一种平滑的基于链路排序的节能算法

黄 鸿 虞红芳
(电子科技大学通信与信息工程学院 成都 611731)

摘 要 针对目前网络级节能方案中存在的链路状态切换频繁的问题,提出了一种平滑的基于链路排序的启发式节能算法。该算法使用了一种新的排序机制,使得连续两次节能策略下的链路状态切换尽可能小,以达到平滑的目的。此外,该算法还综合考查了线卡能耗与链路能耗,以获得更高的节能效率。仿真实验表明,与一种基于链路排序的贪心节能算法相比,该算法链路状态切换频率更小,且节能效率更高;与追求最优节能效率的greenTE算法相比,在节能效率相差不大的情况下,该方法具有更好的平滑性以及更低的时间复杂度。

关键词 节能算法,链路排序,平滑,低复杂度
中图法分类号 TP393.0 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.011

Smooth Energy-saving Algorithm Based on Link-ranking

HUANG Hong YU Hong-fang
(Institute of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract To decrease the high switching frequency of link states, which is existed in many of today's energy-saving algorithms, a smooth energy-saving algorithm based on link-ranking was proposed. In the algorithm, there is a unique sorting mechanism to minimize the switching frequency of link state during continuous time, so that the strategy of the algorithm is smooth. Moreover, the proposed algorithm balances the energy consumption of line-cards and links, in order to achieve higher energy-saving efficiency. The simulation results shows that this method outperforms a greedy algorithm based on link-ranking in terms of energy-saving efficiency and switching frequency of link state, and outperforms greenTE, which pursues the optimal efficiency of energy-saving, in terms of switching frequency of link state and time complexity.

Keywords Energy-saving algorithm, Link-ranking, Smooth, Low time complexity

随着网络规模的不断扩大以及未来网络的逐渐成型,能源效率应作为网络设计标准及运行机制的一部分,渗透到网络的各个层次。对绿色网络的需求主要受两个方面因素的驱动:1)出于对环境的保护,需要增大能源的利用效率,减少能源浪费,并缓解 CO_2 排放所带来的影响;2)出于运营成本的考虑,运营商总是希望减少在设备能耗上的投资,同时保障理想的服务水平。文献[1]指出,到2020年,网络设备的耗能将达到95 GW,网络中的能耗问题亟待解决。现有的节能技术可以分为两种:1)设备级节能技术,该技术已发展多年,目前商用网络设备通常具备一定的动态节能能力。但为了应对随时可能发生的网络流量,设备处于"随时待命"状态,大部分模块无法深度休眠,节能收益通常小于15%[2]。2)网络级节能技术,该技术一般通过节能路由算法的手段使网络中部分网元(如链路端口和板卡等)不承载流量,从而网元可进入深度节能状态,降低了整网能耗;或者通过调节链路速率,达到节能的目标。

考虑基于链路关断的节能策略时,通常会涉及到一些问题:1)关断部分链路后所得到的新拓扑,必须能承担当前的业务量,即在当前业务量需求以及路由策略下,该拓扑中任一链路的利用率都不应超过一定的阈值 η;2)由于线卡能耗远大于链路能耗[3],采用怎样的链路关断策略才能使关断的线卡数目最多,以便最大化地节能;3)如何在连续时间的节能策略中减少链路的状态切换,从而更好地保障网络的稳定性;4)怎样设计算法,以使算法的时间复杂度尽可能低,从而适用于大规模网络。

在解决上述问题的过程中,如何找到折衷方案,并使节能效果尽可能好,是本文研究的重点所在。

本文第1节介绍网络节能的研究动态与发展现状;第2节介绍基于链路排序和业务量重路由的启发式算法;第3节为算法仿真及结果分析;最后总结全文。

1 研究动态与发展现状

目前,对节能的研究主要着眼于两个方面,即数据中心与一般网络。数据中心由于其结构的特殊性,多采用设备级的

到稿日期:2016-05-31 返修日期:2016-10-18 本文受华为网络按需用电项目资助。
黄 鸿(1992—),女,硕士生,主要研究方向为网络虚拟化、绿色网络,E-mail:flyaway0821@qq.com;虞红芳(1975—),女,博士,教授,主要研究方向为网络虚拟化、软件定义网络、云计算、绿色网络、复杂网络。

节能技术，如光学技术、能耗监测模块等[4-7]，这些技术缺乏在一般网络环境下的实用性。研究者们通过分析[8]发现，一般网络中的节能主要依靠网络级的节能策略来实现。所谓网络级节能策略，主要指的是对网络设备（如链路、接口、路由器、交换机等）进行合理的调度或者使用，以使这些网络设备的使用效率尽可能大，减少设备的闲置状态。这些节能策略大体可以分为两类。

（1）基于线性规划模型的节能算法。该类方法的基本思想是根据拓扑和业务量矩阵以及设备中链路和线卡的能耗，建立线性规划模型，求出网络中所有链路和线卡的状态配置，使全网的能耗最低。文献[9]提出的 greenTE 算法以及文献[10]提出的 GAMS Based Optimization 算法就是其中的代表，其主要思想是给出网络的拓扑和业务量矩阵，找到一个在满足链路利用率及包延时的性能约束条件和基本假设下，通过关闭链路和线卡能够节省最多能耗的路由方案。该类算法节能效果佳，但同时也存在时间复杂度高等不足。

（2）启发式节能算法。启发式算法主要用于解决优化问题复杂度高的弊端。文献[11]提出利用遗传算法以及模拟退火算法对网络中的链路不断进行关断尝试的方法，该方法随着迭代次数的增加，时间复杂度剧增，对链路的关断选择较为随机，没有考虑线卡的能耗，节能效果一般。文献[12]提出了 ESACON 算法，该算法对网络中的链路先按照拓扑重要度进行排序，然后按序选择链路进行关断尝试。ESACON 算法没有考虑链路利用率，可能导致关断链路后的网络无法承载当前业务量，同时也没有考虑线卡能耗，节能效果不佳。文献[13]提出了 AGHCS 算法，该算法根据链路利用率进行排序，从利用率低的链路开始依次进行关断判决。该算法的优点是时间复杂度低，能满足链路利用率约束，但同样未考虑到线卡能耗，节能效率一般。文献[14]提出了 HDEER 分布式路由机制，使网络中各链路承载不同的流量，调整链路速率，从而达到节能。该方法只适用于链路速率可改变的网络设备，而且算法收敛时间较长，不利于 QoS 的保障。文献[15]提出了 MILP-EWO 算法，其通过调整链路权重，使部分链路被关断，以达到节能。该算法易于实现，但没有考虑线卡的能耗，而且链路权重的频繁改变对于依赖 OSPF 协议的网络来说，同样不利于 QoS 的保障。

总的来说，基于线性规划模型的节能方法，时间复杂度高，依赖线性规划问题求解器，不容易应用于实际网络。而启发式节能算法，由于追求较低的时间复杂度，降低了对能耗模型描述的精确性，比如单纯考虑链路能耗，未考虑到线卡能耗，这样会降低算法的节能效率。其中，分布式的启发式节能算法更会使得 QoS 较难得到保障。

特别要指出的是，除了上述缺点，这两类方法通常都忽略了对策略平滑性的考虑。以 greenTE 算法为例，在 greenTE 节能算法中，针对每个时间片的业务量矩阵都能得到一个节能策略，但每个时间片下的节能策略是相互独立的，可能发生连续的两个时间片下的业务量矩阵经过 greenTE 节能算法计算后得到的节能策略却大大不同的情况，造成部分链路频繁地关闭和重启。表 1 列出了其在实际运用中典型时间片下的策略切换方式。

表 1　greenTE 在典型时间片下的切换方式

时间片编号	715	716	717	718
关闭链路编号	1,2,3,8,9	0,1,5,8	1,2,3,7,9	0,1,5,8

从表 1 可以看到，在短短的 4 个时间片下，greenTE 节能算法几乎每次都要调整节能策略，然而关断的链路数基本相同，带来的节能能耗相差不多。

各个时间片下的节能策略频繁地切换情景是不愿看到的，因为网络设备的关闭与重启可能需要很长的时间，例如一个板块的重启时间可能长达几分钟；同时，频繁的策略切换也会导致网络中路由策略的频繁切换，进而影响服务的稳定性。所以，网络设备频繁的关闭和重启，一方面大大降低了网络的性能，另一方面对设备本身的寿命也会造成很大的影响。因此，我们需要考虑策略的平滑性，即在不影响节能效率的情况下使相邻时间片下的节能决策尽可能地一致。

2　基于链路排序和业务量重路由的启发式算法

2.1　研究背景

如前文所述，致力于网络级节能的研究者们，通过对网络性能的评估分析，提出了一些较有意义的算法。其中，greenTE[9]作为建立能耗模型，用线性规划方法求解最优化解决方案的代表，具有很好的节能效果，但随之而来的时间复杂度高以及策略平滑性不佳的问题，却成为它投入实际使用的最大障碍。

而与之相对的启发式算法，则主要解决的是节能算法时间复杂度高的问题。例如 AGHCS 算法[13]，它的时间复杂度低，但没有考虑线卡能耗，就节能效果而言，是低于 greenTE 的，而且同样在策略平滑性上表现不佳。

本文提出的基于链路排序和业务量重路由的启发式算法，则对这两类节能算法存在的弊端做出了如下改进。

（1）较一般启发式算法而言，考虑了线卡能耗模型，更大程度地发掘节能潜力，同时保持较低的时间复杂度。

（2）保证较好的策略平滑性，即考虑连续时间多次执行节能算法后，尽可能保证链路的开启/关闭状态切换频率小。

2.2　算法核心思想

基于链路排序和业务量重路由的启发式算法（Turn Off Link and Traffic Re-Routing Algorithm，TOL-TRA）的核心思想是：根据当前链路利用率、链路所属线卡利用率、链路的拓扑关键性对链路进行综合排序，每次选择排序最优的链路，对该链路进行关断与否的判决，若成功关断，则将其承载的业务量迁移到剩余网络中，并更新所有源、目节点（源节点与目的节点）的路由；若未成功关断，则按序选择次优链路进行判断，直到所有排序链路都被遍历，算法结束，从而得到最终的拓扑以及路由信息。

2.3　算法描述

TOL_TRA 算法主要包括以下两个部分：

（1）链路排序；

（2）进行关断判决以及业务量重路由。

以下对各步骤进行详述：

（1）链路排序

该步骤是整个 TOL_TRA 算法的关键，本算法所设计的

排序方法是算法策略平滑性能以及节能效率的重要保障。下面将对排序方法进行详细描述，同时解释排序过程中的每个步骤，在提高策略平滑性能以及节能效率上所起到的作用。

1)根据各链路的 $\tau(l)$ 和 $U(l)$ 值，将当前网络中各链路分为两组——P(Prior)组与 I(Inferior)组。具体做法是，将 $U(l)\leqslant U_T$ 且 $\tau(l)\leqslant\tau\_th$ 的链路放入 P 组，其余链路放入 I 组。其中，$U(l)$ 表示链路 l 的链路利用率，$\tau(l)$ 表征的是链路 l 的拓扑重要度，值越大说明链路 l 的拓扑重要度越大，其定义由式(1)给出。这个步骤的目的是将链路初步分为两个优先级，P 组中的链路具有利用率低以及拓扑重要性低的特点，应该作为优先考虑关断的链路。I 组中的链路利用率较高，且拓扑重要性较高，考虑到关断 I 组链路可能会造成较高的代价，不应将 I 组链路作为优先考虑的对象，因此排序在 P 组之后。U_T 以及 τ_th 是人为设定的参数，根据仿真效果，一般取 U_T 为 0.3，τ_th 为 0.9。

$$\tau(l)=(\tau(G)-\tau(G-l))/\tau(G) \tag{1}$$

其中，$\tau(G)$ 和 $\tau(G-l)$ 分别指图 G 和图 $G-l$ 的生成树棵数。

2)设置 8 个有序排列的容器，顺序为 A,B,C,D,a,b,c,d。设置容器的目的在 3)中解释。

3)对 P,I 两组中的链路进行分类，P 组链路分别装入 A,B,C,D 4 个容器中，I 组链路分别装入 a,b,c,d 4 个容器中，即 A,B,C,D 容器是对 P 组链路的分类，a,b,c,d 容器是对 I 组链路的分类。具体的装入规则如表 2 所列，以下将根据表 2 对各容器的性质进行解释。

• A 容器中链路的性质：①在上一次节能策略中曾经被关断。②其连接的线卡目前利用率不为 1，即目前线卡上已有部分链路被判断为关断，优先考虑 A 容器中的链路，则有更大的可能使该线卡利用率降到 0，从而关断线卡。因此，为了追求策略平滑性，减少链路状态切换，并且尽可能对线卡进行关断，优先对 A 容器中的链路进行关断判决是合理的，这也是设置 A 容器的理由。

• B 容器中链路的性质：①在上一次节能策略中曾经被关断。②其连接的线卡目前利用率为 1，即目前线卡上尚未有链路被判断为可关断。B 容器优先级低于 A 容器，是因为关断 B 容器中的链路并没有使关断线卡的可能性变大。但考虑 B 容器中的链路仍然可以在一定程度上保障策略的平滑性，把 B 容器中的链路优先级排在了第二位。

• C 容器中链路的性质：①在上一次节能策略中未被关断。②其连接的线卡目前利用率不为 1。TOL_TRA 算法将策略平滑性优先于节能效率考虑，因此，在上一次节能策略中未被关断的链路，其优先级一定是在 A 和 B 容器链路之后的。但又因为考虑关断 C 容器中的链路可以使关断线卡的可能性增加，所以将其排序在 P 组链路中的倒数第二位。

• D 容器中链路的性质：①在上一次节能策略中未被关断。②其连接的线卡目前利用率为 1。可看出，关断 D 容器中的链路既不能提高策略平滑性，又不会增加线卡被关断的可能，因此在 P 组链路中排位最后。

剩余的 a,b,c,d 容器是对 I 组链路的分类，分类原则以及意义与 A,B,C,D 容器相同，不再赘述，只是因为 I 组链路的整体优先级低于 P 组链路的，所以 a,b,c,d 容器总体排在 A,B,C,D 组容器之后。

表 2 链路的分类规则表

链路分组	是否在上一次策略中被关断	当前对应线卡利用率是否为 1	装入容器
P 组	是	否	A
	是	是	B
	否	否	C
	否	是	D
I 组	是	否	a
	是	是	b
	否	否	c
	否	是	d

4)经过以上步骤，网络中的链路被大体划分到了 8 个不同优先级的分组中，但要选出候选链路，还需要找到优先级最高的那一条特定的链路，因此还需要更细致的排序。TOL_TRA 算法提供了细致的排序方案，描述如下：按照 A,B,C,D,a,b,c,d 的顺序，仅对第一个非空容器中的链路进行升序的排序。排序的依据是 $sort\_index(l)$ 取值的大小，其具体计算公式如式(2)所示：

$$sort\_index(l)=x\times U'(l)-y\times power(l) \tag{2}$$

其中，$U'(l)$ 是 $U(l)$ 的归一化结果；$power(l)$ 表示链路 l 的基础能耗；x,y 分别是对应的加权值，由仿真结果给出。要指出的是，式(2)中的 $power(l)$ 没有进行归一化处理，是因为 y 的取值可以控制最终结果 $y\times power(l)$ 与 $x\times U'(l)$ 是否是同一量级，这需要在测试结果中根据算法在不同量级下的性能表现决定，不必担心 $power(l)$ 与 $U'(l)$ 表面上量纲的不同。

另外，仅对第一个非空容器进行排序是为了减少不必要的计算，从而降低算法的时间复杂度。因为每次选择的候选链路都是第一个非空容器中排序首位的链路，所以其他容器中的链路暂时没有进行细致排序的必要。

经过以上步骤描述可以看出，经过容器划分和细致排序之后，所得到的优先级最高的链路，其对应的能耗较高、利用率较低且较能保证策略平滑性，优先对该链路进行关断尝试有利于策略平滑性以及节能效率。这就是 TOL_TRA 算法对链路进行排序的意义所在。

(2)候选链路的选择、判断及业务量重路由

1)选择 8 个容器中第一个已细致排序的非空容器中的首位链路作为候选链路 l^*，记 l^* 所在容器为 CONTAINER*，将 l^* 从容器 CONTAINER* 中弹出。

2)使用 JUDGE_LINK 算法判断 l^* 是否可以关断。JUDGE_LINK 算法重点考虑到了对时间复杂度的降低，通过在 $G-l^*$ 寻找 l^* 两端 k 短路的方式，找到可以承载 l^* 上业务量的可行路径族。这样，一方面可以使新路线经过的跳数尽可能少，减小对 QoS 的影响；另一方面在 k 较小的情况下，求 k 短路算法相较于其他寻找可行路的算法(比如最大流算法)，时间复杂度会更低。用 r_k 表示 l^* 两端的 k 短路，R_l^* 表示 l^* 上业务量重路由后的路径族(该路径族由 l^* 两端各条 k 短路组成，即可能有多条路径共同承担原 l^* 上的业务量)，记 f_l^* 为 l^* 上承载的业务量，p 和 q 分别为 l^* 两端点，S 为决定关断的链路集合，k_max 控制迭代次数，表示最多允许寻找的最短路数量，以免算法时间复杂度过高。JUDGE_LINK 算法如算法 1 所示。

算法 1　JUDGE_LINK(k_max)

```
1. Input:G−l*,l*,所有链路利用率
2.   k=1
3.   计算 p 到 q 的 k 短路 r_k
4.   计算满足链路利用率约束,能被 r_k 承载的最大流量 f_max
5. while(k≤k_max) do
6.   if(f_l* ≥f_max) then
7.     f_l* =f_l* −f_max
8.     将路径 r_k 加入路径族 R_l* 中
9.     k=k+1
10.  else do
11.    将 r_k 加入路径族 R_l* 中
12.    将 l* 加入集合 S
13.    判断 l* 为可以被关断
14.      break
15.    end if
16. end while
17. if(k>k_max) then
18.    判断 l* 为不可以关断
19. end if
20. Output:l* 关断与否的结果以及对应的 R_l*
```

3)若 JUDGE_LINK 算法判断 l^* 可以被关断，根据R_l^* 更新路由表以及各链路利用率 $U(l)$，更新拓扑为 $G=G-l^*$，转到步骤(1)。若判断 l^* 不可以被关断，则转到步骤(2)4)。

4)取当前第一个非空已排序容器的首位链路为 l^*，若第一个非空容器未排序，则先进行排序，再选取其首位链路作为 l^*，然后转到步骤(2)2)。若不存在任何非空容器，则 TOL_TRA 算法结束，关断 S 中的链路以及随后利用率变为 0 的线卡。

2.4　算法流程

TOL_TRA 算法的完整流程描述如下。

输入：网络拓扑、流量矩阵、链路最大利用率阈值。

步骤 1　所有链路分类进各容器，进行细致排序，选择容器中排序第一的链路为候选链路，并将该链路弹出容器。

步骤 2　用 JUDGE_LINK 函数判断在最大链路利用率约束下候选链路是否能被关断。

步骤 3　如果候选链路能够被关断，则将该链路加入到集合 S 中，更新网络拓扑、各链路利用率、各路由表，然后转到步骤 1；否则转到步骤 4。

步骤 4　如果容器中的所有链路都已经过判断，则算法终止；否则，选择已排序容器中的下一条未判断链路作为候选链路，并转到步骤 2。

输出：全体可关断链路集合 S 以及关断 S 中链路后利用率降为 0 的线卡。

2.5　算法时间复杂度分析

TOL_TRA 算法是一种启发式算法，可以在多项式时间内完成如 greenTE 模型所述节能问题的近似求解。其算法复杂度的分析如下。

首先，用雅克比法计算 $\tau(l)$，其时间复杂度为 $O(n^2)$，但此步骤由于是预处理步骤，在 TOL_TRA 算法中只执行一次，因此可不计入算法时间复杂度内。

在 TOL_TRA 算法中，最复杂的情况是所有链路都在同一个容器中，每次取出此容器中排序后的队首链路都能被关闭。队首链路被关闭后都需要对这个容器重新进行排序，反复执行此步骤，并最多关闭 $|E|-|V|+1$ 条链路，其中 $|E|$ 是 G 中的边数，$|V|$ 是 G 的节点数。

在尝试关闭第一条链路前，排序所需要的时间复杂度为 $O(|E|\log|E|)$，判断此链路是否能关闭的复杂度为 $O(\log|V|)$；在尝试关闭第二条链路前，重新排序所需要的时间复杂度为 $O((|E|-1)\log(|E|-1))$，判断此链路是否能关闭的复杂度为 $O(\log|V|)$；在尝试关闭最后一条链路前，重新排序所需要的时间复杂度为 $O((|E|-|V|+1)\log(|E|-|V|+1))$，判断此链路是否能关闭的复杂度为 $O(\log|V|)$。所以，所需要的总时间复杂度由式(3)给出：

$$T(n)=O(|E|\log|E|)+O((|E|-1)\log(|E|-1))+O(\log|V|)+\cdots+O((|E|-|V|+1)\log(|E|-|V|+1))+O(\log|V|) \tag{3}$$

化简后为：$O(|V||E|\log|E|)$。

3　算法仿真及结果分析

3.1　仿真环境介绍

TOL_TRA 算法的测试网络拓扑如图 1 所示，仿真使用的流量文件是 Internet2 的美国骨干网络中一周内的流量记录，每 5 分钟一个时间片，每个时间片一个流量文件，共 2016 个流量文件。考虑到仿真效率，本文选取前 1000 个流量文件用于仿真。仿真使用的拓扑文件和流量文件均可在文献[16]中获得。使用 C++以及 STL 库函数来实现 TOL_TRA 以及 AGHCS 算法，运行环境为 vs2010。greenTE 算法利用了 GLPK 线性规划求解器对问题模型进行求解。

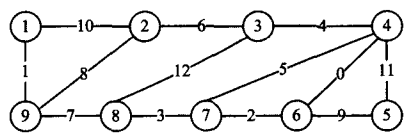

图 1　仿真测试拓扑示意图

3.2　排序依据中 *sort_index*(l)的参数设置

TOL_TRA 节能算法根据 $sort_index(l)$对第一个非空容器中的所有链路进行排序，链路排序的结果将在很大程度上影响最终的节能效率。在 $sort_index(l)$中，最关键的就是分析链路利用率和链路能耗之间的关系，我们希望链路利用率比较低的链路优先被考虑，也希望链路能耗较高的链路优先被考虑。所以分能耗优先(Energy_prior)、利用率和能耗等价(EU_equ)以及利用率优先(Util_prior) 3 种情况来分析。为保证能耗优先，将保证 $y\times power(l)$的数量级大于$x\times U'(l)$，实验中取 $x=1$，$y=0.1$；同理，为保证利用率和能耗等价，将保证 $y\times power(l)$的数量级等于 $x\times U'(l)$，实验中取 $x=10$，$y=0.05$；为保证利用率优先，将保证 $y\times power(l)$的数量级小于 $x\times U'(l)$，实验中取 $x=10$，$y=0.001$。

在网络节能的场景下，节能算法的优劣性评价最主要的依据就是它们的节能效率。在节能效率的分析中，将对各个算法的节能效率进行比较。为了方便比较，使用式(4)表示归一化的节能百分比：

$$saving\_pct=\frac{\sum_{l\in E}P_l x_l+\sum_{lc\in M}P_{lc}y_{lc}}{\sum_{l\in E}P_l+\sum_{lc\in M}P_{lc}} \tag{4}$$

其中，M 表示线卡集合。当该线卡上的所有链路都关闭后，即线卡利用率为 0 时，该线卡即可关闭。x_l 表示链路 l 的开闭情况，$x_l=1$ 时链路 l 关闭，$x_l=0$ 时链路 l 开启。y_{lc} 表示线卡 lc 的开闭情况，$y_{lc}=1$ 时线卡 lc 关闭，$y_{lc}=0$ 时线卡 lc 开启。

图 2 示出了在 3 种情况下的 TOL_TRA 节能算法的节能百分比的对比情况。

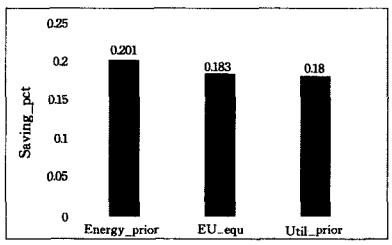

图 2 3 种情况下 TOL_TRA 算法的节能百分比的对比示意图

从图 3 可以看出，在能耗优先的情况下，节能百分比达 20.1%，而利用率和能耗等价以及利用率优先的情况下节能百分比要低于能耗优先的情况，分别仅为 18.3% 以及 18.0%。所以，通过实验可以得知，在排序依据 $sort\_index(l)$ 中，能耗的优先级应该高于链路利用率的优先级。

考查完 3 种情况下的能耗表现，再来考查 3 种情况下的节能策略切换频率表现，即策略的平滑性。在大规模网络节能分析中，我们希望能够快速获得节能策略，同时还希望各时间片之间的节能策略不要过于频繁切换。为了衡量各个算法的节能策略切换频率，定义切换频率函数如式(5)所示：

$$switching\_freq_i=|p_i-p_{i+1}|,i\in T \tag{5}$$

其中，p_i 表示在时间片 i 下的节能策略，$|p_i-p_{i+1}|$ 则为相邻的两个时间片节能策略的差别。例如，假设时间片 i 下关闭 1,2,3,4 号链路，时间片 $i+1$ 下关闭 5,6,7,8 号链路，则两个时间片下的节能策略差别为 8。

图 3 反映了 TOL_TRA 算法在 3 种情况下的策略切换频率变化过程，各时间点对应直线的幅度代表了 $switching\_freq_i$ 的大小。从图 3 中可以看到，3 种情况下节能策略切换频率变化类似，只有在少数时间片下策略切换频率会有不同。下面将量化 3 种情况下的节能策略切换频率。

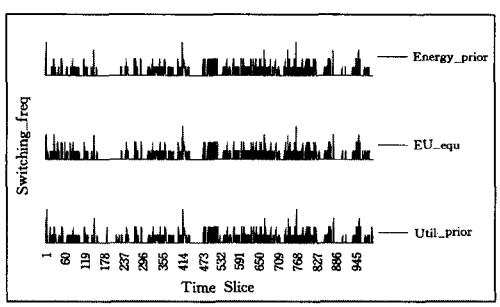

图 3 3 种情况下策略切换频率变化过程示意图

图 4 示出了 3 种情况下的平均切换频率值对比，可以看出，3 种情况下的平均策略切换频率基本相等，最大的差距也只有 0.012。

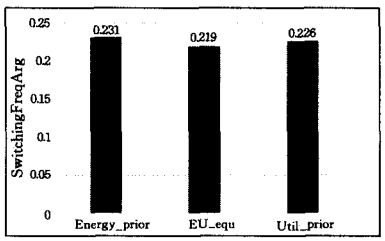

图 4 3 种情况下的平均切换频率值对比示意图

图 5 显示了 3 种情况下的策略切换频率方差。从图中可以看出，虽然在利用率和能耗等价的情况下，策略切换方差要稍低于其他两种情况，但其实 3 种情况下的策略切换方差差距并不大，最大的方差差距也只有 0.05 左右。

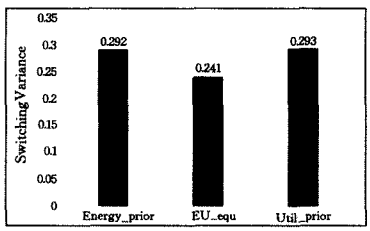

图 5 3 种情况下的策略切换频率方差对比示意图

综上，可以得知，能耗优先的情况下的节能效率高于其他两种情况，而 3 种情况下的策略切换频率无较大差距。

造成上述差异的原因是，当相邻时间片的业务量矩阵差别较较大时，即不得不改变节能策略时，第一个非空容器中的排序就很重要，此时的排序也会影响到下一个时间片的节能策略的制定，3 种不同的参数设置会导致不同的策略出现；而当相邻时间片的业务量矩阵差距不大时，计算此时间片的节能策略时，虽然 3 种不同的参数设置会导致第一个非空容器中的排序不同，但是由于较上一个时间片业务量矩阵无太大变化，第一个非空容器中的链路集合仍然是优先可以删除的，因此这个时间片下的节能策略与上一个时间片的节能策略相同。

这样就会导致，当相邻时间片的业务量矩阵相差特别大而不得不改变节能策略时，3 种不同的参数设置会得到不同的节能策略，不同的节能策略的节能效率是不同的，而变化后的节能效率是累加的，累加起来的节能效率就会有较大的差距。

比较以上测试结果后，排序依据的参数制定为能耗优先，将保证 $y\times power(l)$ 的数量级大于 $x\times U'(l)$，在后面的测试中将取 $x=1,y=0.1$。

3.3 TOL_TRA 与 AGHCS 以及 greenTE 算法的性能对比

本节将从节能效率和策略平滑度两个方面给出 TOL_TRA，AGHCS 以及 greenTE 算法的性能对比情况。

3.3.1 节能效率对比

图 6 示出了 TOL_TRA，AGHCS 以及 greenTE 算法的节能百分比对比示意图。

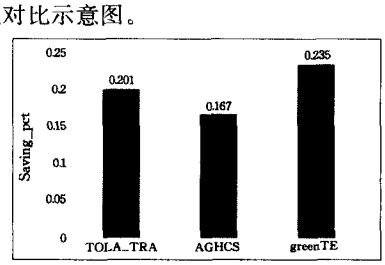

图 6 各算法节能百分比示意图

从图 6 中可以看到,greenTE 算法的平均节能效率为 0.235,TOL_TRA 算法的平均节能效率为 0.201,AGHCS 算法的平均节能效率为 0.167。greenTE 算法节能效率最优;其次是 TOL_TRA 算法,节能效率可达到 greenTE 节能算法的 85.5%;而一般的启发式算法 AGHCS 得到的节能百分比与 TOL_TRA 算法得到的节能百分比有一定差距。因此,在节能效率上,TOL_TRA 算法得到的节能效率与 greenTE 的节能效率相差不算太多,其节能性能是比较优秀的。

3.3.2　节能策略平滑性对比

图 7 示出了 TOL_TRA,AGHCS 与 greenTE 算法的节能策略切换频率变化的过程。从图中可以明显看出,在 1000 个时间片下,greenTE 算法得到的节能策略切换很频繁,而且峰值很高,说明 greenTE 节能算法不仅在各个时间片下的策略切换次数多,而且各个时间片之间的策略相差巨大。AGHCS 节能算法得到的节能策略切换频率虽然峰值不高,但在较长时间内都有保持在一个较为频繁的水平上,说明 AGHCS 节能算法虽然在相邻时间片之间的策略没有相差过大,但在各时间片下都保持着较多的策略切换次数。反观 TOL_TRA 节能算法,其相邻时间片下得到的节能策略相差很小,且切换频率并不频繁。

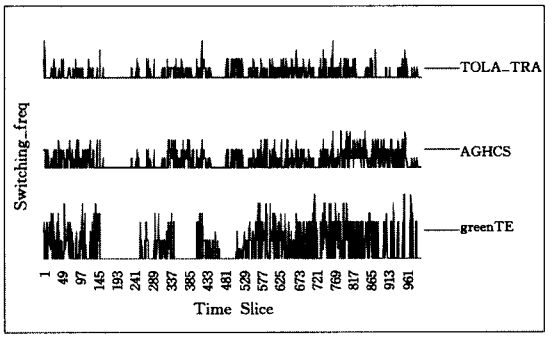

图 7　各算法策略切换频率变化过程示意图

下面将量化分析上述 3 个节能算法的切换频率。图 8 示出了 TOL_TRA,AGHCS 和 greenTE 算法的平均切换频率值对比。

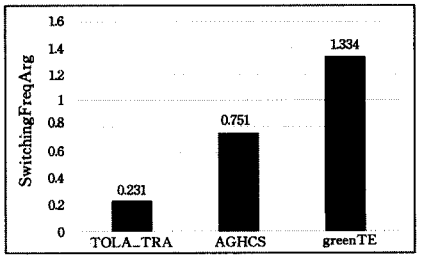

图 8　各算法平均切换频率值对比示意图

从图 8 中可以看到,作为参照的 greenTE 节能算法的平均策略切换频率为 1.334,AGHCS 节能算法的平均策略切换频率为 0.751,TOL_TRA 节能算法的平均策略切换频率为 0.231。3 种算法中,TOL_TRA 算法的平均切换频率远低于其他两种算法。

图 9 示出了 TOL_TRA,AGHCS 和 greenTE 算法的切换频率方差对比。从图中可以看到,作为参照的 greenTE 节能算法的策略切换频率方差为 3.281,AGHCS 节能算法的策略切换频率方差为 0.976,TOL_TRA 节能算法的策略切换频率方差为 0.292。同样地,TOL_TRA 算法的策略切换频率方差远低于其他两种算法,说明 TOL_TRA 算法在相邻时间片上的策略切换相差情况最小,具有较好的策略平滑性。

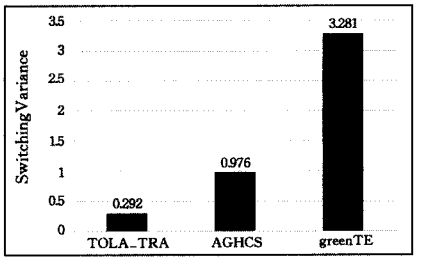

图 9　各算法切换频率方差对比示意图

由上述结果可以看出,不仅 TOL_TRA 节能算法的平均策略切换频率远低于 greenTE 节能算法,而且其方差也远低于 greenTE 节能算法。在保证了节能效率的同时大大降低了节能策略切换频率,让节能策略变得平滑,使得网络设备不需要频繁地关闭和重启,一方面大大提升了网络的性能,另一方面也大大延长了设备本身的寿命。而同样作为启发式算法的 AGHCS 算法,在节能效率和平滑度两方面的性能表现都不及 TOL_TRA 算法优秀。

结束语　本文系统地论述了一种时间复杂度较低的基于链路排序和业务量重路由的启发式节能算法(TOL_TRA—Turn Off Link and Traffic Re-Routing algorithm),具体包括以下工作:

1)论述分析了现有的网络级节能算法的优势与劣势。

2)提出了一种时间复杂度小、策略平滑度好,且节能效率较高的节能算法——TOL_TRA。详细描述了其功能实现,并分析了其时间复杂度。

3)对 TOL_TRA 进行了软件仿真以及性能分析。分析了 TOL_TRA 算法中各参数取值对于算法性能的影响,对比了 TOL_TRA 与 greenTE 以及 AGHCS 几种算法在节能效率、节能策略平滑性两个方面的性能表现。

最后得到结论,TOL_TRA 算法是一种在多项式时间内求节能优化问题近似最优解的启发式算法,其节能效率较高、节能策略平滑性较好,在实际网络中有较高使用价值。

参 考 文 献

[1] BOLLA R,BRUSCHI R,CARREGA A,et al. Cutting the energy bills of internet service providers and telecomsthrough power management:an impact analysis[J]. Computer Networks,2012,56(10):2320-2342.

[2] PLAN G A. An Inefficient Truth. Global Action Plan Report [R/OL]. http://globalactionplan. org. uk.

[3] NEDEVSCHI S,POPA L,IANNACCONE G,et al. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation [C]//Proceedings of USENIX NSDI. 2008.

[4] White paper:cisco energy management:a case study in implementing energy as a service[OL]http:www. cisco. com/en/US/

prod/collateral/…/ CiscoEMSWhitePaper. pdf.

[5] WANG H, GARG A, BERGMAN K. Design and demonstration of an all-optical hybrid packet and circuit switched network platform for next generation data centers[C]// OFC 2010, OTuP3. 2010: 1-3.

[6] SINGLA A, SINGH A, Ramachandran K, et al. Feasibility study on topology malleable data center networks (DCN) using optical switching technologies[C]// Optical Fiber Communication Conference and Exposition (OFC) and the National Fiber Optic Engineers Conference(NFOEC). 2011: 1-3.

[7] BALIGA J, HINTON K, TUCKER R S. Energy Consumption of the Internet[C]// Proc. Joint International Conference on Optical Internet and the 32nd Australian Conference on Optical Fibre Technology (COIN-ACOFT 2007). Melbourne, Australia, 2007: 1-3.

[8] GAO S, ZHOU J, AYA T, et al. Reducing network power consumption using dynamic link metric method and power off links [C]// IEICE Communications. 2009.

[9] ZHANG M, YI C, LIU B, et al. GreenTE: Power-aware traffic engineering[C]// 2010 18th IEEE International Conference on Network Protocols (ICNP). Kyoto, 2010: 21-30.

[10] CHABAREK J, SOMMERS J, BARFORD P, et al. Power awareness in network design and routing[C]// Proc. IEEE INFOCOM. April 2008: 457-465.

[11] TSENG P K, CHUNG W H. Near optimal link on/off scheduling and weight assignment for minimizing IP network energy consumption [J]. Computer Communications, 2012, 35 (6): 729-737.

[12] CUOMO F, ABBAGNALE A, CIANFRANI A, et al. Keeping the connectivity and saving the energy in the internet[C]// 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Shanghai, China, 2011: 319-324.

[13] WANG N, MICHAEL C, HO K H. Disruption-Free Green Traffic Engineering with NotVia Fast Reroute[J]. IEEE Communications Letters, 2011, 15(10): 1123-1125.

[14] ZHOU B, ZHANG F, WANG L, et al. HDEER: A Distributed Routing Scheme for Energy-Efficient Networking [J]. IEEE Journal on Selected Areas in Communications, 2016, 34 (5): 1713-1727.

[15] AMALDI E, CAPONE A, GIANOLI L G. Energy-aware IP traffic engineering with shortest path routing [J]. Computer Networks, 2013, 57(6): 1503-1517.

[16] Network Group at UESTC. topo_tm_files[OL]. https://bitbucket. org/netlab-uestc/greentm/src.

(上接第 35 页)

行时间，分析了 Spark 系统在不同缓存策略下的性能。DWRP 在内存不足时的性能略低于 LRU 策略，这是因为 DWRP 策略需要考虑 RDD 的分布式特征，在选择淘汰 RDD 分区时需要进行通信，会产生网络通信开销。随着网络性能的不断提高和网络通信延迟的降低，DWRP 策略将充分发挥其优势，提高 Spark 系统的性能。

参 考 文 献

[1] ZAHARIA M, CHOWDHURY M, FRANKLINM J, et al. Spark: cluster computing with working sets[C]// Usenix Conference on Hot Topics in Cloud Computing. 2010: 10.

[2] WARNEKE D, LENG C. A Case For Dynamic Memory Partitioning in Data Centers[C]// The Workshop on Data Analytics in the Cloud. 2013: 41-45.

[3] LI H, GHODSI A, ZAHARIA M, et al. Tachyon: Reliable, memory speed storage for cluster computing frameworks[C]// Proceedings of the ACM Symposium on Cloud Computing. ACM, 2014: 1-15.

[4] ANANTHANARAYANAN G, GHODSI A, WANG A, et al. PACMan: coordinated memory caching for parallel jobs[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2012: 20.

[5] DUAN M, LI K, TANG Z, et al. Selection and replacement algorithms for memory performance improvement in Spark[J]. Concurrency and Computation: Practice and Experience, 2015, 28 (8): 2473-2486.

[6] FENG L. Research and Implementation of Memory Optimization Based on Parallel Computing Engine Spark[D]. Beijing: Tsinghua University, 2013. (in Chinese)
冯琳. 集群计算引擎 Spark 中的内存优化研究与实现[D]. 北京：清华大学，2013.

[7] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing: UCB/EECS-2011-82[R]. EECS Department, University of California, Berkeley, 2011.

[8] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2.

[9] GRISHCHENKO A. Spark Architecture: Shuffle [EB/OL]. (2015-08)[2016-09]. https://0x0fff. com/spark-architecture-shuffle.

[10] WHITE T. Hadoop: The Definitive Guide, 3E. [M]. California: O'Reilly Medis, 2012: 226-227.

[11] WANG L, ZHAN J, LUO C, et al. Bigdatabench: A big data benchmark suite from internet services[C]// 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2014: 488-499.

[12] GAO Y J. DataProcessing with Spark, Technology, Application and Performance Optimization [J]. Beijing: China Machine Press, 2014: 38-39. (in Chinese)
高彦杰. Spark 大数据处理技术，应用与性能优化[M]. 北京：机械工业出版社，2014: 38-39.