Erratum to: MINE: Module Identification in Networks

Kahn Rhrissorrakrai¹ and Kristin C. Gunsalus²*

Erratum
It was brought to our attention that there was a discrepancy between the description and implementation of the MINE algorithm in our article [1]. We regret any inconvenience that may have resulted from this inaccuracy.

The primary difference is that as implemented, only the immediate neighborhood of each node is searched to identify initial clusters, and the merging process then facilitates cluster growth and removal of redundant clusters. This implementation results in accelerated cluster prediction relative to a more exhaustive search, which would perform comparably to a depth-first search across a larger portion of the network for each node. The published results and performance of MINE are unaffected; the description of the method, algorithm pseudocode, Fig. 1, and Additional file 1: Figure S1 have been updated to accurately reflect the implemented version of MINE.

Scoring
MINE receives as input any number of interaction files. The network is treated as an undirected, unweighted graph. Each vertex v in the graph G = (V, E) is then weighted based upon its local neighborhood N, defined as the set all vertices connected directly to v (at a depth of 1); we call the set N inclusive of v itself {N∪v}, which we denote simply as N∪v. The vertex weight (vw) is the product of the maximal number of edges connected to any single node in N∪v (kmax) and the density of N∪v (d): vw = kmax * d. Density is calculated as d = 2 * eN∪V / (VN∪V * (VN∪V - 1)), where VN∪V is the number of vertices in N∪v (i.e. v and its direct neighbors) and eN∪V is the number of edges in N∪v. A cluster (C) is then established by iterating through each vertex in N∪v, in order of highest to lowest weight, and adding neighbors if either of two criteria are satisfied: A) the neighbor vertex weight is above a minimum threshold (as determined by the user-defined vertex weight percentage (vwp) of the seed vertex) and does not decrease the cluster modularity score (by an amount equal to or greater than the user-defined modularity score percentage (msp)); B) the modularity score for the cluster is improved by msp. Cluster modularity (Cmod) is defined as the ratio of edges between nodes of a cluster (Ein) and edges between cluster members and non-members (Eout): Cmod = Ein/Eout - The process is performed for all vertices in N∪v, and the resulting cluster can be optionally refined by removing all vertices with k = 1 (if the flag Trim is set). By default, clusters are non-exclusive (i.e. members are allowed to participate in several clusters); a new cluster will be merged with any existing cluster if their vertices overlap by >50 %, otherwise it will be stored as a new cluster. Each (new or merged) cluster is then scored (Cv) as the product of its density (d) and the number of members in the cluster (VC): Cv = d * VC. The cluster identification process is repeated for each vertex in V in order of descending vw. The final set of clusters is evaluated for improvements in modularity scores when individual members are excluded, and the membership of each cluster is updated accordingly (by removing nodes from a cluster if the modularity score increases by more than msp in its absence).

¹ Correspondence: kcg1@nyu.edu
²Center for Genomics and Systems Biology, Department of Biology, New York University, 12 Waverly Place, New York, NY 10003, USA
Full list of author information is available at the end of the article.

© 2016 Rhrissorrakrai and Gunsalus. Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.
**Algorithm**

1. **Vertex Weighting**

   procedure Vertex-Weighting
   
   input: graph: $G = (V,E)$
   
   for all $v$ in $G$
   
   $N$ = set of immediate neighbors of $v$ (depth = 1)
   
   $k_{\text{max}}$ = maximum number of edges from any one vertex in set $N \cup v$
   
   $d$ = density of $N \cup v$
   
   $v_w = \text{weight} = k_{\text{max}} \cdot d$
   
   end for
   
   end procedure

2. **Cluster Prediction**

   procedure Cluster-Prediction
   
   input: graph: $G = (V,E)$; vertex weight: $v_w$; vertex weight percentage: $vwp$; modularity score percentage: $msp$; merge percentage: $mp$
   
   for $v$ $\in$ $V_w$ (from high $\rightarrow$ low weight)
   
   $N$ = set of immediate neighbors of $n$ (depth = 1)
   
   push (tocheck, $v$)
   
   push (tocheck, $N$)
   
   for $n$ $\in$ tocheck
   
   if $v_w$ of $n$ $\geq$ ($v_w$ of $v$)($1 - vwp)$ then
   
   if modularity-score($C \cup n$) $>$ modularity-score($C$) - modularity-score($C$)*$msp$ then
   
   add $n$ to cluster $C$
   
   else if modularity-score($C \cup n$) $>$ modularity-score($C$) + modularity-score($C$)*$msp$
   
   add $n$ to cluster $C$
   
   end if
   
   end if
   
   end for
   
   if trim == true then call: Trim ($C$)
   
   if percent overlap $C$ with existing cluster(s) $\geq mp$
   
   Merge($C$) with existing cluster(s)
   
   $C_{\text{score}} = \text{density}(C) \cdot \text{sizeof}(C)$
   
   end for

   procedure ModularityCleanup
   
   input: cluster: $C$
   
   $v_r = v$ that maximizes modularity-score($(C \setminus v)$)
   
   while (sizeof($C$) $>$ 3 and (modularity-score($(C \setminus v_r)$) $>$ modularity-score($C$) + modularity-score($C$)*$msp$))
   
   remove $v_r$ from $C$
   
   $v_r = v$ that maximizes modularity-score($(C \setminus v)$)
   
   end while
   
   end procedure

   procedure Trim
   
   input: cluster: $C$
   
   for all $v$ in $C$
   
   if $k$ of $v$ $<$ 2 then remove $v$ from $C$
   
   end for
   
   end procedure

   procedure modularity-score
   
   input: cluster: $C$
   
   $in = \text{number of edges exclusively between members of } C$
   
   $out = \text{number of edges exclusively between members and non-members of } C$
   
   score $= \text{in/out}$
   
   end procedure

   for $C$ $\in$ AllClusters
   
   ModularityCleanup($C$)
   
   end for

   end procedure

**Additional file**

**Additional file 1: Figure S1.** Expanded Conceptual Overview of MINE Procedure. The general procedure for the core MINE algorithm is presented with an example input network, $vwp = 0.4$ and $msp = 0.9$. The clustering procedure begins with the highest weighted node, where node weights are assigned as $v_w = k_{\text{max}} \cdot d$ ($v_w$ = vertex weight; $k_{\text{max}}$ = largest edge count).
within node's local neighborhood, $d = \text{density}$. Nodes are numbered in order visited. Current clusters (blue) are compared to candidate clusters (orange). A node is added to a growing cluster if it passes either of the following criteria: 1) its $v_w$ is within the specified range, or 2) the new cluster modularity ($m$) is within the specified range. The values of $v_w$ and $m$ for current and candidate clusters are indicated below each illustrated step; a check mark (or “x”) is placed next to each value if addition of the new node passes (or fails) the corresponding test. After all possible candidate nodes are visited, the preliminary cluster is processed to remove singly-connected nodes and merged with existing clusters if amount of overlap meets the user-defined threshold. (PDF 338 kb)

Acknowledgement
We would like to thank Dan Kortschak from the University of Adelaide, who pointed out the discrepancy in our article to us.

Author details
1IBM Computational Biology Center, 1101 Kitchawan Rd., Yorktown Heights, NY 10598, USA. 2Center for Genomics and Systems Biology, Department of Biology, New York University, 12 Waverly Place, New York, NY 10003, USA.

Received: 1 February 2016 Accepted: 1 February 2016 Published: 17 February 2016

Reference
1. Rhrissorrakrai K, Gunsalus KC. MINE: Module Identification in Networks. BMC Bioinformatics. 2011;12:192.