Heuristic algorithms for learning of random neural network and energy distribution of energy packet networks

Student: Yonghua Yin,
Supervisor: Professor Erol Gelenbe

Intelligent Systems and Networks Group, Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT, UK

1. Introduction

This section begins with the literature reviews of two areas, i.e., the random neural network and energy network, which illustrates the related problems, general frameworks and existing alternatives. Then, this section concludes what has been done in this report contributing to these two areas.

1.1. Random neural network

Random neural network (RNN) is a novel biophysically inspired model proposed by Gelenbe in [44], which has been applied in various areas [1, 2, 45, 46]. The neurons in the RNN are spiking neurons, where each one can fire unit amplitude spikes to each other, simulating the brain spiking behaviors. Compared to other neural network models, e.g., the conventional multi-layer perceptron (MLP) [3], extreme learning machine (ELM) [4, 5, 53] and orthogonal polynomial network (OPN) [6, 54], the RNN is a closer model to the biological neuronal network and can better represent how signals are transmitted in the human brain. Each neuron is represented by its potential that is a non-negative integer showing how many customers are there in the queue (the neuron) [44] (Here we use the concepts "customers" and "queue" since the RNN is proposed based on the theory of queueing networks). In [47], Gelenbe presented important results about the probability that the potential of a neuron is positive. The first result is that, under certain conditions, the probability converges to a steady-state value. The second one is that, the steady-state probability function of potentials of all neurons is the product of all marginal-probability functions of neuron potentials. These results make the steady-state probability distribution of the RNN a system of analytical equations, where the system of equation can be efficiently computed simply by iterations without using numerical methods such as the Newton method [7] and therefore the probability distribution can be obtained directly without using the Monte Carlo method [8].

There are three main approaches to implement the RNN model [10]. The most accessible approach is to use computer software tools based on CPU to simulate the RNN spiking behaviors or conduct the RNN equations computations. However, this approach can be of low efficiency because it does not utilize the fact that the RNN is an extremely distributed model and can be computed or simulated in an extremely parallel manner. The second approach is still to implement the RNN in computers but in a parallel manner. Specifically, the computations related to the RNN can be conducted simultaneously by a very large number of simple processing elements, where the rapid development of chip multiprocessor (CMP) and Graphics Processing Unit (GPU) and the improvement of the parallel computation supported languages, e.g., the widely-used MATLAB language [14–16] and the domain-specific language for machine learning (OptiML) [16], make it more accessible. The third approach is to implement the RNN model in hardware using special-purpose digital or analog components [10]. Via specially designed hardware, the high parallelism of the RNN model can be fully utilized and the speed in applying the RNN would achieve a large increase. For the first approach, the work by Abdelbaki [9] implemented the RNN model in the MATLAB software. To the best of the authors' knowledge, there is still no investigation to implement the RNN using the second approach. In the level of hardware, another work of Abdelbaki in [10] shows that the RNN can be implemented by a simple continuous analog circuit that contains only the addition and multiplication operations. In addition, the work by Cerkez in [11] proposed a digital realization for the RNN using discrete logic integrated circuits. The work by Kocak in [12] implemented the RNN-based routing engine of the cognitive packet networks (CPN) in hardware and decreased significantly the number of weight terms (from $2n^2$ to $2n$) needed to be stored for each RNN in the engine, where the CPN is an alternative to the IP-based network architectures proposed by Gelenbe [13] which is outside the scope of this report and will not be discussed further here.

Many (or say, most) practical applications of artificial neural networks require solving supervised learning problems [17], which is an important subject in machine learning and more generally, artificial intelligence [18]. In general, supervised learning of a system requires it to learn a mapping from the input data (that can be integer and real numbers) to the output data (that can be labels and real numbers) in a given training dataset. In the RNN case, it has been applied in encoding an image into fewer bits in order to achieve image compression by using a RNN with a feed-forward architecture to learn the mapping from the image to the same image through a narrow passage with fewer hidden-layer neurons [19], where the extension to video compression applied the similar principles [20, 21]. The work in [22] exploited multi RNNs to learn from training data and recognize shaped objects in clutter by combining results of all RNNs. Gelenbe [23] used a feed-forward RNN to learn a mapping from the scanning electron microscopy intensity waveform to the

Email addresses: y.yin14@imperial.ac.uk (Student: Yonghua Yin), e.gelenbe@imperial.ac.uk (
Supervisor: Professor Erol Gelenbe)

cross-section shape that is also called the profile such that the profile can be reconstructed from the intensity waveform and the destruction caused by acquiring a cross-section image can be avoided. The work by Oke in [24] first formulated the task of Denial of Service (DoS) detection as a pattern classification problem, where one important step is to measure and select useful input features, and then exploited the RNNs with both the feed-forward and recurrent architectures to fulfill the task. Gelenbe proposed a RNN model with multiple classes in [29] as a generalization following the work of the single-class RNN [44]. This multiple-class RNN can be a mathematical framework to represent networks which simultaneously process information of different types. Then, by setting the excitatory input rates and the desired outputs of the neuron states as the normalized pixel values, this multiple-class RNN is utilized to learn a given color texture and generate a synthetic color texture that imitates the original one [30]. The earlier work in [28] that used the single-class RNN for black and white texture generation utilized the similar principle. The RNN learning has also been applied to image segmentation [25], vehicle classification [26] and texture classification and retrieval [27]. More applications related to utilizing the learning capability of the RNN can be seen from the critical review in [1]. The RNN has also already been applied in many other areas that can be seen from the reviews in [45, 46] covering the RNN applications until 2010, which is not the main focus of this report and will not be discussed further.

Many investigations have been done to develop the learning methods for the RNN. In [47], Gelenbe developed a gradient-descent learning algorithm for the RNN, which seeks nonnegative weights in terms of the gradient descent of a quadratic cost function and is the most widely-used learning algorithm for the RNN. In the further work of Gelenbe [29], this gradient learning algorithm is also extended to the multiple-class RNN and applied to color texture generation (as formerly mentioned). An inspiring and important work done by Gelenbe and Mao in [113] proved that a specific type of multi-variable polynomials can be mapped into the RNN and the RNN is a universal function approximator. This work sets up a bridge between the RNN and another type of feed-forward polynomial-based neural networks (PNN) [6, 31–35, 54] whose activation functions are constructed based on the product form of multiple polynomials. This type of neural networks (the PNN) has been applied to multi-variable function approximation and pattern classification and achieved excellent performance. In addition, many structure-selection methods have been developed and investigated to determine the structures of the PNN, including the number of hidden neurons, the choices of neurons to constitute the network (this is necessary since each neuron in the PNN is different), according to the information extracted from the training data. What is more, the training of the PNN is extremely fast since the learning (or say, training) problems of them can be formulated as a standard least-square problem whose solutions can be obtained directly by the formal pseudoinverse method [4–6, 31–34, 53, 54] and variations of the pseudoinverse method [4] based on the size of training datasets. The training of the PNN can also be achieved using the slower back-propagation (BP) method [54]. It also means that there is no local minimal in the learning of the PNN regardless of the number of dimensions because of the fact that the least-square problem itself is strictly convex and can be solved to optimality. It is well known that gradient-descent algorithms including the one for the RNN [47] may suffer from the weaknesses of local minimal [37] and slow convergence [36]. To the best of the authors' knowledge, in the learning of the RNN, the number of the neurons is usually set as the number of problem dimension or a number related to that, and there is still no (not much) literature investigating deeply the selection of the number of neurons. Taking advantage of the results of the PNN, especially the structure-selection methods and the learning methods, can achieve the improvements in terms of local minimal, slow convergence and structure determination for the learning of the RNN. The basic idea is that an PNN is first constructed with its structure selected and trained based on the related methods developed in [6, 31–35, 54], and then mapped into the RNN. Realizing this idea to the RNN will activate the potential of the RNN in the area of function approximation and pattern classification, since the tool will possess both the excellent approximation and prediction capabilities from the PNN and the high parallelism from the RNN.

In addition to the gradient-descent algorithm, alternatives for the learning of the RNN have been proposed. The work by Likas [38] trained the random neural network using quasi-Newton methods (including the BFGS and DFP methods), which utilizes the derivative results of gradient descent in [47]. It tested the proposed algorithm for the RNN with two different architectures, where the fully recurrent architecture yielded poor results while the feed-forward architecture yielded smaller errors with less steps than the pure gradient-descent algorithm. However, it is also reported that the proposed algorithm has higher computational complexity and can not be applied to online learning. Another alternative exploiting the Levenberg-Marquardt optimization procedure and adaptive momentum is proposed by Basterrech [39] based on the gradient-descent algorithm. Its performance was evaluated on several problems. Compared to the gradient-descent algorithm, it achieved faster convergence speed in almost all cases but less robust in solving the XOR problem. The RPROP learning algorithm introduced in [41], which performs a direct adaptation of the weight step based on the local gradient information, is combined with the pure gradient-descent algorithm for the learning of the RNN [39]. Georgiopoulos gave a brief description and some critical comments on this RPROP algorithm in [1]. It is reported in [39] that, in the learning of the geometrical images, this RPROP algorithm outperformed the gradient-descent algorithm. Georgiopoulos also applied the adaptive inertia weight particle swarm optimization (AIW-PSO) [42] and

differential evolution (DE) approach [43] for the RNN learning in [1]. Then, the performances of the gradient-descent algorithm, RPROP algorithm, AIW-PSO and DE approach for classifying 12 datasets were compared, and there is no absolute winner among them.

All the above mentioned alternatives require solving the RNN system of equations in each iteration of the search process and most of them are developed utilizing the derivative results of gradient descent in [47] meaning that they require the computations of the derivatives of the objective function with respect to each RNN weight, while the alternative proposed by Timotheou thought of the learning problem of the RNN from a different perspective [48, 49]. Specifically, instead of solving the learning problem of the RNN directly, the RNN system of equations associated with learning is approximated to obtain a non-negative least-square (NNLS) problem. This NNLS problem is strictly convex and can be solved to optimality. The solution to the NNLS problem obtained by the algorithm developed in [48, 49] is directly exploited as the one to the learning problem of the RNN. It is reported in [48, 49] that this algorithm achieved better performance than the gradient-descent algorithm in a combinatorial optimization problem emerging in disaster management. Note that the solution space of the NNLS problem is different from that of the learning problem. Specifically, the cost function used in the NNLS problem (denoted by f) is different from that of the learning problem (denoted by E), and the solving process of the NNLS problem searches for the optimal solution (i.e., the weights) that achieves the minimal f to best approximate the RNN system of equations. However, when these weights found by solving the NNLS problem are substituted into the RNN system of equations, it is not certain that the steady-state probabilities with this RNN system can produce a small E. Although many numerical results (as shown in Section 2) have verified that the optimal solution to the NNLS problem is generally a good solution to the learning problem, there is no rigorous theoretical basis to guarantee the performance of this algorithm in handling the learning problem. Thus, we call this algorithm a heuristic algorithm to the RNN learning. There are two main advantages of this heuristic algorithm. The first advantage is that it can be operated very fast since it requires neither solving the RNN system of equations nor computing the gradient descent in the whole search process. The heuristic algorithm in [48, 49] may not be able to find the optimal solutions to the NNLS problem in some cases, leading to poor performance for the RNN learning. This will be further discussed in Section 2. The second advantage is that the optimal solutions of the NNLS problem can be found regardless of the number of dimensions. These two advantage are essentially important when the algorithm is applied to solving a problem with extremely-high dimension that requires real-time responses, which is usually the case in the real world. Here we point out again that the solution of a standard least-square problem can be obtained directly by the formal pseudoinverse method [4–6, 31–34, 53, 54] and variations of the pseudoinverse method [4]. However, the solution of the NNLS problem needs to be searched via an iterative procedure, due to the non-negative constraint of the weights in the RNN. A caveat here is that the iterative procedure may show a high dependency on the value of learning step size, where a too large step size may cause oscillation and a too small one may cause slow convergence. The line-search technique has been demonstrated to be efficient for solving unconstrained minimization problems [54–56]. For example, in [54], the line-search technique was exploited to select the step size in the iterative procedure for finding appropriate weights of the OPN in a multi-dimensional sinc regression problem, and a fast-convergence performance, a training error as small as the one obtained by the pseudoinverse method and sometime a smaller generalization error have been achieved. In Section 2, we will show that the line-search procedure can be utilized to find excellent solutions to the NNLS problem in almost all cases considered, and these solutions perform quite well in the RNN learning. We also show that the computational complexity of the solving process of the NNLS problem as well as the memory required can be reduced by wiping off the "zero" terms. The idea of the NNLS problem formulation and solving procedure can be extended to solve the optimization problems of the energy packet networks efficiently, where the problems considered are more realistic and have high dimensions. The related context will be further discussed in Subsection 1.2 and Section 3.

1.2. Energy network

Smart grid, which is a network of computers and power infrastructure that monitors and manages energy usage [89], is emerging in the recent years and expected to become the next generation electricity grid [90]. Different literature may have different definitions of the smart grid, but most of these definitions share consistent themes, where the core is advanced metering infrastructure (AMI) that refers to smart metering and two-way communications between electricity providers and customers [58]. According to [58], a smart grid should serve as the backbone that enables widespread penetration of new cutting-edge technologies in metering, transmission, distribution and electricity storage, and both providers and customers access to the grid-related information such that more efficient consumption and provision of electricity can be achieved. One reason why the smart grid is so important is that the existing electricity network is no longer able to fully satisfy the rapidly-increasing demand of electricity [91], which can be well illustrated by the example of China [60–62]. In developing countries, China is one of the fast growing economics and has the second largest economy in the world with the gross domestic product (GDP) being 7.298 trillion U.S. dollars in 2002 [60, 61]. Corresponding to the energy-based nature of the rapid economic growth, China is expected to have a 233% growth of electricity demand from 2007 to 2050 [60]. Another reason is that the highly-interconnected but unidirectional power system may not be able to handle

increased risks of large-scale cascade failures [92]. A smart grid should not be regarded as an electricity-only grid, since an important goal of the smart grid is to integrate all forms of energy [57], including gas, heat, hydro power, solar power, off-shore wind, biomass energy, wave energy and many other combined distributed power, according to the future smart grid vision shown in [96]. Energy sources, such as the solar power, off-shore wind and wave energy, can be classified as renewable energy sources (RES). In terms of climate goals, RES are environment friendly and recent studies show that a smart grid incorporating RES can decrease annual CO_2 emissions by 5 to 16 percent [58]. In addition, RES are becoming commercially profitable in the energy markets, forcing the smart grid to include them.

The whole smart grid is a huge complex system, and many different frameworks have been proposed in the literature to describe what the smart grid is [59, 60, 93, 96]. The work by Gungor in [59] summarized the main issues in the smart grid and presented the framework of a smart gride that includes: 1) the physical infrastructure for energy generation, transmission and distribution; 2) the communication infrastructure for transferring information through the grid; 3) information technology that enables modeling, analysis, web visualization and commercial transactions; and 4) potential applications creating business for smart grid users. Looking at the smart grid from a technical view point, Fang divided the smart grid into three sub-systems: smart infrastructure system, smart management system and smart protection system [93]. The smart infrastructure system is the basis of the smart grid supporting both the two-way information flow and two-way energy flow. Both the smart management and protection systems will take advantage of the smart infrastructure system. The introduction of the two-way energy flow is especially important since the RES are becoming more plentiful [91] and electricity users may also utilize the RES to generate electricity and give it back into the grid, making the electricity storage in the grid extremely distributed. This extremely-distributed energy makes the energy generation and energy market more feasibly [91, 93], however, controlling it can be much more complicated [94]. The smart management system provides advanced management and control services that realize various functions and management objectives, such as energy-efficiency improvement, demand profile shaping, operation cost reduction, emission control, utility maximization and price stabilization. What is more important, the system is able to keep evolving with new cutting-edge developments of management applications and techniques embed, leading to a "smarter" smart grid. The smart protection system is more advanced that provides system reliability and failure protection, and security and privacy protection. The reliability of the system is crucial to guarantee the quality of service. Otherwise, unexpected outages and cascading blackouts could happen [63]. Security objectives in the smart grid is different from, or say, more complicate than most of the other industries, since electricity must always be available for safety reasons meaning that the system can never be locked during an emergency (in contrast, system locking may be a usual safety measure that other industry systems would use) [64].

The key power-generation paradigm in the smart grid is distributed generation (DG) that usually refers to small-scale power generation via distributed energy-resource systems such as solar panels, microturbines, fuel cells and wind power turbines [65]. Implementing DGs may encounter several challenges. The first one is that the electricity-generation patterns from DG systems, which are usually based on very fluctuating and uncontrollable RES, are far from equal to the electricity demands, making it difficult to keep the demand and supply in balance [66]. The second one is that the unit capital costs per kilowatt in electricity of DG systems are usually higher than large-scale central power plants [65, 67]. Specifically, it is reported in [67] that, although operating costs are very low for photovoltaic systems, they are still uncompetitive due to high capital costs. To handle these challenges, the concept of the virtual power plant (VPP) has been introduced and investigated [68, 69]. A definition of the VPP is introduced in [69]: a VPP is a cluster of distributed generators, controllable loads and storages systems, aggregated in order to operate as a unique power plant. The heart of the VPP is the an energy management system (EMS) that coordinates the power flows. Since the electricity productivity of the distributed generators can not be predicted with 100% accuracy [70], the VPP requires the accurate data of energy generation and energy consumption for energy monitoring and control. Therefore, the phasor measurement unites (PMU) utilizing the information from the global positioning system (GPS) are designed to measure and transmit accurate energy information [69]. Since, in most cases, the optimization algorithms for the VPP do not take the network stability into account and may lead to overload of the power system, the system observability becomes an important issue to avoid these possible consequences. For the unknown values in the power system, state estimation can be exploited via the least square method utilizing the measurement information from the PMU, and higher accuracy of the PMU is preferred in order to get better results. Based on the framework of the VPP, an optimization control procedure was presented in [69] for a specific case where there are three different energy suppliers and four energy consumers in the VPP and the objective of the EMS is to minimize the energy-generation costs and avoid the energy loss. In [68], the VPP optimization was first considered to minimize the product cost, which, however, might result in unacceptable voltage levels. Then, the optimization operations from the distributed system operator is introduced in order to maintain the system states within acceptable levels. The work by Hikihara in [94] focused on a small-scale power grid, i.e., the in-home power grid, that also utilizes the RES and faces the DG issues. Hikihara proposed two types of in-home power distribution systems, where the first one is a circuit switching system handling AC power while the second one is a DC power dispatching system via power packets that makes the

control of DG easier. It means that, in the DC power distribution system, supply can be easily regulated by controlling the number of power packets. The system consists of multiple sources, multiple loads, a mixer, a router, and a single distribution line connecting the mixer and the router. The power packet is composed of a header including a start signal and a destination address, a payload and a footer. The mixer gathers power packets and sends them to the single distribution line. The router distributes the received power packets to the destination load. Hikihara did further research on this dispatching system and confirmed its feasibility in physical layer via experiments at the in-home DC network [95]. It is believed in [94, 95] that the power packet dispatching system can be one of the future power distribution methods and will be a key to solve the problems related to the introduction of the RES.

A new grid concept, called microgrid, has also been developed to handle the DG [74, 96]. The main idea of microgrid is that a localized self-contained energy network (namely, microgrid) can be disconnected and reconnected from the main grid (namely macrogrid). This disconnection (or say, islanded) mode of the microgrid has the potential to provide a higher local reliability by avoiding the faults coming from the macrogrid. Specifically, during disturbances, the energy generations and loads in the microgrid can be separated from the macrogrid, and the integrity of the microgrid is not harmed, thereby maintaining services [74]. The description of the microgrid in [74] stressed the plug-and-play functionality, meaning that a unit can be placed at any point in the microgrid without modification of existing equipments. To illustrate this plug-and-play functionality, [74] presented a simple example of a microgrid architecture with four users, where the users with sensitive loads are powered by both the macrogrid and the local generations while the non-critical user is powered only by the macrogrid. When there is a problem with the macrogrid, a static switch will open to isolate the sensitive loads. By doing this, the users with sensitive loads keep powered by the local generation and run well while just the non-critical user is affected. Regarding to the control of the microgrid, a key element is that each DG controller must be able to respond effectively to system changes, such as picking up its share to the loads, without requiring data from the loads or other DG sources. The work by Fang in [71] that is related to the microgrid first investigated the approach to find the most efficient and reliable power supply among a large group of distributed RES for a user in a microgrid and accordingly proposed a discovery approach to discover all the available RES within a microgrid. It is assumed that the power can be intentionally delivered from a distributed generator to a user. It is reported that this can be done by exploiting the above mentioned work of Hikihara about the DC power dispatching system via power packets, in which information, such as destination address, is added to the power and the power is distributed according to this information [94, 95]. Then, Fang investigated the issue of how to find the power supply among the available RES that can deliver the highest profit and accordingly proposed two distributed algorithms based on the machine-learning algorithms in [72, 73]. The work in [98] focused on the power grid in buildings and outlined the potential for building energy saving via efficient operation. The microgrid was the infrastructure used in [98]. The objective was to minimize the total energy cost via the integrated scheduling of the multiple energy supply sources, where this scheduling problem was formulated into an optimization problem with varieties of constraints, including grid constraints, cost calculation formulations, constraints of combined heat and power systems and operation of PV panels. In addition, the significant uncertainties in the demand profile and RES energy supply were also considered, leading to another formulation of the scheduling problem. The mentioned optimization problems were solved by the CPLEX solver.

The concept of Energy Packet Network (EPN), which was started by Gelenbe in [91, 97], can be a new framework for describing (or say, modeling) power grids, which will contribute to the area of the smart grid. Specifically, in [91, 97], Gelenbe proposed a mathematical model called the EPN that takes both the distributed energy generations and conventional sources of energy into consideration. The EPN can be an appropriate framework for both the macroscopic smart grid and micro-scale energy harvesting networks, and it is developed based on the theory of the generalized queueing networks (G-networks) [79, 82, 83]. The G-networks are related to the RNN but different from it in many ways, such as the origins, the research directions and the main usages. As discussed in Subsection 1.1, the research of the RNN began from the paper [44], which was a new class of random networks different from the standard queueing networks with only "positive" customers [75, 76]; while the G-networks is a new generalization of the standard queueing networks by introducing both "positive" and "negative" customers [77], where the positive customers can be considered to be resource requests, while negative customers can correspond to decisions to cancel requests of resources. Note that, before the proposal of the G-networks, the standard queueing network has been widely used in computer and communication system performance modeling and in operations research. The main focuses of the research on the G-networks and RNN rapidly bifurcated with distinct objectives, as outlined in [78]. The followings are three main focuses of the RNN research that have been discussed or mentioned in Subsection 1.1 (a broader range of the focuses can be founded in [78]): 1) utilizing the RNN learning ability for various applications and developing learning algorithms for the RNN; 2) showing that the RNN has the capability for multi-variable function approximation (which sets up connections to the research of the PNN [6, 31–35, 54]); and 3) exploiting the RNN as a decision tool for routing in the CPN [12, 13].

As concluded by Gelenbe [78], the research concerning the G-networks focuses on extending the initial model of the G-networks [77] to include new types of customers, accompanied with the research of proving the existence of product-

form stationary solutions. For example, specific types of customers, usually called signals, arriving the networks can affect the networks behaviors in various ways, which leads to a variety of different models of the G-networks. To model systems in which customers can move from one queue to another queue instantaneously (or say, a customer is routed upon certain events), paper [79] introduced a new type of customers, called signals, into the G-networks, where these signals, different from regular customers, do not receive service from the queueing network but trigger the passage of a customer from one queue to another with certain probability. The effect of the signals on the behaviors of the G-networks was further extended in [80], in which, with certain probabilities, the signals may trigger displacements of customers from one queue to another or force a batch of customers to leave the network. Later research concerning the G-networks further extends the model to contain multi classes of customers [81, 82]. Note that, in the later research, the term "signal" is used to cover negative customers (whose function is to destroy regular/positive customers) and triggers. Paper [81] extended the basic model of the G-networks in [77] to a model of multi classes of positive and negative customers. Paper [82] did further extension and considered the model of the G-networks with multi classes of signals and positive customers. Both models have been proven to have product-form stationary solutions with appropriate assumptions, e.g., exponentially-distributed service times and specific service disciplines, such as first-in-first-out (FIFO) and processor sharing (PS). In addition to negative customers and signals, further research on the G-networks introduced a new type of customers, the "reset" customers [83, 84]. In a system composed of multi unreliable sub-systems, one subsystem in working condition is able to discover the failure of another sub-system and reset it, by which the global reliability of the system can be enhanced. The system can be modeled by the G-networks while the "reset" behavior of the system can be modeled by the "reset" customers [83]. Each step on the extension research of the G-networks makes them more useful tools for analytical modeling of complex systems.

In [80], a simple example application of the G-networks to modeling flow control in communication networks was presented, where the information of the sub-networks can be transformed into flow control packets triggering instantaneous displacements of regular customers. The modeling of packet computer networks by using the G-networks in [85] is much more complex, which incorporates the control traffic as well as the control actions taken at each node and uses multiple classes of traffic and triggers. In addition to the work of modeling, paper [85] presented a gradient-descent based optimization algorithm, where the cost function in the algorithm can include various optimization goals, e.g., delay, traffic loss, power consumption or a combination of them. In [86], the G-networks were applied as power consumption models for wired packet networks. Both models in [85] and [86] take both routers and links as nodes of the networks but separate the nodes into two sets: the set of routers and the set of links. As mentioned by [85], the advantage of the separation is that it offers additional flexibility to the modeling of the system. More specifically, in [86], this separation makes it simpler to model separately the impact of routers and links on delay, traffic loss and energy consumption and to explicitly represent packet re-routing. After presenting the G-networks model, the routing optimization in [86] was expressed as the minimization of a cost function combining both the network power consumption and the average delay by selecting appropriate control parameters $Q(r, k, l)$ which is the probability that a user packet of class k at router r is directed by the corresponding control packet of type (r, k) to the link l, where this minimization problem was solved by the gradient-descent algorithm. The related further work in [87] based on [86] examined how this gradient-descent algorithm (accompanied with the G-networks model) can be used to save energy upon the shortest-path routing and a smart adaptive algorithm called energy aware routing protocol (EARP) [88], where the comparisons were conducted on a 23-node test bed and the test on a large-scale network may not be practical since the algorithm is of time complexity $O(N^3)$ (as reported in [86]). As formerly mentioned, the proposal of the EPN is another application example of the G-networks for analytical modeling [91, 97, 110]. More specifically, in the mathematical model for the EPN of [91], the nodes of the EPN include energy sources, energy storage centers and energy consumption centers, where the energy sources can be distributed renewable energy sources (i.e., the DG). The energy in the EPN is stored, distributed and consumed in the form of energy packets. Although paper [91] did not discussed how the energy flow can be regulated into energy unit in physical systems, the work by Hikihara presented similar concept named as power packets and verified the feasibility of a power packet dispatching system in the physical layer [94, 95]. Then, the EPN can be regarded as a queueing network, the energy storage of the nodes are the queues, the energy packets are the regular customers in the queues and there may be control/request packets in the EPN corresponding to the signals (negative customers or triggers) in the G-networks. Using the G-network theory in an approximation manner and certain assumptions e.g., unlimited storage capacity and energy packet generation and consumption submitting to Poisson process, the stead-state probabilities related to the nodes can be expressed mathematically by a system of equations. The purpose of the EPN in [91] is to meet the surges in energy demand in a grid, where the grid has both steady energy sources and distributed renewable energy sources. In most of the time, the energy demand in the grid is satisfied by steady energy sources, but, in some instants, the energy demand may exceed the maximum level of steady energy and these excess energy requests may be met by renewable energy sources which are managed by the EPN. Most concept used in the EPN model of [97] is similar to that of [91], but the EPN model is more specific and designed for the energy management for the Cloud computing servers, where the energy consumption centers become the Cloud

computing centers. In the cases that scarce sources of energy must be shared by multi computational units, the EPN can manage the energy in the storage centers, which is stored from DG when energy demand is not high, to best match and smooth the intermittent energy supply. Similarly, using the G-network theory and certain assumptions, the stead-state probabilities related to the nodes in the EPN can be approximately expressed by a system of equations. By analyzing the solutions of the related equations, the approximate behaviors of the EPNs (in both [91] and [97]) can be analyzed so that parameters in the EPN can be adjusted accordingly in order to achieve different objectives, e.g., satisfying energy needs of energy consumption centers, or say, the Cloud computing centers. A recent work in [110] considered an interconnected distributed computer system with multiple computation centers (e.g., the above mentioned Cloud computing servers) and introduced a new class of regular customers into the EPN model, i.e., the discrete units of computational work (jobs). Another class of regular customers in the EPN is the energy packets, while the discrete representations of data packets used for control operations in the EPN correspond to the G-network "triggers". There are only two types of nodes in the EPN: the computation centers (CCs) and energy storage centers (ESCs). The CCs request energy from the ESCs while they are busy processing jobs. In addition, the CCs can also get energy from other sources. Assuming all job and energy packet arrivals are Poisson and service rates are exponential, the probabilities that the CCs are busy processing job, the local energy storages of the CCs are non-empty and the ESCs have at least one energy packet can be expressed mathematically by a system of equations based on the G-network theory. Then, the average job response time can also be approximately calculated with the solutions of the system of equations. By analyzing the system of equations (or say, the EPN model), paper [110] compared the choices of centralised or distributed energy storage and determined a better choice that offered smaller response time.

The EPN modelling based on the G-network theory can be promising in the area of the smart grid. However, in the existing EPN models (e.g., [91, 97, 110]), some assumptions may not be applicable to the modelling of many practical electrical grids.

- The energy flows in the EPNs are discrete representation of energy in regulated energy packets, while, in most existing electrical grids, electricity is transmitted in an analog and continuous quantity. It means that the EPN models may NOT be DIRECTLY used to model and analyze these electrical grid. In an approximate manner, the EPN models may still be applicable. In addition, some research about the future electrical grids has already recognized the advantage and feasibility of transmitting energy in regulated packets [71, 94, 95].

- To the best of the authors' knowledge, the existing EPN models have NOT taken the transit delay from one node to another node into consideration. It may be practical not to consider the delay in a small-scale electrical grid with short-distance links (e.g., the microgrid). But the delay may no longer be negligible in a large-scale electrical grid. One possible solution may be to include the links as nodes into the EPN models.

- The Poisson arrivals of regular customers (such as, energy packets and jobs) and exponential service rates are assumed in the EPN models in order to apply the G-network theory, which may be able to model the energy generation from renewable energy sources since it is similar to the Poisson process. But, it may NOT be used to model steady energy generation or those quite different from Poisson process.

- In most cases of the EPN models, it is assumed that the nodes have unlimited capacities so as to simplify the equations used to describe the system. The energy storage centers may have sufficiently large capacities, but the energy consumption centers usually have small capacities or sometimes have no capacity, where, in these cases, the modeling is NOT exact. To deal this situation, paper [91, 97] presented approximation equations used to describe nodes with limited capacities. However, these approximation equations are beyond the G-network theory and the efficacy of them has not been theoretically proven.

1.3. Work done in this report

This rest of this report is organized into two parts.
• Heuristic line-search aided non-negative least-square learning for random neural network.
• Energy distribution for energy packet networks.
The first part incorporate the idea of the non-negative least-square problem formulation in [48, 49] and the line-search technique in [55] and then design a heuristic but efficient learning algorithm for the RNN, which is termed the line-search aided non-negative least-square (LNNLS) learning algorithm. The second part considers the energy-distribution problems of different EPNs and then designs effective optimization algorithms for solving the problems, such as the gradient-descent algorithm, the cooperative particle swarm optimizer and the heuristic algorithms developed based on part one.

2. Heuristic line-search aided non-negative least-square learning for random neural network

This section first presents briefly the mathematical model of the RNN and its learning problem. Hoping to find appropriate solutions for the learning problem, a heuristic LNNLS algorithm is then designed based on [48, 49] by approximating the RNN mathematical equations. Numerical results show that this heuristic algorithm is capable of finding satisfactory solutions in most cases.

2.1. RNN model

Let us consider a RNN with N neurons, where $k_i(t) \geq 0$ denotes the potential (or say, state) of the ith neuron at time t with $i = 1, \cdots, N$ and correspondingly vector $k(t) = [k_1(t)\ k_2(t)\ \cdots\ k_N(t)]$ denotes the state of the whole RNN. All neurons are fully connected and exchange positive (or say, excitatory) and negative (or say, inhibitory) signals in the form of unit amplitude spikes (by firing). Each positive signal increases the potential of the receiving neuron by 1, while each negative signal reduces the potential that is larger than zero by 1. The ith neuron is excited when $k_i(t) > 0$ or idle when $k_i(t) = 0$. When excited, the ith neuron fires according to an independent exponential distribution with rate r_i. In addition, its potential is reduced by 1 each time it fires. The fired spike heads for the jth neuron as a positive signal with probability $p^+_{i,j}$ or as a negative signal with probability $p^-_{i,j}$, where $j = 1, \cdots, N$, or it departs from the network with probability d_i. Correspondingly,

$$\sum_{j=1}^{N}(p^+_{i,j} + p^-_{i,j}) + d_i = 1. \tag{1}$$

All neurons also receive signals from the outside world. Positive and negative signals arrive at the ith neuron from the outside world according to Poisson processes of rates Λ_i and λ_i, respectively.

Let $q_i = \lim_{t\to\infty} \text{Prob}(k_i(t) > 0)$ denote the stationary probability of the ith neuron being excited. Based on [47], the signal flows in the RNN can be described by the following system of nonlinear equations:

$$q_i = \begin{cases} \dfrac{\lambda^+_i}{r_i + \lambda^-_i}, & \text{if } \lambda^+_i < r_i + \lambda^-_i \\ 1, & \text{if } \lambda^+_i \geq r_i + \lambda^-_i \end{cases} \tag{2}$$

where $\lambda^+_i = \Lambda_i + \sum_{j=1}^{N} q_j w^+_{j,i}$, $\lambda^-_i = \lambda_i + \sum_{j=1}^{N} q_j w^-_{j,i}$, $w^+_{j,i} = r_j p^+_{j,i}$, $w^-_{j,i} = r_j p^-_{j,i}$ and $i = 1, \cdots, N$.

Suppose that the weights $\{w^+_{j,i}, w^-_{j,i} | i, j = 1, \cdots, N\}$ in the RNN are known and the values of $\underline{\Lambda} = [\Lambda_1\ \cdots\ \Lambda_N]$ and $\underline{\lambda} = [\lambda_1\ \cdots\ \lambda_N]$ at the cth computation are set as $\underline{\Lambda}_c = [\Lambda_{1,c}\ \cdots\ \Lambda_{N,c}]$ and $\underline{\lambda}_c = [\lambda_{1,c}\ \cdots\ \lambda_{N,c}]$, then the stationary states of the RNN $\underline{q} = [q_1\ \cdots\ q_N]$ at this computation can be calculated from (2), which are represented by $\underline{q}_c = [q_{1,c}\ \cdots\ q_{N,c}]$. Then, the learning problem of the RNN can be described as follows.

Given the values of $\underline{\Lambda}$ and $\underline{\lambda}$ at the cth computation (i.e., $\underline{\Lambda}_c$ and $\underline{\lambda}_c$) and the desired values $\underline{y}_c = [y_{1,c}\ \cdots\ y_{N,c}]$ for $\underline{q}_c$, we want to minimize the difference between $\underline{q}_c$ and $\underline{y}_c$ by finding appropriate values for $\{w^+_{j,i}, w^-_{j,i} | i, j = 1, \cdots, N\}$, where $c = 1, \cdots, C$.

2.2. Heuristic LNNLS algorithm

The basic idea of the heuristic LNNLS algorithm is to find appropriate weights to approximate the RNN system of equations (2) as close as possible. By introducing $\{\underline{y}_c | c = 1, \cdots, C\}$ into this approximation problem, we hope that the found weights can also make the RNN states $\{\underline{q}_c | c = 1, \cdots, C\}$ close to their desired values (i.e., $\{\underline{y}_c | c = 1, \cdots, C\}$). For better understanding, Figure 1 shows the procedure of exploiting the algorithms to find weights for the RNN learning. It is worth pointing out that there is no theoretical result to guarantee that the weights found by the heuristic LNNLS algorithm are satisfactory solutions to the learning problem of the RNN.

First, this approximation problem is formulated as a non-negative least-square problem based on [48, 49]. Then, the heuristic LNNLS algorithm is designed to solve this problem by combining the algorithm procedure in [48, 49] and the line-search technique in [55].

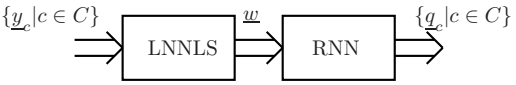


Figure 1: Block diagram of the heuristic LNNLS algorithm exploited to find weights for the learning of the RNN.

2.2.1. Approximation problem formulation

We first rewrite RNN equations (2) as

$$\begin{aligned} &\frac{q_{i,c}\sum_{j=1}^{N}\left(w^+_{i,j} + w^-_{i,j}\right)}{1 - d_i} + q_{i,c}\sum_{j=1}^{N} q_{j,c} w^-_{j,i} \\ &- \sum_{j=1}^{N} q_{j,c} w^+_{j,i} - (\Lambda_{i,c} - q_{i,c}\lambda_{i,c}) = 0, \end{aligned} \tag{3}$$

with $i = 1, \cdots, N$ and $c = 1, \cdots, C$.

If we substitute $q_{i,k}$ with $y_{i,k}$ into (3), we obtain a linear system of NK equations with $2N^2$ nonnegative unknowns (i.e., the weights $w^+_{i,j}$ and $w^-_{i,j}$). However, a unique solution to this linear system may not be available due to the uncertain relationship between NC and $2N^2$ and the constraint of the weights. To approach equality as much as possible, the work [48, 49] formulate (3) as a non-negative least-square problem:

$$\begin{aligned} &\text{minimize } f(\underline{w}) = \text{minimize } \frac{1}{2}\|A\underline{w} - b\|_2^2, \\ &\text{subject to } \underline{w} \geq 0, \end{aligned} \tag{4}$$

where $A \in R^{NC \times 2N^2}$ and $b \in R^{NC \times 1}$ are defined as

$$\begin{aligned} &A(h_{i,c}, h^+_{i,j}) = \frac{q_{i,c}}{1 - d_i}, \forall j \neq i \\ &A(h_{i,c}, h^-_{i,j}) = \frac{q_{i,c}}{1 - d_i}, \forall j \neq i \\ &A(h_{i,c}, h^+_{i,j}) = \frac{q_{i,c}}{1 - d_i} - q_{j,c}, \forall j = i \\ &A(h_{i,c}, h^-_{i,j}) = \frac{q_{i,c}}{1 - d_i} + q_{i,c}q_{j,c}, \forall j = i \\ &A(h_{i,c}, h^+_{j,i}) = -q_{j,c}, \forall j \neq i \\ &A(h_{i,c}, h^-_{j,i}) = q_{i,c}q_{j,c}, \forall j \neq i \\ &A(h_{i,c}, \text{otherwise}) = 0, \\ &b(h_{i,c}) = \Lambda_{i,c} - q_{i,c}\lambda_{i,c}, \end{aligned} \tag{5}$$

and $\underline{w} \in R^{2N^2 \times 1}$ consists of the weights $w^+_{i,j}$ and $w^-_{i,j}$ and is given by $\underline{w}(h^+_{i,j}) = w^+_{i,j}$ and $\underline{w}(h^-_{i,j}) = w^-_{i,j}$ with $h^+_{i,j} = N^2 + (i-1)N + j$ and $h^-_{i,j} = (i-1)N + j$.

2.3. LNNLS algorithm procedure

The solution of a standard least-square problem can be obtained directly by the pseudoinverse method [54]. However, due to the non-negative constraint of $\underline{w}$, the solution needs to be searched via an iterative procedure. Based on [48], in the mth iteration, where $m = 1, \cdots, M$ with M being the maximum number of iterations, the following weight-update formula can be exploited:

$$\underline{w}^{(m+1)} = P[\underline{w}^{(m)} - \eta \nabla f(\underline{w}^{(m)})], \tag{6}$$

where

$$\nabla f(\underline{w}^{(m)}) = A^{\mathrm{T}}(A\underline{w}^{(m)}) - A^{\mathrm{T}}b, \tag{7}$$

$\eta > 0$ is the step size and, with $h = 1, \cdots, 2N^2$,

$$P[\underline{w}(h)] = \begin{cases} \underline{w}(h), & \text{if } \underline{w}(h) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Besides, $q_{i,c}$ in A is substituted with $y_{i,c}$ in each iteration.

The iterative procedure by using (6) may show a high dependency on the value of step size η, where a too large step size may cause oscillation and a too small one may cause slow convergence. In this report, we extend the line-search technique that is originally designed for solving unconstrained minimization problems [54–56] and apply it to selecting η dynamically and appropriately. Based on (6) and the line-search technique, the LNNLS learning algorithm for the RNN can finally be designed and its detailed procedure is provided in Algorithm 1, where

$$f(\underline{w}) = \frac{1}{2}\|A\underline{w} - b\|_2^2 \tag{8}$$

is used to measure the process of the algorithm in finding appropriate weights. Note that the influence of the way of initializing $\underline{w}^{(1)}$ on the efficacy of the LNNLS learning algorithm can be negligible in most cases if M is sufficiently large. However, initializing $\underline{w}^{(1)}$ with the zero value may be an acceptable choice, which is further illustrated in Section 2.5.

2.4. Heuristic A-matrix free LNNLS algorithm

In Algorithm 1, one difficulty is to construct and store matrix $A \in R^{NC \times 2N^2}$ due to its large dimensionality. To be more specific, Table 1 shows the memory needed to store A under different values of N and C. Evidently, it may not be possible to store matrix A and conduct computations with A in a Personal Computer when $N \geq 1000$. Therefore, it is important to modify the NNLS learning algorithm (i.e, Algorithm 1) such that it does not require storing A and becomes suitable for solving the learning problem of the RNN with large dimensionality. Moreover, from (5), it can be found that only $4N - 2$ of the $2N^2$ elements in each row of A are nonzero. We therefore can reduce the computational complexity of Algorithm 1 by wiping off the computations about the 'zero' elements.

For presentation convenience, let $\nabla = \nabla f(\underline{w}^{(m)}) \in R^{2N^2 \times 1}$ and $\nabla(h)$ with $h = 1, \cdots, 2N^2$ denotes the hth element in ∇. Based on (5) and (7), we can deduce equations (9) and (10) for calculating ∇ without constructing A, where

Algorithm 1 The LNNLS learning algorithm

```
Construct A and b;
initialize w^(1) and calculate f(w^(1)) via (8);
η ← 1.5; ϑ ← 2; m ← 1;
η_new ← η, w_new ← w^(1), f_new ← f(w^(1));
while m ≤ M do
  calculate ∇f(w^(m)) via (7);
  w^(m+1) ← P[w^(m) − η∇f(w^(m))];
  calculate f(w^(m+1)) via (8);
  while f(w^(m+1)) < f_new do
    η_new ← η; w_new ← w^(m+1);
    f_new ← f(w^(m+1)); η ← ηϑ;
    calculate ∇f(w^(m)) via (7);
    w^(m+1) ← P[w^(m) − η∇f(w^(m))];
    calculate f(w^(m+1)) via (8);
  end while
  while f(w^(m+1)) > f_new & η > 10^−20 do
    η ← η/ϑ;
    calculate ∇f(w^(m)) via (7);
    w^(m+1) ← P[w^(m) − η∇f(w^(m))];
    calculate f(w^(m+1)) via (8);
    if f(w^(m+1)) < f_new then
      η_new ← η; w_new ← w^(m+1); f_new ← f(w^(m+1));
    end if
  end while
  η ← η_new; w^(m+1) ← w_new; f(w^(m+1)) ← f_new;
  m ← m + 1;
end while
```

$G \in R^{NC \times 1}$ and its $h_{i,c}$th element is

$$\begin{aligned} G(h_{i,c}) = & \sum_{\alpha=1(\alpha\neq i)}^{N} \left(\frac{q_{i,c}}{1-d_i} \left(\underline{w}(h_{i,\alpha}^{+}) + \underline{w}(h_{i,\alpha}^{-}) \right) \right) \\ & + \sum_{\beta=1(\beta\neq i)}^{N} \left(q_{\beta,c} \left(q_{i,c}\underline{w}(h_{i,\alpha}^{-}) - \underline{w}(h_{i,\alpha}^{+}) \right) \right) \\ & + \left(\frac{q_{i,c}}{1-d_i} - q_{i,c} \right) \underline{w}(h_{i,i}^{+}) + \left(\frac{q_{i,c}}{1-d_i} + q_{i,c}^2 \right) \underline{w}(h_{i,i}^{-}) \end{aligned} \tag{11}$$

Note that the derivation of equations (9), (10) and (11) is given in Appendix. In addition, learning performance measurement f can be calculated without constructing A:

$$f(\underline{w}) = \frac{1}{2}\|G - b\|_2^2. \tag{12}$$

The detailed procedure of the A-matrix free LNNLS learning algorithm is provided in Algorithm 2

Remark 2. The heuristic LNNLS learning algorithm (including Algorithms 1 and 2) is designed for the RNN with only output neurons. But, by using similar procedure of the weight-initialization algorithm in [48], the LNNLS learning algorithm can be easily extended to the case where the RNN is composed of both output and non-output neurons. Suppose the RNN has N_{out} output and $N_{\bar{\mathrm{out}}}$ non-output neurons. The weights and desired outputs of non-output neurons $y_{i_{\bar{\mathrm{out}}},c}, \forall i_{\bar{\mathrm{out}}} \in N_{\bar{\mathrm{out}}}, c \in C$ in the LNNLS learning algorithm with weight initialization are initialized for L times, where the detailed procedure is given in Algorithm 3.

$$\nabla(h_{i,j}^{+}) = \begin{cases} \sum_{c=1}^{C} \left(\frac{q_{i,c}\left(G(h_{i,c}) - b(h_{i,c})\right)}{1-d_i} - q_{i,c}\left(G(h_{j,c}) - b(h_{j,c})\right) \right), & \forall i \neq j, \\ \sum_{c=1}^{C} \left(\left(\frac{q_{i,c}}{1-d_i} - q_{i,c} \right) \left(G(h_{i,c}) - b(h_{i,c})\right) \right), & \forall i = j, \end{cases} \tag{9}$$

$$\nabla(h_{i,j}^{-}) = \begin{cases} \sum_{c=1}^{C} \left(\frac{q_{i,c}\left(G(h_{i,c}) - b(h_{i,c})\right)}{1-d_i} + q_{i,c}q_{j,c}\left(G(h_{j,c}) - b(h_{j,c})\right) \right), & \forall i \neq j \\ \sum_{c=1}^{C} \left(\left(\frac{q_{i,c}}{1-d_i} + q_{i,c}^2 \right) \left(G(h_{i,c}) - b(h_{i,c})\right) \right), & \forall i = j, \end{cases} \tag{10}$$

Algorithm 2 The A-matrix free LNNLS learning algorithm

Construct b; initialize $\underline{w}^{(1)}$ and calculate $f(\underline{w}^{(1)})$ via (12);
$\eta \leftarrow 1.5$; $\vartheta \leftarrow 2$; $m \leftarrow 1$;
$\eta_{\text{new}} \leftarrow \eta$, $\underline{w}_{\text{new}} \leftarrow \underline{w}^{(1)}$, $f_{\text{new}} \leftarrow f(\underline{w}^{(1)})$;
while $m \leq M$ **do**
  calculate $\nabla f(\underline{w}^{(m)})$ via (9) and (10);
  $\underline{w}^{(m+1)} \leftarrow P[\underline{w}^{(m)} - \eta \nabla f(\underline{w}^{(m)})]$;
  calculate $f(\underline{w}^{(m+1)})$ via (12);
  while $f(\underline{w}^{(m+1)}) < f_{\text{new}}$ **do**
    $\eta_{\text{new}} \leftarrow \eta$; $\underline{w}_{\text{new}} \leftarrow \underline{w}^{(m+1)}$;
    $f_{\text{new}} \leftarrow f(\underline{w}^{(m+1)})$; $\eta \leftarrow \eta\vartheta$;
    calculate $\nabla f(\underline{w}^{(m)})$ via (9) and (10);
    $\underline{w}^{(m+1)} \leftarrow P[\underline{w}^{(m)} - \eta \nabla f(\underline{w}^{(m)})]$;
    calculate $f(\underline{w}^{(m+1)})$ via (12);
  end while
  while $f(\underline{w}^{(m+1)}) > f_{\text{new}} \& \eta > 10^{-20}$ **do**
    $\eta \leftarrow \eta/\vartheta$;
    calculate $\nabla f(\underline{w}^{(m)})$ via (9) and (10);
    $\underline{w}^{(m+1)} \leftarrow P[\underline{w}^{(m)} - \eta \nabla f(\underline{w}^{(m)})]$;
    calculate $f(\underline{w}^{(m+1)})$ via (12);
    if $f(\underline{w}^{(m+1)}) < f_{\text{new}}$ **then**
      $\eta_{\text{new}} \leftarrow \eta$; $\underline{w}_{\text{new}} \leftarrow \underline{w}^{(m+1)}$; $f_{\text{new}} \leftarrow f(\underline{w}^{(m+1)})$;
    end if
  end while
  $\eta \leftarrow \eta_{\text{new}}$; $\underline{w}^{(m+1)} \leftarrow \underline{w}_{\text{new}}$; $f(\underline{w}^{(m+1)}) \leftarrow f_{\text{new}}$;
  $m \leftarrow m + 1$;
end while

Algorithm 3 The LNNLS learning algorithm with weight initialization

Initialize $\underline{w}^{(1)}$ and $y_{i_{\text{o}\bar{\text{u}}\text{t}},c}, \forall i_{\text{o}\bar{\text{u}}\text{t}} \in N_{\text{o}\bar{\text{u}}\text{t}}, c \in C$;
$l \leftarrow 1$;
while $l \leq L$ **do**
  obtain $\underline{w}_{\text{new}}$ via Algorithm 1 or 2 with $\underline{w}^{(1)}$ and $y_{i_{\text{o}\bar{\text{u}}\text{t}},c}, \forall i_{\text{o}\bar{\text{u}}\text{t}} \in N_{\text{o}\bar{\text{u}}\text{t}}, c \in C$;
  solve (2) using $\underline{w}_{\text{new}}$ for $q_{i_{\text{o}\bar{\text{u}}\text{t}},c}, \forall i_{\text{o}\bar{\text{u}}\text{t}} \in N_{\text{o}\bar{\text{u}}\text{t}}, c \in C$;
  $\underline{w}^{(1)} \leftarrow \underline{w}_{\text{new}}$;
  $y_{i_{\text{o}\bar{\text{u}}\text{t}},c} \leftarrow q_{i_{\text{o}\bar{\text{u}}\text{t}},c}, \forall i_{\text{o}\bar{\text{u}}\text{t}} \in N_{\text{o}\bar{\text{u}}\text{t}}, c \in C$;
end while

Table 1: Memory needed to store A under different values of N and C

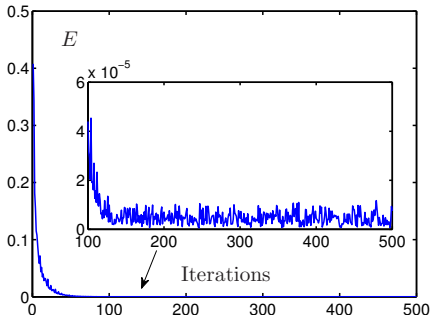
	N=10	N=100	N=1000
C=1	0.0153MB	15.2588MB	14.9012GB
C=10	0.1526MB	152.5879MB	149.0116GB
C=100	1.5259MB	1.4901GB	1.4552TB
C=1000	15.2588MB	14.9012GB	14.5519TB

2.5. Numerical results

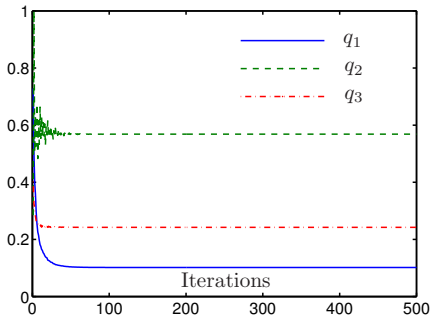
In this section, numerical experiments are conducted to test the performance of the heuristic LNNLS algorithm for the RNN. (The source codes of Algorithm 1 in MATLAB language and Algorithm 2 in C language are available at http://www.yonghuayin.icoc.cc/.) All of the numerical experiments are conducted in a MATLAB R2014a environment, which is operated on a personal computer (CPU: Intel i7-4770 3.40 GHz; memory: 8.00 GB). Note that, for simplicity, d_i with $i = 1, \cdots, N$ are set as zero. In addition, the root mean square error (RMSE) is used to measure the performances of the RNN and learning algorithm:

$$E = \sqrt{\frac{1}{NC} \sum_{i=1}^{N} \sum_{c=1}^{C} (y_{i,c} - q_{i,c})^2}. \tag{13}$$

First, let us consider a simple dataset named as *Simple* with $N = 3$ and $C = 1$. In addition, $\underline{\Lambda}_{1,1} = [0.5934\ 0.5501\ 0.9935]$, $\underline{\lambda}_{1,1} = [0.1814\ 0.1896\ 0.8415]$ and $\underline{y}_1 = [0.1018\ 0.5684\ 0.2422]$, the values of which are randomly generated among range $[0, 1]$. Then, the heuristic LNNLS algorithm is exploited for the RNN to learn the *Simple* dataset. The numerical results are shown Figs. 2 and 3. In the numerical experiment of Fig. 2, initial values of weights (i.e., $\underline{w}^{(1)}$) are set as zero. From Fig. 2(a), we can see that the RMSE decreases rapidly to tiny values during the iterative process. To be more specific, the RMSE decreases to 4.3980×10^{-5} at the 100th iteration and reaches the smallest value of 5.5626×10^{-7} at 246th iteration. Fig. 2(b) illustrates that each RNN output converges rapidly to the corresponding desired output. In the numerical experiment of Fig. 3, initial values of weights are randomly selected and 10 trials are conducted. It can be seen from the results in Fig. 3 that, under different initial weights, all the RMSEs decrease rapidly to tiny values and become smaller than 3.0×10^{-3}



(a) RMSE versus number of iterations



(b) RNN outputs versus number of iterations

Figure 2: Learning performance of the RNN equipped with the LNNLS algorithm for the *Simple* dataset under "zero" initial weights.

after 100 iterations.

For comparison, the projected gradient algorithm in [49, 111] is also exploited for the RNN to learn the *Simple* dataset. The results are given in Fig. 4, where we can see that this algorithm is highly affected by the values of initial weights. Comparing results in Figs. 2 and 3 and Fig. 4, we can conclude that the heuristic LNNLS algorithm proposed in this paper outperforms the one in [49, 111] for the RNN learning.

Then, let us consider datasets with higher dimensions. The numerical experiments in Fig. 5 are conducted based on datasets with $N = 50, 100, 200, 400$, where the values of inputs and desired outputs are randomly generated among range $[0, 1]$. In addition, $C = 1$. All results (including Figs. 2 through 5) demonstrate that, by exploiting the heuristic LNNLS algorithm, satisfactory weights can be found for the RNN such that the RMSEs become acceptably small.

3. Energy distribution for energy packet networks

3.1. A simplified energy packet network

The EPN consists of G generators, S storage units and C consumers. Before presenting the mathematical model of the EPN, the following notations need to be clarified.

1) Let Λ_g, $\forall g \in G$ represent the energy generation rate of the gth generators.

2) Let δ_s, $\forall s \in S$ represent the energy transfer rate from the sth storage unit to other units.

3) Let γ_s, $\forall s \in S$ represent the energy leakage rate of the

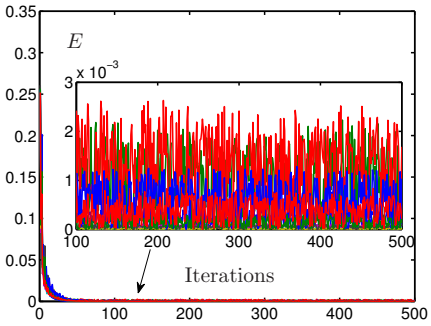


Figure 3: Learning performance of the RNN equipped with the LNNLS algorithm for the *Simple* dataset under random initial weights.

sth storage unit.

4) Let $\hat{q}_s$, $\forall s \in S$ represent the probability that the sth storage unit has energy in storage.

5) Let μ_c, $\forall c \in C$ represent the energy consumption rate of the cth consumer.

6) Let q_c, $\forall c \in C$ represent the probability that the cth consumer has energy to consume.

7) Let $\hat{p}_{g,s}$ represent the proportion of energy that the gth generator sends to the sth storage unit.

8) Let $\tilde{p}_{g,c}$ represent the proportion of energy that the gth generator sends to the cth consumer.

9) Let $p_{s,c}$ represent the proportion of energy that the sth storage unit sends to the cth consumer.

The EPN can then be described by the following system of equations:

$$\begin{cases} \hat{q}_s = \dfrac{\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})}{\gamma_s + \sum_{c=1}^{C}(\delta_s p_{s,c})} = \dfrac{\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})}{\gamma_s + \delta_s}, \\ q_c = \dfrac{\sum_{g=1}^{G}(\Lambda_g \tilde{p}_{g,c}) + \sum_{s=1}^{S}(\hat{q}_s \delta_s p_{s,c})}{\mu_c}, \end{cases} \tag{14}$$

where $\forall s \in S$ and $\forall c \in C$. In addition, $\sum_{c=1}^{C} p_{s,c} = 1$ and $\sum_{c=1}^{C} \tilde{p}_{g,c} + \sum_{s=1}^{S} \hat{p}_{g,s} = 1$.

We want to maximize the amount of work done by the consumers per unit time in the EPN (14), or equivalently: given $\{\Lambda_g | g = 1, \cdots, G\}$, $\{\mu_c | c = 1, \cdots, C\}$, $\{\delta_s | s = 1, \cdots, S\}$ and $\{\gamma_s | s = 1, \cdots, S\}$,

$$\text{maximize } K = \sum_{c=1}^{C}(\mu_c q_c), \tag{15}$$

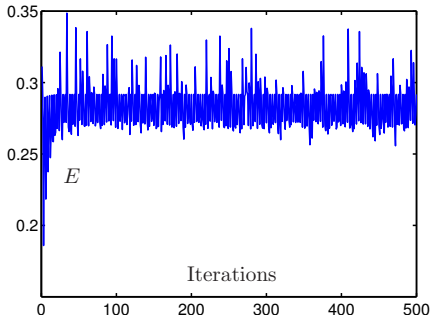
by using $\{\hat{p}_{g,s} | g = 1, \cdots, G; s = 1, \cdots, S\}$, $\{\tilde{p}_{g,c} | s = 1, \cdots, S; c = 1, \cdots, C\}$ and $\{p_{s,c} | s = 1, \cdots, S; c = 1, \cdots, C\}$.

3.2. Optimal solutions

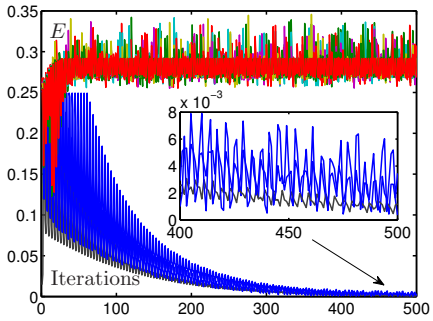
Two cases need to be considered. One case is that the energy is limited. The other is that the energy is sufficient.

3.2.1. The energy is limited

In this case, the energy is limited such that $\sum_{g=1}^{G} \Lambda_g \leq \sum_{c=1}^{C} \mu_c$. We want to make use of all energy and do not want the energy to go to the storage centers because the energy will leak without doing anything useful. So, it is reasonable to



(a) "Zero" initial weights



(b) Random initial weights

Figure 4: For comparison: learning performance of the RNN equipped with the projected gradient algorithm in [49, 111] for the *Simple* dataset under "zero" and random initial weights.

set $\hat{p}_{g,s} = 0, \forall g \in G, s \in S$. Since $\sum_{c=1}^{C} \tilde{p}_{g,c} + \sum_{s=1}^{S} \hat{p}_{g,s} = 1$, then $\sum_{c=1}^{C} \tilde{p}_{g,c} = 1$. From (14), we have $\hat{q}_s = 0$. Then,

$$\begin{cases} \hat{q}_s = 0 \\ q_c = \dfrac{\sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c})}{\mu_c}. \end{cases} \tag{16}$$

Let $H = \sum_{c=1}^{C} \mu_c$. In addition, let us set $\tilde{p}_{g,c} = \mu_c / H, \forall c \in C$. Then,

$$\frac{q_{c_1}}{q_{c_2}} = \frac{\mu_{c_2} \sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c_1})}{\mu_{c_1} \sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c_2})} = \frac{\mu_{c_2} \sum_{g=1}^{G} \left(\Lambda_g \frac{\mu_{c_1}}{H}\right)}{\mu_{c_1} \sum_{g=1}^{G} \left(\Lambda_g \frac{\mu_{c_2}}{H}\right)} = 1. \tag{17}$$

Then, we have $q_{c_1} = q_{c_2}, \forall c_1, c_2 \in C$.

Then,

$$K = \sum_{c=1}^{C} (\mu_c q_c) = \sum_{c=1}^{C} \sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c}) = \sum_{g=1}^{G} \Lambda_g. \tag{18}$$

In addition, we also have $q_c \leq 1, \forall c \in C$ since $\sum_{c=1}^{C} (\mu_c q_c) = \sum_{g=1}^{G} \Lambda_g \leq \sum_{c=1}^{C} \mu_c$. We can see that all energy has been fully used (no energy is wasted). Based on the law of conservation of energy, we can say that $K^* = \sum_{g=1}^{G} \Lambda_g$ is the optimal result for the maximization problem (15), where **the optimal solution is** $\hat{p}_{g,s} = 0$ **and** $\tilde{p}_{g,c} = \mu_c / H$, $\forall g \in G, s \in S, c \in C$. **In addition,** $p_{s,c}$ $\forall s \in S, c \in C$ **can be any non-negative value satisfying** $\sum_{c=1}^{C} p_{s,c} = 1$.

3.2.2. The energy is sufficient

In this case, the energy is sufficient such that $\sum_{g=1}^{G} \Lambda_g > \sum_{c=1}^{C} \mu_c$. Let us set $\hat{p}_{g,s} = 0$ and $\tilde{p}_{g,c} = \mu_c / H$, $\forall g \in G, s \in S, c \in C$. Since q_c can not be larger than 1, the system can no

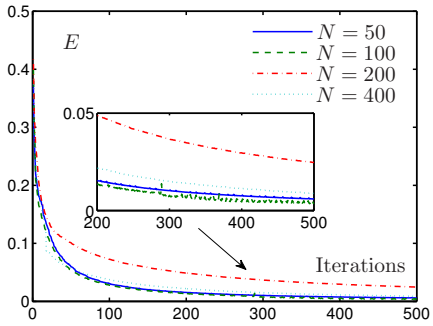


Figure 5: Learning performance of the RNN equipped with the LNNLS algorithm for datasets with $N = 50, 100, 200, 400$.

longer be described by (16). The system should be described by

$$\begin{cases} \hat{q}_s = 0 \\ q_c = \min\{1, \dfrac{\sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c})}{\mu_c}\}. \end{cases} \tag{19}$$

From (17), we know that

$$\frac{\sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c})}{\mu_{c_1}} = \frac{\sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c})}{\mu_{c_2}}.$$

So, we still have $q_{c_1} = q_{c_2}, \forall c_1, c_2 \in C$. In this case, the energy is sufficient such that $q_c = 1, \forall c \in C$, meaning the system is saturated.

Then, $K = \sum_{c=1}^{C} (\mu_c q_c) = \sum_{c=1}^{C} \mu_c$. Although much energy is wasted, $K^* = \sum_{c=1}^{C} \mu_c$ is the optimal result for the maximization problem (15) in this case, where **the optimal solution is also** $\hat{p}_{g,s} = 0$ **and** $\tilde{p}_{g,c} = \mu_c / H$, $\forall g \in G, s \in S, c \in C$. **In addition,** $p_{s,c}$ $\forall s \in S, c \in C$ **can be any non-negative value satisfying** $\sum_{c=1}^{C} p_{s,c} = 1$.

3.3. Gradient-descent algorithm

We can also design a gradient-descent algorithm to solve the maximization problem (15).

First, let us define $P \in R^{(SC+GC+GS)\times 1}$ as a vector that consists of $\hat{p}_{g,s}$, $\tilde{p}_{g,c}$ and $p_{s,c}$, where

$$P(h_{s,c}) = p_{s,c}, \ P(\tilde{h}_{g,c}) = \tilde{p}_{g,c}, \ P(\hat{h}_{g,s}) = \hat{p}_{g,s},$$

with $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$ and $\hat{h}_{g,s} = SC + GC + (g-1)S + s$. Then, we need to derive the expression of $\partial K / \partial P$ such that a P-update formula can be derived.

To simplify the derivation, we first ignore the constraints of P, which are $P \geq 0$, $\sum_{c=1}^{C} p_{s,c} = 1$ and $\sum_{c=1}^{C} \tilde{p}_{g,c} + \sum_{s=1}^{S} \hat{p}_{g,s} = 1$. From (14), we know that

$$\frac{\partial \hat{q}_{\hat{s}}}{\partial p_{s,c}} = 0, \ \frac{\partial \hat{q}_{\hat{s}}}{\partial \tilde{p}_{g,c}} = 0,$$
$$\frac{\partial \hat{q}_{\hat{s}}}{\partial \hat{p}_{g,s}} = \frac{\Lambda_g}{\gamma_s + \delta_s}, \ \forall \hat{s} = s,$$
$$\frac{\partial \hat{q}_{\hat{s}}}{\partial \hat{p}_{g,s}} = 0, \ \forall \hat{s} \neq s,$$

Then,

$$\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial p_{s,c}} = \frac{\partial\left(\sum_{\hat{s}=1}^{S}(\hat{q}_{\hat{s}}\delta_{\hat{s}}p_{\hat{s},\hat{c}})\right)}{\partial p_{s,c}} = \hat{q}_s\delta_s,\ \forall\hat{c}=c,$$
$$\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial p_{s,c}} = 0,\ \forall\hat{c}\neq c,$$
$$\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial \tilde{p}_{g,c}} = \frac{\partial\left(\sum_{\hat{g}=1}^{G}(\Lambda_{\hat{g}}\tilde{p}_{\hat{g},\hat{c}})\right)}{\partial \tilde{p}_{g,c}} = \Lambda_g,\ \forall\hat{c}=c,$$
$$\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial \tilde{p}_{g,c}} = 0,\ \forall\hat{c}\neq c,$$

$$\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial \hat{p}_{g,s}} = \frac{\partial\left(\sum_{\hat{s}=1}^{S}(\hat{q}_{\hat{s}}\delta_{\hat{s}}p_{\hat{s},\hat{c}})\right)}{\partial \hat{p}_{g,s}} = \sum_{\hat{s}=1}^{S}\left(\delta_{\hat{s}}p_{\hat{s},\hat{c}}\frac{\partial\hat{q}_{\hat{s}}}{\partial\hat{p}_{g,s}}\right)$$
$$= \delta_s p_{s,\hat{c}}\frac{\partial\hat{q}_s}{\partial\hat{p}_{g,s}} = \frac{\delta_s p_{s,\hat{c}}\Lambda_g}{\gamma_s+\delta_s},$$

Then,

$$\frac{\partial K}{\partial p_{s,c}} = \frac{\partial\left(\sum_{\hat{c}=1}^{C}(\mu_{\hat{c}}q_{\hat{c}})\right)}{\partial p_{s,c}} = \frac{\partial(\mu_c q_c)}{\partial p_{s,c}} = \hat{q}_s\delta_s,$$
$$\frac{\partial K}{\partial \tilde{p}_{g,c}} = \frac{\partial\left(\sum_{\hat{c}=1}^{C}(\mu_{\hat{c}}q_{\hat{c}})\right)}{\partial \tilde{p}_{g,c}} = \frac{\partial(\mu_c q_c)}{\partial \tilde{p}_{g,c}} = \Lambda_g,$$
$$\frac{\partial K}{\partial \hat{p}_{g,s}} = \frac{\partial\left(\sum_{\hat{c}=1}^{C}(\mu_{\hat{c}}q_{\hat{c}})\right)}{\partial \hat{p}_{g,s}} = \sum_{\hat{c}=1}^{C}\left(\frac{\partial(\mu_{\hat{c}}q_{\hat{c}})}{\partial\hat{p}_{g,s}}\right)$$
$$= \sum_{\hat{c}=1}^{C}\left(\frac{\delta_s p_{s,\hat{c}}\Lambda_g}{\gamma_s+\delta_s}\right).$$

Then, the P-update formula can be

$$p_{s,c}^{(l+1)} = p_{s,c}^{(l)} + \eta\frac{\partial K}{\partial p_{s,c}} = p_{s,c}^{(l)} + \eta\hat{q}_s\delta_s,$$
$$\tilde{p}_{g,c}^{(l+1)} = \tilde{p}_{g,c}^{(l)} + \eta\frac{\partial K}{\partial \tilde{p}_{g,c}} = \tilde{p}_{g,c}^{(l)} + \eta\Lambda_g, \tag{20}$$
$$\hat{p}_{g,s}^{(l+1)} = \hat{p}_{g,s}^{(l)} + \eta\frac{\partial K}{\partial \hat{p}_{g,s}} = \hat{p}_{g,s}^{(l)} + \eta\sum_{c=1}^{C}\left(\frac{\delta_s p_{s,c}\Lambda_g}{\gamma_s+\delta_s}\right),$$

where l denotes the current number of iterations and $\eta>0$ is the step size. After each iteration, we make P satisfy the corresponding constraints with minimal adjustments via linear normalization or other techniques.

Remark 1. *Since we do not integrate the constraints of P into the derivation of the expression of $\partial K/\partial P$, the solutions found by the the P-update formula (20) may not be optimal. However, the solutions are close to the optimal solutions. Note that, since the step size η is fixed during the iterative procedure, it should be set as a small value, such as 0.5, to avoid oscillations.*

3.4. A heuristic algorithm: line-search aided non-negative least-square algorithm with weight initialization

In this subsection, we design a heuristic algorithm based on the results in Section 2, hoping to find acceptable solutions to the maximization problem (15).

In problem (15), we want to maximize $K=\sum_{c=1}^{C}(\mu_c q_c)$. Equivalently, we want to minimize $-K=-\sum_{c=1}^{C}(\mu_c q_c)$. Equivalently, we want to minimize $V-K=V-\sum_{c=1}^{C}(\mu_c q_c)$, where $V>0$ is a constant. Let $V=\sum_{c=1}^{C}\mu_c$. Then, equivalently, we can minimize $\sum_{c=1}^{C}(\mu_c-\mu_c q_c)$. It means that we want $\mu_c q_c$ to be as close to μ_c as possible (or q_c to be as close to 1 as possible). Then, we can design a heuristic algorithm as follows based on the results in Section 2.

We can obtain a linear system by substituting q_c with 1 and $\hat{q}_s$ with given parameters y_s into (14). However, this linear system may not have a unique solution. In order to approach equality as close as possible, we can solve the following non-negative least-square problem for P:

$$\text{minimize } f(P) = \text{minimize } \frac{1}{2}\|AP-b\|_2^2, \tag{21}$$

where matrix $A\in R^{(S+C)\times(SC+GC+GS)}$ and vector $b\in R^{(S+C)\times 1}$ are defined as

$A(\underline{\hat{h}}_s, h_{s,c}) = \hat{q}_s\delta_s, A(\underline{\hat{h}}_s, \hat{h}_{g,s}) = -\Lambda_g, A(\underline{\hat{h}}_s, \text{otherwise}) = 0,$

$A(\underline{h}_c, h_{s,c}) = \hat{q}_s\delta_s, A(\underline{h}_c, \tilde{h}_{g,c}) = \Lambda_g, A(\underline{h}_c, \text{otherwise}) = 0,$

$b(\underline{\hat{h}}_s) = -\hat{q}_s\gamma_s, b(\underline{h}_c) = q_c\mu_c,$

with $\underline{\hat{h}}_s = s$, $\underline{h}_c = S+c$, $h_{s,c} = (s-1)C+c$, $\tilde{h}_{g,c} = SC+(g-1)C+c$ and $\hat{h}_{g,s} = SC+GC+(g-1)S+s$. Then, the P-update formula can be

$$P^{(l+1)} = P^{(l)} - \eta\nabla f(P^{(l)}),$$

where $\nabla f(P^{(l)}) = A^{\mathrm{T}}(AP^{(l)}) - A^{\mathrm{T}}b$, l denotes the current number of iterations and $\eta>0$ is the step size. In addition, we substitute q_c with 1 and $\hat{q}_s$ with given parameters y_s in each iteration, where y_s associated with $\hat{q}_s$ are adjusted in the algorithm. Similarly, after each iteration, we make P satisfy the corresponding constraints with minimal adjustments. The procedure of the LNNLS algorithm for problem (21) is similar to Algorithm 1 and thus omitted.

The final heuristic algorithm to find solutions for problem (15) is designed based on Algorithm 1 and named as the LNNLS algorithm with weight initialization (LNNLS-W), where the procedure is similar to Algorithm 2.

3.5. Numerical results

The following numerical experiments are conducted to verify the analytic optimal solutions and results and test the performance of the algorithms. **All results are summarized into Table 2.** Note that we consider only the case that the energy is limited.

3.5.1. $G=2$, $S=2$ and $C=2$

Let us consider a simple EPN (14) with $G=2$, $S=2$ and $C=2$. We know that $\Lambda_1=0.07$, $\Lambda_2=0.08$, $\mu_1=1$, $\mu_2=10$, $\delta_1=0.19$, $\delta_2=0.2$, $\gamma_1=0.01$ and $\gamma_2=0.05$.

First, we use the analytic optimal solution in Subsection 3.2.1. The numerical result $K=0.15$, which is the same as the analytic optimal result $K^*=0.15$.

Since the dimension of P is small, we can randomly select the values of P for many trials (e.g., 20000 trials) to find an nearly optimal solution. Fig. 6 presents the values of K in these 20000 trials. The best result is $K=0.1496$.

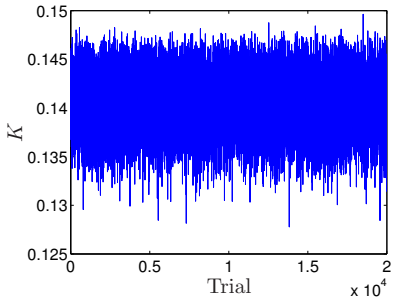


Figure 6: Values of K of the EPN (14) with $G = 2$, $S = 2$ and $C = 2$ using randomly generated P for energy distribution.

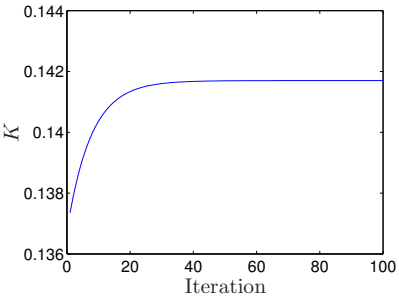


Figure 7: Values of K of the EPN (14) with $G = 2$, $S = 2$ and $C = 2$ using the gradient algorithm for energy distribution.

Fig. 7 presents the results by using the gradient algorithm, where the number of iterations is 100. The best result is $K = 0.1417$, which is close to the optimal result $K^* = 0.15$.

Fig. 8 presents the results by using the heuristic LNNLS-W, where $L = 5$ and $Z = 20$. The best result is $K = 0.15$, which is the same as the optimal result $K^* = 0.15$.

3.5.2. $G = 10$, $S = 5$ and $C = 16$

Let us consider an EPN (14) with $G = 10$, $S = 5$ and $C = 16$, where Λ_g, δ_s and γ_s with $\forall g \in G, s \in S$ are randomly generated in the ranges of $[0, 0.2]$, $[0, 1]$ and $[0, 0.1]$, respectively. In addition, we assume that the consumers differ from each other significantly. So, we set $\mu_c = 1 + 9(c-1)/(C-1), \forall c \in C$.

First, we use the analytic optimal solution in Subsection 3.2.1. The numerical result $K = 1.0931$, which is the same as the analytic optimal result $K^* = \sum_{g=1}^{10} \Lambda_g$.

We then randomly select the values of P for 20000 trials. However, since the dimension of P is not small, an nearly optimal solution may not be found. In these 20000 trials, the best result is $K = 1.0756$.

Fig. 9 presents the results by using the gradient algorithm, where the number of iterations is 100. The best result is $K = 1.0684$.

Fig. 10 presents the results by using the heuristic LNNLS-W, where $L = 5$ and $Z = 20$. The best result is $K = 1.0931$, which is the same as K^*.

More results are given in Table 2. In all cases except the case of $G = 2$, $S = 2$ and $C = 2$, Λ_g, δ_s and γ_s with $\forall g \in G, s \in S$ are randomly generated in the ranges of $[0, 0.2]$, $[0, 1]$ and $[0, 0.1]$, respectively, while $\mu_c = 1+9(c-1)/(C-1), \forall c \in C$. We can see that, among the algorithms, the the heuristic LNNLS-W performs the best.

3.6. An EPN with disconnections

Suppose C_1 consumers are too far to get energy directly from the generators (or say, these consumers are disconnected from the generators), where $0 < C_1 < C$. Let $C_2 = C - C_1$. The EPN can then be described by the following system of equations:

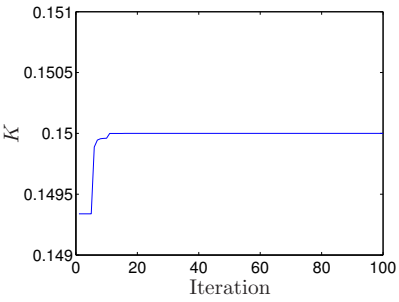


Figure 8: Values of K of the EPN (14) with $G = 2$, $S = 2$ and $C = 2$ using the LNNLS algorithm with weight initialization for energy distribution.

$$\begin{cases} \hat{q}_s = \dfrac{\sum_{g=1}^{G} (\Lambda_g \hat{p}_{g,s})}{\gamma_s + \sum_{c_1=1}^{C_1} (\delta_s \bar{p}_{s,c_1}) + \sum_{c_2=1}^{C_2} (\delta_s p_{s,c_2})} \\ \quad = \dfrac{\sum_{g=1}^{G} (\Lambda_g \hat{p}_{g,s})}{\gamma_s + \delta_s}, \\ \bar{q}_{c_1} = \dfrac{\sum_{s=1}^{S} (\hat{q}_s \delta_s \bar{p}_{s,c_1})}{\bar{\mu}_{c_1}}, \\ q_{c_2} = \dfrac{\sum_{g=1}^{G} (\Lambda_g \tilde{p}_{g,c_2}) + \sum_{s=1}^{S} (\hat{q}_s \delta_s p_{s,c_2})}{\mu_{c_2}}, \end{cases} \tag{22}$$

where $\forall s \in S$, $\forall c_1 \in C_1$ and $\forall c_2 \in C_2$. In addition, $\sum_{c_1=1}^{C_1} \bar{p}_{s,c_1} + \sum_{c_2=1}^{C_2} p_{s,c_2} = 1$ and $\sum_{c_2=1}^{C_2} \tilde{p}_{g,c_2} + \sum_{s=1}^{S} \hat{p}_{g,s} = 1$.

We want to maximize the amount of work done by the consumers per unit time in the EPN (22), or equivalently: given $\{\Lambda_g | g = 1, \cdots, G\}$, $\{\bar{\mu}_{c_1} | c_1 = 1, \cdots, C_1\}$, $\{\mu_{c_2} | c_2 = 1, \cdots, C_2\}$, $\{\delta_s | s = 1, \cdots, S\}$ and $\{\gamma_s | s = 1, \cdots, S\}$,

$$\text{maximize } K = \sum_{c_1=1}^{C_1} (\bar{\mu}_{c_1} q_{c_1}) + \sum_{c_2=1}^{C_2} (\mu_{c_2} q_{c_2}), \tag{23}$$

by using $\{\hat{p}_{g,s} | g = 1, \cdots, G; s = 1, \cdots, S\}$, $\{\tilde{p}_{g,c_2} | g = 1, \cdots, G; c_2 = 1, \cdots, C_2\}$, $\{\bar{p}_{s,c_1} | s = 1, \cdots, S; c_1 = 1, \cdots, C_1\}$ and $\{p_{s,c_2} | s = 1, \cdots, S; c_2 = 1, \cdots, C_2\}$.

3.7. Optimal solutions

Two cases need to be considered. One case is that the energy is limited. The other is that the energy is sufficient.

Table 2: K^* and the best K found by using different algorithms for energy distribution of the EPN (14) with different G, S and C

(G,S,C)	K^*	Optimal	Random	Gradient	LNNLS
$(2,2,2)$	**0.1500**	**0.1500**	0.1496	0.1417	**0.1500**
$(10,5,16)$	**1.0931**	**1.0931**	1.0756	1.0684	**1.0931**
$(10,10,30)$	**1.0976**	**1.0976**	1.0702	1.0755	**1.0976**
$(20,10,60)$	**1.8948**	**1.8948**	1.8565	1.8634	**1.8948**
$(100,100,300)$	**9.0363**	**9.0363**	N/A	8.7008	**9.0363**

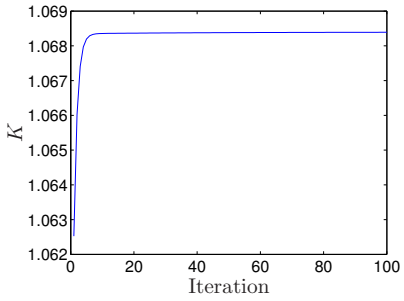


Figure 9: Values of K of the EPN (14) with $G = 10$, $S = 5$ and $C = 16$ using the gradient algorithm for energy distribution.

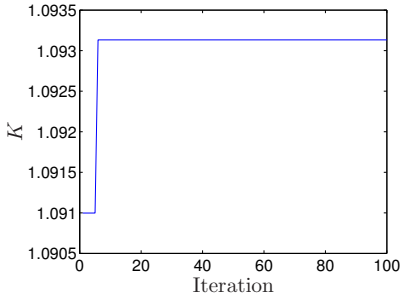


Figure 10: Values of K of the EPN (14) with $G = 10$, $S = 5$ and $C = 16$ using the LNNLS algorithm with weight initialization for energy distribution.

3.7.1. The energy is limited

In this case, the energy is so limited that $\sum_{g=1}^{G}\Lambda_g \leq \sum_{c_2=1}^{C_2}\mu_{c_2}$. Let us set $\hat{p}_{g,s} = 0, \forall g \in G, s \in S$. Since $\sum_{c_2=1}^{C_2}\tilde{p}_{g,c_2} + \sum_{s=1}^{S}\hat{p}_{g,s} = 1$, then $\sum_{c_2=1}^{C_2}\tilde{p}_{g,c_2} = 1$. Then, the system (22) becomes

$$\begin{cases} \hat{q}_s = 0, \\ \bar{q}_{c_1} = 0, \\ q_{c_2} = \dfrac{\sum_{g=1}^{G}(\Lambda_g\tilde{p}_{g,c_2})}{\mu_{c_2}}. \end{cases} \tag{24}$$

Let $H_2 = \sum_{c_2=1}^{C_2}\mu_{c_2}$. In addition, let us set

$$\tilde{p}_{g,c_2} = \frac{\mu_{c_2}}{H_2}, \forall c_2 \in C_2.$$

Then, we have

$$q_{c_2} = q_{\tilde{c}_2}, \forall c_1, \tilde{c}_2 \in C_2.$$

Since $\sum_{g=1}^{G}\Lambda_g \leq \sum_{c_2=1}^{C_2}\mu_{c_2}$, we have $q_{c_2} \leq 1, \forall c_2 \in C_2$.

Then,

$$K = \sum_{c_2=1}^{C_2}(\mu_{c_2}q_{c_2}) = \sum_{c_2=1}^{C_2}\sum_{g=1}^{G}(\Lambda_g\tilde{p}_{g,c_2}) = \sum_{g=1}^{G}\Lambda_g. \tag{25}$$

We can say that $K^* = \sum_{g=1}^{G}\Lambda_g$ is the optimal result for the problem (23), where **the optimal solution is** $\hat{p}_{g,s} = 0$ **and** $\tilde{p}_{g,c_2} = \mu_{c_2}/H_2$**,** $\forall g \in G, s \in S, c_2 \in C_2$**. In addition,** $\bar{p}_{s,c_1}, p_{s,c_2}$ $\forall s \in S, c_1 \in C_1, c_2 \in C_2$ **can be any non-negative value satisfying** $\sum_{c_1=1}^{C_1}\bar{p}_{s,c_1} + \sum_{c_2=1}^{C_2}p_{s,c_2} = 1$**.**

3.7.2. The energy is sufficient

In this case, the energy is sufficient such that $\sum_{c_2=1}^{C_2}\mu_{c_2} < \sum_{g=1}^{G}\Lambda_g \leq \sum_{c_2=1}^{C_2}\mu_{c_2} + \Upsilon_{s_1}$, where $s_1 = \arg\min_s \Upsilon_s = \arg\min_s(\gamma_s + \delta_s)$ with $s \in S$. First, we use part of the energy to make $q_{c_2} = 1, \forall c_2 \in C_2$. Let $B = \sum_{g=1}^{G}\Lambda_g$. Let us set $\tilde{p}_{g,c_2} = \mu_{c_2}/B, p_{s,c_2} = 0$ $\forall g \in G, s \in S, c_2 \in C_2$. Then, from (22),

$$q_{c_2} = \frac{\sum_{g=1}^{G}\left(\Lambda_g\frac{\mu_{c_2}}{B}\right)}{\mu_{c_2}} = 1, \forall c_2 \in C_2.$$

And, $\sum_{c_2=1}^{C_2}(\mu_{c_2}q_{c_2}) = \sum_{c_2=1}^{C_2}\mu_{c_2} = H_2$.

Let $H = \sum_{c_1=1}^{C_1}\bar{\mu}_{c_1}$. We then set $\bar{p}_{s,c_1} = \bar{\mu}_{c_1}/H, \forall s \in S, c_1 \in C_1$. Then, each $\bar{q}_{c_1}, \forall c_1 \in C_1$ is equal to each other. If $\bar{q}_{c_1} < 1, \forall c_1 \in C_1$, in view of $\sum_{c_1=1}^{C_1}\bar{p}_{s,c_1} = 1$ and (22), then

$$\sum_{c_1=1}^{C_1}(\bar{\mu}_{c_1}\bar{q}_{c_1}) = \sum_{c_1=1}^{C_1}\sum_{s=1}^{S}(\hat{q}_s\delta_s\bar{p}_{s,c_1})$$
$$= \sum_{c_1=1}^{C_1}\sum_{s=1}^{S}\left(\frac{\sum_{g=1}^{G}(\Lambda_g\hat{p}_{g,s})}{\gamma_s/\delta_s + 1}\bar{p}_{s,c_1}\right) = \sum_{s=1}^{S}\left(\frac{\sum_{g=1}^{G}(\Lambda_g\hat{p}_{g,s})}{\gamma_s/\delta_s + 1}\right).$$

To find a maximal value of $\sum_{c_1=1}^{C_1}(\bar{\mu}_{c_1}\bar{q}_{c_1})$, it is reasonable to make the denominators as large as possible while the numerators as small as possible. In addition, we know $\sum_{c_2=1}^{C_2}\tilde{p}_{g,c_2} + \sum_{s=1}^{S}\hat{p}_{g,s} = H_2/B + \sum_{s=1}^{S}\hat{p}_{g,s} = 1$. We then set $\hat{p}_{g,s_{\min}} = (B - H_2)/B$ with $s_{\min} = \arg\min_s(\gamma_s/\delta_s), \forall s \in S$ and $\hat{p}_{g,s} = 0$ with $s \in S$ and $s \neq s_{\min}$. Then,

$$\sum_{c_1=1}^{C_1}(\bar{\mu}_{c_1}\bar{q}_{c_1}) = \frac{\sum_{g=1}^{G}(\Lambda_g(B - H_2)/B)}{\gamma_{s_{\min}}/\delta_{s_{\min}} + 1} = \frac{B - H_2}{\gamma_{s_{\min}}/\delta_{s_{\min}} + 1}.$$

Then,

$$K^* = \sum_{c_1=1}^{C_1}(\bar{\mu}_{c_1}\bar{q}_{c_1}) + \sum_{c_2=1}^{C_2}(\mu_{c_2}q_{c_2}) = \frac{B-H_2}{\gamma_{s_{\min}}/\delta_{s_{\min}}+1} + \sum_{c_2=1}^{C_2}\mu_{c_2},\ \text{if } \bar{q}_{c_1}<1, \forall c_1 \in C_1,$$

or

$$K^* = \sum_{c_1=1}^{C_1}\bar{\mu}_{c_1} + \sum_{c_2=1}^{C_2}\mu_{c_2},\ \text{if } \bar{q}_{c_1}=1, \forall c_1 \in C_1, \tag{26}$$

may be the optimal result for the maximization problem (23), where **the optimal solution is** $\tilde{p}_{g,c_2} = \mu_{c_2}/B, p_{s,c_2} = 0$ $\forall g \in G, s \in S, c_2 \in C_2$**,** $\hat{p}_{g,s_{\min}} = (B-H_2)/B$ **with** $s_{\min} = \arg\min_s(\gamma_s/\delta_s), \forall s \in S$ **and** $\hat{p}_{g,s} = 0$ **with** $s \in S$ **and** $s \neq s_{\min}$**. In addition,** $\bar{p}_{s,c_1} = \bar{\mu}_{c_1}/H, \forall s \in S, c_1 \in C_1$**.**

Remark 2. *If $C_1 = C$, the system becomes*

$$\begin{cases} \hat{q}_s = \dfrac{\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})}{\gamma_s + \delta_s} \\ \bar{q}_{c_1} = \dfrac{\sum_{s=1}^{S}(\hat{q}_s\delta_s\bar{p}_{s,c_1})}{\bar{\mu}_{c_1}}. \end{cases} \tag{27}$$

Then, the optimal result may be $K^* = B\delta_{s_{min}}/(\gamma_{s_{min}} + \delta_{s_{min}})$ if $\bar{q}_{c_1} < 1, \forall c_1 \in C_1$ or $K^* = \sum_{c_1=1}^{C_1}\bar{\mu}_{c_1}$ if $\bar{q}_{c_1} = 1, \forall c_1 \in C_1$, where **the optimal solution is** $\hat{p}_{g,s_{\min}} = 1$ **with** $s_{\min} = \arg\min_s(\gamma_s/\delta_s), \forall s \in S$ **and** $\hat{p}_{g,s} = 0$ **with** $s \in S$ **and** $s \neq s_{\min}$**. In addition,** $\bar{p}_{s,c_1} = \bar{\mu}_{c_1}/H\ \forall s \in S, c_1 \in C_1$**.**

Remark 3. *It can be complicate to analyze the optimal result and optimal solution for the maximization problem (23) if the energy is sufficient such that $\sum_{g=1}^{G}\Lambda_g > \sum_{c_2=1}^{C_2}\mu_{c_2} + \Upsilon_{s_1}$ with $s_1 = \arg\min_s(\gamma_s+\delta_s), \forall s \in S$. In this case, designing an optimization algorithm to find approximate solutions may be a more practical choice.*

3.8. Cooperative particle swarm optimizer

It can be difficult to apply the gradient algorithm to solving the maximization problem (23). Thus, we choose to use a simply yet effective algorithm, term the cooperative particle swarm optimizer (or namely, the CPSO-S algorithm) in [105] that is first introduced by Van den Bergh and Engelbrecht [108].

First, let us define $\bar{P} \in R^{(SC+GC_2+GS)\times 1}$ as a vector that consists of $\hat{p}_{g,s}$, $\tilde{p}_{g,c_1}$, $\bar{p}_{s,c_1}$ and p_{s,c_2}, where

$$\bar{P}(h_{1s,c_1}) = \bar{p}_{s,c_1}, \bar{P}(h_{2s,c_2}) = p_{s,c_2},$$
$$\bar{P}(\tilde{h}_{g,c_2}) = \tilde{p}_{g,c_2}, \bar{P}(\hat{h}_{g,s}) = \hat{p}_{g,s},$$

with $h_{1s,c_1} = (s-1)C_1 + c_1$, $h_{2s,c_2} = SC_1 + (s-1)C_2 + c_2$, $\tilde{h}_{g,c_2} = SC+(g-1)C_2+c_2$ and $\hat{h}_{g,s} = SC+GC_2+(g-1)S+s$.

Let $M = SC + GC_2 + GS$ and N respectively denote the number of swarms and the number of particles in each swarm, where N is selected by the algorithm user. Let $\bar{P}_n(m)$ and $\tilde{\bar{P}}_n(m)$ with $\forall m \in M, n \in N$ respectively denote the current state and best state of the nth particle in the mth swarm. Let $\bar{P}^{(\text{best})}$ denote the best $\bar{P}$ found so far. Assume that $\kappa_n \sim U(0,1)$ and $\hat{\kappa}_n \sim U(0,1)$ with $\forall n \in N$ are uniform random sequences in the range $(0,1)$, which will be updated after each iteration, then, the state-update formula can be

$$\bar{P}_n^{(l+1)}(m) = \bar{P}_n^{(l+1)}(m) + v_{n,m}^{(l+1)}, \tag{28}$$

where

$$v_{n,m}^{(l+1)} = \omega v_{n,m}^{(l)} + 2\kappa_n^{(l)}(\tilde{\bar{P}}_n^{(l)}(m) - \tilde{P}_n^{(l)}(m)) + 2\bar{\kappa}_n^{(l)}(\bar{P}^{(\text{best})}(m) - \bar{P}_n^{(l)}(m)),$$

ω is called the inertia weight and setup to vary from 1 to near 0 during the search process, l denotes the current number of iterations. The CPSO-S algorithm for solving problem (23) is similar to the one in [105] and thus omitted here.

Remark 4. *The CPSO-S algorithm performs well in solving the problem (23). But, it requires solving the system (22) for $3MN$ times in a single iteration, which can be quite time-consuming when dealing with a system with high dimensions.*

3.9. A heuristic algorithm: LNNLS-W

To avoid text repetition, the following presents briefly how a heuristic algorithm named as the LNNLS-W used to find solutions for the maximization problem (23) is designed based on the design procedure in Subsection 3.4.

From (22), the following non-negative least-square problem is first presented:

$$\text{minimize } \bar{f}(\bar{P}) = \text{minimize } \frac{1}{2}\|\bar{A}\bar{P} - \bar{b}\|_2^2,$$

where matrix $\bar{A} \in R^{(S+C)\times(SC+GC_2+GS)}$ and vector $\bar{b} \in R^{(S+C)\times 1}$ are defined as

$$\bar{A}(\underline{\hat{h}}_s, h_{1s,c_1}) = \bar{A}(\underline{\hat{h}}_s, h_{2s,c_1}) = \hat{q}_s\delta_s,$$
$$\bar{A}(\underline{\hat{h}}_s, \hat{h}_{g,s}) = -\Lambda_g, \bar{A}(\underline{\hat{h}}_s, \text{otherwise}) = 0,$$
$$\bar{A}(\underline{h}_{1c_1}, h_{1s,c_1}) = \bar{A}(\underline{h}_{2c_2}, h_{2s,c_2}) = \hat{q}_s\delta_s,$$
$$\bar{A}(\underline{h}_{1c_1}, \text{otherwise}) = 0,$$
$$\bar{A}(\underline{h}_{2c_2}, \tilde{h}_{g,c_2}) = \Lambda_g, \bar{A}(\underline{h}_{2c_2}, \text{otherwise}) = 0,$$
$$\bar{b}(\underline{\hat{h}}_s) = -\hat{q}_s\gamma_s, \bar{b}(\underline{h}_{1c_1}) = \bar{q}_{c_1}\bar{\mu}_{c_1}, \bar{b}(\underline{h}_{2c_2}) = q_{c_2}\mu_{c_2},$$

with $\underline{\hat{h}}_s = s$, $\underline{h}_{1c_1} = S + c_1$ and $\underline{h}_{2c_2} = S + C_1 + c_2$, $\forall c_1 \in C_1, c_2 \in C_2$. To solve the above non-negative least-square problem, the $\bar{P}$-update formula can be

$$\bar{P}^{(l+1)} = \bar{P}^{(l)} - \eta\nabla\bar{f}(\bar{P}^{(l)}),$$

where $\nabla\bar{f}(\bar{P}^{(l)}) = \bar{A}^{\mathrm{T}}(\bar{A}\bar{P}^{(l)}) - \bar{A}^{\mathrm{T}}\bar{b}$, l denotes the current number of iterations and $\eta > 0$ is the step size. In addition, we substitute $\bar{q}_{c_1}, q_{c_2}$ with 1 and $\hat{q}_s$ with given parameters y_s in each iteration, where y_s associated with $\hat{q}_s$ are adjusted in the algorithm. After each iteration, we make $\bar{P}$ satisfy the corresponding constraints with minimal adjustments.

The detailed procedure of the heuristic LNNLS-W for solving problem (23) is similar to Algorithm 2 and thus omitted here.

Remark 5. *Initial values of the parameters y_s with $s = 1, \cdots, S$ affect the performance of this algorithm for solving problem (23). Multi trials with different initial y_s can be conducted to find better results.*

3.10. Numerical results

In this subsection, $\bar{\mu}_{c_1} = 1 + 9(c_1 - 1)/(C_1 - 1), \forall c_1 \in C_1$, $\mu_{c_2} = 1 + 9(c_2 - 1)/(C_2 - 1), \forall c_2 \in C_2$, $C_1 > 1$, $C_2 > 1$. In addition, δ_s and γ_s with $\forall s \in S$ are randomly generated in the ranges of $[8, 10]$ and $[1, 2]$, respectively.

3.10.1. The energy is limited

In this case, $\sum_{g=1}^{G} \Lambda_g \leq \sum_{c_2=1}^{C_2} \mu_{c_2} = H_2$ and $\Lambda_g, \forall g \in G$ is randomly generated in the range of $[0, H_2/G]$.

Table 3 summaries the results by using the optimal solutions in Subsection 3.7.1, the randomly-generated solutions (20000 trials), the CPSO algorithm ($L = 50, N = 3$) and the heuristic LNNLS-W ($L = 1, Z = 100$ and 100 trials) to solve problem (23).

3.10.2. The energy is sufficient

In this case, $H_2 < \sum_{g=1}^{G} \Lambda_g \leq H_2 + \Upsilon_{s_1}$ with $s_1 = \arg\min_s(\gamma_s + \delta_s), \forall s \in S$ and $\Lambda_g, \forall g \in G$ is randomly generated in the range of $[H_2/G, (H_2 + \Upsilon_{s_1})/G]$.

Table 4 summaries the results by using the randomly-generated solutions (20000 trials), the CPSO algorithm ($L = 50, N = 3$) and the heuristic LNNLS-W ($L = 1, Z = 100$ and 10 trials) to solve problem (23).

3.10.3. The energy is more sufficient

In this case, $\sum_{g=1}^{G} \Lambda_g > H_2 + \Upsilon_{s_1}$ with $s_1 = \arg\min_s(\gamma_s + \delta_s), \forall s \in S$ and $\Lambda_g, \forall g \in G$ is randomly generated in the range of $[(H_2 + \Upsilon_{s_1})/G, (H_2 + \sum_{s=1}^{S}(\gamma_s + \delta_s))/G]$.

Table 5 summaries the results by using the optimal solutions in Subsection 3.7.2, the randomly-generated solutions (20000 trials), the CPSO algorithm ($L = 50, N = 3$) and the LNNLS algorithm with weight initialization ($L = 1, Z = 100$ and 10 trials) to solve problem (23).

3.11. A system that provides energy on demand

The EPN that provides energy on demand can be described by the system of equations:

$$\begin{cases} \hat{q}_s = \dfrac{\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})}{\gamma_s + \sum_{c=1}^{C}(q_c \mu_c r_{c,s} \phi_{s,c} p_{s,c})}, \\ q_c = \dfrac{\sum_{s=1}^{S}(\hat{q}_s q_c \mu_c r_{c,s} \phi_{s,c} p_{s,c}) + \sum_{g=1}^{G}(\Lambda_g \tilde{p}_{g,c})}{\mu_c}, \end{cases} \tag{29}$$

with $\forall s \in S, c \in C$, the request probabilities $\sum_{s=1}^{S} r_{c,s} \leq 1$, the allocation probabilities $\sum_{c=1}^{C} p_{s,c} \leq 1$, and the allocation fractions $\phi_{s,c}$ which may be greater than or smaller than one, depending on how a particular storage unit reacts to the requests from a specific consumer. In particular, we will expect that the total energy provision rate for any given storage unit will be bounded in some way:

$$\hat{q}_s \sum_{c=1}^{C} q_c \mu_c r_{c,s} \phi_{s,c} p_{s,c} \leq \Gamma_s.$$

We want to maximize the amount of work done by the consumers per unit time in the EPN (29), or equivalently: given $\{\Lambda_g | g = 1, \cdots, G\}$, $\{\mu_c | c = 1, \cdots, C\}$, $\{\gamma_s | s = 1, \cdots, S\}$ and $\{\Gamma_s | s = 1, \cdots, S\}$,

$$\text{maximize } K = \sum_{c=1}^{C} (\mu_c q_c), \tag{30}$$

by using $\{\hat{p}_{g,s} | g = 1, \cdots, G; s = 1, \cdots, S\}$, $\{\tilde{p}_{g,c} | g = 1, \cdots, G; c = 1, \cdots, C\}$, $\{p_{s,c} | s = 1, \cdots, S; c = 1, \cdots, C\}$, $\{r_{c,s} | c = 1, \cdots, C; s = 1, \cdots, S\}$ and $\phi_{s,c} | s = 1, \cdots, S; c = 1, \cdots, C\}$ with constraints $\hat{p}_{g,s}, \tilde{p}_{g,c}, p_{s,c}, r_{c,s}, \phi_{s,c} \geq 0$, $\sum_{c=1}^{C} p_{s,c} \leq 1$, $\sum_{c=1}^{C} \tilde{p}_{g,c} + \sum_{s=1}^{S} \hat{p}_{g,s} = 1$, $\sum_{s=1}^{S} r_{c,s} \leq 1$ and $\hat{q}_s \sum_{c=1}^{C} q_c \mu_c r_{c,s} \phi_{s,c} p_{s,c} \leq \Gamma_s$.

3.12. Optimal solutions

Two cases need to be considered.

In the first case, the energy is limited such that $\sum_{g=1}^{G} \Lambda_g \leq \sum_{c=1}^{C} \mu_c$. In view of the analysis in Subsection 3.2.1, we know that $K^* = \sum_{g=1}^{G} \Lambda_g$ is the optimal result for the maximization problem (30).

In the second case, the energy is sufficient such that $\sum_{g=1}^{G} \Lambda_g > \sum_{c=1}^{C} \mu_c$. We know that $K^* = \sum_{c=1}^{C} \mu_c$ is the optimal result for the maximization problem (30).

Note that in both cases, **the optimal solution is** $\hat{p}_{g,s} = 0$ **and** $\tilde{p}_{g,c} = \mu_c / H$ **with** $H = \sum_{c=1}^{C} \mu_c$**,** $\forall g \in G, s \in S, c \in C$**. In addition,** $p_{s,c}$ **and** $r_{c,s}, \forall s \in S, c \in C$ **can be any non-negative value satisfying** $\sum_{c=1}^{C} p_{s,c} \leq 1$ **and** $\sum_{s=1}^{S} r_{c,s} \leq 1$**, while** $\phi_{s,c}, \forall s \in S, c \in C$ **can be any positive value since** $\hat{q}_s \sum_{c=1}^{C} q_c \mu_c r_{c,s} \phi_{s,c} p_{s,c} \leq \Gamma_s$ **always holds with** $\hat{q}_s = 0$**.**

3.13. Cooperative particle swarm optimizer

First, let us define $\tilde{P} \in R^{(3SC+GC+GS)\times 1}$ as a vector that consists of $\hat{p}_{g,s}$, $\tilde{p}_{g,c}$, $p_{s,c}$, $r_{c,s}$ and $\phi_{s,c}$, where

$$\tilde{P}(h_{s,c}) = p_{s,c}, \tilde{P}(\tilde{h}_{g,c}) = \tilde{p}_{g,c}, \tilde{P}(\hat{h}_{g,s}) = \hat{p}_{g,s},$$
$$\tilde{P}(\bar{h}_{c,s}) = r_{c,s}, \tilde{P}(\bar{\bar{h}}_{s,c}) = \phi_{s,c},$$

with $\forall s \in S, c \in C, g \in G$, $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$, $\hat{h}_{g,s} = SC + GC + (g-1)S + s$, $\bar{h}_{c,s} = SC + GC + GS + (s-1)C + c$ and $\bar{\bar{h}}_{s,c} = 2SC + GC + GS(s-1)C + c$.

Let $M = 3SC + GC + GS$ and N respectively denote the number of swarms and the number of particles in each swarm, where N is selected by the algorithm user. Let $\tilde{P}_n(m)$ and $\bar{\tilde{P}}_n(m)$ with $\forall m \in M, n \in N$ respectively denote the current state and best state of the nth particle in the mth swarm. Let $\tilde{P}^{(\text{best})}$ denote the best $\tilde{P}$ found so far. Assume that

Table 3: K^* and the best K found by using different algorithms for energy distribution of the EPN (22) with different G, S, C_1 and C_2 when the energy is limited

(G, S, C_1, C_2)	K^*	Optimal	Random	CPSO-S	LNNLS
$(2, 2, 2, 2)$	**7.3110**	**7.3110**	7.0086	**7.3110**	**7.3110**
$(10, 5, 8, 8)$	**14.6502**	**14.6502**	14.0626	14.6498	**14.6502**
$(10, 10, 15, 15)$	**38.3693**	**38.3693**	36.1363	38.3254	**38.3693**
$(20, 10, 30, 30)$	**68.3396**	**68.3396**	65.2869	68.3377	**68.3396**
$(100, 10, 150, 150)$	**416.1667**	**416.1667**	N/A	N/A	**416.1667**

Table 4: K^* and the best K found by using different algorithms for energy distribution of the EPN (22) with different G, S, C_1 and C_2 when the energy is sufficient

(G, S, C_1, C_2)	K^*	Optimal	Random	CPSO-S	LNNLS
$(2, 2, 2, 2)$	**15.4128**	**15.4128**	14.4421	15.2019	15.3463
$(10, 5, 8, 8)$	**48.6020**	**48.6020**	42.7710	48.3117	48.4840
$(10, 10, 15, 15)$	**86.2535**	**86.2535**	74.8169	85.7801	86.1077
$(20, 10, 30, 30)$	**168.4479**	**168.4479**	142.1887	168.0898	168.3106
$(100, 10, 150, 150)$	**828.9338**	**828.9338**	N/A	N/A	828.6729

$\kappa_n \sim U(0,1)$ and $\hat{\kappa}_n \sim U(0,1)$ with $\forall n \in N$ are uniform random sequences in the range $(0,1)$, which will be updated after each iteration, then, the state-update formula can be

$$\tilde{P}_n^{(l+1)}(m) = \tilde{P}_n^{(l+1)}(m) + v_{n,m}^{(l+1)}, \tag{31}$$

where

$$\begin{aligned} v_{n,m}^{(l+1)} =& \omega v_{n,m}^{(l)} + 2\kappa_n^{(l)}(\bar{\tilde{P}}_n^{(l)}(m) - \tilde{P}_n^{(l)}(m)) \\ &+ 2\bar{\kappa}_n^{(l)}(\tilde{P}^{(\text{best})}(m) - \tilde{P}_n^{(l)}(m)), \end{aligned}$$

ω is called the inertia weight and setup to vary from 1 to near 0 during the search process, l denotes the current number of iterations. The CPSO-S algorithm for solving problem (30) is similar to the one in [105] and thus omitted here.

3.14. A heuristic algorithm: LNNLS-W

First, let us define $\chi_{s,c} = r_{c,s}\phi_{s,c}p_{s,c}$. Then, the system of equations (29) can be rewritten as

$$\begin{cases} \hat{q}_s = \dfrac{\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})}{\gamma_s + \sum_{c=1}^{C}(q_c\mu_c\chi_{s,c})}, \\ q_c = \dfrac{\sum_{s=1}^{S}(\hat{q}_s q_c \mu_c \chi_{s,c}) + \sum_{g=1}^{G}(\Lambda_g \tilde{p}_{g,c})}{\mu_c}. \end{cases} \tag{32}$$

In addition, the constrain of $\hat{q}_s \sum_{c=1}^{C} q_c\mu_c r_{c,s}\phi_{s,c}p_{s,c} \le \Gamma_s$ becomes $\hat{q}_s \sum_{c=1}^{C} q_c\mu_c\chi_{s,c} \le \Gamma_s$.

Then, let us define $\hat{P} \in R^{(SC+GC+GS)\times 1}$ as a vector that consists of $\hat{p}_{g,s}$, $\tilde{p}_{g,c}$ and $\chi_{s,c}$, where

$$\hat{P}(h_{s,c}) = \chi_{s,c}, \hat{P}(\tilde{h}_{g,c}) = \tilde{p}_{g,c}, \hat{P}(\hat{h}_{g,s}) = \hat{p}_{g,s},$$

with $\forall s \in S, c \in C, g \in G$, $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$ and $\hat{h}_{g,s} = SC + GC + (g-1)S + s$.

Based on Section 3.4, a heuristic algorithm named as LNNLS-W can also be designed. First, the following non-negative least-square problem is presented:

$$\text{minimize } \hat{f}(\hat{P}) = \text{minimize } \frac{1}{2}\|\hat{A}\hat{P} - \hat{b}\|_2^2, \tag{33}$$

where matrix $\hat{A} \in R^{(S+C)\times(SC+GC+GS)}$ and vector $\hat{b} \in R^{(S+C)\times 1}$ are defined as

$$\hat{A}(\underline{\hat{h}}_s, h_{s,c}) = \hat{q}_s q_c \mu_c, \hat{A}(\underline{\hat{h}}_s, \hat{h}_{g,s}) = -\Lambda_g, \hat{A}(\underline{\hat{h}}_s, \text{otherwise}) = 0,$$
$$\hat{A}(\underline{h}_c, h_{s,c}) = \hat{q}_s q_c \mu_c, \hat{A}(\underline{h}_c, \tilde{h}_{g,c}) = \Lambda_g, \hat{A}(\underline{h}_c, \text{otherwise}) = 0,$$
$$\hat{b}(\underline{\hat{h}}_s) = -\hat{q}_s\gamma_s, \hat{b}(\underline{h}_c) = q_c\mu_c,$$

with $\forall s \in S, c \in C, g \in G$, $\underline{\hat{h}}_s = s$, $\underline{h}_c = S + c$, $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$ and $\hat{h}_{g,s} = SC + GC + (g-1)S + s$. To solve problem (33), the $\hat{P}$-update formula can be

$$\hat{P}^{(l+1)} = \hat{P}^{(l)} - \eta\nabla\hat{f}(\hat{P}^{(l)}),$$

where

$$\nabla\hat{f}(\hat{P}^{(l)}) = \hat{A}^{\mathrm{T}}(\hat{A}\hat{P}^{(l)}) - \hat{A}^{\mathrm{T}}\hat{b},$$

l denotes the current number of iterations and $\eta > 0$ is the step size. In addition, we substitute q_c with 1 and $\hat{q}_s$ with given parameters y_s in each iteration, where y_s associated with $\hat{q}_s$ are adjusted in the algorithm. After each iteration, we make $\hat{P}$ satisfy the corresponding constraints with minimal adjustments. The detailed procedure of the heuristic LNNLS-W for the problem (30) is given in Algorithm 4.

Remark 6. *A caveat here is that Algorithm 4 is only effective under the condition that $\chi_{s,c}$ that makes $r_{c,s}$, $\phi_{s,c}$ and $p_{s,c}$ a whole can be defined. Algorithm 4 does not adjust $r_{c,s}$ and $p_{c,s}$ in the whole searching process, which simplifies the process.*

Table 5: K^* and the best K found by using different algorithms for energy distribution of the EPN (22) with different G, S, C_1 and C_2 when the energy is more sufficient

(G, S, C_1, C_2)	K^*	Optimal	Random	CPSO-S	LNNLS
$(2, 2, 2, 2)$	N/A	N/A	21.6368	**22.0000**	**22.0000**
$(10, 5, 8, 8)$	N/A	N/A	56.0484	68.6422	**69.7767**
$(10, 10, 15, 15)$	N/A	N/A	101.2606	123.6435	**125.8052**
$(20, 10, 30, 30)$	N/A	N/A	176.4236	220.0482	**222.4845**
$(100, 10, 150, 150)$	N/A	N/A	N/A	N/A	**1285.6231**

Algorithm 4 LNNLS-W for solving (30)

Use Algorithm 2 to obtain $\hat{P}^{(\text{best})}$;
$r_{c,s}^{(\text{best})} \leftarrow 1/S, \forall c \in C, s \in S$;
$p_{s,c}^{(\text{best})} \leftarrow 1/C, \forall s \in S, c \in C$;
$\phi_{s,c}^{(\text{best})} \leftarrow \chi_{c,s}^{(\text{best})}/(r_{c,s}^{(\text{best})} p_{s,c}^{(\text{best})}), \forall s \in S, c \in C$.

3.15. Another heuristic algorithm: multi line-search aided non-negative least-square algorithm with weight initialization

In this subsection, we generalize the heuristic LNNLS-W and design another similar but more general heuristic algorithm, named as multi line-search aided non-negative least-square algorithm with weight initialization (MLNNLS-W).

First, let us define $P_1, P_2, P_3 \in R^{(SC+GC+GS)\times 1}$ as vectors that consist of $\hat{p}_{g,s}$, $\tilde{p}_{g,c}$, $p_{s,c}$, $r_{c,s}$ and $\phi_{s,c}$, where

$$\begin{aligned}
&P_1(h_{s,c}) = p_{s,c}, P_2(h_{s,c}) = r_{c,s}, P_3(h_{s,c}) = \phi_{s,c},\\
&P_1(\tilde{h}_{g,c}) = P_2(\tilde{h}_{g,c}) = P_3(\tilde{h}_{g,c}) = \tilde{p}_{g,c},\\
&P_1(\hat{h}_{g,s}) = P_2(\hat{h}_{g,s}) = P_3(\hat{h}_{g,s}) = \hat{p}_{g,s},
\end{aligned}$$

with $\forall s \in S, c \in C, g \in G$, $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$ and $\hat{h}_{g,s} = SC + GC + (g-1)S + s$.

Then, based on Section 3.4, the following three non-negative least-square problems are presented based on the system of equations (29):

$$\begin{aligned}
&\text{minimize } f_1(P_1) = \text{minimize } \frac{1}{2}\|A_1P_1 - b_1\|_2^2,\\
&\text{minimize } f_2(P_2) = \text{minimize } \frac{1}{2}\|A_2P_2 - b_2\|_2^2,\\
&\text{minimize } f_3(P_3) = \text{minimize } \frac{1}{2}\|A_3P_3 - b_3\|_2^2,
\end{aligned}$$

where matrix $A_1, A_2, A_3 \in R^{(S+C)\times(SC+GC+GS)}$ and vector $b_1, b_2, b_3 \in R^{(S+C)\times 1}$ are defined as

$$\begin{aligned}
&A_1(\hat{\underline{h}}_s, h_{s,c}) = \hat{q}_s q_c \mu_c r_{c,s} \phi_{s,c},\\
&A_2(\hat{\underline{h}}_s, h_{s,c}) = \hat{q}_s q_c \mu_c \phi_{s,c} p_{s,c},\\
&A_3(\hat{\underline{h}}_s, h_{s,c}) = \hat{q}_s q_c \mu_c r_{c,s} p_{s,c},\\
&A_1(\hat{\underline{h}}_s, \hat{h}_{g,s}) = A_2(\hat{\underline{h}}_s, \hat{h}_{g,s}) = A_3(\hat{\underline{h}}_s, \hat{h}_{g,s}) = -\Lambda_g,\\
&A_1(\hat{\underline{h}}_s, \text{otherwise}) = A_2(\hat{\underline{h}}_s, \text{otherwise})\\
&= A_3(\hat{\underline{h}}_s, \text{otherwise}) = 0,\\
&A_1(\underline{h}_c, h_{s,c}) = \hat{q}_s q_c \mu_c r_{c,s} \phi_{s,c},\\
&A_2(\underline{h}_c, h_{s,c}) = \hat{q}_s q_c \mu_c \phi_{s,c} p_{s,c},\\
&A_3(\underline{h}_c, h_{s,c}) = \hat{q}_s q_c \mu_c r_{c,s} p_{s,c},\\
&A_1(\underline{h}_c, \tilde{h}_{g,c}) = A_2(\underline{h}_c, \tilde{h}_{g,c}) = A_3(\underline{h}_c, \tilde{h}_{g,c}) = \Lambda_g,\\
&A_1(\underline{h}_c, \text{otherwise}) = A_2(\underline{h}_c, \text{otherwise})\\
&= A_3(\underline{h}_c, \text{otherwise}) = 0,\\
&b_1(\hat{\underline{h}}_s) = b_2(\hat{\underline{h}}_s) = b_3(\hat{\underline{h}}_s) = -\hat{q}_s \gamma_s,\\
&b_1(\underline{h}_c) = b_2(\underline{h}_c) = b_3(\underline{h}_c) = q_c \mu_c,
\end{aligned}$$

with $\forall s \in S, c \in C, g \in G$, $\hat{\underline{h}}_s = s$, $\underline{h}_c = S + c$, $h_{s,c} = (s-1)C + c$, $\tilde{h}_{g,c} = SC + (g-1)C + c$ and $\hat{h}_{g,s} = SC + GC + (g-1)S + s$.

Then, the update formulas of P_1, P_2 and P_3 can be

$$\begin{aligned}
P_1^{(l+1)} &= P_1^{(l)} - \eta \nabla f(P_1^{(l)}),\\
P_2^{(l+1)} &= P_2^{(l)} - \eta \nabla f(P_2^{(l)}),\\
P_3^{(l+1)} &= P_3^{(l)} - \eta \nabla f(P_3^{(l)}),
\end{aligned}$$

where

$$\begin{aligned}
\nabla f_1(P_1^{(l)}) &= A_1^{\mathrm{T}}(A_1 P_1^{(l)}) - A_1^{\mathrm{T}} b_1,\\
\nabla f_2(P_2^{(l)}) &= A_2^{\mathrm{T}}(A_2 P_2^{(l)}) - A_2^{\mathrm{T}} b_2,\\
\nabla f_3(P_3^{(l)}) &= A_3^{\mathrm{T}}(A_3 P_3^{(l)}) - A_3^{\mathrm{T}} b_3,
\end{aligned}$$

l denotes the current number of iterations and $\eta > 0$ is the step size. In addition, we substitute q_c with 1 and $\hat{q}_s$ with given parameters y_s in each iteration, where y_s associated with $\hat{q}_s$ are adjusted in the algorithm. After each iteration, we make P_1, P_2 and P_3 satisfy the corresponding constraints with minimal adjustments.

Finally, the more general heuristic algorithm (the MLNNLS-W) for solving the problem (30) can be designed with the procedure given in Algorithm 5.

Algorithm 5 The MLNNLS algorithm with weight initialization

Initialize $P_1^{(1)}$, $P_2^{(1)}$, $P_3^{(1)}$ and $y_s, \forall s \in S$;
$P_1^{(\text{best})} \leftarrow P_1^{(1)}$, $P_2^{(\text{best})} \leftarrow P_2^{(1)}$, $P_3^{(\text{best})} \leftarrow P_3^{(1)}$, $x \leftarrow 1$;
while $x \leq X$ **do**
  use Algorithm 2 to obtain $P_1^{(\text{best})}$ with $P_1^{(1)}$, $r_{c,s}^{(\text{best})}$, $\phi_{s,c}^{(\text{best})}$ and $y_s, \forall s \in S, c \in C$;
  solve (29) using $P_1^{(\text{best})}$ for $q_s, \forall s \in S$;
  $p_{s,c}^{(1)} \leftarrow p_{s,c}^{(\text{best})}, \tilde{p}_{g,c}^{(1)} \leftarrow \tilde{p}_{g,c}^{(\text{best})}, \hat{p}_{g,s}^{(1)} \leftarrow \hat{p}_{g,s}^{(\text{best})}, y_s \leftarrow q_s, \forall s \in S, c \in C, g \in G$;
  use Algorithm 2 to obtain $P_2^{(\text{best})}$ with $P_2^{(1)}$ $p_{s,c}^{(\text{best})}$, $\phi_{s,c}^{(\text{best})}$ and $y_s, \forall s \in S, c \in C$;
  solve (29) using $P_2^{(\text{best})}$ for $q_s, \forall s \in S$;
  $r_{c,s}^{(1)} \leftarrow r_{c,s}^{(\text{best})}, \tilde{p}_{g,c}^{(1)} \leftarrow \tilde{p}_{g,c}^{(\text{best})}, \hat{p}_{g,s}^{(1)} \leftarrow \hat{p}_{g,s}^{(\text{best})}, y_s \leftarrow q_s, \forall s \in S, c \in C, g \in G$;
  use Algorithm 2 to obtain $P_3^{(\text{best})}$ with $P_3^{(1)}$, $p_{s,c}^{(\text{best})}$, $r_{c,s}^{(\text{best})}$ and $y_s, \forall s \in S, c \in C$;
  solve (29) using $P_1^{(\text{best})}$ for $q_s, \forall s \in S$;
  $\phi_{s,c}^{(1)} \leftarrow \phi_{s,c}^{(\text{best})}, \tilde{p}_{g,c}^{(1)} \leftarrow \tilde{p}_{g,c}^{(\text{best})}, \hat{p}_{g,s}^{(1)} \leftarrow \hat{p}_{g,s}^{(\text{best})}, y_s \leftarrow q_s, \forall c \in C, s \in S$;
  $x \leftarrow x + 1$;
end while

3.16. Numerical results

For reading convenience, numerical results are summarized into Table 6. Note that we consider only the case that the energy is limited.

When $G = 2$, $S = 2$ and $C = 2$, $\Lambda_1 = 0.07$, $\Lambda_2 = 0.08$, $\mu_1 = 1$, $\mu_2 = 10$, $\gamma_1 = 0.01$, $\gamma_2 = 0.05$, $\Gamma_1 = 6$ and $\Gamma_2 = 8$. In all other cases, $\mu_c = 1 + 9(c-1)/(C-1), \forall c \in C$, Λ_g, γ_s and Γ_s with $\forall g \in G, s \in S$ are randomly generated in the ranges of $[0, 1]$, $[0, 0.5]$ and $[5, 10]$, respectively. These results in Table 6 have well verified the analytic optimal solutions and demonstrated the efficacy of the algorithms for energy distribution of EPN (29). In addition, all three algorithms can be good choices for the EPN with low dimensions (small G, S and C), while the heuristic algorithms, i.e., the LNNLS-W and MLNNLS-W, are more applicable if the dimensions of the EPN are high.

3.17. A disconnection-included system that provides energy on demand

Suppose C_1 consumers in an EPN that provides energy on demand are disconnected from the generators, where $0 < C_1 < C$. Let $C_2 = C - C_1$. This EPN can be described by the system of equations:

$$\begin{cases} \hat{q}_s = \Big(\sum_{g=1}^{G}(\Lambda_g \hat{p}_{g,s})\Big) \Big/ \Big(\gamma_s + \sum_{c_1=1}^{C_1}(\bar{q}_{c_1}\bar{\mu}_{c_1}\bar{r}_{c_1,s}\bar{\phi}_{s,c_1}\bar{p}_{s,c_1}) \\ \qquad + \sum_{c_2=1}^{C_2}(q_{c_2}\mu_{c_2}r_{c_2,s}\phi_{s,c_2}p_{s,c_2})\Big), \\ \bar{q}_{c_1} = \dfrac{\sum_{s=1}^{S}(\hat{q}_s\bar{q}_{c_1}\bar{\mu}_{c_1}\bar{r}_{c_1,s}\bar{\phi}_{s,c_1}\bar{p}_{s,c_1})}{\bar{\mu}_{c_1}}, \\ q_{c_2} = \dfrac{\sum_{s=1}^{S}(\hat{q}_s q_{c_2}\mu_{c_2}r_{c_2,s}\phi_{s,c_2}p_{s,c_2}) + \sum_{g=1}^{G}(\Lambda_g\tilde{p}_{g,c_2})}{\mu_{c_2}}, \end{cases} \tag{34}$$

with $\forall s \in S, c_1 \in C_1, c_2 \in C_2$, the request probabilities $\sum_{s=1}^{S}\bar{r}_{c_1,s} \leq 1$ and $\sum_{s=1}^{S} r_{c_2,s} \leq 1$, the allocation probabilities $\sum_{c=1}^{C}(\bar{p}_{s,c_1} + p_{s,c_2}) \leq 1$, and the allocation fractions $\bar{\phi}_{s,c_1}, \phi_{s,c_2}$ which may be greater than or smaller than one, depending on how a particular storage unit reacts to the requests from a specific consumer. In particular, we will expect that the total energy provision rate for any given storage unit will be bounded in some way:

$$\hat{q}_s\Big(\sum_{c_1=1}^{C_1}(\bar{q}_{c_1}\bar{\mu}_{c_1}\bar{r}_{c_1,s}\bar{\phi}_{s,c_1}\bar{p}_{s,c_1}) + \sum_{c_2=1}^{C_2}(q_{c_2}\mu_{c_2}r_{c_2,s}\phi_{s,c_2}p_{s,c_2})\Big) \leq \Gamma_s. \tag{35}$$

We want to maximize the amount of work done by the consumers per unit time in the EPN (34), or equivalently: given $\{\Lambda_g | \forall g \in G\}$, $\{\bar{\mu}_{c_1} | \forall c_1 \in C_1\}$, $\{\mu_{c_2} | \forall c_2 \in C_2\}$, $\{\gamma_s | \forall s \in S\}$ and $\{\Gamma_s | \forall s \in S\}$,

$$\text{maximize } K = \sum_{c_1=1}^{C_1}(\bar{\mu}_{c_1}\bar{q}_{c_1}) + \sum_{c_2=1}^{C_2}(\mu_{c_2}q_{c_2}), \tag{36}$$

by using $\{\hat{p}_{g,s} | \forall g \in G; \forall s \in S\}$, $\{\tilde{p}_{g,c_2} | \forall g \in G; \forall c_2 \in C_2\}$, $\{\bar{p}_{s,c_1} | \forall s \in S; \forall c_1 \in C_1\}$, $\{\bar{r}_{c_1,s} | \forall c_1 \in C_1; \forall s \in S\}$, $\{\bar{\phi}_{s,c_1} | \forall s \in S; \forall c_1 \in C_1\}$, $\{p_{s,c_2} | \forall s \in S; \forall c_2 \in C_2\}$, $\{r_{c_2,s} | \forall c_2 \in C_2; \forall s \in S\}$ and $\{\phi_{s,c_2} | \forall s \in S; \forall c_2 \in C_2\}$ with constraints $\hat{p}_{g,s}, \tilde{p}_{g,c_2}, \bar{p}_{s,c_1}, \bar{r}_{c_1,s}, \bar{\phi}_{s,c_1}, p_{s,c_2}, r_{c_2,s}, \phi_{s,c_2} \geq 0$, $\sum_{c_1=1}^{C_1}\bar{p}_{s,c_1} + \sum_{c_2=1}^{C_2} p_{s,c_2}) \leq 1$, $\sum_{c_2=1}^{C_2}\tilde{p}_{g,c_2} + \sum_{s=1}^{S}\hat{p}_{g,s} = 1$, $\sum_{s=1}^{S}\bar{r}_{c_1,s} \leq 1$, $\sum_{s=1}^{S} r_{c_2,s} \leq 1$ and inequality (35).

3.18. Optimal solutions

In the first case, the energy is so limited that $\sum_{g=1}^{G}\Lambda_g \leq \sum_{c_2=1}^{C_2}\mu_{c_2} = H_2$. In view of the analysis in Subsection 3.7.1, $K^* = \sum_{g=1}^{G}\Lambda_g$ is the optimal result for the maximization problem (23), where **the optimal solution is** $\hat{p}_{g,s} = 0$ **and** $\tilde{p}_{g,c_2} = \mu_{c_2}/H_2$, $\forall g \in G, s \in S, c_2 \in C_2$**. In addition,** $\bar{p}_{s,c_1}$**,** p_{s,c_2}**,** $\bar{r}_{c_1,s}$ **and** $r_{c_2,s}$ $\forall g \in G, s \in S, c_1 \in C_1, c_2 \in C_2$ **can be any non-negative value satisfying** $\sum_{c=1}^{C}(\bar{p}_{s,c_1} + p_{s,c_2}) \leq 1$**,** $\sum_{s=1}^{S}\bar{r}_{c_1,s} \leq 1$ **and** $\sum_{s=1}^{S} r_{c_2,s} \leq 1$**.**

In the second case, the energy is sufficient such that $\sum_{g=1}^{G}\Lambda_g > \sum_{c_2=1}^{C_2}\mu_{c_2} = H_2$. However, it can be complicate

Table 6: K^* and the best K found by using different algorithms for energy distribution of the EPN (29) with different G, S and C

(G,S,C)	K^*	Optimal	Random	CPSO-S	LNNLS	MLNNLS
$(2,2,2)$	**0.1500**	**0.1500**	0.1495	**0.1500**	0.1420	0.1486
$(10,5,16)$	**6.1525**	**6.1525**	5.8955	6.1525	6.1505	6.1515
$(10,10,30)$	**4.6191**	**4.6191**	4.0355	4.6171	4.6178	**4.6191**
$(20,10,60)$	**11.1612**	**11.1612**	10.4624	11.1599	11.1594	**11.1612**
$(100,100,300)$	**52.7882**	**52.7882**	N/A	N/A	52.7756	**52.7882**

to deduce the analytical optimal solution and result. Thus, it may be more practical to handle this case by designing optimization algorithms.

3.19. Cooperative particle swarm optimizer

Although the CPSO-S algorithm can be quite time-consuming, it performs well in solving the problems (23) and (30). Thus, we apply the CPSO-S algorithm to solving the problem (36).

First, let us define $\underline{\tilde{P}} \in R^{(3SC+GC_2+GS)\times 1}$ as a vector that consists of $\hat{p}_{g,s}$, $\tilde{p}_{g,c_2}$, $\bar{p}_{s,c_1}$, $\bar{r}_{c_1,s}$, $\bar{\phi}_{s,c_1}$, p_{s,c_2}, $r_{c_2,s}$ and ϕ_{s,c_2}, where

$$\begin{aligned}
&\underline{\tilde{P}}(\underline{h}_{s,c_1}) = \bar{p}_{s,c_1}, \underline{\tilde{P}}(h_{s,c_2}) = p_{s,c_2}, \underline{\tilde{P}}(\tilde{h}_{g,c_2}) = \tilde{p}_{g,c_2},\\
&\underline{\tilde{P}}(\hat{h}_{g,s}) = \hat{p}_{g,s}, \underline{\tilde{P}}(\bar{\underline{h}}_{c_1,s}) = \bar{r}_{c_1,s}, \underline{\tilde{P}}(\bar{h}_{c_2,s}) = r_{c_2,s},\\
&\underline{\tilde{P}}(\bar{\underline{h}}_{s,c_1}) = \bar{\phi}_{s,c_1}, \underline{\tilde{P}}(\bar{\underline{h}}_{s,c_2}) = \phi_{s,c_2},
\end{aligned}$$

with $\forall s \in S, c_1 \in C_1, c_2 \in C_2, g \in G$, $\underline{h}_{s,c_1} = (s-1)C_1 + c_1$, $h_{s,c_2} = SC_1 + (s-1)C_2 + c_2$,, $\tilde{h}_{g,c_2} = SC + (g-1)C_2 + c_2$, $\hat{h}_{g,s} = SC + GC_2 + (g-1)S + s$, $\bar{\underline{h}}_{c_1,s} = SC + GC_2 + GS + (s-1)C_1 + c_1$, $\bar{h}_{c_2,s} = SC + GC_2 + GS + SC_1 + (s-1)C_2 + c_2$, $\bar{\underline{h}}_{s,c_1} = 2SC + GC_2 + GS + (s-1)C_1 + c_1$ and $\bar{\underline{h}}_{s,c_2} = 2SC + GC_2 + GS + SC_1 + (s-1)C_2 + c_2$.

Let $M = 3SC + GC_2 + GS$ and N respectively denote the number of swarms and the number of particles in each swarm, where N is selected by the algorithm user. Let $\underline{\tilde{P}}_n(m)$ and $\bar{\underline{\tilde{P}}}_n(m)$ with $\forall m \in M, n \in N$ respectively denote the current state and best state of the nth particle in the mth swarm. Let $\underline{\tilde{P}}^{(\text{best})}$ denote the best $\underline{\tilde{P}}$ found so far. Assume that $\kappa_n \sim U(0,1)$ and $\hat{\kappa}_n \sim U(0,1)$ with $\forall n \in N$ are uniform random sequences in the range $(0,1)$, which will be updated after each iteration, then, the state-update formula can be

$$\underline{\tilde{P}}_n^{(l+1)}(m) = \underline{\tilde{P}}_n^{(l+1)}(m) + v_{n,m}^{(l+1)}, \tag{37}$$

where

$$\begin{aligned}
v_{n,m}^{(l+1)} =& \omega v_{n,m}^{(l)} + 2\kappa_n^{(l)}(\bar{\underline{\tilde{P}}}_n^{(l)}(m) - \underline{\tilde{P}}_n^{(l)}(m))\\
&+ 2\bar{\kappa}_n^{(l)}(\underline{\tilde{P}}^{(\text{best})}(m) - \underline{\tilde{P}}_n^{(l)}(m)),
\end{aligned}$$

ω is called the inertia weight and setup to vary from 1 to near 0 during the search process, l denotes the current number of iterations. The CPSO-S algorithm for solving problem (36) is similar to the one in [105] and thus omitted here.

3.20. A heuristic algorithm: MLNNLS-W

Hoping to find acceptable solutions for problem (36), a heuristic algorithm named as MLNNLS-W is designed in this subsection following the procedure in Subsection 3.15. Note that the design procedure is brief to avoid repetition.

First, let us define $\underline{P}_1, \underline{P}_2, \underline{P}_3 \in R^{(SC+GC_2+GS)\times 1}$ as vectors that consist of $\hat{p}_{g,s}$, $\tilde{p}_{g,c_2}$, $\bar{p}_{s,c_1}$, $\bar{r}_{c_1,s}$, $\bar{\phi}_{s,c_1}$, p_{s,c_2}, $r_{c_2,s}$ and ϕ_{s,c_2}, where

$$\begin{aligned}
&\underline{P}_1(\underline{h}_{s,c_1}) = \bar{p}_{s,c_1}, \underline{P}_1(h_{s,c_2}) = p_{s,c_2}, \underline{P}_2(\underline{h}_{s,c_1}) = \bar{r}_{c_1,s},\\
&\underline{P}_2(h_{s,c_2}) = r_{c_2,s}, \underline{P}_3(\underline{h}_{s,c_1}) = \bar{\phi}_{s,c_1}, \underline{P}_3(h_{s,c_2}) = \phi_{s,c_2},\\
&\underline{P}_1(\tilde{h}_{g,c_2}) = P_2(\tilde{h}_{g,c_2}) = P_3(\tilde{h}_{g,c_2}) = \tilde{p}_{g,c_2},\\
&\underline{P}_1(\hat{h}_{g,s}) = P_2(\hat{h}_{g,s}) = P_3(\hat{h}_{g,s}) = \hat{p}_{g,s},
\end{aligned}$$

with $\forall s \in S, c_1 \in C_1, c_2 \in C_2, g \in G$, $\underline{h}_{s,c_1} = (s-1)C_1 + c_1$, $h_{s,c_2} = SC_1 + (s-1)C_2 + c_2$,, $\tilde{h}_{g,c_2} = SC + (g-1)C_2 + c_2$, $\hat{h}_{g,s} = SC + GC_2 + (g-1)S + s$.

Then, the following three non-negative least-square problems are presented based on the system of equations (34):

$$\text{minimize } \underline{f}_1(\underline{P}_1) = \text{minimize } \frac{1}{2}\|\underline{A}_1\underline{P}_1 - \underline{b}_1\|_2^2,$$

$$\text{minimize } \underline{f}_2(\underline{P}_2) = \text{minimize } \frac{1}{2}\|\underline{A}_2\underline{P}_2 - \underline{b}_2\|_2^2,$$

$$\text{minimize } \underline{f}_3(\underline{P}_3) = \text{minimize } \frac{1}{2}\|\underline{A}_3\underline{P}_3 - \underline{b}_3\|_2^2,$$

where matrix $\underline{A}_1, \underline{A}_2, \underline{A}_3 \in R^{(S+C)\times(SC+GC_2+GS)}$ and vector

$\underline{b}_1, \underline{b}_2, \underline{b}_3 \in R^{(S+C)\times 1}$ are defined as

$$\underline{A}_1(\hat{\underline{h}}_s, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{r}_{c_1,s} \bar{\phi}_{s,c_1},$$
$$\underline{A}_1(\hat{\underline{h}}_s, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} r_{c_2,s} \phi_{s,c_2},$$
$$\underline{A}_2(\hat{\underline{h}}_s, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{\phi}_{s,c_1} \bar{p}_{s,c_1},$$
$$\underline{A}_2(\hat{\underline{h}}_s, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} \phi_{s,c_2} p_{s,c_2},$$
$$\underline{A}_3(\hat{\underline{h}}_s, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{r}_{c_1,s} \bar{p}_{s,c_1},$$
$$\underline{A}_3(\hat{\underline{h}}_s, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} r_{c_2,s} p_{s,c_2},$$
$$\underline{A}_1(\hat{\underline{h}}_s, \hat{h}_{g,s}) = \underline{A}_2(\hat{\underline{h}}_s, \hat{h}_{g,s}) = \underline{A}_3(\hat{\underline{h}}_s, \hat{h}_{g,s}) = -\Lambda_g,$$
$$\underline{A}_1(\hat{\underline{h}}_s, \text{otherwise}) = \underline{A}_2(\hat{\underline{h}}_s, \text{otherwise}),$$
$$= \underline{A}_3(\hat{\underline{h}}_s, \text{otherwise}) = 0,$$
$$\underline{A}_1(\bar{\underline{h}}_{c_1}, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{r}_{c_1,s} \bar{\phi}_{s,c_1},$$
$$\underline{A}_1(\underline{h}_{c_2}, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} r_{c_2,s} \phi_{s,c_2},$$
$$\underline{A}_2(\bar{\underline{h}}_{c_1}, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{\phi}_{s,c_1} \bar{p}_{s,c_1},$$
$$\underline{A}_2(\underline{h}_{c_2}, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} \phi_{s,c_2} p_{s,c_2},$$
$$\underline{A}_3(\bar{\underline{h}}_{c_1}, \underline{h}_{s,c_1}) = \hat{q}_s \bar{q}_{c_1} \bar{\mu}_{c_1} \bar{r}_{c_1,s} \bar{p}_{s,c_1},$$
$$\underline{A}_3(\underline{h}_{c_2}, h_{s,c_2}) = \hat{q}_s q_{c_2} \mu_{c_2} r_{c_2,s} p_{s,c_2},$$

$$\underline{A}_1(\underline{h}_{c_2}, \tilde{h}_{g,c_2}) = \underline{A}_2(\underline{h}_{c_2}, \tilde{h}_{g,c_2}) = \underline{A}_3(\underline{h}_{c_2}, \tilde{h}_{g,c_2}) = \Lambda_g,$$
$$\underline{A}_1(\bar{\underline{h}}_{c_1}, \text{otherwise}) = \underline{A}_2(\bar{\underline{h}}_{c_1}, \text{otherwise}),$$
$$= \underline{A}_3(\bar{\underline{h}}_{c_1}, \text{otherwise}) = 0,$$
$$\underline{A}_1(\underline{h}_{c_2}, \text{otherwise}) = \underline{A}_2(\underline{h}_{c_2}, \text{otherwise}),$$
$$= \underline{A}_3(\underline{h}_{c_2}, \text{otherwise}) = 0,$$
$$b_1(\hat{\underline{h}}_s) = b_2(\hat{\underline{h}}_s) = b_3(\hat{\underline{h}}_s) = -\hat{q}_s \gamma_s,$$
$$b_1(\bar{\underline{h}}_{c_1}) = b_2(\bar{\underline{h}}_{c_1}) = b_3(\bar{\underline{h}}_{c_1}) = \bar{q}_{c_1} \bar{\mu}_{c_1},$$
$$b_1(\underline{h}_{c_2}) = b_2(\underline{h}_{c_2}) = b_3(\underline{h}_{c_2}) = q_{c_2} \mu_{c_2},$$

with $\forall s \in S, c_1 \in C_1, c_2 \in C_2, g \in G$, $\hat{\underline{h}}_s = s$, $\bar{\underline{h}}_{c_1} = S + c_1$, $\underline{h}_{c_2} = S + C_1 + c_2$, $\underline{h}_{s,c_1} = (s-1)C_1 + c_1$, $h_{s,c_2} = SC_1(s-1)C_2 + c_2$, $\tilde{h}_{g,c_2} = SC + (g-1)C_2 + c_2$ and $\hat{h}_{g,s} = SC + GC_2 + (g-1)S + s$.

Then, the update formulas of $\underline{P}_1$, $\underline{P}_2$ and $\underline{P}_3$ can be

$$\underline{P}_1^{(l+1)} = \underline{P}_1^{(l)} - \eta \nabla \underline{f}(\underline{P}_1^{(l)}),$$
$$\underline{P}_2^{(l+1)} = \underline{P}_2^{(l)} - \eta \nabla \underline{f}(\underline{P}_2^{(l)}),$$
$$\underline{P}_3^{(l+1)} = \underline{P}_3^{(l)} - \eta \nabla \underline{f}(\underline{P}_3^{(l)}),$$

where $\nabla \underline{f}_1(\underline{P}_1^{(l)}) = \underline{A}_1^{\mathrm{T}}(\underline{A}_1 \underline{P}_1^{(l)}) - \underline{A}_1^{\mathrm{T}} \underline{b}_1$, $\nabla \underline{f}_2(\underline{P}_2^{(l)}) = \underline{A}_2^{\mathrm{T}}(\underline{A}_2 \underline{P}_2^{(l)}) - \underline{A}_2^{\mathrm{T}} \underline{b}_2$, $\nabla \underline{f}_3(\underline{P}_3^{(l)}) = \underline{A}_3^{\mathrm{T}}(\underline{A}_3 \underline{P}_3^{(l)}) - \underline{A}_3^{\mathrm{T}} \underline{b}_3$, l denotes the current number of iterations and $\eta > 0$ is the step size. In each iteration, $\bar{q}_{c_1}$ and q_{c_2} are substituted with 1 while $\hat{q}_s$ are substituted with given parameters y_s which are adjusted in the algorithm. After each iteration, we make $\underline{P}_1$, $\underline{P}_2$ and $\underline{P}_3$ satisfy the corresponding constraints with minimal adjustments.

Finally, the heuristic MLNNLS-W for solving the problem (36) can be designed by using the update formulas of $\underline{P}_1$, $\underline{P}_2$ and $\underline{P}_3$. Note that the procedure of the heuristic MLNNLS-W is similar to Algorithm 5 and thus omitted.

3.21. Numerical results

In the numerical experiments of this subsection, $\bar{\mu}_{c_1} = 1 + 9(c_1 - 1)/(C_1 - 1), \forall c_1 \in C_1$, $\mu_{c_2} = 1 + 9(c_2 - 1)/(C_2 - 1), \forall c_2 \in C_2$, $C_1 > 1$, $C_2 > 1$. In addition, γ_s and Γ_s with $\forall s \in S$ are randomly generated in the ranges of $[0, 0.5]$ and $[5, 10]$, respectively.

3.21.1. The energy is limited

In this case, $\sum_{g=1}^{G} \Lambda_g \leq \sum_{c_2=1}^{C_2} \mu_{c_2} = H_2$ and $\Lambda_g, \forall g \in G$ is randomly generated in the range of $[0, H_2/G]$.

Table 7 summaries the results by using the optimal solutions in Subsection 3.18, the randomly-generated solutions (20000 trials), the CPSO algorithm ($L = 50, N = 3$) and the heuristic MLNNLS-W ($L_1 = L_2 = L_3 = 5, Z_1 = Z_2 = Z_3 = 2$, $X = 4$ and 20 trials) to solve problem (36).

3.21.2. The energy is sufficient

In this case, $\sum_{g=1}^{G} \Lambda_g > H_2, \forall s \in S$ and $\Lambda_g, \forall g \in G$ is randomly generated in the range of $[H_2/G, H_2/G + S(H_1 + H_2)/G]$, where $H_1 = \sum_{c_1=1}^{C_1} \bar{\mu}_{c_1}$.

Table 8 summaries the results by using the optimal solutions in Subsection 3.7.2, the randomly-generated solutions (20000 trials), the CPSO algorithm ($L = 50, N = 3$) and the heuristic MLNNLS-W ($L_1 = L_2 = L_3 = 5, Z_1 = Z_2 = Z_3 = 2$, $X = 4$ and 20 trials) to solve problem (36).

4. Research Plan for the PhD

What should be done in the following two years concerns two main objectives: EPN for the smart grid and RNN for data analysis. The following two paragraphs list the sub-objectives/millstones needed to be achieved, where the fist paragraph is for the EPN for the smart grid while the second one is for the RNN for data analysis. For better illustration, Table 9 is the rough schedule of the research plan from 06/2015 to 10/2017. Note that the schedule is needed to be modified in the future according to the actual research process.

4.1. EPN for Smart Grid

First, the heuristic algorithms developed in this ESA report have the advantage of fast speed. But the disadvantage is that the algorithms are not flexible enough and the solutions found may not be near the optimal solutions. It means that improvements are needed for them. The strategy for the improvements may be to use the gradient-descent/back-propagation (BP) algorithms to fine tune the solutions found by the heuristic algorithms, where the BP-based fine-tuning operation is frequently used and proven to be effective in neural-network training [114].

Second, models of the EPN need to take more parameters in the real-world electrical energy network into consideration such as the energy transit nodes/units and nodes/units with limited capacity. Section 1.2 has listed several related points

Table 7: K^* and the best K found by using different algorithms for energy distribution of the EPN (34) with different G, S, C_1 and C_2 when the energy is limited

(G, S, C_1, C_2)	K^*	Optimal	Random	CPSO-S	MLNNLS
$(2, 2, 2, 2)$	**8.3565**	**8.3565**	8.5301	8.3544	**8.3565**
$(10, 5, 8, 8)$	**19.2945**	**19.2945**	18.4816	19.2945	**19.2945**
$(10, 10, 15, 15)$	**44.9075**	**44.9075**	41.6961	44.9031	**44.9075**
$(20, 10, 30, 30)$	**78.0313**	**78.0313**	71.936	78.0201	**78.0313**
$(100, 10, 150, 150)$	**372.5017**	**372.5017**	N/A	N/A	**372.5017**

Table 8: K^* and the best K found by using different algorithms for energy distribution of the EPN (34) with different G, S, C_1 and C_2 when the energy is more sufficient

(G, S, C_1, C_2)	K^*	Optimal	Random	CPSO-S	MLNNLS
$(2, 2, 2, 2)$	N/A	N/A	**22.0000**	11.0000	21.0000
$(10, 5, 8, 8)$	N/A	N/A	54.9998	44.0000	**70.1429**
$(10, 10, 15, 15)$	N/A	N/A	82.5035	82.5000	**132.2978**
$(20, 10, 30, 30)$	N/A	N/A	165.0000	165.0000	**193.7587**
$(100, 10, 150, 150)$	N/A	N/A	N/A	N/A	**825.0000**

about further work to be done on the EPN models. For example, the links (usually cables) connecting energy generation, storage and consumption centers should be regarded as nodes in the EPN models so as to describe the transit delay of energy or other effects caused during the transit process. In addition, further work can also be done to generalize the arrival manners of energy packets and other types of regular customers in the EPN models, which are Poisson arrivals in the original models.

Third, with enough mathematical analysis and numerical verifications conducted, a simulation platform for the EPN model equipped with the developed online optimization algorithms is needed to be built such that virtual experiments can be conducted and the EPN model and algorithms can be improved accordingly. Note that this may be the crucial step to the final practical application for the EPN model. Fourth, a physical platform based on the EPN model is hoped to be built subject to the laboratory equipments such that physical experiments can be conducted.

4.2. RNN for data analysis

First, based on [54] and [113], a neural-network model namely the fast training and testing neural network (FNN) whose advantages are fast training speed, low computational complexity and good prediction capability, is meant to be developed and tested on different tasks. The theoretical basis of the FNN comes from the research on the feedforward polynomial-based neural networks (PNN) [6, 31–35, 54] whose activation functions are constructed based on the product form of multiple polynomials. In addition, the efficacy of the PNN for multi-variable function approximation and pattern classification has been verified via numerous numerical experiments [6, 31–35, 54], meaning that the FNN should also achieve acceptable (or even excellent) performance in the related tasks. Developing efficient training methods and structure selection methods for the FNN is also required. About the structure selection methods, the strategy is to follow the developed methods in [6, 31–35, 54], e.g., the pruning algorithm in [32] and the cross-validation algorithm in [54]. The training methods for the FNN can be developed in three main approaches. The first approach is to simplify the structure of the FNN by setting the connecting weights between the input and hidden layer to zero, making the connecting weights between the hidden and output layers the only parameters to be adjusted. Then, the training of the FNN can be formulated into a standard least-square problem that can be directly solved by the formal pseudoinverse method [4–6, 31–34, 53, 54] as well as its variations [4] based on the size of training datasets. The second approach is to derive the gradient descent for the adjustable weights, which can include all connecting weights and biases in the FNN, and develop gradient-descent based training algorithms, e.g., the pure gradient-descent algorithm and its combinations with the Levenberg-Marquardt optimization procedure [39], quasi-Newton methods [38] and the RPROP learning algorithm [41]. The third approach is to apply computational methods that do not require the gradient information, such as the particle swarm optimization [42, 105, 108, 109] and the genetic algorithms [115].

Second, this FNN is meant to be mapped to the brain-like RNN, making this FNN suitable for highly parallel computation. It is worth pointing out the mapping is feasible since the work of Gelenbe in [113] has built the main theoretical basis.

Third, to fully utilize the advantage of highly parallel computation, it is needed to implement the RNN in different ways. One way is to utilize the parallel computation of the chip multiprocessor (CMP) and Graphics Processing Unit (GPU) to implement the RNN, which has high feasibility due to the rapid development of the CMP and GPU and

the improvement of the parallel computation supported languages e.g., the widely-used MATLAB language [14–16] and the domain-specific language for machine learning (OptiML) [16]. The second way is to simulate the spiking behaviors of neurons in the RNN. The feasibility of this way is still unknown. The third way is to implement the RNN model in hardware using special-purpose digital or analog components [10]. Via specially designed hardware, the high parallelism of the RNN model can be fully utilized and the speed in applying the RNN would achieve a large increase.

Fourth, a deep neural network (DNN) that has a deep structure (multi layers) is meant to be proposed for handling Big Data that is extremely large and with extremely-high dimension. The construction of the DNN may be done following the similar manner of the multilayer perceptron and the extension of the ELM for handling Big Data [5]. This DNN is also needed to be mapped to the brain-like RNN, which can also be done based on the work of Gelenbe in [113].

Fifth, based on the research work on implementing the RNN, it is needed to investigate the best choice to implement the RNN mappings of the FNN and DNN such that the best performance (related to the speed, robustness and energy efficiency) of the FNN and DNN can be achieved. In the author's understanding, the hardware implementation of the FNN and DNN may be the best choice to fully utilize the property of highly parallel computation of the RNN mappings, which, however, may also be the most difficult way among three above mentioned ways to implement the RNN.

Appendix. Derivation of (9) through (11)

From (7), we have

$$\nabla(h_{i,j}^+) = \sum_{n=1}^{N}\sum_{c=1}^{C}\Big(A(h_{n,c},h_{i,j}^+)\big(G(h_{i,c})-b(h_{n,c})\big)\Big), \quad (38)$$

where

$$\begin{aligned} G(h_{i,c}) = \sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}\Big(&A(h_{i,c},h_{\alpha,\beta}^+)\underline{w}(h_{\alpha,\beta}^+) \\ &+ A(h_{i,c},h_{\alpha,\beta}^-)\underline{w}(h_{\alpha,\beta}^-)\Big). \end{aligned} \quad (39)$$

We substitute (5) into (38) and wipe off the terms that equal zero. Then, we have

$$\begin{aligned} \nabla(h_{i,j}^+) =& \sum_{c=1}^{C}(A(h_{i,c},h_{i,j}^+)(G(h_{i,c})-b(h_{i,c}))) \quad (40)\\ &+\sum_{c=1}^{C}(A(h_{j,c},h_{i,j}^+)(G(h_{j,c})-b(h_{i,c}))), \forall i\neq j, \end{aligned}$$

and

$$\nabla(h_{i,j}^+) = \sum_{c=1}^{C}(A(h_{i,c},h_{i,j}^+)(G(h_{i,c})-b(h_{i,c}))), \forall i=j. \quad (41)$$

Further, equations (40) and (41) become

$$\begin{aligned} \nabla(h_{i,j}^+) =& \sum_{c=1}^{C}\left(\frac{q_{i,c}}{1-d_i}(G(h_{i,c})-b(h_{i,c}))\right) \\ &-\sum_{c=1}^{C}(q_{i,c}(G(h_{j,c})-b(h_{i,c}))), \\ =& \sum_{c=1}^{C}\Bigg(\frac{q_{i,c}}{1-d_i}(G(h_{i,c})-b(h_{i,c})) \\ &-q_{i,c}(G(h_{j,c})-b(h_{i,c}))\Bigg), \forall i\neq j, \end{aligned} \quad (42)$$

and

$$\nabla(h_{i,j}^+) = \sum_{c=1}^{C}\left(\left(\frac{q_{i,c}}{1-d_i}-y_{i,c}\right)(G(h_{i,c})-b(h_{i,c}))\right), \forall i=j. \quad (43)$$

Combining (42) and (43) yields equation (9). Note that equation (10) can be obtained via a similar derivation.

The following shows how to obtain (11) from (39). Evidently, $G = AW$ and therefore the $G(h_{i,c})$ should be the summation of $4N-2$ terms. Then, equation (39) becomes

$$\begin{aligned} G(h_{i,c}) =& \sum_{\alpha=1(\alpha\neq i)}^{N}\Big(A(h_{i,c},h_{i,\alpha}^+)\underline{w}(h_{i,\alpha}^+)\Big) \\ &+\sum_{\beta=1(\beta\neq i)}^{N}\Big(A(h_{i,c},h_{\beta,i}^+)\underline{w}(h_{\beta,i}^+)\Big) \\ &+A(h_{i,c},h_{i,i}^+)\underline{w}(h_{i,i}^+) \\ &+\sum_{\alpha=1(\alpha\neq i)}^{N}\Big(A(h_{i,c},h_{i,\alpha}^-)\underline{w}(h_{i,\alpha}^-)\Big) \\ &+\sum_{\beta=1(\beta\neq i)}^{N}\Big(A(h_{i,c},h_{\beta,i}^-)\underline{w}(h_{\beta,i}^-)\Big) \\ &+A(h_{i,c},h_{i,i}^-)\underline{w}(h_{i,i}^-). \end{aligned} \quad (44)$$

Substituting (5) into (44) yields

$$\begin{aligned} &G(h_{i,c}) \\ &=\sum_{\alpha=1(\alpha\neq i)}^{N}\left(\frac{q_{i,c}}{1-d_i}\underline{w}(h_{i,\alpha}^+)\right)+\sum_{\beta=1(\beta\neq i)}^{N}\Big(-q_{\beta,c}\underline{w}(h_{\beta,i}^+)\Big) \\ &+\left(\frac{q_{i,c}}{1-d_i}-q_{i,c}\right)\underline{w}(h_{i,i}^+)+\sum_{\alpha=1(\alpha\neq i)}^{N}\left(\frac{q_{i,c}}{1-d_i}\underline{w}(h_{i,\alpha}^-)\right) \\ &+\sum_{\beta=1(\beta\neq i)}^{N}\Big(q_{\beta,c}y_{i,c}\underline{w}(h_{\beta,i}^-)\Big)+\left(\frac{q_{i,c}}{1-d_i}+q_{i,c}^2\right)\underline{w}(h_{i,i}^-). \end{aligned} \quad (45)$$

Collecting the terms in (45) yields equation (11).

Table 9: Research Plan for the PhD from 06/2015 to 10/2017

Date: MM/YY	06/15	09/15	12/15	03/16	06/16	09/16	12/16	03/17	06/17	08/17	10/17
Online optimization algorithms	√	√	√								
More realistic EPN model		√	√	√	√						
EPN simulation platform				√	√	√	√				
Virtual experiment and improvement for EPN						√	√	√	√		
Physical platform for EPN								√	√	√	√
Fast neural network (FNN)	√	√	√								
FNN Mapping to RNN		√	√	√							
Deep neural network (DNN)				√	√	√	√	√			
DNN Mapping to RNN					√	√	√	√	√		
Parallel implement of RNN		√	√	√	√	√	√	√	√	√	
Hardware scheme for RNN mappings of FNN and DNN								√	√	√	√

References

[1] Georgiopoulos, M., Li, C., & Kocak, T. (2011). Learning in the feed-forward random neural network: A critical review. Performance Evaluation, 68(4), 361-384.

[2] Radhakrishnan, K., & Larijani, H. (2011). Evaluating perceived voice quality on packet networks using different random neural network architectures. Performance Evaluation, 68(4), 347-360.

[3] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE.

[4] Huang, G. B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 42(2), 513-529.

[5] Cambria, E., Huang, G. B., Kasun, L. L. C., Zhou, H., Vong, C. M., Lin, J., ... & Liu, J. (2013). Extreme learning machines [trends & controversies]. Intelligent Systems, IEEE, 28(6), 30-59.

[6] Zhang, Y., Yu, X., Xiao, L., Li, W., & Fan, Z. (2013). Weights and structure determination of artificial neuronets (chapter 5). Self-organization: theories and methods. Nova Science, New York.

[7] Deuflhard, P. (2011). Newton methods for nonlinear problems: affine invariance and adaptive algorithms (Vol. 35). Springer Science & Business Media.

[8] Metropolis, N., & Ulam, S. (1949). The monte carlo method. Journal of the American statistical association, 44(247), 335-341.

[9] Abdelbaki, H. M. (1999). Random neural network simulator for use with Matlab. Technical report. University of Central Florida.

[10] Abdelbaki, H., Gelenbe, E., & El-Khamy, S. E. (2000). Analog hardware implementation of the random neural network model. In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on (Vol. 4, pp. 197-201). IEEE.

[11] Cerkez, C., Aybay, I., & Halici, U. (1997, June). A digital neuron realization for the random neural network model. In Neural Networks, 1997., International Conference on (Vol. 2, pp. 1000-1004). IEEE.

[12] Kocak, T., Seeber, J., & Terzioglu, H. (2003). Design and implementation of a random neural network routing engine. Neural Networks, IEEE Transactions on, 14(5), 1128-1143.

[13] Gelenbe, E., Lent, R., & Xu, Z. (2001). Design and performance of cognitive packet networks. Performance Evaluation, 46(2), 155-176.

[14] Kirk, D. (2007, October). NVIDIA CUDA software and GPU parallel computing architecture. In ISMM (Vol. 7, pp. 103-104).

[15] Kong, J., Dimitrov, M., Yang, Y., Liyanage, J., Cao, L., Staples, J., ... & Zhou, H. (2010, March). Accelerating MATLAB image processing toolbox functions on GPUs. In Proceedings of the 3rd Workshop on General-Purpose

Computation on Graphics Processing Units (pp. 75-85). ACM.

[16] Sujeeth, A., Lee, H., Brown, K., Rompf, T., Chafi, H., Wu, M., ... & Olukotun, K. (2011). OptiML: an implicitly parallel domain-specific language for machine learning. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 609-616).

[17] Anthony, M., & Bartlett, P. L. (2009). Neural network learning: Theoretical foundations. cambridge university press.

[18] Zhu, X. (2005). Semi-supervised learning literature survey.

[19] Gelenbe, E., & Sungur, M. (1994). Random network learning and image compression. In Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on (Vol. 6, pp. 3996-3999). IEEE.

[20] Cramer, C., Gelenbe, E., & Gelenbe, P. (1998). Image and video compression. Potentials, IEEE, 17(1), 29-33.

[21] Cramer, C., Gelenbe, E., & Bakircioglu, H. (1996, August). Video compression with random neural networks. In Neural Networks for Identification, Control, Robotics, and Signal/Image Processing, 1996. Proceedings., International Workshop on (pp. 476-484). IEEE.

[22] Bakircioglu, H., & Gelenbe, E. (1998, April). Random neural network recognition of shaped objects in strong clutter. In Photonics West'98 Electronic Imaging (pp. 22-28). International Society for Optics and Photonics.

[23] Gelenbe, E., Kocak, T., & Wang, R. (2004). Wafer surface reconstruction from topdown scanning electron microscope images. Microelectronic Engineering, 75(2), 216-233.

[24] Oke, G., & Loukas, G. (2007). A denial of service detector based on maximum likelihood detection and the random neural network. The Computer Journal, 50(6), 717-727.

[25] Lu, R., & Shen, Y. (2006, January). Image segmentation based on random neural network model and gabor filters. In Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the (pp. 6464-6467). IEEE.

[26] Hussain, K. F., & Moussa, G. S. (2005, March). Laser intensity vehicle classification system based on random neural network. In Proceedings of the 43rd annual Southeast regional conference-Volume 1 (pp. 31-35). ACM.

[27] Teke, A., & Atalay, V. (2006). Texture classification and retrieval using the random neural network model. Computational Management Science, 3(3), 193-205.

[28] Gelenbe, E., Hussain, K. F., & Abdelbaki, H. (2000, April). Random neural network texture model. In Electronic Imaging (pp. 104-111). International Society for Optics and Photonics.

[29] Gelenbe, E., & Fourneau, J. M. (1999). Random neural networks with multiple classes of signals. Neural computation, 11(4), 953-963.

[30] Gelenbe, E., & Hussain, K. F. (2002). Learning in the multiple class random neural network. Neural Networks, IEEE Transactions on, 13(6), 1257-1267.

[31] Zhang, Y., Lao, W., Yin, Y., Xiao, L., & Chen, J. (2012, August). Weights and structure determination of pruning-while-growing type for 3-input power-activation feed-forward neuronet. In Automation and Logistics (ICAL), 2012 IEEE International Conference on (pp. 212-217). IEEE.

[32] Zhang, Y., Yin, Y., Yu, X., Guo, D., & Xiao, L. (2012, July). Pruning-included weights and structure determination of 2-input neuronet using Chebyshev polynomials of Class 1. In Intelligent Control and Automation (WCICA), 2012 10th World Congress on (pp. 700-705). IEEE.

[33] Zhang, Y. N., Liu, J. R., Yin, Y. H., & Xiao, L. (2012). Weights and Structure Determination of Multi-input Laguerre-orthogonal-polynomial Feed-forward Neural Network. Computer Science, 39(12).

[34] Zhang, Y., Chen, J., Guo, D., Yin, Y., & Lao, W. (2012, May). Growing-type weights and structure determination of 2-input Legendre orthogonal polynomial neuronet. In Industrial Electronics (ISIE), 2012 IEEE International Symposium on (pp. 852-857). IEEE.

[35] Zhang, Y., & Tan, N. (2010). Weights direct determination of feedforward neural networks without iterative BP-training. Intelligent Soft Computation and Evolving Data Mining, IGI Globa, Hershey, 197-225.

[36] Wilson, D. R., & Martinez, T. R. (2001). The need for small learning rates on large problems. In Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on (Vol. 1, pp. 115-119). IEEE.

[37] Yu, X. H., & Chen, G. A. (1994). On the local minima free condition of backpropagation learning. IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council, 6(5), 1300-1303.

[38] Likas, A., & Stafylopatis, A. (2000). Training the random neural network using quasi-Newton methods. European Journal of Operational Research, 126(2), 331-339.

[39] Basterrech, S., Mohammed, S., Rubino, G., & Soliman, M. (2009). LevenbergMarquardt Training Algorithms for Random Neural Networks. The computer journal, bxp101.

[40] Hubert, C. (1993, October). Pattern completion with the random neural network using the RPROP learning algorithm. In Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on (pp. 613-617). IEEE.

[41] Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Neural Networks, 1993., IEEE International Conference on (pp. 586-591). IEEE.

[42] Qin, Z., Yu, F., Shi, Z., & Wang, Y. (2006). Adaptive inertia weight particle swarm optimization. In Artificial Intelligence and Soft ComputingICAISC 2006 (pp. 450-459). Springer Berlin Heidelberg.

[43] Storn, R., & Price, K. (1997). Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341-359.

[44] Gelenbe, E. (1989). Random neural networks with negative and positive signals and product form solution. Neural computation, 1(4), 502-510.

[45] Timotheou, S. (2010). The random neural network: a survey. The computer journal, 53(3), 251-267.

[46] Bakircioglu, H., & Koak, T. (2000). Survey of random neural network applications. European journal of operational research, 126(2), 319-330.

[47] Gelenbe, E. (1993). Learning in the recurrent random neural network. Neural Computation, 5(1), 154-164.

[48] Timotheou, S. (2009). A novel weight initialization method for the random neural network. Neurocomputing, 73(1), 160-168.

[49] Timotheou, S. (2008). Nonnegative least squares learning for the random neural network. In Artificial Neural Networks-ICANN 2008 (pp. 195-204). Springer Berlin Heidelberg.

[50] Gelenbe, E., & Hussain, K. F. (2002). Learning in the multiple class random neural network. Neural Networks, IEEE Transactions on, 13(6), 1257-1267.

[51] Gelenbe, E., & Sungur, M. (1994). Random network learning and image compression. In Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on (Vol. 6, pp. 3996-3999). IEEE.

[52] Gelenbe, E., Hussain, K. F., & Abdelbaki, H. (2000, April). Random neural network texture model. In Electronic Imaging (pp. 104-111). International Society for Optics and Photonics.

[53] Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. Neurocomputing, 70(1), 489-501.

[54] Zhang, Y., Yin, Y., Guo, D., Yu, X., & Xiao, L. (2014). Cross-validation based weights and structure determination of Chebyshev-polynomial neural networks for pattern classification. Pattern Recognition, 47(10), 3414-3428.

[55] Zhang, Y., Yang, Y., Cai, B., & Guo, D. (2012). Zhang neural network and its application to Newton iteration for matrix square root estimation. Neural Computing and Applications, 21(3), 453-460.

[56] Diniz-Ehrhardt, M. A., Martnez, J. M., & Raydn, M. (2008). A derivative-free nonmonotone line-search technique for unconstrained optimization. Journal of computational and applied mathematics, 219(2), 383-397.

[57] Palensky, P., Kupzog, F. (2013). Smart grids. Annual Review of Environment and Resources, 38, 201-226.

[58] Hledik, R. (2009). How green is the smart grid?. The Electricity Journal, 22(3), 29-41.

[59] Gungor, V. C., Sahin, D., Kocak, T., Ergut, S., Buccella, C., Cecati, C., & Hancke, G. P. (2013). A survey on smart grid potential applications and communication requirements. Industrial Informatics, IEEE Transactions on, 9(1), 28-42.

[60] Fadaeenejad, M., Saberian, A. M., Fadaee, M., Radzi, M. A. M., Hizam, H., & AbKadir, M. Z. A. (2014). The present and future of smart power grid in developing countries. Renewable and Sustainable Energy Reviews, 29, 828-834.

[61] Outlook IWE. Nominal GDP comparison of China, Germany, France, Japan and USA. 2012.

[62] Hashmi, M. (2011). Survey of smart grids concepts worldwide. VTT Technical Research Centre of Finland.

[63] CHALLENGES, I. (2008). Initial review of methods for cascading failure analysis in electric power transmission systems.

[64] Baumeister, T. (2010). Literature review on smart grid cyber security. Collaborative Software Development Laboratory at the University of Hawaii.

[65] Zareipour, H., Bhattacharya, K., & Canizares, C. A. (2004, August). Distributed generation: current status and challenges. In Annual North American Power Symposium (NAPS) (pp. 1-8).

[66] Molderink, A., Bakker, V., Bosman, M. G., Hurink, J. L., & Smit, G. J. (2010). Management and control of domestic smart grid technology. Smart grid, IEEE transactions on, 1(2), 109-119.

[67] Fraser, P. (2002). Distributed generation in liberalised electricity markets. In International symposium on distributed generation: power system and market aspects (pp. 1G-12).

[68] Caldon, R., Patria, A. R., & Turri, R. (2004). Optimal control of a distribution system with a virtual power plant. Bulk Power System Dynamics and Control, Cortina. dAmpezzo, Italy.

[69] Lombardi, P., Powalko, M., & Rudion, K. (2009, July). Optimal operation of a virtual power plant. In Power & Energy Society General Meeting, 2009. PES'09. IEEE (pp. 1-6). IEEE.

[70] Ernst, B., Rohrig, K., & Jursa, R. (2002, December). Online-monitoring and prediction of wind power in German transmission system operation centres. In Proceedings of the First IEA Joint Action Symposium on Wind Forecasting Techniques, Norrkping, Sweden (pp. 125-145).

[71] Fang, X., Yang, D., & Xue, G. (2011, December). Online strategizing distributed renewable energy resource access in islanded microgrids. In Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE (pp. 1-6). IEEE.

[72] Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995, October). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on (pp. 322-331). IEEE.

[73] Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. SIAM Journal on Computing, 32(1), 48-77.

[74] Lasseter, R. H., & Paigi, P. (2004, June). Microgrid: a conceptual solution. In Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual (Vol. 6, pp. 4285-4290). IEEE.

[75] Gelenbe, E., & Mitrani, I. (2010). Analysis and synthesis of computer systems (Vol. 4). World Scientific.

[76] Gelenbe, E., & Pujolle, G. (1998). Introduction to networks of queues. John Wiley Ltd, New York and Chichester.

[77] Gelenbe, E. (1991). Product-form queueing networks with negative and positive customers. Journal of applied probability, 656-663.

[78] Gelenbe, E. (2011). Introduction to the special issue on G-Networks and the random neural network. Performance Evaluation, 68(4), 307-308.

[79] Gelenbe, E. (1993). G-networks with triggered customer movement. Journal of Applied Probability, 742-748.

[80] Gelenbe, E. (1993). G-networks with signals and batch removal. Probability in the Engineering and Informational Sciences, 7(03), 335-342.

[81] Fourneau, J. M., Gelenbe, E., & Suros, R. (1996). G-networks with multiple classes of negative and positive customers. Theoretical Computer Science, 155(1), 141-156.

[82] Gelenbe, E., & Labed, A. (1998). G-networks with multiple classes of signals and positive customers. European Journal of Operational Research, 108(2), 293-305.

[83] Gelenbe, E., & Fourneau, J. M. (2002). G-networks with resets. Performance Evaluation, 49(1), 179-191.

[84] Fourneau, J. M., & Gelenbe, E. (2004). Flow equivalence and stochastic equivalence in G-networks. Computational Management Science, 1(2), 179-192.

[85] Morfopoulou, C. (2011). Network routing control with G-networks. Performance Evaluation, 68(4), 320-329.

[86] Gelenbe, E., & Morfopoulou, C. (2011). A framework for energy-aware routing in packet networks. The Computer Journal, 54(6), 850-859.

[87] Gelenbe, E., & Morfopoulou, C. (2012). Gradient optimisation for network power consumption. In Green Communications and Networking (pp. 125-134). Springer Berlin Heidelberg.

[88] Gelenbe, E., & Mahmoodi, T. (2011). Energy-aware routing in the cognitive packet network. ENERGY, 7-12.

[89] McDaniel, P., & McLaughlin, S. (2009). Security and privacy challenges in the smart grid. IEEE Security & Privacy, (3), 75-77.

[90] Farhangi, H. (2010). The path of the smart grid. Power and Energy Magazine, IEEE, 8(1), 18-28.

[91] Gelenbe, E. (2012, March). Energy packet networks: smart electricity storage to meet surges in demand. In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (pp. 1-7). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[92] Amin, S. M., & Wollenberg, B. F. (2005). Toward a smart grid: power delivery for the 21st century. Power and Energy Magazine, IEEE, 3(5), 34-41.

[93] Fang, X., Misra, S., Xue, G., & Yang, D. (2012). Smart grid-The new and improved power grid: A survey. Communications Surveys & Tutorials, IEEE, 14(4), 944-980.

[94] Takuno, T., Koyama, M., & Hikihara, T. (2010, October). In-home power distribution systems by circuit switching and power packet dispatching. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on (pp. 427-430). IEEE.

[95] Tashiro, K., Takahashi, R., & Hikihara, T. (2012, November). Feasibility of power packet dispatching at in-home DC distribution network. In Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on (pp. 401-405). IEEE.

[96] Bamberger, Y., Baptista, J., Belmans, R., Buchholz, B. M., Chebbo, M., Doblado, J. L. D. V., ... & Wasiluk-Hassa, M. (2006). Vision and Strategy for Europes Electricity Networks of the Future: European Technology PlatformSmartGrids. Office for Official Publications of the European Communities.

[97] Gelenbe, E. (2012, April). Energy packet networks: adaptive energy management for the cloud. In Proceedings of the 2nd International Workshop on Cloud Computing Platforms (p. 1). ACM.

[98] Guan, X., Xu, Z., & Jia, Q. S. (2010). Energy-efficient buildings facilitated by microgrid. Smart Grid, IEEE Transactions on, 1(3), 243-252.

[99] Molderink, A., Bakker, V., Bosman, M. G., Hurink, J. L., & Smit, G. J. (2010, January). A three-step methodology to improve domestic energy efficiency. In Innovative Smart Grid Technologies (ISGT), 2010 (pp. 1-8). IEEE.

[100] Bakker, V., Bosman, M. G. C., Molderink, A., Hurink, J. L., & Smit, G. J. M. (2010, October). Demand side load management using a three step optimization methodology. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on (pp. 431-436). IEEE.

[101] Aquino-Lugo, A., & Overbye, T. (2010, January). Agent technologies for control applications in the power grid. In System Sciences (HICSS), 2010 43rd Hawaii International Conference on (pp. 1-10). IEEE.

[102] Roozbehani, M., Dahleh, M., & Mitter, S. (2010, October). Dynamic pricing and stabilization of supply and demand in modern electric power grids. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on (pp. 543-548). IEEE.

[103] Hatami, S., & Pedram, M. (2010, October). Minimizing the electricity bill of cooperative users under a quasi-dynamic pricing model. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on (pp. 421-426). IEEE.

[104] Bu, S., Yu, F. R., & Liu, P. X. (2011, April). Stochastic unit commitment in smart grid communications. In Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on (pp. 307-312). IEEE.

[105] Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. Evolutionary Computation, IEEE Transactions on, 8(3), 225-239.

[106] Mohsenian-Rad, A. H., & Leon-Garcia, A. (2010). Optimal residential load control with price prediction in real-time electricity pricing environments. Smart Grid, IEEE Transactions on, 1(2), 120-133.

[107] Anderson, R. N., Boulanger, A., Powell, W. B., & Scott, W. (2011). Adaptive stochastic control for the smart grid. Proceedings of the IEEE, 99(6), 1098-1115.

[108] Van den Bergh, F., & Engelbrecht, A. P. (2000). Cooperative learning in neural networks using particle swarm optimizers. South African Computer Journal, (26), p-84.

[109] Mitra, J., Patra, S. B., & Ranade, S. J. (2006, June). Reliability stipulated microgrid architecture using particle swarm optimization. In Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on (pp. 1-7). IEEE.

[110] Gelenbe, E., & Ceran, E. T. (2015, April). Central or distributed energy storage for processors with energy harvesting. In Sustainable Internet and ICT for Sustainability (SustainIT), 2015 (pp. 1-3). IEEE.

[111] Lin, C. B. (2007). Projected gradient methods for nonnegative matrix factorization. Neural computation, 19(10), 2756-2779.

[112] Lent, R. (2011, November). Evaluating the performance and power consumption of systems with virtual machines. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 778-783). IEEE.

[113] Gelenbe, E., Mao, Z. H., & Li, Y. D. (1999). Function approximation with spiked random networks. Neural Networks, IEEE Transactions on, 10(1), 3-9.

[114] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

[115] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on, 6(2), 182-197.