Neurocomputing 99 (2013) 124–133

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Extracting elite pairwise constraints for clustering

He Jiang [a,*], Zhilei Ren [a], Jifeng Xuan [a], Xindong Wu [b]

[a] School of Software, Dalian University of Technology, Dalian, LiaoNing 116621, China
[b] Computer Science Department, University of Vermont, Burlington Vermont 05403, United States

ARTICLE INFO

Article history:
Received 3 November 2011
Received in revised form
3 April 2012
Accepted 3 June 2012
Communicated by D. Tao
Available online 11 July 2012

Keywords:
Semi-supervised
Elite pairwise constraints
Clustering

ABSTRACT

Semi-supervised clustering under pairwise constraints (i.e. must-links and cannot-links) has been a hot topic in the data mining community in recent years. Since pairwise constraints provided by distinct domain experts may conflict with each other, a lot of research work has been conducted to evaluate the effects of noise imposing on semi-supervised clustering. In this paper, we introduce elite pairwise constraints, including elite must-link (EML) and elite cannot-link (ECL) constraints. In contrast to traditional constraints, both EML and ECL constraints are required to be satisfied in every optimal partition (i.e. a partition with the minimum criterion function). Therefore, no conflict will be caused by those new constraints. First, we prove that it is NP-hard to obtain EML or ECL constraints. Then, a heuristic method named Limit Crossing is proposed to achieve a fraction of those new constraints. In practice, this new method can always retrieve a lot of EML or ECL constraints. To evaluate the effectiveness of Limit Crossing, multi-partition based and distance based methods are also proposed in this paper to generate faux elite pairwise constraints. Extensive experiments have been conducted on both UCI and synthetic data sets using a semi-supervised clustering algorithm named COP-KMedoids. Experimental results demonstrate that COP-KMedoids under EML and ECL constraints generated by Limit Crossing can outperform those under either faux constraints or no constraints.

1. Introduction

Clustering is a well-known unsupervised learning technique in data mining with numerous applications, including pattern recognition, web mining, textual document analysis, and bioinformatics [1,2]. The goal of clustering is to partition a data set into k clusters such that data instances are more similar in the same cluster, while more dissimilar in distinct clusters. During the clustering process, removing outliers may also be conducted [5]. In recent years, semi-supervised clustering has become a hot topic attracting intensive efforts from the data mining community [11]. In contrast to traditional (unsupervised) clustering, semi-supervised clustering conducts the clustering process under the guidance of some supervisory information. Among the supervisory information, pairwise constraints (i.e. must-links (ML) and cannot-links (CL)) [3,4,6–8] are most widely used in semi-supervised clustering. A ML $m(x_i,x_j)$ indicates that instances x_i and x_j must be assigned to the same cluster, while a CL $c(x_i,x_j)$ indicates that instances x_i and x_j must be assigned to distinct clusters. Clustering algorithms using pairwise constraints fall into two categories, i.e., distance based and constraint based methods. The distance based methods [9,10,30,28,29] train their distance functions from pairwise constraints either before or during the clustering process. In addition, some related work of distance based methods can also be found in cross-domain applications [27] and dimension reduction [19,24]. The constraint based methods [4,7] use pairwise constraints to modify the cluster assignment stage during the clustering process so that constraints could be satisfied as many as possible. Most research efforts in this area assumed that those pairwise constraints were available a priori by consulting domain experts [12]. However, those pairwise constraints acquired by consulting are not always correct. At the meantime, some pairwise constraints provided by different experts may conflict with each other. Therefore, it is essential to verify those pairwise constraints before clustering. In the literature, many papers [7,10,13–16] have investigated how to evaluate the effects of noise onpairwise constraints and proposed a few methods to detect those noisy constraints.

In this paper, we introduce new concepts of elite pairwise constraints, i.e. elite must-links (EML) and elite cannot-links (ECL). An EML $em(x_i,x_j)$ indicates that instances x_i and x_j should appear together in every optimal partition (an optimal partition is one with the minimum criterion function). Similarly, an ECL $ec(x_i,x_j)$ requires that instances x_i and x_j be assigned to distinct

* Corresponding author.
E-mail addresses: hejiang@ieee.org, jianghe@dlut.edu.cn (H. Jiang), ren@mail.dlut.edu.cn (Z. Ren), xuan@mail.dlut.edu.cn (J. Xuan), xwu@uvm.edu (X. Wu).
URL: http://oscar-lab.org/people/-jxuan/ (J. Xuan).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.06.013

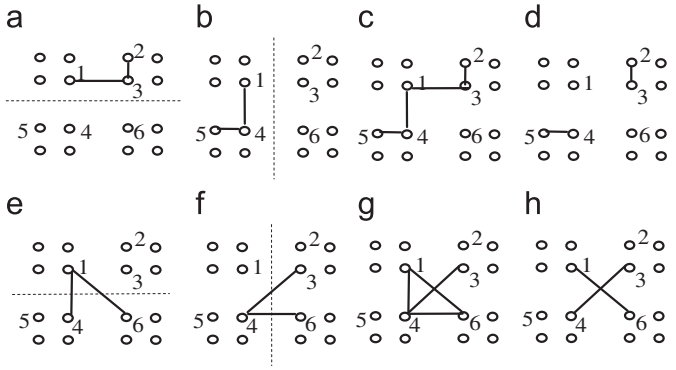

Fig. 1. Examples of pairwise constraints and elite pairwise constraints.

clusters in every optimal partition. In contrast to traditional pairwise constraints, EML and ECL constraints can always provide beneficial guidance for clustering. For example, Fig. 1 shows a clustering problem on a data set with 16 instances and the goal is to partition all instances into two clusters such that the sum of the dissimilarity measures of all instances from their closest centroids is minimized. There are only two optimal partitions as shown in Fig. 1(a) and (b), respectively. Fig. 1(a) presents ML $m(1,3)$ and $m(2,3)$ by solid lines. Fig. 1(b) presents ML $m(1,4)$ and $m(4,5)$. Obviously, when $m(1,3)$ and $m(2,3)$ are used together in semi-supervised clustering, the optimal partition in Fig. 1(a) is likely to be found. Similarly, when $m(1,4)$ and $m(4,5)$ are employed in semi-supervised clustering, the optimal partition in Fig. 1(b) is also apt to be found. However, if all these four ML constraints are simultaneously satisfied (see Fig. 1(c)), no optimal partition can be achieved! This example shows that the combination of ML constraints from distinct domain experts may lead to no optimal partition, even when such constraints are helpful if separately used. On the other hand, ML $m(2,3)$ and $m(4,5)$ are also EML constraints. Since both $m(2,3)$ and $m(4,5)$ can be satisfied in any optimal solution (see Fig. 1(d)), they can always lead the clustering process towards good partitions. This example implies that EML constraints could be more helpful than ML constraints in semi-supervised clustering. A similar observation can also be found for ECL constraints. In Fig. 1(e), two CL constraints $c(1,4)$ and $c(1,6)$ are given. Fig. 1(f) presents CL $c(4,6)$ and $c(3,4)$. However, when those four CL constraints are employed together, no optimal partition can be found (see Fig. 1(g)). Among those four CL constraints, both $c(1,6)$ and $c(3,4)$ are ECL constraints. When $c(1,6)$ and $c(3,4)$ are satisfied, all those optimal partitions are apt to be found.

In this paper, we investigate the computational complexity for EML and ECL constraints and develop a heuristic method to obtain a part of such constraints.

First, we show that it is NP-hard to obtain either EML or ECL constraints. The basic idea is as follows. Given a data set, we can transform it to a new data set with a unique optimal partition. For this new data set, it can be easily verified that the optimal partition can be obtained from those EML (or ECL) constraints, and vice versa. On the other hand, this optimal partition is also optimal for the original data set. Hence, obtaining the optimal partition for the original data set is equivalent to calculating EML (or ECL) constraints for the new data set. Therefore, we can derive our conclusion regarding the NP-hardness for obtaining the optimal partition of the original data set.

Second, an effective method named Limit Crossing is proposed to extract some EML and ECL constraints. According to the definition, no optimal partition can be achieved if an EML constraint is violated. Therefore, we can check whether there exists an EML between instances x_i and x_j as follows. If every partition satisfying CL $c(x_i,x_j)$ is worse than an optimal partition of the original data set, then there exists an EML $em(x_i,x_j)$. Otherwise, there is no EML $em(x_i,x_j)$. Since it is intractable to obtain those optimal partitions, Limit Crossing employs a relaxed method in the following way. Given a data set, a lower bound is calculated for every partition satisfying CL $c(x_i,x_j)$. Besides, an upper bound is also computed for the optimal partition of this data set. Then an EML $em(x_i,x_j)$ can be determined if the lower bound exceeds the upper bound. Similarly, an ECL constraint can be determined as well.

To evaluate the effectiveness of Limit Crossing, extensive experiments have been conducted on both UCI data sets and synthetic data sets. For comparison, two other methods are adopted in the experiments for generating faux EML and ECL constraints. Experimental results demonstrate that a semi-supervised clustering algorithm under EML and ECL constraints (generated by Limit Crossing) can achieve better partitions than those under no constraints or faux constraints.

This paper is organized as follows. In Section 2, we briefly summarize the related work of our paper. In Section 3, we provide some definitions related to clustering, EML, and ECL. The computational complexity of retrieving elite pairwise constraints is investigated in Section 4. The Limit Crossing method is proposed in Section 5 to obtain some EML and ECL constraints. In Section 6, a semi-supervised K-Medoids algorithm is presented to evaluate the effectiveness of those EML and ECL constraints. As a comparison, we also calculate two kinds of faux elite pairwise constraints. Experimental results on both UCI data sets and synthetic data sets are then demonstrated. Finally, we conclude this paper in Section 7.

2. Related work

In the literature, a lot of research work has been conducted on the computational complexity related to pairwise constraints and the ways to detect noisy constraints before clustering.

Klein et al. [10] theoretically showed that it is NP-hard to find feasible partitions under the CL constraints (a full proof can be found in [7]), while there exists a polynomial time algorithm to find a feasible partition under the ML constraints.

Some empirical results were presented by Davidson et al. [13] for the effects of ML and CL constraints imposing on clustering. According to their paper, an easy–hard phase transition exists for a semi-supervised clustering algorithm to find a feasible partition as the number of CL constraints increases. When ML and CL constraints are used together, an easy–hard–easy phase transition exhibits to find a feasible partition. In addition, an algorithm was proposed to generate a subset of ML and CL constraints which was easy to satisfy in clustering.

Davidson et al. [14] demonstrated that it is NP-hard to repair a partition violating a set of ML and CL constraints so that every constraint is satisfied. In addition, given a set of constraints for which no feasible partition exists, they showed that it is also NP-hard to identify the minimal subset of constraints to prune so that a feasible partition exists for the remaining constraints.

Freund et al. [15] investigated the effect of noise on ML and CL constraints in semi-supervised clustering. After transforming those constraints to a graph, they introduced several intuitive noise models. Then they derived a quantification of the effect which noisy constraints imposed on semi-supervised clustering. Their results showed that a fraction of noisy constraints can have the ground truth constraints useless in semi-supervised clustering.

Yip et al. [16] showed that even a small proportion of incorrect ML and CL constraints could make a semi-supervised clustering algorithm perform worse than an un-supervised clustering.

They proposed a heuristic method to detect potentially incorrect constraints. A semi-automatic approach was also given to have user manually verify those suspect constraints.

According to the above discussions, it is difficult to determine a feasible partition under a set of ML and CL constraints [10,13]. In addition, it is also hard to detect those noisy ML or CL constraints [14–16]. In contrast to ML and CL constraints, there will be no conflicts in a set of EML and ECL constraints. Therefore, it is easy to find a feasible partition under those elite pairwise constraints.

3. Notations

Since there exists no universally agreed upon definition for traditional clustering, we consider the mostly used one in this paper as follows.

Given a data set $X=\{x_1,x_2,\ldots,x_n\}$ and the distance function $d(x_i,x_j)\in R^+\cup\{0\}$,[1] a k-partition is a set $C=\{C_1,C_2,\ldots,C_k\}$ consisting of k disjoint clusters, such that:

(I) $C_i\neq\emptyset$, $\forall i\in\{1,2,\ldots,k\}$,
(II) $\bigcup_{i=1}^{k}C_i=X$,
(III) $C_i\cap C_j=\emptyset$, $\forall i,j\in\{1,2,\ldots,k\}$ and $i\neq j$.

The aim of clustering is to seek for an optimal k-partition under some predefined criterion function, which is heavily dependent on the representing instance types of clusters. Among those types, the mean (average of the instances in a cluster) and the medoid (an instance with minimal average distance to all the instances in a cluster) are two of the mostly used representing instances in the literature. Since the mean is sensitive to outliers (or noise) and the computation for the mean is sometimes unavailable in many real applications [17], we adopt the medoid for representing every cluster in the following part.

Given a k-partition $C=\{C_1,C_2,\ldots,C_k\}$, the criterion function [18] is then defined as $J(X,C)=\sum_{i=1}^{k}\sum_{x_j\in C_i}d(x_j,\mu_i)$, where $\mu_i\in C_i$ is the medoid for cluster C_i. Under the above definition, the optimal k-partitions are those ones with minimum criterion functions. It has been proven that it is NP-hard to obtain the optimal k-partitions [20]. If there is only one optimal k-partition for a data set, we say that this data set has a unique optimal k-partition.

Given a data set $X=\{x_1,x_2,\ldots,x_n\}$, if instances x_i and x_j always appear together in one cluster for every optimal partition, then we say there is an EML constraint (denoted by $em(x_i,x_j)$). The set of all EML constraints is denoted by $EM(X)$. Similarly, we can define ECL as follows. If instances x_i and x_j always appear in distinct clusters for every optimal partition, then there is an ECL constraint (denoted by $ec(x_i,x_j)$). The set of all ECL constraints is denoted by $EC(X)$.

4. Intractability of elite pairwise constraints

In this section, we will prove that it is NP-hard to obtain either $EM(X)$ or $EC(X)$. Given a data set $X=\{x_1,x_2,\ldots,x_n\}$, we will assume that the distance function $d(x_i,x_j)\in Z^+\cup\{0\}$ in the following proof. Otherwise, for the data set with floating instance distances, we can simply rescale the distances to nonnegative integers by multiplying a sufficiently large number. The optimal k-partitions for the data set with rescaled distance function are identical to the original data set.

[1] There exist many distance measures [20] in the literature, including Minkowski distance, Euclidean distance, Cosine similarity, and City-block distance. For brevity, we use a general format as $d(x_i,x_j)\in R^+\cup\{0\}$.

Given a data set X, its biased data set is defined as $\widehat{X}=\{x_1,x_2,\ldots,x_n\}$ with the distance function $\widehat{d}(x_i,x_j)=d(x_i,x_j)+1/2^{i\times n+j}$ for any $1\leq i<j\leq n$ and $\widehat{d}(x_i,x_j)=\widehat{d}(x_j,x_i)$ for $1\leq j<i\leq n$. Given a k-partition $C=\{C_1,C_2,\ldots,C_k\}$ for $\widehat{X}$, its criterion function is calculated as $J(\widehat{X},C)=\sum_{i=1}^{k}\sum_{x_j\in C_i}\widehat{d}(x_j,\mu_i)$.

Lemma 1. *Given a data set $X=\{x_1,x_2,\ldots,x_n\}$ with the distance function $d(x_i,x_j)\in Z^+\cup\{0\}(1\leq i\neq j\leq n)$, there exists a unique optimal k-partition for its biased data set $\widehat{X}$.*

Proof. To prove Lemma 1, we just need to show that $J(\widehat{X},C^1)\neq J(\widehat{X},C^2)$ for any two distinct k-partitions C^1,C^2. Since $C^1\neq C^2$, there must exist a cluster $C^1_{i^*}\in C^1$ and a cluster $C^2_{j^*}\in C^2$ such that $C^1_{i^*}\cap C^2_{j^*}\neq\emptyset$ and $C^1_{i^*}\neq C^2_{j^*}$. It implies that $C^1_{i^*}\backslash(C^1_{i^*}\cap C^2_{j^*})\neq\emptyset$ or $C^2_{j^*}\backslash(C^1_{i^*}\cap C^2_{j^*})\neq\emptyset$. We only consider the case that $C^1_{i^*}\backslash(C^1_{i^*}\cap C^2_{j^*})\neq\emptyset$ in the following proof (the proof for the other case that $C^2_{j^*}\backslash(C^1_{i^*}\cap C^2_{j^*})\neq\emptyset$ can be done in a similar way). Therefore, there must exist an instance $x_{i'}\in C^1_{i^*}\backslash(C^1_{i^*}\cap C^2_{j^*})$. We now consider the medoid (denoted by $x_{j'}$) of the cluster $C^1_{i^*}$ as follows.

Case 1: $x_{j'}\in C^1_{i^*}\cap C^2_{j^*}$

Since $x_{i'}\notin C^2_{j^*}$, we have that $x_{i'}$ and $x_{j'}$ must belong to distinct clusters in k-partition C^2. When encoded in binary string, the $\min(i',j')\times n+\max(i',j')$th bit of the fraction part of $J(\widehat{X},C^1)$ equals to 1. However, the same bit of $J(\widehat{X},C^2)$ will be zero. Therefore, we have that $J(\widehat{X},C^1)\neq J(\widehat{X},C^2)$ holds.

Case 2: $x_{j'}\in C^1_{i^*}\backslash(C^1_{i^*}\cap C^2_{j^*})$

Since $C^1_{i^*}\cap C^2_{j^*}\neq\emptyset$, there must exist an instance $x_{i''}\in C^1_{i^*}\cap C^2_{j^*}$. In a similar way to Case 1, the $\min(i'',j')\times n+\max(i'',j')$ th bit of the fraction part of $J(\widehat{X},C^1)$ equals to 1, while the same bit of $J(\widehat{X},C^2)$ is zero. We have that $J(\widehat{X},C^1)\neq J(\widehat{X},C^2)$ too.

Thus, this lemma is proven. □

Lemma 2. *Given a data set $X=\{x_1,x_2,\ldots,x_n\}$ with the distance function $d(x_i,x_j)\in Z^+\cup\{0\}(1\leq i\neq j\leq n)$, the unique optimal k-partition for its biased data set $\widehat{X}$ is also optimal for X.*

Proof. Otherwise, there must exist a k-partition (denoted by C) for X such that $J(X,C)<J(X,C^*)$, where C^* is the unique optimal k-partition for its biased data set $\widehat{X}$ (C^* is also a k-partition for the data set X). Since the distance function $d(x_i,x_j)\in Z^+\cup\{0\}$ $(1\leq i\neq j\leq n)$, we have that $J(X,C)-J(X,C^*)\leq -1$. Given any cluster $C_i\in C$, let $J(X,C_i)=\sum_{x_j\in C_i}d(x_j,x_{i^*})$, where x_{i^*} is the medoid of C_i for X. Let $J(\widehat{X},C_i)=\sum_{x_j\in C_i}\widehat{d}(x_j,x_{i'})=\sum_{x_j\in C_i}(d(x_j,x_{i'})+1/2^{\min(j,i')\times n+\max(j,i')})$, where $x_{i'}$ is the medoid of C_i for $\widehat{X}$. Let $\delta_i=\sum_{x_j\in C_i}1/2^{\min(j,i')\times n+\max(j,i')}$, it is easy to verify that $0<\delta_i<1$. Hence $J(\widehat{X},C_i)<\sum_{x_j\in C_i}d(x_j,x_{i'})+1$. Then, we can derive that $J(\widehat{X},C_i)=J(X,C_i)+\delta_i$, otherwise:

Case 1: $\sum_{x_j\in C_i}d(x_j,x_{i'})<J(X,C_i)$

Then the medoid of C_i for X should be $x_{i'}$ rather than x_{i^*}. A contradiction is met.

Case 2: $\sum_{x_j\in C_i}d(x_j,x_{i'})>J(X,C_i)$

According to the distance function definition, we have that $\sum_{x_j\in C_i}d(x_j,x_{i'})\geq J(X,C_i)+1$. Therefore, the medoid of C_i for $\widehat{X}$ should be x_{i^*} rather than $x_{i'}$. A contradiction is met.

Hence, we have that $J(\widehat{X},C)=J(X,C)+\sum_{C_i\in C}\delta_i$. On the other hand, we have that $\sum_{C_i\in C}\delta_i<\sum_{1\leq j<l\leq n}1/2^{j\times n+l}<1$. Therefore, we have that $0<J(\widehat{X},C)-J(X,C)<1$ holds. In a similar way, it can be proven that $0<J(\widehat{X},C^*)-J(X,C^*)<1$. Therefore, we have the following inequality:

$$J(\widehat{X},C)-J(\widehat{X},C^*)=J(X,C)-J(X,C^*)+(J(\widehat{X},C)-J(X,C))-(J(\widehat{X},C^*)-J(X,C^*))$$
$$\leq -1+(J(\widehat{X},C)-J(X,C))-(J(\widehat{X},C^*)-J(X,C^*))<0$$

It implies that the unique optimal k-partition for $\widehat{X}$ should be C rather than C^*. It contradicts with the assumption that C^* is the optimal k-partition for $\widehat{X}$.

Thus this lemma is proven. □

Lemma 3. *Given a data set $X=\{x_1,x_2,\ldots,x_n\}$ with the distance function $d(x_i,x_j)\in Z^+\cup\{0\}(1\le i\ne j\le n)$, the unique optimal k-partition for its biased data set $\widehat{X}$ can be obtained from $EM(\widehat{X})$ (or $EC(\widehat{X})$) in polynomial time.*

Proof. By Lemma 1, the biased data set $\widehat{X}$ has a unique optimal k-partition (denoted by C^*), it is easy to verify that there must be an EML constraint $em(x_i,x_j)$ in $EM(\widehat{X})$ for any two distinct instances x_i,x_j appearing in the same cluster of C^*. Under such observation, we can obtain C^* from $EM(\widehat{X})$ by the following steps. First, an arbitrary instance $x\in\widehat{X}$ is chosen as the first instance for the cluster C_1. Then every instance x' associated with an EML constraint $em(x,x')$ is added to the cluster C_1. After C_1 is constructed, $\widehat{X}$ is updated by deleting all the instances in C_1. Then, we arbitrarily choose another instance from $\widehat{X}$ to construct C_2. In a similar way, all other clusters can be constructed. Obviously, the whole construction of C^* from $EM(\widehat{X})$ can be done in polynomial time ($O(n)$).

To show that C^* can be also obtained from $EC(\widehat{X})$ in polynomial time, we just need to acquire $EM(\widehat{X})$ from $EC(\widehat{X})$ in polynomial time. It can be easily done by generating an EML constraint $em(x_i,x_j)$ for every pair of instances x_i,x_j without an ECL constraint $ec(x_i,x_j)$.

Thus lemma is proven. □

Theorem 1. *It is NP-hard to obtain the set of EML constraints.*

Proof. Otherwise, there exist an algorithm (denoted by A) which can obtain the set of EML constraints in polynomial time. Given any arbitrary data set $X=\{x_1,x_2,\ldots,x_n\}$ with the distance function $d(x_i,x_j)\in Z^+\cup\{0\}(1\le i\ne j\le n)$, we shall show its optimal k-partition can be achieved in the following way. First, we can easily construct its biased data set $\widehat{X}$ in $O(n^2)$ running time. According to the assumption, the set of EML constraints $EM(\widehat{X})$ for $\widehat{X}$ can be obtained by A in polynomial time (denoted by $P(n)$). By Lemma 3, the unique optimal k-partition (denoted by C^*) for $\widehat{X}$ can be calculated from $EM(\widehat{X})$ in $O(n)$ running time. On the other hand, C^* is also optimal for X (by Lemma 2). Therefore, we can obtain an optimal k-partition for X in $O(n^2)+P(n)+O(n)$ running time. It contradicts with the fact that the clustering problem is NP-hard.

Thus this theorem is proven. □

Corollary 2. *It is NP-hard to obtain the set of ECL constraints.*

5. Limit Crossing method

By Theorem 1 and Corollary 2, there exists no algorithm guaranteeing to obtain $EM(X)$ or $EC(X)$ in polynomial time under the assumption $P\ne NP$ [21]. In this section, we will introduce Limit Crossing algorithm which can always find a fraction of $EM(X)$ or $EC(X)$ in practice.

5.1. Check-Links algorithm

Given a data set X, let $J_{opt}(X)$ be the criterion function of its optimal k-partitions. Given any arbitrary k-partition C, we have that $J_{opt}(X)\le J(X,C)$ holds. Given instances $x_i,x_j\in X$, let $J_{opt-(x_i,x_j)}(X)=\min_{C\in\sqcap(x_i,x_j)}J(X,C)$, where $\sqcap(x_i,x_j)$ is the set of all k-partitions such that x_i and x_j appear in distinct clusters. Let $J_{opt+(x_i,x_j)}(X)=\min_{C\in\sqcup(x_i,x_j)}J(X,C)$, where $\sqcup(x_i,x_j)$ is the set of all k-partitions such that x_i and x_j appear in the same cluster. Then we can easily verify the following propositions.

Proposition 1. *Given any EML constraint $em(x_i,x_j)\in EM(X)$, we have that $J_{opt}(X)<J_{opt-(x_i,x_j)}(X)$.*

Proposition 2. *Given any ECL constraint $ec(x_i,x_j)\in EC(X)$, we have that $J_{opt}(X)<J_{opt+(x_i,x_j)}(X)$.*

Under the above discussions, the Check-Links algorithm (see Table 1) is proposed to determine $EM(X)$ or $EC(X)$. It iteratively compares $J_{opt}(X)$ with $J_{opt-(x_i,x_j)}(X)$ or $J_{opt+(x_i,x_j)}(X)$ for every pair of instances $x_i,x_j\in X$. If $J_{opt}(X)$ is smaller than $J_{opt-(x_i,x_j)}(X)$, we can claim that there exists an EML constraint $em(x_i,x_j)$ by Proposition 1. Otherwise, if $J_{opt}(X)$ is smaller than $J_{opt+(x_i,x_j)}(X)$, there must exist an ECL constraint $ec(x_i,x_j)$. Although Check-Links is an exact algorithm seeking for $EM(X)$ or $EC(X)$, it is intractable in practice due to the fact that it is NP-hard to compute the value of $J_{opt}(X)$. In addition, there is no efficient way to calculate the values of $J_{opt-(x_i,x_j)}(X)$ and $J_{opt+(x_i,x_j)}(X)$.

5.2. Limit Crossing

In this subsection, we present Limit Crossing algorithm to obtain a fraction of $EM(X)$ or $EC(X)$.

5.2.1. Motivation

In contrast to Check-Links, Limit Crossing is a heuristic algorithm which employs upper bound and lower bound functions to return a part of $EM(X)$ or $EC(X)$. The idea of Limit Crossing is first proposed by Climer et al. [22] for obtaining the backbone of the traveling salesman problem (TSP). TSP is a well-known NP-hard problem in the literature. Given a set of cities and the distance between every pair of cities, TSP aims to find the shortest tour which visits every city once and exactly once. According to the definition, a tour consists of a set of edges between cities. The backbone is defined as the shared common parts of all optimal tours for TSP. Obviously, no optimal tour can be found when an edge belonging to the backbone is forbidden to appear in the tours. To detect whether an edge belongs to the backbone or not, Climer et al. introduced a relaxed method which employs an upper bound and a lower bound function for TSP. Its basic idea is as follows. Given edge (i,j) between city i and j, if the lower bound of the length of those tours containing no (i,j) is greater than the upper bound of optimal tours for the original problem, it can be claimed that the edge (i,j) belongs to the backbone. Its validity is straightforward, since it implies that an optimal tour cannot be obtained unless the edge (i,j) from the backbone is included. Since Limit Crossing is sometimes time-consuming and sensitive to the upper bounds, Ren et al. [23] recently develop the Accelerated Limit Crossing based Multilevel Algorithm (ALCMA). The upper bound sensitivity is removed by a dynamic pseudo upper bound mechanism in ALCMA.

5.2.2. Limit Crossing algorithm

We first introduce some related definitions, and then present our Limit Crossing algorithm (see Table 2). Given a data set X, let $Up(X)$ be a upper bound of $J_{opt}(X)$, i.e., $Up(X)\ge J_{opt}(X)$. Given instances $x_i,x_j\in X$, let $Lr_{-(x_i,x_j)}(X)$ be a lower bound of $J_{opt-(x_i,x_j)}(X)$,

Table 1
Check-Links algorithm.

Check-Links (data set)
1. let $EM(X)=\emptyset$, $EC(X)=\emptyset$
2. for every pair of instances $x_i,x_j\in X$ do
   if $J_{opt}(X)<J_{opt-(x_i,x_j)}(X)$, then add $em(x_i,x_j)$ to $EM(X)$;
   else if $J_{opt}(X)<J_{opt+(x_i,x_j)}(X)$, then add $ec(x_i,x_j)$ to $EC(X)$;
3. return $EM(X)$,$EC(X)$.

i.e., $Lr_{-(x_i,x_j)}(X) \le J_{opt-(x_i,x_j)}(X)$, let $Lr_{+(x_i,x_j)}(X)$ be a lower bound of $J_{opt+(x_i,x_j)}(X)$. Therefore, we can derive the following propositions.

Proposition 3. *Given a data set X and instances $x_i,x_j \in X$, if $Lr_{-(x_i,x_j)}(X) > Up(X)$, then there exists an EML constraint $em(x_i,x_j)$.*

It is easy to verify the validity of Proposition 3. Since $Up(X) \ge J_{opt}(X)$ and $Lr_{-(x_i,x_j)}(X) \le J_{opt-(x_i,x_j)}(X)$, then we have $J_{opt}(X) < J_{opt-(x_i,x_j)}(X)$ holds if $Lr_{-(x_i,x_j)}(X) > Up(X)$. In a similar way, we have Proposition 4 for determining an ECL constraint.

Proposition 4. *Given a data set X and instances $x_i,x_j \in X$, if $Lr_{+(x_i,x_j)}(X) > Up(X)$, then there exists an ECL constraint $ec(x_i,x_j)$.*

By Propositions 3 and 4, we can find some elite pairwise constraints by comparing the upper bound and lower bounds related to every pair of instances. In such a way, the computation can be avoided for the values of $J_{opt}(X)$, $J_{opt-(x_i,x_j)}(X)$, and $J_{opt+(x_i,x_j)}(X)$, which is usually intractable in practice. Although this relaxed method cannot guarantee to find all the elite pairwise constraints due to the tightness of the lower bounds and the upper bound, it is shown to be effective by experimental results with respect to the NP-hardness for obtaining $EM(X)$ or $EC(X)$ (see Theorem 4 and Corollary 5).

Based on the above discussion, our new algorithm Limit Crossing is presented in Table 2. After generating an upper bound $Up(X)$ and initializing $EM(X)$ and $EC(X)$, the new algorithm checks every pair of instances iteratively as follows. First, it checks whether a new EML constraint can be implied by $EM(X)$ (Step 3.1). If succeeded, this new EML constraint is added to $EM(X)$. Otherwise, the new algorithm continues to check whether an ECL constraint can be directly inferred by $EM(X)$ and $EC(X)$ (see Step 3.2). If succeeded, the ECL set $EC(X)$ will be updated. Otherwise, the algorithm applies Propositions 3 and 4 to determine whether there is an EML or ECL constraint (see Steps 3.3 and 3.4). The running time complexity of Limit Crossing can be analyzed as follows. Let $f(n)$ be the running time for computing $Up(X)$ of X (any existing clustering algorithm can be adopted for computing a k-partition as the upper bound). Let $g(n)$ and $h(n)$ be the running time for computing lower bound $Lr_{-(x_i,x_j)}(X)$ and $Lr_{+(x_i,x_j)}(X)$, respectively. Both Check-EM and Check-EC can be conducted in $O(n)$ running time. Since there are totally $(n^2-n)/2$ pairs of instances, the worst-case running time of Limit Crossing is $f(n)+(n^2-n)(2O(n)+g(n)+h(n))/2$.

5.2.3. Upper bound and lower bound

In Limit Crossing algorithm, an upper bound and a lower bound are required for applying Propositions 3 and 4. According to the definition of $J_{opt}(X)$, an upper bound can be given by the criterion function of a k-partition generated by an existing clustering algorithm. We will show that the lower bounds can be obtained by the relaxation of the integer programming formulations for clustering. Our clustering problem can be formulated as follows [18]:

$$\min \quad \sum_{s=1}^{n}\sum_{r=1}^{n} d(x_r,x_s)w_{rs}$$

$$\text{s.t.} \quad \sum_{s=1}^{n} w_{rs} = 1, \quad r \in \{1,2,\ldots,n\} \tag{1}$$

$$\sum_{s=1}^{n} y_s = k \tag{2}$$

$$w_{rs} \le y_s, \quad r,s \in \{1,2,\ldots,n\} \tag{3}$$

$$w_{rs}, y_s \in \{0,1\}, \quad r,s \in \{1,2,\ldots,n\} \tag{4}$$

where a k-partition to the clustering problem is defined by an allocation matrix $W_{n\times n}$ and a vector Y indicating which instances are chosen as the medoids. Constraints (1) and (3) ensure that each instance x_r is allocated to only one medoid x_s. Constraint (2) determines the exact number of k medoids to be chosen and constraint (4) gives the integer conditions. Aiming to calculate the lower bound $Lr_{-(x_i,x_j)}(X)$, we should impose extra constraints on the integer programming formulations above as follows. According to the definition of $J_{opt-(x_i,x_j)}(X)$, instances x_i and x_j are required to appear in distinct clusters. When incorporated into the integer programming formulations, this requirement turns to two constraints, i.e., the following constraints:

$$w_{il_1} = w_{jl_2} = y_{l_1} = y_{l_2} = 1, \quad l_1 < l_2, l_1,l_2 \in \{1,2,\ldots,n\} \tag{5}$$

$$w_{il_1} = w_{jl_2} = y_{l_1} = y_{l_2} = 1, \quad l_1 > l_2, l_1,l_2 \in \{1,2,\ldots,n\} \tag{6}$$

We have that $J_{opt-(x_i,x_j)}(X) = \min\{H_1,H_2\}$, where $H_1 = \min \sum_{s=1}^{n} \sum_{r=1}^{n} d(x_r,x_s)w_{rs}$ subject to Constraints (1)–(5), $H_2 = \min \sum_{s=1}^{n} \sum_{r=1}^{n} d(x_r,x_s)w_{rs}$ subject to Constraints (1)–(4) and Constraint (6). Although it is intractable to exactly calculate either H_1 or H_2, we can easily obtain their lower bounds (denoted by $Lr(H_1)$ and $Lr(H_2)$, respectively) using the Lagrangian relaxation method [25]. Therefore, we can calculate $Lr_{-(x_i,x_j)}$ by $Lr_{-(x_i,x_j)} = \min\{Lr(H_1), Lr(H_2)\}$.

Similarly, the lower bound $Lr_{+(x_i,x_j)}$ can also be computed as follows. In addition to constraints (1)–(4), constraint (7) is needed to ensure that instances x_i and x_j always belong to the same cluster:

$$w_{il_1} = w_{jl_2} = y_{l_1} = y_{l_2} = 1, \quad l_1 = l_2, l_1,l_2 \in \{1,2,\ldots,n\} \tag{7}$$

Then, we have that $J_{opt+(x_i,x_j)}(X) = \min \sum_{s=1}^{n} \sum_{r=1}^{n} d(x_r,x_s)w_{rs}$ subject to constraints (1)–(4) and (7). Therefore, $Lr_{+(x_i,x_j)}$ can be calculated by the Lagrangian relaxation method.

As shown in [25], Lagrangian relaxation can be solved by a series of iterations. Each iteration calls an $O(n^2)$ subroutine to optimize Lagrangian multipliers. Therefore, the running time to calculate $Lr_{-(x_i,x_j)}$ or $Lr_{+(x_i,x_j)}$ will be $O(dn^2)$, where d is the number of iterations (in this paper, d is set to be 5000).

Table 2
Limit Crossing algorithm.

Limit Crossing (data set X)
1. calculate the upper bound (denoted by $Up(X)$) of X
2. let $EM(X) = \emptyset$, $EC(X) = \emptyset$
3. for every pair of instances $x_i,x_j \in X$ do
   3.1 if Check-EM(x_i, x_j, $EM(X)$) is true, then add $em(x_i,x_j)$ to $EM(X)$
   3.2 else if Check-EC(x_i, x_j, $EM(X)$, $EC(X)$) is true, then add $ec(x_i,x_j)$ to $EC(X)$
   3.3 else if $Lr_{-(x_i,x_j)}(X) > Up(X)$, then add $em(x_i,x_j)$ to $EM(X)$
   3.4 else if $Lr_{+(x_i,x_j)}(X) > Up(X)$, then add $ec(x_i,x_j)$ to $EC(X)$
4. return $EM(X)$, $EC(X)$.

Check-EM (instances x_i,x_j, EML set $EM(X)$)
1. if $\exists x_l \in X$ s.t $em(x_i,x_l) \in EM(X)$ and $em(x_j,x_l) \in EM(X)$, then return true;
2. return false;

Check-EC(instances x_i,x_j,EML set $EM(X)$, ECL set $EC(X)$)
1. if $\exists x_l \in X$ s.t. $em(x_i,x_l) \in EM(X)$ and $ec(x_l,x_j) \in EC(X)$, then return true;
2. if $\exists x_l \in X$ s.t. $em(x_j,x_l) \in EM(X)$ and $ec(x_l,x_i) \in EC(X)$, then return true;
3. return false.

6. Experiments

In this section, we will evaluate the effectiveness of those elite pairwise constraints generated by Limit Crossing algorithm. A semi-supervised clustering algorithm is used in this evaluation. For comparison, two kinds of faux pairwise constraints are also generated. We will evaluate those pairwise constraints in terms of

the converging speed, the criterion function, the purity, and the normalized mutual information (NMI).

6.1. Experiments set up

All the algorithms are coded in C++ on a PC of intel PIV and 2G memory running WinXP. In this paper, both UCI and synthetic data sets are used to evaluate the effectiveness of Limit Crossing algorithm.

(1) UCI data sets
We select four data sets from the UCI repository (http://archive.ics.uci.edu/ml/). The glass data set (see Table 3) consists of 214 data instances with 10 attributes forming six classes. The soybean data set consists of 47 data instances with 34 attributes forming four classes. The wine data set consists of 178 data instances with 13 attributes forming three classes. The iris data set contains three classes of 50 data instances each and every data instance consists of four attributes.

(2) Synthetic data sets
Every synthetic data set is generated with intrinsic cluster patterns by the following steps. All data instances are to be located within a 100×100 2D space. First, k data instances are generated randomly such that the distance between any two of them is greater than $100/2\sqrt{k}$. These data instances are regarded as the virtual centers of the clusters in the data generation process. Second, every data instance x_i ($i=1,2,\ldots,n$) in the data set is randomly assigned to a virtual cluster and x_i is randomly chosen from all the positions within a distance of $100/4\sqrt{k}$ from this chosen cluster center. By this data generation procedure, data instances are closer in the same cluster, while farther in different clusters. Following the above data generation procedure, two groups of data sets (see Table 4) are generated. Each group consists of five data sets. For the data sets in Group1, the number of clusters is set to be 5, while the number of data instances varies between 200 and 1000. In contrast to Group1, every data set in Group2 contains 1000 data instances, while the number of clusters varies from 5 to 25. It should be noted that we use the same data set for both SynK-5 and SynN-1.

Table 3
Parameters for UCI data sets.

Data set	# of Attributes	n	k
Glass	10	214	6
Soybean	34	47	4
Wine	13	178	3
Iris	4	150	3

Table 4
Parameters for synthetic data sets.

Data group	Data set	n	k
Group1	SynK-1	200	5
	SynK-2	400	5
	SynK-3	600	5
	SynK-4	800	5
	SynK-5	1000	5
Group2	SynN-1	1000	5
	SynN-2	1000	10
	SynN-3	1000	15
	SynN-4	1000	20
	SynN-5	1000	25

6.2. Elite pairwise constraints by Limit Crossing

We have run Limit Crossing on both UCI data sets and synthetic data sets. In the experiments, we use COP-KMedoids (see Section 6.3) taking no constraints to generate the upper bound for every data set.

The sixth and seventh columns of Table 5 present the returned EML and ECL constraints of UCI data sets. Since the data instances are more likely to be assigned to distinct clusters when $k>1$, we can find that the number of ECL constraints is far more than that of EML constraints. The sixth and seventh columns of Table 6 present the results of synthetic data sets. Since SynK-5 is identical to SynN-1, we just run Limit Crossing once and return the same sets of EML and ECL constraints for these two data sets. Table 6 shows for Group1 that the number of EML constraints grows as the number of instances increases in every data set. The number of ECL constraints follows a similar growth trend. For Group2, the number of EML constraints sharply decreases along with the growth of k, while the number of ECL constraints slightly increases. The trend of the number of EML Constraints can be explained as follows. According to the generation method of Group2, there exists a unique optimal k-partition for every data set in Group2, i.e., all the data instances assigned to a virtual center form a cluster in the optimal k-partition. Therefore, an EML constraint $em(x_i,x_j)$ exists for any two distinct data instances x_i and x_j in a cluster of the optimal k-partition. For every data set in Group2, there are $1000/k$ data instances on average in every cluster of the optimal k-partition. Therefore, the total number of EML constraints is $k*1000/k*(1000/k-1)/2=500*(1000/k-1)$. Hence, the total number of EML constraints gradually decreases along with the growth of the value of k. In a similar way, we can explain why the number of ECL constraints slightly increases.

6.3. Semi-supervised K-medoids

To evaluate the effectiveness of elite pairwise constraints, we employed such constraints in a semi-supervised clustering algorithm named COP-KMedoids (see Table 7). Similar to the modified k-means algorithm (COP-KMeans) [4], COP-KMedoids employs those pairwise constraints to guide the whole clustering process.

Table 5
Pairwise constraints for UCI data sets

Data Set	# of em_m	# of ec_m	# of em_d	# of ec_d	# of em	# of ec
Glass	1086	12 477	416	1445	6815	13 004
Iris	1913	6892	221	797	3556	7205
Soybean	104	624	81	241	247	820
Wine	4005	12 613	247	628	5312	10 441

Table 6
Pairwise constraints for synthetic data sets.

Data set	# of em_m	# of ec_m	# of em_d	# of ec_d	# of em	# of ec
SynK-1	1741	11 860	304	681	3911	15 989
SynK-2	9359	53 134	562	1429	15 949	63 851
SynK-3	13 413	110 756	905	1716	35 767	143 933
SynK-4	37 267	242 531	1185	2766	63 668	255 932
SynK-5	43 449	361 707	1499	4894	99 622	399 878
SynN-1	43 449	361 707	1499	4894	99 622	399 878
SynN-2	23 415	391 532	1460	2774	50 068	449 432
SynN-3	13 346	407 569	1458	3185	33 092	466 334
SynN-4	12 434	455 142	1441	4746	24 810	474 690
SynN-5	8035	439 436	1608	2902	19 813	479 687

Table 7
COP-KMedoids algorithm.

COP-KMedoids(data set X, $Con_=$, $Con_{\neq}$)
1. let $\mu_1,\mu_2,\ldots,\mu_k$ be the initial cluster medoids for cluster $C_1,C_2,\ldots,C_k$, respectively
2. for each instance $x_i \in X$, assign it to the closest cluster medoid μ_j such that VIOLATE-CONSTRAINTS(x_i, C_j, $Con_=$, $Con_{\neq}$) is false. If no such cluster exists, fail (return { })
3. for each cluster, update its medoid
4. iterate between (2) and (3) until convergence
5. return $\{C_1,C_2,\ldots,C_k\}$
VIOLATE-CONSTRAINTS(instance x, cluster C', $Con_=$, $Con_{\neq}$)
1. for each $em(x,x_=) \in Con_=$ do if $x_= \notin C'$, return true.
2. for each $ec(x,x_{\neq}) \in Con_{\neq}$ do if $x_{\neq} \in C'$, return true.
3. return false

The algorithm takes in a data set (X), a set of EML constraints ($Con_=$), and a set of ECL constraints ($Con_{\neq}$). It returns a k-partition satisfying all specified constraints. COP-KMedoids works as follows. First, k random instances are chosen as the initial medoids for the k-partition. Then, it iteratively assigns every instance to the closest cluster medoid without violating the constraints. If a conflict occurs, the algorithm exits without returning a valid k-partition. Otherwise, the new k medoids are recalculated for clusters and instances are then assigned to new medoids. This process continues until converging.

6.4. Faux elite pairwise constraints

For comparison, we also generate two kinds of faux elite pairwise constraints for every data set as follows.

(1) *Multi-partition based EML set and ECL set*: According to the definition of the EML set, there exists an EML constraint $em(x_i,x_j)$ for instances $x_i,x_j \in X$ if they always appear together in the same cluster of every optimal k-partition. The key idea is really straightforward for the multi-partition based EML set. Given multiple k-partitions (generated by a clustering algorithm), if instances x_i and x_j always appear in the same cluster for these k-partitions, then they are also likely to appear together in the same cluster of every optimal k-partition. In this paper, we run COP-KMedoids without constraints (i.e., both $Con_=$ and $Con_{\neq}$ are set to empty) for 10 times to generate 10 k-partitions. Then, we checked for every pair of instances x_i,x_j whether they appear together in the same cluster of these 10 k-partitions. If so, a faux EML constraint (denoted by $em_m(x_i,x_j)$) is added to the multi-partition based EML set (denoted by $EM_m(X)$). In a similar way, we can also obtain the multi-partition based ECL set (denoted by $EC_m(X)$). If instances x_i and x_j always belong to distinct clusters of these 10 k-partitions, a faux ECL constraint (denoted by $ec_m(x_i,x_j)$) is added to $EC_m(X)$. In the calculation, we expand $EM_m(X)$ and $EC_m(X)$ by using the transitive closure for the faux EML constraints. Given $em_m(x_i,x_j)$ and $em_m(x_i,x_l)$, there also exists a faux EML constraint $em_m(x_j,x_l)$. Similarly, given $em_m(x_i,x_j)$ and $ec_m(x_i,x_l)$, we have a faux ECL constraint $ec_m(x_j,x_l)$.

(2) *Distance based EML set and ECL set*: This kind of constraints is based on the observation that instances are closer in the same cluster, while farther in different clusters. We check every pair of instances $x_i,x_j \in X$, if one instance is the nearest neighbor of the other, then a faux EML constraint (denoted by $em_d(x_i,x_j)$)is added to the distance based EML set (denoted by $EM_d(X)$). On the contrary, if one instance is the farthest neighbor of the other, then a faux ECL constraint (denoted by $ec_d(x_i,x_j)$) is added to the distance based ECL set (denoted by $EC_d(X)$). We also expand $EM_d(X)$ and $EC_d(X)$ using the transitive closure.

Table 5 gives the results for UCI data sets. The second and third columns of Table 5 present the multi-partition based EML sets and ECL sets, respectively. Obviously, the number of ECL constraints is far more than that of EML constraints. The fourth and fifth columns demonstrate the distance based EML sets and ECL sets, respectively. It can be observed that there are more distance based ECL constraints than distance based EML constraints. We can also find that far more EML and ECL constraints are generated by multi-partition based method than by distance based method.

Table 6 gives the results for our synthetic data sets. For Group1, we can find that the number of multi-partition based EML constraints sharply increases along with the growth of n (i.e., the number of instances in the data set). The number of multi-partition based ECL constraints follows a similar growth trend. For Group2, the number of multi-partition based EML constraints slightly decreases as the cluster number increases, while the number of multi-partition based ECL constraints follows a nearly reverse trend. The fourth and fifth columns present the distance based EML and ECL sets. For Group1, the number of distance based EML and ECL constraints also gradually increases, when more instances are contained in a data set. For Group2, the number of distance based EML constraints slightly varies between 1441 and 1608, while the number of distance based ECL constraints varies between 2774 and 4894. Similar to UCI data sets, far more EML and ECL constraints are generated by multi-partition based method than by distance based method.

6.5. Experimental results

To evaluate the effectiveness of Limit Crossing method, we performed COP-KMedoids algorithm taking in those elite pairwise constraints generated by Limit Crossing method. We have run COP-KMedoids on every data set for 50 times and averaged the running time, the number of iterations, the criterion function, the purity, and the normalized mutual information in every run. Since the results of the running time demonstrate similar trends like the number of iterations, we only present the number of iterations in the following part. For comparison, we have also run COP-KMedoids taking in two kinds of faux elite pairwise constraints. In addition, we also run an unsupervised COP-KMedoids taking in no constraints. In every run, all these algorithms use the same randomly generated initial cluster medoids. For brevity, we denote COP-KMedoids(L) as COP-KMedoids taking in elite pairwise constraints by Limit Crossing method, COP-KMedoids(M) as COP-KMedoids taking in constraints by multi-partition based method, COP-KMedoids(D) as COP-KMedoids taking in constraints by distance based method, and COP-KMedoids(U) as unsupervised COP-KMedoids.

(1) *Results on UCI data sets*: Fig. 2 shows the number of iterations of COP-KMedoids using different pairwise constraints for UCI

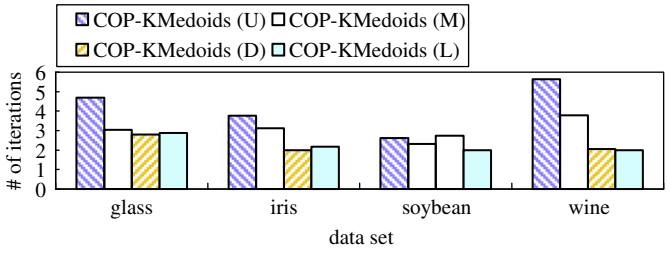

Fig. 2. Number of iterations of COP-KMedoids with different elite pairwise constraints for UCI data sets.

data sets. Out of these four algorithms, COP-KMedoids(U) usually needs more iterations towards convergence. COP-KMedoids(M) usually takes more iterations than COP-KMedoids(D) and COP-KMedoids(L) except soybean data set. Besides, COP-KMedoids(L) converges faster than COP-KMedoids(D) in two data sets, i.e., soybean and wine. Fig. 3 presents the criterion functions of those algorithms for UCI data sets. The criterion functions of COP-KMedoids(M), COP-KMedoids(D), and COP-KMedoids(L), are normalized by that of COP-KMedoids(U), respectively. It can be observed that all those algorithms achieve similar criterion functions over every UCI data set. COP-KMedoids(L) can achieve the best criterion functions, though the criterion function differences among those four algorithms are unremarkable.

We also compare those algorithms in terms of the purity. Given a data set $X=\{x_1,x_2,\ldots,x_n\}$ and a k-partition $C=\{C_1,C_2,\ldots,C_k\}$, the purity of C is defined as $\sum_{i=1}^{k}|C_i\cap\omega_i|/n$, where $W=\{\omega_1,\omega_2,\ldots,\omega_k\}$ represents the set of classes. Fig. 4 presents the purity of those algorithms for UCI data sets. Based on the bars of those algorithms, the differences among those four algorithms are unremarkable and no solid conclusion can be drawn. The reason lies in the classes for UCI data sets are not given under the criterion function used in this paper. Therefore, the classes of UCI data sets cannot guarantee to be optimal k-partitions. For example, the criterion function of the classes of glass data set is 993.34, a really poor k-partition, as compared to the average criterion function (427.87 in this paper) achieved by COP-KMedoids(L).

In addition, we present the normalized mutual information (NMI) of those algorithms in Fig. 5. NMI is a good way to trade off the quality of the clustering against the number of clusters [26]. Given a data set $X=\{x_1,x_2,\ldots,x_n\}$, a k-partition $C=\{C_1,C_2,\ldots,C_k\}$, and the set of classes $W=\{\omega_1,\omega_2,\ldots,\omega_k\}$, NMI is defined as $NMI(W,C)=I(W;C)/([H(W)+H(C)]/2)$, where $I(W;C)$ is mutual information, $H(W)$ and $H(C)$ are entropies. Due to the same reason as in Fig. 4, no clear tendency can be observed.

(2) *Results on synthetic data sets*: Fig. 6 presents the number of iterations of those algorithms for Group1 data sets. Obviously, COP-KMedoids(U) and COP-KMedoids(M) need far more iterations to converge than COP-KMedoids(D). COP-KMedoids(L) usually needs least iterations to converge among these four algorithms, except SynK-2. COP-KMedoids(D) takes the least iterations to converge on SynK-2.

Fig. 7 gives the results of the normalized criterion functions for Group1 data sets. It can be found that the criterion function obtained by COP-KMedoids(U) is the worst among these four algorithms for every data set in Group1. The criterion function by COP-KMedoids(M) is slightly better than that by COP-KMedoids(U), while COP-KMedoids(D) outperforms COP-KMedoids(M) on all data sets in Group1 except SynK-3. Out of these four algorithms, COP-KMedoids(L) always achieves the best criterion function for every data set in Group1.

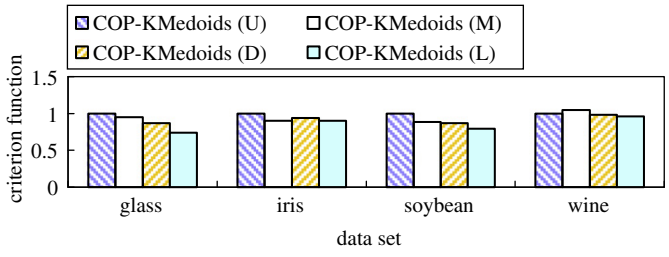

Fig. 3. Normalized criterion functions of COP-KMedoids with different elite pairwise constraints for UCI data sets.

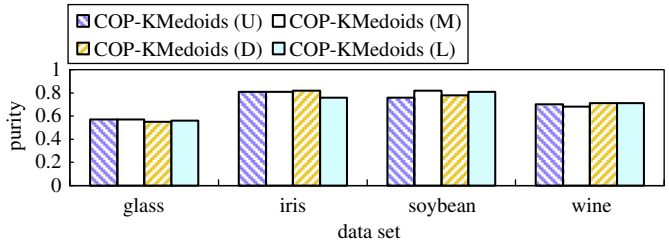

Fig. 4. Purity of COP-KMedoids with different elite pairwise constraints for UCI data sets.

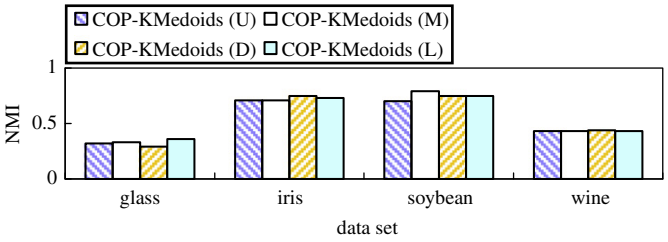

Fig. 5. Normalized mutual information of COP-KMedoids with different elite pairwise constraints for UCI data sets.

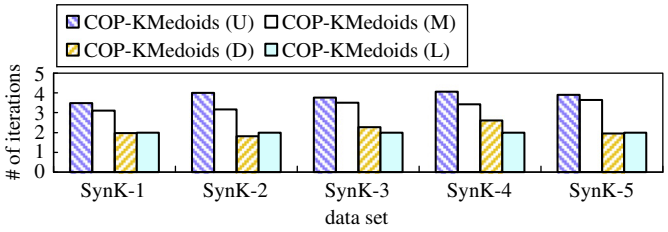

Fig. 6. Number of iterations of COP-KMedoids with different elite pairwise constraints for group1 data sets.

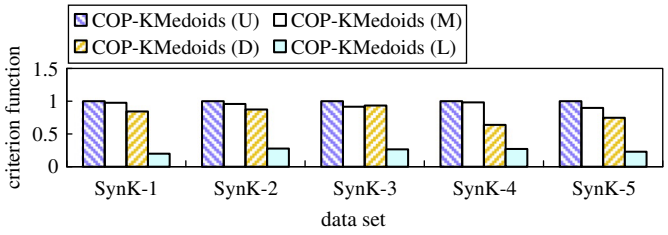

Fig. 7. Normalized criterion functions of COP-KMedoids with different elite pairwise constraints for group1 data sets.

We also compare the purity of every algorithm for Group1 data sets in Fig. 8. It can be observed that the purity of COP-KMedoids(U) is close to that of COP-KMedoids(M) and COP-KMedoids(D). Among COP-KMedoids(U), COP-KMedoids(M) and COP-KMedoids(D), no algorithm remarkably outperforms the other two algorithms. Out of all the four algorithms, COP-KMedoids(L) always achieves the highest purity, 100%, in Group1 data sets. In fact, as discussed in Section 6.2, the set of classes of each data set in Group1 is the unique k-partition. Meanwhile, COP-KMedoids(L) can achieve the unique k-partition for every data set in Group1. For example, the criterion function (7790.49) of the classes of SynK-1 equals to that of the average criterion function returned by COP-KMedoids(L).

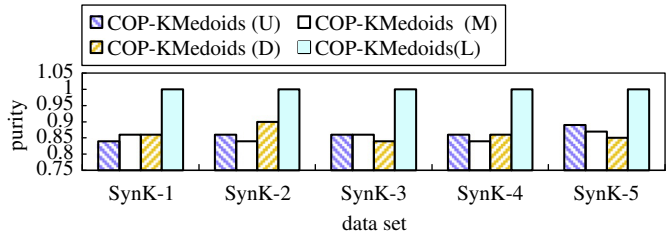

Fig. 8. Purity of COP-KMedoids with different elite pairwise constraints for group1 data sets.

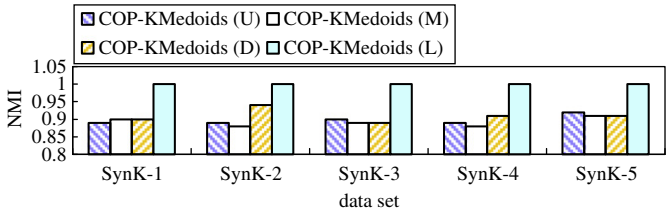

Fig. 9. Normalized mutual information of COP-KMedoids with different elite pairwise constraints for group1 data sets.

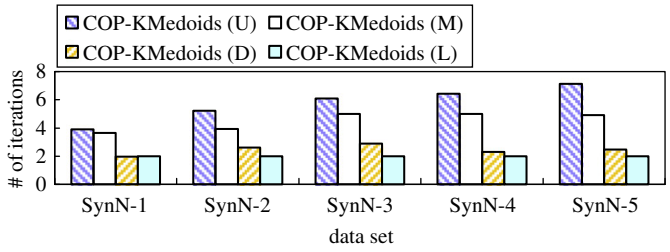

Fig. 10. Number of iterations of COP-KMedoids with different elite pairwise constraints for group2 data sets.

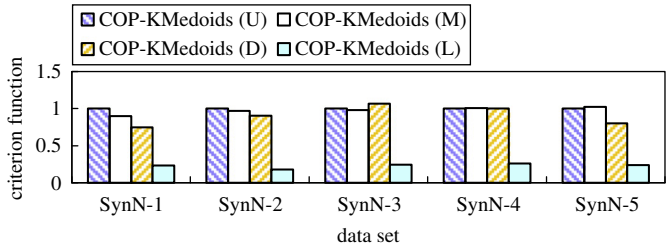

Fig. 11. Normalized criterion functions of COP-KMedoids with different elite pairwise constraints for group2 data sets.

Fig. 9 presents the comparison results of NMI values for those algorithms over Group1 data sets. It can be observed that similar distribution as Fig. 8 can be achieved. Out of all the four algorithms, COP-KMedoids(L) can always achieve the highest NMI value on every data set in Group1.

Fig. 10 shows the number of iterations of COP-KMedoids using different pairwise constraints for Group2 data sets. It can be found that COP-KMedoids(U) takes more iterations to converge than other algorithms. In contrast to COP-KMedoids(D) and COP-KMedoids(L), COP-KMedoids(M) always needs more iterations until converging. Obviously, COP-KMedoids(L) can easily converge on data sets in Group2. Similar to Fig. 7, the normalized criterion functions for Group2 data sets are given in Fig. 11. COP-KMedoids(U), COP-KMedoids(M), and COP-KMedoids(D) are close to each other on the criterion functions of SynN-2, SynN-3, and SynN-4. COP-KMedoids(D) achieves better criterion functions on SynN-1, SynN2, and SynN-5 than both COP-KMedoids(U) and

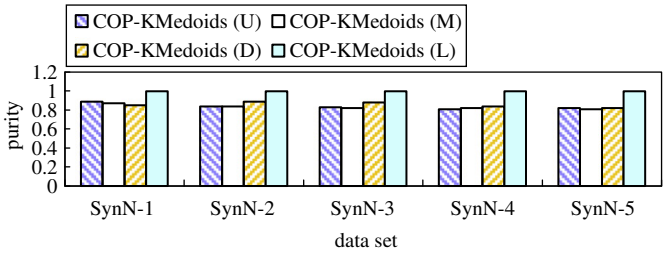

Fig. 12. Purity of COP-KMedoids with different elite pairwise constraints for group2 data sets.

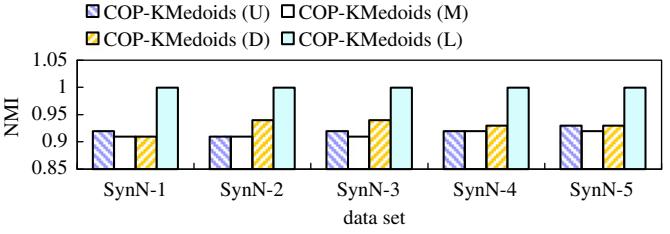

Fig. 13. Normalized mutual information of COP-KMedoids with different elite pairwise constraints for group2 data sets.

COP-KMedoids(M). COP-KMedoids(L) always obtains the best criterion functions among these four algorithms.

Similar to Group1 data sets, we present the purity and NMI of each algorithm for Group2 data sets in Figs. 12 and 13, respectively. As shown in Fig. 12, no clear difference among COP-KMedoids(U), COP-KMedoids(M), and COP-KMedoids(D) can be found in terms of the purity. Out of all the four algorithms, COP-KMedoids(L) can achieve the highest value of the purity for every data set in Group2. For Fig. 13, we can draw a similar conclusion as that for Group1 data sets. Among COP-KMedoids(U), COP-KMedoids(M) and COP-KMedoids(D), no one can remarkably outperforms the other two algorithms in terms of NMI. In contrast, COP-KMedoids(L) is the best one.

7. Conclusions

In this paper, we introduced elite pairwise constraints, including elite must-link (EML) and elite cannot-link (ECL) constraints, which are to be satisfied in every optimal k-partition. In contrast to traditional must-link and cannot-link constraints, these new constraints would not conflict with each other. Therefore, any set of EML and ECL constraints could always guide clustering algorithms towards feasible k-partitions. We have shown that it is intractable to retrieve EML or ECL constraints. Then, Limit Crossing was proposed to achieve a fraction of such new constraints. As a heuristic algorithm, Limit Crossing could always find some EML or ECL constraints in practice. Experimental results on both UCI and synthetic data sets have demonstrated the effectiveness of Limit Crossing in semi-supervised clustering.

There are several directions for future research. First, it will be interesting to find some efficient ways other than Lagrangian relaxation to calculate the lower bounds in Limit Crossing. In this paper, we had to calculate the lower bound for every pair of instances in order to determine an EML or ECL constraint. It will be very useful if a new method can simultaneously calculate multiple lower bounds. Second, we can investigate on how to use Limit Crossing to verify traditional pairwise constraints. Since EML and ECL constraints can be viewed as a subset of traditional pairwise constraints. We can possibly verify traditional pairwise constraints by using Limit Crossing. For example, if we can find an

EML constraint (by Limit Crossing) from a set of cannot-links, it can be claimed that those cannot-links are not reliable.

Acknowledgments

This work is partially supported by the US National Science Foundation (NSF) under Grant CCF-0905337, the Natural Science Foundation of China under Grant 61175062 and 61033012, and the Fundamental Research Funds for the Central Universities under Grant No. DUT12JR02.

References

[1] B. Everitt, S. Landau, M. Leese, Cluster Analysis, Arnold, London, 2001.
[2] A.K. Farahat, M. Kamel, Statistical semantics for enhancing document clustering, Knowl. Inf. Syst. 28 (2011) 365–393.
[3] K. Wagstaff, C. Cardie, Clustering with instance-level constraints, in: Proceedings of the 17th ICML, 2000, pp. 1103–1110.
[4] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained K-means clustering with background knowledge constraints, in: Proceedings of the 18th ICML, 2001, pp. 577–584.
[5] Y. Shi, L. Zhang, COID a cluster-outlier iterative detection approach to multi-dimensional data analysis, Knowl. Inf. Syst. 28 (2011) 709–733.
[6] S. Basu, A. Banerjee, R.J. Mooney, Active semi-supervision for pairwise constrained clustering, in: Proceedings of the 4th SDM, 2004, pp. 333–344.
[7] I. Davidson, S.S. Ravi, Clustering with constraints: feasibility issues and the k-means algorithm, in: Proceedings of the 5th SDM, 2005, pp. 138–149.
[8] C. Domeniconi, J. Peng, B.J. Yan, Composite kernels for semi-supervised clustering, Knowl. Inf. Syst. 28 (2011) 99–116.
[9] E.P. Xing, A.Y. Ng, M.I. Jordan, S. Russell, Distance Metric Learning with application to clustering with side-information, in: Proceedings of the 15th NIPS, 2002, pp. 505–512.
[10] D. Klein, S.D. Kamvar, C.D. Manning, From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. in: Proceedings of the 19th ICML, 2002, pp. 307–314.
[11] S. Basu, I. Davidson, K. Wagstaff, Clustering with Constraints: Algorithms, Applications, Chapman & Hall/CRC Press, 2008.
[12] D. Cohn, R. Caruana, A. McCallum, Semi-supervised clustering with user feedback, Technical Report, Cornell University, TR2003-1892, 2003.
[13] I. Davidson, S.S. Ravi, Identifying and generating easy sets of constraints for clustering, in: Proceedings of the 22th AAAI, 2006, pp. 336–341.
[14] I. Davidson, S.S. Ravi, Intractability and clustering with constraints, in: Proceedings of the 24th ICML, 2007, pp. 201–208.
[15] A. Freund, D. Pelleg, Y. Richter, Clustering from constraint graphs, in: Proceedings of the 8th SDM, 2008, pp. 301–311.
[16] K.Y. Yip, M.K. Ng, D.W. Cheung, Input validation for semi-supervised clustering, in: Proceedings of the 6th ICDM Work-shops, 2006.
[17] L. Kaufman, P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.
[18] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, Math. Program. 1–3 (1997) 191–215.
[19] T. Zhou, D.C. Tao, X.D. Wu, Manifold elastic net: a unified framework for sparse dimension reduction, Data Min. Knowl. Discov. 3 (2011) 340–371.
[20] R. Xu, D. Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Networks 3 (2005) 645–678.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[22] S. Climer, W. Zhang, Searching for backbones and fat: a limit crossing approach with applications, in: Proceedings of the 18th AAAI, 2002, pp. 707–712.
[23] Z.L. Ren, H. Jiang, J.F. Xuan, Z.X. Luo, An accelerated limit crossing based multilevel algorithm for the p-median problem, IEEE Trans. Syst. Man Cybern. Part B: Cybern. (2012), http://dx.doi.org/10.1109/TSMCB.2012.2188100.
[24] W. Bian, D.C. Tao, Max-min distance analysis by using sequential SDP relaxation for dimension reduction, IEEE Trans. Pattern Anal. Mach. Intell. 5 (2011) 1037–1050.
[25] L.F.E. Senne, A.N.L. Lorena, Lagrangian/Surrogate Heuristics for p-median Problem, Computing Tools for Modelling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, Kluwer Academic Publisher, Dordrecht, 2000 115–130.
[26] C.D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[27] B. Geng, D. CTao, C. Xu, DAML: domain adaptation metric learning, IEEE Trans. Image Process. 10 (2011) 2980–2989.
[28] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighbourhood components analysis, in: Proceedings of the 17th NIPS, 2004, pp. 513–520.
[29] Z.J. Zha, T. Mei, M. Wang, Z. Wang, X.S. Hua, Robust distance metric learning with auxiliary knowledge, in: Proceedings of the 21th IJCAI, 2009, pp. 1327–1332.
[30] W. Bian, D.C. Tao, Learning a distance metric by empirical loss minimization, in: Proceedings of the 22th IJCAI, 2011, pp. 1186–1191.

He Jiang is an associate professor in the School of Software, Dalian University of Technology. His research interests include computational intelligence and its applications in software engineering and data mining. Jiang received his Ph.D. in computer science from the University of Science and Technology of China (USTC). He is a program co-chair of the 2012 International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2012). He is also a member of the IEEE, the ACM, and the CCF.

Zhilei Ren is a Ph.D. candidate in the School of Software, Dalian University of Technology. His research interests include metaheuristic algorithm design, data mining, and their applications in software engineering. Ren received his B.Sc. in software engineering from Dalian University of Technology. He is a student member of the ACM and the CCF.

Jifeng Xuan is a Ph.D. candidate in the School of Software, Dalian University of Technology. His research interests include mining software repositories, human centred software engineering, and empirical studies. Xuan received his B.Sc. in software engineering from Dalian University of Technology. He is a student member of the ACM, the ACM Sigsoft, and the CCF.

Xindong Wu is a Professor of the Department of Computer Science at the University of Vermont (USA), a Yangtze River Scholar in the School of Computer Science and Information Engineering at the Hefei University of Technology (China), and a Fellow of the IEEE. He holds a Ph.D. in Artificial Intelligence from the University of Edinburgh, Britain. His research interests include data mining, knowledge-based systems, and Web information exploration.