Tsinghua Science and Technology

Volume 23 | Issue 3 Article 5

2018

Distributed Energy Sharing in Energy Internet Through Distributed Averaging

Yangyang Ming
the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China.

Jie Yang
the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China.

Junwei Cao
the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China.

Ziqiang Zhou
the Electric Power Research Institute of State Grid, Zhejiang Electric Power Company, Hangzhou 310009, China.

Chunxiao Xing
the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/13 pp233–242
DOI: 10.26599 / TST.2018.9010039
Volume 23, Number 3, June 2018

Distributed Energy Sharing in Energy Internet Through Distributed Averaging

Yangyang Ming, Jie Yang, Junwei Cao*, Ziqiang Zhou, and Chunxiao Xing

Abstract: This paper proposes a distributed averaging iteration algorithm for energy sharing in microgrids of Energy Internet based on common gossip algorithms. This algorithm is completely distributed and only requires communications between neighbors. Through this algorithm, the Energy Internet not only allocates the energy effectively based on the load condition of grids, but also reasonably schedules the energy transmitted between neighboring grids. This study applies theoretical analysis to discuss the condition in which this algorithm can finally reach supply-and-demand balance. Subsequently, the related simulation validates the performance of the algorithm under various conditions.

Key words: distributed averaging; energy sharing; gossip algorithm; Energy Internet

1 Introduction

As a modern energy-utilizing system, Energy Internet[1] effectively combines internet networking technology with highly effective usage of distributed renewable energies through advanced information and communication infrastructure and technologies[2, 3]. With basic characteristics of openness and sharing[4], Energy Internet can enable energy sharing between microgrids, to improve the energy utilizing efficiency. However, the Energy Internet may be unsuitable for a central administration institution sometimes. To maintain high controlling and managing performance, large-scale Energy Internet behaves better by using distributed consensus control strategies. Thus, energy sharing in distributed consensus control has become an important research topic[5]. In this paper, we achieve energy sharing in Energy Internet based on distributed averaging technologies (mainly using gossip algorithms). Through distributed averaging iterations, our designed algorithm can allocate energy in the microgrids according to the energy supply and load demand of all users.

With the development of Energy Internet, the concept of a cyber-physical system is advanced[6] and further develops into cyber-physical integration of infrastructure in Energy Internet[7], which means that the topology of the information network and energy transmission network is isomorphic, and the related facilities of information and energy are co-located and controlled at the same time. The integration of infrastructure for energy and information not only increases the energy-utilizing efficiency but also enhances the control ability of Energy Internet, and finally, this becomes the foundation of the algorithm proposed in this paper.

In many cases, some nodes need to reach a common state in distributed means. This procedure is called "consensus". The consensus algorithm can be used in a dynamically changing environment[8, 9], such as time-varying topology[10, 11], time-varying delays[12], limited bandwidth[13], and quantization[10, 14] or non-quantization. It can reach consensus not only in linear motions[15, 16] and sometimes asynchronously[17, 18]. Some algebraic theories

• Yangyang Ming, Jie Yang, Junwei Cao, and Chunxiao Xing are with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China. E-mail: jcao@tsinghua.edu.cn.
• Ziqiang Zhou is with the Electric Power Research Institute of State Grid, Zhejiang Electric Power Company, Hangzhou 310009, China.
* To whom correspondence should be addressed.
Manuscript received: 2017-06-30; accepted: 2017-11-30

can be used in consensus algorithms such as least mean square[19] and LaSalle's invariance principle[20].

Distributed averaging is a special kind of consensus technology that has been used widely such as in flight or vehicle formation[21], fire hazard control, network time synchronization, and others. It is always executed in distributed means without any centralized control.

Before distributed averaging, every node will have a different initial value and aims to achieve the average value for certain reasons. Through proper communication and calculation between neighbors, all connected nodes reach consensus. The consensus value becomes the same as the average value precisely if neither quantization nor limited-to-integer value exists.

Gossips are typically distributed averaging algorithms[22] and have many types, such as random[23], broadcast[24], and geography gossip[25]. In random gossip, a node communicates with a randomly selected neighbor and they set each other's value to be their average. In geography gossip, the selected neighbor node is not limited to one hop, so averaging of many hops is possible. In broadcast gossip, all neighbor nodes of the target node update their value using the broadcast value they have received. By extending the node's selection range, geography gossip may have faster convergence rate than random gossip. Through broadcast update, broadcast gossip has the fastest convergence value, but this algorithm cannot keep the total value (or the average value) unchanged. Thus, the consensus state may deviate from the average value.

The gossip algorithm can be used to compute some character values, such as sums, averages, random samples, quantiles, and other aggregation functions[26]; this new algorithm has been used in energy sharing.

The rest of this paper is organized as follows. The selection of gossip algorithm is briefly explained in Section 2. In Section 3, the algorithm is described in detail. Related theorems on convergence are proved in Section 4. In Sections 5 and 6, simulation and results, and further discussion are provided. Finally, Section 7 concludes the paper.

2 Gossip Selection

The basic theory of all nodes' convergence to average value is based on stochastic matrices and doubly-stochastic matrices. In stochastic matrices, every element is non-negative and the sum of every row is equal to 1, that is:

$$\boldsymbol{A}\times\mathbf{1}=\mathbf{1},$$

$\boldsymbol{A}$ is the stochastic matrix and $\mathbf{1}$ is an $n\times n$ dimensional vector of all elements equal to one.

With connected fixed topology, this method guarantees the final value being converged. In doubly stochastic matrices, the sum of every row and column is 1, so it satisfies both

$$\boldsymbol{A}\times\mathbf{1}=\mathbf{1} \text{ and } \mathbf{1}^{\mathrm{T}}\times\boldsymbol{A}=\mathbf{1}^{\mathrm{T}}$$

and $A(i,j)\geqslant 0$ and $A(i,j)\leqslant 1$ for any i,j.

This with connected topology and some other proper constraints will not only guarantee convergence, but also ensure converging to the average value. In random and geography gossip, the transfer matrices are doubly-stochastic. However, in broadcast means, the transfer matrix only belongs to the stochastic matrix. Thus, the latter algorithm is not included in the present study.

In this paper, we first use a simple random-gossip algorithm without considering the quantization effect. In this algorithm, one node is randomly selected, and the communication node is selected as the one that has the maximum different value with the selected node. Then, the value of the nodes is averaged. This algorithm is simple and has proved to reach consensus with the average value. Although gossip is the basic algorithm used in our energy sharing algorithm, it is not the research focus of this paper; it is only considered as a basic component. So the algorithm with higher performance can be selected later.

3 Algorithm Description

3.1 Scene description

Based on the forecast of load and energy changes in Energy Internet, some grids may have spare energy, while others may face an energy shortage. Then, the microgrid network can balance the energy demand and supply by using this algorithm. Based on the forecast time scale, the advanced time for calculation can be one day, one hour, or even five minutes. Then, before the time is reached, the energy transmission is scheduled by the algorithm. So far, this algorithm only regulates the total energy transmission in quantities for a period, but real detailed power transmission can also be executed by the same means in the future with the aid of other suitable algorithms, such as that used in demand response (linear regulation).

As we are using the theory of cyber-physical integration in infrastructure for energy and information, the information-handling and energy transmission nodes are located and controlled together (energy router[27]), and

the calculated energy-sharing result can be directly applied to the energy transmission. This method is the foundation of the algorithm proposed in the following section.

3.2 Algorithm description (shown in Fig.1)

(1) Every node (microgrid) forecasts its energy supply and load demand for the next period. Each node can contain an energy-producing unit and/or energy-storage device, and the latter is called a combined node and treated equally with a common node. The energy upper limit is the sum of local energy-producing capacity and energy-storage quantities.

(2) The load of every node is set as the required energy, and the result of energy redundancy is calculated as Eq. (1). The elements in Eq. (1) are both vectors.

$$redundancy = energy - load \tag{1}$$

(3) The values of the nodes that have positive energy redundancy are set to 1, and the negative ones to 0. Then, the distributed averaging algorithm is executed, and the final value *Tavg* is obtained. It is equal to the ratio of positive redundancy node number to the whole node number (shown in Fig. 2).

(4) The range of change factor k (used in Eq. (7), verified by Lemma 3) is calculated as

$$0 < k < 2/Tavg \tag{2}$$

(5) Based on the energy redundancy data, every node changes its value with its neighbors using the selected gossip algorithm to reach consensus. In the process, the energy transmitted along the transmission line during every average action is recorded and summed in every line.

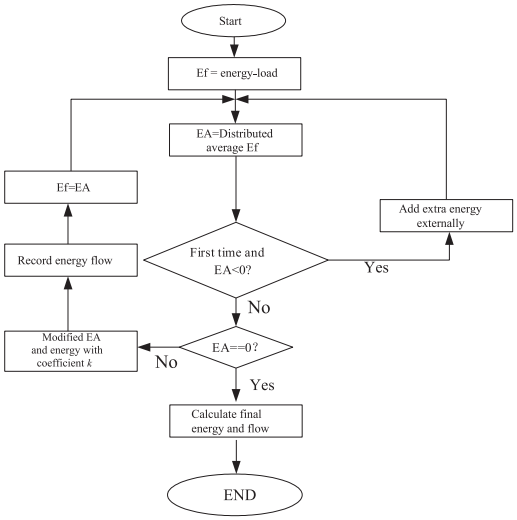

Fig. 1 The proceeding of this algorithm.

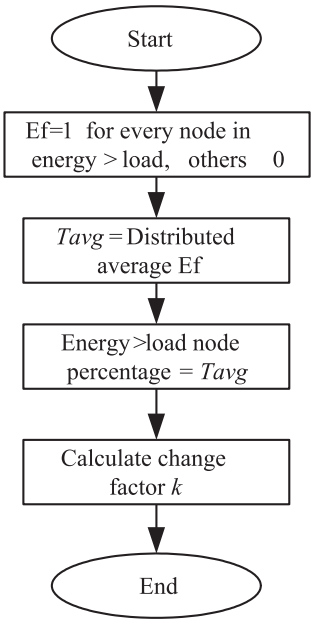

Fig. 2 Calculation of coefficient k.

The related function is
$[flow\_m1, temp\_load] = distributed\_avg(load\_energy, link\_m)$,
where *flow*_m1 represents the transferred energy along the transmission line, *temp_load* is the averaged energy redundancy, *load_energy* is the energy redundancy changed in the last iteration, and *link_m* is the topology matrix of links connecting the grid nodes.

In this function, the variables are changed as
$flow\_m(m,\text{j1}) = flow\_m(m,\text{j1}) + (temp\_energy(1,m) - temp\_energy(1,\text{j1}))/2$;
$flow\_m(\text{j1},m) = flow\_m(\text{j1},m) - (temp\_energy(1,m) - temp\_energy(1,\text{j1}))/2$;
$temp\_energy(1,m) = (temp\_energy(1,m) + temp\_energy(1,\text{j1}))/2$;
$temp\_energy(1,\text{j1}) = temp\_energy(1,m)$;
where *flow_m* is the calculated flow in the average proceeding between node *j* to *m*, $temp\_energy(1,m)$ and $temp\_energy(1,\text{j1})$ are set to the average value between node *j* to *m*.

(6) If this is the first iteration, and the calculated distributed averaging result is less than zero (which means that total *energy* is less than total *load*), then we have to import external energy (mainly from the backbone grid) to ensure non-negative energy redundancy. The related initial values are changed accordingly and the algorithm is re-performed.

(7) In other situations, we modify the energy and energy redundancy according to the distributed averaging result for energy redundancy and the change factor k. If

the average result is equal to zero, then the supply and demand is balanced, so the iteration is terminated. If the average result is positive, then we reduce the energy and energy redundancy on selected nodes; otherwise, if the average is negative, then we increase the energy and energy redundancy on selected nodes. The selected nodes are only limited to nodes with *energy*>*load* and satisfy other constraints, thereby ensuring the convergence of the algorithm. The related code is

$if\ temp\_load(1,i) > 0\ \&\&\ load\_energy(1,i) > 0\ \&\&$
$energy\_m(1,i) > 0,$
$energy1 = energy\_m(1,i);$
$energy\_m(1,i) = max((energy\_m(1,i) -$
$temp\_load(1,i) \times k), 0);$
$temp\_load(1,i) = temp\_load(1,i) - (energy1 -$
$energy\_m(1,i));$
$else\ if\ temp\_load(1,i) < 0\ \&\&\ load\_energy(1,i) >$
$0\ \&\&\ energy\_m(1,i) < load\_m(1,i),$
$energy1 = energy\_m(1,i);$
$energy\_m(1,i) = min((energy\_m(1,i) + temp\_load,$
$(1,i) \times k), load\_m(1,i));$
$temp\_load(1,i) = temp\_load(1,i) + energy\_m(1,i) -$
$energy1;$
end.

In the preceding code, *temp_load* is the average energy redundancy result, *load_energy* is the initial energy redundancy, *energy_m* is the energy allocation result in the last iteration, and *load_m* is the initial load in every node.

In the preceding code, when energy redundancy is larger than zero (with other proper constraints), if *energy_m* minus *temp_load* is more than zero, then set *energy_m* to be equal to the minus result; otherwise, set *energy_m* to zero, and subtract the changed energy $(energy1 - energy\_m(1,i))$ from *temp_load*. Similarly, when energy redundancy is less than zero (with other proper constraints), and if *energy_m* plus *temp_load* is no more than *load_m*, then set *energy_m* to be equal to the result of addition; otherwise, set *energy_m* to *load_m* and add the changed energy $(energy\_m(1,i) - energy1)$ to *temp_load*.

(8) Based on modified energy redundancy, the distributed averaging algorithm is executed for the next round. The iteration is stopped when distributed averaging result reaches very close to zero, and the balance is reached.

(9) When the algorithm terminates, the recorded flow in every iteration and every transmission line is summed, and the energy supply of every grid is also calculated using the following code:

$flow\_m = flow\_m + flow\_m1;$
energy_m=*energy_m*−*load_energy,*

where $flow\_m1$ is the changed energy flow in the last iteration, and *energy_m* is the final energy provided by every node.

4 Proof of Related Theories

Some theories on the algorithm's convergence characteristics are listed and proven as follows.

Lemma 1 If the total energy is more than total load initially, and $k = 1$, then the algorithm finally converges to zero.

Proof If the total energy is more than the total load, positive energy redundancy nodes with number n always exist, and $0 < n \leqslant totalnodenumber$. Then, if $k = 1$ and through step 7 in Section 3.2, the energy redundancy in every iteration remains non-negative (below the energy redundancy result calculated in the k-th iteration is noted as $result[k]$). At the same time, we can easily prove that $result[k] > result[k+1] \geqslant 0$ if $result[k] != 0$.

Thus, based on the limit theory, the algorithm finally converges to zero. ■

Lemma 2 If the total energy is more than the total load initially, then when $k < 2$, the algorithm finally converges to zero.

Proof We can obtain that the first calculated energy redundancy $result[1] > 0$, and then the proof can be divided into two cases:

Case1: If $result[i] \geqslant 0$ for all i, as in Lemma 1, we can obtain $result[k] > result[k+1]$ if $result[k] > 0$, and finally the algorithm converges to zero.

Case 2: Otherwise, $result[i] < 0$ exists for some *i*.

As the first calculated energy redundancy $result[1] > 0$, there will always be $result[i+k] \geqslant 0\ (k \geqslant 1)$, after $result[i] < 0$ for any *i*.

When $k < 2$, we can easily prove that:

(1) If $result[i] < 0$ and $result[i+1] < 0$ for any possible iteration, then $|result[i]| > |result[i+1]|$.

(2) If $result[i] > 0$ and $result[i+1] < 0$ for any possible iteration, then $|result[i]| > |result[i+1]|$.

(3) If $result[i] < 0$ and $result[i+1] > 0$ for any possible iteration, then $|result[i]| > |result[i+1]|$.

(4) If $result[i] > 0$ and $result[i+1] > 0$ for any possible iteration, then $|result[i]| > |result[i+1]|$.

Based on this foundation, $|result[i]| > |result[i+1]| \geqslant 0$ can be obtained for all conditions, so this algorithm will converge to zero.

From Lemma 2, this algorithm works for any initial conditions with $k < 2$. ■

Lemma 3 If the total energy is more than the total load initially, then a possible range of k for which the iteration converges, is $(0, 2/Tavg)$.

Proof When $|result[i]| > |result[i+1]|$, the calculated energy redundancy converges to 0. We set

$$|result[i+1]| = |result[i]| + |delta[i+1]|.$$

When $result[i] > 0$, then $delta[i+1] < 0$, and if $|result[i]| > 0.5 \times |delta[i+1]|$, we can obtain $|result[i]| > |result[i+1]|$.

Similarly, when $result[i] < 0$, then $delta[i+1] > 0$, and if $|result[i]| > 0.5 \times |delta[i+1]|$, we also get $|result[i]| > |result[i+1]|$. Thus, a sufficient condition for convergence is

$$|result[i]| > 0.5 \times |delta[i+1]| \tag{3}$$

Propose that the initial positive redundancy result node number is equal to $Tavg \times N$ (*N* is total node number). Through Eq. (3), we have the sufficient condition (summed together):

$$|N \times result[i]| > 0.5 \times sum|delta[i+1]| \tag{4}$$

because

$$|Tavg \times N \times K \times result[i]| \geqslant \sum |delta[i+1]| \tag{5}$$

We can set a more strict condition, which is

$$|N \times result[i]| > 0.5 \times |Tavg \times N \times K \times result[i]| \tag{6}$$

As the number of nodes to be processed (changing redundancy) in every iteration is always no more than $Tavg \times N$, Eq. (6) becomes a sufficient condition for the convergency of the algorithm.

Then we can prove if

$$Tavg > 0 \text{ and } K < 2/Tavg \tag{7}$$

the energy redundancy converges to 0.

As $Tavg$ is always $\leqslant 1$, Lemma 2 is a special case of Lemma 3. ■

5 Simulation and Result

5.1 Topology set up

We randomly create a cluster of microgrids on Energy Internet with 7 nodes (microgrids). The simplified topology is shown in Fig. 3 ($G = (V, E)$). The circle nodes

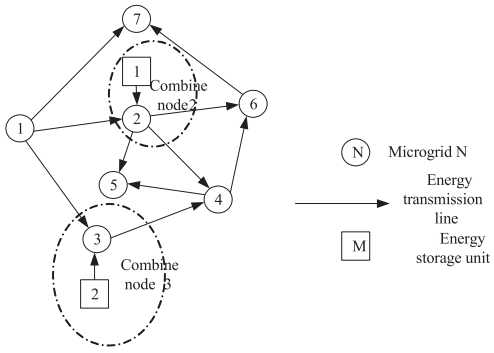

Fig. 3 Microgrid topology.

($\in V$) represent microgrids, the square nodes ($\in V$) represent extra energy storage units (forming the combined node with the circle nodes), and the lines ($\in E$) represent the energy transmission lines between the nodes. These lines can be bidirectional in energy transmission. To run the energy-sharing algorithm, we set up the corresponding connection topology matrix ($link\_m$) for this cluster. The nodes are notated from 1 to *N*, and the size of the connection matrix is 7×7.

If a transmission line connects nodes *i* and *j*, then set

$$link\_m[i, j] = link\_m[j, i] = 1.$$

Otherwise, the other elements are set to 0. The $link\_m$ of Fig. 3 is

$$\begin{bmatrix} 0,1,1,0,0,0,1 \\ 1,0,0,1,1,1,0 \\ 1,0,0,1,0,0,0 \\ 0,1,1,0,1,1,0 \\ 0,1,0,1,0,0,0 \\ 0,1,0,1,0,0,1 \\ 1,0,0,0,0,1,0 \end{bmatrix}.$$

To record the energy flow between the connected nodes, we define the positive direction of the flow as from a node with a small number to that with a large number, and a negative value represents the opposite direction.

Before simulation, we have to assign the energy supply and load demand to every node (as shown in Fig. 4). In the process, we can add a priority level to every microgrid. If important devices are in the microgrid, the priority becomes high, and the grid can set energy redundancy in the initial value set if any energy storage device exists.

Otherwise, the priority may be low and without energy redundancy. At the same time, the needs of demand response or demand side management can be considered accordingly by modifying the initial energy and load value.

To simplify the algorithm, we set the initial total energy supply to be more than the initial total load demand, which

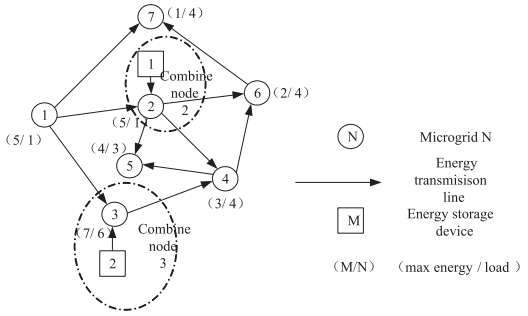

Fig. 4 Initialenergy/load set.

does not change the substance of the algorithm.

The initial energy vector is [5, 5, 7, 3, 4, 2, 1] and the load vector is [1, 1, 6, 4, 3, 4, 4], where the i-th value of the two vectors corresponds to the value of node i.

5.2 Gossip algorithm test

We have to test the convergence performance of the gossip algorithm first. We set the load demand as $rand(1,7)\times 100$ and energy supply as $rand(1,7)\times 100+rand(1,7)\times \mathrm{k1}$. The coefficient k1 is set as 0, 20, 50, 100, 200 individually. Then, we calculate the distributed averaging value of *energy supply minus load demand.*

We also calculate the averaged abs_error in every iteration as

$$\begin{aligned} &n=|initial\_value|_0,\\ &avg\_value=sum(initial\_value)/n,\\ &abs\_error=sum(abs(value-avg\_value))/n \end{aligned} \tag{8}$$

where $|initial\_value|_0$ represents the number of vector $initial\_value$ and vector *value* represents the averaging result of $initial\_value$ in every iteration.

When $abs\_error<k$, the iteration terminates. We can determine if k remains unchanged; although k1 changes significantly, the ranges of results are similar, thereby easing the simulation. If we set $k=0.1$, all the simulation examples converge to the average value before approximately 48 iterations. If we set $k=0.01$, the iteration round number is approximately 70. If we set $k=0.001$, then the value is approximately 95. Figures 5–8 shows some results of statistic histogram for every 100 iterations. The horizontal axis represents iteration turns and the vertical axis represents the corresponding count numbers.

5.3 Simulation and result

First, we calculate the $Tavg$ in Step 3 in Section 3.2 using the gossip algorithm (input vector: [1, 1, 1, 0, 1, 0, 0]). We can obtain the result vector [0.5714, 0.5714, 0.5714, 0.5714, 0.5714,

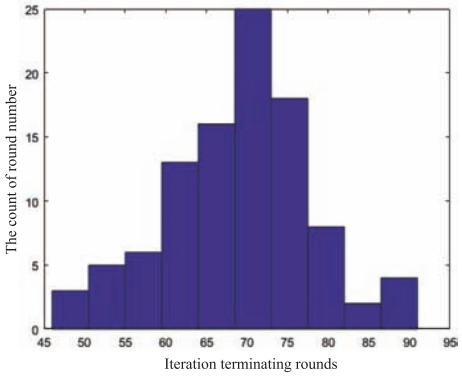

Fig. 5 The histogram of iteration round number for k=0.001, k1=0.

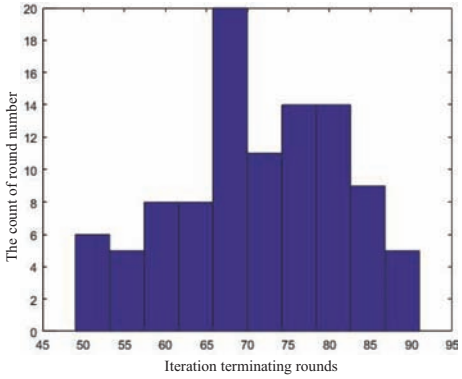

Fig. 6 The histogram of iteration round number for k=0.001, k1=20.

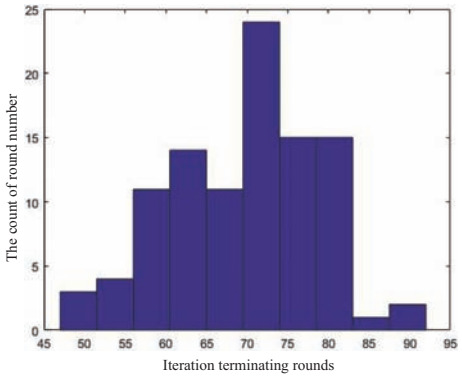

Fig. 7 The histogram of iteration round number for k=0.001, k1=50.

0.5714, 0.5714] for 100 rounds, which are equal to the ratio of the initial number of positive energy redundancy nodes to the whole node number (4/7) in the above topology, thereby validating Step 3.

Before running the algorithm, we first set the change factor k as $Tavg=4/7$. According to Lemma 3, we can determine if k is less than 3.5, and the correct result is derived (however, this is not a necessary condition). Thus, we set the k as [1, 1.2, 1.5, 1.6, 1.8, 2, 3, 3.4] and run them in turn (k less than 1 is unnecessary; it only reduces the convergence rate).

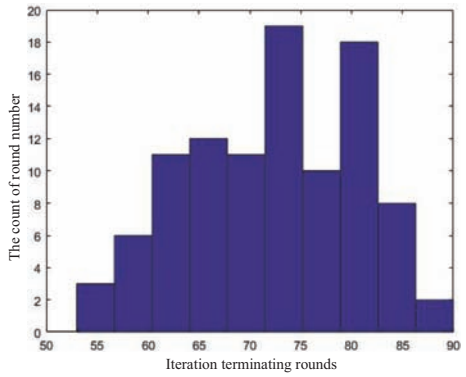

Fig. 8 The histogram of iteration round number for k=0.001, k1=200.

We run the energy-sharing algorithm on the designed topology, and on that with one node deleted or one edge deleted. All scenes show the desired results (Figs. 9–13).

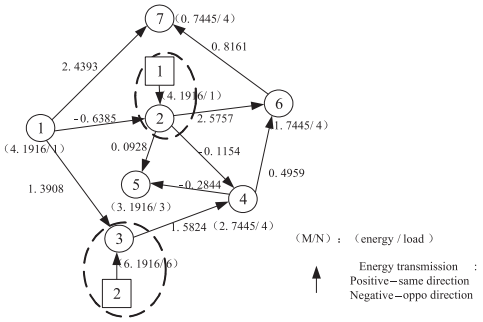

Fig. 9 Normal state simulation, k=1.2.

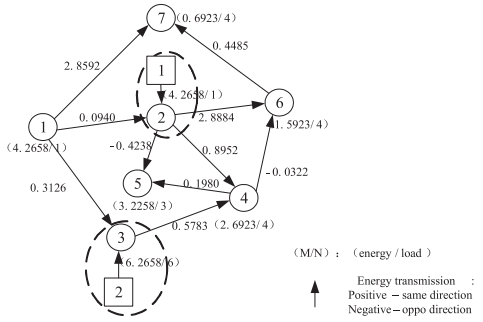

Fig. 10 Normal state simulation, k=1.

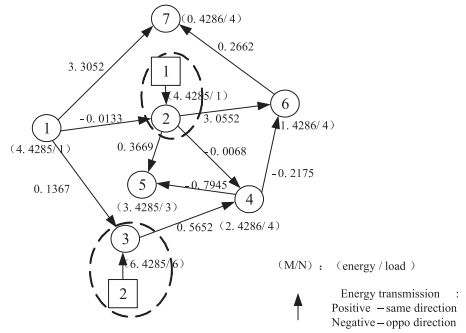

Fig. 11 Normal state simulation, k=2.

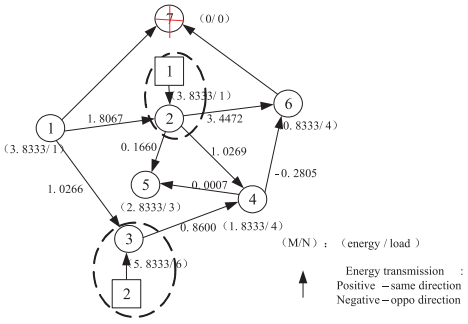

Fig. 12 Node failure state simulation, k=2.

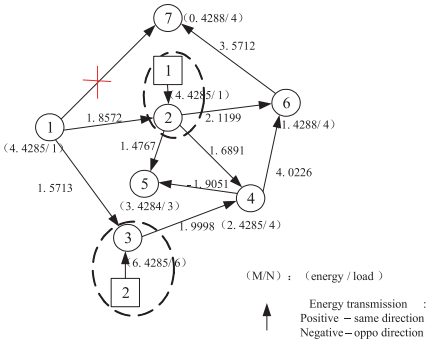

Fig. 13 Line failure state simulation, k=2.

From the preceding results, we can observe that the total energy assigned for every node is the algebraic sum of the self-providing energy and energy transmitted from neighbor nodes (added), and energy transmitting to neighbor nodes (subtracted), which is equal to the load of every node.

As shown, if the total initial energy is larger than the total initial load, the final energy value is always less than the maximum energy, which can be provided by every node; in other words, spare energy exists in every node. This is a special advantage for energy sharing in microgrid networks because this energy redundancy can be used to face an emergency situation or charge the energy storage unit, thereby improving the stability of the entire network and improving the robustness of the microgrid network.

The simulation also shows the convergence phenomenon for various change factors (k) in Fig. 14. We can observe that when the change factor is small, the energy redundancy monotonically decreases. As the change factor increases, the value of the first iteration decreases and some fluctuations occur around the zero axis when $k > 1.6$, which coincides with our theoretical analysis. That is, if $k \geqslant 1/Tavg$, then a negative result with high probability will emerge. All iterations end before seven turns. The instance with negative results often converges faster than only positive ones except $k = 1$.

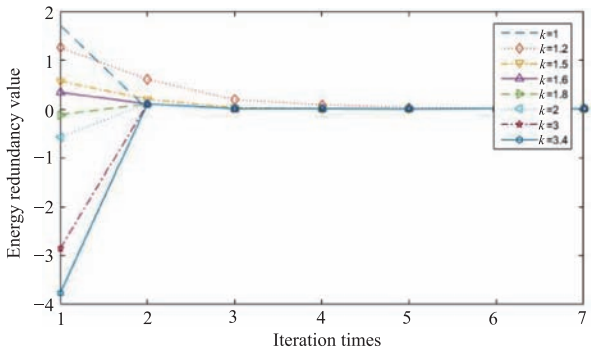

Fig. 14 Energy redundancy in every iterations with different k.

Through in-depth observation and deduction, we can find that if the initial condition (demand and supply) and the change factor in two simulations are the same, then the final energy allocation in every grid will also be the same, but the energy transmitted between the grids will be different. At the same time, if the initial energy and load are equal in a pair of nodes (such as nodes 1 and 2), then the final energy load of this pair will be equal, as proven by theoretical analysis.

These two cases can be observed for all $k > 1$. The only exception is in $k = 1$, which may be due to the effect of averaging residue of gossip in the positive node selection scheme, thereby influencing subsequent handling results. However, the algorithm still converges and the result is reasonable. Thus, we can analyse the performance of the algorithm in every instance by only one example.

Excluding exception $k = 1$ (although it has the same trend listed below), the final result is reported in Table 1.

The (+) sign indicates that the initial supply is more than the initial demand in this node, and the (-) sign indicates the opposite condition.

From the table combined with the result of Fig. 11, we can observe that if $k < 1.8$ (energy redundancy is monotonically decreasing), with the factor k increasing, the energy provided by nodes with (+) sign monotonically decreases, while those with (-) sign monotonically increase (equally, the variance of vector *energy-load* for all nodes decreases with k. Otherwise, if $k \geqslant 1.8$ (with energy redundancy fluctuation) and the factor k changes, the energy allocation remains almost unchanged. This result can lead us to properly select the change factor k to satisfy the extra energy demand.

6 Further Discussion

6.1 Executing gossip in parallel

Our gossip algorithm can be executed in parallel, when two neighbor nodes agree to average, and it can be executed in parallel with other node executions. When two nodes select the same neighbor node, a simple “OK” or “reject” would be enough because the process mainly uses wired communication.

6.2 Communication and bandwidth constraint

As the communication media between microgrids mainly uses fiber optics, the topological change becomes slow or constant, and the communication power and bandwidth constraint is not as tight as the wireless channel. Thus, no quantization is needed. Furthermore, the necessary characteristics of the topology can be very simple (i.e., only a proper connection is required).

6.3 Gossip algorithm termination condition

When the gossip algorithm is executed in distributed means, every node should know when to terminate the iteration. Here, every node can broadcast its value to its one-hop neighbors if it changes. When the node finds that its value and its neighbors’ value remain unchanged for a limited period, then it can terminate the iteration locally.

6.4 Distributed calculating and central processing

Although the energy-sharing algorithm uses distributed computing technologies, it can also be used in central processing. If central processing units exist, they can

Table 1 Energy allocation for different k and nodes.

k	Node						
	1(+)	2(+)	3(+)	4(-)	5(+)	6(-)	7(-)
1.2	4.1916	4.1916	6.1916	2.7445	3.1916	1.7445	0.7445
1.5	4.0976	4.0976	6.0976	2.8698	3.0976	1.8698	0.8698
1.6	4.0703	4.0703	6.0703	2.9062	3.0703	1.9062	0.9062
1.8	4.4286	4.4286	6.4286	2.4286	3.4286	1.4286	0.4286
2	4.4285	4.4285	6.4285	2.4286	3.4285	1.4286	0.4286
3	4.4286	4.4286	6.4286	2.4286	3.4286	1.4286	0.4286
3.4	4.4286	4.4286	6.4286	2.4286	3.4286	1.4286	0.4286

gather every grid's energy and load forecast data and use the iteration algorithm to calculate the energy allocation and obtain the energy transmitted along every lines. The central processing needs only one iteration by setting the total supply as equal to the total demand.

6.5 State renewing

As the energy sharing is linear, when the energy and load changes, we can use either enhanced learning or an update calculation to recalculate the network energy-sharing result. The two means are the same in substance, and the enhanced learning should also satisfy the energy constraint.

7 Conclusion

In this paper, a completely distributed iteration algorithm is designed for energy sharing in a typical Energy Internet situation. Through a properly designed algorithm process and change factor selection, the allocated energy and load requirement are finally balanced and the proof validates the performance of the algorithm. The algorithm is tested in different conditions and reasonable results are observed. Future research will consider the real constraints on energy transmission (such as capacity constraint on the transmission line) or on satisfying special transmission demands by grid owners (such as demand-side management). Furthermore, the real performance between different gossip algorithms will be evaluated and compared.

Acknowledgment

This study was partly supported by the National Natural Science Foundation of China (No. 61472200), Beijing Municipal Science and Technology Commission (No. Z161100000416004), and the project of Blockchain Application Research on Energy Internet (No. 52110417000G).

References

[1] J. W. Cao and M. B. Yang, Energy Internet–Towards smart grid 2.0. presented at the 4^{th} Int. Conf. Networking and Distributed Computing, Los Angeles, CA, USA, 2013.

[2] J. W. Cao, Y. X. Wan, G. Y. Tu, S. Q. Zhang, A. X. Xia, X. F. Liu, Z. Chen, and C. Lu, Information system architecture for smart grids, (in Chinese), *Chin.J. Comput.*, vol. 36, no. 1, pp. 143–167, 2013.

[3] A. Q. Huang, M. L. Crow, G. T. Heydt, J. P. Zheng, and S. J. Dale, The future renewable electric energy delivery and management (FREEDM) system: The Energy Internet, *Proc. IEEE*, vol. 99, no. 1, pp. 133–148, 2011.

[4] Z. Y. Dong, J. H. Zhao, F. S. Wen, and Y. S. Xue, From smart grid to Energy Internet: Basic concept and research framework, (in Chinese), *Autom. Electric Power Syst.*, vol. 38, no. 15, pp. 1–11, 2014.

[5] J. Rifkin, *The Third Industrial Revolution*, (in Chinese). T. W. Zhang and Y. N. Sun, trans. Beijing, China: China Citic Press, 2012.

[6] Y. X. Wan, J. W. Cao, S. Q. Zhang, G. Y. Tu, C. Lu, X. T. Xu, and K. Q. Li, An integrated cyber-physical simulation environment for smart grid applications, *Tsinghua Sci. Technol.*, vol. 19, no. 2, pp. 133–143, 2014.

[7] J. W. Cao, M. B. Yang, D. H. Zhang, K. Meng, Z. Chen, and C. Lin, Energy Internet: An infrastructure for cyber-energy integration, (in Chinese), *Southern Power Syst. Technol.*, vol. 8, no. 4, pp. 1–10, 2014.

[8] A. Olshevsky and J. N. Tsitsiklis, Convergence rates in distributed consensus and averaging, presented at the 45^{th} IEEE Conf. Decision and Control, San Diego, CA, USA, 2006.

[9] M. Cao, A. S. Morse, and B. D. O. Anderson, Reaching a consensus in a dynamically changing environment: Convergence rates, measurement delays, and asynchronous events, *SIAM J. Control Optim.*, vol. 47, no. 2, pp. 601–623, 2008.

[10] S. Kar and J. M. F. Moura, Distributed consensus algorithms in sensor networks: Quantized data and random link failures, *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1383–1400, 2010.

[11] Y. Hatano and M. Mesbahi, Agreement over random networks, *IEEE Trans. Automat. Control*, vol. 50, no. 11, pp. 1867–1872, 2005.

[12] F. Xiao and L. Wang, Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays, *IEEE Trans. Automat. Control*, vol. 53, no. 8, pp. 1804–1816, 2008.

[13] T. Li, M. Y. Fu, L. H. Xie, and J. F. Zhang, Distributed consensus with limited communication data rate, *IEEE Trans. Automat. Control*, vol. 56, no. 2, pp. 279–292, 2011.

[14] T. C. Aysal, M. J. Coates, and M. G. Rabbat, Distributed average consensus with dithered quantization, *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4905–4918, 2008.

[15] W. W. Yu, G. R. Chen, M. Cao, and J. Kurths, Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 40, no. 3, pp. 881–891, 2010.

[16] P. Lin and Y. M. Jia, Consensus of second-order discrete-time multi-agent systems with nonuniform time-delays and dynamically changing topologies, *Automatica*, vol. 45, no. 9, pp. 2154–2158, 2009.

[17] M. Cao, A. S. Morse, and B. D. O. Anderson, Agreeing asynchronously, *IEEE Trans. Automat. Control*, vol. 53, no. 8, pp. 1826–1838, 2008.

[18] C. C. Moallemi and B. Van Roy, Consensus propagation, *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4753–4766,

2006.

[19] L. Xiao, S. Boyd, and S. J. Kim, Distributed average consensus with least-mean-square deviation, in *Proc. 17th Int. Symp. Mathematical Theory of Networks and Systems*, Kyoto, Japan, 2006, pp. 1–9.

[20] D. Z. Cheng, J. H. Wang, and X. M. Hu, An extension of LaSalle's invariance principle and its application to multi-agent consensus, *IEEE Trans. Automat. Control*, vol. 53, no. 7, pp. 1765–1770, 2008.

[21] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control*. London, UK: Springer, 2008.

[22] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, On distributed averaging algorithms and quantization effects, *IEEE Trans. Automat. Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[23] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, Randomized gossip algorithms, *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[24] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, A. Scaglione, Broadcast gossip algorithms for consensus, *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2748–2761, 2009.

[25] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, Geographic gossip: Efficient averaging for sensor networks, *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1205–1216, 2008.

[26] D. Kempe, A. Dobra, and J. Gehrke, Gossip-based computation of aggregate information, presented at the 44th Ann. IEEE Symp. Foundations of Computer Science, Cambridge, MA, USA, 2003, pp. 482–491.

[27] Y. Xu, J. H. Zhang, W. Y. Wang, A. Juneja, and S. Bhattacharya, Energy router: Architectures and functionalities toward Energy Internet, presented at the 2nd IEEE Int. Conf. Smart Grid Communications, Brussels, Belgium, 2011, pp. 31–36.

Yangyang Ming received the bachelor's degree from Nanjing University of Posts and Telecommunications in 2004, the master's degree from Chongqing University of Posts and Telecommunications in 2007, and the PhD degree from Beijing University of Posts and Telecommunications in 2012. He is a postdoctoral researcher at Tsinghua University now. His research area is communication and information systems.

Jie Yang obtained the bachelor's degree from Jinan University in 2007, master's degree from Beijing Institute of Technology in 2010, and PhD degree from the same university in 2015. Since 2016, she serves in Tsinghua University as a Postdoctoral Researcher. Her research area is automation control.

Junwei Cao obtained the bachelor's degree from Tsinghua University in 1996, master's degree from the same university in 1998, and PhD from University of Warwick in 2001. From 2002 to 2006, he worked at the NEC European Laboratory in Germany and the MIT/LIGO laboratory in the United States. He has been working at Tsinghua University since 2006 as a Professor and Vice Dean of the Research Institute of Information Technology. He has published more than 200 academic papers and 8 books. His research area is advanced computing technology and application.