

文章编号:1001-9081(2010)06-1447-04

针对 Ad Hoc 网络组播组发现的蚁群算法

原萍¹, 海龙²

(1. 上海工程技术大学 城市轨道交通学院, 上海 201620; 2. 东北大学 信息科学与工程学院, 沈阳 110819)

(pingyuan211@126.com; hl850121@163.com)

摘要:针对将蚁群算法应用于 Ad Hoc 网络组播寻路中存在无法同时找到多目标的局限性,提出了一种采用逆向寻路的解决方案。当前进蚂蚁在到达接收端时,会自动复制出若干个返回蚂蚁并进行回溯,而这些返回的蚂蚁并不是按原路返回,而是进行反向的寻路,同时原前进蚂蚁将继续寻找其他多目标并进行相同的操作。仿真结果与原始蚁群算法进行了比较,可以发现在延迟、带宽消耗、发包数上逆向蚁群算法要优于原始蚁群算法。仿真实验表明,改进的蚁群算法减少了为寻找多目标所造成的延迟,并且提高了算法的收敛速度。

关键词:组播路由;移动自组织网络;蚁群算法;前进蚂蚁;返回蚂蚁

中图分类号: TP393; TP18 **文献标志码:** A

ACO algorithm for discovery of multicast group in Ad Hoc network

YUAN Ping¹, HAI Long²

(1. College of Urban Railway Transportation, Shanghai University of Engineering Science, Shanghai 201620, China;

2. College of Information Science and Engineering, Northeastern University, Shenyang Liaoning 110819, China)

Abstract: Ant colony algorithm applied to Ad Hoc network multicast routing has its own limitation that multi-objective could not be found at the same time. With regard to the limitation, an improved scheme called Contrary Ant Colony Optimization (CACO) routing algorithm was proposed. Some backward ants would be copied to find the routing from the contrary direction when a forward ant reached a destination node. After that the forward ant continued to find other multicast destinations with the same operation. The simulation results were compared with that of the original ant colony algorithm. Delay, overhead and packet number of CACO were better than ACO. The results indicate that CACO reduce the delay of finding multi-objective and enhance the convergence rate of the ant colony algorithm.

Key words: multi-objective; Mobile Ad Hoc Network (MANET); ant colony algorithm; forward ant; backward ant

0 引言

移动自组织网络(Mobile Ad Hoc Network, MANET)是一种无线自组织网络。节点的移动性,网络拓扑的频繁变化,节点的能源消耗使得 MANET 的通信方式与传统的通信方式有着明显的不同。从节点的信息状态,到路由发现过程,再到路由维护,MANET 都有它自身的特点^[1]。考虑到以上 MANET 的特点,如何能够在需要通信的时候,及时发现一条满足该通信需求的路由,就成为了 MANET 亟须解决的问题之一。

在 MANET 的路由发现过程中,传统的 MANET 应用洪泛的方法来寻找路由^[2]。洪泛算法是最简单,寻路最全面,收敛速度最快的算法。但是在 MANET 中,洪泛算法也存在很大的缺陷:1)多余的重播。当一个移动节点决定将一条广播消息重播给自己相邻的节点的时候,所有这些相邻节点却已经有了该条广播消息。2)竞争。在一个移动节点广播一条消息后,加入其多个相邻节点决定重播该条消息,那么这些发送可能产生严重的相互竞争。3)碰撞。由于没有采用退避机制,缺乏 RTS/CTS 控制分组对话,以及没有碰撞检测(Collision Detection, CD),所以很可能发生碰撞,导致更严重的损害^[3]。

近年来,许多人把智能算法应用到 MANET 中^[3-7]。例如:文献[4]把蚁群算法应用到 MANET 中。蚁群算法对网络

变化有很高的适应性,使用主动式路径收集,具有很强的鲁棒性,并能提供多路径路由,而且极大地减少了网络散播数据的负载^[4]。但是蚁群算法也存在收敛速度慢,容易陷入“早熟”和“停滞”状态的缺陷^[5]。针对蚁群算法的缺点,产生了许多蚁群算法的改进方案^[5-7]。例如,文献[5]通过 GPS 来定位目的节点的方向,并通过相关的算法来确定蚂蚁寻路的方向,从而使得蚂蚁在寻路初期不会因为方向问题而陷入“寻路障碍”之中。同样的,文献[6]也通过应用 GPS 来对移动节点定位,然后通过 ACO 算法来进行路由搜索。与文献[5]不同的在于,文献[6]应用 GPS 定位是为了对路径稳定性的评估,蚂蚁会根据路径稳定度高低留下不同量的信息素,这样做可以使得发现的路由稳定度较高,适合 MANET 的特点。但这两种方案都对外围设备要求较高,并且要求节点之间还要发送和存储位置信息数据,这就对网络中的带宽和节点能源要求较高。

人们对通信的要求不仅局限于一对一的通信,而更多的是进行一对多或多对多的通信。所以在 MANET 中组播路由协议是发展的一个重要方向。在 MANET 中组播协议的信息传输要通过维护一个组播分组来完成。在这样的协议中每一个节点都可以随时加入或离开这个分组。组播分组的维护方式包括基于树结构和基于网格结构两种^[8]。比如 MAODV^[9]和 ODMRP^[10],前者对每个多目标组设立一个组长来进行分

收稿日期:2009-12-03;修回日期:2010-01-20。 基金项目:上海市教育委员会重点学科建设项目(J51401)。

作者简介:原萍(1963-),女,辽宁沈阳人,教授,博士,主要研究方向:Ad Hoc 网络、卫星通信;海龙(1985-),男,辽宁沈阳人,硕士研究生,主要研究方向:Ad Hoc 网络。

组维护和管理,而后者则是直接通过源节点来进行分组的维护和管理^[11]。

近些年来,已经有人把智能算法应用到 MANET 的路由协议中^[12-14],并取得了很好的效果。通过把智能算法应用到组播路由协议中,可以减少网络带宽的消耗^[12],减少节点能源的消耗,以及增加路由的稳定性^[13]。但把蚁群算法应用到组播路由协议中还存在许多的问题。比如一个源节点进行多目标寻路,如果一只前进蚂蚁找到了多目标中的一个目标时,该蚂蚁就会变成返回蚂蚁按原路返回到源节点,并在返回的链路上留下信息素。而留下的信息素则会招来更多的前进蚂蚁到该目标节点来,这就会造成该目的节点在算法初期吸引了较多的前进蚂蚁,而其他的一些目的节点则被“冷落了”,从而极大地减慢了蚁群算法在多目标寻路上的收敛速度。

本文针对以上问题提出了逆向蚁群优化(Contrary Ant Colony Optimization, CACO)算法的方案。该方案对蚁群算法进行了改进,使其适合于应用在组播路由协议中。

1 蚁群算法的公式和符号表示

本文用常数 τ 来表示信息素。源节点发送出的前进蚂蚁只进行简单的寻路操作。而当前进蚂蚁到达目的节点时,就会变成返回蚂蚁并返回源节点,返回蚂蚁在返回的链路上留下信息素。 t_p 时刻,当一只返回蚂蚁从节点 R 到达节点 Q 时,它会在节点 Q 的路由表中增加节点 R 的信息素。设上一只返回蚂蚁从 R 到 Q 的时刻为 t_{p-1} ,则 t_p 时刻节点 Q 上关于 R 点的信息素为:

$$\tau_R(t_p) = \tau_p(t_{p-1}) + \tau$$

等式还可以写成:

$$\tau_R(t_p) = \tau_0 + n_{pR}\tau$$

其中 n_{pR} 表示从开始到 t_p 时刻从 R 点到达 Q 点的返回蚂蚁的数。返回蚂蚁的返回路线是固定的,它是前进蚂蚁在寻路时记下的一条从源节点到目的节点可行的路由。前进蚂蚁从节点 Q 向其邻节点寻路时,是通过选择概率进行下一跳节点的选择。设节点 Q 的邻节点有 m 个,则选择概率公式为:

$$p_{QR}(t_p) = \frac{\tau_R(t_p)}{\sum_{i=1}^m \tau_i(t_p)} = \frac{\tau_0 + n_{pR}\tau}{m\tau_0 + \tau \sum_{i=1}^m n_{pi}} \quad (3)$$

其中 n_{pi} 表示从开始到 p 时刻从 i 节点到达 Q 节点的返回蚂蚁数。设一个变量 $y_R(t_p)$,表示在 t_p 时刻是否有返回蚂蚁到达 Q 节点。它的取值为:

$$y_R(t_p) = \begin{cases} 1, & \text{有返回蚂蚁到达 } Q \text{ 节点} \\ 0, & \text{没有返回蚂蚁到达 } Q \text{ 节点} \end{cases} \quad (4)$$

则式(2)可变化为:

$$\tau_R(t_p) = \tau_0 + \sum_{i=1}^p y_R(t_i)\tau \quad (5)$$

选择概率公式为:

$$p_{QR}(t_p) = \frac{\tau_R(t_p)}{\sum_{i=1}^m \tau_i(t_p)} = \frac{\tau_0 + \sum_{i=1}^p y_R(t_i)\tau}{m\tau_0 + \tau \sum_{j=1}^m \sum_{i=1}^p y_j(t_i)} \quad (6)$$

在蚂蚁算法中还要有一个蒸发过程,蒸发过程是指信息素会随时间变化而不断减少。本文用 α ($0 < \alpha < 1$) 来表示蒸发系数,蒸发系数的设定将会影响寻路的情况。当蒸发系数较小时,表示信息素蒸发速度快,可以很快地得到可行的最优路径,算法的收敛速度较快,但是容易陷入局部最优;当蒸发

系数较大时,算法容易找到全局最优,但是算法的收敛速度会减慢。所以蒸发系数的设定要根据具体的通信环境而定。

把蒸发系数代入式(1)中,可得:

$$\tau'_R(t_p) = \alpha[\tau_R(t_{p-1}) + \tau] \quad (7)$$

则式(5)可变成:

$$\tau'_R(t_p) = \alpha^p \tau_0 + \sum_{i=1}^p y_R(t_i) \alpha^{p-i+1} \tau \quad (8)$$

则式(6)可变为:

$$p_{QR}(t_p) = \frac{\tau'_R(t_p)}{\sum_{i=1}^m \tau'_i(t_p)} = \frac{\alpha^p \tau_0 + \sum_{i=1}^p y_R(t_i) \alpha^{p-i+1} \tau}{m\alpha^p \tau_0 + \tau \sum_{j=1}^m \sum_{i=1}^p y_j \alpha^{p-i+1} (t_i)} \quad (9)$$

蒸发过程在算法的初期会影响到算法的收敛,所以本文对节点开始进行蒸发的时刻进行了设定。当节点 Q 收到从节点 R 返回的返回蚂蚁数量达到 b 个时,则节点 Q 路由表中节点 R 的信息素开始加入蒸发过程。设 q 时刻时,从节点 Q 到达节点 R 的返回蚂蚁数达到了 b 个,则这时的信息素公式可表示为:

$$W_{RQ}(t_p) = \tau_R(t_p) = \alpha^{p-q} \tau_0 + \sum_{i=1}^q y_R(t_i) \tau + \sum_{i=q}^p y_R(t_i) \alpha^{p-i+1} \tau \quad (10)$$

$$M_Q(t_p) = \sum_{i=1}^m W_{iQ}(t_p) = \alpha^{p-q} m\tau_0 + \sum_{j=1}^m \sum_{i=1}^q y_j(t_i) \tau + \sum_{i=q}^p y_j(t_i) \alpha^{p-i+1} \tau \quad (11)$$

其中: W_{RQ} 表示节点 Q 路由表中记录的节点 R 的信息素, M_Q 表示节点 Q 记录的所有邻节点信息素的总和。这时选择概率的等式可变为:

$$p_{QR}(t_p) = \frac{W_{RQ}(t_p)}{M_Q(t_p)} = \frac{\alpha^{p-q} \tau_0 + \sum_{i=1}^q y_R(t_i) \tau + \sum_{i=q}^p y_R(t_i) \alpha^{p-i+1} \tau}{\alpha^{p-q} m\tau_0 + \sum_{j=1}^m \sum_{i=1}^q y_j(t_i) \tau + \sum_{i=q}^p y_j(t_i) \alpha^{p-i+1} \tau} \quad (12)$$

在文献[3]中提到了阈值的概念,本文引入常量 Y 和 L_p 来表示一个阈值。当 $W_{RQ}(t_p) \geq Y$, 或者 $p_{QR}(t_p) \geq L_p$ ($L_p = Y/M_Q$) 时,认为节点 R 是节点 Q 的最优下一跳节点。阈值的引入说明本文中蚁群算法找到的路由可能不是理论上的最优路径,而是相对最优路径。阈值是根据具体的通信环境来进行设定的。

2 CACO 算法的描述

2.1 ACO 算法的改进方案

蚁群算法相对洪泛极大地节省了网络资源,但蚁群算法也有它的不足。由于发出的蚂蚁的数目有限,寻路初期蚁群算法容易陷入“早熟”、“停滞”现象。而且把蚁群算法应用到多播协议中时,由于蚁群算法寻路的特点,多目标很难同时被发现并找到最优路由。

本文针对蚁群算法的缺陷对蚁群算法进行了改进。首先

对蚂蚁算法的寻路方式进行了改进。原始蚁群算法的寻路方式是前进蚂蚁寻找到一条可行的路由时,前进蚂蚁就会变成返回蚂蚁,沿着已经被找到的路由返回源节点,并在返回的途中留下信息素。这样做在进行多目标寻路时会产生延迟。比如,当到达目的节点1的路由已经被发现时,还没有被前进蚂蚁到达过目的节点2。而由于目的节点1过早被发现会使得更多的前进蚂蚁来到目的节点1,目的节点2就会很难吸引到更多的前进蚂蚁。这样就会使目的节点1和目的节点2的路由发现过程产生很大的延迟。针对蚁群算法在多播协议中应用的缺陷,本文中前进蚂蚁找到一个目的节点时将不会返回(如图1所示),而是生成若干返回蚂蚁进行逆向的“单目标”寻路。

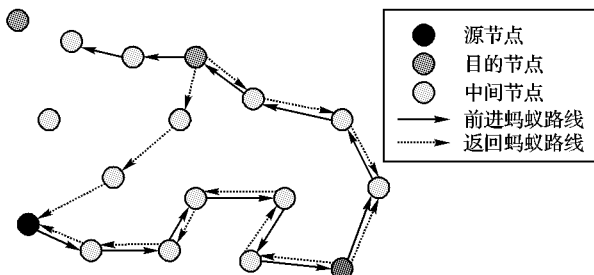


图1 CACO 算法的路由模型

前进蚂蚁所寻找的路由并不是每个目标节点的最优路由。因为前进蚂蚁到达一个目的节点时,并不能保证它是从源节点直接过来的(除了初次到达的目的节点以外)。这是很显然的,前进蚂蚁的寻路过程是要依次经过若干目的节点的。那么想要找到源节点到达每个目的节点的最优路由,逆向的单目标寻路就是必不可少的。这样就能保证能够找到源节点到每个目的节点的最优路由。返回蚂蚁的逆向寻路方式与传统蚁群算法相似,当逆向寻路的返回蚂蚁找到源节点时,会按原路(源节点到该目的节点)返回一个蚂蚁在该路径上留下信息素。

2.2 CACO 的算法过程

首先源节点 p 激活计时器并发出若干前进蚂蚁报文进行路由搜索,而后每收到一定数目的返回蚂蚁后就发送一个前进蚂蚁。当计时器为0时,表示此次路由搜索完成,停止发送蚂蚁,并开始发送数据。源节点发出的前进蚂蚁按照每个邻节点所拥有的信息素总和来进行节点选择概率的计算。选择概率计算公式如下(设有 m 个多目标, n 个邻节点):

$$P_{Q_i}(n_1) = \frac{\sum_{j=1}^n W_{ij}}{\sum_{i=1}^m \sum_{j=1}^n W_{ij}} \quad (13)$$

其中: $\sum_{j=1}^n W_{ij}$ 表示节点 Q 的邻节点 i 含有的所有信息素, $\sum_{i=1}^m \sum_{j=1}^n W_{ij}$ 表示节点 Q 的所有邻节点含有的信息素总和。这样设定可以使前进蚂蚁的寻找路线集中在到达每个目标节点都相对较近的线路上。比如目标节点大部分聚集在网络拓扑的某一边,通过以上的概率选择,可以使大部分前进蚂蚁集中到目的节点较多的那边去寻找目的节点。

前进蚂蚁的报文格式类似于 ODMRP 中的 JQ 报文格式,包括报文类型、 TTL 、 ID 、源节点地址、多目标节点地址、上一跳节点地址等。 TTL 是一个递减的计时器,表示该蚂蚁的剩余生存时间。 ID 是每只蚂蚁的序列号,它的作用是当一只蚂蚁

到达节点 R 时,节点 R 会验证蚂蚁的 ID ,看这个 ID 是否到达过本节点。如果到达过,就扔掉该蚂蚁。在 JQ 报文格式的基础上,前进蚂蚁报文中还增加了一个进度标识,进度标识表示前进蚂蚁到达目标节点的进度,如进度标识中的值是 10010101 就表示第 1,3,5,8 的目的节点已经到达过;当进度标识的值为全为 1 时,或者 TTL 为 0 时,该前进蚂蚁就会自己死亡。

设源节点 P 发送前进蚂蚁进行多目标寻路。前进蚂蚁到达每个节点时,节点先会检查蚂蚁报文中目的节点地址表,看有没有和自己的地址匹配的目的地。如果有则生成若干返回蚂蚁,返回蚂蚁会把这个目的节点设为自己寻路的源节点,而原来的源节点会被设为自己寻路的目的节点,然后返回蚂蚁就开始逆向寻路。如果没有匹配的地址,就会直接转发前进蚂蚁。

目的节点 i 的返回蚂蚁报文中包含报文类型、 TTL 、 ID 、源节点地址(目的节点 i 的地址)、目的节点地址(源节点 P)、上一跳节点地址。

这里的返回蚂蚁和传统的蚁群算法的返回蚂蚁不太一样。这里的返回蚂蚁并不是直接按寻找到的路由返回到源节点,而是进行逆向的寻路。这样就保证可以找到源节点与该目的节点之间的最优路由。

表1为节点的路由表。其中 W_{ij} 表示邻节点 j 关于目的节点 i 的信息素, M_j 表示邻节点 j 所包含的所有目的节点的信息素的总和, TOF 表示路由表的操作剩余时间, TOF 计时器是由第一个到达该节点的前进或返回蚂蚁来启动的。当 TOF 为0时该节点将不再接收蚂蚁,同时计算 W_{ij} 的值,当 W_{ij} 的值大于阈值 Y 时,就留下这个值;若 W_{ij} 的值不大于 Y ,就删除这个值。

表1 节点的信息素表

TOF	total	Des address 1	Des address 2	...	Des address m
Ne add 1	M_1	W_{11}	W_{21}	...	W_{m1}
Ne add 2	M_2	W_{12}	W_{22}	...	W_{m2}
...
Ne add n	M_n	W_{1n}	W_{2n}	...	W_{mn}

由于是通过阈值的设定来进行路径保留的判断,这就可能会出现不止一条的可用路径。而且当阈值取值较小时,可能会保留下多条可用路径;当阈值取值较大时,可能保留下来的路径会较少。这也为冗余路径提供了条件。

通过 CACO 算法建立起来的组播组具有很强的路径冗余性,当一条路径不可用时,可以查找相关的冗余路径来替代不可用路径。这样就减少了启动路由发现过程的次数。只有所有或者有若干冗余路径不可用时,才启动路由发现过程。这使得网络结构的鲁棒性得到增强,便于进行分组的维护。

3 仿真与结果分析

本文把 CACO 算法应用在 ODMRP 协议中进行仿真,并与传统的 ACO 算法进行比较。本文主要对 CACO 在组播路由的发现过程进行研究,主要考虑算法的收敛性、鲁棒性,以及对网络开销这些方面的表现。

仿真环境设定为 $2200\text{ m} \times 1000\text{ m}$,节点采用随机无停顿的移动方式;节点的移动速度在 $0 \sim 30\text{ m/s}$ 。节点数目设定在 $20 \sim 80$ 时,蒸发系数设定为 0.8 ;节点数目为 $80 \sim 120$ 时,蒸发系数设定为 0.6 ;节点数在 $120 \sim 200$ 时,蒸发系数设定为 0.5 。阈值 L_p 设定为 0.8 。

图2表示应用这两种不同的寻路方法找到多目标路由后所耗费的时间。设定多目标的数量为5个,从图中可以看到,由于采用了CACO,在相同网络规模下,所用的时间要比ACO少,而且这个特点会随着网络规模的变大而增大。这说明了CACO在一定程度上减少了ACO在多目标寻路上所产生的延迟。

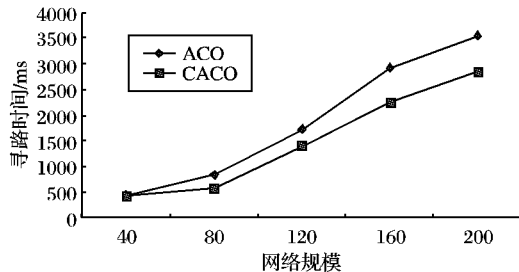


图2 寻找多目标路由的时间消耗比较

图3表示不同的迭代次数下算法对带宽的消耗情况。这里设定网络规模为80个节点,多目标数为5个。从图中可以看到,CACO算法在迭代初期占用的网络带宽要比ACO大,随着迭代次数的增加,CACO所占用的带宽会逐渐小于ACO。这种现象是合理的,因为CACO的返回蚂蚁是新生成的,而不像ACO的返回蚂蚁是由前进蚂蚁转变而成的,这就会在网络中产生大量的返回蚂蚁,而每只蚂蚁又在作寻路操作。这就使得CACO在迭代初期所占用的带宽比ACO的大,而到了迭代后期,由于CACO针对多目标算法的收敛速度要比ACO快,所以后期大多数返回蚂蚁都是按照最优路径返回,就使得应用CACO的网络消耗的带宽小于应用ACO算法的网络。

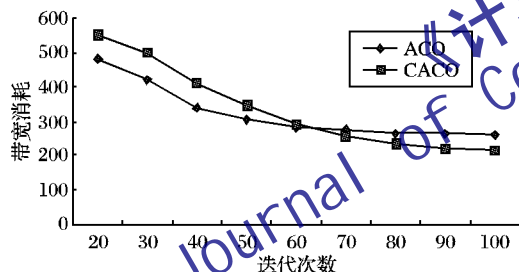


图3 两种算法的带宽消耗比较

图4表示在ODMRP中应用不同的寻路方法所产生的包数目,由于原ODMRP所应用的是洪泛的方法,所以在这里把洪泛的方法也加进来进行比较。这里同样设定多目标个数为5个,从图4中可以看出,洪泛的方法所产生的包数目要远高于ACO和CACO。而CACO所产生的包数在小规模的网络中会高于ACO,但在较大规模的网络中会小于ACO,这是因为CACO与ACO都在进行搜索时,CACO所产生的包数目要大于ACO。但是由于CACO的收敛速度比ACO快,CACO会先于ACO找到最优路由,所以先于ACO停止发送包,随着网络规模的变大,这种现象就越明显,产生的包数目就会逐渐比ACO小。

4 结语

基于蚁群算法的多目标的寻找往往不能同时找到所有的目标,从而产生了很大的延迟。本文对原始蚁群算法进行了改进,使其采用复制蚂蚁并逆向寻路的方式,来达到减少多目标寻找所造成的延迟的效果。这样的改进摆脱了原始ACO算法的“单目标”寻找的局限性,使得ACO算法适合于应用在组播路由协议中。仿真结果也证实了CACO算法在多目标

寻路中要优于原ACO算法,减少了不能同时找到多目的地节点产生的延迟,使得CACO算法更适合于应用在组播路由协议中。

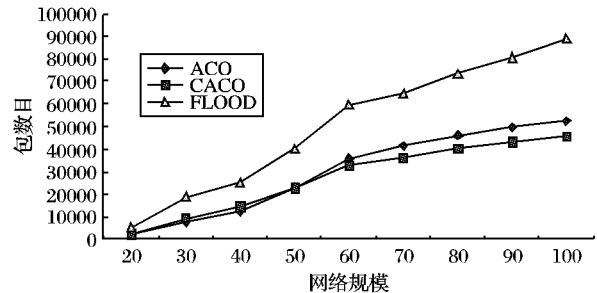


图4 产生的包数目比较

参考文献:

- [1] 陈林星, 曾曦, 曹毅. 移动 Ad Hoc——自组织分组无线网络技术 [M]. 北京: 电子工业出版社, 2006: 7-9, 48-51.
- [2] 郑相全. 无线自组网技术实用教程 [M]. 北京: 清华大学出版社, 2004: 161-179.
- [3] ROSATI L, BERIOLI M, REALI G. On ant routing algorithms in Ad Hoc networks with critical connectivity [J]. Ad Hoc Network, 2008, 6(6): 827-859.
- [4] CARO G D, DUCATELLE F, CAMMARIELLA L M. AntHocNet: An adaptive nature inspired algorithm for routing in mobile Ad Hoc networks [J]. European Transactions on Telecommunications, 2005, 16(2): 443-455.
- [5] WANG H, SHI Z, GE A F, et al. An optimized ant colony algorithm based on the gradual changing orientation factor for multi-constraint QoS routing [J]. Computer Communications, 2009(32): 586-593.
- [6] KADONO D, IZUMI T, OOSHITA F, et al. An ant colony optimization routing based on robustness for Ad Hoc networks with GPSs [J]. Ad Hoc Networks, 2009, 8(1): 63-76.
- [7] WANG J P, OSAGIE E, THULASIRAMAN P, et al. HOPNET: A hybrid ant colony optimization routing algorithm for mobile Ad Hoc network [J]. Ad Hoc Networks, 2009, 8(4): 690-705.
- [8] HAI L J, LIU X, XIA Y D. Research on multicast routing protocols for mobile Ad Hoc networks [J]. Computer Networks, 2008, 52(5): 988-997.
- [9] ROYER E, PERKINS C. Multicast operation of the Ad Hoc on-demand distance vector routing protocol [C]// Proceedings of MobiCom. Seattle: [s. n.], 1999.
- [10] CHIANG C C, GERLA M, LEE S J. On-demand multicast routing protocol in multihop wireless mobile networks [J]. Mobile Networks and Applications, 2002, 7(6): 441-453.
- [11] MENCHACA-MENDEZ R, VAISHAMPAYAN R, GARCIA-LUNA-ACEVES J J, et al. DPUMA: A highly efficient multicast routing protocol for mobile Ad Hoc networks [M]. Berlin: Springer, 2005: 178-191.
- [12] YUN S Y, YI K C, HAN C C, et al. A genetic algorithm for energy-efficient based multicast routing on MANETs [J]. Computer Communications, 2008, 31(4): 858-869.
- [13] CHIANG T C, LIU C H, HUANG Y M. A near-optimal multicast scheme for mobile Ad Hoc networks using a hybrid genetic algorithm [J]. Expert Systems with Applications, 2007, 33(3): 734-742.
- [14] HUANG C J. Using particle swarm optimization for QoS in Ad Hoc multicast [J]. Engineering Applications of Artificial Intelligence, 2009, 22(8): 1188-1193.