Randomized Shortest Paths with Net Flows and Capacity Constraints

Draft manuscript subject to changes

Sylvain Courtain, Pierre Leleux, Ilkka Kivimaki,
Guillaume Guex & Marco Saerens

Abstract

This work extends the randomized shortest paths model (RSP) by investigating the net flow RSP and adding capacity constraints on edge flows. The standard RSP is a model of movement, or spread, through a network interpolating between a random walk and a shortest path behavior [54, 80, 97]. The framework assumes a unit flow injected into a source node and collected from a target node with flows minimizing the expected transportation cost together with a relative entropy regularization term. In this context, the present work first develops the net flow RSP model considering that edge flows in opposite directions neutralize each other (as in electrical networks) and proposes an algorithm for computing the expected routing costs between all pairs of nodes. This quantity is called the net flow RSP dissimilarity measure between nodes. Experimental comparisons on node clustering tasks show that the net flow RSP dissimilarity is competitive with other state-of-the-art dissimilarities. In the second part of the paper, it is shown how to introduce capacity constraints on edge flows and a procedure solving this constrained problem by using Lagrangian duality is developed. These two extensions should improve significantly the scope of applications of the RSP framework.

1 Introduction

Link analysis and network science are devoted to the analysis of network data and are currently studied in a large number of different fields (see, e.g., [6, 13, 19, 29, 33, 56, 61, 69, 82, 88, 93]). One recurring problem in this context is the definition of meaningful measures capturing interesting properties of the network and its nodes, like distance measures between nodes or centrality measures. These quantities usually take the structure of the whole network into account.

However, many such measures are based on some strong assumptions about the movement, or communication, occurring in the network whose two extreme cases are the optimal behavior and the random behavior. Indeed, the two most popular distance measures in this context are the least-cost distance (also called shortest-path distance) and the resistance distance [55] (equal to the effective resistance and proportional to the commute-time distance when considering a random walk on the graph [32]). The same holds with the betweenness centrality; popular measures are the shortest path betweenness introduced by Freeman [36] (based on shortest paths) and the random-walk

betweenness (also called the current-flow betweenness) introduced both by Newman [68] and by Brandes and Fleischer [14], independently. Still another example is provided by the measures of compactness of a network, the Wiener index (based on shortest paths; see, e.g., [13]) and the Kirchhoff index (based on random walks or electrical networks; see, e.g., [55]).

But, in reality, behavior is seldom based on complete randomness or optimality. Therefore, in the last years, there has been a large effort invested in defining models interpolating between shortest-path and random-walk behaviors, especially in the context of defining distance measures between nodes taking both proximity and high connectivity into account [33]. Some of these models are based on extensions of electrical networks [3, 45, 70, 92], some on combinatorial analysis arguments [15, 16, 17], some on mixed L1-L2 regularization [62, 63], some on entropy-regularized network flow models [7, 43], and still others on entropy-regularized least-cost paths ([34, 54, 80, 97], inspired by the transportation model [2] and related to [89]) – the **randomized shortest paths** (RSP) framework which is the main subject of this paper. Recently, the RSP has been successfully used in, e.g., defining betweenness measures interpolating between a variant of the shortest-path betweenness and the random-walk betweenness [53], modelling the behavior of animals [73], and fraud detection [21]. For a more thorough discussion of families of distances between nodes, see, e.g., [33, 34, 54].

This effort is pursued here by proposing two extensions of this RSP model, considering

- a new dissimilarity measure extending the RSP dissimilarity [54, 80, 97], namely the **net-flow RSP dissimilarity**. This new dissimilarity is – like the standard RSP dissimilarity – the expected cost needed for reaching the target node from the source node, but now considering that edge flows in two opposite directions cancel out, as for electrical currents. Therefore, this model of movement based on net flows resembles more to electric flows when the temperature of the system is high. An algorithm is proposed for computing the net-flow RSP dissimilarity matrix between all pairs of nodes.

- the introduction of **capacity constraints** in the model. Capacity constraints on edge flows are very common in practice [1, 27] and it would certainly increase significantly the applicability of the RSP model if such constraints could be integrated. Therefore, the main contributions related to capacity constraints are (1) to show how the model can accommodate such constraints, for both raw edge flows and net flows, and (2) to provide an algorithm for solving the constrained RSP model in the case of a single pair of source/target nodes.

Capacity constraints are common in network flow problems and there is a vast literature on the subject (see, e.g., [1, 8, 27, 31, 40, 50, 58] and references therein). They appear for instance in the standard *minimum cost flow* problem which aims at finding the minimum cost source-target flow subject to some capacity constraints. As discussed in [1], this task frequently appears in real-world problems; see this reference for concrete applications. Capacity constraints are present for, e.g., avoiding congestion, spreading the traffic, or simply because the flow is restricted. Various algorithms for solving the standard minimum cost flow problem have been proposed: minimum mean cycle-cancelling, successive shortest paths, network simplex, primal-dual, to name a few (see the above mentioned references). The difference with our work is that, in the RSP, the model is expressed in terms of full paths and a Kullback-Leibler regularization term

is introduced, inducing randomized policies that can be computed by standard matrix operations. Because of their practical usefulness, we found that it would be valuable to introduce such capacity constraints within the RSP framework, which is developed in Section 4.

The content of the paper are as follows. Section 2 summarises the standard RSP framework. Section 3 introduces the net flow RSP dissimilarity. Then, Section 4 develops new algorithms for constraining the flow capacity on edges, while Section 5 deals with net flow capacity constraints. Illustrative examples and experiments on node clustering tasks are described in Section 6. Finally, Section 7 is the conclusion.

2 The standard randomized shortest paths framework

As already discussed, the main contributions of the paper are based on the RSP framework, interpolating between a least-cost and a random-walk behavior, and allowing to define dissimilarity measures between nodes [33, 54, 80, 97]. The formalism, inspired by models developed in transportation science [2], is based on full paths instead of standard "local" edge flows [1, 27] and is briefly described in this section for completeness. We start by providing a short account of the RSP formalism before introducing, in the next sections, the net flow RSP dissimilarity as well as the algorithm for solving the flow-capacity constrained problem on a directed graph.

2.1 Background and notation

Let us first introduce some necessary notation [33, 34]. First, notice that column vectors are written in bold lowercase and matrices are in bold uppercase. Moreover, in this work, we consider a weighted directed[1], strongly connected, graph, $G = (\mathcal{V}, \mathcal{E})$, with a set $\mathcal{V}$ of n nodes and a set $\mathcal{E}$ of edges. An edge connecting node i to node j is denoted by (i, j) or $i \to j$.

Furthermore, we are given an adjacency matrix $\mathbf{A} = (a_{ij}) \geq 0$ quantifying the directed, local, affinity between pairs of nodes i, j. We further assume that there are no self-loops in the network, that is, $a_{ii} = 0$ for all i. From this adjacency matrix, the standard **reference random walk** on the graph is defined in the usual way: the **transition probabilities** associated to each node are set proportionally to the affinities and then normalized in order to sum to one,

$$p_{ij}^{\text{ref}} = \frac{a_{ij}}{\sum_{j'=1}^{n} a_{ij'}} = \frac{a_{ij}}{d_i} \tag{1}$$

where d_i is the (out)degree of node i. The matrix $\mathbf{P}_{\text{ref}} = (p_{ij}^{\text{ref}})$ is stochastic and is called the transition matrix of the natural, reference, random walk on the graph.

In addition, a transition **cost**, $c_{ij} \geq 0$, is associated to each edge (i, j) of G. If there is no edge linking i to j, the cost is assumed to take an infinite value, $c_{ij} = \infty$. For consistency, we assume that $c_{ij} = \infty$ if $a_{ij} = 0$. The cost matrix is defined accordingly, $\mathbf{C} = (c_{ij})$. Costs are usually set independently of the adjacency matrix; they quantify the immediate cost of a transition, depending on the application at hand (see [34] for

[1] If the graph is undirected, we consider that it is made of two, reciprocal, directed edges with the same affinities and costs.

a discussion). In electrical networks, the costs are resistances and the affinities are conductances; in this context, they are linked by $a_{ij} = 1/c_{ij}$.

A **path** or **walk** $\wp$ is a finite sequence of hops to adjacent nodes on G (including cycles), initiated from a source node s and stopping in some ending, target, node t. A **hitting path** is a path where the last node t does not appear as an intermediate node. In other words, a hitting path to node t stops when it reaches t for the first time. We consider hitting paths to the fixed target node t by setting this target node as absorbing (or "killing"). Computationally this is achieved by setting the corresponding row t of the adjacency matrix and of the transition matrix to zero.

The node along path $\wp$ at position τ on the path is denoted by $\wp(\tau)$. The **total cost** of a path, $\tilde{c}(\wp)$, is simply the sum of the edge costs c_{ij} along $\wp$ while its **length** $\ell(\wp)$ is the number of steps, or hops, needed for following that path.

2.2 The standard randomized shortest paths formalism

The main idea behind the RSP model is the following. Let us consider the set of all hitting paths, or walks, $\wp \in \mathcal{P}_{st}$ from a node $s \in \mathcal{V}$ (source) to an absorbing, killing, node $t \in \mathcal{V}$ (target) on G. Then, we assign a probability distribution $\mathrm{P}(\cdot)$ on the discrete set of paths $\mathcal{P}_{st}$ by minimizing the **free energy** [7, 48, 54, 74, 78],

$$\left| \begin{array}{ll} \underset{\{\mathrm{P}(\wp)\}_{\wp \in \mathcal{P}_{st}}}{\text{minimize}} & \phi(\mathrm{P}) = \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\dfrac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) \\ \text{subject to} & \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) = 1 \end{array} \right. \tag{2}$$

where $\tilde{c}(\wp) = \sum_{\tau=1}^{\ell(\wp)} c_{\wp(\tau-1)\wp(\tau)}$ is the total cumulated cost along path $\wp$ when visiting the nodes $(\wp(\tau))_{\tau=0}^{\ell(\wp)}$ in the sequential order. Furthermore, $\tilde{\pi}(\wp) = \prod_{\tau=1}^{\ell(\wp)} p^{\mathrm{ref}}_{\wp(\tau-1)\wp(\tau)}$ is the product of the reference transition probabilities along path $\wp$, i.e., the random walk probability of path $\wp$.

The objective function (Eq. 2) is a mixture of two dissimilarity terms, with the temperature $T > 0$ balancing the trade-off between them. The first term is the expected cost for reaching target node from source node (favoring shorter paths – *exploitation*). The second term corresponds to the relative entropy, or Kullback-Leibler divergence, between the path probability distribution and the reference (random walk) paths probability distribution (introducing randomness and diversity [75] – *exploration*). For a low temperature T, low-cost paths are favoured whereas when T is large, paths are chosen according to their likelihood in the reference random walk on G.

The problem (2) corresponds to a standard minimum cost flow problem, as discussed in the introduction[2], with a Kullback-Leibler regularization term expressed in terms of full paths in the RSP formalism. There are, however, some subtle differences, like for instance the fact that in the standard minimum cost flow problem, the flows are unidirectional whereas the RSP defines a Markov chain for which flows are generally bi-directional.

This argument, akin to maximum entropy [22, 48, 51], leads to a **Gibbs-Boltzmann distribution** on the set of paths (see, e.g., [34]),

$$\mathrm{P}^*(\wp) = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]}{\sum_{\wp' \in \mathcal{P}_{st}} \tilde{\pi}(\wp')\exp[-\theta\tilde{c}(\wp')]} = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]}{\mathcal{Z}} \tag{3}$$

[2]Here, without capacity constraints which will be introduced in Section 4.

where $\theta = 1/T$ is the inverse temperature and the denominator $\mathcal{Z} = \sum_{\wp \in \mathcal{P}_{st}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]$ is the **partition function** of the system. This provides the randomized routing policy in terms of paths.

Interestingly, if we replace the probability distribution $\mathrm{P}(\cdot)$ by the optimal distribution $\mathrm{P}^*(\cdot)$ provided by Eq. 3 in the objective function (Eq. 2), we obtain

$$\begin{aligned} \phi^* = \phi(\mathrm{P}^*) &= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}^*(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}^*(\wp) \log\left(\frac{\mathrm{P}^*(\wp)}{\tilde{\pi}(\wp)}\right) \\ &= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}^*(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}^*(\wp)\Big(-\tfrac{1}{T}\tilde{c}(\wp) - \log \mathcal{Z}\Big) \\ &= -T \log \mathcal{Z} \end{aligned} \tag{4}$$

which is called the directed **free energy distance** [54], playing the role of a potential in the equivalent continuous state-space diffusion process [38].

2.3 Computing quantities of interest

The quantities of interest can be computed by taking the partial derivative of the optimal free energy provided by Eq. 4 [33, 34, 54, 80, 97]. Here, we only introduce the quantities that are needed in order to derive the algorithms developed later.

Fundamental matrix and partition function It turns out that the partition function can be computed in closed form from an auxiliary matrix[3] $\mathbf{W}$. First, the **fundamental matrix** of the RSP system is defined as

$$\mathbf{Z} = \mathbf{I} + \mathbf{W} + \mathbf{W}^2 + \cdots = (\mathbf{I} - \mathbf{W})^{-1} \quad \text{with} \quad \mathbf{W} = \mathbf{P}_{\mathrm{ref}} \circ \exp[-\theta \mathbf{C}] \tag{5}$$

where $\mathbf{C}$ is the cost matrix and $\circ$ is the elementwise (Hadamard) product. The equation sums up contributions of different paths lengths, starting from zero-length paths (identity matrix $\mathbf{I}$). The partition function is then provided by $\mathcal{Z} = [\mathbf{Z}]_{st} = z_{st}$ [34, 54, 80, 97].

Computation of flows and numbers of visits The directed **flow** on edge (i, j) (the expected number of passages through (i, j) when going from s to t) can be obtained from Eq. 4 and Eq. 5 for a given inverse temperature θ (see [53, 80] for details) by

$$\bar{n}_{ij} \triangleq \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\, \eta\big((i,j) \in \wp\big) = -\tfrac{1}{\theta} \frac{\partial \log \mathcal{Z}}{\partial c_{ij}} = \frac{z_{si} w_{ij} z_{jt}}{z_{st}} \tag{6}$$

where $\eta\big((i,j) \in \wp\big)$ counts the number of times edge (i, j) appears on path $\wp$. Now, as only the row s and the column t of fundamental matrix $\mathbf{Z}$ are needed, two systems of linear equations can be solved instead of matrix inversion in Eq. 5. Let us further define the matrix containing the expected number of passages through edges (i, j) by $\mathbf{N} = (\bar{n}_{ij})$. From the last Eq. 6, the expected **number of visits** to a node j can also be defined, and easily computed, as

$$\bar{n}_j = \sum_{i \in \mathcal{P}red(j)} \bar{n}_{ij} + \delta_{sj} = \sum_{i=1}^{n} \bar{n}_{ij} + \delta_{sj} = \frac{z_{sj} z_{jt}}{z_{st}} + \delta_{sj}. \tag{7}$$

[3] Recall that the target node t is made killing and absorbing by setting the corresponding row of the reference transition matrix to zero, which implies that row t of $\mathbf{W}$ is also zero.

because a unit flow of $+1$ is injected in node s. Note that $\mathcal{P}red(j)$ is the set of predecessor nodes of node j and we used $\sum_{i=1}^{n} z_{si} w_{ij} = z_{sj}$.

Optimal transition probabilities Finally, the optimal transition probabilities of following any edge $(i, j) \in \mathcal{E}$ (the policy) induced by the set of paths $\mathcal{P}_{st}$ and their probability mass (Eq. 3) are [80]

$$p^*_{ij} = \frac{\bar{n}_{ij}}{\sum_{j'=1}^{n} \bar{n}_{ij'}} = \frac{z_{jt}}{z_{it}} w_{ij} \quad \text{for all } i \neq t \tag{8}$$

which defines a **biased random walk** (a Markov chain) on G – the random walker is "attracted" by the target node t. The lower the temperature, the larger the attraction. Interestingly, these transition probabilities do not depend on the source node and correspond to the optimal randomized strategy, or policy, for reaching target node, minimizing free energy (Eq. 2).

3 The net flow randomized shortest paths dissimilarity

In this section, we introduce the **net flow RSP** dissimilarity extending the standard RSP dissimilarity developed in [54, 80, 97]. As for the standard RSP dissimilarity, it corresponds to the expected cost for reaching target node t from source node s but with the important difference that *net flows* are considered instead of raw flows. These measures are now introduced in this section.

3.1 Definition of the net edge flow

In some situations, for instance electrical networks [10, 28], only net flows matter. Intuitively, it means that the edge flows in opposite directions $i \to j$ and $j \to i$ compensate each other so that only the positive net flow, defined as $|\bar{n}_{ij} - \bar{n}_{ji}|$, is taken into account, where edge flows are given by Eq. 6. In many situations, net flows look intuitively more natural because of the flow compensation mechanism common to electricity. Net flows have already been used in the RSP framework in order to define node betweenness measures [53], generalizing two previous models based on electrical currents [14, 68]. They are further investigated in this section in order to define a new dissimilarity measure between nodes of a graph.

As for electrical currents [10, 28], the non-negative net flow in each edge (i, j), denoted here as j_{ij}, is given in the RSP formalism by

$$j_{ij} = \max\big((\bar{n}_{ij} - \bar{n}_{ji}), 0\big) \tag{9}$$

or, in matrix form,

$$\mathbf{J} = \max\big((\mathbf{N} - \mathbf{N}^{\mathrm{T}}), \mathbf{0}\big) \tag{10}$$

where the maximum is taken elementwise. This means that, for each edge, the net flow is defined (that is, positive) in only one direction[4], and is equal to zero in the other direction.

Interestingly, because the flow is equal to zero in one of the two edge directions, the net flow defines a directed graph from the source to the destination node, even if the original graph is undirected.

[4]Another common convention is to consider $j_{ij} = (\bar{n}_{ij} - \bar{n}_{ji})$, and thus $j_{ji} = -j_{ij}$.

3.2 Expected net cost and net RSP dissimilarity measure

The **expected cost** until absorption by target node t at temperature T can easily be computed in closed form from the RSP formalism. This expected cost spread in the network has been used as a dissimilarity measure between nodes [54, 80, 97] and was called the directed **RSP dissimilarity**. More formally, in the standard RSP framework, the expected cost spread in the network [37] is given by

$$\langle \tilde{c} \rangle = \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \tilde{c}(\wp) \tag{11}$$

Let us now express the cost along path $\wp$ as $\tilde{c}(\wp) = \sum_{(i,j)\in\mathcal{E}} \eta\big((i,j) \in \wp\big)\, c_{ij}$ where we saw that $\eta\big((i,j) \in \wp\big)$ is the number of times edge (i,j) appears on path $\wp$ and $\mathcal{E}$ is the set of edges. Injecting this last result in Eq. 11 provides

$$\langle \tilde{c} \rangle = \sum_{(i,j)\in\mathcal{E}} \Bigg(\underbrace{\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\, \eta\big((i,j) \in \wp\big)}_{\text{expected number of passages } \bar{n}_{ij}} \Bigg) c_{ij} = \sum_{(i,j)\in\mathcal{E}} \bar{n}_{ij} c_{ij} \tag{12}$$

or, in matrix form,

$$\langle \tilde{c} \rangle = \mathbf{e}^{\mathrm{T}} (\mathbf{N} \circ \mathbf{C}) \mathbf{e} \tag{13}$$

where $\circ$ is the elementwise matrix product and $\mathbf{N}$ is the matrix of expected number of passages through edges defined in Eq. 6. This quantity is just the cumulative sum of the expected number of passages through each edge times the cost of following the edge.

When dealing with net flows instead, the Eq. 13, now computing the expected *net cost*, becomes

$$\langle \tilde{c}_{\mathrm{net}} \rangle = \mathbf{e}^{\mathrm{T}} \big[\max((\mathbf{N} - \mathbf{N}^{\mathrm{T}}), \mathbf{0}) \circ \mathbf{C} \big] \mathbf{e} = \mathbf{e}^{\mathrm{T}} (\mathbf{J} \circ \mathbf{C}) \mathbf{e} \tag{14}$$

This quantity can be interpreted as the net cost needed to reach the target node t from the source node s in a biased random walk (defined by Eq. 3 or Eq. 8) attracting the walker toward the target node t. It is therefore the equivalent of the expected first passage cost defined in Markov chain theory [72, 87], translated in the RSP formalism and for net flows. It can be seen as a *directed dissimilarity* between node s and node t taking both proximity and amount of connectivity between s and t into account.

When the temperature is large, $T \to \infty$, the directed dissimilarity $\langle \tilde{c}_{\mathrm{net}} \rangle_{st}$ reduces to the least-cost distance between s and t, while when $T \to 0^+$, it tends to the expected net cost for a random walker moving according to the reference random walk (and thus electrical current). This quantity is in fact equivalent to the so-called R_1 distance introduced in [70], i.e., the weighted-by-costs sum of the net flows.

Therefore, the **net flow RSP dissimilarity** (nRSP, the counterpart of the standard RSP dissimilarity [54, 80, 97]) between node s and node t is defined as the symmetrized quantity

$$\Delta_{st}^{\mathrm{nRSP}} = \langle \tilde{c}_{\mathrm{net}} \rangle_{st} + \langle \tilde{c}_{\mathrm{net}} \rangle_{ts} \tag{15}$$

where the starting and ending node are specified again. This is an equivalent of the symmetric commute-cost quantity appearing in Markov chains [32].

Notice the difference between this quantity and the energy spread in an electrical network. Indeed, if the costs are viewed as resistances, then, in the context of a resistive network, the energy weights the costs by the *squared* net flows – instead of the simple net flows in Eq. 14 [10, 28].

3.3 Net flows define a directed acyclic graph

Let us now show that the RSP net flows to a fixed target node t define a directed acyclic graph (DAG) when the reference probabilities are defined by Eq. 1 on the weighted undirected graph G. This comes from the fact that the net flows provided by Eq. 9 can be considered as an electrical current generated from a new graph $\hat{G}_t$ derived from G by redefining its edge weights. Moreover, it is well-known that electrical current defines a DAG because current always follows edges in the direction of decreasing potential (voltage). This potential therefore defines a topological ordering [20] of the nodes from higher potential to lower one.

More precisely, for a fixed target t, let us define the graph $\hat{G}_t$ by considering the following weights on edges (i, j) (conductances in electrical circuits)

$$\hat{a}_{ij} \triangleq z_{it} w_{ij} z_{jt} d_i = z_{it} p^{\text{ref}}_{ij} \exp(-\theta c_{ij}) z_{jt} d_i = z_{it} a_{ij} \exp(-\theta c_{ij}) z_{jt} \tag{16}$$

which is symmetric when $\mathbf{A}$ and $\mathbf{C}$ are symmetric. Accordingly, we define immediate costs $\hat{c}_{ij} = 1/\hat{a}_{ij}$ (resistances in electrical circuits). The natural transition probabilities on this new graph are provided by Eq. 1 where we replace a_{ij} by $\hat{a}_{ij}$,

$$\hat{p}_{ij} = \frac{\hat{a}_{ij}}{\sum_{j'=1}^{n} \hat{a}_{ij'}} = \frac{w_{ij} z_{jt}}{\sum_{j'=1}^{n} w_{ij'} z_{j't}} = \frac{z_{jt}}{z_{it}} w_{ij} \quad \text{for all } i \neq t \tag{17}$$

which, from Eq. 8, are exactly the optimal RSP transition probabilities. Note that we used the relation $z_{it} = \sum_{j'=1}^{n} w_{ij'} z_{j't} + \delta_{it}$, which can be easily derived from the definition of the fundamental matrix (Eq. 5) (see also, e.g., [33, 54, 80, 97]).

This shows that the net flows resulting from the optimal biased random walk provided by Eq. 8 are generated by a random walk on $\hat{G}_t$ (a Markov chain). From the equivalence between random walks on an undirected graph and electrical current on $\hat{G}_t$ [28], this current defines a DAG[5].

3.4 Computation of the net flow randomized shortest paths dissimilarity

This subsection shows how the net flow RSP dissimilarity between all pairs of nodes (Eq. 15) can be computed in matrix form from previous work [53]. Unfortunately, the computation of the net flow RSP dissimilarities is more time-consuming than computing the standard RSP dissimilarities, where it suffices to perform a matrix inversion [54]. This is because before being able to compute the dissimilarities, we need to find the net flows, which involves a non-linear function (max). It is, however, still feasible for small- to medium-size networks.

The algorithm for computing the net flow RSP dissimilarity is detailed in Algorithm 1. It uses a trick introduced in [34] for calculating the net flows between all sources-destinations s, t in a particular edge (i, j) without having to explicitly turn node t into a killing, absorbing, node. More specifically, the procedure is an adaptation of Algorithm 2 in [53] (following Eq. 12 in this work, providing net flows) to the case of an undirected graph and the computation of net flow dissimilarities, instead of betweenness centrality. It is also optimized in order to loop over (undirected) edges only once: on

[5]Net flows defined by a random walk on an undirected graph $\hat{G}_t$ correspond to the electrical currents [28].

Algorithm 1 Computing the net flow randomized shortest paths dissimilarity matrix.

Input:

– An undirected, connected, graph G containing n nodes.
– The $n \times n$ symmetric adjacency matrix $\mathbf{A}$ associated to G, containing non-negative affinities.
– The $n \times n$ reference transition probabilities matrix $\mathbf{P}_{\mathrm{ref}}$ associated to G (usually, the transition probabilities associated to the natural random walk on the graph, $\mathbf{P}_{\mathrm{ref}} = \mathbf{D}^{-1}\mathbf{A}$).
– The $n \times n$ symmetric non-negative cost matrix $\mathbf{C}$ associated to G.
– The inverse temperature parameter $\theta > 0$.

Output:

– The $n \times n$ randomized shortest paths net flow dissimilarity matrix $\boldsymbol{\Delta}$ defined on all source-destination pairs.

1. $\mathbf{W} \leftarrow \mathbf{P}_{\mathrm{ref}} \circ \exp[-\theta \mathbf{C}]$ ▷ elementwise exponential and multiplication $\circ$
2. $\mathbf{Z} \leftarrow (\mathbf{I} - \mathbf{W})^{-1}$ ▷ the fundamental matrix
3. $\boldsymbol{\Delta} \leftarrow \mathbf{0}$ ▷ initialize net RSP flow dissimilarity matrix
4. **for** $i = 1$ to n **do** ▷ compute contribution of each node i
5. $\quad \mathbf{z}_i^{\mathrm{c}} \leftarrow \mathbf{col}_i(\mathbf{Z}), \mathbf{z}_i^{\mathrm{r}} \leftarrow \mathbf{row}_i(\mathbf{Z})$ ▷ copy column i and row i of $\mathbf{Z}$ transformed into a column vector
6. $\quad$ **for** $j \in \mathcal{N}(i)$ with $j > i$ **do** ▷ loop on neighboring nodes j, considering each (undirected) edge only once
7. $\quad\quad \mathbf{z}_j^{\mathrm{c}} \leftarrow \mathbf{col}_j(\mathbf{Z}), \mathbf{z}_j^{\mathrm{r}} \leftarrow \mathbf{row}_j(\mathbf{Z})$ ▷ copy column j and row j of $\mathbf{Z}$ transformed into a column vector
8. $\quad\quad \mathbf{N}_{ij} \leftarrow w_{ij}\left[\left(\mathbf{z}_i^{\mathrm{c}}(\mathbf{z}_j^{\mathrm{r}})^{\mathrm{T}} \div \mathbf{Z}\right) - \mathbf{e}\left((\mathbf{z}_i^{\mathrm{c}} \circ \mathbf{z}_j^{\mathrm{r}}) \div \mathbf{diag}(\mathbf{Z})\right)^{\mathrm{T}}\right]$ ▷ matrix of flow in edge $i \to j$ for all source-destinations (see [53], Eq. 17)
9. $\quad\quad \mathbf{N}_{ji} \leftarrow w_{ji}\left[\left(\mathbf{z}_j^{\mathrm{c}}(\mathbf{z}_i^{\mathrm{r}})^{\mathrm{T}} \div \mathbf{Z}\right) - \mathbf{e}\left((\mathbf{z}_j^{\mathrm{c}} \circ \mathbf{z}_i^{\mathrm{r}}) \div \mathbf{diag}(\mathbf{Z})\right)^{\mathrm{T}}\right]$ ▷ matrix of flow in edge $j \to i$ for all source-destinations (see [53], Eq. 17)
10. $\quad\quad \mathbf{Net}_{ij} \leftarrow \mathrm{abs}(\mathbf{N}_{ij} - \mathbf{N}_{ji})$ ▷ net flow contribution from edge $i \leftrightarrow j$
11. $\quad\quad \boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + c_{ij}\mathbf{Net}_{ij}$ ▷ update dissimilarity matrix with the contribution of edge $i \leftrightarrow j$
12. $\quad$ **end for**
13. **end for**
14. $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + \boldsymbol{\Delta}^{\mathrm{T}}$ ▷ the resulting net RSP dissimilarities matrix
15. **return** $\boldsymbol{\Delta}$

line 10, the contributions of the two directions of edge (i, j) (one is necessarily equal to 0 and the other is equal to $|\bar{n}_{ij} - \bar{n}_{ji}|$) are summed together.

Following [53], the time complexity is $\mathcal{O}(n^3 + mn^2)$ where m is the number of edges and n the number of nodes, or $\mathcal{O}(m\,n^2)$ overall because $m \geq n$ for an undirected, connected, graph. Indeed, the algorithm contains a matrix inversion, which is $\mathcal{O}(n^3)$. Thereafter, there is a loop over all edges repeating some standard matrix operations of order $\mathcal{O}(n^2)$, which finally provides $\mathcal{O}(n^3 + mn^2)$. Therefore the algorithm does not scale well for large graphs; in its present form, it can only be applied on medium-size graphs.

4 Dealing with edge flow capacity constraints

In this section, an algorithm computing the optimal policy (Eq. 8) under capacity flow constraints on edges is derived. For convenience, we assume a weighted, undirected, connected, network G with a single source node (node s) and one single target node (node t). An input flow is injected in node s and absorbed in node t, but the model can easily be generalized to multiple sources and destinations as well as directed graphs. As before, it is assumed that the target node t is killing and absorbing, meaning that the transition probabilities $p_{tj}^{\mathrm{ref}} = 0$ for all nodes j, including node $j = t$.

The idea now is to constrain the flow visiting some edges belonging to a set of constrained edges $\mathcal{C}$. The expected number of passages through those edges (see Eq. 6) is therefore forced to not exceed some predefined values (upper bound),

$$\bar{n}_{ij} \leq \sigma_{ij} \quad \text{for edges } (i,j) \in \mathcal{C} \tag{18}$$

which ensures that the flows on edges in $\mathcal{C}$ are limited by the capacity $\sigma_{ij} > 0$ and thus must remain in the interval $[0, \sigma_{ij}]$. In this section, we consider that, although the graph G is undirected, the capacity constraints are directed and thus active in only one direction of an edge. Therefore, each undirected edge $i \leftrightarrow j$ of G possibly leads to two directed edges (i,j) and (j,i) in $\mathcal{C}$, with different capacity constraints. Thus the set of constrained edges $\mathcal{C}$ contains directed edges, limiting the directional flow through them.

Moreover, we assume that the constraints are feasible[6]. The solution will minimize the free energy objective function (Eq. 2) while satisfying these inequality constraints.

4.1 The Lagrange function in case of capacity constraints

From standard nonlinear optimization theory [9, 23, 41, 67], the Lagrange function is

$$\begin{aligned}
\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) \\
&\quad + \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\big(\bar{n}_{ij} - \sigma_{ij}\big) \\
&= \underbrace{\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy, } \phi(\mathrm{P})} + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) \\
&\quad + \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\Bigg(\underbrace{\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\,\eta\big((i,j)\in\wp\big)}_{\bar{n}_{ij}\,\text{(see Eq. 6)}} - \sigma_{ij}\Bigg)
\end{aligned} \tag{19}$$

where there is a Lagrange parameter μ associated to the sum to one constrain and a Lagrange parameter associated to each constrained edge. The Lagrange parameters $\{\lambda_{ij}\}$, $(i,j) \in \mathcal{C}$, are all non-negative in the case of inequality constraints and are stacked into the parameter vector $\boldsymbol{\lambda}$.

Note that, by inspecting (19) and from the convexity of the Kullback-Leibler divergence, the primal objective function to be minimized with respect to the discrete probabilities (which is similar to Eq. 2) is convex and the equality constraints are all linear. Therefore, the duality gap between the primal and dual problems is zero, which will be exploited for solving the problem (see, e.g., [9, 23, 41, 67]).

Now, the Lagrange function in Eq. 19 can be rearranged as

$$\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) = \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \underbrace{\left(\tilde{c}(\wp) + \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\,\eta\big((i,j)\in\wp\big)\right)}_{\text{augmented cost } \tilde{c}'(\wp) \text{ cumulated on path } \wp}$$

[6] If not, the duality gap in our algorithm will not converge to zero.

$$
\begin{aligned}
&+ T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) - \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\sigma_{ij} \\
&= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \sum_{(i,j)\in\mathcal{E}} \eta\big((i,j) \in \wp\big) \underbrace{\big(c_{ij} + \delta\big((i,j) \in \mathcal{C}\big)\, \lambda_{ij}\big)}_{\text{augmented costs } c'_{ij}} \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) - \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\sigma_{ij} \\
&= \underbrace{\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\tilde{c}'(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy based on augmented costs, } \phi'(\mathrm{P})} + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) \\
&\quad - \sum_{(i,j)\in\mathcal{C}} \lambda_{ij}\sigma_{ij}
\end{aligned}
\tag{20}
$$

where the symbol $\delta\big((i,j) \in \mathcal{C}\big)$ is defined as 1 when edge $(i,j) \in \mathcal{C}$ and 0 otherwise. Note that we used $\tilde{c}(\wp) = \sum_{(i,j)\in\mathcal{E}} \eta\big((i,j) \in \wp\big)\, c_{ij}$ (Eq. 12) computing the total cost along path $\wp$.

During this derivation, we observed that the costs c_{ij} can be redefined into **augmented costs** integrating the additional "virtual" costs needed for satisfying the constraints, and provided by the Lagrange parameters,

$$
c'_{ij} = \begin{cases} c_{ij} + \lambda_{ij} & \text{when edge } (i,j) \in \mathcal{C} \\ c_{ij} & \text{otherwise} \end{cases}
\tag{21}
$$

and $\mathbf{C}' = (c'_{ij})$ will be the matrix containing these augmented costs. Thus the Lagrange parameters have an interpretation similar to the dual variables in linear programming; they represent the extra cost to pay associated to each edge in order to satisfy the constraints [9, 23, 41, 67, 90]. This is also common in many network flow problems [1, 27].

Let $\phi'(\mathrm{P})$ be the free energy obtained in Eq. 20 from these augmented costs (Eq. 21). We now turn to the problem of finding the Lagrange parameters λ_{ij} by exploiting Lagrangian duality.

4.2 Exploiting Lagrangian duality

In this subsection, we will take advantage of the fact that, in the formulation of the problem, the Lagrange dual function and its gradient are easy to compute; see for instance [49] for similar arguments in the context of supervised classification. Indeed, as the objective function is strictly convex, all the equality constraints are linear and the support set for the path probabilities is convex, it is known that there is only one global minimum and the duality gap is zero provided that the problem is feasible[7] [9, 12, 23, 41, 67]. The optimum is a saddle point of the Lagrange function and a common optimization procedure (often called the Arrow-Hurwicz-Uzawa algorithm [5, 67]) consists in sequentially (i) solving the primal (finding the optimal probability distribution) while considering the Lagrange parameters as fixed and then (ii) maximizing the

[7] Recall that we assume that the problem is feasible; see the discussion at the end of Subsection 4.3.

dual (which is concave) with respect to the Lagrange parameters, until convergence. This procedure converges to the global minimum – see the mentioned references.

In our context, this provides the following steps [41], which are iterated until convergence,

$$\begin{cases} \mathscr{L}(\boldsymbol{\lambda}) = \mathscr{L}(\mathrm{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) = \underset{\{\mathrm{P}(\wp)\}_{\wp \in \mathcal{P}_{st}}}{\text{minimize}} \mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) & \text{(compute the dual function)} \\ \boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \mathscr{L}(\boldsymbol{\lambda}) & \text{(maximize the dual function)} \\ \boldsymbol{\lambda} = \boldsymbol{\lambda}^* \end{cases} \tag{22}$$

and this is the procedure that will be followed, where the dual function will be maximized through a simple gradient ascent procedure.

Computing the dual function

For computing the dual function $\mathscr{L}(\mathrm{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda})$ in Eq. 22, we first have to find the optimal probability distribution P^* in terms of the Lagrange parameters. We thus have to compute the minimum of Eq. 20 for a constant $\boldsymbol{\lambda}$. But this Lagrange function (Eq. 20) is identical to the Lagrange function associated to the standard RSP optimization problem (Eq. 2), except that the costs c_{ij} are replaced by the augmented costs c'_{ij}, and the introduction of the last, additional, term, which does not depend on the probability distribution $\mathrm{P}(\cdot)$. Therefore, the probability distribution minimizing Eq. 20 is a Gibbs-Boltzmann distribution of the form of Eq. 3, but now depending on the augmented costs instead of the original costs.

Then, we replace the probability distribution $\mathrm{P}(\cdot)$ in $\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda})$ by the optimal Gibbs-Boltzmann distribution $\mathrm{P}^*(\cdot)$ depending on the augmented costs and thus also on the Lagrange parameters. From the result of Eq. 4, the obtained dual function (Eq. 22) is

$$\mathscr{L}(\boldsymbol{\lambda}) = \mathscr{L}(\mathrm{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) = -T \log \mathcal{Z}' - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \tag{23}$$

In this equation, $\mathcal{Z}'$ is the partition function computed from the augmented costs $\mathbf{C}'$ and thus depends on $\boldsymbol{\lambda}$. We now need to maximize this dual function in function of these Lagrange parameters.

Maximizing the dual function

Let us now compute the gradient of the dual function with respect to the non-negative λ_{ij} for edges $(i,j) \in \mathcal{C}$. This maximization of the dual function can be done, e.g., by using the simple method developed by Rockafellar (see [79], Eqs. 10 and 12). From $-T\,\partial \log \mathcal{Z} / \partial c_{ij} = \bar{n}_{ij}$ (Eq. 6), the gradient of the dual Lagrange function (Eq. 23), for $(i,j) \in \mathcal{C}$, is simply $\partial \mathscr{L}(\boldsymbol{\lambda}) / \partial \lambda_{ij} = \partial(-T \log \mathcal{Z}' - \sum_{(k,l) \in \mathcal{C}} \lambda_{kl} \sigma_{kl}) / \partial \lambda_{ij} = \bar{n}_{ij} - \sigma_{ij}$, where we used the definition of the augmented costs in Eq. 21 as well as Eq. 6. It can be observed that we simply recover the expressions for the capacity constraints; this is actually a standard result when dealing with maximum entropy problems (see, e.g., [49, 51]).

For computing the Lagrange parameters, we thus follow [79] who proposed the following gradient-based updating rule[8]

$$\lambda_{ij} \leftarrow \max\left(\lambda_{ij} + \alpha(\bar{n}_{ij} - \sigma_{ij}), 0\right) \text{ for all } (i,j) \in \mathcal{C} \tag{24}$$

[8] Note that [79] used 2α for the gradient step.

which is guaranteed to converge in the concave case, as far as α is positive, is not too large, and the problem is feasible [79]. The update is iterated until convergence. Of course, we could have used other, more sophisticated and more efficient, optimization techniques (see, e.g., [9, 12, 41, 71]), but this simple procedure worked well for all our tests on small to medium graphs. The parameter α had to be tuned manually for each different data set, but this did not create any difficulty.

4.3 The resulting algorithm and discussion

The resulting algorithm is presented in Algorithm 2. It computes the optimal policy (Eq. 8) minimizing the objective function (Eq. 2) while satisfying the inequality constraints (Eq. 18). This optimal policy guides the random walker to the target state with a trade-off between exploitation and exploration monitored by temperature parameter T. The different steps of the procedure are the following:

- Initialize the Lagrange parameters to 0 and the augmented costs to the original edge costs $\mathbf{C}$.
- Then, iterate the following steps until convergence:
  - The elements of the fundamental matrix that are needed are recomputed from the current augmented costs (Eq. 5) by solving two systems of linear equations.
  - The expected number of passages through each edge (edge flows) is computed (Eq. 6).
  - The Lagrange parameters and the augmented costs are updated (Eqs. 24, 21).
- Compute and return the optimal policy (transition probabilities) according to Eq. 8.

Because two systems of n linear equations need to be solved at each iteration, its complexity is of order $\mathcal{O}(kn^3)$, depending on the number of iterations k needed for convergence, which is unknown in advance. However, for sparse graphs, the complexity could be reduced by taking advantage of the recent effort for developing special numerical methods for solving sparse systems of linear equations [85]. Indeed, for undirected graphs with a symmetric cost matrix, the auxiliary matrix $\mathbf{W}$ is similar to a symmetric diagonally dominant matrix for which fast solvers might be applied. This interesting avenue is left for further work.

Note also that lower bounds on the expected flow have also been considered. However, in this case, the resulting virtual costs (Lagrange parameters) can become negative in order to augment the flow in the edge. This sometimes lead to numerical instabilities when some costs become negative and negative cycles appear. One quick fix is to prohibit negative costs but then, in some situations, the problem becomes unfeasible. Therefore, we decided to leave the study of lower-bounded capacities on flows for further work.

Moreover, it was assumed that the problem is feasible, which can be difficult to check in practice. Indeed, the model injects a unit flow in the network so that the max flow through the network must at least be equal to one. This means that we cannot blindly assign capacities because, in the case where the problem is not feasible, the

Algorithm 2 Randomized shortest paths with capacity constraints.

Input:

– A weighted undirected graph G containing n nodes. Node s will be the source node and node t the absorbing, killing, target node.
– The $n \times n$ reference transition matrix $\mathbf{P}_{\mathrm{ref}}$ of the natural random walk on G.
– The $n \times n$ cost matrix $\mathbf{C}$ associated to G defining non-negative costs of transitions. An infinite cost is associated to missing links.
– The set of constrained edges $\mathcal{C}$.
– The set of non-negative capacities on flows, $\{\sigma_{ij}\}$, defined on the set of constrained edges, $(i,j) \in \mathcal{C}$.
– The inverse temperature parameter $\theta > 0$.
– The gradient ascent step $\alpha > 0$.

Output:

– The $n \times n$ randomized policy provided by the transition matrix $\mathbf{P}^*$, defining a biased random walk on G satisfying the constraints.

$\boldsymbol{\lambda} \leftarrow \mathbf{0}$ ▷ initialize the $n \times n$ Lagrange parameters vector
$\mathbf{C}' \leftarrow \mathbf{C}$ ▷ initialize the augmented costs matrix
Set row t of matrix $\mathbf{P}_{\mathrm{ref}}$ to $\mathbf{0}^{\mathrm{T}}$ ▷ target node t is made absorbing and killing
repeat ▷ main iteration loop
    $\mathbf{W} \leftarrow \mathbf{P}_{\mathrm{ref}} \circ \exp[-\theta \mathbf{C}']$ ▷ update $\mathbf{W}$ matrix (elementwise exponential and multiplication $\circ$)
    Solve $(\mathbf{I} - \mathbf{W})\mathbf{z}_t = \mathbf{e}_t$ ▷ backward variables $\mathbf{z}_t$ (column t of the fundamental matrix $\mathbf{Z}$) with elements z_{it}
    Solve $(\mathbf{I} - \mathbf{W})^{\mathrm{T}}\mathbf{z}_s = \mathbf{e}_s$ ▷ forward variables $\mathbf{z}_s$ (row s of the fundamental matrix $\mathbf{Z}$ viewed as a column vector) with elements z_{si}
    $\mathbf{N} \leftarrow \frac{\mathbf{Diag}(\mathbf{z}_s)\mathbf{W}\mathbf{Diag}(\mathbf{z}_t)}{z_{st}}$ ▷ compute the expected number of passages in each edge
    for all $(i,j) \in \mathcal{C}$ **do** ▷ gradient ascend: update all quantities associated to constrained edges
        $\lambda_{ij} \leftarrow \max\left(\lambda_{ij} + \alpha(\bar{n}_{ij} - \sigma_{ij}), 0\right)$ ▷ update Lagrange parameters
        $c'_{ij} \leftarrow c_{ij} + \lambda_{ij}$ ▷ update augmented costs
    end for
until convergence
$\mathbf{P}^* \leftarrow (\mathbf{Diag}(\mathbf{z}_t))^{-1}\mathbf{W}\mathbf{Diag}(\mathbf{z}_t)$ ▷ compute optimal policy
return $\mathbf{P}^*$

algorithm does not converge. One way to check the overall capacity of the network is to run a standard max-flow algorithm [1, 8, 27, 31, 40, 50, 58].

5 Dealing with net flow capacity constraints

Let us now consider the case where the capacities $\sigma_{ij} > 0$ with $(i,j) \in \mathcal{C}$ are defined on *net flows* instead of raw flows. It is therefore assumed in this section that the original graph is undirected and that the adjacency matrix as well as the cost matrix are symmetric. In this situation (see Eq. 9 and its discussion), the constraints operate on the net flows instead of the raw flows,

$$j_{ij} = \max\big((\bar{n}_{ij} - \bar{n}_{ji}), 0\big) \leq \sigma_{ij},$$
$$\text{or equivalently both} \begin{cases} \bar{n}_{ij} - \bar{n}_{ji} \leq \sigma_{ij} \\ \bar{n}_{ji} - \bar{n}_{ij} \leq \sigma_{ij} \end{cases} \quad \text{for each edge } (i,j) \in \mathcal{C} \tag{25}$$

In the net flows setting, we further consider that σ_{ji} is always defined when there exists a capacity constraint σ_{ij} and that $\sigma_{ji} = \sigma_{ij}$ (symmetry). Then, in this last equation, only one of the two net flows is positive so that the constraint only operates in this direction of the flow: the second constraint is automatically satisfied. However, we formulate the constraints in both directions since we do not know a priori which one is

potentially active. To summarize, if the set of constraints nodes $\mathcal{C}$ contains edge (i,j), it also necessarily contains its reciprocal (j,i) (they come by pair) with the same capacity value, $\sigma_{ji} = \sigma_{ij}$. We now turn to the derivation of the Lagrange function and the algorithm.

5.1 The Lagrange function in case of net flow capacity constraints

The Lagrange function then becomes, after re-arranging the terms like in Eq. 20 and using $\bar{n}_{ij} = \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp)\, \eta\big((i,j) \in \wp\big)$,

$$\begin{aligned}
\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) \\
&\quad + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \left[\bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}\right] \\
&= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \left(\tilde{c}(\wp) + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \left(\eta\big((i,j) \in \wp\big) - \eta\big((j,i) \in \wp\big)\right)\right) \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij}
\end{aligned}$$

Now, from the symmetry of edges (edges are present by pairs; for each $(i,j) \in \mathcal{C}$: $(j,i) \in \mathcal{C}$), we deduce $\sum_{(i,j) \in \mathcal{C}} \lambda_{ij}\, \eta\big((j,i) \in \wp\big) = \sum_{(j,i) \in \mathcal{C}} \lambda_{ij}$
$\eta\big((j,i) \in \wp\big) = \sum_{(i,j) \in \mathcal{C}} \lambda_{ji}\, \eta\big((i,j) \in \wp\big)$. Injecting this result in the previous equation and proceeding in the same way as in Eq. 20 provides

$$\begin{aligned}
\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \underbrace{\left(\tilde{c}(\wp) + \sum_{(i,j) \in \mathcal{C}} \big(\lambda_{ij} - \lambda_{ji}\big)\, \eta\big((i,j) \in \wp\big)\right)}_{\text{augmented cost } \tilde{c}'(\wp) \text{ cumulated on path } \wp} \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \\
&= \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \sum_{(i,j) \in \mathcal{E}} \eta\big((i,j) \in \wp\big) \underbrace{\big(c_{ij} + \delta\big((i,j) \in \mathcal{C}\big)\, \big(\lambda_{ij} - \lambda_{ji}\big)\big)}_{\text{augmented costs } c'_{ij}} \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \\
&= \underbrace{\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \tilde{c}'(\wp) + T \sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy based on augmented costs, } \phi'(\mathrm{P})} + \mu\left(\sum_{\wp \in \mathcal{P}_{st}} \mathrm{P}(\wp) - 1\right) \\
&\quad - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij}
\end{aligned} \tag{26}$$

which has exactly the same form as in the raw flow case (see Eq. 20) – only the definition of the augmented costs differs in the two expressions. Indeed, as before, the costs

c_{ij} can be redefined into *augmented* costs,

$$c'_{ij} = \begin{cases} c_{ij} + \lambda_{ij} - \lambda_{ji} & \text{when edge } (i,j) \in \mathcal{C} \\ c_{ij} & \text{otherwise} \end{cases} \tag{27}$$

and we must stress that we require that the constraints in Eq. 25 are symmetric and come by pair.

By following the same reasoning as in the previous section (see Eq. 23 and 24), it can immediately be observed that the dual function has the same form and is provided by Eq. 23.

5.2 The resulting algorithm and discussion

In the net flow context, by proceedings in the same way as in previous section (see Subsection 4.2), the gradient of the dual Lagrange function (Eq. 26) for augmented costs provided by Eq. 27 is $d\mathscr{L}(\boldsymbol{\lambda})/d\lambda_{ij} = (\partial\mathscr{L}(\boldsymbol{\lambda})/\partial c'_{ij})(\partial c'_{ij}/\partial\lambda_{ij}) + (\partial\mathscr{L}(\boldsymbol{\lambda})/\partial c'_{ji})(\partial c'_{ji}/\partial\lambda_{ij}) + \partial\mathscr{L}(\boldsymbol{\lambda})/\partial\lambda_{ij} = \bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}$. From this last result, we can easily derive the update of the Lagrange parameters,

$$\lambda_{ij} \leftarrow \max\left(\lambda_{ij} + \alpha(\bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}), 0\right) \text{ for all } (i,j) \in \mathcal{C} \tag{28}$$

The Algorithm 2 is easy to adapt in order to consider symmetric net flow capacity constraints ($\sigma_{ij} = \sigma_{ji}$): only lines 10 and 11 must be modified according to Eqs. 27 and 28.

In practice, we observed that the net flow constraint algorithm is slower to converge, and more sensitive to the gradient step, than simple flow constraints; in other words, the problems looks harder to solve. This is especially the case for small values of θ, where the process behaves more like a random walker. Indeed, in that situation, the addition of capacity constraints seems to be partly in conflict with the objective of walking randomly and moving back and forth. One way to handle this issue [26] would be to first define a directed acyclic graph (DAG, deduced, e.g., from the electrical potential on nodes when imposing a $+1$ potential at source node and a 0 potential at target node, followed by a topological sort). Then, a RSP problem with simple capacity constraints is solved on this DAG. This has the additional advantage that it is much more efficient (we avoid cycles and can use dynamic programming for computing the RSP solution) and it should scale to larger graphs. This extension is left for further work.

6 Experiments

In this section, we first present two illustrative examples of the use of capacity constraints on the edge flows of a graph as well as a brief study of the scalability of the net flow Algorithm 1. Then, we evaluate the net flow RSP on unsupervised classification tasks and compare its results to other state-of-the-art graph node distances. We have to stress that our goal here is not to propose new node clustering algorithms outperforming state-of-the-art techniques. Rather, the aim is to investigate if the net flow RSP model is able to capture the community structure of networks in an accurate way, compared to other dissimilarity measures between nodes.

6.1 Illustrative examples

First example The first example illustrates the expected number of visits to nodes (see Eq. 7) over the RSP paths distribution between one source node s and one target node t on a 20×20 grid, in two different situations obtained after running the Algorithm 2. Nodes are linked to their neighbors[9] with a unit affinity and a unit cost. In the first situation, we do not set any capacity constraint and the expected number of visits is represented in Fig. 1(a). As expected, it follows a trajectory close to the diagonal of the grid, representing the shortest paths between the source node and the target node.

In the second situation, we placed two obstacles (walls) by constraining the capacities of all the edges linking the nodes represented in red in Fig. 1(b) to 0.01. As can be observed in 1(c), this time, the expected number of visits to nodes no longer concentrates around the diagonal, but follows as close as possible the obstacles. This trajectory represents the least-cost paths between the source node and the target node avoiding the low-capacity obstacles.

Second example Our second illustrative example is taken from [76] and makes a link between the RSP with net flow capacity constraints and the maximum flow problem. Its aim is to show that the Algorithm 2 could be used to compute the maximum flow (which takes a value of 12 in this example) between the source node s and the target node t in the undirected graph G presented in Fig. 2. Each edge of this graph has a unit cost and affinity. Note that, in practice, because the RSP model assumes a unit input flow, all capacities are scaled[10] so that the value of the maximum flow after scaling lies between 0 and 1. The maximum flow of G is then obtained by the reverse transformation.

To this end, we add a directed edge from s to t with infinite capacity and an edge cost of 10 to the original graph. This new edge introduces a "shortcut" allowing the passage of all the overflow of the graph, at the price of a higher cost. Theoretically, our algorithm will avoid going trough this shortcut as much as possible because it has a high cost compared to the other edges in the graph. Therefore, it should try to maximize the flow that travels through the original network before using this shortcut edge.

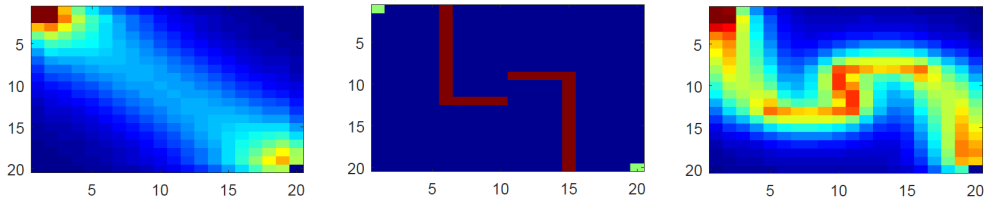
Fig. 3 shows the evolution of the flow between the nodes $\{f, g, h\}$ and t (the flow through the original graph), provided by Algorithm 2, and thus satisfying the capacity constraints, in function of the value of the θ parameter. As observed in Fig. 3, this flow through the original network reaches almost exactly the maximum flow value of 12 for all θ larger than 1. However, when θ is low (close to zero), the flows become more and more random, without considering costs any more (see Eq. 2). For that reason, part of the total flow goes through the shortcut, even if this is not optimal in terms of expected cost. This explains the reduction of the flow through the original network when θ is close to zero.

6.2 Scalability experiment

In this subsection, we study the scalability of the Algorithm 1, which computes the full net flow randomized shortest paths dissimilarity matrix, through a small experiment. To evaluate the time-complexity, we generate 120 graphs with the benchmark algorithm of Lancichinetti, Fortunato and Radicchi (LFR) [59]. More precisely, we create 10 graphs

[9] Only horizontal and vertical neighbors are considered (no diagonal edge).

[10] We divide by a graph cut (an upper bound on the the min-cut), for instance the cut between nodes $\{f, g, h\}$ and t, 22 in our example.



(a) Expected number of visits to nodes (see Eq. 6), with $\theta = 0.05$ and $\alpha = 40$. Red represents higher expected number of visits and blue lower ones.

(b) Introduction of capacity constraints on edges belonging to the obstacles (walls in red). Source node s (upper left corner) and target node t (lower right corner) are in green.

(c) Expected number of visits to nodes (see Eq. 6) after introducing capacity constraints on the walls, for an intermediate $\theta = 0.05$ and gradient step $\alpha = 40$.

Figure 1: Illustrative example of the capacity constrained RSP on a 2D grid.

for each of the following sizes of {50, 100, 150, 200, 250, 300, 500, 1000, 1500, 2000, 2500, 3000} nodes. Moreover, for each size, among the 10 graphs, we changed the mixing parameter value (μ) each 2 graphs between all theses values {0.1, 0.2, 0.3, 0.4, 0.5}.

We then run the Algorithm 1 on each of theses graphs and report the average computation time in seconds over the 10 graphs for each different size (in terms of number of nodes and number of computed distances) on the Fig. 4. All results were obtained with Matlab (version R2017a) running on an Intel Core i7-8750H CPU@4.10GHz and 32 GB of RAM.

As previously mentioned, the time-complexity of this algorithm is $\mathcal{O}(n^3 + mn^2)$ with n being the number of nodes and m being the number of edges. Clearly, although applicable on medium-size graphs, it can be observed that the algorithm does not scale well on large graphs in its present form.

6.3 Nodes clustering experiments

In this subsection, we present an application of the **Net Flow Randomized Shortest Paths** (nRSP) in a graph nodes clustering context. Note that a methodology very close to [83] is used – see this paper for details.

Experimental Setup

Baselines As part of the node clustering experiment, four dissimilarity matrices between nodes as well as five kernels on a graph will be used as baselines to assess our nRSP method.

Baseline dissimilarities

- The **Free Energy** distance (FE) and the **Randomized Shortest Paths Dissimilarity** (RSP) depending on an inverse temperature parameter $\theta = 1/T$. Already presented earlier, these methods have been shown to perform well in a node clustering context [83] as well as in semi-supervised classification tasks [34].

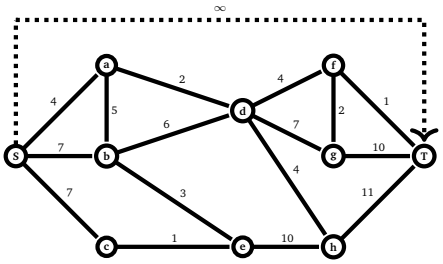


Figure 2: A small undirected graph composed of 8 nodes [76]. The values on the edges represent capacities; moreover all edge costs are set to 1, except the added shortcut edge whose cost is 10. The maximum flow between the source node s and the target node t is 12, and is equal to the min-cut between nodes $\{a, b, c\}$ and $\{d, e\}$.

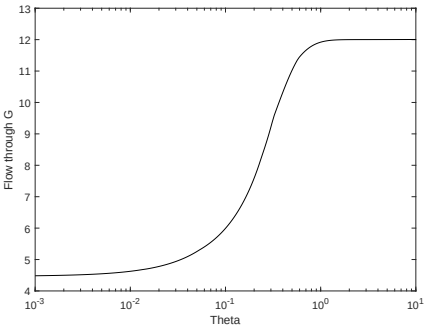


Figure 3: Evolution of the flow between nodes $\{f, g, h\}$ and t satisfying the capacity constraints (total flow in the original network G, without the shortcut edge), provided by Algorithm 2, in function of the θ parameter, with a gradient step $\alpha = 1/\theta$. The intersection between the curve and the y-axis when $\theta \to 0^+$ is 4.467. Note that the x-axis is scaled logarithmically.

- The **Logarithmic Forest** distance (LF). Introduced in [15], it relies on the matrix forest theorem [18] and defines a family of distances interpolating (up to a scaling factor) between the shortest-path distance and the resistance distance [55], depending on a parameter α.
- The **Shortest Path** distance (SP). The distance corresponds to the cost along the least cost path between two nodes i and j, derived from the cost matrix $\mathbf{C}$.

These dissimilarity matrices are transformed into inner products (kernel matrices) by classical multidimensional scaling (see later).

Baseline kernels on a graph

- The **Neumann** kernel [81] (Katz), initially proposed in [52] as a method of computing similarities, it is defined as $\mathbf{K} = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I}$. The α parameter has to be chosen positive and smaller than the inverse of the spectral radium of $\mathbf{A}$, $\rho(\mathbf{A}) = \max_i(|\lambda_i|)$.
- The **Logarithmic Communicability** kernel (lCom) proposed in [47] as the logarithmic version of the exponential diffusion kernel [57], and also called the communicability measure [30], $\mathbf{K} = \ln(\exp(t\mathbf{A}))$, $t > 0$, where $\exp$ is the matrix exponential.
- The **Sigmoid Commute Time** kernel (SCT). Proposed in [95], it is obtained by applying a sigmoid transform [81] on the commute time kernel [32]. An alternative version, the **Sigmoid corrected Commute Time** kernel (SCCT), based on the correction of the commute time suggested in [91], is also used as part of the experiments. For both methods, the parameter α controls the sharpness of the sigmoid.

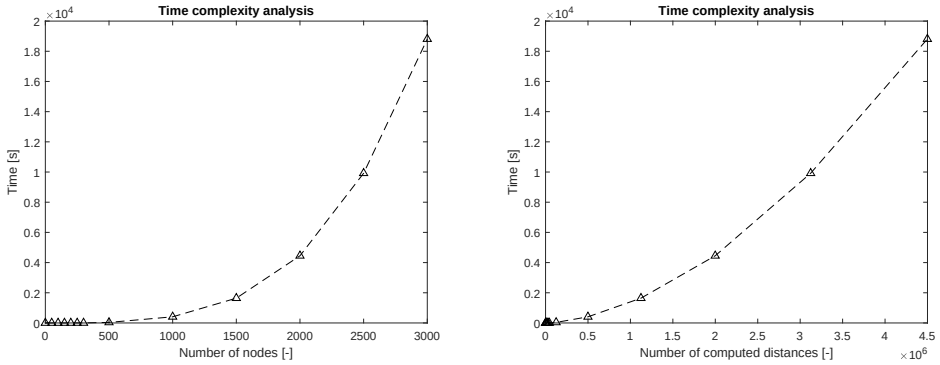


Figure 4: Average computation time in seconds over the 10 LFR graphs for each different network size, in terms of number of nodes (left) and number of computed distances (right).

Dataset Name	Clusters	Nodes	Edges
Dolphin_2	2	62	159
Dolphin_4	4	62	159
Football	12	115	613
LFR1	3	600	6142
LFR2	6	600	4807
LFR3	6	600	5233
Newsgroup_2_1	2	400	33854
Newsgroup_2_2	2	398	21480
Newsgroup_2_3	2	399	36527
Newsgroup_3_1	3	600	70591
Newsgroup_3_2	3	598	68201
Newsgroup_3_3	3	595	64169
Newsgroup_5_1	5	998	176962
Newsgroup_5_2	5	999	164452
Newsgroup_5_3	5	997	155618
Political books	3	105	441
Zachary	2	34	78

Table 1: Datasets used in our experiments.

In addition, the **Modularity matrix Q** is used as last baseline (Q), and is computed by $\mathbf{Q} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^{\mathrm{T}}}{\mathrm{vol}}$ where $\mathbf{d}$ contains the node degrees and the constant vol is the volume of the graph (see, e.g., [69]). Recall that modularity is a measure of the quality of a partition of the nodes (a set of communities). Here, the kernel k-means is executed directly on matrix $\mathbf{Q}$.

Datasets A collection of 17 datasets is investigated for the experimental comparisons of the dissimilarity measures. For each dataset, costs are computed as the reciprocal of affinities, $c_{ij} = 1/a_{ij}$, like in electrical networks. The collection includes Zachary's karate club [98], the Dolphin datasets [64, 65], the Football dataset [39], the Political books[11], three LFR benchmarks [59] and 9 Newsgroup datasets [60, 96]. The list of datasets along with their main characteristics are presented in Table 1.

[11] Collected by V. Krebs and labelled by M. Newman, to the best of our knowledge, this dataset is not published, but available for download at `http://www-personal.umich.edu/~mejn/netdata/`.

Algorithm	Parameter values
FE RSP nRSP	$\theta = (0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1,\ 0.5,\ 1,\ 3,\ 5,\ 10,\ 15,\ 20)$
LF	$\alpha = (0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1,\ 0.5,\ 1,\ 3,\ 5,\ 10,\ 15,\ 20)$
Katz	$\alpha = (0.05,\ 0.10, \ldots,\ 0.95) \times (\rho(\mathbf{A})^{-1})$
lCom	$t = (0.01,\ 0.02,\ 0.05,\ 0.1,\ 0.2,\ 0.5,\ 1,\ 2,\ 5,\ 10)$
SCT SCCT	$\alpha = (5,\ 10,\ 15,\ \ldots,\ 50)$

Table 2: Parameter range for the investigated methods.

Evaluation metrics Each partition provided by an investigated clustering technique will be assessed by comparing it with the "observed partition" of the dataset. Two criteria will be used to evaluate the similarity between both partitions.

- The **Normalized Mutual Information (NMI)** [35, 86] between two partitions $\mathcal{U}$ and $\mathcal{V}$ is computed by dividing the mutual information [22] between the two partitions by the average of the respective entropy of $\mathcal{U}$ and $\mathcal{V}$ (see also [66]).
- The **Adjusted Rand Index (ARI)** [46] is an extension of the Rand Index (RI) [77], which measures the degree of matching between two partitions. The RI presents the drawback of not showing a constant expected value when working with random partitions. On the contrary, the ARI has an expected value of 0 and a maximum value of 1.

Experimental methodology

The experiments rely on a kernel k-means introduced in [96]. For the dissimilarities, the experimental methodology is similar to the one used in [83]. For each given dataset, the dissimilarity matrix $\mathbf{D}$ obtained by the different methods providing dissimilarities[12] is transformed into a kernel $\mathbf{K}$ (a inner product matrix) using classical multidimensional scaling [11]. If the resulting kernel is not positive semi-definite, we simply set the negative eigenvalues to zero when computing the kernel.

The kernel k-means is run 30 times (trials) on $\mathbf{K}$ with different initializations. The NMI and ARI are then computed for the partition maximizing the modularity among these 30 trials. This operation is repeated 30 times (leading to a total of 900 runs of the k-means) to obtain the average modularity, NMI and ARI scores over these 30 repetitions for a given method (dissimilarity matrix), with a given value of its parameter (for instance, θ in the case of methods based on RSP), on a specific dataset. Finally, the reported NMI and ARI score for each method and dataset is the average (over the 30 repetitions) for the parameter value showing the largest modularity. Thus, modularity (which is unsupervised) is used as a metrics to tune the parameters of the algorithms [84]. The parameters that are tuned are the θ for the nRSP, the FE and the RSP, the α for the LF and for Katz, the t for the lCom, and the α for the sigmoid transform of the SCT and SCCT. The values tested for these parameter are listed in the Table 2.

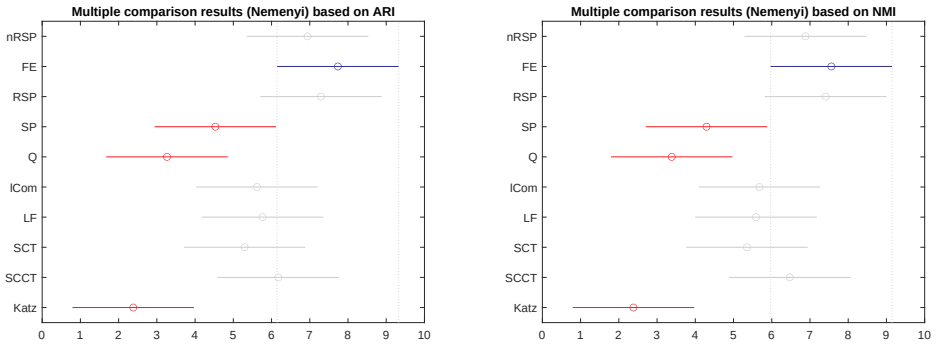


Figure 5: Mean ranks and 95% Nemenyi confidence intervals of the tested methods across the 17 datasets for the ARI (left) and the NMI (right) measures. The larger the rank, the best.

Method	nRSP	FE	RSP	SP	Q	lCom	LF	SCT	SCCT	Katz
nRSP		0.389	0.497	**0.017**	**0.004**	0.241	0.188	0.151	0.561	**0.001**
FE	0.277		0.626	**0.015**	**0.002**	0.068	**0.025**	**0.015**	0.127	**0.001**
RSP	0.588	0.153		**0.011**	**0.002**	**0.025**	**0.02**	**0.026**	0.194	**0.001**
SP	**0.019**	**0.015**	**0.031**		0.246	**0.028**	0.093	0.062	**0.011**	**0.004**
Q	**0.002**	**0.001**	**0.001**	0.055		**0.006**	**0.005**	**0.004**	**0.002**	0.076
lCom	0.241	**0.042**	0.078	0.068	**0.004**		0.855	1.000	0.241	**0.001**
LF	0.151	**0.011**	0.078	0.062	**0.003**	0.952		0.934	0.217	**0.001**
SCT	0.055	**0.012**	**0.030**	0.163	**0.004**	0.359	0.359		**0.009**	**0.001**
SCCT	0.252	**0.048**	0.153	**0.044**	**0.002**	0.808	0.426	**0.004**		**0.001**
Katz	**0.001**	**0.001**	**0.001**	**0.002**	0.062	**0.001**	**0.001**	**0.001**	**0.001**	

Table 3: The p-values provided by a pairwise Wilcoxon signed-rank test, for the NMI in the upper right triangle and the ARI in the lower left.

Experimental results

The different methods are assessed globally using the same method as in [83] based on a non-parametric Friedman-Nemenyi test [25]. Additionally, a Wilcoxon signed-rank test [94, 99] is performed pairwise to measure the significance of the difference in the algorithms performance.

The results for the Friedman-Nemenyi are summarized in Fig. 5. Additionally, the Table 3 contains the pairwise p-values for the Wilcoxon signed-rank test. More specifically, the upper-right side of the diagonal of the matrix contains the p-values when considering NMI and the lower-left contains the p-values when using the ARI instead.

We can observe on the Nemenyi plot that the three leading methods appear to be the nRSP, the FE and the RSP. More specifically, the FE is the best method on the investigated datasets, and in this setup. Based on the Wilcoxon test using ARI, the FE performs significantly better than all the other methods, at a $\alpha = 0.05$ level, except for the RSP and nRSP. Note, however, that, for the NMI score, the difference between the FE and the lCom as well as the SCCT are not significant.

The introduced method (nRSP) obtains results comparable to the RSP and the FE for both the ARI and the NMI measures. None of these differences are significant after performing a Wilcoxon signed rank test ($\alpha = 0.05$). Although not the best overall, the introduced nRSP method proves to be competitive with respect to the FE and the

[12]For methods directly providing a kernel, the obtained kernel matrix is directly used.

RSP which, in turn, performed best in a similar but more extensive node clustering comparison [83]. It can also be observed that the nRSP and the standard RSP obtain very similar performances, which was somewhat expected because both distances are based on the same framework.

7 Conclusion

In this work, we developed two extensions of the randomized shortest paths (RSP) formalism. The first extension introduces an algorithm for computing the expected net costs between all pairs of node, by considering the *net flows* between nodes instead of the raw flows. This quantity is called the net flow RSP dissimilarity; it quantifies the level of accessibility between nodes and serves as a dissimilarity measure. The second contribution shows how to deal with capacity constraints on edges for both raw and net flows in the RSP formalism. An algorithm solving the constrained problem is developed.

As already discussed in [44], the originality of the RSP model, in comparison with competing methods, lies in the fact that it adopts a *path-based formalism* with relative entropy regularization at the level of paths probabilities. That is, the quantities of interest are defined on the set of all possible paths (or walks) between two nodes of the network. Another interesting feature is that most quantities of interest can be computed in closed form by using standard matrix operations.

These contributions extend the scope of the RSP formalism, which basically defines a model of movement, or spread, through the network. Indeed, most of the traditional models are based on two common paradigms about the transfer of information, or more generally the movement, occurring in the network: an optimal behavior based on least cost paths and a random behavior based on a random walk on the graph. Contrarily to these standard models, the RSP interpolates between a pure random walk on the graph and an optimal behavior based on shortest paths [34, 54, 80]. It depends on a temperature parameter allowing to monitor the amount of randomness of the trajectories. This acknowledges the fact that in many practical cases the (random) walker on the graph is neither completely rational, nor completely stochastic.

Experimental comparisons on clustering tasks showed that the net flow RSP is competitive in comparison with other state-the-art baseline methods. This indicates that the model is able to capture the cluster structure of networks in an accurate way. Actually, the net flow RSP obtained results comparable to the simple RSP and the free energy dissimilarities, based on the same framework.

In conclusion, the contributions of this paper should enlarge the range of possible applications of the RSP formalism. Indeed, many problems related to the spread of information in a network involve capacity constraints on edges. Moreover, in many real cases, it can be argued that a model based on unidirectional net flows is more realistic than raw flows going back and forth.

Concerning further work, we are interested in applications of the proposed models to operations research problems. For instance, it has been shown that solving optimal transport problems with entropic regularization can be significantly more efficient than using standard linear programming methods in some situations [24, 42, 44]. It would therefore be interesting to compare the RSP solution (with entropy regularization) to more standard algorithms solving minimum cost flow problems with capacity constraints [1, 8, 27, 31, 40, 50, 58].

We also plan to explore two ideas that could improve the scalability of the proposed algorithms on sparse graphs. The first aims to explore recent work devoted to the development of special numerical methods for solving sparse systems of linear equations [85] (see the discussion in Subsection 4.3). The second would be to extract a directed acyclic graph (DAG) from the original s-t graph by, for instance, computing the electrical current flow between source node s and target node t [4]. We saw in Subsection 3.3 that the electrical current defines a DAG. Then, we should be able to compute efficiently the optimal policy (and therefore also directed flows) on this DAG by using the Bellman-Ford-like expression computing the free energy directed distance developed in [34]. It would also be interesting to explore capacity constraints on DAG, as already discussed in Subsection 5.2.

Acknowledgements

This work was partially supported by the Immediate and the Brufence projects funded by InnovIris (Brussels Region), as well as former projects funded by the Walloon region, Belgium. We thank these institutions for giving us the opportunity to conduct both fundamental and applied research. Moreover, we thank Professor Bernard Fortz (Université Libre de Bruxelles) for his suggestions. Finally, we also appreciated the remarks of the anonymous reviewers which helped us to improve significantly the manuscript.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice Hall, 1993.

[2] T. Akamatsu. Cyclic flows, Markov process and stochastic traffic assignment. *Transportation Research B*, 30(5):369–386, 1996.

[3] M. Alamgir and U. von Luxburg. Phase transition in the family of p-resistances. In *Advances in Neural Information Processing Systems 24: Proceedings of the NIPS 2011 conference*, pages 379–387. MIT Press, 2011.

[4] R. Anantharaman, K. Hall, V. Shah, and A. Edelman. Circuitscape in julia: High performance connectivity modelling to support conservation decisions. *Manuscript preprint available at arXiv:1906.03542*, 2019.

[5] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and non-linear programming*. Stanford University Press, 1958.

[6] A. L. Barabasi. *Network science*. Cambridge University Press, 2016.

[7] F. Bavaud and G. Guex. Interpolating between random walks and shortest paths: A path functional approach. In K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, and C. Guéret, editors, *Proceedings of the 4th International Conference on Social Informatics (SocInfo '12)*, volume 7710 of *Lecture Notes in Computer Science*, pages 68–81. Springer, 2012.

[8] D. P. Bertsekas. *Network optimization: Continuous and discrete models*. Athena Scientific, 1998.

[9] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 2nd edition, 1999.

[10] B. Bollobas. *Modern graph theory*. Springer, 2nd edition, 1998.

[11] I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 1997.

[12] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[13] U. Brandes and T. Erlebach, editors. *Network analysis: Methodological foundations*. Springer, 2005.

[14] U. Brandes and D. Fleischer. Centrality measures based on current flow. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05)*, pages 533–544, 2005.

[15] P. Chebotarev. A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics*, 159(5):295–302, 2011.

[16] P. Chebotarev. The walk distances in graphs. *Discrete Applied Mathematics*, 160(10–11):1484–1500, 2012.

[17] P. Chebotarev. Studying new classes of graph metrics. In F. Nielsen and F. Barbaresco, editors, *Proceedings of the 1st International Conference on Geometric Science of Information (GSI '13)*, volume 8085 of *Lecture Notes in Computer Science*, pages 207–214. Springer, 2013.

[18] P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514, 1997.

[19] F. Chung and L. Lu. *Complex graphs and networks*. American Mathematical Society, 2006.

[20] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 3rd edition, 2009.

[21] S. Courtain, B. Lebichot, I. Kivimaki, and M. Saerens. Graph-based fraud detection with the free energy distance. In *Proceedings of the 8th International Conference on Complex Networks and their Applications (Complex Networks 2019)*, pages 40–52. Springer, 2019.

[22] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 2nd edition, 2006.

[23] J. Culioli. *Introduction a l'optimisation*. Ellipses, 2012.

[24] M. Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: Proceedings of the NIPS '13 conference*, pages 2292–2300. MIT Press, 2013.

[25] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[26] R. Dial. A probabilistic multipath assignment model that obviates path enumeration. *Transportation Research*, 5:83–111, 1971.

[27] A. Dolan and J. Aldous. *Networks and algorithms: An introductory approach*. Wiley, 1993.

[28] P. G. Doyle and J. L. Snell. *Random walks and electric networks*. The Mathematical Association of America, 1984.

[29] E. Estrada. *The structure of complex networks*. Oxford University Press, 2012.

[30] E. Estrada and N. Hatano. Communicability in complex networks. *Physical Review E*, 77(3):036111, 2008.

[31] L. Ford and D. Fulkerson. *Flows in networks*. RAND Corporation, 1962.

[32] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[33] F. Fouss, M. Saerens, and M. Shimbo. *Algorithms and models for network data and link analysis*. Cambridge University Press, 2016.

[34] K. Francoisse, I. Kivimäki, A. Mantrach, F. Rossi, and M. Saerens. A bag-of-paths framework for network data analysis. *Neural Networks*, 90:90–111, 2017.

[35] A. L. Fred and A. K. Jain. Robust data clustering. In *Proceedings of the 2003 IEEE International Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, volume 2, pages 128–133, 2003.

[36] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[37] S. García-Díez, F. Fouss, M. Shimbo, and M. Saerens. A sum-over-paths extension of edit distances accounting for all sequence alignments. *Pattern Recognition*, 44(6):1172–1182, 2011.

[38] S. García-Díez, E. Vandenbussche, and M. Saerens. A continuous-state version of discrete randomized shortest-paths. In *Proceedings of the 50th IEEE International Conference on Decision and Control (CDC '11)*, pages 6570–6577, 2011.

[39] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the USA*, 99(12):7821–7826, 2002.

[40] M. Gondran and M. Minoux. *Graphs and algorithms*. Wiley, 1984.

[41] I. Griva, S. Nash, and A. Sofer. *Linear and nonlinear optimization*. SIAM, 2nd edition, 2008.

[42] G. Guex. Interpolating between random walks and optimal transportation routes: Flow with multiple sources and targets. *Physica A: Statistical Mechanics and its Applications*, 450:264–277, 2016.

[43] G. Guex and F. Bavaud. Flow-based dissimilarities: shortest path, commute time, max-flow and free energy. In B. Lausen, S. Krolak-Schwerdt, and M. Bohmer, editors, *Data science, learning by latent structures, and knowledge discovery*, volume 1564 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 101–111. Springer, 2015.

[44] G. Guex, I. Kivimäki, and M. Saerens. Randomized optimal transport on a graph: framework and new distance measures. *Network Science*, 7(1):88–122, 2019.

[45] M. Herbster and G. Lever. Predicting the labelling of a graph via minimum p-seminorm interpolation. In *Proceedings of the 22nd Conference on Learning Theory (COLT '09)*, pages 18–21, 2009.

[46] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[47] V. Ivashkin and P. Chebotarev. Do logarithmic proximity measures outperform plain ones in graph clustering? In *International Conference on Network Analysis*, pages 87–105. Springer, 2016.

[48] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

[49] T. Jebara. *Machine learning: discriminative and generative*. Kluwer Academic Publishers, 2004.

[50] D. Jungnickel. *Graphs, networks, and algorithms*. Springer, 4th edition, 2013.

[51] J. N. Kapur. *Maximum-entropy models in science and engineering*. Wiley, 1989.

[52] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[53] I. Kivimäki, B. Lebichot, J. Saramäki, and M. Saerens. Two betweenness centrality measures based on randomized shortest paths. *Scientific Reports*, 6:srep19668, 2016.

[54] I. Kivimäki, M. Shimbo, and M. Saerens. Developments in the theory of randomized shortest paths with a comparison of graph node distances. *Physica A: Statistical Mechanics and its Applications*, 393:600–616, 2014.

[55] D. J. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.

[56] E. D. Kolaczyk. *Statistical analysis of network data: Methods and models*. Springer Series in Statistics. Springer, 2009.

[57] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning (ICML '02)*, pages 315–322, 2002.

[58] B. Korte and J. Vygen. *Combinatorial optimization. Theory and algorithms, 6th ed*. Springer, 2018.

[59] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[60] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Machine Learning Conference (ML95)*, pages 331–339, 1995.

[61] T. Lewis. *Network science*. Wiley, 2009.

[62] Y. Li, Z.-L. Zhang, and D. Boley. The routing continuum from shortest-path to all-path: A unifying theory. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pages 847–856. IEEE Computer Society, 2011.

[63] Y. Li, Z.-L. Zhang, and D. Boley. From shortest-path to all-path: The routing continuum theory and its applications. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1745–1755, 2013.

[64] D. Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(suppl_2):S186–S188, 2003.

[65] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[66] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[67] M. Minoux. *Programmation mathematique, 2nd ed*. Lavoisier, 2008.

[68] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, 2005.

[69] M. E. J. Newman. *Networks: An introduction*. Oxford University Press, 2nd edition, 2018.

[70] C. Ngyen and H. Mamitsuka. New resistance distances with global information on large graphs. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS '16)*, pages 639–647, 2016.

[71] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2nd edition, 2006.

[72] J. R. Norris. *Markov chains*. Cambridge University Press, 1997.

[73] M. Panzacchi, B. Van Moorter, O. Strand, M. Saerens, I. Kivimäki, C. St Clair, I. Herfindal, and L. Boitani. Predicting the continuum between corridors and barriers to animal movements using step selection functions and randomized shortest paths. *Journal of Animal Ecology*, 85(1):32–42, 2016.

[74] L. Peliti. *Statistical mechanics in a nutshell*. Princeton University Press, 2011.

[75] M. Pióro and D. Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[76] W. L. Price. *Graphs and networks: an introduction*. London Butterworths, 1971.

[77] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[78] L. E. Reichl. *A modern course in statistical physics*. Wiley, 2nd edition, 1998.

[79] R. Rockafellar. The multiplier method of Hestenes and Powell appplied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[80] M. Saerens, Y. Achbany, F. Fouss, and L. Yen. Randomized shortest-path problems: Two related models. *Neural Computation*, 21(8):2363–2404, 2009.

[81] B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

[82] T. Silva and L. Zhao. *Machine learning in complex networks*. Springer, 2016.

[83] F. Sommer, F. Fouss, and M. Saerens. Comparison of graph node distances on clustering tasks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2016)*. *Lecture Notes in Computer Science*, volume 9886, pages 192–201, 2016. Springer.

[84] F. Sommer, F. Fouss, and M. Saerens. Modularity-driven kernel k-means for community detection. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2017)*. *Lecture Notes in Computer Science*, volume 10614, pages 423–433, 2017. Springer.

[85] D. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the annual ACM Symposium on the Theory of Computing (STOC 2004)*, volume 4, 2004.

[86] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[87] H. M. Taylor and S. Karlin. *An introduction to stochastic modeling*. Academic Press, 3rd edition, 1998.

[88] M. Thelwall. *Link analysis: An information science approach*. Elsevier, 2004.

[89] E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 1369–1375. MIT Press, 2007.

[90] S. Vajda. *Mathematical programming*. Addison-Weslay, 1961.

[91] U. von Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the commute distance. In *Advances in Neural Information Processing Systems 23: Proceedings of the NIPS '10 Conference*, pages 2622–2630, 2010.

[92] U. von Luxburg, A. Radl, and M. Hein. Hitting and commute times in large random neighborhood graphs. *Journal of Machine Learning Research*, 15(1):1751–1798, 2014.

[93] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge University Press, 1994.

[94] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[95] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. Graph nodes clustering based on the commute-time kernel. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '07)*, volume 4426 of *Lecture Notes in Artificial Intelligence*, pages 1037–1045. Springer, 2007.

[96] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data & Knowledge Engineering*, 68(3):338–361, 2009.

[97] L. Yen, A. Mantrach, M. Shimbo, and M. Saerens. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pages 785–793, 2008.

[98] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[99] J. Zar. *Biostatistical analysis*. Prentice Hall, 4th edition, 1999.
