第35卷 第23期 计 算 机 工 程 2009年12月
Vol.35 No.23 Computer Engineering December 2009
·人工智能及识别技术· 文章编号：1000—3428(2009)23—0168—04 文献标识码：A 中图分类号：TP18, TP311

基于多帧数据的目标分群算法

龙真真[1,2]，张 策[2]，吴伟胜[3]，刘飞裔[1]
(1. 国防科技大学信息系统与管理学院系统工程系，长沙 410073；2. 空军装备研究院，北京 100085；
3. 中国华阴兵器试验中心，华阴 714200)

摘 要：针对在多帧数据条件下的目标分群问题，提出一种基于数据流聚类的动态目标分群算法TG-Stream，该算法由在线和离线2个部分组成。在线部分采用临时存储结构(TSS)和金字塔时间框架保存侦察数据集的概要信息，离线部分采用CNM算法对时间框架的信息进行聚类，最终得到分群的结果。实验结果表明，TG-Stream具有灵活的精度和效率平衡性，能较好地满足决策辅助系统处理实时信息的需要。
关键词：目标分群；多帧数据；数据流聚类；态势估计

Target Grouping Algorithm Based on Multi-frame Data

LONG Zhen-zhen[1, 2], ZHANG Ce[2], WU Wei-sheng[3], LIU Fei-yi[1]
(1. Department of System Engineering, School of Information System and Management, National University of Defense Technology, Changsha 410073; 2. Equipment Academy of Air Force, Beijing 100085; 3. Ordnace Test Center of Huayin China, Huayin 714200)

【Abstract】In order to solve the target grouping below multi-frame data, this paper introduces a new grouping algorithm, TG-Stream, which can be divided into two parts: on-line part and off-line part. In on-line part, it uses the concepts of a pyramidal time frame and a Temporary Storage Structure(TSS) to save summary information of sensor data. In off-line part, it uses CNM algorithm to cluster the suitable data and output the grouping result. Experimental results show that TG-Stream algorithm has good equilibrium between accuracy and efficiency.
【Key words】target grouping; multi-frame data; data stream clustering; situation assessment

1 概述

目标分群(target grouping)，也称实体分群(operational entity grouping)，是减轻现代战场上指挥员认知压力的重要手段，其主要过程为：在一级数据融合[1]所提供的大量敌方作战目标状态信息基础上，通过同类信息的聚合(aggregation)和深层次的信息抽取(exploitation)，将敌方的战斗实体，即目标逐渐划分为更高抽象层次的作战群，从而为军事决策提供更为有效的参考。与原始的作战目标状态信息集相比，通过目标分群以后的战场信息减少了指挥员所需要关注的“焦点”数量，可以大大提高他们的决策速度和质量。

目标分群是一个典型的NP问题[2]，目前的分群方法基本上都没有考虑侦察数据随时间的变化问题，其本质上都属于目标分群的静态解决方案，无法满足现实需要。因此，本文将在多帧数据(multi-frame data)条件下的目标分群问题看作数据流聚类问题，将其分解为在线(on-line)和离线(off-line)2个阶段，提出基于CNM算法[3]和金字塔时间框架[4]的动态分群算法TG-Stream，与现有的解决方案相比，该算法更符合作战过程中的实际需求。

2 问题描述

数据流聚类即是针对数据流的聚类问题。如果把随时间不断变化的侦察数据看作数据流，那么多帧数据的目标分群问题可以看作是一个数据流聚类问题，将等时间间隔的侦察数据帧序列 X_1, X_2, …, X_k, …看作数据流，其中，数据集 $X_k=\{O_{k1},O_{k2},\cdots,O_{kN}\}$，$O_{ki}=(P_{1_{ki}},P_{2_{ki}},\cdots,P_{m_{ki}})$ 代表 $T=k$ 时刻的数据帧内所包含的 N 个目标的 m 个状态数据集合，即本文所需要解决的问题是将一个时间段 $[t_a, t_b]$ 内的目标状态集 $\Omega_{(t_b-t_a)}=X_{t_a}\cup X_{t_a+1}\cup\cdots\cup X_{t_b}$ 中的各个目标划分到多个互不相交的作战集群中，形成对 $\Omega_{(t_b-t_a)}$ 的一个划分 $\Psi_1,\Psi_2,\cdots,\Psi_n$，使

$$\bigcup_{i=1}^{n}\Psi_i=\Omega_{(t_b-t_a)},\ \forall i\neq j,\ \Psi_i\bigcap\Psi_j=\phi$$

其中，每个集合 Ψ_i $(1\leqslant i\leqslant n)$ 对应于一个作战集群，n 为上述目标集划分的作战集群数量。多帧数据的目标分群问题中侦察数据的数据量以惊人的速度不断增长对分群算法提出了新的挑战。一般来说，基于数据流聚类的多帧数据目标分群方法必须以增量方式跟踪数据流的变化，同时不能占用过多的计算和内存资源[5]。多帧数据目标分群方法与静态目标分群方法的区别在于：

(1)由于数据量无限增长，对目标侦察数据集的扫描次数仅限于单遍，即除非刻意保存，每个数据只可以被处理一次。而在静态分群方法中，数据集通常保存在数据文件中，可对数据集进行多遍扫描。

(2)数据帧序列的快速流动性要求算法对数据的分析处理速度不能低于数据帧的到达速度，因此，算法对新数据的处理时间是严格受限的。而在静态目标分群方法中，数据静态保存在数据库中，对算法的处理时间并无此限制。

(3)相对于数据帧序列的无限数据量，系统中的内存容量是微不足道的，因此，对多帧数据目标分群方法来说，空间复杂度是严格受限的。

作者简介：龙真真(1983－)，男，硕士研究生，主研方向：系统论证与仿真评估；张 策，高级工程师、硕士；吴伟胜，助理工程师、学士；刘飞裔，硕士研究生
收稿日期：2009-04-02 **E-mail：**longzhenzhen@gmail.com

(4)数据帧序列数据量的无限性使多帧数据目标分群往往无法保留原始数据，仅能在内存中维护原始数据的一系列概要信息，并基于这些概要信息生成最终结果。多帧数据目标分群结果往往是有一定允许误差的近似结果。而在静态目标分群中，数据量相对较小，且无处理时间、扫描遍数等限制，因而可获得相对准确的分群结果。

3 目标分群方法

3.1 数据转换及维护

首先要将侦察的数据进行转换，以便进行分群计算，本文将侦察数据集转化为一个加权图。

定义 1(加权图(weighted graph)) 图 $G=(V,E,W)$ 是由有穷非空节点集 V、边集 E 和边权值函数 W 构成的三元组。其中，$V=(v_1,v_2,\cdots,v_N)$ 代表一帧侦察数据中的目标，无序偶对 (v_i,v_j) 表示目标 $v_i\in V$ 与 $v_j\in V$ 之间的联系，w_{ij} 则为边 (v_i,v_j) 的权值。设(X, H)是一度量空间，其中 X 为数据对象集，H 为这个数据对象集合上的距离函数，设常数 $C\sim o(\min H(x, y))$，连接 2 个节点的边的权值函数为

$$W_{xy}=\frac{1}{C+H(x,y)}$$

针对不同的数据类型，距离函数 H 的选取也不同，本文采用欧氏距离(euclidean distance)作为 2 个目标间的距离，假设每个目标数据集包含 p 个变量，则目标 i 和目标 j 之间的距离为

$$d(i,j)=\sqrt{(|z_{i1}-z_{j1}|^2+|z_{i2}-z_{j2}|^2+\ldots+|z_{ip}-z_{jp}|^2)}$$

$$i=(x_{i1},x_{i2},\cdots,x_{ip});\ j=(x_{j1},x_{j2},\cdots,x_{jp})$$

其中，$z_{if}=\dfrac{x_{if}-\min_f}{\max_f-\min_f}$，$\min_f$ 和 $\max_f$ 分别是向量 $\boldsymbol{f}$ 的最大和最小值。

为处理每一帧新到的侦察数据，本文定义一个临时存储结构存储概要信息：

定义 2(临时存储结构(Temporary Storage Structure, TSS)) 定义在目标状态集 $\{O_1,O_2,\cdots,O_n\}$ 上的临时存储结构可以表示为一个 3 元组：

$$TSS=(S,G,\boldsymbol{T})$$

$S=\{(\max_1,\min_1),(\max_2,\min_2),\cdots,(\max_r,\min_r)\}$ 为目标属性集中不同属性项最大最小值，r 为向量数目。$G=(V,E,W)$ 是定义 1 中所提到的加权图，目标的状态向量值用“$\sqcup$”表示，如 $O_3\cdot ID$ 表示目标 O_3 的固定编号，则其中 $V=(O_1\cdot ID,O_2\cdot ID,\cdots,O_n\cdot ID)$ 表示当前参与分群的目标集；无序偶对 $(O_i\cdot ID,O_j\cdot ID)$ 表示目标 $O_i\cdot ID\in V$ 与 $O_j\cdot ID\in V$ 之间当前的联系，w_{ij} 则为边 $(O_i\cdot ID,O_j\cdot ID)$ 当前的权值。$\boldsymbol{T}=(PT_{O_1\cdot ID},PT_{O_2\cdot ID},\cdots,PT_{O_n\cdot ID})$ 为目标的更新时间向量，$PT_{O_i\cdot ID}$ 为目标 $O_i\cdot ID$ 的状态信息最近一次更新时间。

定义 3(加法规则(additive rule)) 设 TSS_i 是 $T=i$ 时刻的临时存储结构，$X_j=\{O_{j1},O_{j2},\cdots,O_{jn}\}$ 是 $T=j$ 时刻($j>i$)的数据帧内所包含的即时目标状态集；设任意一目标状态向量元素最多为 s 个，则 $\mu=(\mu_1,\mu_2,\cdots,\mu_s)$ 是各状态向量的权重；$O=\{V_1,V_2,\cdots\}$ 是状态空间中有关向量的原点，μ 和 O 都为指挥员人工指定。

将数据集 X_j 加入临时存储结构 $TSS_i=(S_i,G_i,T_i)$ 后得到的新的临时存储结构为 TSS_j：

$$TSS_j=TSS_i+X_j=(S_j,G_j,T_j)$$

(1) S_j 的构建方法

将数据集 X_j 中的目标状态集中的间隔数值类型状态向量值与 S_i 中相应的最大最小值进行比较：

1)当 X_j 中对应的目标状态集中的向量值超出 S_i 中相应值的范围时，更新 S_i 中对应的值，同时更新 G_i 中的节点权值。

2)当 X_j 中对应的目标状态集中的向量值未超出 S_i 中相应值的范围时，保留 S_i 中对应的值。

例如，在 X_j 中目标 $O_{j2}\cdot P_{3_{j2}}$ 为 2 700，而 $(\max_3,\min_3)\in S_i$ 为 (2 400,0), $O_{j2}\cdot P_{3_{j2}}>\max_3$，则更新 $(\max_3,\ \min_3)\in S_j$ 为(2 700,0)。

(2) G_j 的构建方法

定义一个数据结构 K，$K=\{M_1,M_2,L,M_k,L\}$，其中，数据组 M_k 中的成员为 G_i 中所有与原点距离为 k 的目标；输入 X_j 数据集时，先计算 X_j 内各个目标关于原点的距离；然后根据距离的远近将目标编号插入不同的数据组中，从而得到 G_j。

(3) T_j 的构建方法

1)当 $O_{jk}\cdot ID\in V_i$ 时，则更新 T_i 中的对应元素为当前时间 j，例如假设当前时刻为 t_c，$O_{j5}\cdot ID=O_3\cdot ID$，则更新 $PT_{O_3\cdot ID}$ 为 t_c；

2)当 $O_{jk}\cdot ID\notin V_i$ 时，则在 T_i 增加一个新的元素 $PT_{O_{n'+1}\cdot ID}=j$，其中，n' 为加权图 G 中的即时目标数目。

定义 4(减法规则(subtractive rule)) 设 TSS_i 和 TSS_j 分别是当时间为 $T=i$ 和 $T=j$ ($j>i$)时的 2 个临时存储结构，则它们相减后得到的结果为

$$G'_j=TSS_j-TSS_i$$

具体方法如下：

(1)计算 $T'_j=T_j-T_i=(\Delta PT_{O_1\cdot ID},\Delta PT_{O_2\cdot ID},\cdots,\Delta PT_{O_n\cdot ID})$；

(2)由指挥员设定衰变阈值为 t_{limit}，当 $\Delta PT_{O_i\cdot ID}>t_{\text{limit}}$ 时，保留 G_j 中的节点 i；当 $\Delta PT_{O_i\cdot ID}\leqslant t_{\text{limit}}$ 时，删除 G_j 中的节点 i。

3.2 金字塔时间框架

TSS 采用金字塔式时间间隔的存储方式。在每一个存储时刻，当前的 TSS 被存储在一个快照(snapshot)之中。存储时刻分成若干层级，相同层级上的连续 2 个快照间隔相同，高层级的间隔是相邻的低层级间隔的 α 倍，其中 α 为大于 1 的整数。每一层级上的快照维护方法如下：

(1)第 i 层级上的快照时间间隔为 α^i，即数据流开始之后 α^i 整数倍时间点是该层级保存快照的时刻；

(2)在任意时刻，仅保留第 i 层级上最新的 $\alpha+1$ 个快照。

按层级保存快照的目的是平衡存储量与精确性。文献[4]已证明，对于任意指定的时间窗长 h，在从当前时刻向前追溯 $2h$ 个时间单位,至少存在一个已存储的快照。运用该存储策略可以在消耗尽可能小的存储空间的基础上保证离线部分的完成。

3.3 剪枝策略

随着时间的变化，内存中存储的数据不断增加，而内存的存储空间有限，即使利用了金字塔时间框架，存储的信息仍然具有一定冗余，因此，本文提出动态在线维护策略——剪枝策略。剪枝是通过判断目标节点的衰变函数值来完成的，下面是衰变函数的定义：

定义 5(衰变函数) 由于数据流可以看成是随时间不断变化的无限过程，而隐含在其中的目标状态信息是有时效性的，

因此，本文利用一个衰变函数 $f(t)$来描述这种时效性：

$$f(t)=2^{-\lambda t},\quad \lambda>0$$

其中，λ为衰变因子，λ越高目标状态信息的时效性就越强。

剪枝策略是在某一给定时间间隔内对加权网中目标节点的衰变函数值进行不断检查，设置一个最低权限或规则，不满足要求的目标状态信息被视为噪声，从当前的 TSS 中将其删除。具体步骤是：由指挥员人工设定衰变因子 $\lambda=\lambda_n$ 和噪声阈值为 ς，设当前 *TSS* 中的目标数目为 n,当前时间为 t_c，当 $t_c/\alpha^i=x$，x 为整数时，依据 *TSS* 中的 $\boldsymbol{T}=(PT_{O_1\square ID},PT_{O_2\square ID},\cdots,PT_{O_n\square ID})$ 找出衰变值小于噪声阈值的目标删除。

3.4 在线部分

在线部分是任务是随着侦察数据帧序列的不断到达，实时提取其摘要信息并以适当的结构进行有效的存储，每条摘要信息在系统中都是以临时存储结构 *TSS* 的形式，按照金字塔式时间间隔进行存储，以平衡存储消耗和精确性。

在分群系统开始运行之前，首先要创建初始的 *TSS*。这个过程将在在线部分的最开始进行。在开始时(t=1)，将扫描已经存在于磁盘上的目标状态数据集，采用定义 1 中的方法，将其转换为一个加权图 G_1，再初始化 S_i,B_i,T_i，从而得到 TSS_1。

3.5 离线部分

在算法的离线部分中，指挥员拥有更多的自主性，可以在各种不同的时间粒度上对原始数据进行更精确的分析。离线部分是基于整个侦察数据帧序列处理过程的。当指挥员指定某一个时间窗口长度为 h，希望在这个时间段内进行分群计算时，我们就需要找到在这个时间范围内的加权图。下文将介绍如何利用金字塔存储结构找到符合要求的加权图。

设当前的时间为 t_c，指挥员希望在过去时间范围 h 内寻找动态加权图。通过金字塔存储框架，可以找到保存在发生时间为 t_c-h 的快照(金字塔时间框架的应用保证了系统总有可能找到在时间 t_c-h' 上保存的快照，其中，h'的误差在指挥员预定容许的偏差范围内)。假设在 t_c-h 上保存的临时存储结构为 $TSS_{(t_c-h')}$，时间 t_c 上的临时存储结构为 TSS_{t_c}，则运用减法规则可以得到加权网 $G_{tc}=TSS_{t_c}-TSS_{(t_c-h')}$，在此基础上应用 CNM 算法即可得到多帧数据条件下的目标分群结果。

TG-Stream 中的在线部分实际是将数据帧中的目标状态数据进行动态维护的过程，然后通过离线部分的聚类运算得到最终的分群结果，算法的主要流程如图 1 所示。

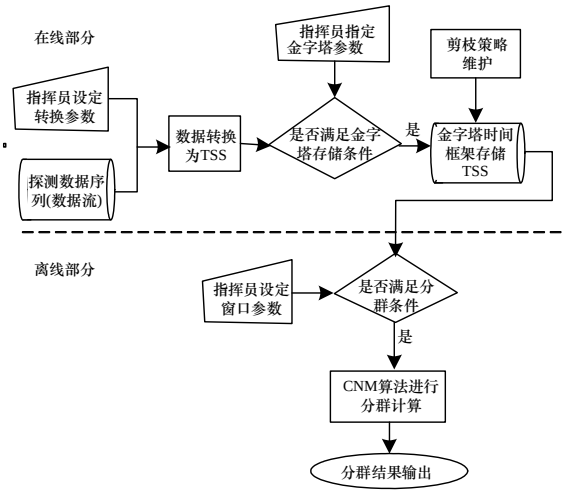


图 1　TG-Stream 流程

4　实验结果

实验采用的测试平台为 Intel T2300 1.66 GHz, 1.5 GB 内存，操作系统 Windows XP，运算软件为 Matlab 7.1。用于实验的仿真数据来自 UCL 的 wine[6]数据集，从数据集的 3 个手工分类中分别随机提取 40 条数据，作为 3 个作战集群共计 120 个目标的状态信息。实验过程为：当 T=1，即在第 1 帧侦察数据中，120 个作战目标分成 3 个作战集群，然后每经过一个时间单位，从其中一指定集群中删除一个目标，最终到 T=40 时，将只剩下 2 个作战集群。

聚类纯度(clustering purity)[7]是目前应用得较为广泛的聚类结果度量指标，但由于在多帧数据条件下分群的数据集随时间不断变化，因此它不适合作为这种情况下衡量目标分群结果的量纲。本文在其基础上提出了一个新的衡量指标——动态分群纯度，定义如下：

定义 6(动态分群纯度(dynamic grouping purity)) 设当前时刻为 t_c，需要进行分群的数据集为 D_{t_c}，令 P_{t_c} 为 D_{t_c} 上一个分群结果(即一个划分)，$p\in P_{t_c}$，表示该划分中的一个群；当前的实际数据集为 E_{t_c}，令 L_{t_c} 表示 E_{t_c} 上的一个手工分类的结果(即一个划分)，$l\in L_{t_c}$，表示该手工分类的一个类别。则有：

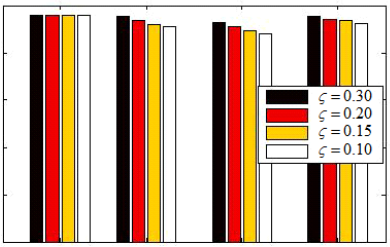
对于 $\forall p\in P_{t_c}$，p 相对于 $l\in L$ 的查准率定义为

$$Precision(p,l)=\frac{|p\cap l|}{|p|}$$

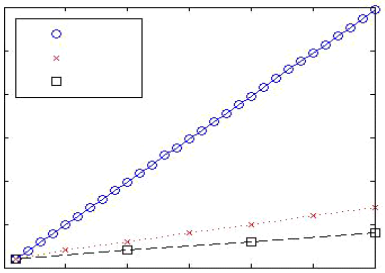
动态分群纯度定义为

$$Purity(P_{t_c},\ L_{t_c})=\sum_{p\in P}\frac{|L_{t_c}|}{|E_{t_c}|}\max_{l\in L}Precision(p,\ l)$$

可见，动态分群纯度是时间为 t_c 时每个群的查准率的加权平均值，而各群的查准率取决于进行分群的数据 D_{t_c} 和实际数据集 E_{t_c} 之间的误差大小以及当时的分群结果相对各手工分类类别的最大查准率之和。该指标能全面反映多帧数据条件下动态分群的效果。图 2 为 ς，h 参数对分群过程的影响。



(a) ς 对分群结果的影响



(b) h 对运行效率的影响

图 2　ς，h 参数对分群过程的影响

图 2(a)为采用 TG-Stream 算法，在 $h=1$, $\lambda_h=1$, $t_{\text{limit}}=1$ 条件下，噪声阈值 ς 分别取 0.30, 0.20, 0.15, 0.10 时，分群效果的对比。可以看到在 TG-Stream 算法中随着噪声阈值的减小，分群的效果也逐渐变差，其主要原因是由于过时的数据未及时消除，造成系统中出现了“虚目标(virtual target)”，但在 T=1, 10, 20, 30 这 4 个时间节点上，动态分群纯度都保持在 90%以上。这说明噪声阈值越大，分群的效果就越好，容易分析得知，由于进行减法操作也需要消耗一定的系统资源，噪声阈值过大，会造成这方面的系统负担过重，因此要按实际情况选择合适的噪声阈值。

图 2(b)为采用 TG-Stream 算法，在 $\varsigma=0.2$, $\lambda_h=1$, $t_{\text{limit}}=1$ 条件下，时间窗口 h 分别取 1, 5, 10 时累积处理耗时对比。可以发现当时间窗口的取值增大时，随着数据帧序列的不断到达，累积处理耗时差别越来越大。显然，时间窗口 h 越大，系统效率越高，但同时带来的副作用是离线部分间隔变大，即分群结果显示的刷新频率变慢。显示的刷新速度主要和指挥员的层次有密切关系，指挥员的层次越高，对刷新频率的敏感度就越低。

5 结束语

目标分群作为态势估计的重要组成，是数据融合过程中的关键。本文以数据流聚类为基础，提出一种新的分群算法 TG-Stream，与以往静态的分群算法相比，该方法更满足军事需求，能为指挥员正确有效分析和决策打下了坚实的基础。

参考文献

[1] Hall D L, Llinas J. An Introduction to Multi-sensor Data Fusion[J]. Proceedings of the IEEE, 1997, 85(1): 6-23.

[2] 蔡益朝, 张维明, 刘　忠, 等. 基于遗传算法的实体分群问题的求解方法[J]. 计算机工程, 2007, 33(5): 4-6.

[3] Claueset A, Newman M E J, Moore C. Finding Community Structure in Very Large Networks[J]. Physical Review E, 2004, 70(6).

[4] Aggarwal C, Han Jiawei, Wang Jianyong, et al. A Framework for Clustering Evolving Data Streams[C]//Proc. of VLDB'03. Berlin, Germany: [s. n.], 2003.

[5] Babcock B, Babu S, Datar M, et al. Models and Issues Data Stream Systems[C]//Proc. of PODS'02. [S. l.]: ACM Press, 2002.

[6] Forina M. PARVUS — — An Extendible Package for Data Exploration, Classification and Correlation[R]. Genoa, Italy: Institute of Pharmaceutical and Food Analysis and Technologies. Tech. Rep.: 16147, 2007.

[7] Sun Ying, Zhu Qiuming, Chen Zhengxin. An Iterative Initial-points Refinement Algorithm for Categorical Data Clustering[J]. Pattern Recognition Letters, 2002, 23(7): 875-884.

编辑　金胡考

（上接第 167 页）

为了进一步验证性能，还分别测试了采用标准 SVM 进行入侵检测、采用标准遗传算法进行特征抽取(以分类错误率为适应度函数)以及进行联合优化的入侵检测技术的性能。实验结果如表 4 所示。

表 4　CEGA-SVM 与 SVM, GA-SVM 的性能比较

数据集	算法	特征数目	正确检测率/(%)	检测时间/ms
Probe	SVM	41	85.0	0.486
Probe	GA -SVM	24	96.5	0.283
Probe	CEGA-SVM	17	98.6	0.261
DoS	SVM	41	83.6	0.484
DoS	GA -SVM	20	96.8	0.273
DoS	CEGA-SVM	19	96.8	0.270
U2R	SVM	41	87.5	0.024
U2R	GA -SVM	21	95.6	0.017
U2R	CEGA-SVM	19	99.6	0.014
R2L	SVM	41	88.3	0.482
R2L	GA -SVM	21	94.6	0.276
R2L	CEGA-SVM	19	97.6	0.270

从分类正确率的角度来看，由于在 CEGA-SVM 算法中采用了将特征抽取和检测模型联合优化的方法，同时在进行特征抽取时考虑到了训练数据的统计特性，本文算法的平均性能要优于其他文献的算法性能，特别是在较难检测的数据集上取得了满意的检测效果。

实验结果表明，采用 CEGA-SVM 所提取的特征数量仅为算法使用特征的一半，为 CEGA-SVM 算法提取特征的 85%左右，因此，在检测时延上比 SVM 和 GA-SVM 算法均有显著减少，同时在检测正确率方面也有提高，比 SVM 和 GA-SVM 算法分别提高了近 10%和 2.4%，漏检率和误检率也有明显下降，可见 CEGA-SVM 是一种有效的入侵检测算法。

5 结束语

本文提出将条件熵遗传算法用于最优特征子集的选择，同时进行 SVM 模型的参数优化，以寻找最佳特征子集和与之相匹配的 SVM 检测模型，从而实现入侵检测。但目前的入侵检测系统与理想的还有较大差距，下一步将完善相关技术，对现有算法做进一步改进。

参考文献

[1] Crosbie M, Spafford G. Defending a Computer System Using Autonomous Agent[R]. [S. l.]: COAST, Purdue University, Technical Report: 95-022, 2007.

[2] Aho A, Corasick M. Efficient String Matching an Aid to Bibliographic Search[J]. Communications of the ACM, 2008, 18(6): 33-40.

[3] Horspool R. Practical Fast Searching in Strings[J]. Software, Practice and Experience, 2007, (10): 230-235.

[4] Sunday D M. A Very Fast Substring Search Algorithm[J]. Communications of The ACM, 2006, 33(3): 132-142

[5] 郑洪英, 倪　霖. 一种无监督网络入侵检测算法[J]. 计算机工程, 2008, 34(18): 184-185.

[6] Coit C J, Staniford S. Toward Faster String Matching for Intrusion, Detection or Exceeding the Speed of Snort[C]//Proc. of the DISCEX'05. [S. l.]: IEEE Computer Society, 2005.

编辑　顾姣健