Neighbor Degree (Mean) | | -0.0342* (0.000117) | -0.0151* (0.000165) | -0.0342* (0.000156) |
|                | log_{10}(Search radius) × Nearest Neighbor Degree (St. Dev.) | | 0.0012* (0.000198) | 0.00021 (0.000202) | 0.0019* (0.000197) |
|                | log_{10}(Search radius) × Clustering Coeff. | | 0.0132* (9.94e-05) | 0.00563* (8.92e-05) | 0.0132* (9.78e-05) |
|                | log_{10}(Search radius) × Degree Assortativity | | 0.009* (0.000254) | 0.00328* (0.000257) | 0.0091* (0.000254) |
|                | log_{10}(Search radius) × Shortest Path Length (Mean) | | 0.0322* (0.000466) | 0.0222* (0.000465) | 0.0323* (0.000464) |
|                | log_{10}(Search radius) × Diameter | | -0.0214* (0.000512) | -0.0113* (0.000513) | -0.0214* (0.000511) |
|                | log_{10}(Search radius) × Exploration difficulty (1 - Task integral) | | 0.171* (8.26e-05) | 0.198* (7.81e-05) | 0.171* (8.16e-05) |
|                | log_{10}(Search radius) × log_{10}(Task number of peaks) | | -0.000562* (2.09e-06) | -0.000292* (1.65e-06) | -0.000563* (1.94e-06) |
|                | log_{10}(Search radius) × Neighborhood alignment | | -0.459* (0.000699) | -0.213* (0.000551) | -0.459* (0.000676) |
|                | log_{10}(Search radius) × Neighborhood interdependence | | 0.0433* (0.000152) | 0.0203* (0.000107) | 0.0432* (0.000147) |

| Statistics | Num. Observations | 55342898.0 | 55342898.0 | 55342898.0 |
|            | R-squared (adj.) | 0.783937 | 0.79167 | 0.797693 |
|            | AIC | -8.67871e+07 | -8.8804e+07 | -9.04275e+07 |
|            | BIC | -8.67865e+07 | -8.8803e+07 | -9.04264e+07 |
|            | F-statistic | 6.24321e+06 | 3.73595e+06 | 3.8605e+06 |
|            | F p-value | < 0.001 | < 0.001 | < 0.001 |

Table S2. Regression coefficients for models 1-3. Parenthetical statements show HC2 standard errors. Coefficients marked with an asterisk (*) have \( p < 0.001 \), and all others have \( p > 0.01 \).
## Table S3. Regression coefficients for models 4-6. Parenthetical statements show HC2 standard errors. Coefficients marked with an asterisk (*) have $p < 0.001$, and all others have $p > 0.01$.  

| Measure groups | Task controls | Measures only | Fixed effects only | Meas. & Fixed eff. only |
|----------------|---------------|---------------|--------------------|-------------------------|
| **Controls**   |               |               |                    |                         |
| Intercept      | 0.98* (0.000326) | 0.609* (0.00031) | 0.935* (0.000396) |
| log$_{10}$ (Time step + 1) | 0.166* (4.93e-05) | 0.166* (4.99e-05) | 0.166* (4.9e-05) |
| Team size      | -0.000268* (3.68e-05) | -0.000858* (3.58e-05) | -0.000699* (3.53e-05) |
| Team size$^2$  | -1.74e-05* (8.22e-07) | 9.24e-06* (7.99e-07) | 2.84e-06* (7.84e-07) |
| log$_{10}$ (Search radius) | 0.0456* (7.05e-05) | 0.0253* (6.01e-05) | 0.0456* (7.03e-05) |
| **Network measures** |               |               |                    |                         |
| Degree Cent. (Mean) | -0.0468* (0.000266) | -0.0263* (0.000254) | -0.0342* (0.000246) |
| Degree Cent. (St. Dev.) | 0.0161* (0.000245) | -0.011* (0.000222) | 0.00208* (0.000226) |
| Eigenvector Cent. (Mean) | 0.0109* (0.000428) | 0.0179* (0.000279) | 0.0073* (0.000319) |
| Betweenness Cent. (Mean) | 0.0215* (0.000212) | -0.00243* (0.000142) | 0.00884* (0.000197) |
| Betweenness Cent. (St. Dev.) | -0.0207* (0.000247) | 0.00693* (0.000211) | 0.0137* (0.000224) |
| Nearest Neighbor Degree (Mean) | -0.000268* (3.92e-06) | 0.00163* (0.000267) | -0.00542* (0.000263) |
| Clustering Coeff. | -0.0118* (0.000152) | -0.000998* (0.000114) | -0.00579* (0.000141) |
| Degree Assortativity | 0.0392* (0.000204) | 0.00429* (0.000192) | 0.00426* (0.000202) |
| Shortest Path Length (Mean) | 0.00314* (3.92e-06) | -0.000977* (4.69e-06) | 0.00208* (0.000226) |
| Diameter | -0.531* (0.000143) | -0.511* (0.000396) | -0.5158* (0.000145) |
| **Task measures** |                |               |                    |                         |
| Exploration difficulty (1 - Task integral) | -0.000255* (0.000136) | -0.0029* (0.000133) | -0.00225* (0.000134) |
| log$_{10}$ (Task number of peaks) | 0.0416* (8.2e-05) | 0.00663* (7.89e-05) | 0.00418* (8.1e-05) |
| Neighborhood alignment | -0.00419* (0.000138) | 0.00191* (0.000133) | -0.00405* (0.000116) |
| Neighborhood interdependence | -0.00224* (0.000163) | -0.00554* (0.000164) | 0.000225* (0.000162) |
| log$_{10}$ (Search radius) × Degree Cent. (Mean) | -0.0158* (0.000147) | -0.013* (0.000149) | -0.0158* (0.000145) |
| log$_{10}$ (Search radius) × Degree Cent. (St. Dev.) | 0.00255* (0.000136) | 0.0029* (0.000133) | 0.00257* (0.000134) |
| log$_{10}$ (Search radius) × Eigenvector Cent. (Mean) | 0.0416* (8.2e-05) | 0.00663* (7.89e-05) | 0.00418* (8.1e-05) |
| log$_{10}$ (Search radius) × Eigenvector Cent. (St. Dev.) | 0.00279* (0.000162) | 0.00208* (0.000148) | 0.00261* (0.000161) |
| log$_{10}$ (Search radius) × Betweenness Cent. (Mean) | 0.00569* (0.000117) | 0.00159 (0.000101) | 0.00559* (0.000116) |
| log$_{10}$ (Search radius) × Betweenness Cent. (St. Dev.) | -0.00419* (0.000138) | 0.00191* (0.000133) | -0.00405* (0.000137) |
| log$_{10}$ (Search radius) × Nearest Neighbor Degree (Mean) | 0.00492* (0.000134) | 0.00153* (0.000103) | 0.00482* (0.000132) |
| log$_{10}$ (Search radius) × Nearest Neighbor Degree (St. Dev.) | -0.00224* (0.000163) | 0.00054* (0.000164) | -0.00225* (0.000162) |
| log$_{10}$ (Search radius) × Clustering Coeff. | -0.00357* (8.54e-05) | -0.00206* (7.67e-05) | -0.00361* (8.43e-05) |
| log$_{10}$ (Search radius) × Degree Assortativity | -0.00224* (0.000163) | 0.00054* (0.000164) | 0.000225* (0.000162) |
| log$_{10}$ (Search radius) × Shortest Path Length (Mean) | 0.00257* (0.000134) | 0.00261* (0.000161) | 0.00257* (0.000134) |
| log$_{10}$ (Search radius) × Diameter | -0.000268* (3.68e-05) | -0.00101* (1.58e-06) | -0.000739* (1.85e-06) |
| log$_{10}$ (Search radius) × Exploration difficulty (1 - Task integral) | 0.183* (7.45e-05) | 0.216* (7.27e-05) | 0.183* (7.49e-05) |
| log$_{10}$ (Search radius) × log$_{10}$ (Task number of peaks) | -0.000739* (2.02e-06) | -0.00101* (1.58e-06) | -0.000739* (1.85e-06) |
| log$_{10}$ (Search radius) × Neighborhood alignment | 0.00268* (0.000112) | -0.00163* (0.000112) | 0.00261* (0.000111) |
| log$_{10}$ (Search radius) × Neighborhood interdependence | 0.028* (0.000118) | 0.00877* (8.28e-05) | 0.0259* (0.000117) |
| **Statistics** |               |               |                    |                         |
| Num. Observations | 63648000.0 | 63648000.0 | 63648000.0 |
| R-squared (adj.) | 0.77444 | 0.758493 | 0.792898 |
| AIC | -9.75632e+07 | -1.00761e+08 | -1.02997e+08 |
| BIC | -9.75632e+07 | -1.00761e+08 | -1.02997e+08 |
| F-statistic | 8.46353e+06 | 4.39242e+06 | 4.66763e+06 |
| F p-value | < 0.001 | < 0.001 | < 0.001 |
| Network type                        | Degree Cent. (Mean) | Degree Cent. (St. Dev.) | Eigenvector Cent. (Mean) | Eigenvector Cent. (St. Dev.) | Betweenness Cent. (Mean) | Betweenness Cent. (St. Dev.) | Nearest Neighbor Degree (Mean) | Nearest Neighbor Degree (St. Dev.) | Clustering Coef. | Degree Assortativity | Shortest Path Length (Mean) | Diameter |
|------------------------------------|---------------------|-------------------------|--------------------------|----------------------------|--------------------------|-----------------------------|-------------------------------|-------------------------------|---------------------|----------------------|-----------------------------|-----------|
| Complete                           | 1.000               | 0.000                   | 0.200                    | 0.000                      | 0.000                    | 1.000                       | 0.000                         | 1.000                         | 0.000               | 0.000                | 1.000                       | 1.000     |
| Empty (Indiv. Learning)            | 0.000               | 0.000                   | 0.200                    | 0.000                      | 0.000                    | 0.000                       | 0.000                         | 0.000                         | 0.000               | 0.000                | 0.000                       | 0.000     |
| Pref. Attach. \((m = 2, p = 0.1)\) | 0.153               | 0.106                   | 0.175                    | 0.099                      | 0.056                    | 0.095                       | 0.253                         | 0.081                         | 0.321               | -0.305              | 2.289                       | 4.124     |
| Pref. Attach. \((m = 2, p = 0.5)\) | 0.153               | 0.111                   | 0.174                    | 0.101                      | 0.058                    | 0.110                       | 0.261                         | 0.081                         | 0.512               | -0.302              | 2.327                       | 4.284     |
| Pref. Attach. \((m = 2, p = 0.9)\) | 0.153               | 0.116                   | 0.173                    | 0.103                      | 0.060                    | 0.131                       | 0.271                         | 0.085                         | 0.686               | -0.315              | 2.387                       | 4.428     |
| Random \((p = 0.1)\)               | 0.101               | 0.059                   | 0.158                    | 0.123                      | 0.075                    | 0.087                       | 0.128                         | 0.050                         | 0.072               | -0.110              | 3.085                       | 6.917     |
| Random \((p = 0.5)\)               | 0.500               | 0.100                   | 0.196                    | 0.038                      | 0.022                    | 0.011                       | 0.521                         | 0.022                         | 0.501               | -0.085              | 1.501                       | 2.164     |
| Random \((p = 0.9)\)               | 0.901               | 0.059                   | 0.200                    | 0.012                      | 0.004                    | 0.001                       | 0.905                         | 0.006                         | 0.901               | -0.080              | 1.099                       | 2.000     |
| Ring                               | 0.083               | 0.000                   | 0.200                    | 0.000                      | 0.239                    | 0.000                       | 0.083                         | 0.000                         | 0.000               | 0.000                | 6.500                       | 12.000    |
| Ring of Clique                    | 0.183               | 0.021                   | 0.199                    | 0.020                      | 0.094                    | 0.118                       | 0.186                         | 0.002                         | 0.840               | -0.100              | 3.167                       | 5.000     |
| Rook’s Graph                     | 0.333               | 0.000                   | 0.200                    | 0.000                      | 0.292                    | 0.000                       | 0.333                         | 0.000                         | 0.429               | 0.000               | 1.667                       | 2.000     |
| Small World \((k = 2, p = 0.1)\)  | 0.083               | 0.017                   | 0.162                    | 0.111                      | 0.214                    | 0.131                       | 0.088                         | 0.012                         | 0.010               | -0.074              | 6.512                       | 16.238    |
| Small World \((k = 2, p = 0.5)\)  | 0.083               | 0.036                   | 0.155                    | 0.127                      | 0.149                    | 0.158                       | 0.103                         | 0.027                         | 0.024               | -0.220              | 5.176                       | 12.441    |
| Small World \((k = 2, p = 0.9)\)  | 0.083               | 0.042                   | 0.152                    | 0.131                      | 0.137                    | 0.161                       | 0.109                         | 0.033                         | 0.025               | -0.230              | 4.715                       | 10.952    |
| Star                              | 0.080               | 0.192                   | 0.167                    | 0.112                      | 0.040                    | 0.200                       | 0.962                         | 0.192                         | 0.000               | 1.000               | 1.920                       | 2.000     |
| Tree                              | 0.080               | 0.039                   | 0.158                    | 0.124                      | 0.183                    | 0.207                       | 0.104                         | 0.030                         | 0.000               | -0.250              | 5.198                       | 12.208    |
| Wheel                             | 0.160               | 0.175                   | 0.177                    | 0.095                      | 0.027                    | 0.174                       | 0.405                         | 0.058                         | 0.643               | -0.333              | 1.840                       | 2.000     |
| Windmill                          | 0.200               | 0.167                   | 0.181                    | 0.086                      | 0.035                    | 0.174                       | 0.367                         | 0.042                         | 0.965               | -0.250              | 1.800                       | 2.000     |

Table S4. Mean values of each network measure for each type of network for teams of \(n = 9\).
10. Team experiments model

a. Description. We ran a second experiment in which the members of the team experiment together rather than experimenting individually. Conceptually, the model was constructed and run the same except that agents perform the following three steps as they update their values:

1. All of the agents sample new values \( x'_i \).
2. Then, all agents evaluate their new neighborhood performance with their neighbors’ new values, so they check if
   \[ g'_i(x'_i, \vec{x}'_j) > g_i(x_i, \vec{x}_j) \]
3. Last, they accept the new value \( x'_i \) if
   \[ g'_i(x'_i, \vec{x}'_j) > g_i(x_i, \vec{x}_j) \].

The full pseudocode follows, along with figures containing the full results in Fig. S12-S20. The corresponding data can be found in SI Datasets S6-S10.

As noted in the main text, we do not see strong positive effects of mediated learning in this set-up, though we do see significant negative effects (Fig. S17). The relative importances of network properties also shifted in this case. While decentralization was still an important positive performer, intermediaries and clustering proved more important, albeit with less confidence in their positive effect (Fig. S20).

b. Model pseudocode.

```plaintext
1: Input Graph with vertices \( i \in I \), neighbors \( j \in J_i \) \( \forall i \in I \), neighborhood objective functions \( g_i(x_i, \vec{x}_j) \) \( \forall i \in I \) where \( \vec{x}_j = (x_{j=1}, \ldots, x_{j=k_i}) \), graph objective function \( h(\vec{x}) \) where \( \vec{x} = (x_{i=1}, \ldots, x_{i=n}) \), and initial positions \( x_i \) \( \forall i \in I \).
2: Output Performance values \( g_i \) \( \forall i \in I \) and \( h \).
3: for each time \( t \in \text{timesteps} \) do
4:    for each \( i \in I \) do
5:      Draw \( \Delta x \) from \( U(-\epsilon, \epsilon) \)
6:      \( x'_i := x_i + \Delta x \)
7:    end for
8:    for each \( i \in I \) do
9:      \( g_i = g_i(x_i, \vec{x}_j) \)
10:     \( g'_i := g_i(x'_i, \vec{x}'_j) \)
11:   end for
12:  for each \( i \in I \) do
13:    if \( g'_i > g_i \) then
14:      \( x_i := x'_i \)
15:    end if
16:  end for
17:  \( h = h(\vec{x}) \)
18: end for
```

11. Datasets Overview

The ten datasets for this work are available at https://osf.io/kyvttd. Datasets S1-S5 pertain to the “individual experiments” model and analyses found in the main text. Datasets S6-S10 pertain to the “team experiments” version contained here in the supplement. These files have all been compressed with the gzip compression algorithm for storage and can be used to recreate the figures throughout this work with the code provided at https://github.com/meluso/multi-disciplinary-learning after decompression. Please note that due to its size, Dataset S10 underwent additional compression with xz before undergoing gzip compression for submission, and so Dataset S10 will need to be decompressed with gzip followed by xz before use.
Fig. S12. Examples of team performances averaged over 250 runs for search radius $r = 0.1$ for team experiments.
Fig. S13. Team experiment network task performance, teams of $n = 9$, search radius of $r = 0.001$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S14. Team experiment network task performance, teams of $n = 9$, search radius of $r = 0.01$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S15. Team experiment network task performance, teams of $n = 9$, search radius of $r = 0.1$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S16. Team experiment network task performance, teams of $n = 9$, search radius of $r = 1.0$. (Center) A heatmap of the average performance, with one column for each network and one row for each task, after subtracting the team performance of the empty graph (individual learning case) on that task. (Top) Average performance of each network across all tasks with one standard deviation, after subtracting the average performance across all networks. (Right) Average performance on each task across all networks with one standard deviation, after subtracting the average performance across all tasks.
Fig. S17. Team experiment relative average performances compared to an empty graph baseline for networks of $n \in \{4, 9, 16, 25\}$.
Fig. S18. Team experiment random forest feature importances. Error bars display the range of one standard deviation. The most important measures for team performance were the task difficulty measures. This was followed by the inefficiency of the network (measured by the mean and standard deviation of betweenness centrality) which is within one standard deviation of decentralization (measured by mean eigenvector centrality), then by variation in the connectedness of one’s neighbors (via the standard deviation of nearest neighbor degree), and shared connections with neighbors (assortativity).
Fig. S19. Team experiment task measure importances, likelihoods, and effect directions. Each point shows how important the indicated measure is (x-axis), and plots those importances against how likely that measure is to either have a positive effect (positive y-axis) or negative effect (negative y-axis) on team performance. Each point summarizes the fraction of valid regression cumulative effects (combining main and search radii interactions) with a positive effect. For example, exploration difficulty was negative in all 18 cumulative effects, while neighborhood interdependence was positive in 10 of 18 cumulative effects, and negative in 8 of 18.
Fig. S20. Team experiment network measure importances, likelihoods, and effect directions. Subfigures show measures of (a) neighbors’ connectedness, (b) network efficiency, (c) shared connections, and (d) individual connectedness. Again, each point shows how important the indicated measure is (x-axis), and plots those importances against how likely that measure is to either have a positive effect (positive y-axis) or negative effect (negative y-axis) on team performance. Decentralization and having many intermediaries are tied for most important (within one standard deviation of each other’s means). NND stands for nearest neighbor degree.
## SI Dataset S1 (execset010_model3xx_team_is_nbhd.pickle.gz)

| Variable                                      | Description                                                                 |
|-----------------------------------------------|------------------------------------------------------------------------------|
| team_size                                     | Number of agents in the team                                                 |
| team_graph_type                               | The type of network                                                          |
| team_fn_type                                  | The type of task                                                             |
| agent_steplim                                 | The allowed search radius                                                    |
| run_step                                      | The run number indicating the time step in that execution of the model       |
| team_graph_centrality_degree_mean            | The mean of the degree centralities of the agents                           |
| team_graph_centrality_degree_stdev           | The standard deviation of the degree centralities of the agents              |
| team_graph_centrality_eigenvector_mean       | The mean of the eigenvector centralities of the agents                       |
| team_graph_centrality_eigenvector_stdev      | The standard deviation of the eigenvector centralities of the agents         |
| team_graph_betweenness_mean                  | The mean of the betweenness centralities of the agents                       |
| team_graph_betweenness_stdev                 | The standard deviation of the betweenness centralities of the agents         |
| team_graph_nearest_neighbor_degree_mean      | The mean of the nearest neighbor degrees of each agent                       |
| team_graph_nearest_neighbor_degree_stdev     | The standard deviation of the nearest neighbor degrees of each agent         |
| team_graph_clustering                         | The clustering coefficient of the team network                               |
| team_graph_assortativity                      | The degree assortativity of the team network                                 |
| team_graph_pathlength                         | The mean shortest path length of the team network                            |
| team_graph_diameter                           | The largest shortest path length of the team network                         |
| team_fn_diff_integral                         | The exploration difficulty of the task                                       |
| team_fn_diff_peaks                            | The exploitation difficulty of the task                                      |
| team_fn_alignment                             | The alignment between task neighborhoods                                     |
| team_fn_interdep                              | The interdependence between task neighborhoods                               |
| team_performance                              | The team's performance at the specific time                                  |
| team_productivity                             | The team productivity at that run step, measured as the difference between the performance at the current time and the prior time, then divided by the the team size |
| team_fn_weight                                | The weighting type used in the task, if applicable                          |
| team_fn_frequency                             | The frequency type used in the task, if applicable                           |
| team_fn_exponent                              | The exponent type used in the task, if applicable                            |
| team_fn                                       | A string combining the other team_fn_ fields                                 |

## SI Dataset S2 (execset010_stats.pickle.gz)

| Variable          | Description                                                                 |
|-------------------|------------------------------------------------------------------------------|
| ci_lo             | Lower bound of the 95% confidence interval for the specified statistic      |
| ci_hi             | Upper bound of the 95% confidence interval for the specified statistic       |
| diff_mean         | Absolute difference between the mean of the empty graph and the network type shown with all other parameters held fixed |
| diff_ci_lo        | Lower bound of the 95% confidence interval for the absolute difference between means |
| diff_ci_hi        | Upper bound of the 95% confidence interval for the absolute difference between means |
| pct_mean          | Percent difference between the mean of the empty graph and the network type shown with all other parameters held fixed |
| pct_ci_lo         | Lower bound of the 95% confidence interval for the percent difference between means |
| pct_ci_hi         | Upper bound of the 95% confidence interval for the percent difference between means |

## SI Dataset S3 (execset010_stats_by_graph.pickle.gz)

| Variable                                      | Description                                                                 |
|-----------------------------------------------|------------------------------------------------------------------------------|
| team_graph_diff_integral                      | The exploration difficulty of the task                                       |
| team_graph_diff_peaks                         | The exploitation difficulty of the task                                      |
| team_graph_alignment                          | The alignment between task neighborhoods                                     |
| team_graph_interdep                           | The interdependence between task neighborhoods                               |
| team_graph_performance                        | The team's performance at the specific time                                  |
| team_graph_productivity                       | The team productivity at that run step, measured as the difference between the performance at the current time and the prior time, then divided by the the team size |
| team_graph_weight                             | The weighting type used in the task, if applicable                          |
| team_graph_frequency                          | The frequency type used in the task, if applicable                           |
| team_graph_exponent                           | The exponent type used in the task, if applicable                            |
| team_graph                                          | A string combining the other team_graph_ fields                              |

## SI Dataset S4 (execset010_by_graph_vs_empty.pickle.gz)

## SI Dataset S5 (execset010_vs_empty.pickle.gz)

## SI Dataset S6 (execset001_model3xx_team_is_nbhd.pickle.gz)

## SI Dataset S7 (execset001_stats.pickle.gz)

## SI Dataset S8 (execset001_stats_by_graph.pickle.gz)

## SI Dataset S9 (execset001_by_graph_vs_empty.pickle.gz)

## SI Dataset S10 (execset001_vs_empty.pickle.xz.gz)
1. S Surjanovic, D Bingham, Virtual library of simulation experiments: Test functions and datasets (2013).

2. JE McGrath, Groups: Interaction and Performance. (Prentice-Hall Englewood Cliffs, NJ) Vol. 14, (1984).

3. AW Woolley, CF Chabris, A Pentland, N Hashmi, TW Malone, Evidence for a collective intelligence factor in the performance of human groups. *Science* **330**, 686–688 (2010).

4. C Riedl, YJ Kim, P Gupta, TW Malone, AW Woolley, Quantifying collective intelligence in human groups. *Proc. Natl. Acad. Sci.* **118** (2021).

5. K Lewis, B Herndon, Transactive memory systems: Current issues and future research directions. *Organ. Sci.* **22**, 1254–1265 (2011).

6. DG Bachrach, et al., Transactive memory systems in context: A meta-analytic examination of contextual factors in transactive memory systems development and team performance. *J. Appl. Psychol.* **104**, 464–493 (2019).

7. JW Rivkin, N Siggelkow, Patterned interactions in complex systems: Implications for exploration. *Manag. Sci.* **53**, 1068–1085 (2007).

8. D Barkoczi, M Galesic, Social learning strategies modify the effect of network structure on group performance. *Nat. Commun.* **7**, 13109–13109 (2016).

9. A Almaatouq, M Alsobay, M Yin, DJ Watts, Task complexity moderates group synergy. *Proc. Natl. Acad. Sci.* **118** (2021).

10. P Virtanen, et al., SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods* **17**, 261–272 (2020).

11. M Newman, *Networks*. (Oxford university press). (2018).

12. AA Hagberg, DA Schult, PJ Swart, Exploring network structure, dynamics, and function using NetworkX in *Proceedings of the 7th Python in Science Conference*, eds. G Varoquaux, T Vaught, J Millman. (Pasadena, CA USA), pp. 11–15 (2008).

13. J Shore, E Bernstein, D Lazer, Facts and figuring: An experimental investigation of network structure and performance in information and solution spaces. *Organ. Sci.* **26**, 1432–1446 (2015).

14. WS Lovejoy, A Sinha, Efficient structures for innovative social networks. *Manag. Sci.* **56**, 1127–1145 (2010).

15. C Fang, J Lee, MA Schilling, Balancing exploration and exploitation through structural design: The isolation of subgroups and organizational learning. *Organ. Sci.* **21**, 625–642 (2010).

16. SD Dionne, H Sayama, FJ Yammarino, Diversity and social network structure in collective decision making: Evolutionary perspectives with agent-based simulations. *Complexity* **2019**, e7591072 (2019).

17. J Meluso, J Austin-Breneman, JP Bagrow, L Hébert-Dufresne, A review and framework for modeling complex engineered system development processes. *IEEE Transactions on Syst. Man, Cybern. Syst.* **52**, 7679–7691 (2022).

18. P Holme, BJ Kim, Growing scale-free networks with tunable clustering. *Phys. review E* **65**, 26107–26107 (2002).

19. R Laskar, C Walls, Chessboard graphs, related designs, and domination parameters. *J. Stat. Plan. Inference* **76**, 285–294 (1999).

20. F Pedregosa, et al., Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

21. S Oldham, et al., Consistency and differences between centrality measures across distinct classes of networks. *PLOS ONE* **14**, e0220061 (2019).

22. S Seabold, J Perktold, Statsmodels: Econometric and statistical modeling with python in *Proceedings of the 9th Python in Science Conference*. (Austin, TX), Vol. 57, pp. 10–25080 (2010).