Lin and Ye *EURASIP Journal on Wireless Communications and Networking* (2018) 2018:120
https://doi.org/10.1186/s13638-018-1145-y

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A bat algorithm for SDN network scheduling

Rongheng Lin* and Zezhou Ye

Abstract

With the development of SDN, end-to-end network reservation becomes a reality. Resource reservation can become an online service for network users, and the network faces new challenges on how to allocate users' resource dynamically. To solve the problem, this paper proposed a bat-based algorithm for SDN network scheduling along with a network request model. The network resource request model is for characterizing users' simultaneous network resource request in SDN. Based on the model, this paper transforms the resource reservation problem into a multiple-knapsack problem and proposes a bat algorithm to optimize the solution. Experiments show that the proposed algorithm is better than greedy and dynamic programming algorithm. The major contribution of this paper is to model the SDN users' resource requests and apply bat algorithm for the solution.

Keywords: SDN, Network scheduling, Bat algorithm

1 Introduction

The appearance of software-defined networks (software-defined network, abbreviated as SDN [1]) makes it possible to control the scheduling of network packets more precisely. SDN technology separates the control plane and data plane of the network, and it provides a new solution for developing new network applications and future Internet technologies [2]. In the traditional seven-layer network architecture, the routing control algorithm located in the network layer lacks the ability of SDN network controller to collect and understand the dynamic information of entire network. Therefore, the traditional routing control algorithm, which is considered as the method of packet scheduling, does not completely release the capability of the network in the SDN architecture. Therefore, it is necessary to propose a new scheduling algorithm to obtain better performance.

The disadvantage of traditional scheduling algorithms in the SDN environment is that ability of the SDN network to collect information and the features of separation between control logic and data-forwarding are not well utilized. The application can inform the SDN controller about the application type and the characteristic of the related data stream in a flat SDN environment. The controller generates data forwarding rules based on this information. In order to improve the network resource utilization in SDN environment, considering the ability of SDN network to gather information as well as learning from the idea of label distribution and resource reservation, more efficient scheduling algorithms are obtained to improve network utilization.

Learning from the idea of distributing data by a label in MPLS [3] and resource reservation in RSVP [4], taking advantage of TOS service type field is not commonly used in IP packets. The SDN controller may reserve resources for the incoming request according to the data flow characteristics of different application types after negotiating with the application server.

2 Related work

SDN has become one of the most important architectures for the management of largescale complex networks by decoupling the control plane from data plane [5]. SDN introduces new possibilities for network management and configuration methods. In this article, authors identify problems with the current state-of-the-art network configuration and management mechanisms and propose mechanisms to improve various aspects of network management [6]. Thus, the network routers/switches just simply forward packets by following the flow table rules set by the control plane. SDN depart from traditional network architectures by explicitly allowing third-party software access to the network's control plane. Thus, SDN protocols give network operators the ability to innovate by authoring or buying network controller software independent of the hardware, such as OpenFlow. However, this split design can make

* Correspondence: rhlin@bupt.edu.cn
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

planning and designing large SDNs even more challenging than traditional networks [7]. Currently, OpenFlow is the most popular SDN protocol and owns a set of designed specifications. SDN has appeared as an efficient network technology, which is capable of supporting the dynamic nature of future network functions and intelligent applications while lowering operating costs through simplified hardware, software, and management [8].

Much research has been done around SDN resource allocation. Sharkh and Jammal consider computational resources and network resources to reflect practical demands accurately. Calls for the support of green clouds are gaining momentum. With that in mind, resource allocation algorithms are dedicated to accomplishing the task of scheduling virtual machines on the servers residing in data centers and consequently scheduling network resources while complying with the problem constraints. Several external and internal factors that affect the performance of resource allocation models are introduced in the article [9]. In [10], Bosshart P and Gibb G propose the RMT (reconfigurable match tables) model, a new RISC-inspired pipelined architecture for switching chips. In addition, they identify the essential minimal set of action primitives to specify how headers are processed in hardware. RMT allows changing the forwarding plane in the field without modifying hardware. As in OpenFlow, the programmer can specify multiple match tables of arbitrary width and depth, with each table configurable for matching on arbitrary fields. However, RMT allows the programmer to modify all header fields much more comprehensively than in OpenFlow. Li L E argues that software-defined networking (SDN) can simplify the design and management of cellular data networks while enabling new services. However, in order to support many subscribers, real-time adaptation introduces new scalability challenges that future SDN architectures should address in the paper [11].

Yang proposed a new metaheuristic method, the bat algorithm, based on the echolocation behavior of bats. He also intends to combine the advantages of existing algorithms into the new bat algorithm. After a detailed formulation and explanation of its implementation, and then compare the proposed algorithm with other existing algorithms, including genetic algorithms and particle swarm optimization algorithms. Since then, the bat algorithm has also proposed many variations.

The bat algorithm has been applied for SDN network scheduling in previous works, but they do have limitations of heuristic intelligence bat algorithm in solving discrete problems under SDN environment. Bat algorithm is hybridized with differential evolution strategies. Besides showing very promising results of the standard benchmark functions, this hybridization also significantly improves the original bat algorithm. A novel algorithm that can be applied in SDN for solving the numerical optimization problem is proposed based on the framework of the original bat algorithm, which is called evolved bat algorithm (EBA). EBA is a new method in the branch of swarm intelligence for solving SDN network-scheduling problems [12]. By reanalyzing the behavior of bats and considering the general characteristics of whole species of bat, the author redefines the corresponding operation to the bats' behaviors. In paper [13], Fister presented a new swarm intelligence algorithm based on the bat algorithm. In the present study, Amir have introduced chaos into BA to increase its global search mobility for SDN network-scheduling optimization [14]. Detailed studies have been carried out on benchmark problems with different chaotic maps. Four different variants of chaotic BA are introduced, and 13 different chaotic maps are utilized for validating each of these four variants. Sathya's proposed work aims to detect the attacks on SDN environment by using a heuristic algorithm inspired by the echolocation behavior of bats. Detecting anomalies in SDN environment will be more manageable and efficient [15].

3 SDN network data request model

Data on the network can be regarded as a stream of data packets, which are composed of protocol header and data payload. Analysis of single data packets is unable to master the basic requirements of application requests. Such as real-time video and online games, which have basic requirements for real-time data throughput. When the basic requirements are not met, and the arriving packet is in an unavailable state, it is better to discard the packet directly. The development of QoS [16] helps to solve this problem under the condition of insufficient information in traditional networks to some extent. Under the environment of SDN network, we are able to collect more features of the request information by using resource reservation to obtain better performance.

3.1 Request model

In this paper, the network requests are considered from three aspects, including the amount of data, time, and the coupling relationship between requests. The amount of data can be represented by occupied bandwidth and occupied time in more detailed requirements. Time reflects the length of the request and may have a connection with the earliest start time and the latest termination time under more detailed requirements. The coupling relationship between requests has different forms of performance, such as two requests separated by a fixed time, or the first request with a small amount of data used for negotiation and the second request with large amount data used for transmission.

As shown in Fig. 1, the optimization goal of SDN network data request model is to place the request into an

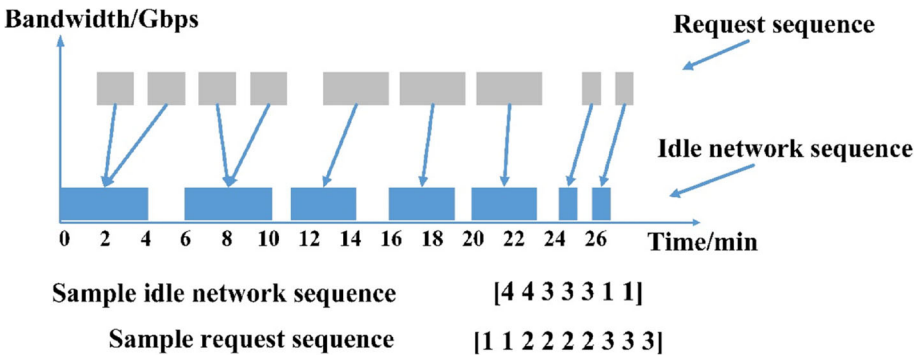

Fig. 1 SDN network data request model. As shown in Fig. 1, the optimization goal of SDN network data request model is to place the request into an idle network, making the total length of request that have been put into the idle network always the longest. The coupling relationship between requests depends on the specific coupling relationship. The network status can be simplified as (T_1, T_2, …, T_m), where T_i is the idle duration of the ith segment. In fact, you can add an extra idle duration of the ($m + 1$)th segment T_{m+1}, but we can omit T_{m+1} due to the last idle time T_{m+1} is infinitely long, considering putting the request in the idle network as front as possible in the sequence. The set of network requests is reduced to (t_1, t_2, …, t_n) and t_i is the time occupied by the ith request. In Fig. 1, the total length of idle network for the first group of networks is 19 and the total length of the request is 19, so the requests are all placed in the idle network. However, the idle network cannot all be filled in other situations that will be discussed later

idle network, making the total length of request that has been put into the idle network always the longest. The network status can be simplified as (T_1, T_2, …, T_m), where T_i is the idle duration of the ith segment. In fact, you can add an extra idle duration of the ($m + 1$)th segment T_{m+1}, but we can omit T_{m+1} due to the last idle time T_{m+1} is infinitely long, considering putting the request in the idle network as front as possible in the sequence. The set of network requests is reduced to (t_1, t_2, …, t_n), and t_i is the time occupied by the ith request.

In Fig. 1, the total length of the idle network for the first group is 19, and the total length of the request is 19, so the requests are all placed in the idle network. However, the idle network cannot all be filled in other situations that will be discussed later. Thus, this paper models network requests as follows:

1. Occupied bandwidth B.
2. Occupied time T.
3. The earliest start time t_1
4. The latest termination time t_2.
5. The coupling relationship between the requests R.

Therefore, the network request is expressed as a 5-tuple request: $\{B,T,t_1,t_2,R\}$. In addition, natural networks can be modeled as a ternary variable list, which is expressed as $\{B,T_1,T_2\}$.

We can transform the model. Regarding bandwidth, a request may require a bandwidth of B, and the total bandwidth of the line is B_t. We discretize the continuous bandwidth and simplify the bandwidth state to a 0–1 model. One request that consumes more bandwidth than others can be considered as multiple simultaneous requests, which consumes less bandwidth. In addition, a network with more bandwidth can be made up of multiple 0–1 bandwidth networks. We replace the original request with multiple coupled requests without having to discuss the bandwidth separately.

The situation is much complicated when algorithms are proposed directly under the model mentioned above. We can simplify the model first, and then gradually modify the algorithm that is proposed in a simplified model, finally apply the algorithm in the complex model.

3.2 Multi-knapsack problem mapping

For a single type 0–1 request, it is easy to map it into multi-knapsack problems, taking the length of occupation time into account only. The simplest model is mapped into the multi-knapsack problem first and then extends into a complex case.

Knapsack problem [17] has been proposed with a lot of mutation as an optimization model a long time ago, one of which is the multi-knapsack problem. The multi-knapsack problem can be described as given n objects of size w_1, w_2, $\cdots$, w_n and m knapsacks of size c_1, c_2, $\cdots$, c_n (where w_i and c_j are both positive integers). It is required to find m disjoint subsets of w such that m packs are filled up as much as possible, which means all the objects in the packs have the largest sum of values. The mathematical model is described as follows:

$$\max\left(\sum_{j=1}^{m}\sum_{i=1}^{n} W_i X_{ij}\right)$$

Moreover, meet these conditions below:

$$\begin{cases} \sum_{j=1}^{m} X_{ij} \le 1, & \forall i \in \{1, 2, \ldots, n\} \\ \sum_{i=1}^{n} W_i X_{ij} \le C_j, & \forall j \in \{1, 2, \ldots, m\} \\ X_{ij} \in \{0, 1\} \end{cases}$$

$X_{ij} = 1$ means that the object i is put into the knapsack j, on the contrary $X_{ij} = 0$ means object i is not put into the knapsack.

Therefore, the request sequence may be regarded as the item sequence, the request duration is similar to the size of the item, the idle network of the 0–1 bandwidth model is similar to the knapsack, and the idle time can be regarded as the size of a knapsack. The optimization goal is to place the request into an idle network, making the total length of request that has been put into the idle network always the longest. In practical applications, the remaining requests can be placed after the last occupied time slice, because the duration of the request placed in the knapsack is the longest, and the total duration of remaining requests is minimized as well as the request sequence is processed in the shortest amount of time.

Further, we can add other conditions to the multi-knapsack problem to map models that are more complex. For the earliest start time t_1 and the latest termination time t_2 in request: $\{B,T,t_1,t_2,R\}$. For knapsack that is considered as idle network, we record the idle starting and ending time T_1, T_2, regarding knapsack as $\{W,T_1,T_2\}$. A request that is put into a backpack needs to meet the necessary conditions: $t_1 + T < T_2$ && $t_2 - T > T_1$. It should be noted that when a request with t_1 or t_2 is placed in (T_1, T_2), subsequent requests would be affected by this restriction because the request can only exist in a specific space in the knapsack. Considering splitting the knapsacks, we will use the heuristic algorithm to specify the placement method. We describe the status of the knapsack depending on our algorithm rules.

It is necessary to modify the knapsack model accordingly. Taking two requests at a fixed time interval as an example, such two requests can be thought as a request that takes up more time including time of two requests and time of interval. The middle of the interval naturally forms an idle network, and then such two requests can be regarded as an item in the knapsack. Further, the knapsack can also be placed in an allocated period, which can be considered as a request with a starting and ending time.

4 Model solution

When the model is established and mapped into to multi-knapsack problem, the problem can be solved. From the point of a deterministic algorithm, enumeration can get the global optimal solution. Using the pruning technology can reduce the search space to a certain extent, and thus improve the efficiency of the algorithm. However, the problem of multiple knapsacks has been proven to be an *NP* problem [18]. The deterministic algorithm cannot be accepted in time.

Solving the problem from the perspective of an approximate algorithm, the simple idea is to use the greedy algorithm. Although with high time efficiency, the local optimal solution is not stable. Various heuristics algorithms [19] have been proposed to solve optimization problems, including ant colony algorithm [20], genetic algorithm [21], and bat algorithm [22]. They are also applied in knapsack problem. Among them, the bat algorithm is a heuristic intelligent algorithm that has been proposed in recent years and performs well in solving various problems. It can overcome the shortcomings of local optimal solution and instability of greedy algorithm, outperforming other algorithms like ant colony algorithm and genetic algorithm. This paper attempts to use the bat algorithm to solve the model by mapping the location of the bat to the knapsack where the data was requested.

4.1 Bat algorithm

The bat algorithm is a heuristic intelligent algorithm proposed by Wang of the University of Cambridge in 2010 that simulates the echolocation principle used in the bat predation process. While elaborating the basic idea of bat algorithm, Yang proposed the basic assumptions of bat algorithm:

1. All bat particles use their echolocation to perceive the distance to the target while discerning differences between the target and the background obstacle in a mysterious way at the same time.
2. The bat's position is x_i, flying at any speed v_i, searching for a target at a fixed frequency $f_{\min}$, variable wavelength l, and loudness A_i. They can determine the distance between themselves and their prey, and automatically adjust the wavelength (or frequency) of the pulse while adjusting the frequency $r \in [0, 1]$ of the pulse as it approaches the target.
3. Loudness changes in many ways, assuming it varies from the maximum value (positive) A_0 to a fixed minimum value $A_{\min}$.
4. Three-dimensional topography and time delays are ignored here, although this is likely to be one of the nice features of multidimensional calculations. However, it is not known how to use it so far, and it greatly increases the amount of computation in multidimensional calculations at the same time.

In this paper, we use the algorithm for a simplified model of the data request in SDN network, taking the occupied time into account under circumstances of 0–1 bandwidth. The network status can be simplified as $(T_1, T_2, \ldots, T_m)$, where T_i is the idle duration of the ith segment. In fact, you can add an extra idle duration of the $(m+1)$th segment T_{m+1}, but we can omit T_{m+1} due to the last idle time T_{m+1} is infinitely long, considering putting the request in the idle network as front as possible in the sequence. The set of network requests is reduced to $(t_1, t_2, \ldots, t_n)$, and t_i is the time occupied by the ith request.

Mapping the location of the bat into the knapsack, the key issue is to update the location of the bat. It is necessary to determine how the bat's position x_i and velocity v_i are updated in the d-dimensional search space. The method of updating speed v_i^t and position x_i^t at step t is shown below:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3}$$

In addition, the algorithm is described below:

Algorithm: Network request scheduling algorithm based on bat search

Input: Current network status = $(T_1, T_2, \ldots, T_m)$, Network requests set = $(t_1, t_2, \ldots, t_n)$

Output: Map Sequence = $(x_1, x_2, \ldots, x_n)$

Valuation function: $g(x) = \sum_{i=1}^{n} y_i * t_i$ $\quad y_i = \begin{cases} 1 & x_i > 0 \\ 0 & x_i = 0 \end{cases}$ Objective function: $f(x) = max(g(x))$

Initialize the Map Sequence and Velocity sequence $v = (v_1, v_2, \ldots, v_n)$

Set the pulse frequency sequence $f = (f_1, f_2, \ldots, f_n)$

Initialize the pulse frequency r_i and loudness sequence A_i

for i=1 to N // Iterate N times

//Adjust the frequency, produce new solutions and update the position and speed (expression(1),(2),(3))

if rand > r

end if

if rand < A & $F_{new} < g(x_*)$ // Valuation of the new solution

end if // Accept the new solution, Increase r_i, decrease A_i

// Compare the new solution with the current one and update the current one

end for

return Map Sequence

Among them, the nearest neighbor of the best solution generates the local optimal solution. The update formula is listed as follows:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t \tag{4}$$

A^t is the average of loudness of all bats at time t, and $\varepsilon \in [-1, 1]$ represents directions and lengths that are randomly generated.

4.2 Discrete bat algorithm

However, the algorithm mentioned above cannot be applied directly to our model because the bat location solution is continuous. In the simplified SDN request model, the correspondence between the request and the idle network is discrete. Therefore, the algorithm should be discretized.

Nakamura [23] and Sabba [24] proposed a binary version of binary bat algorithm (BBA) to solve the problems of feature selection in pattern recognition and graph coloring. Among them, the key to converting continuous values into the binary is the introduction of the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

For certain $x \in (-\infty, +\infty)$, we have $S(x) \in (0, 1)$. Moreover, assign the value of x to 0 or 1 by comparing the rand function with $S(x)$, thus changing the continuous value to a discrete binary value.

$$x = \begin{cases} 1 & \text{if } \text{rand} < S(x) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

However, as for the map sequence = $(x_1, x_2, \ldots, x_n)$, $x_i \in [0, m]$, m is the number of idle network segments, 0 represents the request is not placed in the first m idle network. Moreover, x_i cannot simply take the value of 0 or 1 since the discrete binary value is not enough.

In order to obtain a continuous discrete value between *[0, m]*, we can refer to the method in the graph coloring problem and convert x_i to a vector of length m. Thus, map sequence = $(x_1, x_2, \ldots, x_n)$ is transformed into an $n * m$ map matrix. As shown in expression 7, the $9 * 3$ matrix satisfies the following conditions:

$$\begin{bmatrix} 1\,0\,0 & 1\,0\,0 & 0\,1\,0 \\ 0\,1\,1 & 0\,0\,1 & 1\,0\,0 \\ 0\,0\,0 & 0\,1\,0 & 0\,0\,1 \end{bmatrix} \tag{7}$$

1. For an idle network of m segments and n requests, the map matrix is an $n * m$ matrix. Each column represents the request, and each line represents the idle network.
2. The value located in (i, j) in matrix is "1" means the jth request is put into the ith idle network.
3. Each column only has a "1" at most, which means a request is placed into one idle network at most.

Each point can get 0–1 values in a discrete binary way. However, there are two problems with this approach. First, it is difficult to ensure that each column is discrete with only one value. Second, the matrix is too sparse and space is wasted when m become larger.

Convert x_i to a vector of length k, where $2^{k-1} \le m < 2^k$, k is the length of binary form about m. Then the map matrix is transformed into an $n * k$ matrix. As shown in expression 8, the $9 * 2$ matrix satisfies the following conditions:

Table 1 Idle network sequence and request sequence

Serial number	Idle network sequence	Request sequence
1	[4,4,3,3,3,1,1]	[1,1,2,2,2,2,3,3,3]
2	[4,7,2,5,3,1,2]	[2,1,3,4,1,2,3,5,2]
3	[7,6,1,6,1,2,5]	[2,1,6,3,2,6,5,5,5]
4	[1,4,3,3,3,1,1]	[1,1,2,2,2,2,3,3,3]
5	[6,8,3,5,3,1,1]	[3,1,2,4,1,2,3,5,3]
6	[4,6,2,5,3,1,1]	[2,1,3,2,1,2,3,5,2]

$$\begin{bmatrix} 1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,1\,0\,1\,1\,1\,0\,1 \end{bmatrix} \tag{8}$$

1. For an idle network of m segments and n requests, the map matrix is an $n * k$ matrix, where $2^{k-1} \le m < 2^k$.
2. Each column vector is a binary number. In addition, $a_i = v$ means the ith request is put into the vth idle network.
3. The value of each column satisfies $a_i = v \le m$. In addition, $a_i = 0$ means the ith request is not put in the former idle network.

In this way, 0–1 values can be obtained by binary discretization from each point. In order to take advantage of the last position of bat, it is different from the direct use of $S(v_{\mathrm{id}}^t)$ and $S(\varepsilon A^t)$ in expressions 5 and 6. We calculate x_i^t in expression 3 and x_{new} in expression 4, and then transform it into an $n * k$ matrix. Finally we use $S(x_{\mathrm{id}}^t)$ and $S(x_{\mathrm{new}})$ for discrete.

5 Results and discussion

The simulation experiment in this paper was performed on a PC with an Intel (R) Core (TM) 2 Due CPU E7500E @ 2.94 GHz, and 8.00 GB memory. The evaluation algorithm uses the discrete bat algorithm and compares it with the results of the greedy algorithm, dynamic programming algorithm as well as the optimal solution, respectively. We use MATLAB R2015a to run the experiment code.

Since the selecting process introduces a randomization factor, for the solution that does not satisfy the constraint condition, we discard and re-select it. In order to get the solution that satisfies the constraint, it is necessary to do a lot of random selection in the final convergence to get the reasonable discrete solution. Otherwise, the time overhead is very large and uncontrollable, so the maximum number of choices is introduced, which makes the algorithm get the result in a relatively reasonable time.

The strategy of greedy algorithm is fixed. In this paper, we take the maximum value of the remaining requests for each idle network in turn. Obviously, whether the greedy algorithm can get the optimal solution is determined by the characteristics of the data. This paper chooses the data and considers whether the greedy algorithm can get the optimal solution. Table 1 gives an example of data sets for idle networks and requests.

The results of the greedy algorithm, dynamic programming algorithm, discrete bat algorithm as well as the optimal can be calculated for each set of data. We list the maximum request time that can be placed in an idle network sequence and list the mapping sequence based on the greedy algorithm, dynamic programming algorithm, discrete bat algorithm, and finally, list the ratio of maximum request time to optimal solution.

For the discrete bat algorithm, the parameters are set as follows: the number of population n is 10, the amplitude A is 0.25, the pulse frequency r is 0.5, and the number of iterations N is 20. As shown in Table 2:

Table 1 is an example of a human-constructed data with different characteristics. In section III, for a single type 0–1 request, it is easy to map it into multi-knapsack problems, taking the length of occupation time into account only. Therefore, the request sequence may be regarded as the item sequence, the request duration is similar to the size of item, the idle network of the 0–1 bandwidth model is similar to the knapsack, and the idle time can be regarded as the size of a knapsack. The optimization goal is to place the request into an idle network, making the total length of request that has been put into the idle network always the longest.

Each number x with subscript y in the sequence of each algorithm's solution represents the yth request is put into the xth idle network, and $x = 0$ means the yth request is not put into any idle network. The ratio between the solution of each algorithm and length of

Table 2 Results of greedy algorithm and discrete bat algorithm

Serial number	Optimal solution	Greedy sequence	Greedy solution	Ratio	Bat sequence	Bat solution	Ratio
1	19	[1,2,4,5,0,0,1,2,3]	15	78.95%	[7,6,4,2,2,0,3,5,1]	17	89.47%
2	23	[3,0,4,1,6,7,5,2,4]	22	91.67%	[3,6,2,1,2,7,5,4,2]	23	95.83%
3	27	[6,1,1,0,0,2,4,7,0]	25	89.28%	[6,3,4,0,1,2,7,1,0]	27	96.42%
4	15	[1,2,5,0,0,0,2,3,4]	13	81.25%	[7,6,2,2,4,0,0,3,5]	14	87.5%
5	24	[3,7,0,2,7,2,4,1,5]	22	78.57%	[1,6,2,2,7,1,5,4,3]	24	85.71%
6	21	[3,1,1,4,2,5,4,2,0]	19	86.36%	[5,7,2,3,5,1,2,4,1]	21	95.45%

Table 3 Results of dynamic programming algorithm and discrete bat algorithm

Serial number	Optimal solution	Dynamic programming sequence	Dynamic programming solution	Ratio	Bat sequence	Bat solution	Ratio
1	19	[6,7,4,5,0,0,1,2,3]	15	78.95%	[7,6,4,2,2,0,3,5,1]	17	89.47%
2	23	[3,6,2,1,0,7,2,4,5]	22	91.67%	[3,6,2,1,2,7,5,4,2]	23	95.83%
3	27	[1,0,2,0,6,4,1,0,7]	24	85.71%	[6,3,4,0,1,2,7,1,0]	27	96.42%
4	15	[1,6,5,0,0,0,2,3,4]	13	81.25%	[7,6,2,2,4,0,0,3,5]	14	87.5%
5	24	[3,7,1,1,0,2,2,4,5]	23	82.14%	[1,6,2,1,7,3,5,2,2]	24	85.71%
6	21	[3,1,1,2,6,2,5,4,0]	20	90.91%	[5,7,2,3,5,1,2,4,1]	21	95.45%

each network sequence is calculated in Tables 2 and 3. The bat algorithm has achieved better results than greedy algorithm through the comparison. Various situations are also covered in the data set where the greedy strategy cannot obtain the optimal solution.

The total length of idle network for the first group of networks is 19, the total length of the request is 19, so the requests can be all placed in the idle network in optimal solution. The total length of the idle network for the third group is 28, the total length of the request is 35, the total length of the request is longer than the total length of the idle network, and the idle network cannot all be filled. The total length of the idle network for the fourth group is 16, the total length of the request is 19, total length of the request is longer than the total length of idle network, and the idle network cannot be all filled either. The total length of the idle network for the six group is 22, the total length of the request is 21, and finally the requests are all placed in the idle network in optimal solution and bat algorithm. As shown in Figs. 1 and 2, the discrete bat algorithm in this paper may not be able to obtain the optimal solution, but it outperforms the greedy strategy under various conditions.

In addition, the simplest model can also be mapped into the multi-knapsack problem, in combination with dynamic programming using the mathematical model we have mentioned before. The deterministic algorithm cannot be accepted in time. Comparison between solutions of greedy, dynamic programming and bat algorithm are listed in Table 3, Figs. 2 and 3. The bat algorithm has achieved better results among these algorithms through the comparison.

However, the discrete bat algorithm is essentially a random algorithm and does not necessarily yield better results. The probability of obtaining better results correlates with its parameters. We further explore the impact of different parameters. The data of group one were calculated 30 times with different parameters, and the average value was compared every time. The results are shown in Table 4:

From Table 4, we can infer that in the case of the same number of iterations, it is reasonable that the change in population number n is more likely to cause a change in the result. Because amplitude A and pulse frequency r affect the convergence speed more, they have an effect on the convergence process of bat algorithm, but their impact on the result is relatively small. The algorithm in

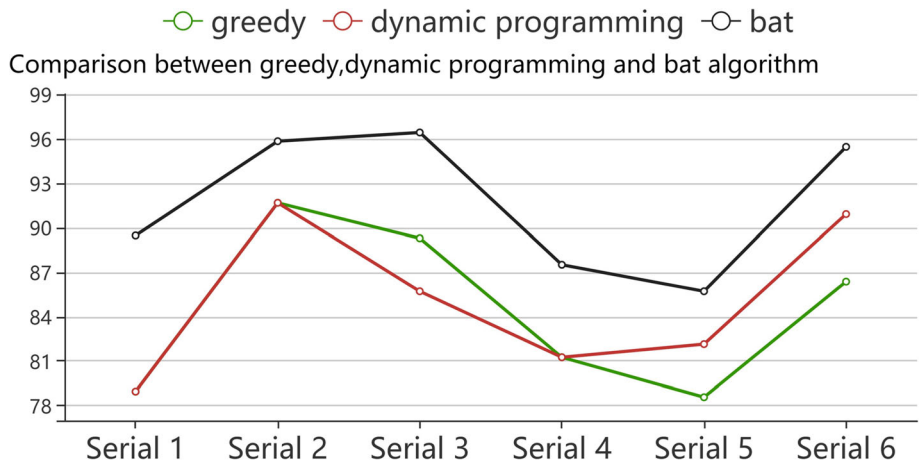

Fig. 2 Comparison of results between greedy, dynamic programming, and bat algorithm—line chart. In Fig. 2, the vertical axis represents the ratio of idle network occpied by request and the horizontal axis represents multiple sets of data. The maximum time of request that can be placed in an idle network sequence, the mapping sequence based on greedy algorithm, dynamic programming and discrete bat algorithm, and the ratio of idle network occpied by request are listed in Tables 2 and 3. From the figure, we can infer that the bat algorithm is modified by binary code and sigmoid function which makes it suitable for discrete simplified model. Compared with the other two algorithms, the algorithm has achieved good results

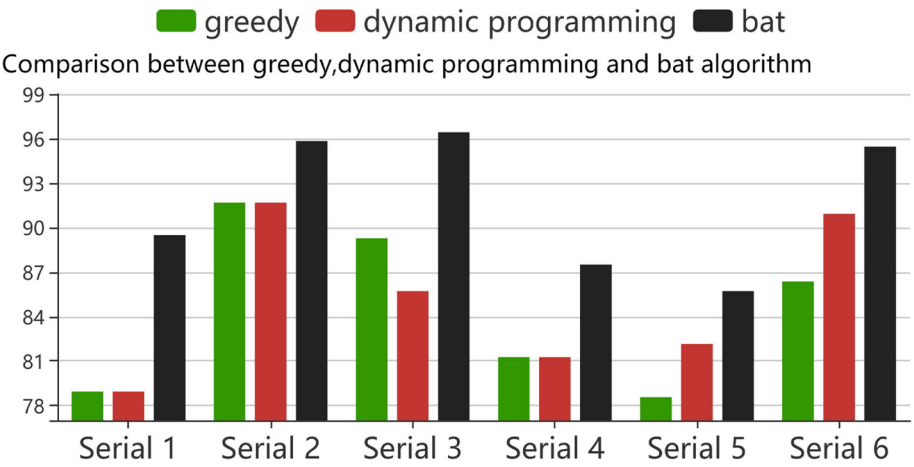

Fig. 3 Comparison of results between greedy, dynamic programming, and bat algorithm—bar chart. The optimal solution, greedy solution, and the solution of the discrete bat algorithm can be calculated for each set of data. In Fig. 3, the vertical axis represents the correct rate and the horizontal axis represents the same meaning in Fig. 2. It can be concluded that results for each data obtained by greedy algorithm and dynamic programming algorithm are relatively similar, but they all get lower ratio of idle network than that of bat algorithm. The bat algorithm has achieved better results than greedy algorithm and dynamic programming algorithm for each set of data, and results of bat algorithm achieves higher ratio of idle network occpied by request

this paper does not modify the amplitude A and the pulse frequency r, so we will not discuss more on this. Then, the number of iterations is reduced and the result is significantly changed because the reduction in the number of iterations may has already output the results when the algorithm have not been stabilized at a better value.

Finally, after reducing N and n to one, the algorithm degenerates to random allocation actually. In short, the bat algorithm can get satisfactory results when choosing the appropriate parameters.

6 Conclusions

Considering the ability of SDN network to gather information as well as learning from the idea of label distribution and resource reservation, the model of SDN network scheduling is proposed in this paper. Against limitations of heuristic intelligence bat algorithm about solving discrete problems in previous works, binary encoding, and sigmoid functions are applied to discrete results to improve the network utilization.

The advent of SDN allows people to control scheduling network packets more finely. Using overall scheduling capability of the controller and data request characteristics of different applications, we could improve network resource utilization. This paper discusses the network-scheduling problem under the SDN environment and establishes the network model and the request model under the environment. The bat algorithm is modified by a binary code and sigmoid function to make it suitable for a discrete simplified model. Compared with greedy and dynamic programming algorithm, the algorithm has achieved excellent results, especially in solving discrete problems. Through simulation experiments and comparison with the greedy strategy, the rationality of bat algorithm applied to SDN network-scheduling model is verified.

Our future work can be done on the one hand to further refine the model to adapt to different complex scenarios, on the other hand, the bat algorithm needs to be modified more meticulously and be used in complex models. Alternatively, we can use other algorithms to solve complex models.

Table 4 The average result of different parameters of the bat algorithm

Number of iterations N	Number of population n	Amplitude A	Pulse frequency r	Average value	Maximum value	Minimum value	Variance
20	10	0.25	0.5	17.2	19	16	0.55
20	5	0.25	0.5	16.8	19	16	1.30
20	10	0.5	0.5	17.2	19	17	0.36
20	10	0.25	0.25	17.1	19	16	0.49
5	10	0.25	0.5	16.7	19	16	0.81
1	1	2	0.5	12.7	17	8	5.13

Abbreviation
BA: Bat algorithm; EBA: Evolved bat algorithm; IP: Internet protocol; MPLS: Multiprotocol label switching; NP: Nondeterministic polynomial time; QoS: Quality of service; RISC: Reduced instruction set computer; RMT: Reliable multicast transport; RSVP: Resource reservation protocol; SDN: Software-defined network; TOS: Type-of-service

Acknowledgements
Thanks to the team members in state key lab. Xin Tao gives some feedback on the experiment.

Funding
This work is supported by the State Grid Corporation of China under the project title: "The Improved Core Analysis Algorithms and Utilities for Smart Grid Big Data" (520940180016) and Beijing Natural Science Foundation (L171010, 4174099).

Availability of data and materials
The dataset was described in the paper, which are some idle sequences and request sequences.

Authors' contributions
RHL contributes the main idea and proposed the model. ZZ helps to implements the code and experiment. Both authors read and approved the final manuscript.

Authors' information
Rongheng Lin was born in Xiamen, P.R. China in 1981. He received the B.S (2004) and Ph. D (2010) degrees in Computer Science from Beijing University of Posts and Telecommunications (BUPT). From 2008 to 2010, he was a Visiting Research Assistant with the GVU Center, Georgia Institute of Technology, USA. Since 2010, he has been an Associate Professor with the State Key Lab of Networking and Switching Technology, BUPT. He is the author of one book, more than 50 articles, and more than 15 inventions. His research interests include data analysis on smart grid, SDN.
Zezhou Ye, graduated student in Beijing University of Posts and Telecommunications (BUPT). He research interests include SDN and big data.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 21 November 2017 Accepted: 7 May 2018
Published online: 16 May 2018

References
1. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. ONF White Paper, 2012.
2. Z Qingyun, C Ming, Z Guangsong, Research on SDN based on OpenFlow [J]. J. Softw. **24**(5), 1078–1097 (2013)
3. Kompella V, Lasserre M. Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling[J]. 2007.
4. PW Paul, RSVP and integrated services in internet: A tutorial [J]. IEEE Commun. Mag. **35**(5), 100–106 (1997)
5. F Hu, Q Hao, K Bao, A survey on software-defined network and openflow: From concept to implementation[J]. IEEE Commun. Surv. Tutorials **16**(4), 2181–2206 (2014)
6. H Kim, N Feamster, Improving network management with software defined networking[J]. IEEE Commun. Mag. **51**(2), 114–119 (2013)
7. DY Huang, K Yocum, AC Snoeren, High-fidelity switch models for software-defined network emulation. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 43–48 (2013)
8. S Sezer, S Scott-Hayward, PK Chouhan, et al., Are we ready for SDN? Implementation challenges for software-defined networks[J]. IEEE Commun. Mag. **51**(7), 36–43 (2013)
9. MA Sharkh, M Jammal, A Shami, et al., Resource allocation in a network-based cloud computing environment: Design challenges[J]. IEEE Commun. Mag. **51**(11), 46–52 (2013)
10. P Bosshart, G Gibb, HS Kim, et al., Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN[C]//ACM SIGCOMM computer communication review. ACM **43**(4), 99–110 (2013)
11. LE Li, ZM Mao, J Rexford, Toward software-defined cellular networks. In: Software defined Networking (EWSDN), 2012 European Workshop On, pp. 7–12 (2012). IEEE
12. PW Tsai, JS Pan, BY Liao, et al., Bat algorithm inspired algorithm for solving numerical optimization problems[C]//applied mechanics and materials. Trans. Tech. Publ. **148**, 134–137 (2012)
13. Fister Jr I, Fister D, Yang X S. A Hybrid Bat Algorithm[J]. arXiv Preprint arXiv: 1303.6310, 2013.
14. AH Gandomi, XS Yang, Chaotic bat algorithm[J]. J. Comput. Sci. **5**(2), 224–232 (2014)
15. R Sathya, R Thangarajan, Efficient anomaly detection and mitigation in Software defined networking environment. In: Electronics and Communication Systems (ICECS), 2015 2nd International Conference On, pp. 479–484 (2015). IEEE
16. S Ehsan, B Hamdaoui, A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks[J]. IEEE Commun. Surv. Tutorials **14**(2), 265–278 (2012)
17. HA Taha, Integer Programming: Theory, Applications, and Computations, (1975). Academic Press
18. G Ausiello, P Crescenzi, G Gambosi, et al., Complexity and approximation. (1999). Springer-Verlag Berlin Heidelberg
19. MM Akbara, MS Rahman, M Kaykobad, et al., Solving the multidimensional multiple-choice knapsackproblem by constructing convex hulls. Comput. Oper. Res. **33**(5), 1259–1273 (2006)
20. L Wang, X Zheng, S Wang, A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem[J]. Knowl.-Based Syst. **48**, 17–23 (2013)
21. PG Tharanipriya, P Vishnuraja, "Hybrid genetic algorithm for solving knapsack problem", in *Proc. International Conference on Information Communication and Embedded Systems (ICICES' 13)*, pp. 416–420, IEEE
22. X-S Yang, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Vol. 284 of Studies in Computational Intelli-Gence*, ed. by J Gonzlez, D Pelta, C Cruz, G Terrazas, N Krasnogor. A new metaheuristic bat-inspired algorithm (Springer, Berlin, 2010), pp. 65–74
23. RYM Nakamura, LAM Pereira, KA Costa, D Rodrigues, JP Papa, XS Yang,a binary bat algorithm for feature selection. In: Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference On, pp. 291–297 (2012). IEEE
24. H Djelloul, S Sabba, S Chikhi, "Binary bat algorithm for graph coloring problem," Complex Systems (WCCS), 2014 Second World Conference on 10–12 Nov, pp. 481,486, 2014. doi: https://doi.org/10.1109/ICoCS.2014.7060988