BMC Bioinformatics

RESEARCH

Open Access

Molecular complex detection in protein interaction networks through reinforcement learning

Meghana V. Palukuri[1,2*], Ridhi S. Patil[3*] and Edward M. Marcotte[1,2*]

*Correspondence: meghana.palukuri@utexas.edu; ridhipatil@utexas.edu; marcotte@utexas.edu

[1] Department of Molecular Biosciences, Center for Systems and Synthetic Biology, University of Texas, Austin, TX 78712, USA
[2] Oden Institute for Computational Engineering and Sciences, University of Texas, Austin, TX 78712, USA
[3] Department of Biomedical Engineering, University of Texas, Austin, TX 78712, USA

Abstract

Background: Proteins often assemble into higher-order complexes to perform their biological functions. Such protein–protein interactions (PPI) are often experimentally measured for pairs of proteins and summarized in a weighted PPI network, to which community detection algorithms can be applied to define the various higher-order protein complexes. Current methods include unsupervised and supervised approaches, often assuming that protein complexes manifest only as dense subgraphs. Utilizing supervised approaches, the focus is not on *how* to find them in a network, but only on learning *which* subgraphs correspond to complexes, currently solved using heuristics. However, learning to walk trajectories on a network to identify protein complexes leads naturally to a reinforcement learning (RL) approach, a strategy not extensively explored for community detection. Here, we develop and evaluate a reinforcement learning pipeline for community detection on weighted protein–protein interaction networks to detect new protein complexes. The algorithm is trained to calculate the value of different subgraphs encountered while walking on the network to reconstruct known complexes. A distributed prediction algorithm then scales the RL pipeline to search for novel protein complexes on large PPI networks.

Results: The reinforcement learning pipeline is applied to a human PPI network consisting of 8k proteins and 60k PPI, which results in 1,157 protein complexes. The method demonstrated competitive accuracy with improved speed compared to previous algorithms. We highlight protein complexes such as C4orf19, C18orf21, and KIAA1522 which are currently minimally characterized. Additionally, the results suggest TMC04 be a putative additional subunit of the KICSTOR complex and confirm the involvement of C15orf41 in a higher-order complex with HIRA, CDAN1, ASF1A, and by 3D structural modeling.

Conclusions: Reinforcement learning offers several distinct advantages for community detection, including scalability and knowledge of the walk trajectories defining those communities. Applied to currently available human protein interaction networks, this method had comparable accuracy with other algorithms and notable savings in computational time, and in turn, led to clear predictions of protein function and interactions for several uncharacterized human proteins.

Keywords: Community detection, Reinforcement learning, Protein complex, Protein interactions

Background

Protein–protein interactions (PPIs) are essential to nearly all cellular functions and biological processes. From antibodies binding antigens to block infections, to protein filaments comprising cellular cytoskeletons, protein interactions are an important organizational principle across biological scales and organisms. Such multi-protein complexes may additionally bind other molecules, such as DNA, RNA, or metabolites, and play critical roles in cellular processes ranging from DNA replication to transcription, multicellular interactions, and tissue organization.

As a consequence, a growing variety of experimental techniques have been developed to determine PPIs at a large scale, notably including affinity purification/mass spectroscopy (AP/MS), co-fractionation/mass spectrometry (CF/MS), cross-linking/mass spectrometry (XL/MS), proximity labeling, and yeast two-hybrid assays (Y2H), which are collectively reviewed in [1–5]. The resulting PPIs define (often weighted) networks of interactions, in which each node represents a protein, an edge represents the interaction confidence, and certain proximal groups of nodes and edges correspond to multiprotein complexes. Importantly, the experimental methods are not completely accurate and suffer both false positive and negative observations. Hence, integrating PPIs across multiple experiments, *e.g.* the networks hu.MAP 1.0 [6] and hu.MAP 2.0 [7] that integrate over 9000 and 15000 mass spectrometry experiments respectively from AP/MS [8–11] and CF/MS data [12–15], can help to mitigate the effects of experimental errors. Combining such approaches with algorithms to cluster proteins and identify complexes from the PPI network should result in a more accurate determination of protein complexes. Community detection algorithms can be applied to a PPI network to identify its communities, *i.e.*, protein complexes [16].

Community detection methods can be unsupervised, *i.e.*, not use any information from known communities in a network and instead rely only on the network topology to cluster it into its communities. Currently, existing unsupervised community detection algorithms tend to rely on many assumptions regarding the topological structures of communities. MCODE (Molecular COmplex DEtection) is an unsupervised method of detecting protein complexes running on the assumption that dense regions of a network represent complexes [17]. Another unsupervised algorithm, CMC (cluster-based on maximal cliques), assumes that communities are mainly in the shape of cliques (again, highly dense subgraphs) [18]. This pattern of similar assumptions carries on to other unsupervised methods such as COACH (core-attachment-based method) [19], ClusterONE (clustering with overlapping neighborhood expansion) [20], and GCE (greedy clique expansion) [21], among others.

The FCAN-MOPSO algorithm is an enhanced clustering method that optimizes communities based on fuzzy clustering logic and multi-objective particle swarm optimization (MOPSO) techniques [22]. It clusters complexes by assigning membership degrees to nodes, allowing for soft assignments, and captures the overlapping nature of clusters. However, there are disadvantages to this algorithm, including computational complexity, sensitivity to parameter settings, and limited applicability to datasets where clusters are expected to be distinct and non-overlapping. Another current community detection algorithm applies the alternating direction method of multipliers (ADMM) to find complexes in a parallel manner [23]. The algorithm decomposes the protein complex detection task into subtasks and applies the ADMM framework to find complexes based on topological and biological assumptions. Though this is computationally efficient, the use of assumptions limits the finding of new protein complexes. HiSCF leverages higher-order structures in biological networks by considering higher-order relationships among nodes and identifies clusters that capture complex patterns and functional modules [24]. The algorithm utilizes Markov clustering (MCL) and iteratively updates and refines clusters based on the expanding inflate operation until groups are well-defined. Other existing algorithms are generally limited to handling only specific connectivity patterns, however, HiSCF was designed to target a wider range of possible patterns [24]. However, HiSCF may face challenges when applied to large interaction networks due to memory requirements and computational complexity. Recent unsupervised algorithms include PC2P, which uses a greedy approximation algorithm based on biclique subgraph properties [25], and MP-AHSA which uses a fitness function for biological similarities within complexes and optimizes a core-attachment-based algorithm for complex identification [26]. Another method, DPCMNE recursively compresses PPI networks, learns multi-level protein embeddings, and applies a core-attachment approach based on the embeddings' similarities [27].

On the other hand, supervised community detection methods do consider different topological features of communities apart from density, and learn a community fitness function (*i.e.*, the probability of being a community) from known complexes using different learning algorithms. One such approach uses a support vector machine (SCI-SVM) and a Bayesian network (SCI-BN) [28]. For both models, subgraphs are represented using 33 features, and a local subgraph growth process is employed starting from a seed node, with the subgraph growth regulated by limited growth rounds, score improvement over iterations, and extent of overlap with other candidate communities. ClusterSS, a cluster with supervised and structural information, is a supervised algorithm using a neural network, 17 subgraph features, and a structural scoring function [29]. All three methods, SCI-SVM, SCI-BN, and ClusterSS use a greedy heuristic algorithm for selecting the neighbor to add to the subgraph in the growth process, with ClusterSS considering only the top neighbors by degree for speed improvements. However, since the methods use serial candidate community sampling, this negatively impacts their scalability to large networks like hu.MAP 1.0 [6] with ~ 8k proteins and ~ 60k interactions, and hu.MAP 2.0 [7] with ~ 10k proteins and over 40k interactions. To combat this, Super.Complex (supervised complex detection algorithm) was developed for high scalability and accuracy [30]. By using AutoML (Automated Machine Learning), it explores different supervised algorithms to utilize the optimal one learned from known communities. Then, it samples candidate subgraphs using the learned fitness function by growing them with an epsilon-greedy heuristic, along with one of four additional heuristics (pseudo metropolis, clique—pseudo metropolis, iterative simulated annealing, and greedy). However, the AutoML pipeline can take a long computation time and the method can still be improved in terms of accuracy.

While current supervised learning methods learn community fitness functions, they do not learn trajectories on the network that can lead to a protein complex, potentially missing complexes that cannot be traversed using the heuristics employed, for instance in a greedy setting where the community fitness function is maximized at each step. We can apply reinforcement learning to learn paths traversed on a graph to recall known complexes with high accuracy. MARL (Multi-Agent Reinforcement Learning) uses Q-learning to form clusters in networks based on a multi-agent environment [31]. In this algorithm, each node is viewed as an agent and each agent chooses actions to grow into a cluster. A team reward is given to train the action-value function, based on the modularity of the partition, which again assumes that all communities are dense subgraphs. The team reward can also lead to unstable learning of each agent's behaviors, apart from taking a long time to compute. Nevertheless, MARL has suggested that there is a lot of potential for the use of reinforcement learning in community detection algorithms.

Rather than using a multi-agent approach with a team reward based on modularity, an unsupervised measure, we use a single-agent approach (where a subgraph is viewed as an agent) with a supervised reward from training communities based on whether the agent selects the right neighbor to grow the subgraph. The rewards are used to train a value iteration algorithm to learn an optimal value function that is used to predict new communities. During the prediction phase, we implement a parallel method with a single agent on each core (process) to increase the speed of the algorithm by predicting communities in a distributed fashion. Applying the reinforcement-learning (RL) algorithm on a human PPI network after learning from experimentally characterized complexes, we can identify candidate protein complexes, and these candidates can then be experimentally characterized. We can create reliable models of protein complexes that allow us to extract more information about their stability, affinity, and specificity.

In the current work, we formulate community detection as a reinforcement learning task and implement a value iteration algorithm, learning from known communities. The RL algorithm accurately and efficiently predicts candidate complexes by learning and using a value function from known communities, which maps the density of a subgraph to the probability (score) that traversing the subgraph will yield a protein complex. The algorithm trains on known complexes that have nodes and edges from the network, to accurately optimize scores for the various densities that could occur on the network. Then, the RL pipeline uses these scores to traverse the network by starting with different seed nodes to create candidate complexes in parallel.

The RL pipeline for community detection presents various advantages when compared to its unsupervised and supervised counterparts. Unlike unsupervised methods that rely on assumptions regarding communities such as 'communities are dense subgraphs of a network', the RL algorithm takes a more flexible approach by learning trajectories on the network to find known communities having different topologies. To sample candidate communities from the network (an NP-hard problem), while supervised methods have previously used pre-defined heuristics, the RL algorithm learns the correct heuristic to use. Compared to supervised machine learning methods such as neural networks and AutoML methods which require sufficient hyper-parameter tuning and have high training time, the RL pipeline uses the simple value iteration algorithm with few parameters, achieving comparable results in a fraction of the time. Combined with its parallel implementation utilizing multiple cores, the RL pipeline is a fast community detection algorithm enabling efficiency and scalability to large networks. In this paper, we demonstrate the algorithm's utility by applying it to a high-quality human PPI network (consisting of 8k proteins and 60k pairwise protein interactions) and learn 1,157 candidate protein complexes. These candidates include complexes containing uncharacterized proteins, for which we can suggest possible functions based on 'honor by association' with their co-complex members.

Methods

Reinforcement learning

Reinforcement learning (RL) uses machine learning to enable an agent to make a sequence of decisions based on rewarding desired behaviors and punishing undesired ones. These rewards reinforce the right decisions so that the agent repeats them. Over time, the algorithm finds the best possible decision or action to take in each situation. Thus, RL is an intuitive method for community detection, as decision-making for the long-term goal of finding a protein complex is needed during the process of traversing a network to grow a complex from a node.

There are three main variables in RL: a state, value function, and reward. A state is a "position" the agent is in, in an environment. A value function is a score given to a state that estimates how beneficial it is for the agent to be in that state to achieve the goal. Lastly, a reward is an incentive that tells the agent if the decision made was correct or not. For example, consider a computer playing against a human in a game of tic tac toe. The AI player is an *agent* created to perform certain *actions* on the tic tac toe grid (the *environment*) based on what *state* the environment is in, in real-time. After each action taken, the agent receives a *reward* based on the four possible outcomes of what the *state* could result in: it wins, the opposing player wins, a draw, or continues the game. If the agent wins, it will receive a positive reward (*i.e.*, 1), and if the opposing player wins or if there is a draw—it will receive a negative reward (*i.e.*, -1), and if the game continues there will be no reward (*i.e.*, 0 or None). The AI player continues to make moves based on what *state* it currently is in to maximize its cumulative reward or *return*. This cycle, also known as an *episode*, terminates when the game ends. While trying to maximize its rewards, the agent will learn the optimal policy, *i.e.*, the best action to be taken in a particular state. The optimal policy is learned by starting with an initial policy and adapting it based on its experience encountering various states.

Value iteration

The optimal policy at a state is performing an action that takes the agent to the next best state that will maximize the probability of achieving the goal. In other words, the optimal policy, out of the states available, moves the agent to the state having the highest value function. The true or optimal value function, *i.e.*, a map of the states to their values can be learned using the value iteration algorithm, a classic reinforcement learning method for problems where a model of the environment dynamics is known, usually with a small number of discrete states. The algorithm is a dynamic programming method that solves the Bellman Optimality Equation (Eq. 1) iteratively, converging to the optimal value function V^*.

$$V^*(s_t) = \max_{a_t} \left(\Sigma_{s_{t+1}} p(s_{t+1}|s_t, a_t) \big(R(a_t, s_t) + \gamma V^*(s_{t+1}) \big) \right) \tag{1}$$

Here, R is the defined reward associated with performing an action a_t when the state is s_t, γ is a discount factor and p is the transition probability from s_t to s_{t+1} when it performs action a_t.

The value iteration update rule, starting with a value of 0 for all states is given by Eq. (2):

$$V_k(s_t) = \max_{a_t} \left(\Sigma_{s_{t+1}} p(s_{t+1}|s_t, a_t) \left(R(a_t, s_t) + \gamma V_{k-1}(s_{t+1}) \right) \right) \tag{2}$$

Here, $V_k\ (s_t)$ is the value function of the current state (at time t in the episode) in the current iteration k (of the value iteration algorithm) and $V_{k\text{-}1}\ (s_{t+1})$ is the value function (from the previous iteration k-1) of the next possible state.

Formulating community detection as a reinforcement learning problem

To build an RL pipeline for the problem of community detection, the algorithm is first trained on known training communities or complexes. Once the training is deemed to be successful, the learned value function from the training is then used to find complexes on a network.

To learn the value function, each episode consists of starting with a seed node from a training complex and iteratively adding neighbors to grow the subgraph into the complex. This process is then repeated with a new seed node from the training complex. Once all the nodes of the complex have been used as seeds, training moves to the next complex. In this scenario, the agent and environment are defined as the current subgraph in the growth process and the full graph including all its neighbors, respectively. We represent the state of the agent, *i.e.*, the current subgraph by its topological feature, density. The state (density d) of the current subgraph is the ratio of the actual number of edges in a subgraph to the total possible number of edges and can be calculated with Eq. (3).

$$d = \frac{2m}{n(n-1)} \tag{3}$$

Here m is the sum of the edge weights of the edges in the subgraph and n is the number of nodes in the subgraph.

For a wide representation of the feature space, the states are discretized into 20 intervals ranging from 0 to 1. The actions performed by the agent comprise adding a neighbor to the current subgraph or terminating the growth process. Choosing a neighboring node in the known complex will provide the agent with a positive reward of $+0.2$, and a negative reward of -0.2 is given if the node chosen is not present in the known complex. The rewards aid the agent in avoiding previous mistakes for it to find an optimal path to create a complex. These rewards allow the state to develop a value function representing the probability of the state resulting in a final community. If none of the remaining neighbors are in the complex, the agent is encouraged to learn to terminate the growth process by receiving a reward of 0, as opposed to choosing a neighbor giving a reward of -0.2.

Once the training completes and an optimal value function is learned, the agent learns candidate complexes on the entire network by starting with seed nodes in parallel and adding neighbors giving the highest value function at each iteration, until the action of terminating a growth process gives a higher value than adding any of the neighbors.

Proof of applicability of RL to community detection

The environment is deterministic and the next state the subgraph moves into, (s_{t+1}), is only dependent on the previous state (s_t), the current subgraph. It does not depend on any other states previously encountered by the agent, satisfying the Markov property (Eq. 4) with a memoryless process.

$$p(s_{t+1}|s_t) = p(s_{t+1}|s_t, s_{t-1}, ...s_0) \tag{4}$$

Here, on the left-hand side, p is the conditional probability of achieving a state given only the previous state, while on the right-hand side, the probability is conditional on all the previous states encountered. Therefore, with this formulation, community detection can be treated as a Markov Decision Process (MDP) and solved using reinforcement learning methods.

Value iteration for community detection

The value iteration update rule with our formulation of the community detection problem is given by Eq. (5):

$$V_k(s_t) = max_{a_t}\left(R(a_t, s_t) + \gamma V_{k-1}(s_{t+1})\right) \tag{5}$$

V_k (s_t) is the value function of the current state (at time t in the episode) in the current iteration k (of the value iteration algorithm), R is the defined reward, γ is the discount factor (0.5), and V_{k-1} (s_{t+1}) is the value function (from the previous iteration k-1) of the next possible state. We obtain this simple update rule, derived from the Bellman optimality equation (Eq. 6) using a transition probability $p(s_{t+1}|s_t, a_t)$ of 1 in Eq. (1) due to the deterministic nature of this problem, *i.e.*, the state transitions from s_t to only s_{t+1} when action a_t is taken.

$$V^*(s_t) = max_{a_t}\left(R(a_t, s_t) + \gamma V^*(s_{t+1})\right) \tag{6}$$

Here, V^* is the optimal value function, which the algorithm's value function converges to after a few iterations using the value iteration update rule (Eq. 5), starting with a value of 0 for all states.

Reinforcement learning community detection algorithm

Overview

There are three main steps for community detection on a network using reinforcement learning:

1. *Training* the algorithm to walk across training complexes by learning a value function corresponding to each state (subgraph) encountered in the process.
2. *Finding* candidate complexes by using the learned value function to walk on the protein–protein interaction network.

3. *Benchmarking* learned complexes against known complexes.

Training the algorithm

For training the RL pipeline, we use the known training complexes and represent each complex as a target subgraph of the protein-interaction network. The agent, *i.e.*, the current subgraph expands by adding, at each step, the neighbor that yields the highest value for the current subgraph (the value is calculated using the reward given for adding this node and the value of the next state, as shown in Eq. 5). The algorithm updates the value of the density of the current subgraph to this new value (Eq. 5). Each time a state (density) is encountered in the process of training on multiple training complexes, the value of that state is updated using the update rule (Eq. 5), moving towards convergence of the value function and eventually learning a value function mapping densities to their probability of leading to a protein complex. Figure 1 shows an example of learning the value function while traversing a single known complex, and Algorithm 1 (Fig. 2) summarizes the training procedure. The initial subgraph or seed is an edge between an initial random node of the complex and the node's neighbor with whom the node shares an edge with the highest edge weight. To calculate the potential value of the current subgraph, *i.e.*, the term within the max () in Eq. 5 for each neighbor of the current subgraph, each neighbor is temporarily added to the subgraph. The density of that temporary subgraph is calculated, followed by querying the value function for that state along with the reward based on whether the neighbor is present in the final protein complex. After calculating the potential value function, the neighbor is removed from the subgraph for this process to be repeated for the rest of the neighbors. Once all the neighbors have been evaluated, the value function of the current state is updated and the neighbor yielding the state that provided the maximum value function is added to the subgraph. This new subgraph will be the new "seed" as this process repeats itself. The subgraph, or complex, will be "complete" and the algorithm stops adding nodes if all the neighboring nodes return lower value functions than the action of terminating the growth process, represented by adding an imaginary node with reward 0, leading to the same state as before, indicating that no new neighbors should be in the complex. Note how a reward of 0 encourages the algorithm to terminate the growth process when all other options are adding wrong nodes, *i.e.*, choosing actions that have a reward of -0.2. Conversely, if a correct node is available, its reward of 0.2 encourages choosing that node over terminating growth. Once a subgraph is deemed as complete, the process is repeated starting with a different node of the same complex, so that other trajectories to build the same complex are explored and the value function is updated accordingly. After starting with each of the nodes of a complex as seeds, training moves to the next complex, starting with a random new seed from this complex, and the process repeats.

Finding candidate complexes

Once we observe that the value functions of different states have converged in the training process, we can use the learned value function to walk paths on the network to find complexes. For each node of the network, we choose its corresponding highest edge-weight as a seed edge to grow a candidate complex. For each seed edge, the

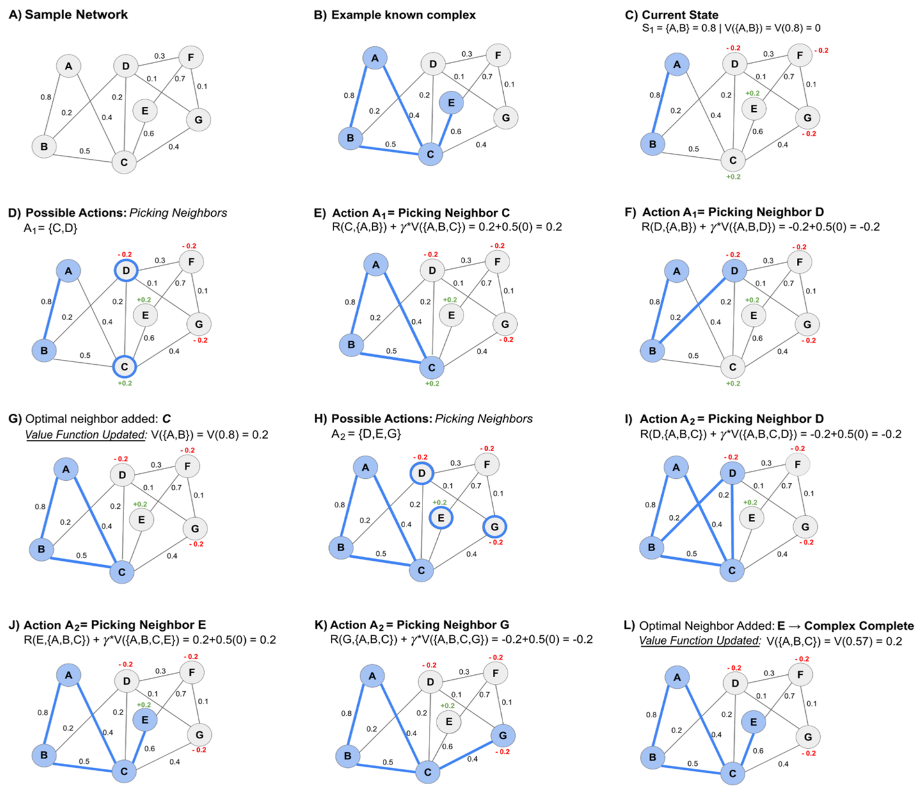
A) Sample Network
B) Example known complex
C) Current State
S1 = {A,B} = 0.8 | V({A,B}) = V(0.8) = 0
D) Possible Actions: Picking Neighbors
A1 = {C,D}
E) Action A1 = Picking Neighbor C
R(C,{A,B}) + γ*V({A,B,C}) = 0.2+0.5(0) = 0.2
F) Action A1 = Picking Neighbor D
R(D,{A,B}) + γ*V({A,B,D}) = -0.2+0.5(0) = -0.2
G) Optimal neighbor added: C
Value Function Updated: V({A,B}) = V(0.8) = 0.2
H) Possible Actions: Picking Neighbors
A2 = {D,E,G}
I) Action A2 = Picking Neighbor D
R(D,{A,B,C}) + γ*V({A,B,C,D}) = -0.2+0.5(0) = -0.2
J) Action A2 = Picking Neighbor E
R(E,{A,B,C}) + γ*V({A,B,C,E}) = 0.2+0.5(0) = 0.2
K) Action A2 = Picking Neighbor G
R(G,{A,B,C}) + γ*V({A,B,C,G}) = -0.2+0.5(0) = -0.2
L) Optimal Neighbor Added: E → Complex Complete
Value Function Updated: V({A,B,C}) = V(0.57) = 0.2
A
B
C
D
E
F
G
0.8
0.2
0.4
0.5
0.2
0.3
0.1
0.7
0.1
0.6
0.4
+0.2
- 0.2

Fig. 1 Example trajectory of training the RL pipeline on a network by learning a value function. **A** This network comprises 7 nodes and 11 weighted edges **B** A known complex consists of the nodes A, B, C, and E. **C** First, a seed edge (A, B) is identified, where the state (density) is $S_1 = 0.8$ and the value function is $V(0.8) = 0$ (all densities are initialized to 0). Once a node is added, a reward of $+0.2$ is given if the node is in the training complex and -0.2 if absent. **D** We evaluate all possible neighbors i.e., C and D, to add to the current subgraph {A,B}. Using the value iteration update rule (with $\gamma = 0.5$), we compute a corresponding value for the current state by adding each neighbor. **E** Adding node C updates $V(0.8) = 0.2$. **F** Adding node D updates $V(0.8) = -0.2$. **G** The neighbor providing the highest value function (C) is added to the candidate complex and the original state's value function $S_1\{A,B\} = 0.8$ is now $+0.2$. **H** Again, we evaluate all possible neighbors of the updated complex $S_2\{A,B,C\} = 0.57$, i.e., D, E, and G. **(I)** Node D updates $V(0.57) = -0.2$. **J** Node E updates $V(0.57) = +0.2$. **K** Node G updates $V(0.57) = -0.2$. **L** Node E is added to the complex and V(0.57) is updated to $+0.2$. This process is repeated until growth termination by adding an imaginary node with reward 0. As the remaining neighbors D, F, and G have a reward of -0.2, the imaginary node is chosen as it results in the highest value function (0.1). The candidate complex is then finalized. A new seed edge is chosen from the network and this process repeats

neighbors are evaluated and the neighbor yielding the subgraph with the maximum value function is added. The growth process stops when adding any neighbor lowers the value function of the current subgraph. For instance, consider a subgraph with a value function of 0.2. If on evaluating each of the subgraph's neighbors, each of them returns a value function lesser than 0.2, the algorithm terminates, and the subgraph will be considered a candidate complex. This process is repeated for the next seed edge in the network. These steps are detailed in Algorithm 2 (Fig. 3) and an example of finding a complex on the network is shown in Fig. 4.

```
Algorithm 1 Training the RL algorithm
Require: Network $G$, Training Complexes $C$, Discount factor $\gamma = 0.5$
Ensure: Value Function $V$
 1: Initialize Value Function $V \leftarrow 0$
 2: for each Known Complex $C_k$ in $C$ do
 3:   Determine reward function $R(n, C_k)$ :
 4:   if Node $n$ in $C_k$ then
 5:     $R(n, C_k) = 0.2$
 6:   else
 7:     $R(n, C_k) = -0.2$
 8:   end if
 9:   for each Seed Node $p$ in $C_k$ do
10:     Seed Edge $e \leftarrow \{p$, highest edge weight neighbor$\}$
11:     Initialize subgraph $S \leftarrow e$
12:     while growing $S$ do
13:       Neighbors of $S \leftarrow N_S$
14:       for each neighbor $n$ in $N_S$ do
15:         Add $n$ temporarily to subgraph
16:         Calculate subgraph density $\rho_{S+n}$ (state)
17:         Retrieve $V_{S+n} \leftarrow V(\rho_{S+n})$
18:         Assign Possible value for adding n:
19:         $V'_n \leftarrow R(n, C_k) + \gamma * V_{S+n}$
20:         Remove $n$ from subgraph
21:       end for
22:       Current value function of $S \leftarrow V_S$
23:       if max$(V'_n)$ >$V_S$ then
24:         $V(\rho_S) \leftarrow V_S = max(V'_n)$
25:         Add max value neighbor to subgraph $S$
26:       else
27:         Terminate growth process for $S$
28:       end if
29:     end while
30:   end for
31: end for
32: return Value Function $V$
```

Fig. 2 Algorithm 1 outlines the training process for the RL algorithm

Post-processing and evaluation

Once we have found all the candidate subgraphs corresponding to the specified seed nodes (in our experiments we use all the nodes of the graph as seed nodes), we perform a post-processing step to merge highly overlapping complexes. Adapting the pairwise merging algorithm in Super. Complex [30], if the overlap of two complexes is more than a specified threshold, we retain the complex with the highest value function of the two complexes and the merged variant and remove the others. To obtain the optimal overlap threshold, similar to [30], we test various thresholds for the *Qi overlap* measure (Eq. 7) and choose the threshold which gives the highest F-similarity-based Maximal Matching F-score (FMMF).

$$\text{Qi overlap measure :} \ \frac{|C_p \cap C_k|}{|C_p|} > t \text{ and } \frac{|C_p \cap C_k|}{|C_k|} > t \tag{7}$$

Here, t is the user-specified overlap threshold, C_p is a predicted complex and C_k is a known complex.

```
Algorithm 2 Finding Candidate Complexes
Require: Network $G$, Seed Edges $E_s$, Learned Value Functions $V$
Ensure: Candidate Complexes $C$
 1: for each Seed Edge $e$ in $E_s$ do
 2:   Initialize subgraph $S$ as $E_s$
 3:   Find neighbors $N_e$
 4:   Current value function of $S$ is $V_s$
 5:   while Growing subgraph $S$ do
 6:     Choose random node $n$ in $S$
 7:     for each neighbor $n$ do
 8:       Add $n$ temporarily to subgraph
 9:       Calculate subgraph density (state)
10:       Retrieve learned value function of neighbor $V_{s+n}$
11:       Remove $n$ from subgraph
12:     end for
13:     if max($V_{s+n}$) >$V_s$ then
14:       Add neighbor $n$ to subgraph
15:     else
16:       Terminate growth process and return current subgraph as a candidate
          complex
17:     end if
18:   end while
19: end for
20: return Candidate Complexes $C$
```

Fig. 3 Algorithm 2 outlines the steps in the RL algorithm to identify candidate complexes

To gauge the accuracy of the RL algorithm, we use different evaluation measures to compare learned complexes with known complexes. The learned complexes are compared with the known complexes after removing nodes missing in the set of known complexes. We employ a variety of evaluation measures such as the FMMF, Community-wise Maximum F-similarity-based F-score (CMMF), and Unbiased Sn-PPV Accuracy (UnSPA) defined in Super.Complex [30], in addition to the Qi et al. F-score [28], F-grand k-clique and F-weighted k-clique [6].

As discussed in [30], one of the more accurate methods for comparison is the FMMF which combines an adapted Maximal Matching Ratio (MMR) with its corresponding precision. The adapted MMR is computed as the fraction of known complexes matching predicted complexes, where the matches are computed as the sum of edge weights of a maximal matching in a bipartite graph between learned and known complexes. The edge weights used in the graph, matching learned and known complexes, are the F-similarity scores (F1) computed as follows.

$$p' = \frac{|C_p \cap C_k|}{|C_p|} \tag{8}$$

$$r' = \frac{|C_p \cap C_k|}{|C_k|} \tag{9}$$

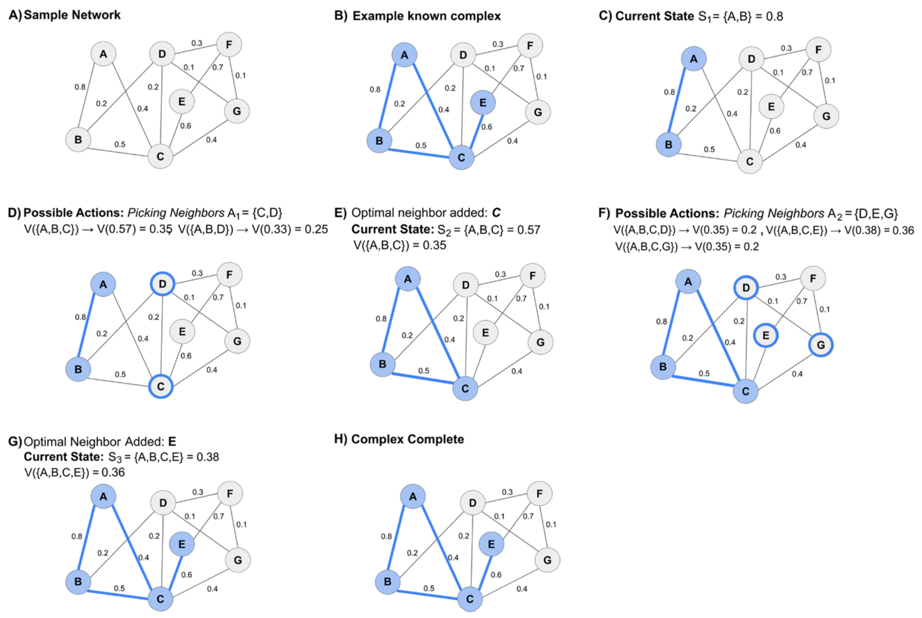
A) Sample Network
B) Example known complex
C) Current State S1 = {A,B} = 0.8
D) Possible Actions: Picking Neighbors A1 = {C,D}
V({A,B,C}) → V(0.57) = 0.35, V({A,B,D}) → V(0.33) = 0.25
E) Optimal neighbor added: C
Current State: S2 = {A,B,C} = 0.57
V({A,B,C}) = 0.35
F) Possible Actions: Picking Neighbors A2 = {D,E,G}
V({A,B,C,D}) → V(0.35) = 0.2 , V({A,B,C,E}) → V(0.38) = 0.36
V({A,B,C,G}) → V(0.35) = 0.2
G) Optimal Neighbor Added: E
Current State: S3 = {A,B,C,E} = 0.38
V({A,B,C,E}) = 0.36
H) Complex Complete

Fig. 4 Example trajectory for finding a complex with the RL pipeline using a learned value function. **A** This network comprises 7 nodes and 11 edges, each with corresponding with an edge weight. **B** In this network, a known complex consists of the nodes A, B, C, and E. The goal is to predict this known complex from the network using the learned value function. **C** A seed edge is identified to begin the walk (edge AB). At this seed edge, the complex is at state (density) $S_1 = 0.8$. **D** Then, we evaluate all possible neighbors of nodes A and B, i.e., C and D. Adding node C gives a temporary complex {A, B, C} with $S_2 = 0.57$, and a learned value V({A, B, C}) = 0.35, while adding node D gives a temporary complex {A, B, D} with $S_2 = 0.33$, V({A, B, D}) = 0.25. **E** The neighbor with the highest value function is node C and hence, node C is added resulting in $S_2 = 0.57$, V({A, B, C}) = 0.35. **F** The next neighbors are evaluated, i.e., D, E, and G. Adding node D leads to $S_3 = 0.35$, V({A, B, C, D}) = 0.2, node E results in $S_3 = 0.38$, V({A, B, C, E}) = 0.36, and node G leads to $S_3 = 0.35$, V({A, B, C, G}) = 0.2. **G** Since the neighbor yielding the highest value function is node E, this node is added to the complex resulting in $S_3 = 0.38$, V ({A, B, C, E}) = V(0.38) = 0.36. **H** Each neighbor (D, F, and G) results in a value function less than the current complex {A, B, C, E}. Thus, no neighbor is added, and the predicted community is complete

$$\frac{2}{F1} = \frac{1}{p'} + \frac{1}{r'} \tag{10}$$

Here, C_p is a derived complex and C_k is a known complex.

Results

To demonstrate the effectiveness of the RL algorithm, we show the algorithm's performance on synthetic and real datasets and discuss the human protein complexes learned by the algorithm. We first evaluate the algorithm on a synthetic toy dataset to help provide intuition for how the RL process works on a simple system that can be fully visualized. Then, we apply RL to the current full-scale human protein interaction network, focusing here specifically on the discovery of higher-order physical protein assemblies. For this purpose, we analyzed the hu.MAP networks (hu.MAP 1.0 [6] and hu.MAP 2.0 [7]), which were originally developed to maximize the quality and scale of protein interactions derived from mass spectrometry proteomics experiments and which have been independently shown to capture true protein interactions to a significant degree [32, 33]. Finally, we analyzed the set of derived complexes for uncharacterized proteins and complexes.

In this section, we start with the details of experiments carried out, and then examine the value functions learned in these experiments as these are the key part of the algorithm, signifying how to learn trajectories on the network corresponding to communities. Then, we present the evaluation metrics of the experiments including comparisons with state-of-the-art methods. In conclusion, we analyze the derived complexes from the experiment on the human protein interaction network, highlighting the complexes with uncharacterized proteins.

Experimental details—synthetic and protein interaction datasets

We first evaluate the RL algorithm on a synthetic toy network (Fig. 5) with 62 nodes and 78 edges, comprising 14 complexes. Of these 14 complexes, 7 are used as training complexes and 7 are used as testing complexes. The testing and training evaluation results can be found in Additional file 1: Table S1. The results showed consistently high scores across various evaluation metrics, indicating strong precision and accuracy in predicting both training and testing toy complexes.

Next, we apply the RL algorithm on a human protein interaction network (hu.MAP 1.0 [6]) comprising 7778 nodes and 56,712 edges to learn candidate complexes using 188 known complexes; these complexes are obtained by pre-processing the CORUM protein complex database [34]. The pre-processing (see methods section of Super.Complex [30]) primarily involves discarding complexes that are internally

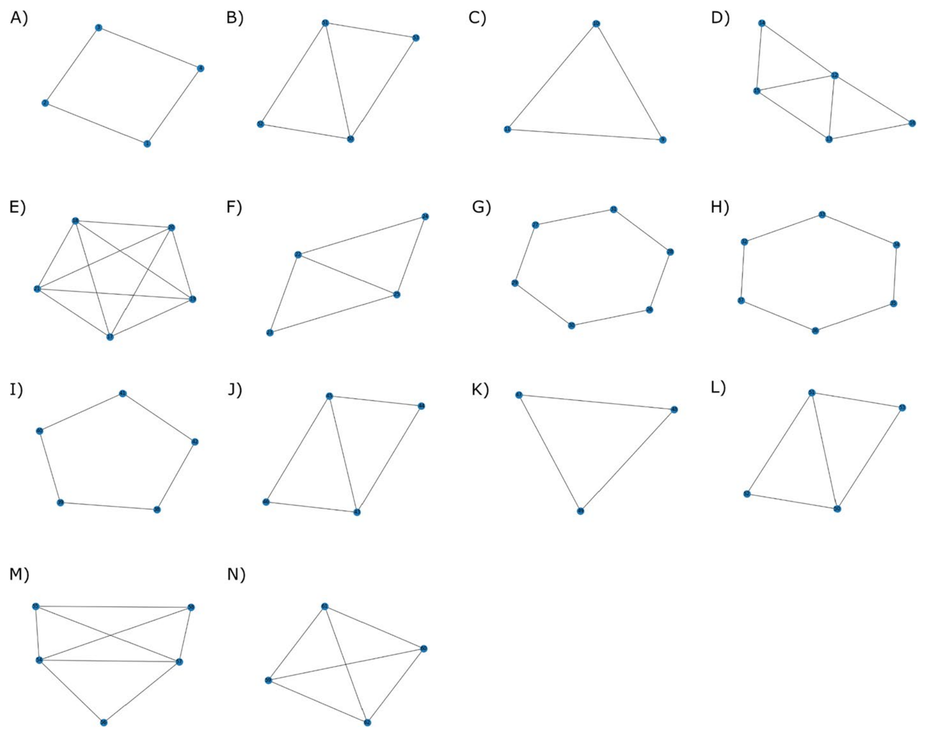

Fig. 5 A synthetic disconnected toy network of complexes. Complexes **A**–**G** are used for training and **H**–**N** are used for evaluating the RL algorithm

disconnected or have fewer than 3 nodes. We also merge complexes with a pairwise overlap of more than 0.6 Jaccard coefficient to remove redundancy, since the CORUM set of complexes includes multiple non-independent complexes, e.g., the 19S and 20S proteasomes are subcomplexes of the larger 26S proteasome. We merge these highly interdependent complexes to help reduce ambiguity while training the RL agent. Since we train on only one complex in each episode, reducing non-independent complexes will reduce negative rewards being given when the RL agent adds a neighbor that does not belong to the complex, but belongs to another overlapping complex. We also discuss an alternate reward scheme in the conclusions section for overlapping community detection.

For a perfect comparison, we use the same pre-processing steps as in Super.Complex for both the network and complexes, as well as the same training and testing complexes (all input data was obtained from the Super.Complex input data [30]). Since we only use the density feature in our algorithm, we also run the Super.Complex pipeline with only the density feature. The testing and training evaluation results can be found in Additional file 1: Table S2.

The RL algorithm is also tested on hu.MAP 2.0 [7], a human PPI network consisting of 10,433 nodes and 43,581 edges, obtained by considering only the edges with a weight of at least 0.02. Again, to compare with Super.Complex, we use the same training and testing complexes from hu.MAP 1.0, and the same preprocessing steps. We perform two experiments; in the first experiment, we transfer the value function trained on hu.MAP 1.0 and in the second, we train a new value function on hu.MAP 2.0. The testing and training evaluation results for hu.MAP 2.0 can be found in Additional file 1: Table S3.

The value function converges in the training phase

In this section, we examine the value functions learned in the experiments, as these signal the key heuristic learned to identify trajectories on the network leading to communities. During the training phase, we track the value for each encountered state (density) over time. Once the value of each of the states starts to converge, it can be assumed that the value has reached its optimum. Figure 6 demonstrates the successful convergence of the value for each density in hu.MAP 1.0.

We also investigate the relationship between a state's density and its value. Figure 7 shows that on the path to a final complex, subgraphs of higher densities are favored since they have higher values.

The learned value functions for the synthetic dataset and hu.MAP 1.0 enable us to accurately predict complexes on the respective networks, as shown in the next section. Further, employing transfer learning, we use the value function learned on hu.MAP 1.0 to accurately predict complexes on hu.MAP 2.0 (Additional file 1: Table S3). This demonstrates that the value functions learned by the RL algorithm can be transferred for community detection problems on similar networks. We also directly train a value function on hu.MAP 2.0 using the same training complexes and find that predicting complexes on hu.MAP 2.0 with this value function also gives accurate results (Additional file 1: Table S3).

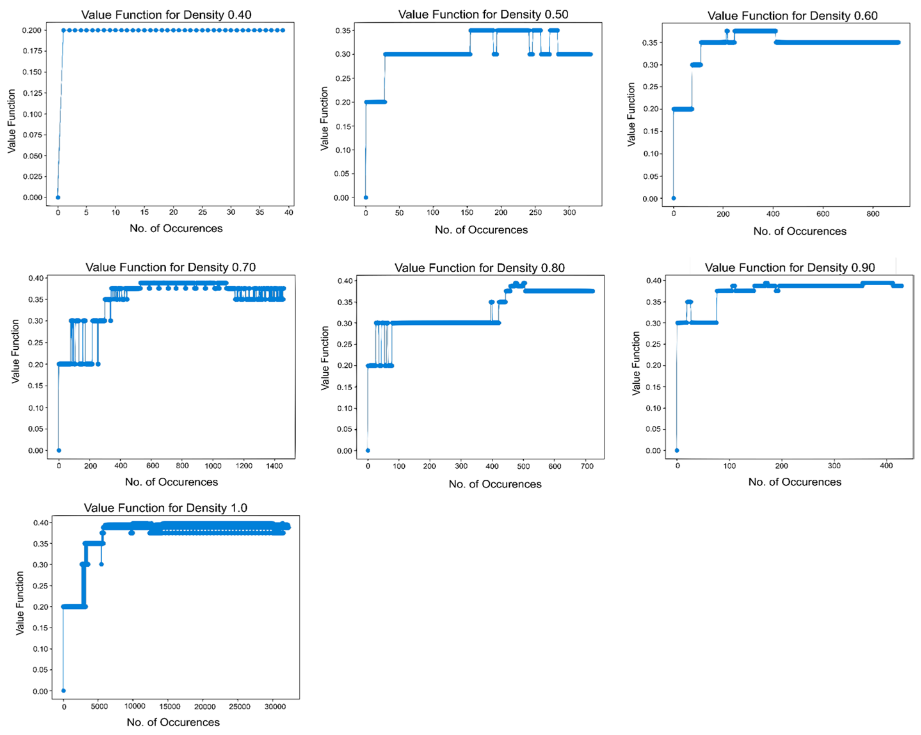

Fig. 6 Convergence of the value for each density. Each density (0.4 through 1) encountered in the training for hu.MAP 1.0 is plotted to see how its value updates over iterations. Although the values fluctuate initially, they converge eventually indicating successful training

The RL algorithm learns accurate communities on synthetic and real datasets

Having learned accurate values of different states (subgraphs) encountered in the process of reconstructing known communities, we now use the value functions learned to find communities on the whole network. Recall the synthetic dataset containing 14 communities used to evaluate the RL algorithm. The performance of the algorithm across different evaluation measures is excellent as summarized in Table 1. Next, we apply the algorithm to the real dataset, hu.MAP 1.0, by training it on 132 complexes. We test different Qi overlap thresholds (Fig. 8A), in the RL algorithm, to merge highly overlapping complexes. The peak in Fig. 8A occurring at 0.325 Qi overlap measure corresponds to the best FMMF score. For this value of the Qi overlap measure, the RL algorithm learns 1,157 complexes. We also analyze the complex size distribution (Fig. 8B) and examine the best known complex matches for derived complexes and the best derived complex matches for known complexes using F1 score distributions (Fig. 8C).

For a perfect comparison, Super.Complex is evaluated on hu.MAP 1.0 using only the subgraph feature density. The best results from Super.Complex are obtained using a k-nearest neighbors' classifier (with k = 76) to train a community fitness function, and from a search process for candidate complexes using maximal cliques as starting seeds and a pseudo-metropolis heuristic (with a probability of 0.1) for complex growth (with an exploration probability ϵ of 0.01). The candidate complexes are then merged with an overlap threshold of 0.2 Jaccard coefficient to yield 798 final complexes. In contrast, the RL algorithm predicts a higher number (1157) of complexes possibly explaining

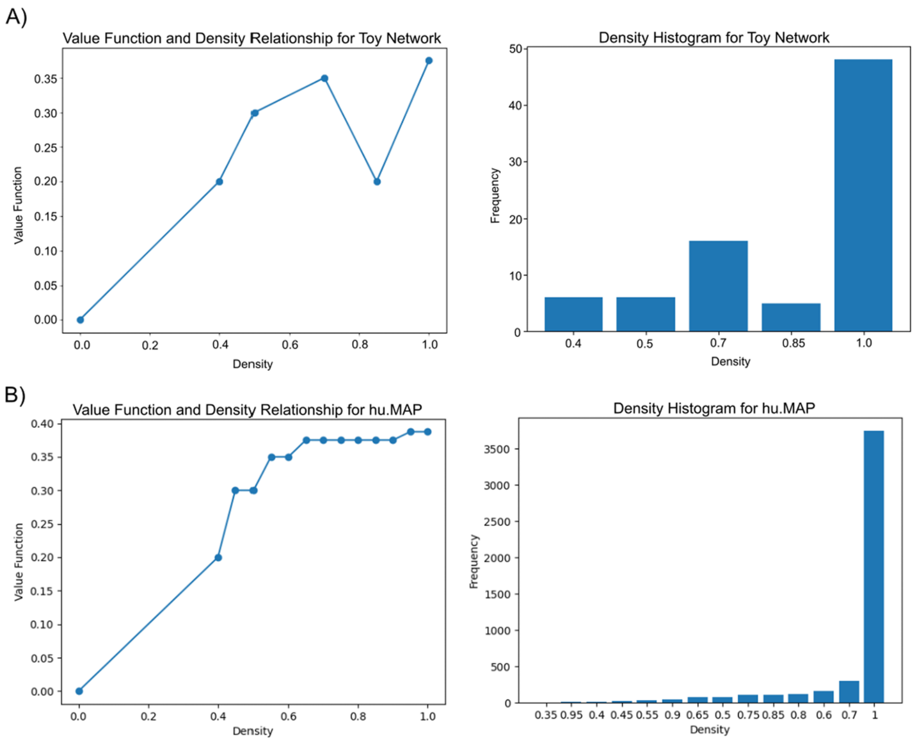

Fig. 7 Higher values favor higher densities. **A** The graph on the toy network shows a positive correlation between value function and density and therefore, complexes and trajectories with higher densities are favored. The density histogram for the training phase shows a higher frequency for higher density subgraphs. **B** For hu.MAP 1.0, the graph shows a stronger correlation between density and value function. The density histogram again shows that higher densities are more frequently observed

Table 1 RL algorithm has strong performance on a synthetic toy dataset

	FMM Precision	FMM Recall	FMM F-score	CMMF	UnSPA	Qi et al. F1 score	SPA	F-Grand K-Clique	F-weighted K-Clique
RL Algorithm	0.963	0.963	0.963	0.963	0.969	1.00	0.959	1.00	1.00
Super. Complex	0.999	0.999	0.999	1.00	0.999	1.00	0.998	1.00	1.00

The algorithm was trained on 7 toy complexes from a synthetic network of 62 nodes and 78 edges. It predicted 14 complexes which are evaluated against the 14 true complexes

FMM, F-similarity-based Maximal Matching; CMMF, Community-wise Maximum F-similarity-based F-score; UnSPA, Unbiased Sn-PPV Accuracy; SPA, Sn-PPV Accuracy

the slightly higher (FMM) recall measure (Table 2). The RL method achieves comparable performance to the supervised method, Super.Complex and tends to outperform 4 recent unsupervised community detection methods, ClusterONE + MCL [6], PC2P [25], MP-AHSA [26], and DPCMNE [27] (Table 2), demonstrating the potential of applying reinforcement learning to community detection.

Comparing Tables 1 and 2, we observe better accuracies for the algorithm on the toy dataset than those on hu.MAP 1.0. This could be attributed to the algorithm being better suited to finding non-overlapping communities, such as the toy communities, when compared to finding overlapping communities, such as the CORUM complexes

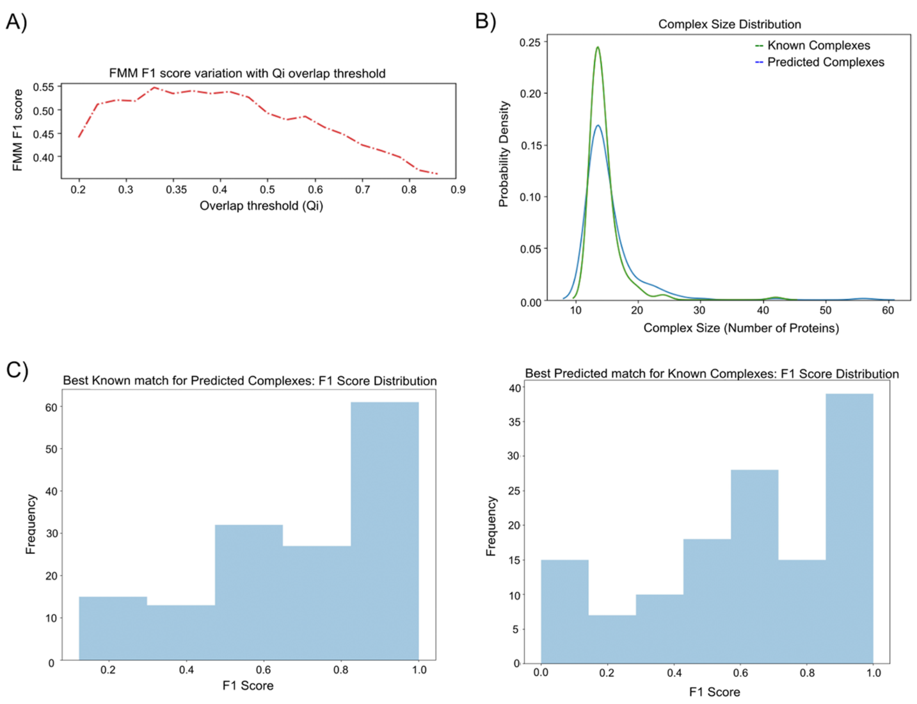

Fig. 8 Evaluating the predictions of the RL algorithm on hu.MAP 1.0. **A** The optimal Qi threshold is 0.325. We tested various overlap thresholds, *i.e.*, Qi values (Eq. 7) between 0.2 and 0.9 in 0.25 intervals. **B** Size distributions of known and predicted complexes. This graph shows that the distribution of the sizes (no. of proteins) of the predicted and known complexes is very similar. **C** F1 score distributions of the best-predicted match for known complexes and vice-versa. In both cases, higher F1 scores have higher counts indicating accurate predictions

Table 2 The RL algorithm yields competitive accuracy compared to other community detection algorithms on hu.MAP 1.0

	FMM Precision	FMM Recall	FMM F-score	CMMF	UnSPA	Qi et al. F1 score	F-Grand K-Clique	F-weighted K-Clique
RL Algorithm	0.612	0.482	0.547	0.654	0.772	0.559	0.789	0.988
Super. Complex	0.835	0.457	0.591	0.720	0.803	0.658	0.991	0.999
ClusterONE + MCL	0.471	0.686	0.579	0.797	0.911	0.794	0.77	0.967
PC2P	0.535	0.589	0.562	0.625	0.571	0.418	0.627	0.891
MP-AHSA	0.421	0.498	0.459	0.424	0.513	0.397	0.516	0.823
DPCMNE	0.457	0.048	0.086	0.408	0.454	0.132	0.609	0.608

The learned complexes on hu.MAP 1.0 are evaluated against all the known cleaned CORUM complexes

FMM, F-similarity-based Maximal Matching; CMMF, Community-wise Maximum F-similarity-based F-score; UnSPA, Unbiased Sn-PPV Accuracy; SPA, Sn-PPV Accuracy

on hu.MAP 1.0. We also evaluated the RL algorithm without merging overlapping communities in the predicted complexes, resulting in an FMM F1-score of 0.102 and a Qi et al. F1-score of 0.255. A comparison with the scores obtained when employing the merging operation (Table 2) reveals a significant improvement in metrics after eliminating overlapping complexes. Removing overlapping complexes reduces any redundancies in the dataset, enhancing the accuracy and efficiency of the algorithm.

While accuracies are comparable, we note that the RL algorithm achieves faster running time relative to Super.Complex (Table 3). Specifically, the RL algorithm trains for ~ 9 s on one core of a personal computer (M1 chip @ 3.2 GHz), making the training significantly faster than Super. Complex's training with the AutoML pipeline, which runs for ~ 540 s on 20 cores of a supercomputer (Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30 GHz). Growing the candidate communities took ~ 300 s when running in parallel across 8 cores (3.2 GHz) for the RL algorithm, compared to ~ 20 s when running in parallel across 72 cores (2.3 GHz) for Super.Complex. This indicates that growing new complexes is also fast in the RL algorithm due to the simple inference using the value function lookup. We employ the four heuristics available in Super.Complex, with default parameters, to find the best one and perform a parameter sweep of 7 thresholds for merging overlaps with each heuristic. In total, we evaluate 28 heuristic-parameter combinations in Super.Complex; the same number of overlap thresholds used in the RL method. Overall, with the best parameters, the RL algorithm took ~ 350 s on the personal computer with 8 cores (note that only the prediction step is parallelized here), compared to Super.Complex which took ~ 650 s on a supercomputer with 72 cores (note that both the learning and the prediction step are parallelized here).

The average time complexity of the prediction phase of the RL algorithm is $O\left(G^2K^2S/P\right)$, where G is the average number of nodes in a complex, K is the average degree of the network and S is the number of seeds chosen (in our experiments, S is the number of nodes in the network). For Super.Complex with all subgraph features, the time complexity of the prediction phase is $O\left(XG^4KS/P\right)$. We note that the prediction phase of the RL algorithm scales better than that of Super.Complex. This is because the complexity of the subgraph feature extraction step reduces from $O\left(G^3\right)$ in Super.Complex to $O(GK)$ in the RL algorithm; this reduction happens since the RL algorithm uses only the feature density with a constant model inference time (X). The time complexity of the RL training algorithm is $O\left(G^3K^2T\right)$, where T is the number of training complexes. In contrast, the training complexity of Super.Complex is $O\left(G^3Tgpm/c\right)$, where g is the number of generations, p is the population size, m is the number of machine learning models and feature preprocessor types tried, and c is the number of processes on the single compute node running the AutoML step.

Table 3 The RL algorithm achieves a faster running time when compared to Super.Complex

Method	Processor specifications	Training		Prediction		Post-processing		Total time (s)
		No. of cores	Time (s)	No. of cores	Time (s)	No. of cores	Time (s)	
RL Algorithm	M1 chip @ 3.2 GHz	1	9	8	320	1	11	340
Super. Complex	Intel(R) Xeon(R) E5-2699 v3 @ 2.30 GHz	20	540	72	17	1	112	669

The time reported here corresponds to runs using the best overlap threshold found for both methods and using the best heuristic found in the case of Super.Complex

We note that the RL algorithm can be especially useful in community detection problems with a small number of known complexes, as demonstrated in our experiments (7 and 132 training complexes in the synthetic and real datasets respectively). Even if the number of known complexes is small, for each complex, the value iteration procedure in the RL algorithm explores several trajectories to learn the complex, incidentally, increasing the size of the training dataset used to learn the complexes. On the other hand, existing supervised community detection methods train on a dataset with a size equal to the number of training complexes. Other benefits of the RL algorithm include the lack of need for extensive hyperparameter tuning and the ability to predict complexes that do not contain smaller complexes. For comparison, in Super.Complex, at each stage of growth in a candidate complex, the pipeline seeks to yield a final protein complex, attempting 4 different heuristics, each with 1–2 hyperparameters. Contrastingly, the RL pipeline learns and traverses the optimal trajectory to find a complex without optimizing for intermediate complexes, and without the need for heuristics, thus saving on searching for parameters in the candidate complex growth step. Thus, the RL algorithm finds the best sequence of steps to grow a complex, while also being efficient.

In summary, relative to more sophisticated supervised ML strategies, the simplicity of the value iteration algorithm and the comparable accuracy along with improved efficiency demonstrates the great potential of the RL algorithm for solving community detection problems.

The learned clusters suggest functions for uncharacterized proteins

Importantly, the RL algorithm returns many well-known human protein complexes accurately (as would be expected from the precision measurements on withheld test complexes), several of which are illustrated in Fig. 9.

Moreover, a noteworthy feature of the RL algorithm is that it adds a protein to a complex only if the protein increases or maintains the value function of the complex. For example, this can be seen when building the learned complex in Fig. 9D. When the seed edge is WASF2-ABI1, the algorithm chooses CYFIP1 as its first neighbor to add, resulting in a value function of 0.39. As the algorithm loops through various neighbors, the only neighbors added to the candidate complex are those that maintain the 0.39 value function of the complex. Interestingly, though in most cases the density of the complex is increasing, there are some additions to the complex that decrease the complex density such as when adding NCK2. This is unlike traditional greedy complex detection algorithms that favor higher-density protein complexes. Similarly, this pattern occurs when building the candidate complex with proteins, C15orf41, CDAN1, ASF1A, and HIRA (Fig. 11). Notably, the algorithm also identifies several additional interaction partners and even potential new subunits within these systems, such as, for example, clustering the guanine nucleotide exchange factor RCC1L with proteins of the mitochondrial ribosome large subunit, consistent with a known role for RCC1L in mitochondrial ribosome biogenesis [35]. Similarly, the RL algorithm recapitulates the nutrient-response-related KICSTOR complex (SZT2, KPTN, ITFG2, and C12orf66) [36] on both hu.MAP 1.0 (SZT2, KPTN, ITFG2, TMCO4 and C12orf66) and hu.MAP 2.0 (SZT2, KPTN, ITFG2, TMCO4, BMT2, and KICS2), suggesting the uncharacterized transmembrane protein TMCO4 to be a potential new interaction partner, and it reconstructs the WAVE1/

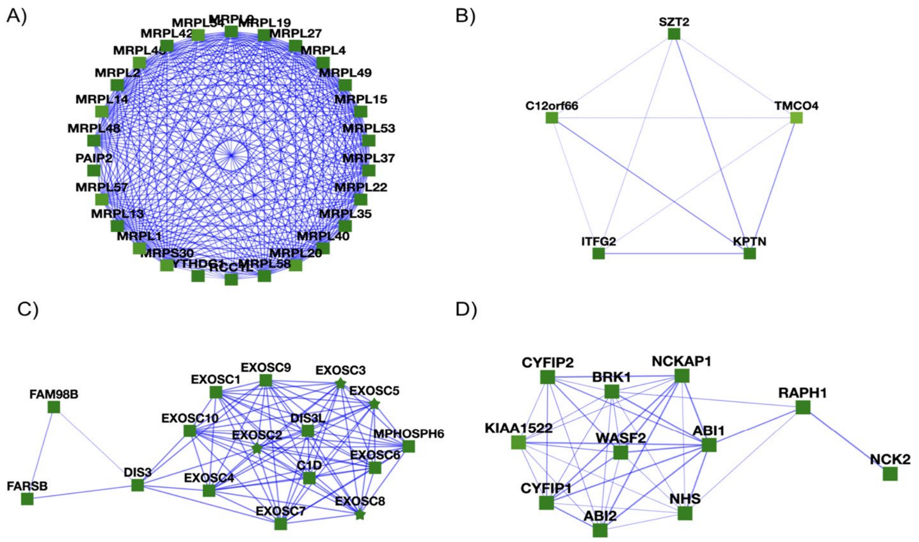

Fig. 9 Learned complexes from the RL pipeline. **A** The large (39S) subunit of the mitochondrial ribosome is present in the RL-determined complexes, but broken up into multiple complexes, one of which is shown here. Note the presence of the guanine nucleotide exchange factor RCC1L, which is known to be essential for mitochondrial ribosome biogenesis [30]. **B** RL recapitulates the KICSTOR complex (C12orf66, KPTN, ITFG2, and SZT2), a multiprotein complex known to regulate mTORC1 and cells' responses to available nutrient levels [31], but finds one additional putative subunit, the uncharacterized transmembrane protein TMCO4. **C** The exosome RNA processing complex is well-reconstructed by RL, with additional interactions observed to the tRNA synthetase FARSB and to FAM98B, a component of tRNA splicing ligase, consistent with possible associations among these systems [35]. **D** The WAVE1/WAVE2 protein complexes, known to regulate actin filament and lamellipodia formation [32, 33], are reconstructed by RL, along with evidence for interaction with the uncharacterized protein KIAA1522. Notably, KIAA1522 was recently suggested by Cho and colleagues to bind WAVE and participate in a community of associated actin-organizing proteins [34]

WAVE2 protein complexes, known regulators of actin filament and lamellipodia formation [37, 38] while also suggesting involvement of KIAA1522, consistent with a recent suggestion for its involvement by Cho and colleagues [39]. To investigate the identified complexes interactively, visualizations are available for the 1157 learned complexes on the supporting website (see the Code and data availability section).

Of particular interest are complexes corresponding to proteins with low annotation scores, as finding the proteins in complexes with better-annotated proteins may help suggest potential functions for these otherwise minimally characterized proteins [40]. We searched specifically for such cases and highlighted complexes with uncharacterized proteins based on available UniProt annotations [41]. Some examples of learned complexes with uncharacterized or minimally characterized proteins are provided in Fig. 10.

For example, in Fig. 10A, C4orf19 (chromosome 4 open reading frame 19) is broadly expressed across human cell types and tissues [42], with high protein levels in the kidney, liver, and GI tract [43, 44] and while little is known about its function, an observed relationship between C4orf19 and colorectal cancer suggests that high expression levels might have some value as a marker for colorectal cancer [45], although elevated C4orf19 expression is also reported to show a favorable association with renal cancer survival [43, 44]. Notably, four of the other proteins in this cluster (PDCD10, STK24, STK25, STK26) are known to associate into a complex with roles in maintaining epithelial integrity [46, 47] and kidney water balance by regulating aquaporin trafficking and abundance

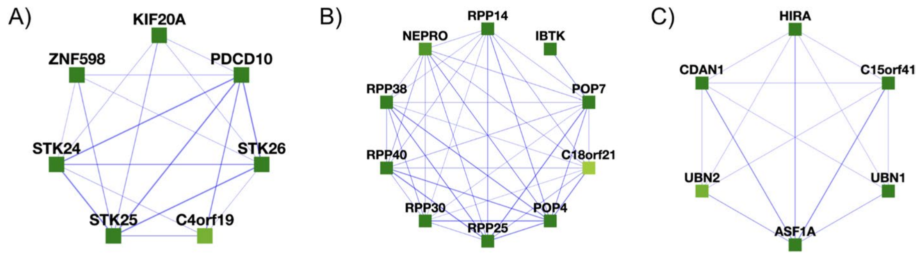

Fig. 10 Participation in protein complexes by the uncharacterized proteins C4orf19 and C18orf21 and the minimally characterized protein C15orf41. **A** We find C4orf19 to belong to a larger complex composed of KIF20A, C4orf19, PDCD10, STK25, ZNF598, STK26, and STK24. **B** C18orf21 is found in a complex with 50% similarity to the Rnase/Mrp complex. **C** C15orf41 is found in a complex with 30% similarity to the cytosolic Codanin-1-Asf1-H3.1-histone H4–importin-4 complex

in kidney tubule epithelial cells [48], suggesting a potential role for C4orf19 in normal kidney function. As for C4orf19, many of the proteins have been reported as potential biomarkers for bladder, gastric, pancreatic, and colorectal cancers [49–52].

As another example of a minimally characterized protein, C18orf21 (chromosome 18 open reading frame 21) is reported to possibly regulate the Rnase/Mrp complex, a ribonucleoprotein complex involved in RNA processing [53]. Both RL algorithm and Super. Complex concur on a connection for C18orf21 to RNA processing: from the learned complexes of Super.Complex, C18orf21 was found to be a part of a complex with a 50% overlap to the Rnase/Mrp complex, comprising all the proteins found in the RL algorithm's learned complex (Fig. 10B), adding support for this protein's possible function in ribonuclease P RNA binding. Further, the RL algorithm learns a similar complex (C18orf21, IBTK, RPP30, POP4, and RPP25L) on hu.MAP 2.0, adding additional support to C18orf21's function from the learned complexes on hu.MAP 1.0.

Somewhat more information can be gleaned for C15orf41 (chromosome 15 open reading frame 41), which, while minimally characterized, has recently been detected to interact with Codanin-1 (CDAN1) in human cells, and this interaction forms a tight, near stoichiometric complex [54]. Moreover, these studies reveal that mutation of C15orf41 can lead to the development of Congenital Dyserythropoietic type 1 disease (CDA-1) [54]. While its function is unknown, studies have noted a high sequence similarity between C15orf41 and archaeal Holliday junction resolvases, which are DNA repair enzymes that remove Holliday junctions [54], and it has been implicated in erythrocyte differentiation [55]. Its putative interaction partners within the complex (Fig. 10C), HIRA and ASF1A, cooperate to promote chromatin assembly [56], and HIRA, ASF1A, and UBN1/2 form a complex and function in histone deposition of variant H3.3 into chromatin, independent of DNA replication [57]. CDAN1 and C15orf41 mutations lead to similar erythroid phenotypes and they were both eliminated from the same animal taxa, suggesting that these 2 proteins may participate in a shared pathway [58].

To obtain more support for the overall physical association of proteins in this cluster, we modeled the 3D structure of the C15orf41-CDAN1 interaction using AlphaFold-Multimer [59], as implemented in Google Colab [60]. The AlphaFold model indicated a high-confidence interface spanning two distinct domains of CDAN1, one contributed from a domain spanning amino acids 1017–1203 and one covering one face of the larger N-terminal domain (2–997) centered on amino acids 427–472 and 843–997; these, in

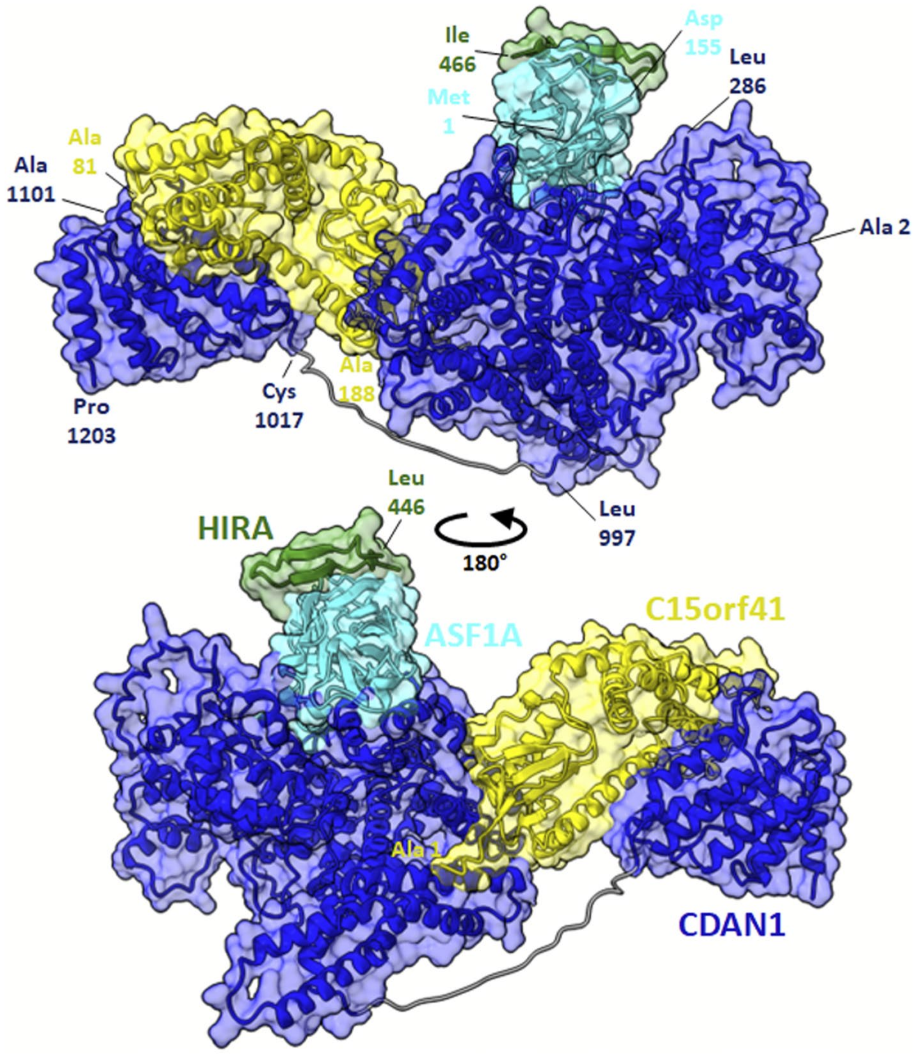

Fig. 11 Structural modeling supports C15orf41, CDAN1, ASF1A, and HIRA participating in a large multiprotein complex. Using AlphaFold-multimer, we find that all four proteins can be simultaneously accommodated within a single multiprotein complex, here showing C15orf41's modeled interaction with CDAN1 residues 2–74 and 286–1203, ASF1A residues 1–155, and HIRA residues 421–479. For illustration purposes, the known crystal structure of HIRA 446–466 [56] has been superimposed onto the AlphaFold model, which is available in full from the supporting GitHub repository with accompanying quality measurements

turn, interact with opposing surfaces of C15orf41 (Fig. 11). The predicted structure is consistent with the prior experimental observation that the C-terminal 227 residues of CDAN1 (residues 1000–1227) are critical for the interaction [54]. To investigate the possibility of additional direct interactions between the C15orf41-CDAN1 heterodimer and one or more of the remaining proteins in the cluster, we took advantage of an available X-ray crystal structure that delineated the ASF1A interaction with HIRA residues amino acids 446–466 (PDB entry 2I32) [56] to further evaluate a larger complex. Using AF2-multimer, we modeled C15orf41, CDAN1 residues 2–74 and 286–1203 (omitting the intrinsically disordered segments, as determined by [60]), ASF1A residues 1–155 (omitting the intrinsically disordered tail), and HIRA residues 421–479, a somewhat larger segment known to be critical for the interaction with ASF1A [55]. As illustrated in full in Fig. 11, AlphaFold suggested a binding site for ASF1A distinct from the C15orf41 binding site that, importantly, did not occlude the experimentally determined HIRA binding site, which AlphaFold also recapitulated. Thus, 3D structural modeling confirmed that four of the proteins in this cluster can be accommodated within the same overall multiprotein complex.

Finally, C11orf42 was found as a subunit in a complex (C11orf42, SNX1, SNX5, VPS29, SNX2, COMMD9) that corresponds to a subcomplex of the Retromer or SNX/BAR complex (e.g. as in [61]), with supporting independent evidence from a learned complex from Super.Complex (C11orf42, SNX1, SNX5, and VPS29). This indicates that C11orf42 may be involved in trafficking with Retromer complex proteins, a notion supported by its localization to intracellular vesicles similar in nature to the other proteins in the complex [43, 62–66]. Another example, C16orf91 constituted a complex (C16orf91, UQCC1, COX20, UQCC2) resembling a learned complex from Super.Complex (C16orf91, UQCC1, COX20).

Conclusions

In conclusion, we asked if reinforcement learning could be applied to learn to walk trajectories on a protein interaction network, and in this way more accurately determine protein complexes. Application of the method to currently available human protein interaction networks performed competitively with other algorithms, with comparable accuracy but notable savings in computational time, and in turn led to clear predictions of protein function and interactions for several uncharacterized human proteins. We could support at least one of these, C15orf41, with independent evidence from 3D structural modeling.

Three main avenues can be explored to improve the RL community detection algorithm: improving the subgraph representations, the RL formulation, and the candidate community search process.

We currently represent a subgraph by a single feature, its density, which gives a problem formulation with a small state space. While performance may be negatively impacted, to improve accuracy, more subgraph features could be included in addition to density. Examples of other subgraph features that could be added include edge weight statistics, node clustering coefficient statistics, and degree correlation statistics. However, as the number of features increases, the state space increases exponentially. For instance, if we incorporate 18 topological features with a discretization of each feature into 10 bins, we increase the number of states to 10^{18}. Different sample-based reinforcement learning methods could be applied to address this challenge and potentially give more accurate results.

The RL formulation could be modified to better accommodate overlapping community detection, by giving a positive reward if, while growing a seed node, a neighbor is present in any of the training complexes that can be built, rather than considering only one training community at a time. Note that in this scenario, the reward for a neighbor can change dynamically based on the possible communities that can be built from that step. Due to the dynamic rewards that need to be computed for each neighbor at each iteration, by checking whether the new subgraph is a part of any of the training communities, computational time would increase significantly compared to the current static reward system, however, it may improve accuracy. Also, different rewards and discount factors can be experimented with in the training phase of the algorithm.

Finally, in the candidate community search process, there may be scenarios where there are multiple highest-scoring neighbors to add at an iteration in the growth process of a subgraph. Currently, we have only added one of the highest-scoring neighbors to grow the complex. Each of the other highest-scoring neighbors can be added as well to grow complexes we may have missed in the current method.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-023-05425-7.

Additional file 1: This file contains the following supplementary figures and tables. **Figure S1.** Convergence of other scores from the training RL algorithm. **Table S1.** RL algorithm performance on training and testing toy complexes. **Table S2.** RL algorithm performance on training and testing hu.MAP 1.0 complexes. **Table S3.** RL algorithm performance on hu.MAP 2.0 complexes.

Acknowledgements

This research was funded by grants to E.M.M. from the National Institute of General Medical Sciences (R35 GM122480), National Institute of Child Health and Human Development (R01 HD085901), and Welch Foundation (F-1515).

Author contributions

EMM and MVP contributed to the conception and design of the work. MVP and RSP worked on the algorithm software. All authors analyzed and interpreted the data. All authors wrote the manuscript text. RSP and EMM prepared the figures. MVP and RSP prepared the tables. All authors reviewed the manuscript.

Funding

This research was funded by grants to E.M.M. from the National Institute of General Medical Sciences (R35 GM122480), National Institute of Child Health and Human Development (R01 HD085901), and Welch Foundation (F-1515).

Availability of data and materials

Code, available on GitHub repository: https://github.com/marcottelab/RL_complex_detection. Data: All learned complexes from hu.MAP 1.0 with their corresponding scores: https://marcottelab.github.io/RL_humap_prediction/humap/res_pred_names.txt. All learned complexes from hu.MAP 2.0 with their corresponding scores: https://marcottelab.github.io/RL_humap_prediction/humap2/res_pred_names_humap2.txt. Interactive visualizations of results on hu.MAP 1.0 and hu.MAP 2.0: To investigate the complexes identified by the RL algorithm interactively, visualizations are available for the learned complexes on hu.MAP 1.0 and hu.MAP 2.0 on this website: https://marcottelab.github.io/RL_humap_prediction/. The website also provides the functionality of sorting complexes and proteins by their annotation score and the number of interactions with SARS-CoV-2 proteins [67]. Structural model for the C15orf41, CDAN1, ASF1A, and HIRA heterotetramer, with sequences and assessments of model quality, can be downloaded from: https://github.com/marcottelab/RL_humap_prediction/ (Folder—CDIN1_CDAN1_2-74_286-1203_ASF1A_1-155_HIRA_421-479_1ModelRelaxed.zip).

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare no competing of interest.

Received: 5 February 2023 Accepted: 20 July 2023

Published online: 02 August 2023

References

1. Richards AL, Eckhardt M, Krogan NJ. Mass spectrometry-based protein–protein interaction networks for the study of human diseases. Mol Syst Biol. 2021;17(1):e8792. https://doi.org/10.15252/msb.20188792.
2. Titeca K, Lemmens I, Tavernier J, Eyckerman S. Discovering cellular protein–protein interactions: technological strategies and opportunities. Mass Spectrom Rev. 2019;38(1):79–111. https://doi.org/10.1002/mas.21574.
3. Smits AH, Vermeulen M. Characterizing protein–protein interactions using mass spectrometry: challenges and opportunities. Trends Biotechnol. 2016;34(10):825–34. https://doi.org/10.1016/j.tibtech.2016.02.014.

4. Snider J, Kotlyar M, Saraon P, Yao Z, Jurisica I, Stagljar I. Fundamentals of protein interaction network mapping. Mol Syst Biol. 2015;11(12):848. https://doi.org/10.15252/msb.20156351.
5. Cafarelli TM, Desbuleux A, Wang Y, Choi SG, De Ridder D, Vidal M. Mapping, modeling, and characterization of protein–protein interactions on a proteomic scale. Curr Opin Struct Biol. 2017;44:201–10. https://doi.org/10.1016/j.sbi.2017.05.003.
6. Drew K, et al. Integration of over 9000 mass spectrometry experiments builds a global map of human protein complexes. Mol Syst Biol. 2017;13(6):932. https://doi.org/10.15252/msb.20167490.
7. Drew K, Wallingford JB, Marcotte EM. huMAP 2.0: integration of over 15,000 proteomic experiments builds a global compendium of human multiprotein assemblies. Mol Syst Biol. 2021;17(5):e10016. https://doi.org/10.15252/msb.202010016.
8. Malovannaya A, et al. Analysis of the human endogenous coregulator complexome. Cell. 2011;145(5):787–99. https://doi.org/10.1016/j.cell.2011.05.006.
9. Hein MY, et al. A human interactome in three quantitative dimensions organized by stoichiometries and abundances. Cell. 2015;163(3):712–23. https://doi.org/10.1016/j.cell.2015.09.053.
10. Huttlin EL, et al. The BioPlex network: a systematic exploration of the human interactome. Cell. 2015;162(2):425–40. https://doi.org/10.1016/j.cell.2015.06.043.
11. Huttlin EL, et al. Architecture of the human interactome defines protein communities and disease networks. Nature. 2017;545(7655):7655. https://doi.org/10.1038/nature22366.
12. Wan C, et al. Panorama of ancient metazoan macromolecular complexes. Nature. 2015;525(7569):7569. https://doi.org/10.1038/nature14877.
13. Kirkwood KJ, Ahmad Y, Larance M, Lamond AI. Characterization of native protein complexes and protein isoform variation using size-fractionation-based quantitative proteomics. Mol Cell Proteom MCP. 2013;12(12):3851–73. https://doi.org/10.1074/mcp.M113.032367.
14. Kristensen AR, Gsponer J, Foster LJ. A high-throughput approach for measuring temporal changes in the interactome. Nat Methods. 2012;9(9):907–9. https://doi.org/10.1038/nmeth.2131.
15. Havugimana PC, et al. A census of human soluble protein complexes. Cell. 2012;150(5):1068–81. https://doi.org/10.1016/j.cell.2012.08.011.
16. Javed MA, Younis MS, Latif S, Qadir J, Baig A. Community detection in networks: a multidisciplinary review. J Netw Comput Appl. 2018;108:87–111. https://doi.org/10.1016/j.jnca.2018.02.011.
17. Bader GD, Hogue CW. An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinform. 2003;4(1):2. https://doi.org/10.1186/1471-2105-4-2.
18. Liu G, Wong L, Chua HN. Complex discovery from weighted PPI networks. Bioinformatics. 2009;25(15):1891–7. https://doi.org/10.1093/bioinformatics/btp311.
19. Wu M, Li X, Kwoh C-K, Ng S-K. A core-attachment based method to detect protein complexes in PPI networks. BMC Bioinform. 2009;10(1):169. https://doi.org/10.1186/1471-2105-10-169.
20. Nepusz T, Yu H, Paccanaro A. Detecting overlapping protein complexes in protein-protein interaction networks. Nat Methods. 2012;9(5):5. https://doi.org/10.1038/nmeth.1938.
21. Lee C, Reid F, McDaid A, Hurley N. Detecting highly overlapping community structure by greedy clique expansion. 2010. arXiv: arXiv:1002.1827, https://doi.org/10.48550/arXiv.1002.1827.
22. Hu L, Yang Y, Tang Z, He Y, Luo X. FCAN-MOPSO: an improved fuzzy-based graph clustering algorithm for complex networks with multi-objective particle swarm optimization. IEEE Trans Fuzzy Syst. 2023. https://doi.org/10.1109/TFUZZ.2023.3259726.
23. Hu L, Yuan X, Liu X, Xiong S, Luo X. Efficiently detecting protein complexes from protein interaction networks via alternating direction method of multipliers. IEEE/ACM Trans Comput Biol Bioinform. 2019;16(6):1922–35. https://doi.org/10.1109/TCBB.2018.2844256.
24. Hu L, Zhang J, Pan X, Yan H, You Z-H. HiSCF: leveraging higher-order structures for clustering analysis in biological networks. Bioinformatics. 2021;37(4):542–50. https://doi.org/10.1093/bioinformatics/btaa775.
25. Omranian S, Angeleska A, Nikoloski Z. PC2P: parameter-free network-based prediction of protein complexes. Bioinformatics. 2021;37(1):73–81. https://doi.org/10.1093/bioinformatics/btaa1089.
26. Wang R, Wang C, Ma H. Detecting protein complexes with multiple properties by an adaptive harmony search algorithm. BMC Bioinform. 2022;23:414. https://doi.org/10.1186/s12859-022-04923-4.
27. Meng X, Xiang J, Zheng R, Wu F-X, Li M. DPCMNE: detecting protein complexes from protein-protein interaction networks via multi-level network embedding. IEEE ACM Trans Comput Biol Bioinform. 2022;19(3):1592–602. https://doi.org/10.1109/TCBB.2021.3050102.
28. Qi Y, Balem F, Faloutsos C, Klein-Seetharaman J, Bar-Joseph Z. Protein complex identification by supervised graph local clustering. Bioinformatics. 2008;24(13):i250–68. https://doi.org/10.1093/bioinformatics/btn164.
29. Dong Y, Sun Y, Qin C. Predicting protein complexes using a supervised learning method combined with local structural information. PLoS ONE. 2018;13(3):e0194124. https://doi.org/10.1371/journal.pone.0194124.
30. Palukuri MV, Marcotte EM. Super.Complex: a supervised machine learning pipeline for molecular complex detection in protein-interaction networks. PLoS ONE. 2021;16(12):e0262056. https://doi.org/10.1371/journal.pone.0262056.
31. Paim EC, Bazzan ALC, Chira C. Detecting communities in networks: a decentralized approach based on multiagent reinforcement learning. In 2020 IEEE symposium series on computational intelligence (SSCI); 2020. pp. 2225–2232. doi: https://doi.org/10.1109/SSCI47803.2020.9308197.
32. Bryant P, Pozzati G, Zhu W, Shenoy A, Kundrotas P, Elofsson A. Predicting the structure of large protein complexes using AlphaFold and Monte Carlo tree search. Nat Commun. 2022;13(1):1. https://doi.org/10.1038/s41467-022-33729-4.
33. Burke DF, et al. Towards a structurally resolved human protein interaction network. Nat Struct Mol Biol. 2023;30(2):2. https://doi.org/10.1038/s41594-022-00910-8.
34. Giurgiu M, et al. CORUM: the comprehensive resource of mammalian protein complexes—2019. Nucleic Acids Res. 2019;47(D1):D559–63. https://doi.org/10.1093/nar/gky973.
35. Arroyo JD, et al. A genome-wide CRISPR death screen identifies genes essential for oxidative phosphorylation. Cell Metab. 2016;24(6):875–85. https://doi.org/10.1016/j.cmet.2016.08.017.

36. Wolfson RL, et al. KICSTOR recruits GATOR1 to the lysosome and is necessary for nutrients to regulate mTORC1. Nature. 2017;543(7645):438–42. https://doi.org/10.1038/nature21423.
37. Suetsugu S, Miki H, Takenawa T. Identification of two human WAVE/SCAR homologues as general actin regulatory molecules which associate with the Arp2/3 complex. Biochem Biophys Res Commun. 1999;260(1):296–302. https://doi.org/10.1006/bbrc.1999.0894.
38. Weiner OD, et al. Hem-1 complexes are essential for Rac activation, actin polymerization, and myosin regulation during neutrophil chemotaxis. PLoS Biol. 2006;4(2):e38. https://doi.org/10.1371/journal.pbio.0040038.
39. Cho NH, et al. OpenCell: endogenous tagging for the cartography of human cellular organization. Science. 2022;375(6585):eabi6983. https://doi.org/10.1126/science.abi6983.
40. Kustatscher G, et al. Understudied proteins: opportunities and challenges for functional proteomics. Nat Methods. 2022;19(7):774–9. https://doi.org/10.1038/s41592-022-01454-x.
41. UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Res. 2021;49(D1):D480–9. https://doi.org/10.1093/nar/gkaa1100.
42. "C4orf19 expression in human." https://bgee.org/gene/ENSG00000154274. Accessed 20 May 2022.
43. Thul PJ, et al. A subcellular map of the human proteome. Science. 2017;356(6340):l3321. https://doi.org/10.1126/science.aal3321.
44. "Tissue expression of C4orf19-Summary-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000154274-C4orf19/tissue. Accessed 16 June 2022.
45. Wang W, et al. Down-regulated C4orf19 confers poor prognosis in colon adenocarcinoma identified by gene co-expression network. J Cancer. 2022;13(4):1145–59. https://doi.org/10.7150/jca.63635.
46. Zheng X, et al. CCM3 signaling through sterile 20-like kinases plays an essential role during zebrafish cardiovascular development and cerebral cavernous malformations. J Clin Invest. 2010;120(8):2795–804. https://doi.org/10.1172/JCI39679.
47. Goudreault M, et al. A PP2A phosphatase high density interaction network identifies a novel striatin-interacting phosphatase and kinase complex linked to the cerebral cavernous malformation 3 (CCM3) protein. Mol Cell Proteom MCP. 2009;8(1):157–71. https://doi.org/10.1074/mcp.M800266-MCP200.
48. Wang R, et al. Pdcd10-Stk24/25 complex controls kidney water reabsorption by regulating Aqp2 membrane targeting. JCI Insight. 2021;6(12):e142838. https://doi.org/10.1172/jci.insight.142838.
49. Xiong M, et al. KIF20A promotes cellular malignant behavior and enhances resistance to chemotherapy in colorectal cancer through regulation of the JAK/STAT3 signaling pathway. Aging. 2019;11(24):11905–21. https://doi.org/10.18632/aging.102505.
50. Stangel D, et al. Kif20a inhibition reduces migration and invasion of pancreatic cancer cells. J Surg Res. 2015;197(1):91–100. https://doi.org/10.1016/j.jss.2015.03.070.
51. "PDCD10 programmed cell death 10 [Homo sapiens (human)]-Gene-NCBI." https://www.ncbi.nlm.nih.gov/gene/11235. Accessed 31 May 2022.
52. Hsu H-P, Wang C-Y, Hsieh P-Y, Fang J-H, Chen Y-L. Knockdown of serine/threonine-protein kinase 24 promotes tumorigenesis and myeloid-derived suppressor cell expansion in an orthotopic immunocompetent gastric cancer animal model. J Cancer. 2020;11(1):213–28. https://doi.org/10.7150/jca.35821.
53. Liang L, Chen V, Zhu K, Fan X, Lu X, Lu S. Integrating data and knowledge to identify functional modules of genes: a multilayer approach. BMC Bioinform. 2019;20(1):225. https://doi.org/10.1186/s12859-019-2800-y.
54. Shroff M, Knebel A, Toth R, Rouse J. A complex comprising C15ORF41 and Codanin-1: the products of two genes mutated in congenital dyserythropoietic anaemia type I (CDA-I). Biochem J. 2020;477(10):1893–905. https://doi.org/10.1042/BCJ20190944.
55. Russo R, et al. Characterization of two cases of congenital dyserythropoietic anemia type I shed light on the uncharacterized C15orf41 protein. Front Physiol. 2019. https://doi.org/10.3389/fphys.2019.00621.
56. Tang Y, et al. Structure of a human ASF1a/HIRA complex and insights into specificity of histone chaperone complex assembly. Nat Struct Mol Biol. 2006;13(10):921–9. https://doi.org/10.1038/nsmb1147.
57. Rai TS, et al. Human CABIN1 Is a functional member of the human HIRA/UBN1/ASF1a histone H3.3 chaperone complex. Mol Cell Biol. 2011;31(19):4107–18. https://doi.org/10.1128/MCB.05546-11.
58. Swickley G, et al. Characterization of the interactions between Codanin-1 and C15Orf41, two proteins implicated in congenital dyserythropoietic anemia type I disease. BMC Mol Cell Biol. 2020;21(1):18. https://doi.org/10.1186/s12860-020-00258-1.
59. Evans R, et al. Protein complex prediction with AlphaFold-Multimer. bioRxiv. 2021. https://doi.org/10.1101/2021.10.04.463034.
60. Mirdita M, Schütze K, Moriwaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: making protein folding accessible to all. Nat Methods. 2022;19(6):679–82. https://doi.org/10.1038/s41592-022-01488-1.
61. Wassmer T, et al. The retromer coat complex coordinates endosomal sorting and dynein-mediated transport, with carrier recognition by the trans-Golgi network. Dev Cell. 2009;17(1):110–22. https://doi.org/10.1016/j.devcel.2009.04.016.
62. "Subcellular-C11orf42-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000180878-C11orf42/subcellular. Accessed 16 June 2022.
63. "Subcellular-SNX5-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000089006-SNX5/subcellular. Accessed 16 June 2022.
64. "Subcellular-VPS29-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000111237-VPS29/subcellular. Accessed 16 June 2022.
65. "Subcellular-SNX2-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000205302-SNX2/subcellular. Accessed 16 June 2022.
66. "Subcellular-SNX1-The Human Protein Atlas." https://www.proteinatlas.org/ENSG00000028528-SNX1/subcellular. Accessed 16 June 2022.
67. Gordon DE, et al. A SARS-CoV-2 protein interaction map reveals targets for drug repurposing. Nature. 2020;583(7816):459–68. https://doi.org/10.1038/s41586-020-2286-9.

Publisher's Note