Community detection in weighted brain connectivity networks beyond the resolution limit
Supplementary Information

Carlo Nicolini, Cécile Bordier, Angelo Bifone

1 Methods

1.1 Detailed description of PACO

Figure S1 describes the details of the PACO algorithm for Surprise and Asymptotical Surprise Optimization. The function PACO takes as input a graph G and returns the nodes community membership vector C. Line 1 initializes the value of Surprise to 0. Line 2 assign to each node in the graph its community. Line 3 creates a list of edges E' sorted in decreasing order by their Jaccard coefficient. Line 4 iterates on every edge e=(u,v) and at line 6 checks if the endpoints they share the same community. Line 8 copies the membership vector to a temporary vector C'. Lines 9-13 choose randomly at chance if to put node u in the community of v or viceversa. Line 14 computes the new value of Surprise S' from the just updated community membership C. The function COMPUTESURPRISE returns the value of Surprise for graph G and partition C. Lines 15 to 18 checks if the new value of Surprise S' is greater than the previously stored value S and update Surprise and the membership vector, otherwise continue to the next edge. Line 19 returns the final community membership assignment.

```
PACO(G)
 1  S ← 0 ▷ Initialize Surprise to 0
 2  C ← (1, ..., |V|) ▷ Initialize membership vector
 3  E' ← SORT-JACCARD(E) ▷ Sort edges in decreasing order by Jaccard index
 4  for each edge (u, v) in E'
 5       do
 6          if C[u] ≠ C[v] ▷ try to move nodes only if in different communities
 7            then
 8                 C' ← C ▷ Create a temporary membership vector
 9                 if UNIFRAND(0,1) < 0.5
10                   then
11                        C'[v] ← C[u]
12                   else
13                        C'[u] ← C[v]
14                 S' = COMPUTESURPRISE(G,C')
15                    do if S' > S
16                         then
17                              C ← C' ▷ update membership
18                              S' ← S ▷ update Surprise
19  return C
```

Figure S1: Pseudocode of the PACO algorithm.

1.2 Detailed description of FAGSO

Figure S2 describes the details of the FAGSO algorithm for Surprise Optimization, as explained in [1].

FAGSO(G)

```
 1  S ← 0 ▷ Initialize Surprise to 0
 2  D ← ∅ ▷ Initialize disjoint set forest
 3  for each vertex v in V[G]
 4       do MAKE-SET(v)
 5  E' ← SORT-JACCARD(E) ▷ Sort edges in decreasing order by Jaccard index
 6  for each edge(u, v) ∈ E', ▷ Taken in decreasing order by Jaccard index
 7       do if FIND-SET(u) ≠ FIND-SET(v)
 8            then if SURPRISE(G, D ∪ {(u, v)}) > S
 9                 D ← D ∪ {(u, v)}
10                 UNION(U,V) ▷ Merge the communities u and v belong
11                 S = SURPRISE(G, D) ▷ Update current Surprise
12  return D
```

MAKE-SET(x)

```
1  p[x] ← x
2  rank[x] ← 0
```

UNION(x, y)

```
1  LINK(FIND-SET(x),FIND-SET(y))
```

LINK(x, y)

```
1  if rank[x] > rank[y]
2     then p[y] ← x
3     else p[x] ← y
4          if rank[x] = rank[y]
5            then rank[y] ← rank[y] + 1
```

FIND-SET(x)

```
1  if x ≠ p[x]
2     then p[x] ← FIND-SET(p[x])
3  return p[x]
```

SURPRISE(G, D)

```
1  m_ξ ← 0 ▷ Number of intracluster edges
2  p_ξ ← 0 ▷ Number of intracluster pairs of vertices
3  m ← |E[G]| ▷ Number of edges
4  p ← (|V[G]| choose 2) ▷ Number of pairs of vertices
5  for each g in CONNECTED-COMPONENTS-SUBGRAPHS(D, G)
6       do m_ξ ← m_ξ + |E[g]|
7          p_ξ ← p_ξ + (|V[g]| choose 2)
8  return −log10 ( Σ_{i=m_ς}^{m} [ (p_ξ choose i)(p − p_ξ choose m − i) / (p choose m) ] )
```

Figure S2: Pseudocode of the FAGSO algorithm together with auxiliary functions to work on the disjoint-set data structure.

1.3 Comparison of PACO and FAGSO

FAGSO is an agglomerative optimization algorithm that builds on a variation of the Kruskal algorithm for minimum spanning tree and is described in [1]. The first step of this method consists in ranking the edges in the graph in decreasing order by the Jaccard index of the neighbors of their two endpoints vertices. An union-find data structure is used to hold the community structure throughout the computation. At the beginning, each community consists only of one vertex. Then, starting from the edge with the highest Jaccard index at the top of the list, the endpoints are

attributed to the same community by disjoint-set union if this operation leads to a strictly better Surprise and if they do not belong already in the same community. This step is repeated for all edges and the final community structure is returned in the disjoint-set. This method finds partitions with high Surprise and it is deterministic, unless two edges with the same Jaccard index are found. In this case, ties are broken at random.

PACO and FAGSO are based on the same idea of greedy agglomeration of edges that leads to increment in Surprise, but the implementation of the agglomeration step is different. In the box A of Figure S3 both the algorithms are considering whether to merge nodes from edge $d - g$ into the same community or not. FAGSO consider the operation of merging nodes d, g in the same community by joining red and blue communities together in one larger module and proceeds if this leads to higher Surprise. PACO instead works at node level, considering whether to randomly move node g in the red community (Box C) or node d in the blue community (Box D). PACO chooses between the two options the one that leads to the highest increment in Surprise. This detail allows PACO to explore more finely the landscape of optimization as it has more run to run variability. Additionally the internal implementation of PACO is faster as it stores the community structure as an array of integers representing the node affiliations, while FAGSO was storing the community structure as a set of nodes for every community.

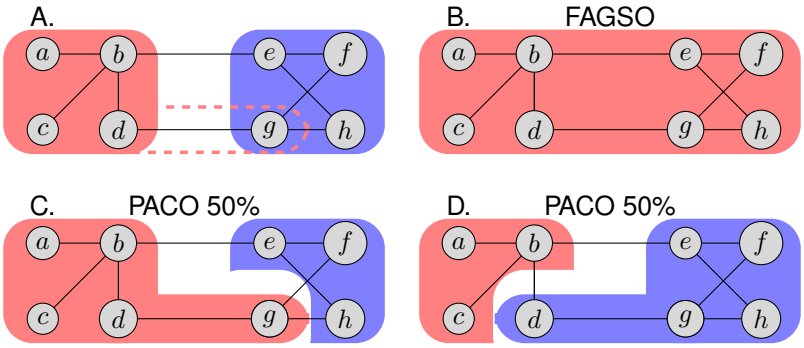

Figure S3: Difference in atomic operation between PACO and FAGSO algorithms.

1.4 Power law ring of cliques

The modified ring of clique network as indicated in the main text is composed by a ring structure of cliques whose sizes are randomly sampled from a power-law distribution with exponent $\tau_c = 1$. The power-law distribution of sizes of cliques in this benchmark is consistent with the heterogeneous size of modules observed in brain connectivity using the binary version of Surprise. An example realization of the benchmark network is shown in Figure S4.

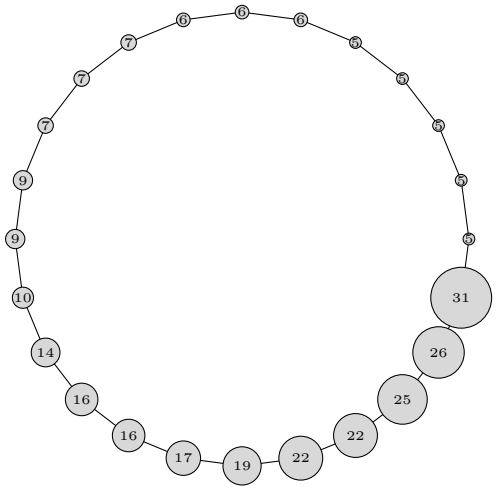

Figure S4: Power-law ring of cliques. Every circle is a complete graph (clique). Every clique is connected to its neighbors with one single link. The sizes of the cliques are randomly sampled from a power-law distribution with minimum clique size $\min_c = 5$, maximum clique size $\max_c = 75$, $n = 150$, $\tau_c = 1$.

2 Results

2.1 Consensus analysis

We applied consensus clustering to check whether the limited performance of Modularity was due to an inadequate choice of the optimal partition or if its inability to detect heterogeneously sized modules could be attributed to the resolution limit. We applied consensus clustering on our synthetic networks based on the ring of clique benchmarks at different levels of noise and number of subjects as shown in Figure S5, using the approach suggested by Lancichinetti [2]. We used $n_p = 15$ independent optimal partitions by means of Modularity maximization with Louvain algorithm, computed the consensus matrix P where P_{ij} was the number of times that node i and node j clustered together, applied a threshold of $\tau = 0.5$ to make the consensus matrix more sparse and then repeated these two steps upon convergence of P. The results suggested that the beneficial effects of consensus are negligible in this kind of synthetic fMRI-like networks and that the limitations of Newman's Modularity are indeed to be attributed to its resolution limit.

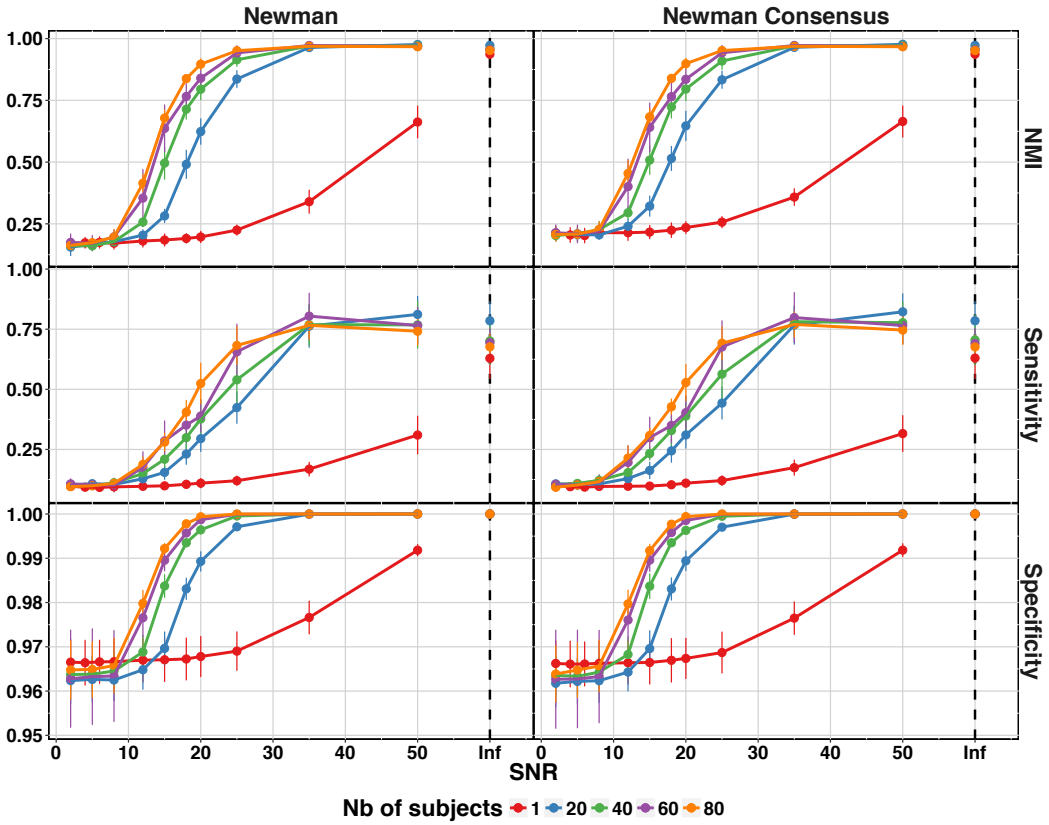

Figure S5: Consensus clustering with Newman's Modularity.

2.2 Accuracy and Matthew Correlation Coefficient

Together with NMI, Sensitivity and Specificity we also analyzed Accuracy and Matthew Correlation Coefficient for both the modified ring of cliques and LFR benchmark networks. Accuracy (Acc) and Matthew Correlation Coefficient (MCC) are defined on the basis of the confusion matrix as:

$$\mathrm{Acc} = \frac{(TP + TN)}{TP + FP + TN + FN} \qquad \mathrm{MCC} = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP, TN, FP, FN are the number of true positives, true negatives, false positives and false negatives identified in the detected partition with respect to the planted partition.

Accuracy takes in account the proportion of correctly classified samples and can present relatively high values even in the case of poorly performing detection methods when the classes have very different size. The Matthew Correlation Coefficient takes into account true and false positives and negatives. It's a balanced coefficient, to use especially when classes are very imbalanced.

As shown in Figure S6, Accuracy of Newman's Modularity is lower for small SNRs and number of subjects and in any case it does not reach 100% of true positives classification even in the no-noise condition. Infomap accuracy is high for SNR greater than 20, largely independent on the number of subjects. The large variance of Infomap for low SNRs is due to the merging of all nodes in a single large community in a few runs, as discussed in the main text. Asymptotical Surprise, behaves well in terms of Accuracy and has the least variability across all methods, plateauing at 100% for SNRs greater than 20 and more than 40 subjects.

In terms of MCC, Figure S6 shows that Asymptotical Surprise behaves comparably or better than Infomap. Newman's modularity is the least performer, due to its resolution limit, and never reaches maximum MCC. Interestingly, Asymptotical Surprise slightly outperforms Infomap in terms of MCC in the single subject case in the no-noise condition. The comparison of the three methos is similar in the case of the LFR benchmark, as shown in Figure S7.

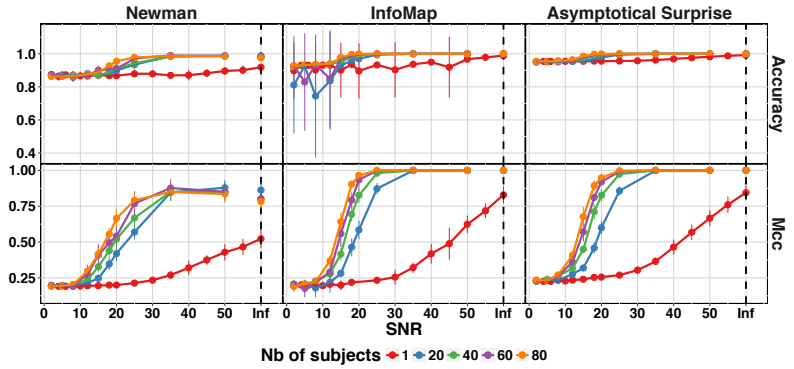

Figure S6: Accuracy and Matthew correlation coefficient on the modified ring of clique benchmark.

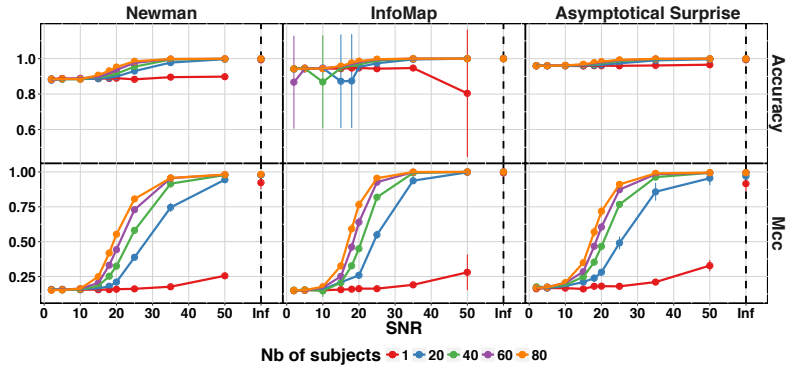

Figure S7: Accuracy and Matthew correlation coefficient on the LFR benchmark.

2.3 Resting state functional connectivity dataset

We applied PACO, Infomap and Newman's Modularity optimization to the resting state functional connectivity as described in the main text. In this supplementary information we report in figures S8, S9, S10, the full partitions for all methods that are not reported in the main text in the interest of space.

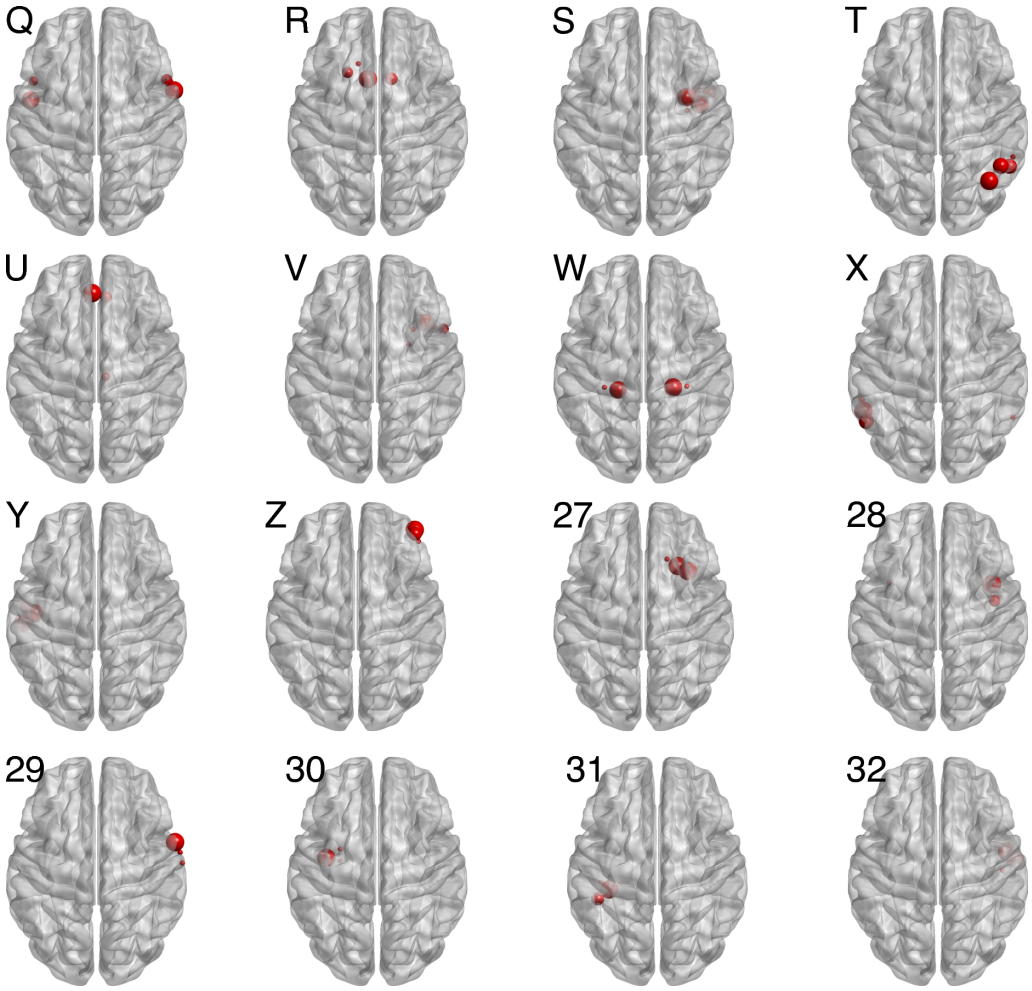

Figure S8: Communities found by Asymptotical Surprise optimization on the resting state dataset indexed by size. Modules ranked by size from the 17th to the 32th largest are reported to accompany the sixteen largest modules shown in the main article. Communities smaller than five nodes are not reported.

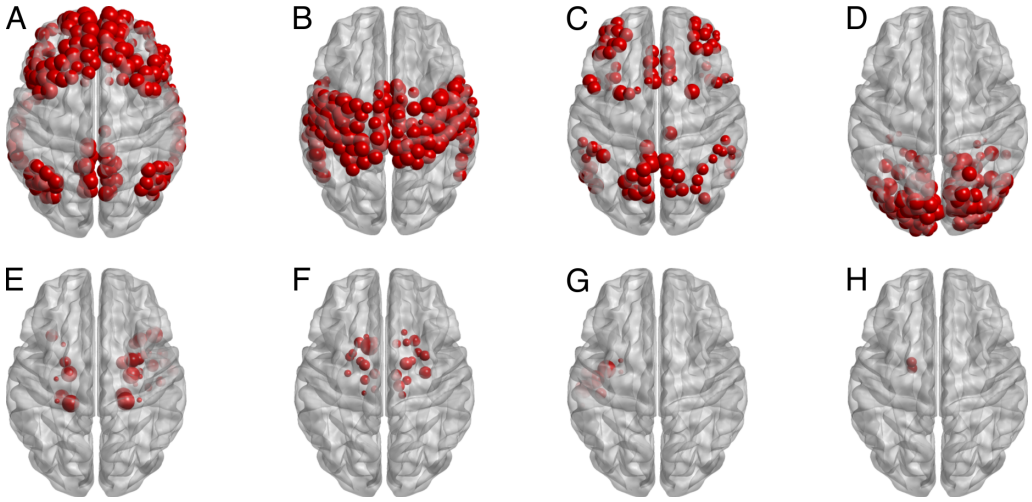

Figure S9: First largest 8 modules of the optimal partition as found by Newman's Modularity, with a value of the Modularity $Q = 0.4967$. Module 9 is not shown as it consists of a single node.

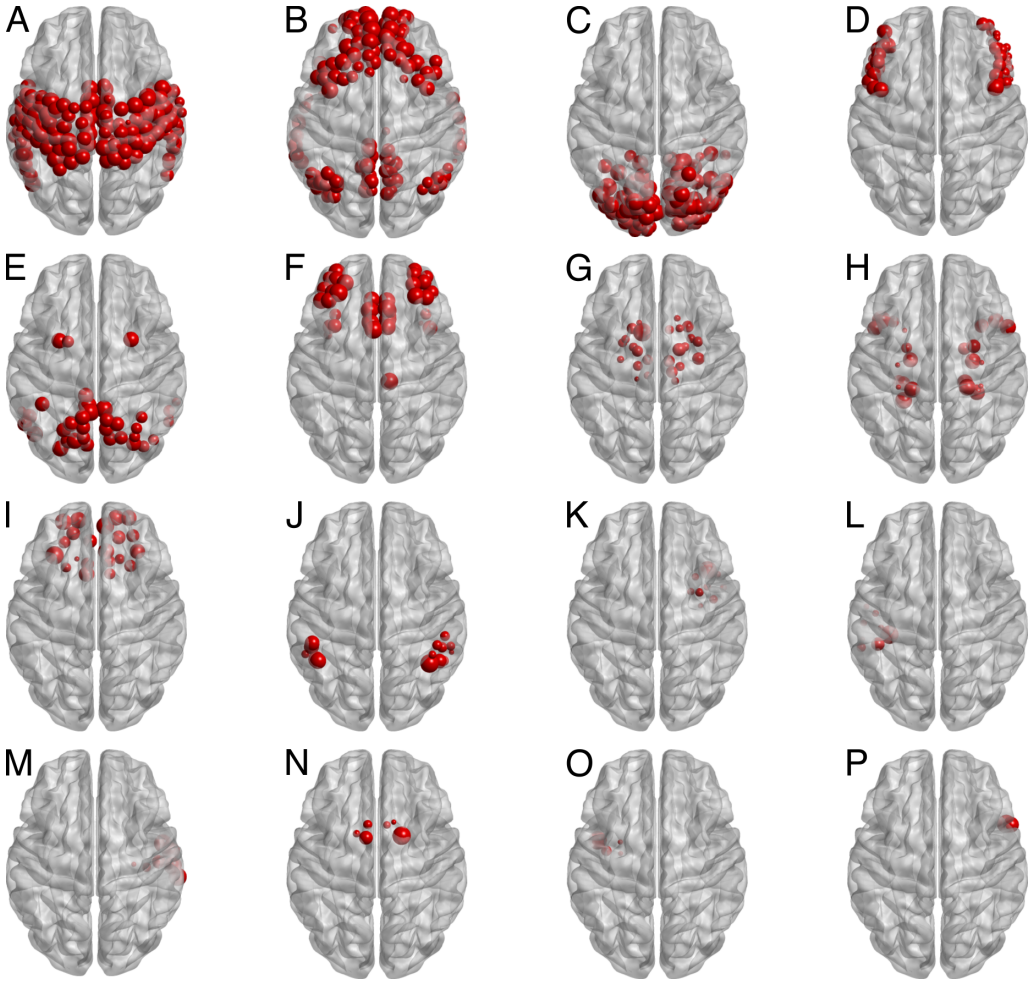

Figure S10: First largest 16 modules of the optimal partition as found by Infomap, with a value of the description length $L = 8.5173$.

2.4 Influence of LFR parameters on community detection performance

Among the two most important parameters in the LFR benchmark are the topological and weights mixing coefficients. The topological mixing coefficient μ_t is the average ratio of intra-cluster neighbors divided by the number of inter-cluster neighbors, as defined in [3]. The weights mixing coefficient is defined as the average ratio of node intra-cluster strength and inter-cluster strength, as defined in [4]. We explored the effects of these paramters on the ability of PACO to retrieve the planted structure in the network. We set $\mu_t = \mu_w$, as setting μ_w greater than μ_t would introduce inconsistency in the relative number and weight of the edges, with intermodule edges carrying the largest weights.

We analyzed the performance of Newman's Modularity, Infomap and PACO on LFR networks where we varied the topological and weights mixing coefficients. In Figure S11 the performance of the three methods is comparable in terms of NMI, with a faster decay of NMI for InfoMap and Newman compared to PACO for large μ_t.

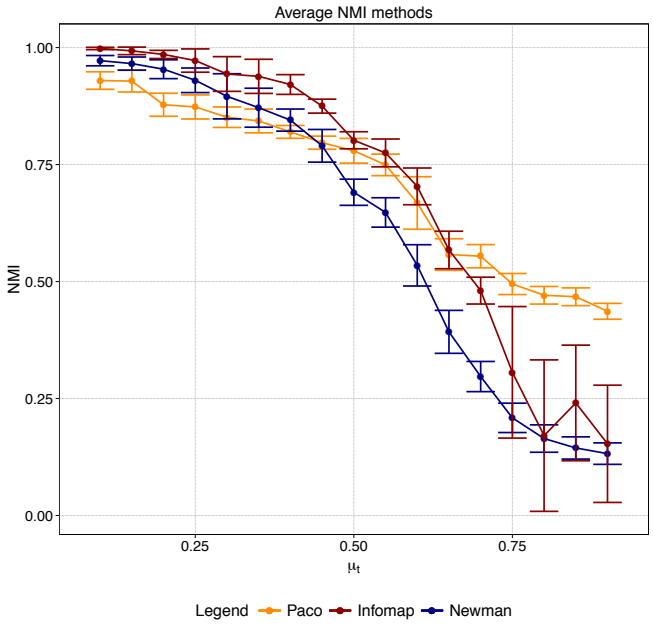

Figure S11: NMI of the retreived vs planted partition of an LFR network as a function of $\mu_t = \mu_w$ for the three community detection methods

2.5 Number of repetitions

In order to get a better idea of how fast the method converges to a maximum, we have plotted the optimal value of Asymptotical Surprise over the 10000 runs for one instance of the LFR network, and for the resting state data-set presented in the Manuscript (Figures S12,S13). From these graphs, it appears that 2000 runs may be sufficient to reach stable community detection for the experimental data-set. A near-optimum value is reached earlier for the LFR network, but it should be noted that we have taken an instance of the synthetic network with no-noise added.

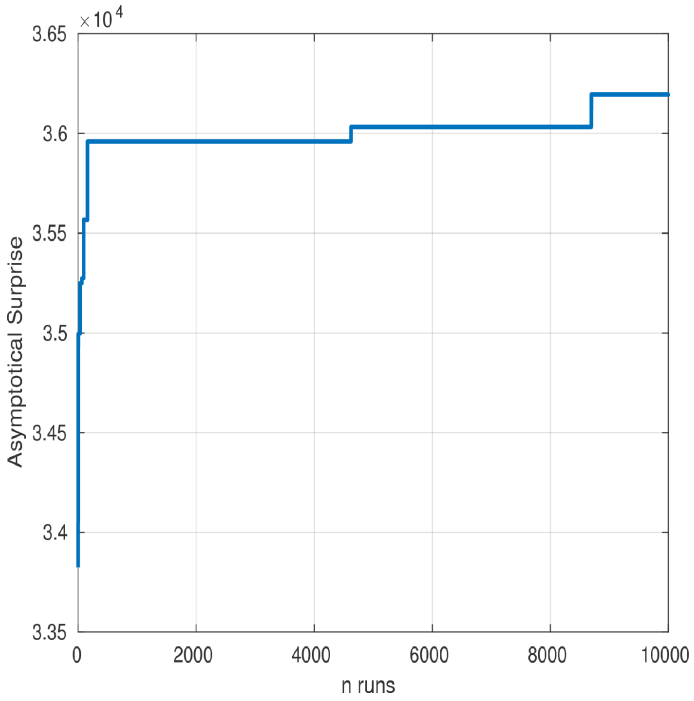

Figure S12: Maximum value of Asymptotical Surprise with respect to number of repetitions on a LFR networks as described in the main text.

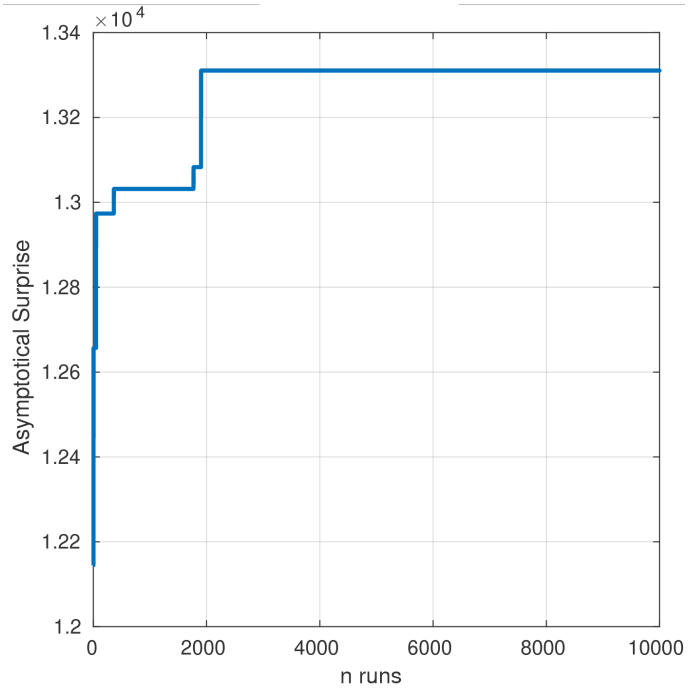

Figure S13: Maximum value of Asymptotical Surprise with respect to number of repetitions on the resting state dataset described in the main text.

2.6 Test-retest reliability

To assess the consistency of community detection methods, we have generated 91 instances of the LFR synthetic network with a SNR of 20. We have run community detection on random subgroups of 50 instances for 20 times and computed average NMI with planted partition as shown in Figure S14. We can observe that Surprise is on average similar to InfoMap and slightly better than Newman's method.

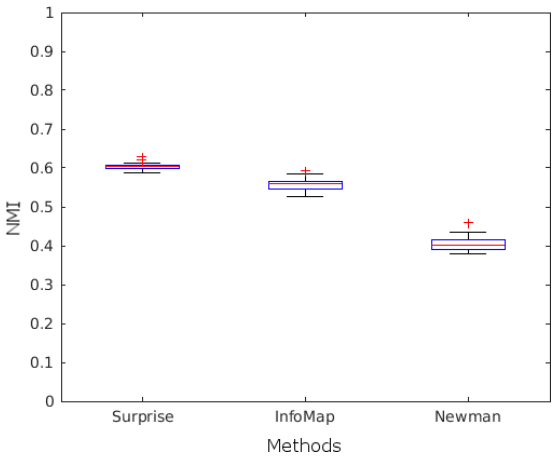

Figure S14: Test-retest reliability of community detection methods on simulated LFR networks.

2.7 Computational running time analysis

We applied PACO on a full-resolution voxelwise connectivity matrix with 51,653 nodes and almost 2 million edges. PACO took 14 minutes for a single repetition on a server with Intel Xeon E5-2643@ 3.40 Ghz CPU and 256 GB ram. We estimated that 2000 repetitions of PACO would take approximately 2.5 weeks on this server. We also tried to run PACO on a standard office PC with 16 GB memory and an Intel Core i7: it took almost 40 minutes. It's important to notice that the running time of PACO is mainly determined by the computation of Jaccard indexes in the initial step. The running time for this computation is in the order of $O(nd^2)$ where n is the number of nodes and d is the average degree of nodes. On a desktop workstation with a 2.5 GHz CPU, PACO runs in some tenths of seconds on a single repetition for a graph of around 600 nodes with a density close to 10%, typical of brain networks. A small benchmark of PACO running times on a desktop workstation is shown in Figure S15, where we found optimal Surprise partitions of a LFR network with the same parameters described in the text but increasing number of nodes.

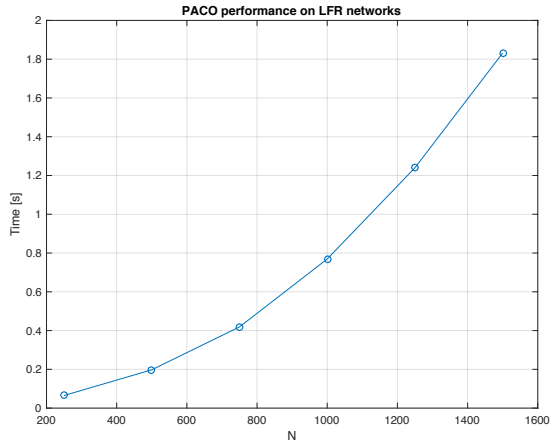

Figure S15: Running time of PACO on a LFR network with different number of nodes.

2.8 Percolation Analysis

Sparsification procedures are normally applied to remove weaker links, which are most affected by experimental noise [5], and to reduce the density of the graph, thus making it computationally more tractable. To this end, we have used percolation analysis to identify a threshold that preserves network structure and connectedness. This procedure, first introduced by Gallos [6], iteratively removes the weakest edges and computes the largest connected component. The percolation threshold corresponds to the point where the largest component starts breaking apart. A threshold just below this point is the one that maximizes the information extracted by subsequent application of community detection algorithms, and has been applied and validated in human [7] and animal [8] studies. For the sake of illustration, Figure S16 reports the size of the largest connected component of the fully-weighted functional connectivity graph as a function of threshold, showing the presence of a percolation threshold.

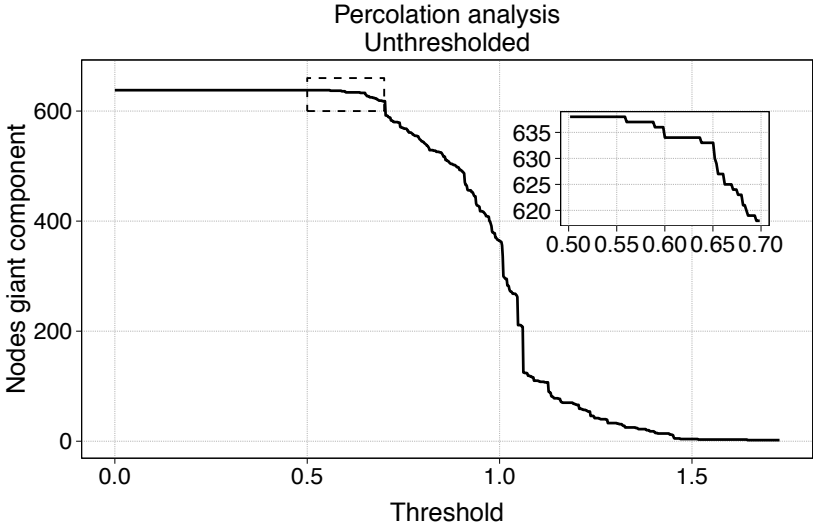

Figure S16: Percolation analysis on the fully-weighted resting state network used in the manuscript.

For LFR networks, which are endowed with a planted modular structure, we assessed directly the effects of thresholding on the ability to retrieve the ground truth modular structure of the graph. In Figure S17, we report a plot of Normalized Mutual Information, a measure of similarity between the partition obtained by Surprise and the planted structure, as a function of threshold. The thresholds resulting from the percolation analysis (for different instances of the network) fall within the grey line, and correspond to a maximum of the information extracted by the community detection algorithm. As apparent from the graph S17, retaining all edges (i.e. not applying any sparsification threshold) results in worse partitions compared to the sparsification procedure we have applied in the Manuscript.

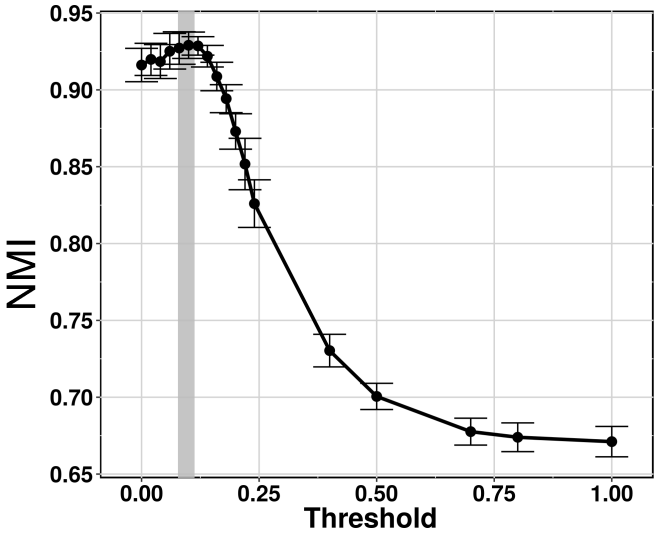

Figure S17: NMI as function of threshold on a set of LFR networks with same parameters as in the manuscript.

References

[1] Nicolini, C. & Bifone, A. Modular structure of brain functional networks: breaking the resolution limit by surprise. *Sci.Rep.* **6**, 19250 (2016).

[2] Lancichinetti, A. & Fortunato, S. Consensus clustering in complex networks. *Sci. Rep.* **2**, 336 (2012).

[3] Lancichinetti, A., Fortunato, S. & Radicchi, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**, 046110 (2008).

[4] Lancichinetti, A. & Fortunato, S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* **80**, 1–8 (2009).

[5] van den Heuvel, M. & Fornito, A. Brain networks in schizophrenia. *Neuropsychology Review* **24**, 32–48 (2014).

[6] Gallos, L. K., Makse, H. a. & Sigman, M. A small world of weak ties provides optimal global integration of self-similar modules in functional brain networks. *Proc. Natl. Acad. Sci. U. S. A.* **109**, 2825–30 (2012).

[7] Alexander-Bloch, A. F. *et al.* Disrupted modularity and local connectivity of brain functional networks in childhood-onset schizophrenia. *Front. Syst. Neurosci.* **4**, 147 (2010).

[8] Bardella, G., Bifone, A., Gabrielli, A., Gozzi, A. & Squartini, T. Hierarchical organization of functional connectivity in the mouse brain: a complex network approach. *Sci. Rep.* **6**, 32060 (2016). 1601.07574.