Eur. Phys. J. B (2022) 95:89
https://doi.org/10.1140/epjb/s10051-022-00336-7

THE EUROPEAN PHYSICAL JOURNAL B

Regular Article - Statistical and Nonlinear Physics

Belief propagation for supply networks: efficient clustering of their factor graphs

Tim Ritmeester[a] and Hildegard Meyer-Ortmanns[b]

School of Science, Jacobs University Bremen, P.O. Box 750561, 28725 Bremen, Germany

Received 6 December 2021 / Accepted 20 April 2022 / Published online 25 May 2022

Abstract. We consider belief propagation (BP) as an efficient and scalable tool for state estimation and optimization problems in supply networks such as power grids. BP algorithms make use of factor graph representations, whose assignment to the problem of interest is not unique. It depends on the state variables and their mutual interdependencies. Many short loops in factor graphs may impede the accuracy of BP. We propose a systematic way to cluster loops of naively assigned factor graphs such that the resulting transformed factor graphs have no additional loops as compared to the original network. They guarantee an accurate performance of BP with only slightly increased computational effort, as we demonstrate by a concrete and realistic implementation for power grids. The method outperforms existing alternatives to handle the loops. We point to other applications to supply networks such as gas-pipeline or other flow networks that share the structure of constraints in the form of analogues to Kirchhoff's laws. Whenever small and abundant loops in factor graphs are systematically generated by constraints between variables in the original network, our factor-graph assignment in BP complements other approaches. It provides a fast and reliable algorithm to perform marginalization in tasks like state determination, estimation, or optimization issues in supply networks.

1 Introduction

Belief propagation (BP) is an algorithm that is known from statistical physics [1,2], computer science, artificial intelligence and information science (for a review see, for example, [3]). It runs also under the name of 'message passing'. In some cases, BP provides an exact rearrangement of the original calculational objective, while in general it calculates a mean-field approximation [1–4] to that. The BP method has two established main advantages over traditional algorithms (such as least-squares and quasi-Newton methods [5,6]). It is fast even for large networks, as the computation time scales linearly in the system size, and it is robust against large differences in the input parameters. Its robustness avoids convergence issues associated with the traditional approach ([7]). These properties make it uniquely suitable for dealing with large datasets and for frequent and large-scale network analyses (if possible online), which are increasingly required for supply networks.

In electric power grids, BP is in principle applicable to optimization problems and state estimation (in Sect. 3 we give a concrete implementation for state estimation). Optimization problems refer to cost-efficient and low-risk performance under diverse sources of uncertainties. State estimation is the procedure of using measurement data to infer an estimate of state variables such as power flows and phase angles as accurately as possible. State estimation is important to convert system measurements into reliable information on the true network state and to ensure the stability of operation. In [8–10], BP-based algorithms were found to outperform traditional (least-squares) approaches to state estimation. In particular, the speed and robustness of the BP-based algorithms enable state estimation in real-time [8,11,12] and statistical analyses of large networks [9,10], even if data are partially missing [10]. Thus, state estimation combined with BP amounts to an important step in view of improving supervisory control and planning decisions in running power grids if a fast online estimation of the true state of the grid is required.

A second range of applications of BP are gas networks that we consider in some more detail in the appendix. Natural gas is still one of the important energy resources worldwide. A reliable and efficient operation of gas-pipeline networks becomes increasingly important due to the liberalization of the European gas market. In [5], the impact of injecting alternative gas supplies at different locations is studied to facilitate

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1140/epjb/s10051-022-00336-7.

[a] e-mail: t.ritmeester@jacobs-university.de
[b] e-mail: h.ortmanns@jacobs-university.de (corresponding author)

decisions on the allowable amount and composition also of alternative gas sources such as hydrogen and biogas.

BP has been applied to further supply networks [13–16], other than power or gas networks. The authors of [13] and [14] used BP to identify faults and contamination sources in water networks. In [15], a BP algorithm was shown to effectively optimize public transport for urban planning and telecommunication networks, while [16] applied BP to a generic nonlinear resource allocation problem.

For a given network with an associated graph, the BP algorithm makes use of an additional auxiliary graph called the factor graph. The novel contribution of this paper is the assignment of an appropriate factor graph to supply networks. If the very factor graph has a tree structure, BP is known to be exact [17]. In general, BP implements an approximation corresponding to Bethe mean-field theory [2–4]. Practically, BP is accurate if there are only a few short loops in the factor graph.

A naively assigned factor graph directly reflects the basic variables on the original network and their mutual dependencies. In the supply networks that we consider in this paper these interdependencies result from Kirchhoff's laws, corresponding to the conservation of flow at the vertices of the network (first law) and the constraint that the drop in variables like the voltage or pressure sums up to zero around elementary loops (second law). In general, these laws impose nonlinear constraints on the flows, and these constraints are responsible for additional loops in a naively assigned factor graph, even if the original network is a tree. It is these loops that we want to avoid by a suitable assignment of factor graphs. The loops are numerous as the constraints between the variables are omnipresent in the network. The mathematical structure of these constraints is found in many supply networks, so that our proposed new algorithm applies to all of them. Although our method succeeds in avoiding the many additional small loops in the factor graphs which result from constraints as mentioned before, it does not address the possible challenges which result from loops in the original supply networks. In the concrete test cases that we consider, the resulting algorithm still shows an excellent performance without addressing these loops. In cases where these loops do impede the performance of BP, our method should be combined with additional approaches. Several such improvements have been proposed and implemented, see, for example, [8,18–21].

Such a combination is possible because our method, which relies on a type of clustering (to be defined below), changes only the factor graph rather than the BP algorithm itself. In contrast to other clustering methods, we propose a systematic way of clustering factor graphs in terms of clusters which by construction depend only on a few variables. This way the price to pay for clustering remains moderate. The resulting method then improves the speed and convergence of the BP algorithm.

The paper is organized as follows. In Sect. 2, we describe the clustering procedure, assigning a factor graph which differs from a straightforward assignment, but prevents by construction additional loops in the factor graph assignment. In Sect. 3, we illustrate the application of the procedure with state estimation for the artificial IEEE-300 electrical grid. We discuss the accuracy and performance of the algorithm. In Sect. 4, we give an outlook to other applications of our algorithm to power grids. The conclusions are summarized in Sect. 5. In Appendix A, we point to further possible supply networks which share the essential structure of the equations; in particular, we work out the case of natural gas-pipeline networks with an example from the steady-state analysis of two realistic GasLib benchmark networks. Here our method enables the very applicability of BP, as with our method, BP converges exceedingly fast while BP with a naive factor graph assignment does not converge at all.

2 Assigning factor graphs to supply networks

We consider supply networks consisting of vertices and links and assume the following generic description:

- To each link in the supply network, a flow of some quantity is assigned which traverses the link; flows from vertex i to vertex j and vice-versa are denoted as f_{ij} and f_{ji}, respectively.
- Associated with each vertex i is a variable v_i; examples are the voltage in electric circuits, or the pressure in fluid networks.
- The flow f_{ij} through a link (ij) is determined by the variables v_i and v_j at the vertices at either end of the link (e.g., Ohms law in electric circuits). The flow is thus a function $f_{ij}(v_i, v_j)$ of the vertex variables v_i and v_j.
- Flows are conserved at each vertex: The sums of in and outgoing flows are equal. Denoting external injections into a vertex i as g_i, we thus have $g_i = \sum_{j\in N(i)} f_{ij}(v_i, v_j)$, where $N(i)$ is the set of vertices adjacent to vertex i. The injection can thus be written as a function $g_i(v_i, v_j : j \in N(i))$ of the vertex variables $\{v_i, v_j : j \in N(i)\}$.

The class of problems that we study on these supply networks is defined by the criterion that the state is determined in terms of a probability distribution $P(\boldsymbol{v})$, depending on the vertex variables $\boldsymbol{v}$ of the following form:

$$P(\boldsymbol{v}) = \prod_i H_i(v_i, v_j : j \in N(i)) \times \prod_{(ij)} H_{(ij)}(v_i, v_j), \tag{1}$$

where the products run over all vertices i and all links (ij) of the network, respectively, and $N(i)$ is the set of vertices j that are connected to i via a link. The function H_i depends on the variables $\{v_i, v_j : j \in N(i)\}$,

which can in particular incorporate a mutual dependence on v_i and on the injection $g_i(v_i, v_j : j \in N(i))$. The H_{ij} are functions depending on pairs of variables v_i, v_j, which may incorporate a dependence on the flows $f_{ij}(v_i, v_j)$ and $f_{ji}(v_j, v_i)$. In Sect. 3 and Appendix A we will study the state estimation of power grids and a network analysis of gas pipelines, where we will explicitly show that these problems reduce to the marginalization of a probability distribution of the form of Eq. 1.

Applications of Eq. 1. Although Eq. 1 may look rather specific, it is in fact a general formulation that comprises the following cases:

- Uncertainty in the state variables: The probability distribution may express the uncertainty in the values of the vertex variables v_i, the flows f_{ij} and/or the injections g_i. Such uncertainty may arise from fluctuating injections (for example, in renewable energy generation), or from uncertainties in the measurements of these quantities (as in the Bayesian state estimation problem studied in Sect. 3). (If the value of a quantity is certain, this can be incorporated as a delta function in the distribution. If nothing about a quantity is known, the associated factor is assigned a very high variance.)
- Optimization problems: For a given cost function $C(\boldsymbol{v})$, one considers the distribution

$$P_T(\boldsymbol{v}) \propto \exp(-C(\boldsymbol{v})/T)\,. \tag{2}$$

  In the limit $T \to 0$ the probability distribution peaks at the minimal costs $C(\boldsymbol{v})$. The exponential turns sums into products, such that the distribution of Eq. 1 incorporates a minimization of a sum of costs on the vertex variables, the flows and the injections. Constraints can be implemented by setting the cost function equal to infinity whenever the constraints are violated. It is possible to explicitly take this limit in the BP Eqs. 4–6, thereby converting them to a form that is more convenient for optimization (called the min-sum algorithm). We refer to [3,22] for details.
- Constraint satisfaction problems (i.e., finding configurations $\boldsymbol{v}$ that satisfy a number of constraints): These can be included by studying products of Dirac delta functions, where each delta function incorporates a constraint. Alternatively, a cost function is optimized that assigns a penalty to each violated constraint. We will use this option to analyze the steady state of gas-pipeline networks in Appendix A.
- Optimization under uncertainty: If costs need to be minimized in an inherently fluctuating environment [23], decisions on the production, for example, may lead to stochastic rather than deterministic costs. For power grids such a situation has been investigated in [24,25], where fluctuations are due to uncertain power injections by renewable resources.

If the distribution of Eq. 1 should be evaluated to gain insight into the probability of individual variables, it amounts to a marginalization of this joint probability distribution by summing or integrating over a subset of variables. This is the place where BP enters in the sense that the sums or integrals are performed in a very efficient way.

2.1 The choice of factor graphs

BP can be used to efficiently calculate marginals of probability distributions such as those of Eq. 1. It is convenient if BP makes use of a graphical representation of the probability distribution in terms of a factor graph. The factor graph is a bipartite graph, made of two types of nodes, variable nodes, represented by circles, and factor nodes, represented by squares. The assignment of variables and factors is not unique and a matter of convenience. The procedure of assigning a factor graph to any probability distribution $P(\boldsymbol{v})$ proceeds in the following steps:

1. Partition the vector $\boldsymbol{v}$ into new 'variables' $\{x_I\}$, where each x_I is a disjoint subset of $\boldsymbol{v}$ (i.e., possibly containing multiple v_i).
2. For each x_I, draw a circle. This defines a *variable node* of the factor graph. As a special case, the correspondence between variables on the original grid and the variable nodes on factor graphs may be one to one in a straightforward assignment (which we refer to as the naive assignment).
3. Define factors $W_a(\boldsymbol{x}_a)$ such that $P(\boldsymbol{v}) = \prod_a W_a(\boldsymbol{x}_a)$, where $\boldsymbol{x}_a$ are (in general overlapping) sets of some of the new variables $\{x_I\}$.
4. For each factor W_a, draw a square. This is a *factor node* of a factor graph.
5. Each factor $W_a(\boldsymbol{x}_a)$ depends on $\boldsymbol{x}_a$, which contains multiple variable nodes x_I. Draw an edge between the factor node $W_a(\boldsymbol{x}_a)$ and each variable node $x_I \in \boldsymbol{x}_a$ on which it depends.

Ambiguities in the assignment of the factor graph result from step (1) and (3).

Example of a straightforward choice of a factor graph. Let us give a concrete example, considering a simple building block of three vertices $\{1, 2, 3\}$ of a larger network, connected by links $(1, 2)$ and $(2, 3)$ (shown in Fig. 1a). The distribution (Eq. 1) we are interested in is then given by:

$$\begin{aligned} P(v_1, v_2, v_3) = &\big[H_1(v_1, v_2) \cdot H_2(v_1, v_2, v_3) \cdot H_3(v_2, v_3)\big] \\ &\times \big[H_{(1,2)}(v_1, v_2) \cdot H_{(2,3)}(v_2, v_3)\big]\,. \end{aligned} \tag{3}$$

Following the steps (1)–(4), the most straightforward way of assigning a factor graph to this distribution is to assign a variable node to each v_i, and a factor node to each H_i and each $H_{(ij)}$. The factor graph corresponding to the distribution (3) is shown in Fig. 1b.

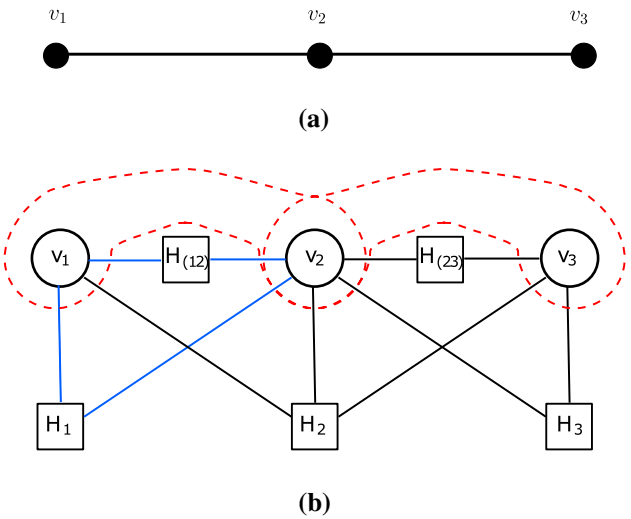

Fig. 1 **a** Simple building block of a supply network, which gives the probability distribution of Eq. 3. **b** The factor graph assigned in a straightforward way (Sect. 2.1) to Eq. 3, representing the simple network shown in (**a**). Note that the four elementary loops (one of them indicated in blue) would be absent if each H_i would depend only on v_i rather than on all $\{v_i, v_j : j \in N(i)\}$. The dotted lines indicate two clusters of vertices, which the approach described in Sect. 2.3 will use to eliminate the loops (after dealing with the overlap at vertex v_2)

In particular, it is important to note that this straightforward assignment has a number of loops (four in this case, one of them indicated in blue) though the original network (Fig. 1a) has none. Responsible for the loops are the factors H_i: Without these factors, the topology of the factor graph would directly reflect the topology of the original supply network (for each vertex it contains a variable node v_i, and for each link it contains a factor node H_{ij} connecting variables nodes v_i and v_j). The clustering applied to this factor graph (to be discussed later) refers to the variable nodes, in Fig. 1b, we indicate two clusters with dotted lines, overlapping in variable node v_2. In Fig. 2, we show how to deal with the overlaps. Moreover, if H_i would depend only on v_i (rather than on all $\{v_i, v_j : j \in N(i)\}$), the factor nodes corresponding to H_i would be leaf nodes and create no loops.

Number of extra loops in a naive assignment. The total number of extra loops can be determined as follows. In general, the total number of loops in a graph, here the factor graph, is given by (♯ connected components + ♯ edges − ♯ vertices) [26]. The naively assigned factor graph contains factor nodes H_i and H_{ij} and variable nodes v_i as well as edges connecting these nodes. As mentioned before, if we consider only the variable nodes, the factor nodes H_{ij} and the edges connecting them, the resulting structure directly reflects the topology of the original supply network, in particular it has the same amount of loops. The amount of extra loops in the factor graph (as compared to the supply network) can thus be calculated as (♯ additional edges due to factor node $\{H_i\}$ − ♯ additional nodes which are of type H_i). On the factor graph, each H_i must be connected by an edge to all variable nodes in $\{v_i, v_j : j \in N(i)\}$: The amount of extra edges is thus given by $\sum_i |v_i, v_j : j \in N(i)| = \sum_i (1 + |N(i)|)$. The number of extra nodes is simply $\sum_i 1$ (one factor node for each H_i). In total the number of extra loops in the factor graph is thus given by $\sum_i (1 + |N(i)|) - \sum_i 1 = \sum_i |N(i)| = 2 \cdot \sharp$ links. This means in the example of Fig. 1b that there are $2 \cdot 2 = 4$ extra loops (where the number of links can be found from Fig. 1a). The dependence of H_i on further variables from $N(i)$ due to ubiquitous constraints on all of the network variables thus leads to a proliferation of loops in the factor graph.

2.2 Sketch of the BP algorithm

To fix the notation, we summarize the basic steps of the BP algorithm. For a given factor graph with variable nodes $\{x_I\}$ and factor nodes $W_a(\boldsymbol{x}_a)$ (such that $P(\boldsymbol{x}) = \prod_a W_a(\boldsymbol{x}_a)$), BP can be used to calculate marginals $P_I(x_I) \equiv \int \prod_{J \neq I} \mathrm{d}x_J P(\boldsymbol{x})$ and $P_a(\boldsymbol{x}_a) \equiv \int \prod_{J: x_J \notin \boldsymbol{x}_a} \mathrm{d}x_J P(\boldsymbol{x})$. Here we will give the basic BP-algorithm, for which several extensions exist [8,18–21], as mentioned in the introduction. The steps of the BP-algorithm for the marginalization are the following:

Initialization. For each factor-variable pair (a, I) that is connected on the factor graph (that is, for which $x_I \in \boldsymbol{x}_a$), messages $\{m_{I\to a}(x_I), m_{a\to I}(x_I)\}$ are initialized uniformly: At time $t = 0$, the messages are set to $m^{t=0}_{I\to a}(x_I) \propto 1$ and $m^{t=0}_{a\to I}(x_I)$ $\propto 1$. The messages are functions of the variables, If the variables are discrete or if the factors are Gaussian (implying also Gaussian messages) the messages can be parameterized by a few real numbers. Otherwise one needs to find an approximation, such as a discretization of the messages [22,27] or the basis function expansions considered in [28]. In case of Gaussian distributions we initialize the messages with zero mean and large variance to generate a uniform distribution.

Updates. At each step t, we keep track of approximations $\{b^t_I(x_I)\}$ and $\{b^t_a(\boldsymbol{x}_a)\}$ (with I and a running over all variable and factor nodes respectively), which for large t are supposed to converge to the marginals of $P(\boldsymbol{x})$ according to $b_I(x_I) \to P_I(x_I)$ and $b_a(\boldsymbol{x}_a) \to P_a(\boldsymbol{x}_a)$. The approximations are given in terms of the messages $\{m^t_{I\to a}(x_I), m^t_{a\to I}(x_I)\}$, which are updated according to:

$$m^{t+1}_{J\to a}(x_J) = \prod_{b\neq a; x_J \in \boldsymbol{x}_b} m^t_{b\to J}(x_J) \tag{4}$$

$$m^{t+1}_{a\to I}(x_I) = \int \Big[\prod_{J\neq I; x_J \in \boldsymbol{x}_a} m^{t+1}_{J\to a}(x_J) \Big] \times \big[W_a(\boldsymbol{x}_a)\big] \times \prod_{J\neq I; x_J \in \boldsymbol{x}_a} \mathrm{d}\, x_J \,, \tag{5}$$

$$b_I^{t+1}(x_I) \propto \prod_{a:x_I \in \boldsymbol{x}_a} m_{a\to I}^{t+1}(x_I)\,. \tag{6}$$

$$b_a^{t+1}(\boldsymbol{x}_a) \propto \prod_{a:x_I \in \boldsymbol{x}_a} m_{I\to a}^{t+1}(x_I)\,. \tag{7}$$

Note that the notation $x_I \in \boldsymbol{x}_a$ means that variable node I and factor node a are connected on the factor graph. The updates are repeated until a reasonable stopping criterion is reached, for example, when $b_I^{t+1}(x_I) - b_I^t(x_I)$ or $b_a^{t+1}(\boldsymbol{x}_a) - b_a^t(\boldsymbol{x}_a)$ reach some desired tolerance.

Output. The sets $\{b_I(x_I)\}$ and $\{b_a(\boldsymbol{x}_a)\}$ then give the approximations of the marginals $\{P_I(x_I)\}$ and $\{P_a(\boldsymbol{x}_a)\}$, respectively, that we want to determine.

2.3 Systematic clustering of short loops in factor graphs

Importantly, in relation to our application to supply networks, we should distinguish between the topology of the original network such as the power grid and the topology of the associated factor graph. As we have seen in Sect. 2.1, even if the graphical representation of the original network is a tree, a naively assigned factor graph may unavoidably contain loops. Our statements below refer always to the topology of the factor graph.

Clustering is a method to find factor graph representations with a reduced number of loops by aggregating multiple variable nodes and/or factor nodes into a reduced set of variable or factor nodes. For variable nodes this corresponds to treating a subset of the variables as a new single variable node, while for factor nodes their clustering corresponds to multiplying factors together to obtain a new, aggregated factor. The catch in the choice of factor graphs is that the larger the subsets $\boldsymbol{x}_a$, the more difficult is the calculation of messages resulting from Eqs. 4–5. In the extreme case, where the whole distribution is clustered into a single factor, the algorithm simply returns the original marginalization problem $b_I(x_I) = \int P(\boldsymbol{x}) \prod_{J\neq I} \mathrm{d}x_J$, such that BP is exact but of no advantage anymore. Thus it is important to find a clustering that 1. guarantees a high accuracy, and 2. avoids difficulties in computing the messages via Eqs. 4–5.

2.3.1 Generating the clustered factor graph

In the clustered factor graph that we propose, we assign a variable node to each link of the original network. Thus, each of these variable nodes is a tuple consisting of the vertex variables at each end of that link. For a simple tree network, as in Fig. 2a, this is indicated in Fig. 2b. In Fig. 2b, the tuples of vertex variables are shown that are supposed to make up the variable nodes of the clustered factor graph. For each link (ij), the variable node at this link consists of the two vertex variables (v_i, v_j) at each end of the link. However, the variable nodes are overlapping in the sense that each vertex variable v_i is contained in multiple variable nodes: The variable node (v_i, v_j) for each link $(ij) : j \in N(i)$ connected to vertex i contains the variable v_i. When copying the vertex variables to decouple the clusters (Fig. 2c), one has to compensate the copying by introducing δ-constraints which enforce that all copies of a given vertex variable remain equal. This copying procedure is a generalization of the Shafer–Shenoy algorithm [29] (in the sense that the Shafer–Shenoy algorithm requires the so-called 'running intersection property', while the copying procedure here does not). Mathematically this amounts to an identity operation, but it allows us to improve the performance of BP by eliminating loops from the factor graph. Connecting the clustered variable nodes to the factor nodes then leads to a tree-like factor graph as in Fig. 2d.

The number of clusters a given vertex variable belongs to, is equal to the number of links connected to the vertex. We thus have to make a copy for each of those links. Thus we define a probability distribution which depends on all the copies of v-variables, while the original v-variables are integrated out by delta constraints:

$$P_c(\boldsymbol{v}^c) \equiv \int P(\boldsymbol{v}) \prod_i \Big(\Big[\prod_{j\in N(i)} \delta(v_{ij}^c - v_i) \Big] \mathrm{d}v_i \Big)\,,$$

where $\boldsymbol{v}^c \equiv \{v_{ij}^c : j \in N(i)\}$ is the set of all copies of vertex variables v_i kept for those links (ij) that are connected to vertex i (thus, v_{ij}^c is a copy of v_i). Stated differently, each link connecting vertices i and j keeps a copy of v_i and v_j at its ends, and the delta function constrains the copies to remain equal to the original variables. Figure 3 shows the multiplication of vertex i by three further copies, carried by the incident links toward vertices j, l, k. From the definitions it is clear that calculating marginals in P_c is equivalent to calculating marginals in the original distribution $P(\boldsymbol{v})$. The advantage is that P_c contains no extra loops, it can be represented as a loop-free factor graph for which BP is exact if the supply network itself has no loops. Writing out P_c explicitly, we get

$$P_c(\boldsymbol{v}^c) = \Big[\prod_i \int \mathrm{d}v_i \Big\{ H_i(v_i, v_{ji}^c : j \in N(i)) \prod_{j\in N(i)} \delta(v_{ij}^c - v_i) \Big\} \Big] \Big[\prod_{(i,j)} H_{(ij)}\big(v_{ij}^c, v_{ji}^c\big) \Big]\,. \tag{8}$$

In the first product, we have explicitly kept the v-integration to include the $\delta-$constraints, while the v-integration has been carried out in the second product. Note that each H_{ij} depends on a single variable on the new factor graph, while H_i may depend on several variables on the factor graph.

Now we are ready to define the clustered factor graph. As anticipated already in Fig. 2a–d, we are able to assign a factor graph to P_c which has the same amount of loops as the original network.

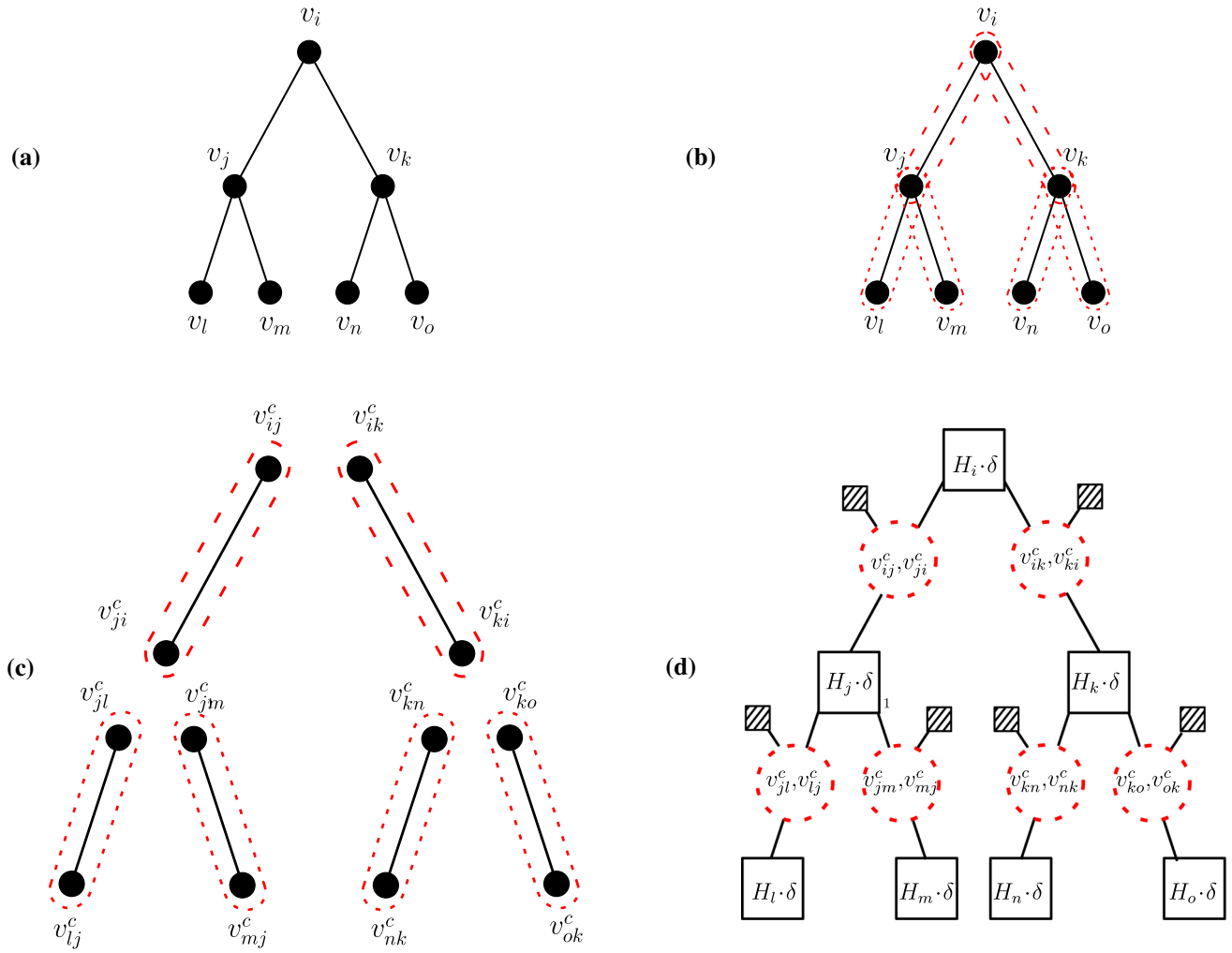

Fig. 2 **a** A simple tree network. **b** The striped ellipses indicate the clusters proposed as new variable nodes: Each cluster is assigned to a link and consists of the vertex variables at each end of that link. **c** Copying process of the vertex variable nodes on the factor graph to avoid overlapping. **d** Adding factor nodes H_i between those variable nodes that enter H_i of Eq. 8, including the δ-constraints, and attaching leaf nodes (dashed squares) (one for each variable node) that represent the H_{ij}-terms of Eq. 8. Note that (**d**) has the same tree structure as (**a**), where the links of (**a**) with attached vertices become variable nodes in (**d**) and the vertices of (**a**) have their counterpart in H_i, δ-factor nodes with edges correspondingly attached. The links of the original network furthermore give an additional leaf node on the factor graph

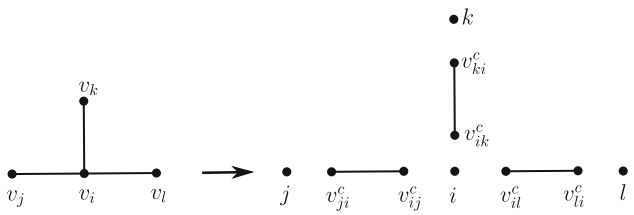

Fig. 3 Copying procedure of the vertex variables as already used in Fig. 2. The variable v_i is copied to v^c_{ij}, v^c_{ik} and v^c_{il}, one copy for each incident link

Variable nodes. As variable nodes we use the tuples (v^c_{ij}, v^c_{ji}), one for each link (ij) of the original network (Fig. 2a).

Factor nodes. Each vertex i of the original network gives a factor node $\int \mathrm{d}v_i \left\{ H_i(v_i, v^c_{ji} : j \in N(i)) \prod_{j\in N(i)} \delta(v^c_{ij} - v_i) \right\}$, abbreviated as $H_i \cdot \delta$, see Fig. 2d. Each link (ij) of the original network gives a factor node $H_{(ij)}(v^c_{ij}, v^c_{ji})$.

The new factor graph is illustrated in Fig. 2d. The new variable nodes partition $\boldsymbol{v}^c$. Multiplying all the factors together gives the full distribution $P_c(\boldsymbol{v}_c)$ of Eq. 8. This is thus a valid factor graph. Note that here the new variable nodes (v^c_{ij}, v^c_{ji}) are assigned to the links rather than to the vertices of the original network. The reason is that the copied variables enter always in pairs, one vertex variable for each end of the link. The original variables can be retrieved from the corresponding

copied variables at any of the links entering the vertex from the various directions.

Claim: The resulting factor graph has exactly as many loops as the original supply network. The factor at each vertex i depends on $\{(v^c_{ij}, v^c_{ji}) : j \in N(i)\}$; thus on the factor graph we connect the factor nodes $(H_i \cdot \delta)$ at each vertex i to all variables (v^c_{ij}, v^c_{ji}) representing links incident to i. Given only variables (v^c_{ij}, v^c_{ji}) and factors at the vertices of the supply network, the factor graph thus has the same topology as the supply network, with a variable for each link and a factor for each vertex, see Fig. 2d. The other factors, associated with each link, depend on only one variable node each and are thus leaf nodes. The topology of the factor graph thus equals the topology of the supply network with the addition of some leaf nodes. In particular, it therefore has the same number of loops. This is graphically seen in Fig. 2d.

2.3.2 The flow-only factor graph as a special case

Before comparing the results of BP on the straightforward and the clustered factor graphs (Sects. 2.1 and 2.3.1), let us first discuss a special case of our clustering. The clustering simplifies if the whole distribution (Eq. 1) can be written in terms of flows $f_{ij}(v_i, v_j)$ rather than vertex variables:

$$P_f(\boldsymbol{f}) \equiv \Big[\prod_i H_i\Big(\sum_{j\in N(i)} f_{ij}\Big)\Big] \times \Big[\prod_{(i,j)} H_{ij}(f_{ij})\Big], \quad (9)$$

and additionally we assume that $f_{ij} = -f_{ji}$ to implement flow conservation. If one would explicitly keep the dependence on $\boldsymbol{v}$, writing the flows as $f_{ij}(v_i, v_j)$, one sees that the distribution still corresponds to a distribution of the form of Eq. 1. If we assign a factor graph to this distribution P_f in a straightforward way, using as variables f_{ij} $(= -f_{ji})$ for each (ij), and factors H_i for each vertex and H_{ij} for each link, the factor graph has exactly the same topology as the clustered factor graph we proposed. The advantage is that the expressions are simpler due to the absent dependence on v_i, v_j.

Not all distributions of the form of Eq. 1 can be written as in Eq. 9 in terms of flow only. For example, the distribution of Eq. 9 cannot constrain the flows to Kirchhoff's second law, so one requirement is that the original network is a tree. In this case, the flows are exclusively determined by the conservation law $g_i = \sum_{j\in N(i)} f_{ij}$, and do not receive extra constraints from the vertex variables $\boldsymbol{v}$. Another requirement is that the choice of the distribution of Eq. 1 does not involve the vertex variables $\boldsymbol{v}$ directly, but only indirectly through $\{g_i\}$ and $\{f_{ij}\}$. If the network does not satisfy these requirements, the flow-only distribution (Eq. 9) can still be used as an approximation, where we ignore the constraints that the vertex variables $\boldsymbol{v}$ induce on the flows. In the case of an electric power grid, this corresponds to ignoring Kirchhoff's second law, taking only power flow conservation into account. We call this approximation the "flow-only"-approximation, which we considered in [10]. We will further compare this approximation to our clustering in Sect. 3.

2.3.3 Comparison of factor graphs

We defined three different ways of constructing a factor graph from a probability distribution of the form of Eq. 1:

- the naively assigned factor graph (Sect. 2.1), from here on denoted as F_v,
- the factor graph clustered according to our procedure of Sect. 2.3.1, denoted as F_c,
- the flow-only factor graph (Sect. 2.3.2), completely ignoring the vertex variables (according to Eq. 9), denoted as F_f.

Figure 4 compares how these factor graphs look like concretely for the simple network of Fig. 1. F_c and F_f have the same amount of loops as the original network, in this case none. F_v has an increased number of loops, where the difference in the amount of loops is given by $\sum_i |N(i)|$, as argued before in Sect. 2.1. These factor graphs lead to three different sets of BP equations (following Eqs. 4–7).

3 A concrete implementation: Bayesian inference for state estimation in power grids

The previous section contains the general formulation of our clustering method; it gives an improved BP for supply networks by considerably reducing the number of loops in the factor graph. In this section we will use a realistic implementation for state estimation in power grids as a concrete test case to show the improvement over the naive assignment of factor graphs. The goal in state estimation problems is to reliably retrieve the underlying state of the system in terms of its variables. The variables are correlated by power flow equations and in principle accessible to measurements, but these are affected by errors, therefore some care is needed to reliably estimate the state.

In AC-power grids, the power flow equations restrict the measured values for active (reactive) power injections at vertex i, the active (reactive) flows between vertex i and vertex j, as well as the voltages, given the conductances and susceptances of the transmission lines. The DC-approximation to the AC-equations, which we consider in more detail, corresponds to a linearization of the AC-equations which is justified for high-voltage grids when angle differences are small and Ohmic power losses can be neglected. In this case, the DC-approximated AC-equations read

$$f_{ij} = B_{ij}(\theta_i - \theta_j)\,, \qquad g_i = \sum_{j\in N(i)} f_{ij} \qquad (10)$$

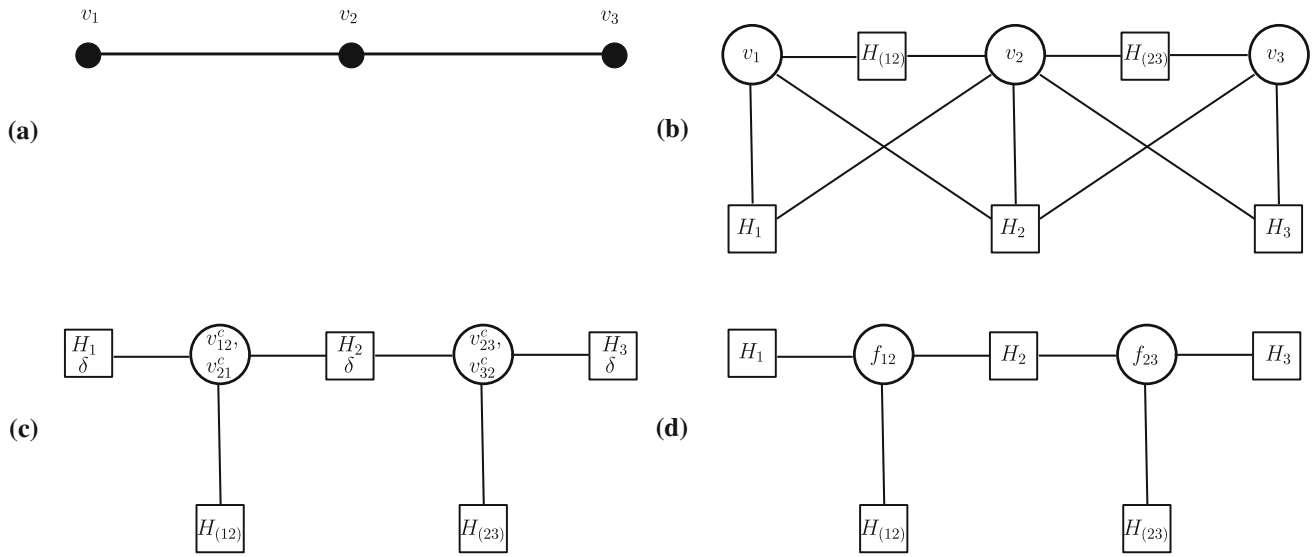

Fig. 4 **a** A simple network, which gives the probability distribution of Eq. 3. **b** The factor graph assigned in the straightforward way (Sect. 2.1) to Eq. 3. **c** The factor graph assigned to the same distribution by our clustering procedure. **d** The factor graph assigned to the flow-only approximation of the same distribution, as described in Sect. 2.3.2

with θ_i the phase angles at the vertices i, g_i representing active power injections at vertices i, f_{ij} the active power flow. The susceptances B_{ij} are provided in the data sets of the considered grids and $N(i)$ denotes the set of vertices directly connected to vertex i by a transmission line. In [10] we ignored the first of Eq. 10 and considered only the flows as variables characterizing the state of the system. This corresponds to the flow-only approximation discussed in Sect. 2.3.2. In principle, the injections g_i, the flows f_{ij} and more recently also the angles (via phasor measurement units (PMUs)) are accessible to direct measurements. However, one can do better than taking these direct measurements for the state estimation and use Bayes' theorem in the form of

$$P(\boldsymbol{x}|\boldsymbol{z}) = \frac{P(\boldsymbol{z}|\boldsymbol{x})P_{\mathrm{pr}}(\boldsymbol{x})}{P_{\mathrm{pr}}(\boldsymbol{z})}\,, \tag{11}$$

where $P(\boldsymbol{z}|\boldsymbol{x})$ is the probability that a state $\boldsymbol{x}$ would give data $\boldsymbol{z}$ and $P_{\mathrm{pr}}(\boldsymbol{x})$ is the prior belief that the state is $\boldsymbol{x}$. If no prior knowledge exists about $\boldsymbol{x}$, it will be chosen as a uniform distribution. The prior belief over data, $P_{\mathrm{pr}}(\boldsymbol{z})$, is independent of $\boldsymbol{x}$; hence it only provides a normalization constant and is irrelevant for our purposes. Bayes' theorem is then used for state estimation and real-time processing of measurement data.

The injections $\{g_i\}$ and flows $\{f_{ij}\} = \{-f_{ji}\}$ are restricted by the DC-approximated power-flow equations (Eqs. 10), so we can write f_{ij} and g_i as functions of the angles, $f_{ij}(\theta_i, \theta_j)$ and $g_i(\theta_i, \theta_j : j \in N(i))$. Denoting a given measurement as z_a (always a scalar), and the subset of variables which enter the measurement as $\boldsymbol{\theta_a}$, we then assume $z_a = f(\boldsymbol{\theta}_a) + \xi_a$, where ξ_a are generated independently from Gaussian distributions with known standard deviations σ_a. The function f represents the scalar quantity which is measured, specified as a function of the angle variables. If we consider direct measurements of state variables (g_i, θ_i or f_{ij}), then we have $f(\boldsymbol{\theta_a})$ equal to $g_i(\theta_i, \theta_j : j \in N(i))$, $\mathbb{1}(\theta_i)$ or $f_{ij}(\theta_i, \theta_j)$, respectively.

Measurement errors of this form thus give $P(z_a|\boldsymbol{\theta}_a) \sim N(f(\boldsymbol{\theta_a}), \sigma_a)$, which contribute to the joint probability distribution $P(\boldsymbol{z}|\boldsymbol{\theta}) = \prod_a P(z_a|\boldsymbol{\theta}_a)$. If no direct measurement is available, it is convenient to write this as a measurement with $\sigma_a \to \infty$ (and z_a arbitrary). With Bayes' theorem we get:

$$\begin{aligned} &P(\boldsymbol{\theta}|\boldsymbol{z}_g, \boldsymbol{z}_f, \boldsymbol{z}_\theta) \\ &\quad = P(\boldsymbol{z}_g, \boldsymbol{z}_f, \boldsymbol{z}_\theta|\boldsymbol{\theta}) P_{\mathrm{pr}}(\boldsymbol{\theta}) \\ &\quad = \Big[\prod_i P(z_{g_i}|g_i(\theta_i, \theta_j : j \in N(i)) \times P(z_{\theta_i}|\theta_i)\Big] \\ &\qquad \times \Big[\prod_{(i,j)} P(z_{f_{ij}}|f_{ij}(\theta_i, \theta_j))\Big] \times P_{\mathrm{pr}}(\boldsymbol{\theta})\,, \end{aligned} \tag{12}$$

where $\boldsymbol{z}_g, \boldsymbol{z}_f$ and $\boldsymbol{z}_\theta$ denote the set of power injection, flow and angle measurements, respectively. Thus, this distribution gives the likelihood that the true state is $\boldsymbol{\theta}$, given the measurements in $\boldsymbol{z}_g, \boldsymbol{z}_f, \boldsymbol{z}_\theta$. In view of the state estimation problem, we are interested in calculating marginals of Eq. 12 such as $P_{(ij)}(\theta_i, \theta_j) \equiv \int \prod_{k \neq i,j} d\theta_k P(\boldsymbol{\theta}|\boldsymbol{z}_g, \boldsymbol{z}_f, \boldsymbol{z}_\theta)$ to calculate likely values of the flow f_{ij} and the corresponding phase angles (and similar for other quantities). To calculate the marginals, we have to deal with a number of integrals of large products over all vertices and transmission lines.

3.1 BP for power grid state estimation: performance in terms of speed and accuracy

The distribution $P(\boldsymbol{\theta}|\boldsymbol{z}_g, \boldsymbol{z}_f, \boldsymbol{z}_\theta)$, as a function of $\boldsymbol{\theta}$, is of the form of the general distribution given in Eq. 1 (assuming a uniform prior). We can thus use BP to solve the state estimation problem and compare the results for the different factor graphs.

In the following, the subscript x shall indicate for which factor graph F_x is evaluated: F_v (the straightforward factor graph), F_c (the clustered factor graph), and F_f (the flow-only factor graph). We use BP to solve the state estimation problem on the IEEE-300 benchmark network [30]. The IEEE-300 network is a realistic and heterogenous benchmark network with 300 vertices, 411 links and 112 loops, it is described in more detail in Appendix C. As explained in Sects. 1 and 2.3, the use of the new algorithm refers to the avoidance of extra loops in the associated factor graph. We do not modify the 112 loops (by clustering some vertices) in the IEEE-grid, but keep them and compare the performance with and without the loops in the assigned factor graphs. Note that the number of additional loops in the naively assigned factor graph would be $\sum_i |N(i)| = 2 \cdot \sharp \text{ links} = 2 \times 411 = 822$ on top of the 112 loops of the IEEE-300 grid, and all these additional loops would be short.

We will consider measurements of the power flows, measurements of the power injections, and, if PMUs are assumed to be present, measurements of the phase angles. We discuss two situations: one where all of the variables are measured (measurement devices at every vertex and transmission line), and one—the more realistic case—where only the flows and injections are measured (i.e., without any PMUs). The flow and injection measurements are assumed to have an error ξ_a with variance $\sigma_a^2 = 10^{-3}$, while the angle measurements, if present, are assumed to have an error ξ_a with variance 10^{-6}. Using these values, we randomly draw the measurements $\boldsymbol{z}$ following the description in the previous section, and use BP to find estimates of the state variables by calculating marginals of $P(\boldsymbol{\theta}|\boldsymbol{z})$.

For comparison, we first make use of a "damping" method proposed in [8] to improve the convergence of BP on F_v (since numerical simulations have shown that naively running BP on F_v gives diverging estimates). According to this damping procedure of [8], for each Gaussian message $m^t_{x\to y} \sim N(\mu^t_{x\to y}, (\sigma^2)^t_{x\to y})$ of Eqs. 4 and 5 one chooses with probability 1/2 either $\delta = 0$ or $\delta = 1$ ($P(\delta = 0) = P(\delta = 1) = 1/2$), and updates:

$$\mu^{t+1}_{x\to y} = \delta \times \hat{\mu}^{t+1}_{x\to y} + (1-\delta) \times 1/2 \times \left(\hat{\mu}^{t+1}_{x\to y} + \mu^t_{x\to y}\right), \tag{13}$$

where $\hat{\mu}^{t+1}_{x\to y}$ is the mean of the message that would have been calculated at step $t+1$ without damping. The variance $\sigma^2_{x\to y}$ is damped equivalently. Thus, with probability 1/2 the message is updated as usual, and otherwise damped by a factor of 1/2. Testing the method according to Eq. 13 for different damping parameters and comparing it to the damping algorithm proposed in [19], we indeed find that damping according to [8] (Eq. 13) improves the convergence the most. For the results for F_v we will use this "damped" version of BP and use it as the best existing alternative, which is still outperformed by our method. For F_c and F_f, the algorithm converges without problem also without damping, so we will consider their undamped versions.

3.2 Results for the factor graphs F_v, F_c and F_f

We present our results for the predictions from the three factor graphs F_v, F_c and F_f. (Details on the implementation are given in Appendix D.) We focus here on the estimation of the flows $\{f_{ij}\}$ (in F_c and F_v these can be retrieved as $B_{ij}(\theta_i - \theta_j)$). BP on F_v, F_c and F_f will produce different estimates of the marginals $\{P_{(ij)}(f_{ij})\}$, which we will denote by $\{b^v_i(f_{ij})\}$, $\{b^c_i(f_{ij})\}$ and $\{b^f_i(f_{ij})\}$, respectively. To retrieve their accuracy, we need a way to compare them with the 'true' marginals $P_{(ij)}(f_{ij})$.

Note that the variables of the factor graph F_c are tuples $\theta^c_{ij}, \theta^c_{ji}$, the beliefs resulting from F_c depend on θ_i, θ_j, so that averages or variances of the flows $B_{ij}(\theta_i - \theta_j)$ can be directly calculated using these beliefs as the probability distribution $P_{(ij)}(\theta_i, \theta_j)$ according to Eq. 6, when calculating expectation values. In contrast, the variables of F_v are θ_i, the resulting beliefs depend on θ_i separately. In this case, the variance of the flow f_{ij} cannot simply be obtained as the sum (B^2_{ij} times the variances of θ_i and θ_j), since θ_i and θ_j are correlated. The marginal distribution of $f_{ij} = B_{ij}(\theta_i - \theta_j)$, in particular its variance, must be calculated from Eq. 7.

Since we assume all factors are Gaussian, the marginal distributions of the flows are Gaussian as well, so we denote them by their mean and standard deviation as $P_{(ij)}(f_{(ij)}) \sim N(\mu_{(ij)}, \sigma_{(ij)})$. Here the subscript (ij) represents the flow variables in the following. We calculate the means $\{\mu_{(ij)}\}$ via the least-squares approach and the standard deviations $\{\sigma_{(ij)}\}$ by a matrix inversion. Both the means and standard deviations that are calculated in this way are presumed to set the accurate benchmark for a comparison to the accuracy of the different implementations of BP. Each implementation of the factor graph with $x \in \{v, f, c\}$ gives an estimate $b^x_{f_{(ij)}}(f_{(ij)}) \sim N(\mu^x_{(ij)}, \sigma^x_{(ij)})$, which should be close to the benchmark values $\mu_{(ij)}, \sigma_{(ij)}$. For a chosen method x we summarize the estimates into the average square error of $\{\mu^x_{(ij)}\}$ and $\{\sigma^x_{(ij)}\}$ by defining:

$$\Delta_\mu \equiv \frac{1}{411} \sum_{(ij)} (\mu^x_{(ij)} - \mu_{(ij)})^2 \tag{14}$$

$$\Delta_\sigma \equiv \frac{1}{411} \sum_{(ij)} (\sigma^x_{(ij)} - \sigma_{(ij)})^2 , \tag{15}$$

where the sum runs over all 411 links of the IEEE-300 network. We use these deviations to assess the accuracy

of the estimates provided by the different implementations of BP. Since random generation of the measurement data $\boldsymbol{z}$ leads to differing errors, we repeat the procedure 100 times. For a fair comparison, we note that per BP iteration the wall-clock time for a native Python 3.7 implementation (on an Intel i5-2400 processor) of F_f, F_c and F_v takes 0.01 s, 0.03 s, and 0.1 s, respectively. The implementation is given in the supplementary material. Native Python is relatively very slow (up to two orders of magnitude slower than other faster implementations) and the calculations can be massively parallelized, so these time-scales can be significantly reduced (the relative speed of BP on the different factor graphs are expected to remain more or less unchanged).

Figure 5 shows how the error on the variance Δ_σ saturates for BP on the different factor graphs. For all situations discussed above, the estimated variances converge fast to small values. The estimates provided by F_c are significantly more accurate than those provided by F_v, which are again significantly more accurate than those provided by F_f. Note that F_v corresponds to the standard implementation of BP with a naive factor graph assignment.

Focusing on the estimates for the mean, the convergence of the estimates for the different scenarios are shown in Fig. 6. Figure 6a shows the situation where PMU measurements are included. After convergence, the means predicted by F_v and F_c are both exact (as is in general true for means predicted by Gaussian BP [31,32]). However, F_c converges in much less iterations than F_v, by around a factor of 400. Looking at the mean estimates for the situation without PMUs, as shown in Fig. 6b, the situation is similar. F_c converges quickly to the exact answer. Although it is not shown here, experiments indicate that eventually the mean predicted by F_v does converge. However, the time scale over which it converges is so much larger (about 10^6 iterations) that it renders the final estimate practically irrelevant. In practice, in the absence of angle measurements even F_f performs better than F_v.

In summary of Figs. 5 and 6, it should be emphasized that our comparison refers to the performance of different BP-algorithms, differing by the assigned factor graph, for which the newly proposed assignment F_c performs best. In contrast to other approaches such as least-squares or quasi-Newton methods, the general supremacy of BP based algorithms was demonstrated already in [8–10], as mentioned in the introduction.

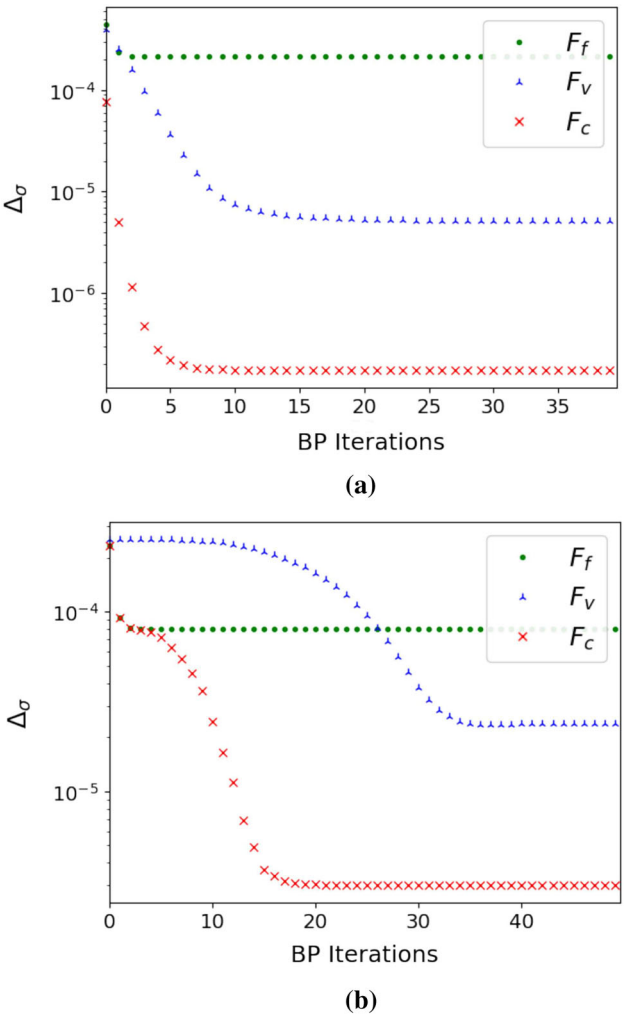

Fig. 5 The saturation of the variance predicted by BP on the different factor graphs, as measured by the average square error Δ_σ (see Eq. 15), showing that the clustered version of BP is most accurate: **a** with angle measurements, **b** without angle measurements. The variances predicted by F_c and F_f do not depend on the values of the chosen measurements. The variances predicted by F_v are slightly different every time BP is run because of the probabilistic damping (Eq. 13); here an average over 100 random measurement sets is shown. Note that F_v corresponds to the standard factor graph assignment and serves as the best version of existing alternatives

4 Outlook to other applications of the algorithm to power grids

Our clustering rule for loopy factor graphs can be used for any distribution of the form of Eq. 1. It thus applies for BP to a variety of supply networks, if the flows are conserved at the vertices and are determined by variables at the vertices ($\boldsymbol{v}$, in our notation). In the following we mention different versions of power flow in electricity grids and discuss applications to gas-pipeline networks and fluid flow networks in the appendix. Problems that are studied in relation to power flow include power flow analysis, i.e., solving the power flow equations, similar to what we analyze for the gas pipe network in Appendix A, optimal power flow [22], state estimation [8,33], as considered in Sect. 3, and optimization under uncertainty [24,25]. In these applications, typically three different power flow equations are distinguished:

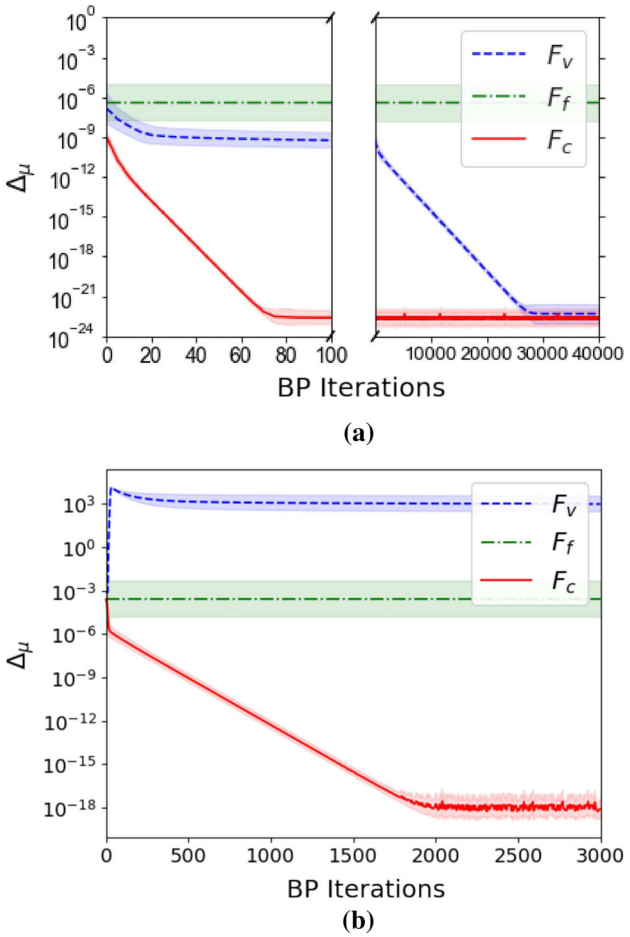

Fig. 6 The convergence of the mean predicted by BP on the different factor graphs, as measured by the average square error Δ_μ (Eq. 14), as a function of the number of BP iterations, showing that BP with only flows (F_f) is fastest but not very accurate, while clustered BP (F_c) converges reasonably fast and gives the exact answer (up to machine precision). When un-clustered BP (F_v) *does* converge it also gives the exact answer; this takes, however, very long: **a** with angle measurements, **b** without angle measurements. The predictions depend on the values of the measurements; here the lines give an average of Δ^μ over 100 sets of random measurements. The filled region gives the standard deviation of Δ^μ between different random sets of measurements

- The DC-approximation as used in Sect. 3 as an approximation of the AC-equations, valid for high voltages at low power losses. Note that despite the name, the application of the DC-approximation is to AC-networks.
- In a Direct Current (DC)-network, the size of the current I_{ij} between vertices i and j is given by Ohm's law $I_{ij} = (V_i - V_j)/R_{ij}$, where V_i and V_j are the voltages at vertices i and j and R_{ij} is the resistance of the link connecting vertices i and j. DC is used in low-voltage distribution grids and in very long distance transmission [34].
- Alternating Current (AC) is typically used for high-voltage long-distance transmission [34]. The network voltages $\{V_i(t)\}$ oscillate at a constant frequency ω, such that $V_i(t) = \sqrt{2}|V_i|\sin(\omega t + \theta_i)$. Here $|V_i|$ is the voltage magnitude. Together with the phase angles θ_i, they can be determined from the equations

$$f_{ij}^P = |V_i||V_j|\left[G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)\right] - |V_i|^2 G_{ij}\,, \tag{16}$$

$$f_{ij}^Q = |V_i||V_j|\left[G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)\right] + |V_i|^2 B_{ij}, \tag{17}$$

where f_{ij}^P and f_{ij}^Q are the active and reactive power flow, each of which is conserved at the vertices. Solving the equations requires the specification of the quantities B_{ij} and G_{ij} for each transmission line, known as the susceptance and the conductance, respectively, which can be calculated from the impedance and resistance of the transmission line [34]. To implement AC in our framework, the tuples $(\theta_i, |V_i|)$ should be considered as the vertex variables v_i. Due to the nonlinearity of Eq. 16, BP should be combined, for example, with GN, as in [33].

For power grids, a situation with uncertain costs due to fluctuations in uncertain power injections by renewable resources was investigated in [24,25]. In this case, each possible production assignment leads to a different distribution of the form of Eq. 1. Uncertain power injections may enter the probability distribution $P(\boldsymbol{v})$ of Eq. 1 as a product over all vertices $\prod_i \exp\left\{-[g_i - \sum_{j\in N(i)} f_{ij}(v_i, v_j)]^2/(2\sigma_i^2)\right\}$ if the production fluctuates according to such a Gaussian distribution with variance σ_i^2 around some mean injection g_i, while fixed and controllable production at vertices k would multiply this term by $\prod_k \delta(g_k - \sum_{j\in N(k)} f_{kj}(v_k, v_j))$. $P(\boldsymbol{v})$ induces a distribution of flows $P(\{f_{ij}\})$ which directly indicates possible overflows of transmission lines. BP may then be used to calculate average costs induced by the production g_k at the set of controllable production vertices. These costs can furthermore include probabilistic constraints on $\{g_k\}$ which enforce, for example, that severe link overloads are rare [24]. In [23], it is shown that such an optimization can be performed with BP using the fact that marginals of the distribution satisfy the BP equations (to be given below). This is mathematically similar to the 'survey propagation' studied in [1] and investigation of large deviations given in [35].

5 Conclusions

For applications of BP, we considered a new assignment method of factor graphs which avoids the generation of

additional loops as compared to the original supply networks. Our method applies to state estimation or optimization problems whose state can be described by a probability distribution that factorizes over the vertices and over the links of the network. If these distributions are summed or integrated over upon marginalization, BP provides an efficient way of organizing the sum or integrals over these products. In the naive assignment of a factor graph, the variables on the original supply network are chosen as variable nodes on the factor graph, while the factors of the probability distribution determine the factor nodes on the factor graph. Constraints and physical laws between the variables on the original grid may then induce additional loops on the factor graph that are detrimental for the convergence speed of BP. Our main goal was to avoid these loops.

This means that our algorithm does not address the handling of loops in the graphical representation of the original supply networks, which may differ in size and reflect the original network architecture. Such loops will survive our factor graph representation and may impede the convergence or accuracy of BP. Our method is supposed to complement other methods such as loop expansions and additional clustering rather than replacing them. For practical applications, we have furthermore focussed on cases in which the messages themselves are Gaussian functions so that the messages reduce to a few real numbers to be sent. In particular, we have assumed Gaussian distributed errors in the state estimation problems, and used successive Gaussian approximations in the steady-state analyses.

In the cases we considered, additional loops in the factor graphs result from constraints between the variables which are analogues to the two Kirchhoff laws in power grids, one corresponding to flow conservation at vertices (the flow in general being electricity, gas, water, air, soil, traffic), the second one restricting a quantity related to energy (voltage, pressure, time, other costs) along loops in the grid. The shared mathematical structure of these constraints explains the wide range of applicability of our algorithm. When the resistance in the transmission lines of the original grid are depending on the flow, the analogue of Kirchhoff' s second law amounts to a nonlinear relation (differently from Ohm's law). In this case an additional iterative method such as the Gauss–Newton method is required, and BP can be applied in the intermediate steps.

While our algorithm enhances the accuracy and improves the convergence speed, the additional computational effort is moderate. We have explicitly worked out this approach for the state estimation on a benchmark power grid. In the appendix we describe the state determination on two benchmark gas-networks with nonlinear flow relations, the latter case detailed in 2.3.1. We compared the performance of three factor graphs: the naively assigned, the newly proposed clustered factor graph, and the flow-only factor graph. The naively assigned one can lead to accurate results, but at the price of slow convergence if at all. The flow-only factor graph ignores constraints from (the analogue of) Kirchhoff's second law, it is thus less accurate but useful for a first estimate and fast. The clustered factor graph is both fast and accurate and—combined with an iterative procedure in case of nonlinear constraints—it is widely applicable. Further applications to other indicated supply networks should be worked out in the future.

Acknowledgements We thank the Bundesministerium für Bildung und Forschung (BMBF) (Grant number 03EK3055D) for financial support.

Author contributions

TR proposed and implemented the algorithm and performed the simulations. Both authors discussed and contributed to the manuscript. HM-O is responsible for its final form.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: Data for the test networks are contained in the given references. Data related to the numerical results are shown in the figures and are available upon request.]

Appendix A: steady-state analysis in natural gas-pipeline networks

Why is a steady-state analysis important? Gas transmission network operators are obliged to offer as much freely allocable capacity as possible. They are supposed to ensure that gas traders or consumers feed in or withdraw gas at their entries or exits without being concerned about the impact on the overall grid. On the other hand, to meet the statutory carbon emission targets, the use of natural gas has to decline in the course of time. New sources of alternatives (hydrogen, biogas) are discussed so that the distribution network may be challenged by a mixture of gas input. In this case, a steady-state analysis of a gas network is required under varying operational conditions. In steady-state anal-

yses, nodal pressures and pipe flows are computed for given values of source node pressures and gas consumption.

We choose our benchmark gas-networks from GasLib [36]. GasLib is a collection of technical gas network descriptions as well as "contract based nomination data". It is based on real-world network data from the gas transport company Open Grid Europe GmbH. "The data are distorted in order to yield a realistic gas network that is significantly different from the original", but to serve as benchmarks for test simulations.

In analogy to state estimation problems as we considered for power grids, state estimation or leak detection may be applied to gas (or water) transport systems to convert system measurements into reliable information on the network state. In our application here, instead we want to determine the steady state, that is the gas flow and pressures throughout a gas-pipe network, based on input that is assumed to be accurate [5,37–39]. We will perform these calculations with BP for the two realistic natural gas networks (Gaslib-134 and Gaslib-40), to compare the factor graphs described in Sect. 2. For the gas-pipe networks, the data of the benchmark networks allow in principle a complete determination of flows and pressures at all vertices of the network. However, to illustrate our method, we do not make use of the input values of the three largest generators of the gaslib-134, but specify instead the pressures at these stations, so that the unknown variables (flow, pressures, injections) are in principle fully determined, but less easily accessible. This mimics a realistic situation in a gas network and serves as test bed for our method. To ensure that the system is fully determined, at each vertex either the injection of gas (positive or negative) or the pressure has to be specified. Given a gas flow equation relating flow Q_{ij} through a given pipe (ij) with the pressure p_i and p_j at the ends, the pressure at the vertices and the flows through the pipes can then be calculated.

As a gas-flow equation we use

$$Q_{ij} = a_{ij}\mathrm{sgn}(p_i^2 - p_j^2)\left(p_i^2 - p_j^2\right)^{0.5} \tag{A.1}$$

for coefficients a_{ij} specific to each transmission line. For more details we refer to Sect. 5. Combined with the continuity equation

$$g_i = \sum_{j\in N(i)} Q_{ij}, \tag{A.2}$$

where g_i is the injection at vertex i, the pressure and flows throughout the network can then be calculated. Treating the flow equations as constraints on $\{p_i\}$ and following Sect. 2, the equations can be solved by BP. Eq. A.2 (Eq. A.1) corresponds to the first (second) Kirchhoff's law, respectively.

Appendix A.1: steady-state analysis for gas networks: BP combined with Gauss–Newton

To explicitly show the application of BP to this problem, we first map the solution to the equations to a (least-squares) minimization problem, a standard method in the numerical solution of non-linear equations. (As for the power state estimation, finding the means (and more generally the Maximum Likelihood Estimate [33]) can be formulated as a least-squares minimization problem, which we used as benchmark for comparison.) For convenience we use the square of the pressures as variables, rather than the pressures themselves and define $v_i \equiv p_i^2$. We denote the set of known (squared-) pressures $\bar{v_i}$ by $\bar{\boldsymbol{v}}$, the set of known injections $\bar{g}_i$ by $\bar{\boldsymbol{g}}$, and assume that these quantities fully determine the system. We then define the cost function in terms of v_i, v_j:

$$C(\boldsymbol{v}) \equiv \sum_{\bar{g}_i\in\bar{\boldsymbol{g}}} \left(\bar{g}_i - \sum_{j\in N(i)} Q_{ij}(v_i, v_j)\right)^2/2 + \sum_{v_i\in\bar{\boldsymbol{v}}} \left(\bar{v}_i - v_i\right)^2/2, \tag{A.3}$$

with $Q_{ij}(v_i, v_j) = a_{ij}\mathrm{sgn}(v_i - v_j)(v_i - v_j)^{0.5}$ from Eq. A.1. One can verify that the global minima correspond to sets of pressures such that $C(\boldsymbol{v}) = 0$ and such that the equations are satisfied. Using Eq. 2 to turn the minimization into a probability distribution gives a distribution of the form of Eq. 1, with a factor of type H_i for each vertex. The factors of type $\{H_{ij}\}$ are absent in this case, but can be included by simply setting the H_{ij} equal to a constant. Thus, we can apply our clustering procedure to let BP effectively calculate the solutions of these equations.

In a direct application of BP, the numerical computation of the message update Eqs. 4–7 is, however, complicated by the non-linearity of the flow equation. This means that the cost function (Eq. A.3) is non-quadratic. Consequently the corresponding distribution is not Gaussian, and the messages do not maintain a Gaussian shape. This would require a more sophisticated method to represent the messages and to perform calculations with such messages (as discussed in Sect. 2.2).

The difficulty in handling non-linearities is of course not restricted to BP. Methods such as Newton's method can then be used to make successive approximations of the cost function, eventually converging to a minimum of Eq. A.3. If each approximation gives a quadratic cost function, the quadratic minimization can be solved efficiently with BP using only Gaussian messages, thus allowing for a simpler implementation of BP. For a quadratic cost function it is the mean of the associated distribution (Eq. 2) that gives the minimum of the cost function. The mean is furthermore independent of T, so we can conveniently set $T = 1$ rather than taking the limit.

Methods that can implement BP in this way are, for example, apart from Newton's method, the Gauss-Newton (GN-)method (employed in [33] to solve the AC state estimation problem and in [13] for water network state estimation), or the fully parallelized BP procedure proposed in [16]. To give an explicitly worked out example, here we use a modified Gauss–Newton method, combined with BP to perform the calculations at each iteration of Gauss–Newton. The procedure is as follows:

1. Set an initial guess for the flows, $\{Q_{ij}^*\}$, obtained from the flow-only-approximation (ignoring the v_i variables). See Appendix D for details on the implementation.
2. At each iteration:
   - Generate a guess $\{v_i{}^*\}$ from the guess $\{Q_{ij}^*\}$, the known values $\{\bar{v}_i\}$ and the flow equation $Q_{ij} = a_{ij}\mathrm{sgn}(v_i - v_j)|v_i - v_j|^{0.5}$. This can be done in a straightforward way with time complexity linear in the system size (see the supplementary material).
   - Linearize the flow equation around this guess $\{v_i{}^*\}$ by expanding $Q_{ij}(v_i, v_j)$ to first order in $(v_i - v_j)$.
   - Using this linearization, the cost function (Eq. A.3) becomes quadratic in $\{v_i\}$. These equations can then

be solved efficiently by Gaussian BP with our clustering method to generate a new guess $\{Q^*_{ij}\}$ (see Appendix D for details on the implementation).

3. If the guesses $\{Q^*_{ij}\}$ and $\{v_i{}^*\}$ have converged to a fixed value, then the non-linear equations have been solved. Stop the iteration if a desired tolerance threshold is reached.

The unmodified GN-algorithm would follow the same steps, but omit the intermediate flow guesses $\{Q^*_{ij}\}$. Newton's method would omit the linearization altogether, and instead expand $C(\boldsymbol{v})$ directly to second order in $\boldsymbol{v}$ around a guess $\boldsymbol{v}^*$. Newton's method does not allow an efficient calculation of the BP-messages given in Appendix D, and requires instead a large number of matrix inversions.

The modification of the GN-algorithm we use here instead takes advantage of the linearity of flow conservation. Comparing it to the unmodified GN algorithm we found that it strongly improves the convergence. On tree networks, the modified GN-algorithm gives the exact result in a single iteration (which is true for neither the unmodified GN-algorithm nor for Newton's method). For the GasLib-134 network (see the next section), instead we restrict ourselves artificially and use only a single $\bar{v}_i$ (instead of all $\bar{\boldsymbol{v}}$) when generating a guess $\{v^*_i\}$ from $\{Q^*_{(ij)}\}$, to allow a comparison of F_c and F_v in an iterative setting.

Appendix A.2: results for GasLib networks as benchmarks

Here we compare BP on the different factor graphs F_v and F_c. Since F_f ignores constraints imposed by the vertex variables (pressures), the flow-only approximation is under-determined and cannot give a valid solution (in contrast to the state estimation problem studied in Appendix A, where the system is over-determined). The flow-only approximation is, however, still useful as an initial guess for the GN-procedure (simply setting undetermined variables to 0), which we have used here.

The GasLib-134 has 134 vertices, 133 links and no loops, while GasLib-40 has 40 vertices, 45 links and 6 loops. We presume that for GasLib-40 the pressure at one vertex is known (Appendix C), while the injection at all other vertices are known. For GasLib-134, we presume that the pressure at three vertices is known (Appendix C), while the injection at all other vertices are known to mimic a realistic situation. This fully determines the system, such that the flows $Q_{(ij)}$ can be calculated. For a guess $Q^x_{(ij)}$, given by a method $x \in \{v, c\}$, we compare the guess to the actual flows $Q_{(ij)}$ (obtained by a least-squares procedure) by defining the average square error:

$$\Delta \equiv \frac{1}{N}\sum_{(ij)}(Q^x_{(ij)} - Q_{(ij)})^2, \tag{A.4}$$

where N is the number of links of the network used. We use the GN-procedure, where for each step BP is either run on F_v or on F_c. Thus, the difference results only from the way of how each step of the GN-procedure is calculated. The results are given in Fig. 7, where the average square error Δ (Eq. A.4) is shown as a function of the total amount of GN-steps and BP-iterations. Very few GN-steps are needed for the convergence to the correct solution; the main difference is in the amount of BP-iterations that is needed to complete a single GN-step.

For each GN-step for the GasLib-40 network, we iterate BP until convergence. (Specifically, the stopping criterion is as follows: We take an amount of iterations T such that the square difference between the estimate at T and the estimate at $T/2$ is smaller than $10^{-10}/40$ (more lenient stopping criteria may decrease the amount of iterations needed by a factor ~ 3).) Figure 7 shows that this takes about 1500 iterations per GN step for BP on F_c, compared to 50000 iterations for each BP step on F_v. On top of this, each BP iteration on F_v takes about three times as much CPU time as an iteration on F_c, such that in terms of speed F_c outperforms F_v by a factor ~ 100.

For the network GasLib-134 the contrast is even stronger, here BP on F_c and F_v perform according to two extremes: This network is a tree network, in which the longest path has a length of 55. Consequently, the associated factor graph F_c is also a tree graph, and within each GN-step BP on F_c converges to the exact result after 55 iterations (Fig. 7). This is not true for F_v, which has many loops despite GasLib-134 being a tree network. We find that BP on F_v shows oscillatory behaviour and does not appear to converge at all (even after millions of iterations).

Appendix B: Outlook to further applications to fluid flow

Typical calculations for fluid flow are similar to those studied for natural gas in Appendix A, where the aim is to calculate the pressures and flows throughout the network, given that we know the injections and pressures at some selected nodes (such that the system is fully determined). Such calculations are required for water or gas infrastructure in urban environments, or for long-distance transmission [5,37–39]. Other calculations in the context of such infrastructure networks refer to optimization [36] and state estimation [40]. The effect of fluctuating inputs or link damage on the fluctuations of pressure and flow ([41]) can be similarly effectively studied with the BP framework constructed in Sect. 2. For

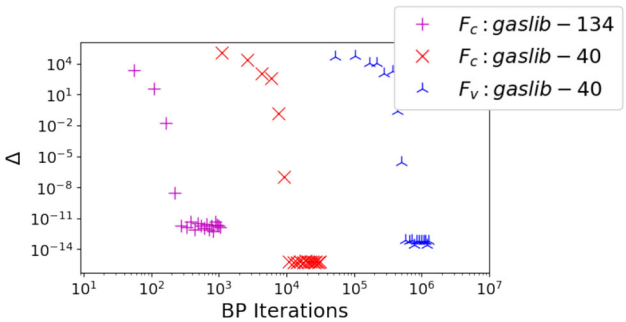

Fig. 7 Total BP-iterations needed for convergence on the factor graphs F_v and F_c for GasLib-40 and F_c for GasLib-134, as measured by the average square error Δ (Eq. A.4). Each marker indicates a single GN-step. Non-zero Δ after convergence is due to numerical round-off and finite tolerance. Results for F_v on GasLib-134 are not displayed as here BP does not converge at all

explicit expressions of the relevant equations for fluid flow, three regimes can be distinguished:

- Pressurized gases, which we studied with a modified GN-method combined with BP in Appendix A. Relations between flow and pressure are partly theoretically and partly phenomenologically motivated [5,38,39]. They are typically of the form $Q_{ij} \propto |p_i^2 - p_j^2 + b_{ij}|^{0.5-0.55}$, as we used for our gas-flow analysis in Appendix A. Here the constant b_{ij} describes the effect of gravity in case of height differences between the ends of the pipe, and the proportionality constant describes friction effects [5,38,39].
- Laminar flow (low-velocity flow, thin pipes): A relation between the flow rate through a pipe and the pressure at either end can be calculated directly from the Navier-Stokes equation [42], giving the linear relationship $Q_{ij} = a_{ij}(P_i - P_j)$. Here Q_{ij} is the flow rate from point i to point j, P_i and P_j are the respective pressures at point i and point j, and a_{ij} is a constant determined by the dimensions of the pipe and the viscosity of the fluid. BP is directly applicable in this case.
- Incompressible fluids (such as liquids or low-pressure gases). Fluids that can be treated as incompressible comprise gas flow in municipal distribution networks, water flow in water works, air flow in ventilation systems in buildings or district heating, or cooling systems. Incompressible fluid means that the pressure drop is negligible and the density remains approximately constant. In common to fluid networks, when the resistance in the pipe depends on the flow, the problem is nonlinear [37,43,44]. As in Appendix A, one can use an iterative procedure combined with our BP algorithm to calculate the distribution of fluid flow through the pipes.

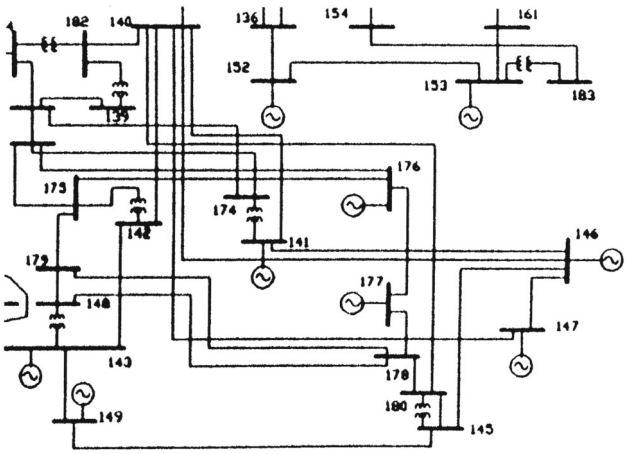

Fig. 8 A section of the IEEE-300 network (the network is based on the power grid of the Midwestern USA), from [45]. In our work, we only distinguish between the buses (vertices), here represented as bars to which links connect, and the links between the buses. The full IEEE-300 network has 300 vertices and 411 links

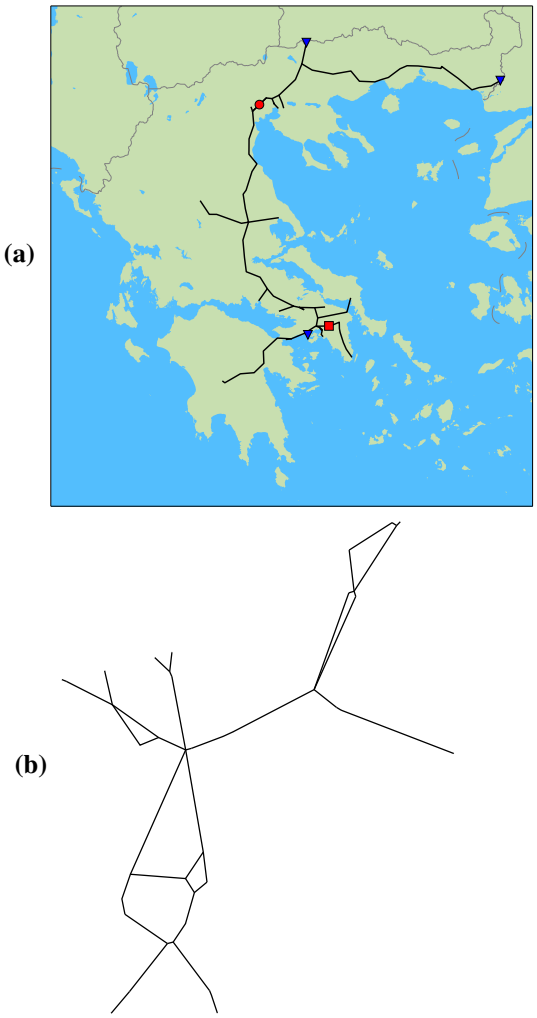

Fig. 9 **a** Topology of the GasLib-134 network, a tree network representing the gas pipeline connection between Greece and Bulgaria. **b** Topology of the Gaslib-40 network. Both figures are from [36]

Appendix C: Benchmark networks

Here we discuss the IEEE and Gaslib benchmark networks, which we have used to compare the efficiency of BP on the different factor graphs. As the benchmark networks represent real networks, their structure and parameters are highly heterogenous and suitable for testing the applicability of algorithms to practical computations.

The IEEE-300 network. The IEEE benchmark systems are a set of networks that were designed to resemble simplified versions of realistic power networks [45]. These are commonly used in the power systems engineering literature to test the applicability of algorithms related to power flow. Systems of varying sizes are available on [45]. Here we used the IEEE-300 network, 'developed by the IEEE Test Systems Task Force under the direction of Mike Adibi in 1993' [45]. The network consist of 300 buses (vertices) and 411 links, and contains 112 loops. The IEEE-grids are realistic (they are based on the power system in Midwestern USA) and therefore highly heterogenous both in the topology and

in the parameters; a section of the IEEE-300 network is shown in Fig. 8.

In principle, different power devices such as transformers are contained in the network; for simplicity we only consider buses and links. The data contain impedances and resistances of each transmission line, as well as the phase angles at each bus, which together fully specify the network in the DC approximation.

The GasLib networks. The GasLib networks [36] were developed to provide researchers with large and realistic gas benchmark networks, as part of a project funded by the German Federal Ministry of Economics and Technology. The GasLib-134 network is based on the gas-pipeline network connecting Greece and Bulgaria (shown in Fig. 9), albeit somewhat distorted to protect sensitive data. GasLib-134 contains 134 vertices, 133 links and no loops. GasLib-40 contains 40 vertices, 45 links and 6 loops. For simplification we do not consider any devices contained in the network, and only distinguish the vertices and links. The GasLib data contain the network topology and the gas injections. For GasLib-134 we have used the scenario data from 2011-11-01. Computation of the coefficients in Eq. A.1 is less straightforward, and here we simply set the coefficients of all pipes to the same value (50 for GasLib-134, 500 for GasLib-40), determined such that pressures maintain realistic values (between 40 and 60 bar). Coefficients with a small random component give similar results. To get results for the steady-state analysis in Appendix A, the vertices with known pressure values are the 'source_1' for GasLib-40 and 'node_1', 'node_20' and 'node_80' for GasLib-134. Combined with the injection at all other vertices, the systems are then fully determined.

Appendix D: Algorithmic details

Appendix D.1: State estimation

In the state estimation problem of Sect. 3, we have available the topology of the power grid, a set of measurements $\{z_{gi}\}$, $\{z_{fij}\}$ and $\{z_{\theta_i}\}$, and their associated variances $\{\sigma^2_{gi}\}$, $\{\sigma^2_{fij}\}$ and $\{\sigma^2_{\theta_i}\}$. Here the index i denotes the vertices of the power grid, and (ij) denotes the links of the power grid. In our case we have generated the measurements ourselves according to the procedure in Sect. 3. We furthermore have access to the set of susceptances $\{B_{ij}\}$, such that the flows are related to the phase angles as $f_{ij} = B_{ij}(\theta_i - \theta_j)$. Our goal is then to accurately determine the flows, based on the Bayes' theorem (Eq. 11). Let us define for all links (ij) the constants $c_{fij|i} = B_{ij}$, $c_{fij|j} = -B_{ij}$ and $c_{gi|j} = -B_{ij}$, and for all vertices i the constants $c_{gi|i} = \sum_{j\in N(i)} B_{ij}$. For a given set of measurements Bayes' theorem gives a probability $P(\boldsymbol{\theta})$ that the phase angles have values $\boldsymbol{\theta}$ (the posterior distribution $P(\boldsymbol{\theta}|\boldsymbol{z})$ from Eq. 11) as

$$P(\boldsymbol{\theta}|\boldsymbol{z}) \equiv P(\boldsymbol{\theta}) = \prod_i H_i(\theta_i, \theta_j : j \in N(i)) \times \prod_{(ij)} H_{(ij)}(\theta_i, \theta_j) \tag{D.5}$$

$$H_{(ij)}(\theta_i, \theta_j) \propto \exp\Big(-(z_{fij} - c_{fij|i}\theta_i - c_{fij|j}\theta_j)^2/2\sigma^2_{fij}\Big), \tag{D.6}$$

$$H_i(\theta_i, \theta_j : j \in N(i)) \propto \exp\Big(-(z_{\theta_i} - \theta_i)^2/2\sigma^2_{\theta_i}\Big) \times \exp\Big(-(z_{gi} - c_{gi|i}\theta_i - \sum_{j\in N(i)} c_{gi|j}\theta_j)^2/2\sigma^2_{gi}\Big). \tag{D.7}$$

This distribution is of the form of Eq. 1, and hence we can use Belief Propagation on the different factor graphs F_v and F_c. To use the message passing equations (Eqs. 4–7), we need the set of variable nodes x_I, the set of factor nodes a and the associated factor functions $W_a(\boldsymbol{x}_a)$. Summarizing the main text, these are given in Table 1. For the flow-only approximation, which ignores Kirchhoff's second law and the phase angle measurements, the posterior distribution described by Bayes' theorem is

$$P_f(\boldsymbol{f}|\boldsymbol{z}) \equiv P_f(\boldsymbol{f}) = \Big[\prod_i H_i(\sum_{j\in N(i)} f_{ij})\Big] \times \Big[\prod_{(i,j)} H_{ij}(f_{ij})\Big], \tag{D.8}$$

$$H_i(\sum_{j\in N(i)} f_{ij}) \propto \exp\Big(-(z_{gi} - \sum_{j\in N(i)} f_{ij})^2/2\sigma^2_{gi}\Big), \tag{D.9}$$

$$H_{ij}(f_{ij}) \propto \exp\Big(-(z_{fij} - f_{ij})^2/2\sigma^2_{fij}\Big). \tag{D.10}$$

This distribution is of the form of Eq. 9. Following the main text, the factor graph F_f is given in Table 2.

Using these factor graphs, we can use the BP Eqs. 4–7 to calculate the marginals we are interested in. We focus on estimating the flows $f_{ij} = B_{ij}(\theta_i - \theta_j)$, so for F_v this is $\{b_{Hij}(\theta_i, \theta_j)\}$ where the index H_{ij} denotes the factor node at the link (ij). For F_c this is calculated from the copies as $b_{(ij)}(\theta_i, \theta_j) = b_{(ij)}(\theta^c_{ij}, \theta^c_{ji})$, where the index (ij) denotes the variable node at the link (ij). For F_f the relevant marginals are $\{b_{(ij)}(f_{ij})\}$, where (ij) is again the variable node at the link (ij). Due to the fact that all H_i and H_{ij} are Gaussian, all of the messages remain Gaussian, such that we can summarize them by their mean and covariance as $m_{I\to a}(x_I) = N(\mu_{I\to a}, K_{I\to a})$ and $m_{a\to I}(x_I) = N(\mu_{a\to I}, K_{a\to I})$ for all variable nodes I and factor nodes a connected on the factor graph (that is, for which $x_I \in \boldsymbol{x}_a$). Together with the topology of the factor graph and the expressions for the functions $W_a(\boldsymbol{x}_a)$ from Tables 1 and 2, we can plug this into the BP Eqs. 4–6 to obtain the algorithm in terms of products and integrals over Gaussians, therefore, requiring only linear algebra for the means and (co-)variances of the messages. It turns out that for H_i and H_{ij} of the form of Eqs. D.6–D.7 and Eqs. D.9–D.10 the updates can be calculated particularly efficiently (that is, not requiring explicit inversion of any matrices larger than 2x2). The pseudocode for the resulting algorithms is given in the supplementary material.

Appendix D.2: Steady-state analysis

For the steady-state analysis problem of Appendix A, we have access to the topology of the gas network, to the known values of the squared pressures $\overline{\boldsymbol{v}} = \{\overline{v}_i\}$ at some vertices

Table 1 The factor graphs F_v and F_c specified for the power grid state estimation problem

Factor graph	Variable nodes x_I	Factor nodes $W_a(\boldsymbol{x}_a)$
F_v	For each vertex i: θ_i	For each vertex i: $H_i(\theta_i, \theta_j : j \in N(i))$ For each link (ij): $H_{(ij)}(\theta_i, \theta_j)$
F_c	For each link (ij): $(\theta^c_{ij}, \theta^c_{ji})$	For each vertex i: $\int \mathrm{d}\theta_i \left\{ H_i(\theta_i, \theta^c_{ji} : j \in N(i)) \prod_{j\in N(i)} \delta(\theta^c_{ij} - \theta_i) \right\}$ For each link (ij): $H_{(ij)}(\theta^c_{ij}, \theta^c_{ji})$

Table 2 The factor graph F_f specified for the power grid state estimation problem

Factor graph	Variable nodes x_I	Factor nodes $W_a(\boldsymbol{x}_a)$
F_f	For each link (ij): f_{ij}	For each vertex i: $H_i(\sum_{j\in N(i)} f_{ij})$ For each link (ij): $H_{ij}(f_{ij})$

i, the known values of injection $\overline{\boldsymbol{g}} = \{\overline{g}_i\}$ at some vertices i, and to the coefficients $\{a_{ij}\}$ determining the flow $Q_{ij}(v_i, v_j) = a_{ij}\mathrm{sgn}(v_i - v_j)(v_i - v_j)^{0.5}$. Here the variables v_i correspond to the square of the pressure p_i^2 at the vertex i. From Appendix A, at each iteration of the modified GN-algorithm we have a distribution $P(\boldsymbol{v}) \propto \exp(-C(\boldsymbol{v}))$, with $C(\boldsymbol{v})$ as in Eq. A.3 (the resulting distribution is Gaussian, implying its mean minimizes $C(\boldsymbol{v})$). The resulting distribution $P(\boldsymbol{v})$ is of the same form as those for the state estimation problem (Eqs. D.6–D.7), only with different coefficients. Exactly the same BP algorithm can thus be used, only replacing $\{\theta_i\}$ with $\{v_i\} = \{p_i^2\}$, and replacing the values of the coefficients as

$$z_{gi} = \begin{cases} \overline{g}_i - \sum_{j\in N(i)} \left(Q_{ij}(v_i^*, v_j^*) - \frac{\partial Q_{ij}}{\partial v_i}\Big|_{v_i^*, v_j^*} v_i^* - \frac{\partial Q_{ij}}{\partial v_j}\Big|_{v_i^*, v_j^*} v_j^* \right) & \\ & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 0 & \text{else}\,, \end{cases}$$

$$\sigma^2_{gi} = \begin{cases} 1 & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 10^8 & \text{else}\,, \end{cases}$$

$$z_{vi} = \begin{cases} \overline{v}_i & \text{if} \quad \overline{v}_i \in \overline{\boldsymbol{v}}\,, \\ 0 & \text{else}\,, \end{cases}$$

$$\sigma^2_{vi} = \begin{cases} 1 & \text{if} \quad \overline{v}_i \in \overline{\boldsymbol{v}}\,, \\ 10^8 & \text{else}\,, \end{cases}$$

$$c_{gi|i} = \begin{cases} \sum_{j\in N(i)} \frac{\partial Q_{ij}}{\partial v_i}\Big|_{v_i^*, v_j^*} & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 1 & \text{else}\,, \end{cases}$$

$$z_{fij} = 0\,,$$

$$c_{fij|i} = -c_{fij|j} = 1\,,$$

$$\sigma^2_{fij} = 10^8\,.$$

$$c_{gi|j} = \begin{cases} \frac{\partial Q_{ij}}{\partial v_j}\Big|_{v_i^*, v_j^*} & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 1 & \text{else}\,. \end{cases}$$

Here the value 10^8 is simply a very high number implementing the absence of any knowledge of the value of the corresponding variable (a value much lower than 10^8 can lead to inaccuracies, while a much higher value can lead to numerical over- or underflow). The linearization point $\boldsymbol{v}^*$ can be directly obtained from a guess $\boldsymbol{Q}^*$ (itself obtained from the previous BP iteration), the precise algorithm is given in the supplementary material. For the gas network we can also use a flow-only approximation, giving a distribution of the form of Eqs. D.9–D.10, with the flow f_{ij} replaced by Q_{ij}, and the coefficients:

$$z_{gi} = \begin{cases} \overline{g}_i & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 0 & \text{else}\,, \end{cases}$$

$$\sigma^2_{gi} = \begin{cases} 1 & \text{if} \quad \overline{g}_i \in \overline{\boldsymbol{g}}\,, \\ 10^8 & \text{else}\,, \end{cases}$$

$$z_{fij} = 0\,,$$

$$\sigma^2_{fij} = 10^8\,.$$

References

1. G. Del Ferraro, C. Wang, D. Martí, M. Mézard, (2014) arXiv:1409.3048 [cond-mat]
2. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pp. 239–269, ISBN 978-1-55860-811-5 (2003)
3. J.S. Yedidia, J. Stat. Phys. **145**, 860–890 (2011) ISSN 0022-4715, 1572-9613
4. J. Yedidia, W. Freeman, Y. Weiss, IEEE Trans. Inf. Theory **51**, 2282–2312 (2005), ISSN 0018-9448
5. M. Abeysekera, J. Wu, N. Jenkins, M. Rees, Appl. Energy **164**, 991–1002 (2016), ISSN 03062619
6. Y.-F. Huang, S. Werner, J. Huang, N. Kashyap, V. Gupta, IEEE Signal Process. Mag. **29**(5), 33–43 (2012). https://doi.org/10.1109/MSP.2012.2187037
7. A.S. Meliopoulos, G.J. Cokkinides, F. Galvan, A.B. Fardanesh, in *Distributed State Estimator Advances and Demonstration*. Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)
8. M. Cosovic, D. Vukobratovic, in *State estimation in electric power systems using belief propagation: An extended DC model*. 2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) (IEEE, Edinburgh, 2016), pp. 1–5, ISBN 978-1-5090-1749-2
9. M. Cosovic, M. Delalic, D. Raca, D. Vukobratovic, (2020) arXiv:1907.10338 [cs, math]
10. T. Ritmeester, H. Meyer-Ortmanns, Phys. Rev. E **102**, 012311 (2020), ISSN 2470-0045, 2470-0053
11. Y. Hu, A. Kuh, T. Yang, A. Kavcic, IEEE Comput. Intell. Mag. **6**, 36–46 (2011), ISSN 1556-603X

12. Y. Weng, R. Negi, M.D. Ilic, in *Graphical model for state estimation in electric power systems*. 2013 IEEE International Conference on Smart Grid Communications (SmartGridComm) (IEEE, Vancouver, 2013), pp. 103–108, ISBN 978-1-4799-1526-2
13. Q. Han, R. Eguchi, S. Mehrotra, N. Venkatasubramanian, in *Enabling State Estimation for Fault Identification in Water Distribution Systems Under Large Disasters*. 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS) (IEEE, Salvador), pp. 161–170 (2018), ISBN 978-1-5386-8301-9
14. E. Ortega, A. Braunstein, A. Lage-Castellanos, Stoch. Environ. Res. Risk Assess. **34**, 493–511 (2020), ISSN 1436-3240, 1436-3259
15. C.H. Yeung, D. Saad, K.Y.M. Wong, Proc. Natl. Acad. Sci. **110**, 13717–13722 (2013), ISSN 0027-8424, 1091-6490
16. K.Y.M. Wong, D. Saad, Phys. Rev. E **76**, 011115 (2007), ISSN 1539-3755, 1550-2376 arXiv: cond-mat/0609367
17. R. Tanner, *IEEE Transactions on Information Theory* **27**, 533–547 (1981), ISSN 1557-9654 conference Name: IEEE Transactions on Information Theory
18. T. Heskes, K. Albers, B. Kappen, in *Approximate inference and constrained optimization*. Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence. UAI'03 (Morgan Kaufmann Publishers Inc., San Francisco, 2002), pp. 313–320, ISBN 0127056645
19. M. Pretti, J. Stat. Mech. **2005**, P11008–P11008 (2005), ISSN 1742-5468
20. B. Wemmenhove, B. Kappen, (2007) arXiv:0705.4566 [cs]
21. J. Mooij, B. Wemmenhove, B. Kappen, T. Rizzo, in *Loop corrected belief propagation*, ed. by M. Meila, X. Shen. Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics. (Proceedings of Machine Learning Research, vol. 2) (PMLR, San Juan, 2007), pp. 331–338
22. K. Dvijotham, M. Chertkov, P. Van Hentenryck, M. Vuffray, S. Misra, *Constraints* **22**, 24–49 (2017), ISSN 1383-7133, 1572-9354
23. F. Altarelli, A. Braunstein, A. Ramezanpour, R. Zecchina, J. Stat. Mech.: Theory Exp. **2011**, P11009 (2011), ISSN 1742-5468
24. T, Summers, J. Warrington, M. Morari, J. Lygeros, in *Stochastic optimal power flow based on convex approximations of chance constraints*. 2014 Power Systems Computation Conference, pp. 1–7 (2014)
25. D. Bienstock, M. Chertkov, S. Harnett, *SIAM Review* **56**, 461–495 (2014), ISSN 0036-1445, 1095-7200
26. F. Harary, *Graph theory*, 15th edn. (Perseus Books, Cambridge, 2001), ISBN 978-0-201-41033-4 oCLC: 248770458
27. M. Isard, J. MacCormick, K. Achan, in *Continuously-adaptive discretization for message-passing algorithms*, ed. by D. Koller, D. Schuurmans, Y. Bengio, L. Bottou. Advances in Neural Information Processing Systems, vol. 21 (Curran Associates, Inc., 2009)
28. N. Noorshams, *Low Complexity Message-Passig Algorithms for Distributed Computation*, PhD-thesis, EECS Department, University of California, Berkeley (2013)
29. P.P. Shenoy, G. Shafer, Uncertain. Artif. Intell. **4**, 169–198 (1990)
30. Power Systems Test Case Archive-UWEE (1993)
31. Y. Weiss, W.T. Freeman, Neural Comput. **13**, 2173–2200 (2001), ISSN 0899-7667, 1530-888X
32. D. Bickson, (2009) arXiv:0811.2518 [cs, math]
33. M. Cosovic, D. Vukobratovic, IEEE Trans. Power Syst. **34** 648–658 (2019), ISSN 0885-8950, 1558-0679
34. J. Machowski, Z. Lubosny, J.W. Bialek, J.R. Bumby, in *Power system dynamics: stability and control*, 3rd edn. (John Wiley, Hoboken, 2020), ISBN 978-1-119-52634-6
35. O. Rivoire, J. Stat. Mech.: Theory Exp. **2005** P07004–P07004 (2005), ISSN 1742-5468
36. M. Schmidt, D. Aßmann, R. Burlacu, J. Humpola, I. Joormann, N. Kanelakis, T. Koch, D. Oucherif, M. Pfetsch, L. Schewe, R. Schwarz, M. Sirvent, Data **2**, 40 (2017), ISSN 2306-5729
37. T.M. Walski, J. Am. Water Works Assoc. **98**, 110–121 (2006), ISSN 0003150X
38. P.M. Coelho, C. Pinho, J. Braz. Soc. Mech. Sci. Eng. **29**, 262–273 (2007), ISSN 1678-5878
39. D. Brikić, Petrol. Sci. Technol. **29**, 366–377 (2011), ISSN 1091-6466, 1532-2459
40. Behrooz H. Ahmadian, R.B. Boozarjomehry, J. Nat. Gas Sci. Eng. **22**, 551–570 (2015), ISSN 18755100
41. E. Katifori, G.J. Szöllősi, M.O. Magnasco, Phys. Rev. Lett. **104**, 048704 (2010), ISSN 0031-9007, 1079-7114
42. G.K. Batchelor, in *An Introduction to Fluid Dynamics*, 1st edn. (Cambridge University Press, 2000), ISBN 978-0-521-66396-0 978-0-511-80095-5
43. G.S. Williams, A.M. Hazen, in *Hydraulic Tables* (John Wiley, Hoboken, 1905)
44. G.O. Brown, in *The History of the Darcy-Weisbach Equation for Pipe Flow Resistance*. Environmental and Water Resources History (American Society of Civil Engineers, Washington, D.C., 2002), pp. 34–43, ISBN 978-0-7844-0650-2
45. R. Christie, in *Power Systems Test Case Archive* (1993), http://labs.ece.uw.edu/pstca/