scientific reports

Check for updates

OPEN

Single-trajectory map equation

Tatsuro Kawamoto

Community detection, the process of identifying module structures in complex systems represented on networks, is an effective tool in various fields of science. The map equation, which is an information-theoretic framework based on the random walk on a network, is a particularly popular community detection method. Despite its outstanding performance in many applications, the inner workings of the map equation have not been thoroughly studied. Herein, we revisit the original formulation of the map equation and address the existence of its "raw form," which we refer to as the single-trajectory map equation. This raw form sheds light on many details behind the principle of the map equation that are hidden in the steady-state limit of the random walk. Most importantly, the single-trajectory map equation provides a more balanced community structure, naturally reducing the tendency of the overfitting phenomenon in the map equation.

Community detection plays a vital role in various disciplines of science dealing with network data. It is an unsupervised learning task to classify the node set in a network into groups, or modules. Each subgraph that is identified as a module has no significant internal structure, while we expect each module to be structurally distinct. Thus, community detection provides a concise explanation of the dataset by ignoring detailed structures as noises within a module while preserving a significant macroscopic organisation as a module structure[1–4]. Analogous to other machine-learning tasks, because it is often not apparent which parts of the dataset represent noise, community detection algorithms suffer from overfitting and underfitting problems[5].

The map equation[6] is a popular community detection method for networks and is formulated as a minimization problem of an information-theoretic objective function describing the average code length of the random walk. Infomap[7]—an implementation for a greedy optimisation of the map equation—has often been used to analyze real-world datasets. Furthermore, there are several extensions of the map equation itself[8–19], mainly focusing on incorporating higher-order network information. However, the map equation is prone to overfitting, particularly for sparse networks. Despite the map equation providing an optimal module structure for the description of the random walk on a network, an excessively fine module structure may be obtained. For example, as illustrated in Fig. 1 (left), many small modules are often identified in addition to a few large modules. When too many modules consisting of only a few nodes are identified, we have difficulty interpreting it as a concise explanation of the dataset.

This study revisits the map equation and considers its raw form, which we refer to as the *single-trajectory* map equation. Its objective function is the average code length of (not necessarily random) walkers with finite path lengths. The concept of the single-trajectory map equation already appears in the original paper[6] for a schematic description of the map equation. Nevertheless, this raw form has never been actively studied or utilised although it is a valuable variant with a mechanism to prune small modules and prevent the map equation from overfitting (as depicted in Fig. 1 (right)).

The emergence of small or highly unbalanced modules has been discussed in various contexts in the community detection literature. It is often considered to be the nature of real-world datasets[20–22], or a phenomenon that occurs because of the implementation details of an algorithm[21,23]. In the context of the inference problem, the emergence of small modules is interpreted as artefacts caused by overfitting. Optimization-based (or maximum likelihood-based) methods are typically prone to overfitting, whereas methods based on Bayesian formulations avoid partitioning a network or subgraph where there is no statistically significant internal structure[24,25], that is, they avoid generating small modules. The map equation also has a Bayesian counterpart[16,19]. Regardless of the underlying mechanism, the pruning of small modules is sometimes preferred in practice because it provides a more concise explanation of the network. A similar issue can be found in the regression problem in supervised learning[26]. Although the ridge regression is a principled method, many variables with extremely small coefficients are often assessed as significant. In contrast, the lasso regression prunes such variables and provides a concise description of the dataset.

The single-trajectory map equation is a variant of the map equation that achieves a partition of coarser-resolution scale. Our approach differs from that of the hierarchical map equation[8] that achieves finer-resolution partitions[27]. Although a high-resolution community detection method is useful when the network consists

Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology, Tokyo 135-0064, Japan. email: kawamoto.tatsuro@aist.go.jp

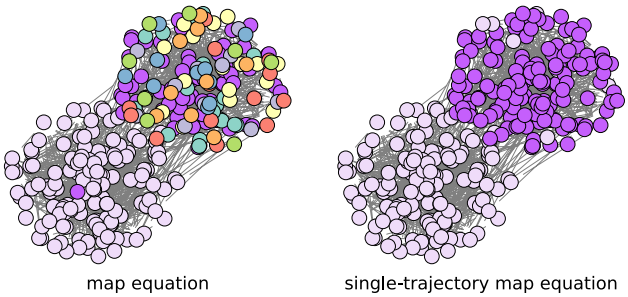

Figure 1. Community detection based on the map equation and the single-trajectory map equation applied to a synthetic network. The nodes in the same colour belong to the same module. See "Experiments" section for details.

of several small modules, a method with a coarser resolution is also needed when an algorithm suffers from overfitting. For bipartite networks[17], showed that a coarser resolution can be obtained by incorporating the bipartiteness property. A different resolution scale can also be obtained by introducing the "Markov time"[12,28], which is an external parameter of the random walk. However, as shown in the following, the framework of the map equation can intrinsically prune small modules when formulated as the single-trajectory map equation and the balanced-size modules can be identified in a principled manner.

Results

Revisiting the map equation. We proceed with the step-by-step formulation of the map equation. A prominent characteristic of the map equation is the hierarchical encoding scheme for the random walk using multiple codebooks that takes account of module structure in a network. As a specific example, let us consider the encoding of a trajectory in a network as shown in Fig. 2a. We let $\boldsymbol{\zeta} = \{\zeta_0, \ldots, \zeta_{T-1}\}$ be a trajectory of a walker, where $\zeta_t \in V$ is the tth visited node. We also consider a partition $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_N\}$ of node set V ($|V| = N$), where $\sigma_i \in \{1, \ldots, K\}$ is the module label of node $i \in V$. K is the number of modules. A trajectory of a walker is encoded using two types of codebooks: inter-module and intra-module codebooks. The inter-module codebook describes the transitions of a walker moving into another module. In contrast, each intra-module codebook describes the walker transiting between nodes within the module or exiting the module.

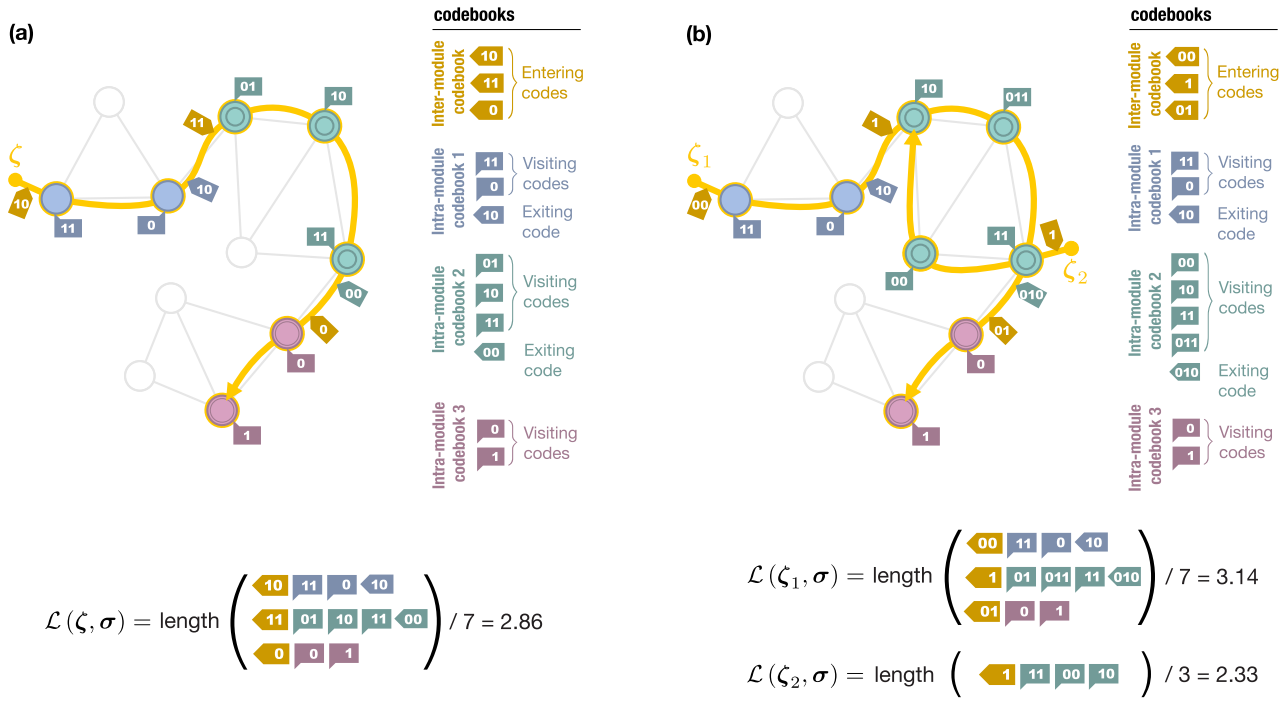

Figure 2. Trajectories (yellow solid lines) on a network and their encoding for a given node partition. Nodes in the same module have the same symbol and colour representations. The codewords in each codebook are listed on the right. Whereas there is only one trajectory in (**a**), another trajectory is added in (**b**). The average code length of each trajectory is shown at the bottom.

The actual codewords for a trajectory based on the Huffman coding[29,30] are shown in Fig. 2a. Starting with the code "10" for the module that the walker has first visited, the trajectory is described by indicating the visited nodes. Every time the walker moves to a different module, the exiting code of the previous module and the entering code of the next module are consumed.

In general, given a trajectory $\boldsymbol{\zeta}$ of length T and module assignments $\boldsymbol{\sigma}$, the average code length $\mathscr{L}(\boldsymbol{\zeta},\boldsymbol{\sigma})$ is expressed as

$$\mathscr{L}(\boldsymbol{\zeta},\boldsymbol{\sigma}) = \frac{1}{T}\left(\sum_{t=0}^{T-1}\ell_0(\zeta_t,\sigma_{\zeta_t}) + \sum_{\substack{t=0\\(\sigma_{\zeta_t}\neq\sigma_{\zeta_{t+1}})}}^{T-2}\left(\ell_0(\curvearrowright,\sigma_{\zeta_t}) + \ell_1(\sigma_{\zeta_{t+1}})\right) + \ell_1(\sigma_{\zeta_0})\right). \tag{1}$$

Here, we denote $\ell_0(i,\sigma)$ as the length of the code in an intra-module codebook, indicating that a walker visits node $i \in V$ in module σ; $\ell_0(\curvearrowright,\sigma)$ as the length of the code in an intra-module codebook, indicating that a walker exits module σ; and $\ell_1(\sigma)$ as the length of the code in an inter-module codebook, indicating that a walker enters module σ. In Eq. (1), the first summation represents the code length for visited nodes and the second summation represents the code length for transitions between modules. The last term is the code length for the module at the starting point, with a negligible contribution when $T \gg 1$. It is important that the codebooks are coupled; that is, because an exiting code from a module belongs to an intra-module codebook, transitions between modules affect the encoding of transitions within each module.

The principle of the map equation framework is that the compression of the average code length through the hierarchical coding reveals a module structure as an optimal partition $\boldsymbol{\sigma}$. Readers might believe that the introduction of codewords for transitions between modules simply makes the code length longer. However, such a hierarchical encoding scheme can compress the average code length because it allows us to assign shorter codewords for visited nodes; for example, although the code "0" is assigned to two different nodes in Fig. 2a, they are distinguishable because they belong to different modules. Therefore, when a trajectory rarely consumes the codewords for transitions between modules, the average code length can be compressed more efficiently.

Equation (1) can also be expressed using visiting frequencies as follows:

$$\mathscr{L}(\boldsymbol{\zeta},\boldsymbol{\sigma}) = \left(\frac{1}{T} + \sum_{\tilde{\sigma}',\tilde{\sigma}}\hat{\mathsf{p}}_{\tilde{\sigma}'\tilde{\sigma}}\right)\mathscr{H}_1 + \sum_{\sigma}\left(\sum_{j\in\sigma}\hat{\mathsf{p}}_j + \sum_{\tilde{\sigma}'}\hat{\mathsf{p}}_{\tilde{\sigma}'\sigma}\right)\mathscr{H}_0^{\sigma}, \tag{2}$$

$$\begin{cases} \mathscr{H}_1 = \sum_{\sigma'}\frac{\sum_{\sigma}\hat{\mathsf{p}}_{\sigma'\sigma}}{1/T+\sum_{\tilde{\sigma}',\tilde{\sigma}}\hat{\mathsf{p}}_{\tilde{\sigma}'\tilde{\sigma}}}\ell_1(\sigma') + \frac{1/T}{1/T+\sum_{\tilde{\sigma}',\tilde{\sigma}}\hat{\mathsf{p}}_{\tilde{\sigma}'\tilde{\sigma}}}\ell_1(\sigma_{\zeta_0}) \\ \mathscr{H}_0^{\sigma} = \sum_{i\in\sigma}\frac{\hat{\mathsf{p}}_i}{\sum_{j\in\sigma}\hat{\mathsf{p}}_j+\sum_{\tilde{\sigma}'}\hat{\mathsf{p}}_{\tilde{\sigma}'\sigma}}\ell_0(i,\sigma) + \frac{\sum_{\sigma'}\hat{\mathsf{p}}_{\sigma'\sigma}}{\sum_{j\in\sigma}\hat{\mathsf{p}}_j+\sum_{\tilde{\sigma}'}\hat{\mathsf{p}}_{\tilde{\sigma}'\sigma}}\ell_0(\curvearrowright,\sigma) \end{cases}, \tag{3}$$

where $\sum_{i\in\sigma}$ represents the sum over the node set in module σ. $\hat{\mathsf{p}}_i$ is the visiting frequency of node $i \in V$ and $\hat{\mathsf{p}}_{\sigma'\sigma}$ is the joint transition frequency from module σ to module σ', i.e.,

$$\hat{\mathsf{p}}_i = \frac{1}{T}\sum_{t=0}^{T-1}\delta_{i,\zeta_t}, \tag{4}$$

$$\hat{\mathsf{p}}_{\sigma'\sigma} = \begin{cases} \frac{1}{T}\sum_{t=0}^{T-2}\delta_{\sigma',\sigma_{\zeta_{t+1}}}\delta_{\sigma,\sigma_{\zeta_t}} & (\text{for } \sigma \neq \sigma') \\ 0 & (\text{for } \sigma = \sigma') \end{cases}, \tag{5}$$

where δ_{ab} represents the Kronecker delta. $\mathscr{H}_0^{\sigma}$ and $\mathscr{H}_1$ are conditional average code lengths within the intra- and inter-module codebooks, respectively.

Recall that the random walk is a stochastic variable; there is no such thing as a single (finite-length) trajectory representing the random walk. Therefore, instead of a specific trajectory, we consider the *expected* average code length $\mathbb{E}_{\boldsymbol{Z}}\left[\mathscr{L}(\boldsymbol{Z},\boldsymbol{\sigma})\right]$ in the map equation, where $\boldsymbol{Z}$ is the stochastic variable representing the random walk; in other words, $\mathbb{E}_{\boldsymbol{Z}}[\cdots]$ is the ensemble average over all possible trajectories (say, from all possible starting points). We assume that the trajectory length T is sufficiently large that the random walk is in a steady state. When the network is strongly connected, the empirical frequencies are converted to the corresponding steady-state probabilities. $\sum_{\sigma}\hat{\mathsf{p}}_{\sigma'\sigma}$ is converted to the entering probability into module σ' denoted by $q_{\sigma'\curvearrowright}$; $\sum_{\sigma'}\hat{\mathsf{p}}_{\sigma'\sigma}$ to the exiting probability from module σ denoted by $q_{\sigma\curvearrowright}$; and $\hat{\mathsf{p}}_i$ to the visiting probability of node i denoted by q_i. The conditional average code lengths $\mathscr{H}_0^{\sigma}$ and $\mathscr{H}_1$ are also converted to the expectations. According to Shannon's source coding theorem[30,31], these expectations are respectively bounded by the Shannon entropies,

$$H_1(\{q_{\sigma\curvearrowright}\}) = -\sum_{\sigma=1}^{K}\frac{q_{\sigma\curvearrowright}}{q_{\curvearrowright}}\log\frac{q_{\sigma\curvearrowright}}{q_{\curvearrowright}}, \tag{6}$$

$$H_0^{\sigma}\left(q_{\sigma\curvearrowright},\{q_i\}_{i\in\sigma}\right) = -\frac{q_{\sigma\curvearrowright}}{p_{\circlearrowright}^{\sigma}}\log\frac{q_{\sigma\curvearrowright}}{p_{\circlearrowright}^{\sigma}} - \sum_{i\in\sigma}\frac{q_i}{p_{\circlearrowright}^{\sigma}}\log\frac{q_i}{p_{\circlearrowright}^{\sigma}}, \tag{7}$$

where $q_{\curvearrowleft} = \sum_{\sigma=1}^{K} q_{\sigma\curvearrowleft}$, $p^{\sigma}_{\circlearrowleft} = q_{\sigma\curvearrowright} + \sum_{i\in\sigma} q_i$, and log is the logarithm with base 2. Then, the expected average code length of the random walk is bounded from below as follows:

$$L(\boldsymbol{\sigma}) = q_{\curvearrowleft} H_1\left(\{q_{\sigma\curvearrowleft}\}\right) + \sum_{\sigma} p^{\sigma}_{\circlearrowleft} H_0^{\sigma}\left(q_{\sigma\curvearrowright}, \{q_i\}_{i\in\sigma}\right) \leq \mathbb{E}_{\boldsymbol{Z}}\left[\mathscr{L}\left(\boldsymbol{Z},\boldsymbol{\sigma}\right)\right]. \tag{8}$$

Note here that the contribution from the starting points of the random walk is excluded. This lower bound asymptotically coincides with the expected average code length itself as $T \to \infty$. This is the objective function of the map equation and the node partition $\boldsymbol{\sigma}$ is optimised so that $L(\boldsymbol{\sigma})$ is minimised.

The assumption that the network is strongly connected plays a vital role in the aforementioned derivation. If this is not the case, the trajectory length T cannot be sufficiently large. Then, the contribution from the starting points of the random walk may not be negligible in $\mathbb{E}_{\boldsymbol{Z}}\left[\mathscr{L}\left(\boldsymbol{Z},\boldsymbol{\sigma}\right)\right]$. Therefore, we can say that the map equation evaluates the code length of the "flow." It is a stochastic variable representing the ensemble of transitions, and it has no information about the starting points of the random walk by definition (as discussed below, this distinction becomes more prominent when we consider the `- -flow-model rawdir` option in Infomap). The only input for the map equation is a network because the connectivity of nodes fully characterises the flow. By the introduction of so-called teleportation[32] to the random walk that moves the walker to another node randomly with a certain probability, we can always let the trajectory length T be infinitely large and make the flow ergodic[6]. Therefore, the map equation is not essentially limited to strongly-connected networks.

Single-trajectory map equation. The average code length $\mathscr{L}\left(\boldsymbol{\zeta},\boldsymbol{\sigma}\right)$ of a trajectory is the raw form of the objective function in the map equation. When we have multiple trajectories $\{\boldsymbol{\zeta}_a\} := \{\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_M\}$ on a common node set, analogous to the expected average code length $\mathbb{E}_{\boldsymbol{Z}}\left[\mathscr{L}\left(\boldsymbol{Z},\boldsymbol{\sigma}\right)\right]$, we consider the following mean average code length:

$$\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right) = \frac{1}{M}\sum_{a=1}^{M} \mathscr{L}\left(\boldsymbol{\zeta}_a,\boldsymbol{\sigma}\right). \tag{9}$$

Each trajectory may have different lengths. Similar to the $L(\boldsymbol{\sigma})$, this mean average code length can be used as a minimization function to determine the optimal module assignments of nodes. We refer to such an optimisation method as the single-trajectory map equation. Note that the trajectories $\{\boldsymbol{\zeta}_a\}$ are provided as inputs in Eq. (9); unlike the map equation, there is no need to assume that they are generated (or simulated) from random walks, although one can consider simulated walks as trajectories.

The average code lengths in the summation of Eq. (9) are not independent because they share codebooks. To illustrate this, let us consider two trajectories as shown in Fig. 2b. Although the trajectory $\boldsymbol{\zeta}_1$ is identical to $\boldsymbol{\zeta}$ in Fig. 2a, the codes describing them are different because we must assign codewords for the nodes that $\boldsymbol{\zeta}_1$ does not go through due to the existence of trajectory $\boldsymbol{\zeta}_2$. In contrast, the nodes where no trajectories go through do not contribute to the average code lengths, reflecting the fact that the trajectories of finite lengths are considered. Those nodes should not have any module labels because there is no information based on the trajectories.

As we have seen, $L(\boldsymbol{\sigma})$ and $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ are conceptually different. In the map equation, $L(\boldsymbol{\sigma})$ is the expected average code length for the flow that is completely specified by the transition probabilities. We can also modify the transitions using teleportation to make the random walk ergodic. By contrast, $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ does not have such a stochasticity. It is the mean of the actual average code lengths, where each element corresponds to a single trajectory. Furthermore, $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ depends explicitly on the coding scheme applied, e.g., the Huffman coding, Shannon–Fano coding[30], etc. Quantitatively, the contribution from the last term in Eq. (1) mainly makes the minimization of $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ distinct from that of $L(\boldsymbol{\sigma})$. The codeword for the module that is required to specify the starting point of a trajectory makes the coding using multiple codebooks less efficient. Recall that an efficient compression is achieved when the inter-module codebook is not frequently used. This implies that the introduction of module labels is more costly in $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ and the single-trajectory map equation avoids generating many small modules.

The single-trajectory map equation searches for the node partition $\boldsymbol{\sigma}$ that achieves the optimal compression for the description of trajectories under a certain coding scheme. The optimality of the coding scheme itself is not required for the effectiveness of the method. Therefore, we can use different types of coding for the intra-module and inter-module codebooks. For example, we can introduce a heterogeneous coding where the code lengths are multiplied by a constant factor $\lambda > 0$ for the codewords in the inter-module codebook. That is, given a trajectory $\boldsymbol{\zeta}$ and a partition $\boldsymbol{\sigma}$, Eq. (1) is modified to

$$\mathscr{L}_{\lambda}(\boldsymbol{\zeta},\boldsymbol{\sigma}) = \frac{1}{T}\left(\sum_{t=0}^{T-1} \ell_0\left(\zeta_t, \sigma_{\zeta_t}\right) + \sum_{\substack{t=0 \\ (\sigma_{\zeta_t} \neq \sigma_{\zeta_{t+1}})}}^{T-2} \left(\ell_0\left(\curvearrowright, \sigma_{\zeta_t}\right) + \lambda \ell_1\left(\sigma_{\zeta_{t+1}}\right)\right) + \lambda \ell_1\left(\sigma_{\zeta_0}\right)\right). \tag{10}$$

Here, λ is a hyperparameter that penalises the emergence of modules when such modules are relatively inefficient for the compression of the code length.

We can also derive a lower bound for the actual code length using Shannon's source coding theorem, similar to how $L(\boldsymbol{\sigma})$ was such an estimate for the random walk in the steady-state limit. To this end, we consider the average code length of the concatenated code, $\sum_{a=1}^{M} T_a \mathscr{L}\left(\boldsymbol{\zeta}_a,\boldsymbol{\sigma}\right) / \sum_{a=1}^{M} T_a$, where T_a is the length of the ath trajectory; equivalently $\overline{\mathscr{L}}\left(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}\right)$ when all trajectories have the same length. We regard the empirical frequencies $\hat{\mathrm{p}}_i$ and

$\hat{\mathrm{p}}_{\sigma'\sigma}$ as the true probabilities for the stochastic variables indicating the codewords and the concatenated code as the expected code length. Then, the conditional average code lengths in Eq. (3) are bounded from below by the Shannon entropies[30] with the empirical frequencies. Therefore, the average code length of the concatenated code is bounded as follows:

$$\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)=\hat{\mathrm{q}}_{\curvearrowleft}H_1\left(\{\hat{\mathrm{q}}_{\sigma\curvearrowleft}\}\right)+\sum_{\sigma}\hat{\mathrm{p}}^{\sigma}_{\circlearrowleft}H^{\sigma}_0\left(\hat{\mathrm{q}}_{\sigma\curvearrowright},\{\hat{\mathrm{q}}_i\}_{i\in\sigma}\right)\le\frac{1}{\sum_{a=1}^{M}T_a}\sum_{a=1}^{M}T_a\mathscr{L}\left(\boldsymbol{\zeta}_a,\boldsymbol{\sigma}\right),\tag{11}$$

where

$$\begin{aligned}\hat{\mathrm{q}}_i&=\frac{1}{\sum_{a=1}^{M}T_a}\sum_{a=1}^{M}\sum_{t=0}^{T_a-1}\delta_{i,\zeta_{at}},\\ \hat{\mathrm{q}}_{\sigma'\sigma}&=\begin{cases}\dfrac{1}{\sum_{a=1}^{M}T_a}\displaystyle\sum_{a=1}^{M}\sum_{t=0}^{T_a-2}\delta_{\sigma',\sigma_{\zeta_{at+1}}}\delta_{\sigma,\sigma_{\zeta_{at}}} & (\sigma\neq\sigma')\\ 0 & (\sigma=\sigma')\end{cases},\\ \hat{\mathrm{q}}_{\sigma'\curvearrowleft}&=\frac{\sum_{a=1}^{M}\delta_{\sigma',\sigma_{\zeta_{a0}}}}{\sum_{a=1}^{M}T_a}+\sum_{\sigma}\hat{\mathrm{q}}_{\sigma'\sigma},\hat{\mathrm{q}}_{\sigma\curvearrowright}=\sum_{\sigma'}\hat{\mathrm{q}}_{\sigma'\sigma},\\ \hat{\mathrm{q}}_{\curvearrowleft}&=\sum_{\sigma}\hat{\mathrm{q}}_{\sigma\curvearrowleft},\hat{\mathrm{p}}^{\sigma}_{\circlearrowleft}=\hat{\mathrm{q}}_{\sigma\curvearrowright}+\sum_{i\in\sigma}\hat{\mathrm{q}}_i.\end{aligned}\tag{12}$$

We regard $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ as an alternative objective function for the single-trajectory map equation. $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ is independent of the coding scheme and its minimization is computationally more efficient than that of $\overline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ because we do not need to construct the codebooks explicitly. Note that $\hat{\mathrm{q}}_{\sigma\curvearrowleft}$ and $\hat{\mathrm{q}}_{\sigma\curvearrowright}$ may not coincide in Eq. (12), whereas $q_{\sigma\curvearrowleft}=q_{\sigma\curvearrowright}$ for any module in $L(\boldsymbol{\sigma})$ owing to the detailed balance condition of the random walk in the steady state. Analogous to $\mathscr{L}_{\lambda}(\boldsymbol{\zeta},\boldsymbol{\sigma})$ in Eq. (10), we can also consider a heterogeneous coding in $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$, i.e.,

$$\underline{\mathscr{L}}_{\lambda}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)=\lambda\,\hat{\mathrm{q}}_{\curvearrowleft}H_1\left(\{\hat{\mathrm{q}}_{\sigma\curvearrowleft}\}\right)+\sum_{\sigma}\hat{\mathrm{p}}^{\sigma}_{\circlearrowleft}H^{\sigma}_0\left(\{\hat{\mathrm{q}}_{\sigma\curvearrowright}\},\{\hat{\mathrm{q}}_i\}_{i\in\sigma}\right).\tag{13}$$

Interestingly, the method using $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ is also related to a variant of the map equation that is implemented in Infomap as an option named `--flow-model rawdir`. In this variant of the map equation, we consider the *flow based on the set of transition probabilities induced by the edges* (i.e., not the random walk on the network). The corresponding objective function is in fact equivalent to $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ where we ignore the codewords for the initial module *and* the initial node in each trajectory (consequently, the total length of trajectories $\sum_{a=1}^{M}T_a$ is also modified to $\sum_{a=1}^{M}T_a-M$). A summary table of the average code lengths is shown in Supplementary Table 1 (Section S1).

Before moving on, let us compare how the minimizations of $L(\boldsymbol{\sigma})$, $\overline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$, and $\underline{\mathscr{L}}\left(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}\right)$ differ using a simple example. We consider a trajectory where a walker visits each node exactly once on a path. Figure 3 shows

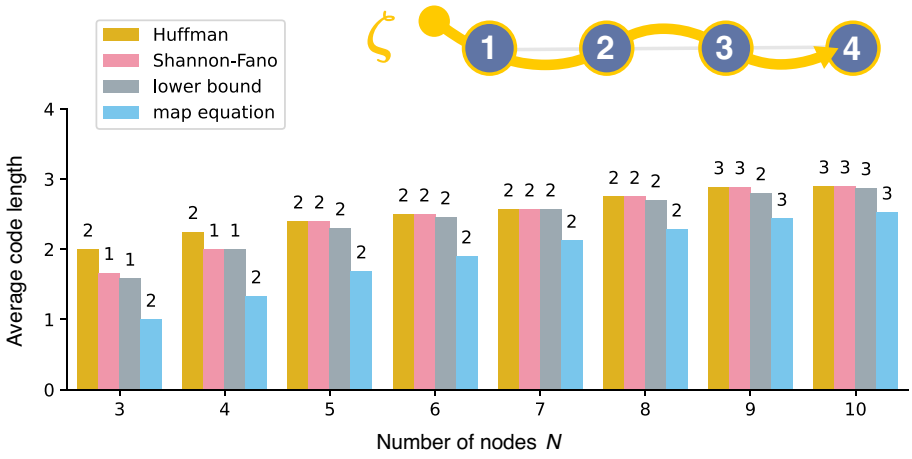

Figure 3. Average code lengths for a trajectory on a path (illustrated at the top) obtained by minimising $\mathscr{L}(\boldsymbol{\sigma};\boldsymbol{\zeta})$ ("Huffman": Huffman coding, "Shannon–Fano": Shannon–Fano coding) and $\underline{\mathscr{L}}(\boldsymbol{\sigma};\boldsymbol{\zeta})$ ("lower bound"). The expected average code length $L(\boldsymbol{\sigma})$ ("map equation") based on the set of transition probabilities induced by the edges, i.e., the `--flow-model rawdir` option, is also shown. The number of detected modules in each method is indicated at the top of each bar.

the results obtained through the exact minimization of the objective functions. It quantifies how the average code lengths approach a common value as N increases, because the contribution from the starting point of the trajectory becomes negligible. $\mathscr{L}(\boldsymbol{\sigma};\boldsymbol{\zeta})$ and $\overline{\mathscr{L}}(\boldsymbol{\sigma};\boldsymbol{\zeta})$ quickly approach to each other, whereas $L(\boldsymbol{\sigma})$ converges relatively slowly, implying that the contribution from the codeword of the initial module can be considerable. We also confirmed that the single-trajectory map equation indeed tends to identify a smaller number of modules, and the resulting partitions can vary depending on the coding scheme applied.

The exact minimization of the (expected) average code length is not computationally feasible unless a dataset is extremely small, and thus, we must rely on approximate heuristics in practice. The greedy heuristic implemented in Infomap is commonly used for the map equation. Therefore, we implemented the optimisation for the single-trajectory map equation as a wrapper of Infomap. That is, we first run Infomap as the initial state of the node partition, and then, reduce overfitting by pruning small modules based on $\overline{\mathscr{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ or $\mathscr{L}(\boldsymbol{\sigma};\boldsymbol{\zeta})$ as a fine-tuning process; our fine-tuning algorithm is also a greedy heuristic. In the following, we refer to this algorithm as Infomap+, and the implementation code is publicly available[33]. Further details of the algorithm are described in Methods section.

Experiments. This section demonstrates that the single-trajectory map equation prevents overfitting using datasets represented as networks and a real-world dataset as a set of trajectories. A network is a special case of trajectory datasets because each directed edge can be regarded as a trajectory with length $T = 2$. We treat each edge as a pair of directed edges in both directions for an undirected network. All networks considered are weakly connected.

For the network datasets, we can also consider simulated walks on the underlying network as the input trajectories. In this setting, we would need to specify the type of simulated walks and choose the values of T and M as hyperparameters. Herein, however, we do not consider the simulated walks and treat the edges set directly as the set of trajectories.

Network datasets. We first consider synthetic networks that are generated by the stochastic block model (SBM)[25,37–39], which is a random graph model having a planted (pre-assigned) module structure. This is a canonical model that is used for analyses in community detection. We particularly consider the so-called symmetric SBM that has two equally-sized planted modules. Each pair of nodes in the same planted module is connected with probability p_{in} and each pair of nodes in different planted modules is connected with probability p_{out}. The symmetric SBM is commonly parameterized by the average degree c and the fuzziness of module structure $\epsilon = p_{\text{out}}/p_{\text{in}}$ instead of p_{in} and p_{out}. The detection of planted modules is easier when ϵ is small because the module structure is clearer. Even when $\epsilon < 1$, there exists a critical value of ϵ above which it becomes impossible to identify the planted module structure better than by chance; this is known as the detectability limit[24,34–36,39,40] (in $N \to \infty$). For these networks, the single-trajectory map equation cannot be the best method, as the Bayesian inference methods based on the SBM can avoid overfitting at all.

Figure 4 shows the results of community detection based on the map equation and the single-trajectory map equation applied to the SBM. Each point represents the relative module size, which is defined as $\sum_{i=1}^{N}\delta_{\sigma_i,\sigma}/N$ for module σ. The results based on modularity maximization (the Louvain[41] and Leiden[42] algorithms) are also shown for comparison. The `- -two-level` option in Infomap indicates that it is the method introduced in the original paper for the map equation.

The Infomap (incorrectly) identifies several small modules even when the module structure is relatively clear, whereas Infomap+ prunes such small modules and identifies the equally-sized modules. The network plots in Fig. 1 are the results of the same experiment but with the SBM parameters $N = 300, c = 8$, and $\epsilon = 0.1$. Although the modularity-based algorithms also identify small modules, Infomap is more prone to overfitting in the region where ϵ is small. This phenomenon can be described by the map equation having a finer resolution limit[27] compared with that of the modularity[43], i.e., the map equation can identify smaller modules. Note, however, that the analysis of the resolution limit is based on an extreme-case example that has a well-defined module structure; it does not describe the whole behaviour in Fig. 4. In the region of ϵ above the detectability limit ($\epsilon \approx 0.15$[40]), the modularity-based algorithms subdivide the planted modules into a number of smaller-sized modules. This is problematic because a practitioner can hardly realise when the resulting partition is due to overfitting. In contrast, most of the map equation-based algorithms do not partition a network in that region, implying that they avoid overfitting.

Although we showed the relative module sizes obtained by the algorithms, readers might wonder whether the identified modules are actually consistent with the planted ones. In Supplementary Fig. 1 (Section S2), we confirmed that the inferred and planted module structures are indeed highly consistent when the number of modules is correctly estimated. Note also that community detection algorithms generally suffer from overfitting and underfitting more severely when the average degree c is smaller. Therefore, all the methods considered here are expected to perform less accurately when c is extremely small.

The experiments here can be conducted for larger networks. In that case, however, some of the plots in Fig. 4 would be unnecessarily difficult to read because we would have many more points due to ovefitting. Moreover, the comparison with the previously known result on the detectability limit may be difficult for larger networks, because it is observed in[40] that algorithmic detectability limits of greedy algorithms can be size-dependent.

We then apply the algorithms for the single-trajectory map equation to real-world networks. Figure 5 shows the relative module sizes obtained using Infomap and Infomap+. It shows that small modules are pruned, yet larger modules remain identified in most of the cases with Infomap+. Although all variants of Infomap+ often provide similar partitions, empirically, the Huffman coding method finds a good balance of module sizes in

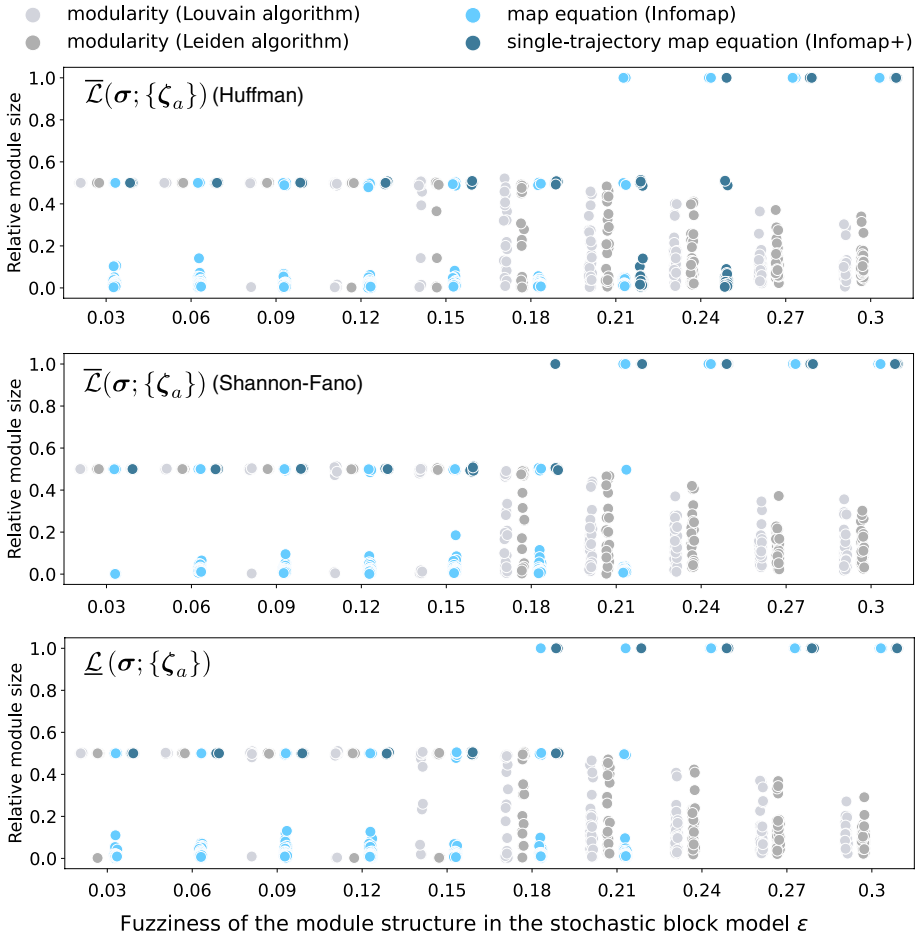

Figure 4. Performance of modularity maximization methods (the Louvain and Leiden algorithms), Infomap (`- -two-level`), and algorithms for the single-trajectory map equation based on $\overline{\mathscr{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ and $\underline{\mathscr{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ ("Infomap+") on the symmetric SBM ($N = 1,000, c = 12$). We generated five instances of the SBM for various fuzziness of the module structure ϵ and plotted the distribution of the resulting relative module sizes. Herein, we set $\lambda = 1$. The performances of all algorithms change around the algorithmic detectability limit $\epsilon \approx 0.15$, which is distinct from the information-theoretic detectability limit located at $\epsilon = (\sqrt{c}-1)/(\sqrt{c}+1) \simeq 0.55$[34–36].

real-world networks. The datasets considered here are often analyzed in the literature on community detection. For example, readers can compare the results here with those of Bayesian inference methods reported in[5,44–47].

In Fig. 5, the value of the hyperparameter λ, which acts as a resolution parameter, is adjusted for each network so that the size of the smallest module is not less than $\min\{3, N/100\}$ (this adjustment can be performed automatically). The selected values of λ and the details of experimental settings and datasets are provided in Supplementary Table 2 (Section S3). We also examined the λ-dependency in Fig. 6, and we found that the number of modules varies within $1 \leq \lambda < 2$ in many datasets; in the Method section, we show that $\lambda = 2$ is a practical upper bound according to the resolution limit. Note that the threshold $\min\{3, N/100\}$ is only a reference to determine a reasonable value of λ; when Infomap+ excessively prunes modules, one can directly tune λ to resolve the underfitting problem.

Bike-sharing dataset: application to a set of trajectories. Finally, we compare the methods using a dataset of a bike-sharing service in London[48,49], which is a dataset consisting of trajectories (sequences of bike stations visited) that individual bikes have travelled in a day (see Supplementary Information (Section S4) for the details of the dataset); thus, T_a is the number of stations that bike a has visited. Figure 7a illustrates trajectories of three bikes in the dataset. Community detection of the trajectories identifies the area within which a bike is often used. Figure 7b shows the partition obtained by minimising $L(\boldsymbol{\sigma})$ using Infomap; here, we constructed a network by decomposing each trajectory into a set of edges between successive pairs of stations. As a result, we obtained eight modules; in addition to four large modules, several modules consisting of only a few stations are also identified. In Fig. 7c which shows the partitions obtained by minimising $\overline{\mathscr{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$, we no longer observe the small modules. Although Fig. 7c is of the Huffman coding method ($\lambda = 1$), we obtain the same partition with the Shannon–Fano coding method ($\lambda = 1$) and with the method minimising $\underline{\mathscr{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ ($\lambda = 1.8$).

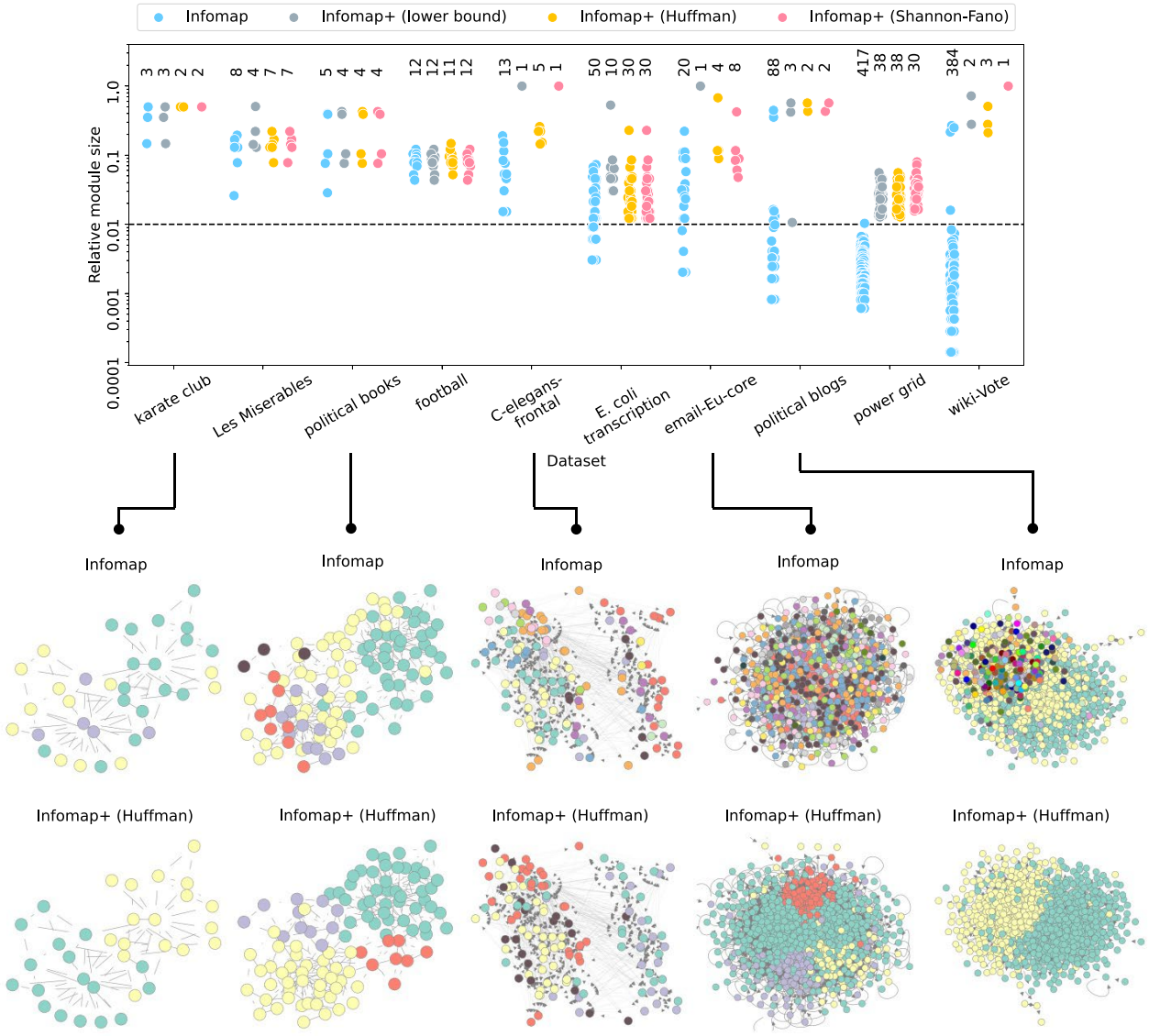

Figure 5. Relative module sizes obtained by Infomap and Infomap+ for real-world networks. The number of identified modules is depicted at the top of each result. The dashed line represents 0.01.

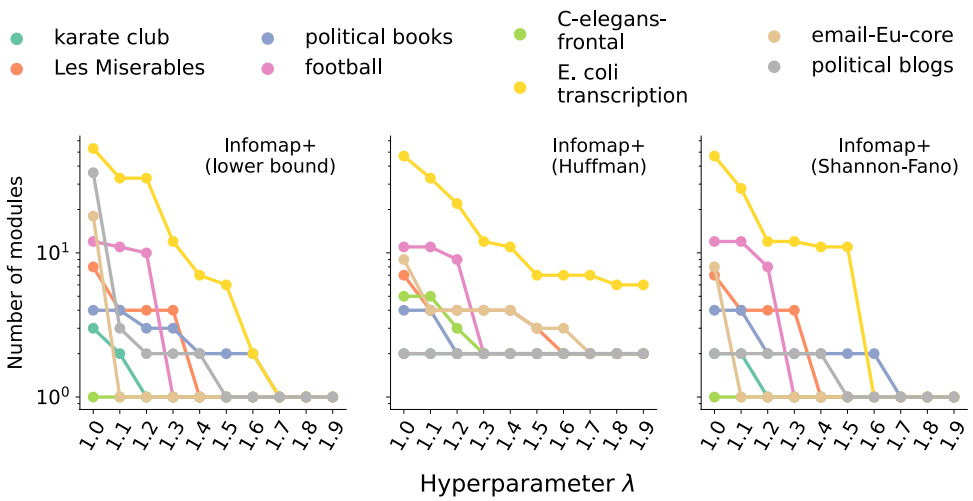

Figure 6. Number of modules identified for each value of the hyperparameter λ in Infomap+.

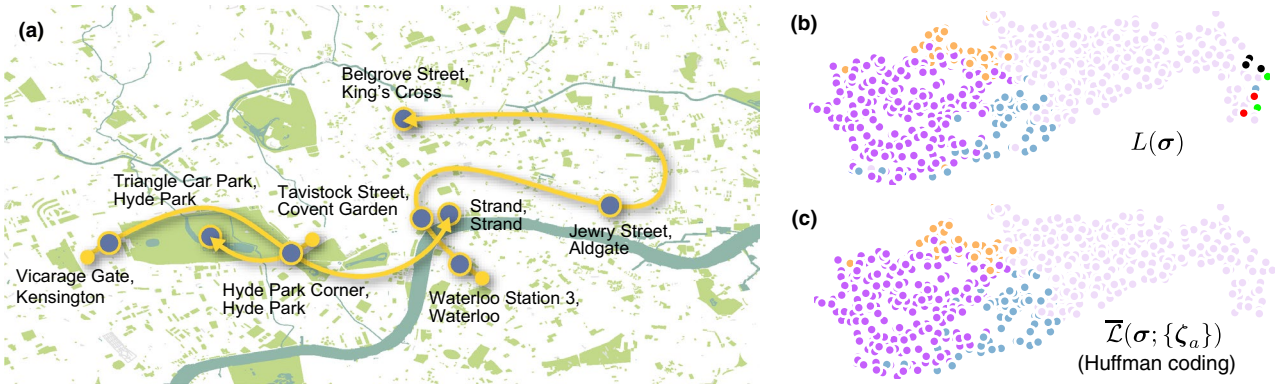

Figure 7. Community detection of the bike-sharing dataset. (**a**) Three trajectories in the dataset, where each point (node) represents the location of a bike station. The partitions of the stations are obtained by minimising (**b**) $L(\boldsymbol{\sigma})$ (detected 8 modules) and (**c**) $\overline{\mathcal{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ based on the Huffman coding (detected 4 modules). The stations in the same module are indicated in the same colour.

Discussion

This study revisited the formulation of the map equation and shed light on many details hidden in its principle. We addressed the fact that the encoding of trajectories is qualitatively distinct from the encoding of the flow on a network and proposed the single-trajectory map equation. Importantly, the proposed method can prune small modules and prevent overfitting.

The single-trajectory map equation provides a more balanced community structure compared with the map equation. Whereas balanced partitions may not always be desirable, it is often beneficial because we can prune spurious modules due to overfitting as demonstrated in Fig. 1. Furthermore, the analysis in the Method section implies that the single-trajectory map equation is not prone to underfitting compared with the map equation because their resolution limits are almost the same when $\lambda = 1$.

Readers might wonder if the present approach is distinct from other variants of the map equation, such as the one with the Markov-time parameter[12,28] and the Bayesian formulation of the map equation[16,19] which is an improved teleportation method[50]. To clarify this point, we also conducted experiments analogous to those described in Experiments section using these methods in Supplementary Information (Section S5). In some cases, these methods also exhibit similar partitions as in Figs. 4 and 5. However, they are apparently not particularly suitable for pruning small modules because these methods are more sensitive/insensitive to the choices of the hyperparameters that we need to search for the optimal values in a finer scale/wider range, while balanced partitions are often obtained without tuning the hyperparameter λ in the single-trajectory map equation. We also emphasise that the single-trajectory map equation is not a generalisation of the map equation, but its raw form, and overfitting is avoided using the principle of the map equation itself.

The bootstrapping method[51] is another approach for avoiding overfitting. However, this approach is computationally expensive[19] and a comparison with the present approach is not very clear because the output is a population of partitions. A more detailed study of the qualitative and quantitative relationships between the single-trajectory map equation and other variants of the map equation is left for future work. Furthermore, because the single-trajectory map equation is a trajectory-based approach, the relationship between the memory-network extension[10,13] of the map equation is another potential research direction because both take a set of trajectories as the input.

The time complexity of the optimisation algorithm is a major issue in the single-trajectory map equation. Whereas the lower bound $\mathcal{L}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ can be optimised as efficiently as Infomap, explicit construction of the codebooks is required for the actual average code length $\overline{\mathcal{L}}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$. Although our implementations of Infomap+ run within a reasonable amount of time for fairly large datasets as demonstrated in Supplementary Information Fig. 6, an improved implementation is also left for future work.

Methods

Optimisation algorithm. Herein, we explain the implementation details of the greedy heuristic. A typical greedy heuristic for community detection, including Infomap, iteratively merges two or more modules that improve the value of an objective function[41,42,52] and equally-sized modules are often preferentially merged[23]. Such an update rule does not effectively compress the average code length at the stage of fine-tuning. This is not surprising because the initial partition is located at a local or global minimum of the objective function in the map equation, which may also be a local minimum in the single-trajectory map equation. Moreover, there is no reason that equally-sized modules should be preferentially merged. Although we typically have a few large and many small modules as the initial partition, it is unlikely that merging those small modules provides better compression of the average code length. Therefore, instead, we iteratively merge the smallest module and its most tightly-connected module regardless of the resulting value of the average code length until only one module is left; among the partitions that form with this merging process, we accept the partition that achieves the minimum average code length. Given an initial partition, this algorithm is deterministic. Although this algorithm

is straightforward and the resulting partition may not be the global optimum, an improved compression of the average code length can be achieved by pruning small modules without being trapped into the local minima of the objective function.

When we use the lower bound $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ as the objective function, the greedy update can be performed as done in Infomap. The expanded form of $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ is

$$\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}) = \lambda\hat{\mathsf{q}}_{\curvearrowleft}\log\hat{\mathsf{q}}_{\curvearrowleft} - \lambda\sum_{\sigma}\hat{\mathsf{q}}_{\sigma\curvearrowleft}\log\hat{\mathsf{q}}_{\sigma\curvearrowleft} + \sum_{\sigma}\hat{\mathsf{p}}^{\sigma}_{\circlearrowleft}\log\hat{\mathsf{p}}^{\sigma}_{\circlearrowleft} - \sum_{\sigma}\hat{\mathsf{q}}_{\sigma\curvearrowright}\log\hat{\mathsf{q}}_{\sigma\curvearrowright} - \sum_{i=1}^{N}\hat{\mathsf{q}}_i\log\hat{\mathsf{q}}_i. \tag{14}$$

The last term in Eq. (14) is independent of partition. Therefore, when we merge two modules, we only need to keep track of changes in $\hat{\mathsf{q}}_{\sigma\curvearrowleft}$, $\hat{\mathsf{q}}_{\sigma\curvearrowright}$, and $\hat{\mathsf{p}}^{\sigma}_{\circlearrowleft}$, which are defined in Eq. (12). In these quantities, $\sum_{a=1}^{M}\delta_{\sigma',\sigma_{\zeta_{a0}}}$ is the population of the starting-point nodes in module σ, $\sum_{\sigma}\hat{\mathsf{q}}_{\sigma'\sigma}$ and $\sum_{\sigma'}\hat{\mathsf{q}}_{\sigma'\sigma}$ are the sums of the populations of the transitions across modules in the set of trajectories, and $\sum_{i\in\sigma}\hat{\mathsf{q}}_i$ is the sum of the node-visiting frequencies in module σ. They are $O(K)$ quantities, such that we can efficiently compute the change in $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ when two modules are merged.

When we use the actual average code length $\overline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ as the objective function, the greedy update cannot be computed as efficiently as for $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$. When two modules are merged, we need to reconstruct the intra-module codebook of the target module as well as the inter-module codebook to compute the updated code length. The time complexity of constructing a codebook depends on the specific coding scheme applied. In Supplementary Fig. 6, we show the running times of Infomap and Infomap+ on the SBM; herein, we used the Infomap API[53] (a C++-based implementation with a Python wrapper) for Infomap and our Python-based implementation[33] for Infomap+.

Resolution limit. Readers might consider that the pruning effect implies that the proposed method is prone to underfitting. To examine this issue, we derive the resolution limit of the single-trajectory map equation focusing on $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ and network datasets. The resolution limit is the smallest module size that the method can identify given a network size such as the total number of edges.

The following analysis shows that, although the method has a relatively coarser-resolution scale compared with the standard or hierarchical map equation, it is still a high-resolution method. The analysis also provides a theoretical explanation of some of the empirical results we obtained through the experiments in Experiments section and an implication to the range that the hyperparameter λ should take.

General form. We closely follow the derivation in[27], which is applied to undirected networks. The present resolution limit is for directed networks and the considered objective function is $\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\})$ instead of $L(\boldsymbol{\sigma})$.

We first rewrite the empirical frequencies of the walkers and the objective function in the single-trajectory map equation in terms of network statistics. When the input trajectories are the edges in a directed network (i.e., the number of trajectories M is the number of directed edges), we have

$$\begin{aligned}\hat{\mathsf{q}}_{\sigma\curvearrowright} &= \frac{\ell^{\text{out}}_{\sigma}}{2M}, \quad \hat{\mathsf{q}}_{\sigma\curvearrowleft} = \frac{\ell_{\sigma}+\ell^{\text{out}}_{\sigma}}{2M} + \frac{\ell^{\text{in}}_{\sigma}}{2M}, \quad \hat{\mathsf{q}}_{\curvearrowleft} = \frac{M+C}{2M},\\ \hat{\mathsf{p}}^{\sigma}_{\circlearrowleft} &= \frac{\ell^{\text{out}}_{\sigma}}{2M} + \frac{2\ell_{\sigma}+\ell^{\text{out}}_{\sigma}+\ell^{\text{in}}_{\sigma}}{2M}, \quad \hat{\mathsf{q}}_i = \frac{d^{\text{in}}_i + d^{\text{out}}_i}{2M},\end{aligned} \tag{15}$$

where ℓ_{σ} is the number of directed edges within module σ; $\ell^{\text{in}}_{\sigma}$ and $\ell^{\text{out}}_{\sigma}$ are the numbers of in-coming and out-going edges of module σ, respectively; d^{in}_i and d^{out}_i are the in- and out-degrees of node i; and C is the cut size of the network, i.e., the total number of directed edges that are crossing different modules. Using Eq. (15), the objective function is recast as

$$\begin{aligned}\underline{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma};\{\boldsymbol{\zeta}_a\}) = \frac{1}{2M}\Bigg(&\lambda(M+C)\log(M+C) + C + \sum_{\sigma}\underline{\mathscr{L}}^{\sigma}_{\lambda} + 2M\\ &- \sum_{i=1}^{N}\left(d^{\text{in}}_i + d^{\text{out}}_i\right)\log\left(d^{\text{in}}_i + d^{\text{out}}_i\right)\Bigg),\end{aligned} \tag{16}$$

where

$$\begin{aligned}\underline{\mathscr{L}}^{\sigma}_{\lambda} = &-\lambda\left(\ell_{\sigma}+\ell^{\text{out}}_{\sigma}+\ell^{\text{in}}_{\sigma}\right)\log\left(\ell_{\sigma}+\ell^{\text{out}}_{\sigma}+\ell^{\text{in}}_{\sigma}\right)\\ &+ 2\left(\ell_{\sigma}+\ell^{\text{out}}_{\sigma}+\frac{1}{2}\ell^{\text{in}}_{\sigma}\right)\log\left(\ell_{\sigma}+\ell^{\text{out}}_{\sigma}+\frac{1}{2}\ell^{\text{in}}_{\sigma}\right) - \ell^{\text{out}}_{\sigma}\log\ell^{\text{out}}_{\sigma}.\end{aligned} \tag{17}$$

In the resolution-limit analysis, we consider two well-defined modules and derive the condition under which their merging is favoured (i.e., the modules are not resolved) for better optimisation of the objective function. Thus, we evaluate the condition such that the difference in the objective function $\Delta\underline{\mathscr{L}}^{\sigma}_{\lambda}$ becomes negative when two modules are merged. We denote the labels of two well-defined modules as A and B and the merged module as AB. We also denote the change in $\sum_{\sigma}\underline{\mathscr{L}}^{\sigma}_{\lambda}$ through the update as R, i.e.,

$$R = \underset{\sim}{\mathscr{L}}_{\lambda}^{AB} - \underset{\sim}{\mathscr{L}}_{\lambda}^{A} - \underset{\sim}{\mathscr{L}}_{\lambda}^{B}. \tag{18}$$

Here, R is a local quantity that depends only on the variables within/around modules A and B. When two well-defined modules are merged, the cut size is decreased by a small δ ($\delta \ll M + C$). The difference in the objective function based on the update is

$$\begin{aligned} \Delta \underset{\sim}{\mathscr{L}}_{\lambda}(\boldsymbol{\sigma}; \{\boldsymbol{\zeta}_a\}) &= \frac{1}{2M}\Big(\lambda(M + C - \delta)\log(M + C - \delta) - \lambda(M + C)\log(M + C) - \delta + R\Big) \\ &\simeq \frac{1}{2M}\Big(-\delta\lambda\log\left(e(M + C)\right) - \delta + R\Big), \end{aligned} \tag{19}$$

where e is the basis of the natural logarithm. Therefore, the resolution limit is generally expressed as

$$R \lesssim \delta\Big(1 + \lambda\log\left(e(M + C)\right)\Big). \tag{20}$$

In the map equation, the cut size C is the only global term that is responsible for the resolution limit (see equation (11) in[27]). By contrast, the single-trajectory map equation has the total number of directed edges M as another global term in Eq. (20). Note, however, that the contribution from M is logarithmic, implying that the single-trajectory map equation is still a high-resolution method. Next, we will derive a more explicit scaling.

Ring of cliques. It is common to consider a "ring of cliques" in a resolution-limit analysis, as illustrated in Fig. 8a. We consider m cliques (each of which consists of n nodes) and connect each with a single edge to form a ring. This is an undirected network. Again, we treat each undirected edge as a pair of directed edges in both directions. We regard each clique as a module, and using this example, derive the resolution limit in a more explicit form.

When we merge two of these cliques, the cut size is decreased by 2. We denote $\ell_\sigma = n(n-1) = \ell$ for an arbitrary module. Assuming that $\ell \gg 1$, we have

$$R \simeq -2(\lambda - 2)(\ell + 3) + 2(\lambda - 1)\big(\log(e\ell)\big). \tag{21}$$

Substituting Eq. (21) into Eq. (20), we obtain

$$\frac{\ell^{\lambda-1}2^{(2-\lambda)(\ell+3)-1}}{e} \lesssim (M + C)^{\lambda}. \tag{22}$$

Each clique is resolved as a module unless $(M + C)^{\lambda}$ is larger than the left-hand side of Eq. (22), which is an exponentially growing function with respect to the clique size ℓ.

Figure 8b depicts the resolution limits of the single-trajectory map equation, together with those of the map equation[27] and modularity[43]. Although n and m are integers, we treat them as real numbers to highlight the scaling of each resolution limit. The resolution limit with $\lambda = 1$ is extremely close to that of the map equation. Therefore, the single-trajectory map equation is not prone to underfitting compared with the map equation.

When λ is large, modules with a small n are not resolved for any network size. However, the limit rapidly disappears as n becomes larger, whereas the resolution limit of the modularity disappears relatively slowly. This dependency of the resolution limit partially explains the favourable behaviour of the single-trajectory map

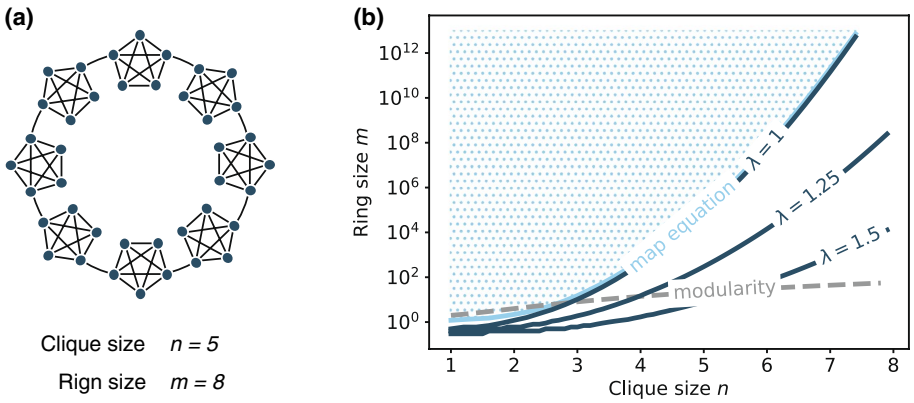

Figure 8. Ring of cliques and its resolution limit. (**a**) Network plot with $n = 5$ and $m = 8$, and (**b**) the resolution limits of the map equation (light blue line), the single-trajectory map equation with $\lambda = (1, 1.25, 1.5)$ (dark blue lines), and the modularity (dashed grey line). Each line represents the phase boundary above which a clique is not resolved as a module because the network is too large (e.g., the marked region represents the undetectable region in the map equation).

equation, i.e., small modules are pruned yet large modules continue to be identified. However, as pointed out in the main text, the resolution limit does not describe the full behaviour of the single-trajectory map equation; it is not λ that plays a critical role in the method and $\lambda = 1$ is often sufficient to avoid overfitting.

In the left-hand side of Eq. (22), the leading coefficient in the exponent becomes negative at $\lambda = 2$. In this case, a clique will not be resolved as a module for any network size regardless of its size n, i.e., the ability as a community detection method will be completely lost. This transition implies that the optimal value of λ is usually located within $1 \leq \lambda < 2$, which is indeed consistent with our experimental results in Fig. 6.

Data availability

The karate club, Les Miserables, political books, football, and power grid datasets were downloaded from http://www-personal.umich.edu/~mejn/netdata/. The C-elegans-frontal, email-Eu-core, and wiki-Vote datasets were downloaded from http://konect.cc/networks/. The political blogs dataset was downloaded from https://snap.stanford.edu/data/. The E. coli transcription dataset was downloaded from https://www.weizmann.ac.il/mcb/UriAlon/e-coli-transcription-network. The bike-sharing dataset was downloaded from https://github.com/konstantinklemmer/bikecommclust.

Code availability

The codes for the single-trajectory map equation are available at https://github.com/tatsuro-kawamoto/single-trajectory_map_equation.

Received: 1 December 2022; Accepted: 20 April 2023
Published online: 22 April 2023

References

1. Schaeffer, S. E. Graph clustering. *Comput. Sci. Rev.* **1**, 27–64 (2007).
2. Fortunato, S. Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010).
3. Fortunato, S. & Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **659**, 1–44 (2016).
4. Jin, D. *et al.* A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Trans. Knowl. Data Eng.* **35**(2), 1149–1170 (2023).
5. Ghasemian, A., Hosseinmardi, H. & Clauset, A. Evaluating overfit and underfit in models of network community structure. *IEEE Trans. Knowl. Data Eng.* **32**, 1722–1735 (2020).
6. Rosvall, M. & Bergstrom, C. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. U.S.A.* **105**, 1118–1123 (2008).
7. https://www.mapequation.org/.
8. Rosvall, M. & Bergstrom, C. T. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE* **6**, e18209 (2011).
9. Viamontes Esquivel, A. & Rosvall, M. Compression of flow can reveal overlapping-module organization in networks. *Phys. Rev. X* **1**, 021025 (2011).
10. Rosvall, M. *et al.* Memory in network flows and its effects on spreading dynamics and community detection. *Nat. Commun.* **5**, 4630 (2014).
11. De Domenico, M., Lancichinetti, A., Arenas, A. & Rosvall, M. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Phys. Rev. X* **5**, 011027 (2015).
12. Kheirkhahzadeh, M., Lancichinetti, A. & Rosvall, M. Efficient community detection of network flows for varying Markov times and bipartite networks. *Phys. Rev. E* **93**, 032309 (2016).
13. Edler, D., Bohlin, L. & Rosvall, M. Mapping higher-order network flows in memory and multilayer networks with infomap. *Algorithms* **10**, 112 (2017).
14. Aslak, U., Rosvall, M. & Lehmann, S. Constrained information flows in temporal networks reveal intermittent communities. *Phys. Rev. E* **97**, 062312 (2018).
15. Emmons, S. & Mucha, P. J. Map equation with metadata: Varying the role of attributes in community detection. *Phys. Rev. E* **100**, 022301 (2019).
16. Smiljanić, J., Edler, D. & Rosvall, M. Mapping flows on sparse networks with missing links. *Phys. Rev. E* **102**, 012302 (2020).
17. Blöcker, C. & Rosvall, M. Mapping flows on bipartite networks. *Phys. Rev. E* **102**, 052305 (2020).
18. Eriksson, A., Edler, D., Rojas, A., de Domenico, M. & Rosvall, M. How choosing random-walk model and network representation matters for flow-based community detection in hypergraphs. *Commun. Phys.* **4**, 1–12 (2021).
19. Smiljanić, J., Blöcker, C., Edler, D. & Rosvall, M. Mapping flows on weighted and directed networks with incomplete observations. *J. Complex Netw.***9** (2021).
20. Arenas, A., Danon, L., Diaz-Guilera, A., Gleiser, P. M. & Guimera, R. Community analysis in social networks. *Eur. Phys. J. B* **38**, 373–380 (2004).
21. Clauset, A., Newman, M. E. J. & Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **70**, 066111 (2004).
22. Leskovec, J., Lang, K. J., Dasgupta, A. & Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Math.* **6**, 29–123 (2009).
23. Wakita, K. & Tsurumi, T. Finding community structure in mega-scale social networks: [extended abstract]. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, 1275–1276 (Association for Computing Machinery, New York, NY, USA, 2007).
24. Moore, C. The computer science and physics of community detection: landscapes, phase transitions, and hardness. *arXiv preprint* arXiv:1702.00467 (2017).
25. Peixoto, T. P. Bayesian stochastic blockmodeling. In *Advances in Network Clustering and Blockmodeling* (eds Doreian, V. & Batagelj, A. Ferligoj.) (Wiley, New York, 2019).
26. Hastie, T. J., Tibshirani, R. J. & Friedman, J. H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics* (Springer, New York, 2009).
27. Kawamoto, T. & Rosvall, M. Estimating the resolution limit of the map equation in community detection. *Phys. Rev. E* **91**, 012809 (2015).
28. Schaub, M. T., Lambiotte, R. & Barahona, M. Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation. *Phys. Rev. E* **86**, 026112 (2012).
29. MacKay, D. J. & Mac Kay, D. J. *Information Theory, Inference and Learning Algorithms* (Cambridge University Press, Cambridge, 2003).

30. Cover, T. M. *Elements of Information Theory* (Wiley, New York, 1999).
31. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948).
32. Brin, S. & Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **30**, 107–117 (1998).
33. https://github.com/authorname/single-trajectory_map_equation.
34. Decelle, A., Krzakala, F., Moore, C. & Zdeborová, L. Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.* **107**, 065701 (2011).
35. Mossel, E., Neeman, J. & Sly, A. Reconstruction and estimation in the planted partition model. *Probab. Theory Relat. Fields* **162**, 431–461 (2015).
36. Massoulié, L. Community detection thresholds and the weak ramanujan property. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14*, 694–703 (ACM, New York, 2014).
37. Holland, P. W., Laskey, K. B. & Leinhardt, S. Stochastic blockmodels: First steps. *Soc. Netw.* **5**, 109–137 (1983).
38. Wang, Y. J. & Wong, G. Y. Stochastic blockmodels for directed graphs. *J. Am. Stat. Assoc.* **82**, 8–19 (1987).
39. Abbe, E. Community detection and stochastic block models: Recent developments. *J. Mach. Learn. Res.* **18**, 1–86 (2018).
40. Kawamoto, T. & Kabashima, Y. Counting the number of metastable states in the modularity landscape: Algorithmic detectability limit of greedy algorithms in community detection. *Phys. Rev. E* **99**, 010301 (2019).
41. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, P10008 (2008).
42. Traag, V. A., Waltman, L. & Van Eck, N. J. From Louvain to Leiden: Guaranteeing well-connected communities. *Sci. Rep.* **9**, 1–12 (2019).
43. Fortunato, S. & Barthélemy, M. Resolution limit in community detection. *Proc. Natl. Acad. Sci. U.S.A.* **104**, 36–41 (2007).
44. Peixoto, T. P. Model selection and hypothesis testing for large-scale network models with overlapping groups. *Phys. Rev. X* **5**, 011033 (2015).
45. Peixoto, T. P. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E* **95**, 012317 (2017).
46. Kawamoto, T. & Kabashima, Y. Cross-validation estimate of the number of clusters in a network. *Sci. Rep.* **7**, 3327 (2017).
47. Kawamoto, T. & Kabashima, Y. Comparative analysis on the selection of number of clusters in community detection. *Phys. Rev. E* **97**, 022315 (2018).
48. Munoz-Mendez, F., Klemmer, K., Han, K. & Jarvis, S. Community structures, interactions and dynamics in London's bicycle sharing network. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers - UbiComp '18*, 1015–1023 (ACM Press, New York, New York, USA, 2018). http://dl.acm.org/citation.cfm?doid=3267305.3274156.
49. https://github.com/konstantinklemmer/bikecommclust.
50. Lambiotte, R. & Rosvall, M. Ranking and clustering of nodes in networks with smart teleportation. *Phys. Rev. E* **85**, 056107 (2012).
51. Rosvall, M. & Bergstrom, C. T. Mapping change in large networks. *PLoS ONE* **5**, 1–7 (2010).
52. Clauset, A., Moore, C. & Newman, M. E. Hierarchical structure and the prediction of missing links in networks. *Nature* **453**, 98–101 (2008).
53. https://mapequation.github.io/infomap/python/.

Acknowledgements

The author is grateful to Martin Rosvall for inspiring discussions. The author also thanks Christopher Blöcker and Teruyoshi Kobayashi for fruitful comments. This work was supported by JST ACT-X Grant No. JPMJAX21A8 and JSPS KAKENHI No. 19H01506.

Author contributions

T.K. designed the research, conducted analytical calculations, analyzed real data, implemented the code, and wrote the manuscript.

Competing interests

The author declares no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-33880-y.

Correspondence and requests for materials should be addressed to T.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.