基于改进蚁群算法的 SDN 网络负载均衡研究

郑本立 李跃辉
(南京邮电大学通信与信息工程学院 南京 210003)

摘 要 考虑到服务器处理性能的 SDN 网络负载均衡研究对于合理分配资源及提高服务性能具有重要意义,文中提出了基于改进蚁群算法的 SDN 网络负载均衡研究。首先对 SDN 网络结构及负载均衡进行了分析;然后根据 SDN 网络负载均衡的实际需求,对传统蚁群算法进行了改进,将每条链路带宽的空闲率作为蚁群算法的信息素,将计算机处理器的性能和需要传输的数据量作为启发信息,采用多重启发方式对传统蚁群算法进行改进,并对改进算法的收敛性进行了证明;最后对改进算法的性能进行验证。仿真结果表明:该算法具有收敛速度快、耗时短的优点。SDN 网络负载均衡仿真实验也证明了该方法的有效性和可行性。

关键词 SDN,蚁群算法,负载均衡
中图法分类号 TN915,TP183 **文献标识码** A

Study on SDN Network Load Balancing Based on IACO

ZHENG Ben-li LI Yue-hui
(School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract The study on SDN network load balancing considering server processing performance is of great significance to reasonably allocate resources and improve service performance. Therefore, this paper studied on SDN load balancing based on improved ant colony algorithm. Firstly, the structure and load balance of SDN are analyzed. Then, according to the actual demand of SDN load balancing, the traditional ant colony algorithm is improved. The idle rate of each link bandwidth is taken as the pheromone of the ant colony algorithm, the performance of computer processor and the amount of data needed to be transmitted is taken as the enlightening information, and the traditional ant colony algorithm is improved by multiple heuristics. The convergence of the improved algorithm is also proved. Finally, performance verification simulation is performed for the improved algorithm. Simulation results verify that the proposed algorithm has the advantages of fast convergence speed and short time consuming. Simulation of SDN network load balancing also proves the validity and feasibility of this method.

Keywords SDN, Ant colony algorithm, Load balancing

1 引言

在 SDN(Software-Defined Networking)网络下选择合适的路径实现链路资源的合理分配,使负载达到平衡,可以合理分配网络资源,提高服务性能及整体灵活性[1-3]。

近年来针对 SDN 网络负载均衡的研究引起了越来越多学者的关注。文献[4]提出了一种基于迭代的算法,并对算法的逼近性能进行了分析,用以解决 SDN 网络中链路和控制器(LBR-LC)负载均衡的路由问题。仿真结果证明,相比于以前的解决方法,该方法只需要增加 3%的最大链路负载,即能够使控制器的最大响应时间减少 70%[4]。文献[5]提出了一种面向服务的 sfc 负载均衡机制,考虑并分类每个终端设备所需的服务类型和优先级,采用启发式算法对 SFCs 之间的传输路径进行规划,以减少各个 SF 的负载,提高整体网络性能。仿真结果表明,该方法可以缩短数据传输时间,达到负载均衡[5]。文献[6]引入了 DSSDN 的需求和供给曲线来研究 SDN 网络的负载均衡,采用了卡鲁什-库恩-塔克条件,有效地提高了响应速度,在实现过程中使用设备控制器上运行的虚拟线程来代替实际的设备进行最优决策实验。结果表明,在迁移过程中,DSSDN 在不造成网络状态干扰的情况下,稳定了负载增长,提高了 QoS,增加了终端用户的效用[6]。文献[7]提出了一种基于 SDN 的机器对机器(M2M)网络的流量感知负载均衡方案,利用 SDN 网络对网络的监控能力,提出的负载均衡方案可以通过流量识别和路由来满足不同服务质量的要求。实验结果表明,与非 SDN 负载均衡方案相比,该方案可以将服务响应时间减少 50%[7]。文献[8-10]对 SDN 网络的负载均衡架构、研究现状以及分层式控制器的负载均衡机制等进行了深入的研究[8-10]。

上述研究主要是针对如何均衡服务器之间的流量负载,并未考虑服务器的性能差别。鉴于此,本文提出了基于改进蚁群算法的 SDN 网络负载均衡研究,将每条链路带宽空闲率作为蚁群算法的信息素 α,将计算机处理器性能和需要传输的数据量作为启发信息,确定初始位置和目标位置,并通过实验仿真验证了本文所提方法的有效性和可行性。

2 SDN 网络架构

2.1 SDN 架构

软件定义网络(SDN)由 Nick 提出。SDN 采用新的架构

郑本立(1992—),男,硕士生,主要研究方向为通信与信息系统;李跃辉(1963—),男,副教授,主要研究方向为通信与信息系统。

替代了传统 TCP/IP 网络控制层与数据层集中在一起的方式，实现了控制层与数据层的分离，通过控制层实现了网络流量的灵活控制，实时监控整个网络的状况[11-12]。SDN 网络从上到下依次分为应用层、控制层、设备层，其基本架构如图 1 所示。

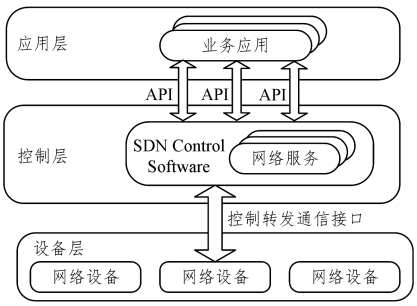

图 1 SDN 的基本架构图

设备层由一系列具有转发功能的网络设备组成。设备层的网络设备按照控制层的指令进行数据转发，并将当前链路状态和统计消息向控制层进行传递[13]。

控制层是 SDN 的核心部分，接收应用层的请求信息，并进行任务解析，将控制命令下发给网络设备层[14]。同时，SDN 还有对网络设备层上报的链路转台和统计信息进行分析的功能。

应用层的业务应用向 SDN 控制器传输业务需求，不需要考虑网络设备情况，只需要将自身的任务传输给控制层，由控制层对任务进行解析派送。

2.2 负载均衡

负载均衡是指将从外部接收到的负载进行处理后平均分配到多台服务器上，以达到最大化利用服务器、提高数据传输效率、增强网络稳定性的目的[15]。通常情况下是先对拥有的服务器资源进行统计，然后根据负载情况进行负载的均衡划分。一般情况下的负载均衡架构如图 2 所示。

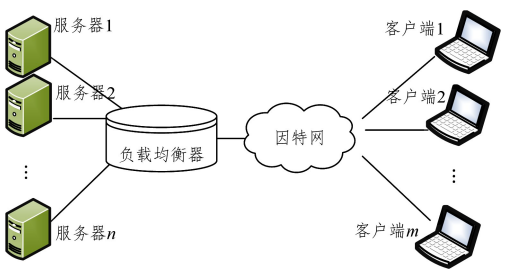

图 2 负载均衡架构

根据网络环境的不同，负载均衡的分类方式也不同。根据实现方式的不同，负载均衡分为硬件负载均衡和软件负载均衡。硬件负载均衡处理速度较快，但是会增加成本；软件负载均衡是指在主机上通过处理软件实现负载任务的划分，但是处理速度会下降。根据流量管理的不同，负载均衡分为静态负载均衡和动态负载均衡。静态负载均衡是在算法实施之前确定负载的分配方案；动态负载均衡指的是在负载分配过程中实时调整负载分配的策略。

3 基于改进蚁群算法的负载均衡研究

3.1 改进蚁群算法

蚁群算法（Ant Colony Optimization，ACO）是由 Dorigo 等提出的群智能算法[16]。蚂蚁在寻找食物时会在路径上释放信息素，后续蚂蚁会沿着信息素最浓的路径寻找食物，这样就会使得路径越短的信息素越强，则通向食物源较短的路径上的信息素会越来越浓，较长路径上的信息素越来越少，从而最终确定出最优路径[17]。为了提高传统 ACO 算法的性能，改进的蚁群算法（Improved Ant Colony Optimization，IACO）被提出。

3.1.1 蚁群算法的基本实现过程

Step1 种群初始化。

假设初始种群有 M 只蚂蚁，记作 $Y(0)$；迭代次数为 $maxgen$；节点数为 n。第 i 只蚂蚁走过路径的适应度记作 fit_i。η_{ij} 为启发因素，表示蚂蚁从第 i 点到第 j 点的启发程度。τ_{ij} 是(i,j)边的信息素，定义初始时刻每条边的大小相等。$\Delta\tau_{ij}^{k}$ 是蚂蚁 k 迭代时在(i,j)边留下的信息素量。ρ 为信息素蒸发系数。$P_{ij}^{k}(t)$是蚂蚁 k 从第 i 点向第 j 点移动的概率；t 为时刻。

Step2 求取适应度。

根据实际问题确定适应度函数，求取每只蚂蚁走过路径的适应度 $fit(y)$。

$$fit(y)=[y_1,y_2,\cdots,y_M]^{\mathrm{T}} \tag{1}$$

$$y_i=\frac{1}{m}\sum_{j=1}^{m}(O_j-T_j)^2 \tag{2}$$

其中，m 是样本总数，O_j 和 T_j 是第 j 个预测输出值及实际输出值。

Step3 根据适应度，释放信息素。

蚂蚁寻找到食物源之后，路径上留下的信息素为：

$$\tau_{ij}(t+n)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij} \tag{3}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{4}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{fit_k}, & \text{蚂蚁 } k \text{ 本次周游经过}(i,j)\\ 0, & \text{蚂蚁 } k \text{ 本次周游不经过}(i,j)\end{cases} \tag{5}$$

其中，Q 是信息素强度，fit_k 是第 k 只蚂蚁的适应度。初始 $\tau_{ij}(0)=C$。为了避免信息素过度集中，设定信息素的取值范围为$[\tau_{\min},\tau_{\max}]$。

Step4 根据信息素及选择函数，确定行走路径。

求取转移概率 $P_{ij}^{k}(t)$，根据轮盘赌方法确定下一个节点，更新禁忌表，遍历所有的节点。

$$P_{ij}^{k}=\frac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{s\in J_k(i)}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}(t)]^{\beta}} \tag{6}$$

$$j\in J_k(i);P_{ij}^{k}=0,j\notin J_k(i)$$

其中，α,β 分别为信息素和启发式因素的重要程度。$J_k(i)$是第 k 只蚂蚁下一次可以行走的所有节点。τ_{ij}，η_{ij} 分别是路径(i,j)的信息素浓度和启发信息，启发式因素为 $\eta_{ij}=1/d_{ij}$。

Step5 将本次所有蚂蚁行走节点信息进行记录，从中筛选出拥有最佳适应度的蚂蚁信息。

Step6 判断是否达到结束条件，即是否达到设定的迭代次数及是否达到设定阈值。若是则停止，若否则转 Step2。

3.1.2 多重启发蚁群算法

信息素和启发素决定着蚁群算法寻优的精度及收敛速度。信息素决定着已经走过节点的信息，启发素决定着下一节点的走向。启发信息指的是根据外界状况指导蚂蚁下一步可选择的节点概率。当前 ACO 中启发信息的计算是求取当前节点和即将行走节点的距离的倒数，$\eta_{ij}=1/d_{ij}$，这种求取

启发信息的方法对其方向的选择带来了很大限制，不能充分利用已走过的节点信息为下一节点的行走做指导。

期望启发因子 β 用于描述启发信息的重要程度，增大 β 能够加快算法的收敛速度，但也降低了其全局寻优的性能[18]。算法求解过程中要合理调整 α，β 的取值，α 表征了蚂蚁行走的随机性，β 表征了 α 在节点选择中的作用，所以在调整时，既要保证目标的明确性，又要保证蚂蚁在行走过程中由于环境的影响每条路径的启发信息存在差异性。因此，本文提出了多重启发的蚁群算法，其示意图如图 3 所示。其中，S_i 是当前节点，S_{j1}，S_{j2}，S_{j3} 是待选节点。对服务器的性能进行综合排序，T_1 和 T_2 为性能最差的服务器。从图 3 中可以看出 S_{j1} 为理想的下一节点，但是数据需要经过性能较差的服务器传输，这样降低了数据处理速度，同样 S_{j2} 和 S_{j3} 也需要经过性能较差的服务器传输数据，所以只采用全局启发信息不能引导蚂蚁选择更好的路径。将当前的环境信息即每台服务器的性能考虑进蚂蚁行走过程中，根据服务器的性能优劣构造蚂蚁的多重启发指导因子，用于指导蚂蚁行走更优的路径。多重启发信息包括全局启发和局部启发：全局启发表示从当前位置到目标位置的距离和可能经过的计算机构成的启发信息，是在整个行程中进行指导的信息；局部启发信息是从当前位置行走到下一位置的指导信息。

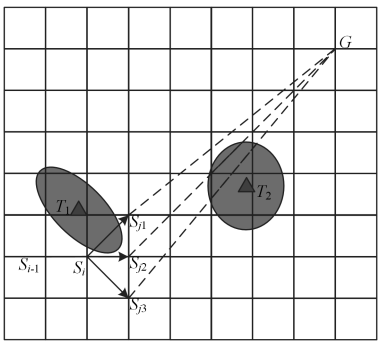

图 3　多重启发示意图

假设当前位置为 s_j，其到终点位置的距离定义为 $d(s_j, G)$，则全局启发的距离 η_{GD} 记作：

$$\eta_{GD}=1/d(s_j,G) \tag{7}$$

此信息可以指导使蚂蚁按照最短的路径行驶到目的地，减少了行走时间。

全局启发信息中的计算机性能 η_{GT} 等于当前位置 s_j 到目标位置需经过的计算机处理器性能 $J_{T,jG}$ 及传输数据 $J_{h,jG}$ 加权和。

$$\eta_{GT}=\omega_2 J_{T,jG}+\omega_3 J_{h,jG} \tag{8}$$

$$J_{T,jG}=\sum_{k=1}^{N_T} J_{T,jG}^{k} \tag{9}$$

局部启发信息为当前位置与待选位置之间的倒数。局部启发信息中计算机的性能 η_{LT} 包括当前位置到下一位置的处理器性能及传输数据加权和的倒数。

$$\eta_{LD}=1/d(s_i,s_j) \tag{10}$$

$$\eta_{LT}=1/(\omega_2 J_{T,ij}+\omega_3 h_j) \tag{11}$$

全局启发和局部启发共同指导蚂蚁的行走。

$$\eta=\lambda\eta_{GD}\eta_{GT}+(1-\lambda)\eta_{LD}\eta_{LT} \tag{12}$$

其中，λ 是全局启发的权重，$0\leqslant\lambda\leqslant 1$。$\lambda$ 的不同取值，决定了全局启发和局部启发的重要程度不同，即为蚂蚁的行走提供不同的指导性意见。

基于多重启发蚁群算法的 SDN 网络负载均衡策略的步骤如下所示。

Step1　任务初始化。将每条链路的带宽空闲率作为蚁群算法的信息素 α，将计算机处理器性能和需要传输的数据量作为启发信息，确定初始位置和目标位置。

Step2　参数初始化：分别确定 α，β，ρ，Q，ω，λ 等参数。

Step3　初始化信息素浓度：求取每个节点的全局启发信息和局部启发信息，并记录。

Step4　在起始节点中放入 N_a 个蚂蚁。

Step5　确定下一步行走位置的集合，并依次计算出行走下一个节点的概率，据此选出最佳的位置行走。

Step6　判断是否到达终点，若到达终点则转 Step7，否则转 Step5。

Step7　选择出最适合蚂蚁行走的路径。

Step8　更新信息素和启发信息。

Step9　判断是否达到终止条件，若达到最大迭代次数或达到设定阈值，则停止迭代，输出最优路径，否则继续 step4 进行迭代。

3.2　收敛性证明

问题 (S, f, Ω) 中 f 为目标函数，S 是候选解，Ω 是约束条件，$s\in S$。为了寻找负载均衡的最优解 s^*，即使 $f(s^*)$ 最小。极小化问题的特征如下所述。

1）定义某问题包含的信息为 $C=\{c_1, c_2, \cdots, c_{Nc}\}$。

2）该问题的状态集合 X，C 中的可能序列为 X 的元素。$x=[c_{x1}, c_{x2}, \cdots, c_{xk}, \cdots]$，定义 $|x|$ 为序列中含有的元素个数。

3）候选解 S 为 X 的子集，$S\subseteq X$。

4）可行状态集合 $\tilde{X}\subseteq X$，X 中符合约束 Ω 的集合。

5）非空最优解集合 S^*，满足 $S^*\subseteq\tilde{X}$ 并且 $S^*\subseteq S$。

考虑计算机处理性能及数据传输量大小的 SDN 网络负载均衡问题属于组合优化问题。文献[19]证明了信息素下限蚁群算法的收敛性。本文提出的多重启发蚁群算法也是信息素浓度限方式，所以，当迭代次数较多的时候，可证明其可收敛至最优解。针对 SDN 网络的负载均衡问题，证明多重启发蚁群算法的收敛性。

引理 1　设信息量 $\tau_{ij}(t)$ 的全局最大值为 $\tau_{\max}$，则 $\forall\tau_{ij}$：

$$\lim_{t\to\infty}\tau_{ij}(t)\leqslant\tau_{\max}=\frac{1}{\rho}\cdot g(s^*) \tag{13}$$

证明：设每段 (i,j) 路径的信息素浓度小于 $g(s^*)$。经过下一次迭代，路径信息素的最大值为 $(1-\rho)\tau_0+g(s^*)$；再经历一次迭代，信息素浓度的最大值为 $(1-\rho)^2\tau_0+(1-\rho)\cdot g(s^*)+g(s^*)$。则 t 次迭代之后，信息素的最大值：

$$\tau_{ij}^{\max}(t)=(1-\rho)^t\cdot\tau_0+\sum_{i=1}^{t}(1-\rho)^{t-i}\cdot g(s^*) \tag{14}$$

因为 $\rho\subset(0,1)$，所以信息素浓度收敛于式(15)。

$$\tau_{\max}=\frac{1}{\rho}\cdot g(s^*) \tag{15}$$

引理 2　设信息素浓度的最小值为 $\tau_{\min}$，则经过信息素更新之后，$\forall(i,j)\subset s^*$，$\tau_{ij}^*(t)\geqslant\tau_{\min}$，因此最优解 $\tau_{ij}^*(t)$ 经过迭代后为：

$$\lim_{t\to\infty}\tau_{ij}^*(t)=\tau_{\max}=\frac{1}{\rho}\cdot g(s^*) \tag{16}$$

定理 1　设 $P^*(t)$ 为经过 t 次迭代后得到的 s^* 最优解概率，那么一定存在某个 t 值，对于任何 $\varepsilon>0$，存在：

$$P^*(t)\geqslant 1-\varepsilon \tag{17}$$

$$\lim_{t\to\infty} P^*(t)=1 \tag{18}$$

证明：因为信息素处于$[\tau_{\min},\tau_{\max}]$之间，可行解状态转移概率$P_{\min}>0$，存在：

$$P_{\min}\geqslant \hat{P}_{\min}=\frac{\tau_{\min}^{\alpha}\eta_{\min}^{\beta}}{(N-1)\cdot\tau_{\max}^{\alpha}\eta_{\max}^{\beta}+\tau_{\min}^{\alpha}\eta_{\min}^{\beta}}>0 \tag{19}$$

其中，N 是蚂蚁行走节点数。若种群中某一只蚂蚁寻找到最优值，则说明算法收敛至最优。$P^*(t)$的下界可以写为：

$$P^*(t)=1-(1-\hat{P})^t \tag{20}$$

则存在$\varepsilon>0$，若 t 足够大，使 $P^*(t)\geqslant 1-\varepsilon$，从而当 $t\to\infty$时，$\lim_{t\to\infty} P^*(t)=1$。证明完毕。

4 实验仿真

4.1 负载均衡仿真

为了验证 IACO 方法在处理 SDN 负载均衡上的有效性，将 ACO，GA-ACO，IACO 算法作为对比进行仿真实验。采用 Matlab2014b 进行算法实验对比，虚拟机内存 1～2 GB，带宽为 512～1024 bit。每个虚拟机的速度为 250～3000 MIPs，处理器数目为 2～6 个。创建 100～500 个任务，其仿真对比图如图 4 所示。

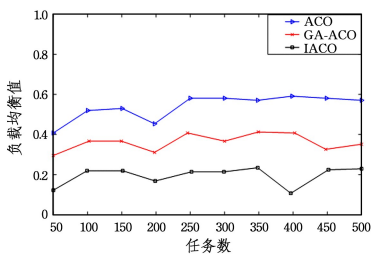

图 4 3 种算法的负载均值比较

4.2 仿真结果分析

从图 4 的仿真可以看出，在处理相同任务时，3 种算法的最短耗时从大到小的顺序为 GA-ACO，ACO，IACO，这说明本文所提的 IACO 方法能够加快蚁群算法的收敛速度，更快地处理完各项任务。从图 4 的对比图中可以看出，相同条件下，本文提出的 IACO 算法的收敛速度高于 GA-ACO 和 ACO 算法。当任务书相等时，IACO 算法的负载均衡值低于 GA-ACO 和 ACO 算法。综上所述，本文所提 IACO 算法在最短完成时间、收敛速度、负载均衡等性能上均优于 ACO 算法和 GA-ACO 算法，从而验证了本文所提 IACO 算法在处理 SDN 网络负载均衡时具有有效性和可行性。

结束语 针对 SDN 网络负载的实际需求，提出了一种改进的蚁群算法，并对算法的收敛性进行了证明。仿真实验结果证明了该算法的可靠性，并且其可以求取出 SDN 负载均衡期望的链路分配给新入网的数据，达到加快数据传输的目的。对于 SDN 负载均衡的研究，未来还需更全面地考虑影响数据流传输的环境因素，并根据影响因子的不同给出最佳权重分配，以期获得智能的负载均衡和更快的处理速度。

参 考 文 献

[1] CELENLIOGLU M R, TUYSUZ M F, MANTAR H A, et al. An SDN-based scalable routing and resource management model for service provider networks[J]. International Journal of Communication Systems, 2018, 31(8): e3530.

[2] SHANG F J, MAO L, GONG, W J. Service-aware adaptive link load balancing mechanism for Software-Defined Networking[J]. Future Generation Computer Systems-The International Journal of Escience, 2018, 81: 452-464.

[3] YANG X W, XU H L, HUANG L S, et al. Joint Virtual Switch Deployment and Routing for Load Balancing in SDNs[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 397-410.

[4] WANG H B, XU H L, LIU S, et al. Load-balancing routing in software defined networks with multiple controllers[J]. Computer Networks, 2018, 141(4): 82-91.

[5] CHIEN W C, LAI C F, CHO H H, et al. A SDN-SFC-based service-oriented load balancing for the IoT applications[J]. Journal of Network and Computer Applications, 2018, 114: 88-97.

[6] SAHOO K S, TIWARY M, S BAHOO, et al. DSSDN: Demand-supply based load balancing in Software-Defined Wide-Area Networks[J]. International Journal of Network Management, 2018, 28(4): 1-25.

[7] CHEN Y J, WANG L C, CHEN M C, et al. SDN-Enabled Traffic-Aware Load Balancing for M2M Networks[J]. IEEE Internet of Things Journal, 2018, 5(3): 1797-1806.

[8] 张敏敏，章韵，段元新. 基于软件定义网络的多控制器负载均衡架构[J]. 计算机工程，2016，42(9)：26-32.

[9] 朱世珂，束永安. 基于软件定义网络的分层式控制器负载均衡机制[J]. 计算机应用，2017，37(12)：3351-3355，3360.

[10] 柳林，周建涛. 软件定义网络控制平面的研究综述[J]. 计算机科学，2017，44(2)：75-81.

[11] SHI JG, ZHU W, JIA K Y, et al. Multi-controller Deployment Algorithm Based on Load Balance in Software Defined Network [J]. Journal of Electronics&Information Technology, 2018, 40(2): 455-461.

[12] WANG Q, GAO L R, YANG Y T, et al. A load-balanced Algorithm for Multi-Controller Placement in Software-Defined Network[J]. Me Chatronic Systems and Control, 2018, 46(2): 72-81.

[13] 胡涛，张建辉，毛明. SDN 中基于迁移优化的控制器负载均衡策略[J]. 计算机应用研究，2018，35(2)：559-563.

[14] YU G O, IVAN V C. SDN Load Balacing for Secure Networks [J]. Systems and Means of Informatic, 2018, 28(1): 123-138.

[15] ZHOU Y, ZHENG K F, NI W, et al. Elastic Switch Migration for Control Plane Load Balancing in SDN[J]. IEEE Access, 2018, PP(99): 3909-3919.

[16] YACINE M, DJAMILA R. High performance of Maximum Power Point Tracking Using Ant Colony algorithm in wind turbine[J]. Renewable energy, 2018, 126: 1055-1063.

[17] RABORN ANTHONY W, LEITE WALTER L. ShortForm: An R Package to Select Scale Short Forms With the Ant Colony Optimization Algorithm[J]. Applied Psychol Ogical Measurement, 2018, 42(6): 516-517.

[18] SIVARAJ R, PRIYA R D. Estimation of incomplete values in heterogeneous attribute large datasets using discretized Bayesian max-min ant colony optimization[J]. Knowledge and Information Systems, 2018, 56(2): 309-334.

[19] STUTZLE T, DORIGO M. A short convergence proof for a class of ant colony optimization algorithms[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 358-365.